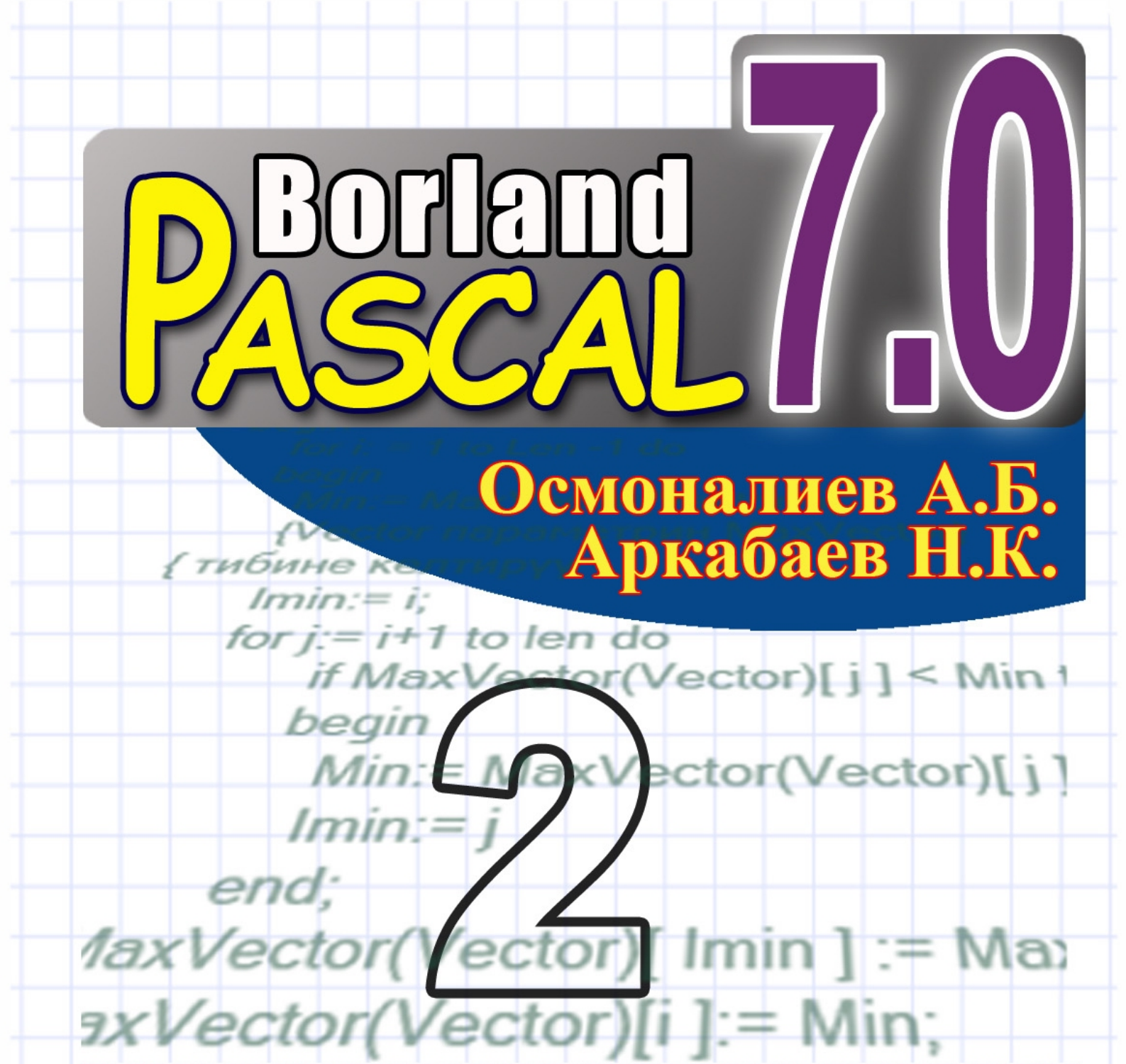
Borland PASCAL 7.0

Осмоналиев А.Б.
Аркабаев Н.К.



2

ПРОГРАММАЛООНУН НЕГИЗДЕРИ

Ош - 2010

МАЗМУНУ

КИРИШҮҮ

Дүйнөнү күч башкарбай акыл башкарат, бирок акыл күчкө таянат.
(Б. Паскаль)

Урматтуу окурман! Бул колуңуздагы китеп 2008-жылы жарык көргөн «Borland Pascal 7.0 Программалоонун негиздери» - окуу китебинин 2-бөлүгү болуп саналат. Сиздер байкагандай биринчи бөлүктө Pascal тили жөнүндө, алгоритмдердин жалпыланган конструкциялары жана алардын Borland Pascal тилинин чөйрөсүндөгү реализацияланыш каражаттары жөнүндө сөз болуп, статикалык структурадагы берилгендер менен иштөөгө басым жасалган эле.

Экинчи бөлүктө динамикалык структурадагы берилгендер менен иштөө үчүн арналган Borland Pascal тилинин каражаттары, чоң программалык комплекстерди талдап иштеп чыгуу методологиясы, анын ичинде структуралык программалоонун максаты, принциптери жана модулдук программалоо маселелери каралды.

Андан сырткары IBM PC нин түзүлүштөрү менен иштөөдөгү стандарттык ыкмалар, атап айтканда, клавиатура, курсор жана үн менен иштөө, тексттик видеорежимде иштөө, графиктик видеорежимде иштөө боюнча маалыматтар киргизилди. Дагы бир белгилеп өткөнгө арзый турган нерсе – программаларды талдап иштеп чыгуунун объекттик-ориентирленген методологиясы, Borland Pascal тилинин объекттик-ориентирленген каражаттары, объекттер жана модулдар менен иштөө, мурастоо, усулдар, виртуалдык жана динамикалык усулдар, конструкторлор жана деструкторлор сыяктуу ООП методологиясынын негизги түшүнүктөрү боюнча маалыматтардын берилиши болуп эсептелет. Бул келтирилген маалыматтар азыркы башка объекттик-ориентирленген программалоо тилдерин (Java, Delphi, C++ ж.б.) окуп үйрөнүү үчүн абдан жакшы даярдык боло алат деп эсептейбиз.

Ошондой эле китептин бул бөлүгүндө Borland Pascal тилинин стандарттык процедуралары жана модулдарынын тизмеги, алардын пайдалануу форматтары, эрежелери жана түшүндүрмөлөрү менен кеңири берилди. Көпчүлүк каралган тилдик каражаттар үчүн атайын тандалган мисалдар келтирилди.

Бул окуу китепти алгоритмдештирүү жана жалпы эле программалоо назарияты боюнча окуу курал катары ошол эле учурда Borland Pascal тилинин айрым каражаттары боюнча сурап-билме (справочниги) катары пайдаланууга болот. Алыбетте, бул китеп эң

оболу жогорку окуу жайларда компъютердик техника жаатында профессионалдык билим алып алып жаткан студенттер үчүн окуу китеп катары арналып жазылды. Бирок, орто мектептерде информатика боюнча атайын курстарды жана факультативдик сабактарды өтүүдө да кеңири пайдаланууга болот. Андан сырткары дегеле Borland Pascal тилинин чөйрөсүндө программалоону үйрөнүүнү каалаган жалпы эле пайдалануучулар үчүн зор пайдасын тийгизе алат деп ишенебиз.

Бул китептин жарыкка чыгышына салымдарын кошкон айрым коллегаларыбызга, сын пикирлерин айткан физика-математика илимдеринин доктору, профессор А.Сопуевге, физика-математика илимдеринин кандидаты, доцент А.Кыбыраевге чоң ыраазычылыктарыбызды билдирүүнү өзүбүздүн сыймыктуу милдетибиз деп эсептейбиз.

Китеп боюнча окурмандардын суроолору, сын-пикирлери жана каалоолору чоң ыраазычылык менен төмөнкү даректер боюнча кабыл алынат: okamil1@rambler.ru, nurkasym@gmail.com.

Авторлор

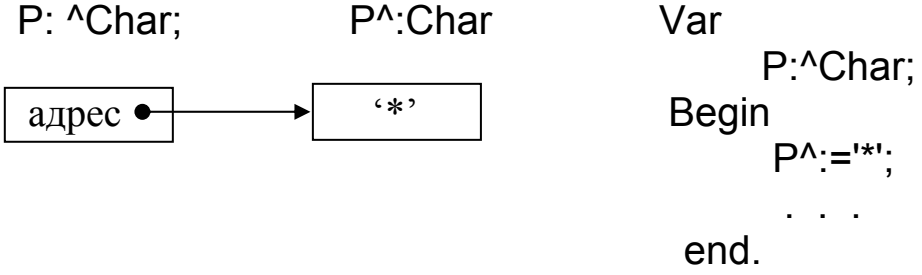
1. ДИНАМИКАЛЫК СТРУКТУРАДАГЫ БЕРИЛГЕНДЕР МЕНЕН ИШТӨӨ

Өзгөнүн кеңешин ук,
бирок өзүңдүн пикириңден тайба.
(В. Шекспир)

Берилгендердин, ошонун ичинде динамикалык структурадагы берилгендердин классификациясы окуу китептин 1-бөлүгүндө келтирилген

1.1. Эстин бөлүштүрүлүшү

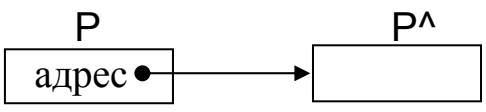
Динамикалык структурадагы берилгендерди жарыялоо, баяндоо бөлүгүндө кандайдыр бир типтеги өзгөрүлмөнүн өзү эмес ага болгон көрсөткүч (шилтеме) көрсөтүлөт. Натыйжада көрсөткүч кадимкидей өзгөрүлмө болуп, ал эми ал көрсөткүч көрсөтүп турган өзгөрүлмө динамикалык болуп калат.



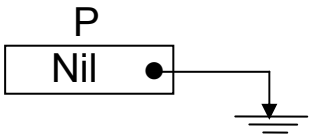
Программада көрсөткүчтүн идентификаторун пайдалануу ал көрсөткүч көрсөтүп турган эстин уячасынын(ячейкасынын) адресине кайрылууну билдирет. Көрсөткүч көрсөтүп турган уячанын мазмунуна кайрылуу үчүн анын идентификаторунан кийин @ символун коюу талап кылынат. Бул амал – атсыздаштыруу (разыменование) амалы деп аталат. Динамикалык өзгөрүлмөлөр үчүн эсти бөлүү жана бошотуу New, Dispose, GetMen, FreeMen, Mark, Release процедураларынын жардамы менен программа иштеп жаткан учурда аткарылат.

P көрсөткүчтүк өзгөрүлмөсү үч абалда болушу мүмкүн:

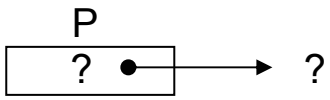
1) Эбак эсте орун бөлүнүп коюлган кандайдыр бир өзгөрүлмөнүн адресин кармап турган болушу мүмкүн.



2) Атайын nil бош адресин кармап турган болушу мүмкүн.

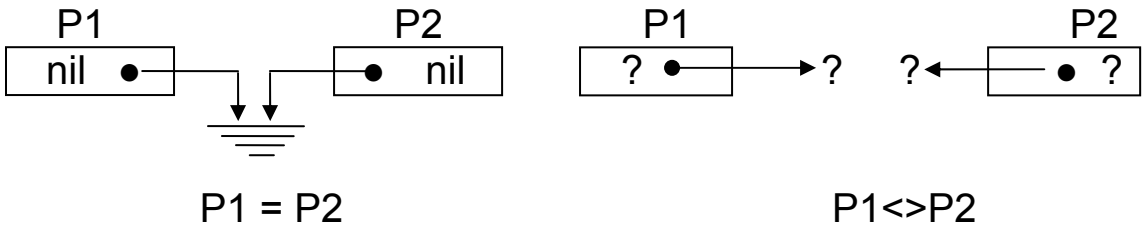


3) Аныкталбаган абалда болушу мүмкүн.

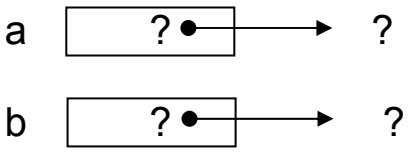
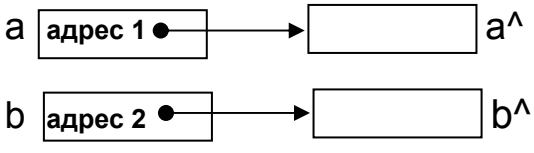


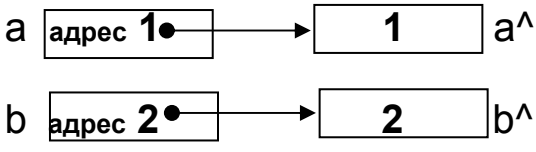
Көрсөткүч мындай аныкталбаган абалда программанын ишинин башында ага биринчи ирет конкреттү адрести же nil бош адресин ыйгарганга чейин, ошондой эле ал көрсөтүп турган эстин областы бошогондон кийин болот.

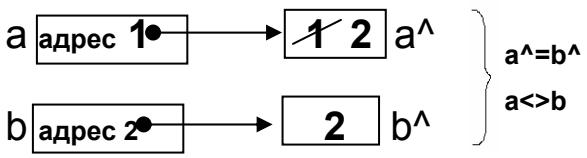
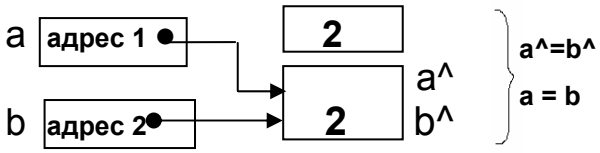
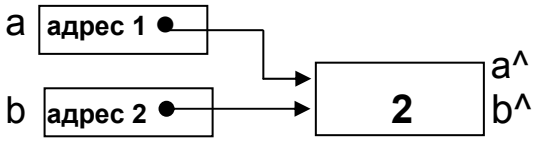
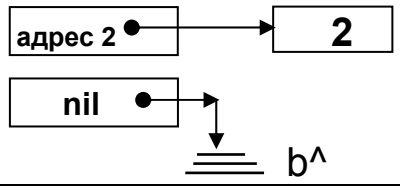
Nil абалы менен аныкталбаган абалдардын ортосундагы айырманы төмөнкү схемадан көрүгө болот:



1.2. Көрсөткүчтөр менен болгон жөнөкөй аракеттер

Аракет	Жыйынтык
1. Жарыялоо type PInt=^integer; var a,b: PInt;	
2. Эсти бөлүп алуу New(a); New(b);	

3. Информацияны киргизүү
a^: = 1;
b^: = 2;
	

4. Информацияны көчүрүү a^:=b^;	
5. а) Адрести көчүрүү a:=b;	
5б) Dispose(a); a:=b;	
6) b:=nil;	

New(a) процедурасы А көрсөткүчү үчүн баяндалган типке тиешелүү түрдө эстин областын бөлөт жана бөлүнгөн эстин адресин көрсөткүчкө жазат.

Dispose (A) процедурасы А көрсөткүчү көрсөтүп турган эстин областын бошотот, ошондон кийин эстин бул областын башка динамикалык өзгөрүлмөлөр үчүн бөлүштүрүүгө мүмкүн болуп калат.

1.3. Байланылбаган динамикалык берилгендер

Байланылбаган динамикалык берилгендер статикалык берилгендердин өзүндөй классификацияланышат жана алар менен иштөө ошого эле окшош аткарылат. Байланылбаган динамикалык берилгендердин динамикалык касиеттерин туюнткан бир гана нерсе –алар программа иштеп жаткан учурда «пайда болуп» жана «жок болуп» тура алышат. Мындай берилгендерди пайдалануунун айырмачылыктары эки аспекттен турат:

- var бөлүгүндө талап кылынган типтеги өзгөрүлмө эмес, ушул типке болгон көрсөткүч жарыяланат;
- пайдалануунун алдында New процедурасын, ал эми пайдалангандан кийин Dispose процедурасын чакыруу зарыл.

Мисал катары окшош статикалык жана байланылбаган динамикалык берилгендердин иштерин салыштыруу таблицасын келтиребиз.

Берилгендердин структурасы	Кадимки өзгөрүлмөлөр	Динамикалык өзгөрүлмөлөр

1. Жөнөкөй өзгөрүлмө	type Tint = 1..100;
	var X: Char;
	Y: Tint;
	Begin
	X:='*';
	Y:= 3;
	. . .
	end.
		type Tint = 1..100;
		var PX: ^Char;
		PY: ^Tint;
		Begin
		New(PX);
		New(PY);
		PX^ :='*';
		PY^ := 3;
		. . .
		Dispose(PX);
		Dispose(PY);
		end.

2. Массив	type
	Vect = array[1..3]
	of Byte;
	var X: Vect;
	i: Byte;
	Begin
	for i:= 1 to 3 do
	Read(X[i]);
	. . .
	end.
		type
		Vect = array[1..3]
		of Byte;
		var PX: ^Vect;
		i: Byte;
		Begin
		New(PX);
		for i:= 1 to 3 do
		Read(PX^[i]);
		. . .
		Dispose(PX);
		end.

3. Жазуу	type
	Rec = record
	A: Char;
	B: Byte;
	end;
	var
	X : Rec;
	Begin
	X.A := '*';
	X.B := 7;
	. . .
	end.
		type
		Rec = record
		A: Char;
		B: Byte;
		end;
		var
		X : ^Rec;
		Begin
		New(PX);
		X.A := '*';
		X.B := 7;
		. . .
		Dispose(PX);
		end.

1.4. Байланылган динамикалык берилгендер

1.4.1. Негизги аныктоолор

Сызыктуу тизмелер – каалаган эки элементинин ортосуна элементтерди кошууга жана каалаган элементти алып салууга мүмкүн болгон сызыктуу байланылган бир тектүү элементтердин жыйындысы болуп эсептелет.

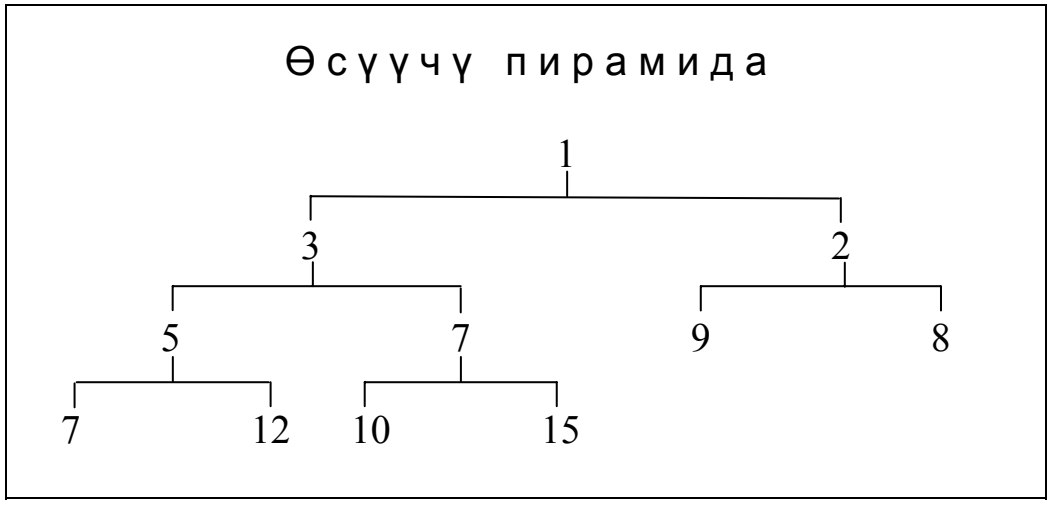
Алкактык тизмелер – бул сызыктуу тизмелерге эле окшогон элементтер, бирок тизменин акыркы жана биринчи элементтеринин ортосундагы кошумча байланышка ээ.

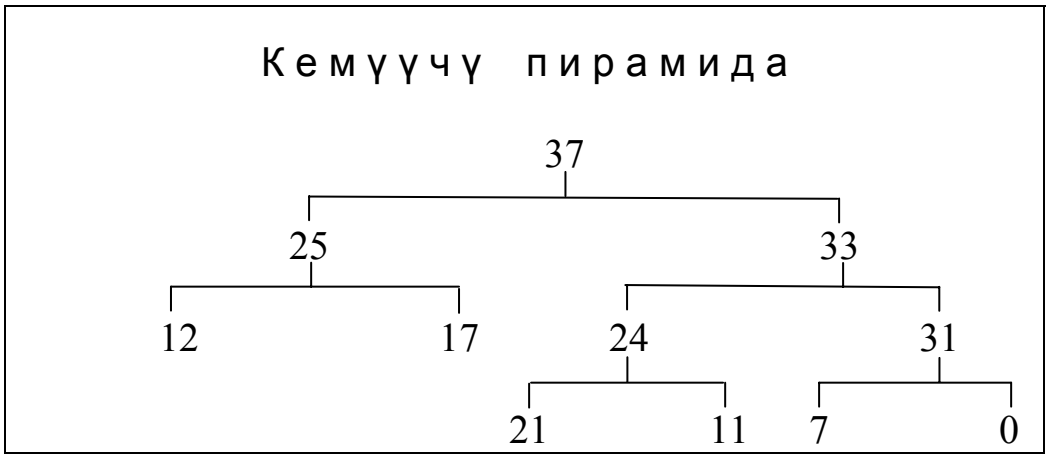
Кезек – сызыктуу бир байламталуу тизменин жекече учуру болуп, анда эки гана аракетти аткарууга уруксат: кезектин ***акырына*** (***куйругуна***) жаңы элементти кошууга жана кезектин ***башталышынан*** (***башынан***) элементти жоготууга (алып салууга) болот.

Стек – сызыктуу бир байламталуу тизменин жекече учуру болуп, анда ***стектин чокусу*** (***башы***) деп аталган кезектин аяк жагынан гана элементтерди кошууга жана алып салууга урусат берилет.

Даректар – бул каалагандай конфигурациядагы иерархиялык структурадагы динамикалык берилгендер. Даректын элементтери ***чокулар*** (***түйүндөр***) деп аталат.

Пирамида (***иреттелген дарак***) деп чокуларынын (түйүндөрүнүн) маанилери кийинки деңгээлге өткөндө дайыма өсүп же кемип туруучу даракты аташат.





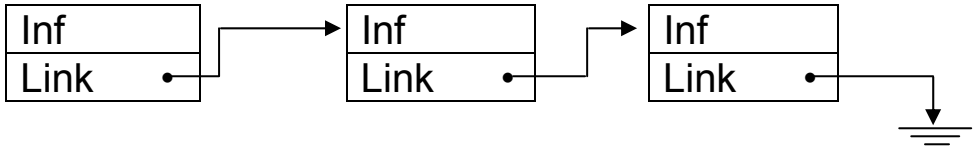
1.4.2. Байланылган динамикалык берилгендерде өз ара байланыштарды уюштуруу

Ар түрдүү конфигурациядагы берилгендерди түзүүдө байланган динамикалык берилгендер жогорку ийкемдүүлүгү менен мүнөздөлөт. Бул болсо программанын ишинин каалаган моментинде элементтер үчүн эсти бөлүп алууга же бошотууга жана көрсөткүчтөрдүн жардамында каалагандай эки элементтин ортосундагы байланышты тургузууга мүмкүн болушунан улам жетишилет.

Динамикалык структурадагы берилгендердин элементтеринин ортосундагы байланыштарды уюштуруу үчүн, ар бир элемент информациялык маанилерден башка жок дегенде бир көрсөткүчтү кармап турушу талап кылынат. Мындан мындай структурадагы элементтер катары ар түрдүү тектүү элементтерди бир бүтүнгө бириктире ала турган жазууларды пайдалануу зарылдыгы келип чыгат.

Жөнөкөй учурда динамикалык структурадагы берилгендердин элементи эки талаадан турушу керек: ***информациялык*** *жана* ***көрсөткүчтүк****.*

Берилгендердин мындай структурасын схемалык түрдө мындай көрсөтүүгө болот:



Буга тиешелүү баяндоо төмөнкү көрүнүштө болот:

```
Type
      TPtr  =  ^ TElem;
      TElem  =  record
            Inf   :   Real;
            Link  :   TPtr
      end;
```

Turbo Pascal тилинде баяндоолордун удаалаштыгын жазуу эрежеси ар бир идентификатордун аны башка баяндоолордо пайдаланууудан мурда баяндалышын талап кылат. Бирок жогорку келтирилген мисалда TPtr көрсөткүчүнүн жана TElem элементинин типтерин баяндоо кандай гана жайланышпасын бул эреже аткарылбайт. Ошондуктан, *динамикалык структурадагы берилгендердин элементтеринин типтерин баяндоо үчүн бул эрежеден четтөө жасалган*.

Динамикалык структурадагы берилгендердин элементине болгон көрсөткүчтүн тиби ущул элементтин тибин баяндоонун алдында баяндалган болушу мүмкүн жана баяндалышы керек.

1.4.3. Кезектер менен иштөө

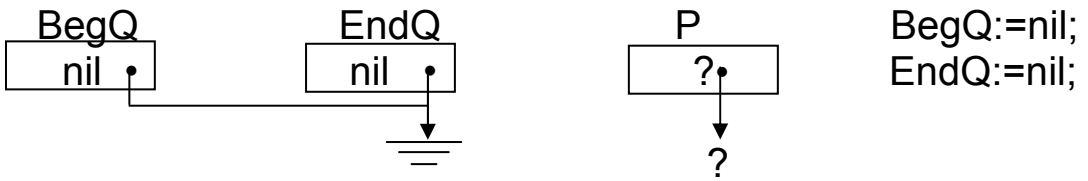
Кезекти түзүү жана аны менен иштөө үчүн жок дегенде эки көрсөткүчтүн болушу зарыл:

- кезектин башына болгон (BegQ идентификаторун алабыз);
- кезектин аягына болгон (EndQ идентификаторун алабыз).

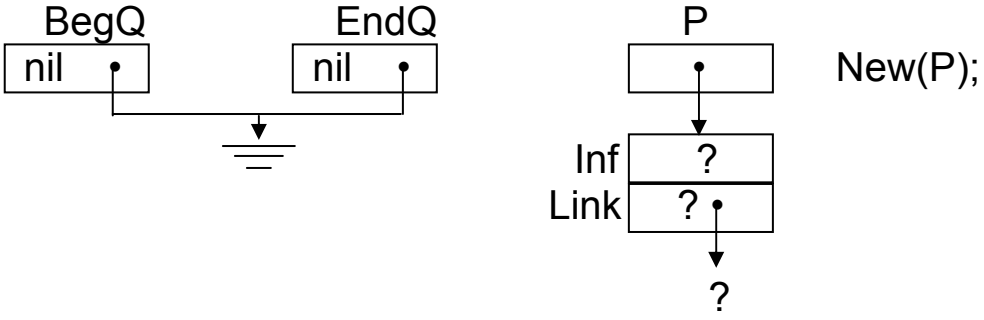
Андан сырткары, өчүрүлүүчү (жоготулуучу) элементтерге берилген эсти бошотуу үчүн кошумча убактылуу көрсөткүч (P идентификаторун алабыз) талап кылынат. Мындай кошумча көрсөткүч ошондой эле башка ситуацияларда да кезектер менен иштөө ыёгайлуу болсун үчүн пайдаланылат.

Кезекти түзүү

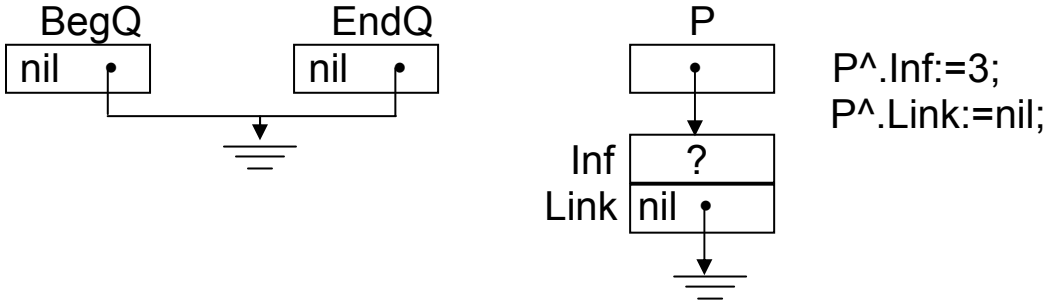
1. Баштапкы абал:



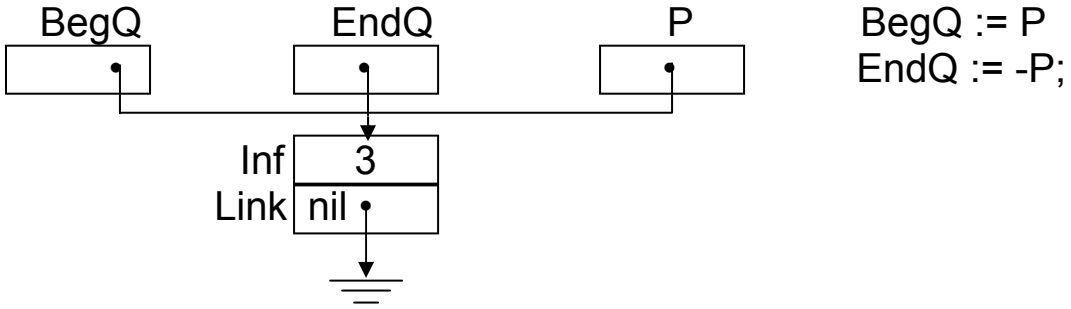
2. Кезектин биринчи элементи үчүн эсти бөлүү:



3.Кезектин биринчи элементине информацияны киргизүү:

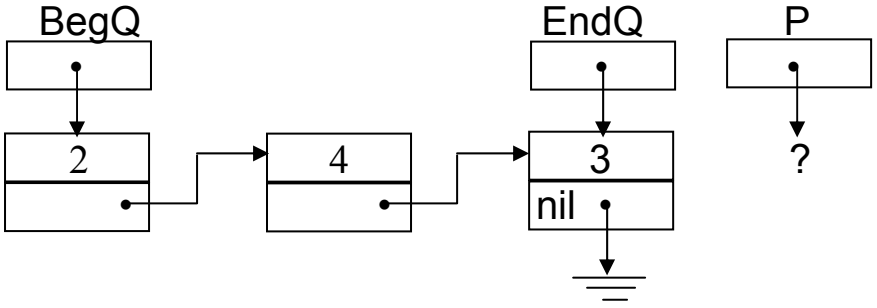


4. BenQ жана EndQ көрсөткүчтөрүн түзүлгөн биринчи элементке коюу:

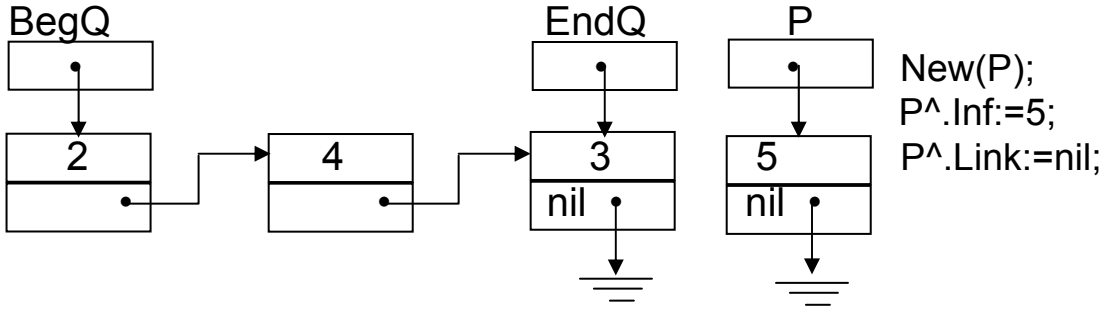


Кезекке элементти кошуу

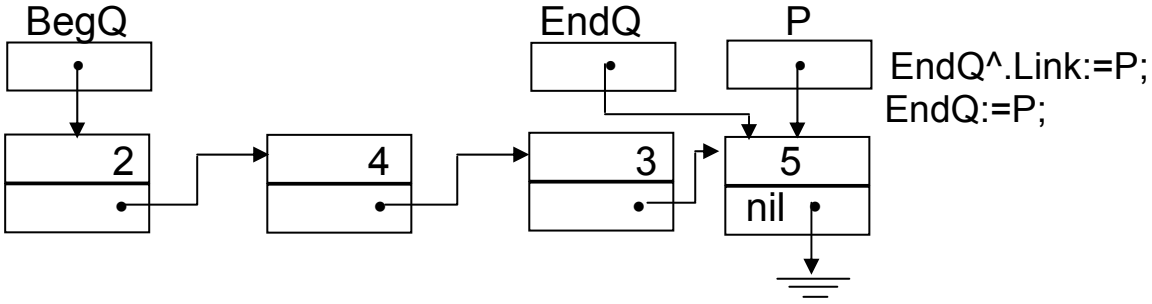
1. Баштапкы абал:



2. Жаңы элементке эсти бөлүү жана ага информацияны киргизүү:

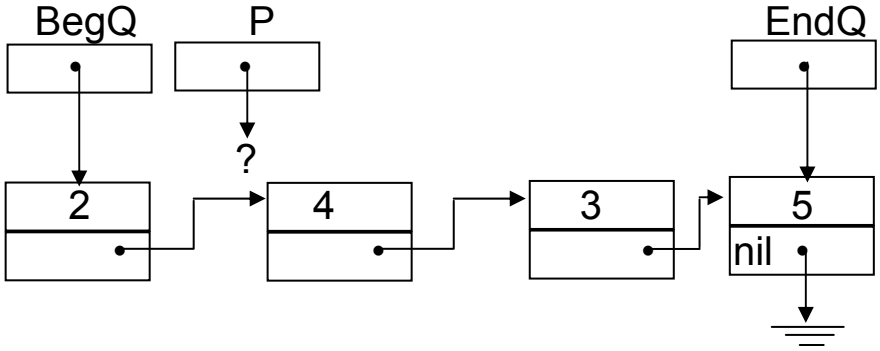


3. Кезектин акыркы жана жаңы элементтеринин ортосундагы байланышты түзүү, ошондой эле кезектин бүтүшүнө болгон EndQ көрсөткүчүн жаңы элементке жылдыруу:

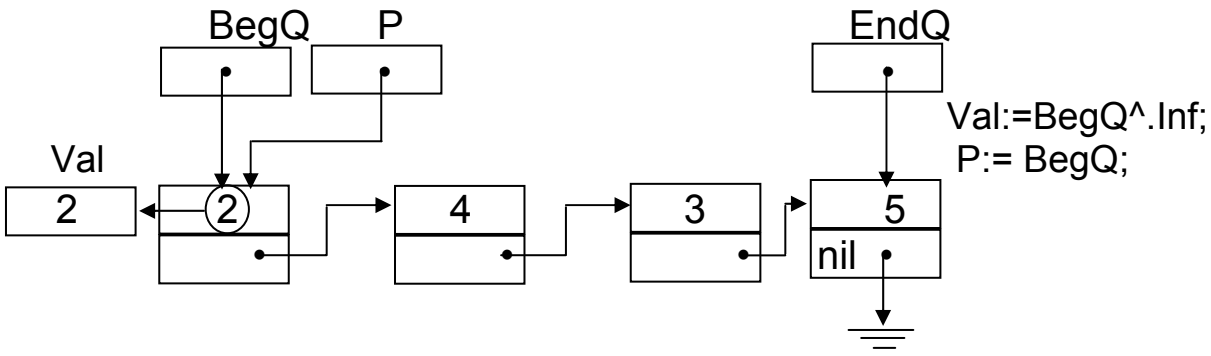


Кезектин элементин өчүрүү (жоготуу)

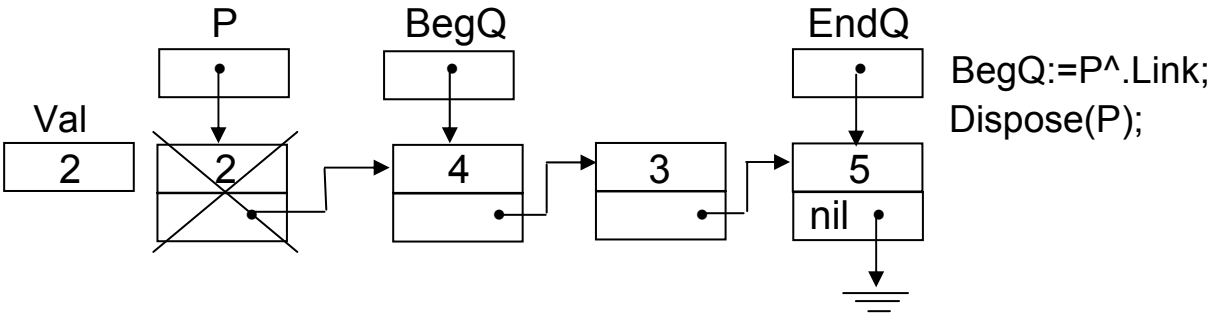
1. Баштапкы абал:



2. Өчүрүрүлүүчү элементтен информацияны Val өзгөрүлмөсүнө чыгарып алуу жана ага P жардамчы көрсөткүчүн коюу:



3. Биринчи элементте сакталып турган Link талаасынын маанисин пайдалануу менен кезектин башын көрсөтүп турган BenQ көрсөткүчүн кийинки элементке жылдырып коюу. Мындан кийин P кошумча көрсөткүчүн пайдалануу менен кезектин башталыш элементи үчүн берилген эс бошотулат.



Мисал катары он элементтен турган кезекти түзүү жана өчүрүү программасын карайбыз. Мында кезектин үстүнөн иштөө үчүн эки процедура пайдаланылат:

1) AddEl процедурасы, ал кезектин абалынан көз каранды түрдө биринчи элементти түзөт же кезектин акырына жаңы элементти кошот.

2) GetDelEl процедурасы, ал кезектин башындагы элементтен ага бөлүнгөн эсти тазалоо менен андагы информацияны алып чыгат.

```
Program Queue;
  uses Crt;
type
     TPtr = ^TElem;
      TElem=record
        Inf:    Real;
        Link:  TPtr
        end;
Var
    BegQ, EndQ: TPtr;
    Value       : Real;
     i             : Byte;
Procedure  AddEl (val: Real);
Var
  P: TPtr;
begin
New(p);
       P^.Inf := val;
       P^.Link := nil;
       if EndQ = nil {эгерде кезектин биринчи элементи түзүлсө}
```

```
            then BegQ := P
                    {эгерде кезектин кезектеги элементи түзүлсө}
        else EndQ^.link := P;
        EndQ := P
end;
Procedure GetDelEl( var Val: Real);
var
   P: TPtr ;
begin
  Val :=BegQ^.Inf;
   P := BegQ;
   BegQ := P^.Link;
If BegQ=nil {эгерде кезектин акыркы элементи өчүрүлсө}
     then EndQ:=nil;
    Dispose(P)
end;
begin
   ClrScr;
         {Көрсөткүчтөрдүн баштапкы коюлушу}
     BegQ := nil;
     EndQ := nil;
          {Он элементтен турган кезекти түзүү}
 for i:=1  to  10  do  AddEl(i);
  {Элементтеринин маанилерин печатка чыгаруу}
                { менен кезекти өчүрүү}
 while BegQ <> nil do
  begin
   GetDelEl (Value);
    Writeln ('Value=' , Value :5 :2)
  end
end.
```

Бул программанын ишинин жыйынтыгы төмөндөгүдөй болот:

```
Value = 1.00
Value = 2.00
Value = 3.00
Value = 4.00
Value = 5.00
Value = 6.00
Value = 7.00
```

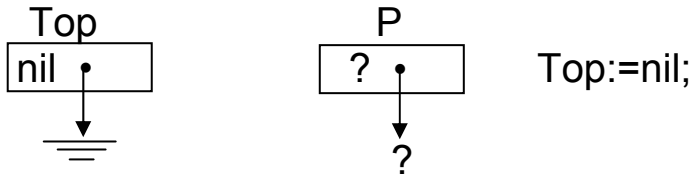
```
Value = 8.00
Value = 9.00
Value = 10.00
```

1.4.4. Стек менен иштөө

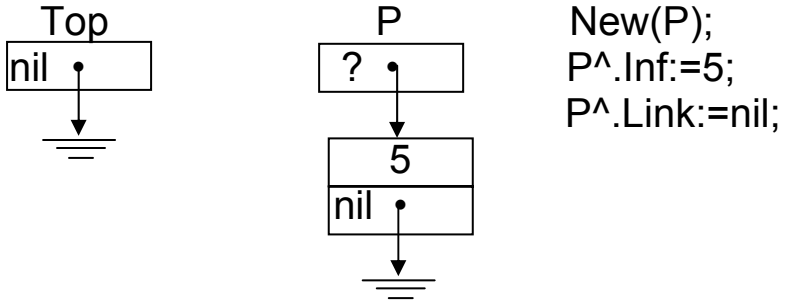
Кезектен айырмаланып стек менен иштөө үчүн стектин чокусун көрсөтүп туруучу көрсөткүчтүн (Top идентификаторун алабыз) жана стектин элементтери үчүн эсти бөлүүдө жана бошотууда пайдаланылуучу бир кошумча убактылуу көрсөткүчтүн (P идентификаторун алабыз) болушу зарыл.

Стекти түзүү

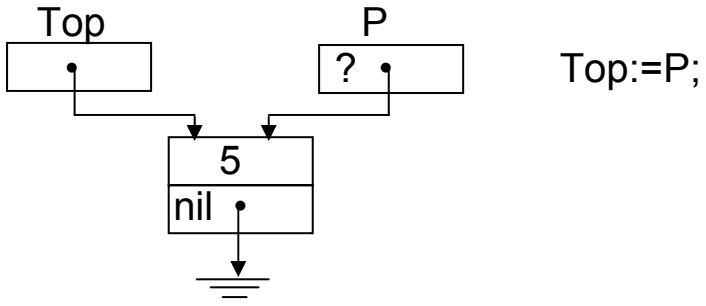
1. Баштапкы абал:



2. Стектин биринчи элементи үчүн эсти бөлүү жана ага информацияны киргизүү:

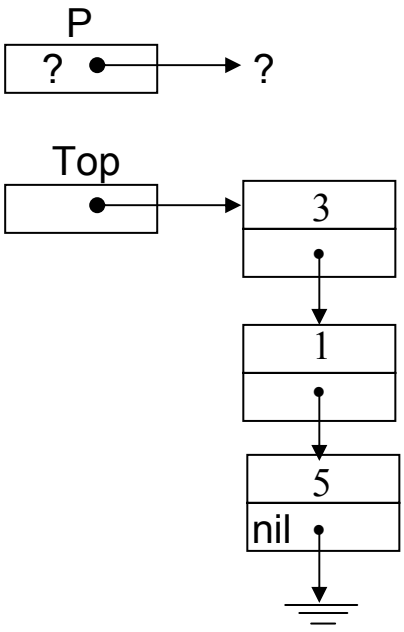


3. Стектин Top чокусун түзүлгөн элементке коюу

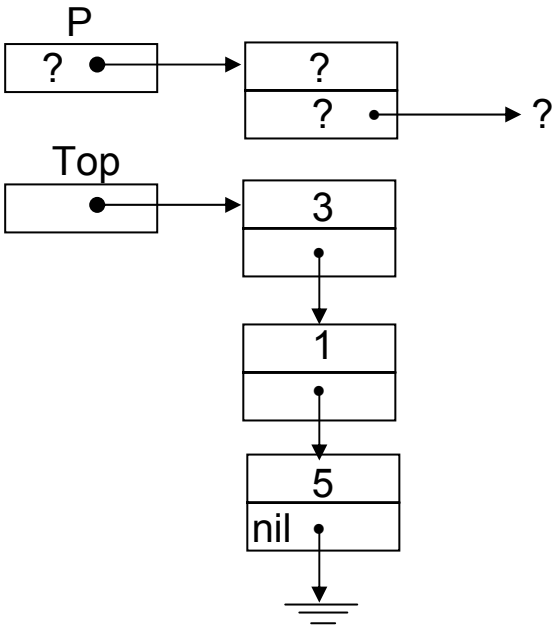


Стекке элементти кошуу

1. Баштапкы абал:

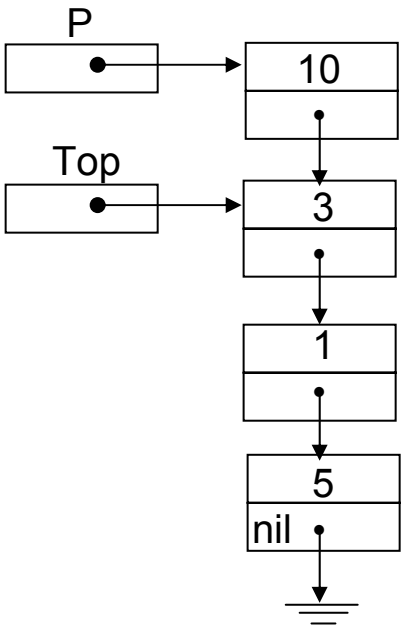


2. Стектин жаңы элементи үчүн эсти бөлүү:



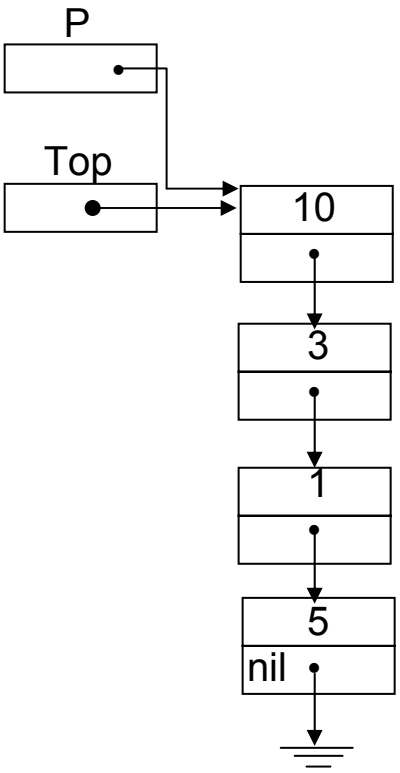
New(P);

3. Жаңы элементтин информациялык талаасына маанини кайрүү жана аны менен стектин Top «эски» чокусунун ортосундагы байланышты тургузуу.



P^.Inf:=Val; {Val=10}
P^.Link:= Top;

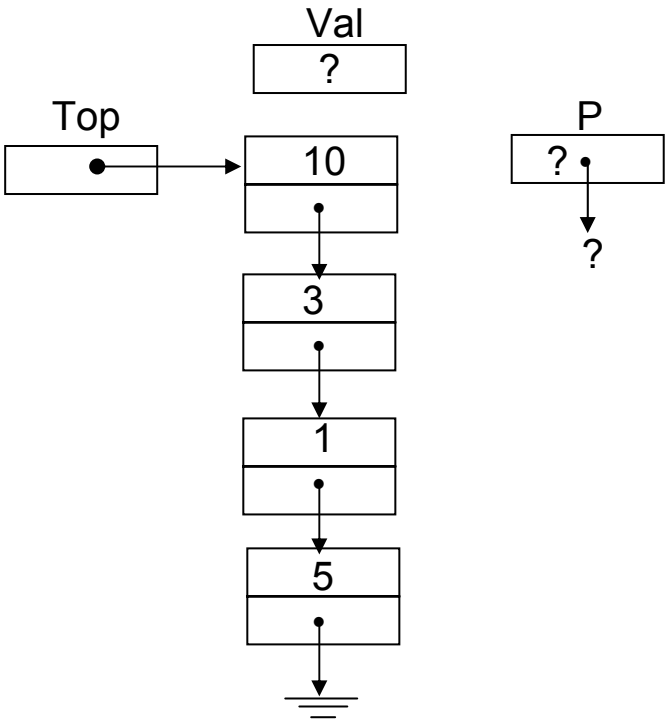
4.Стектин Top чокусун жаңы элементке жылдыруу:



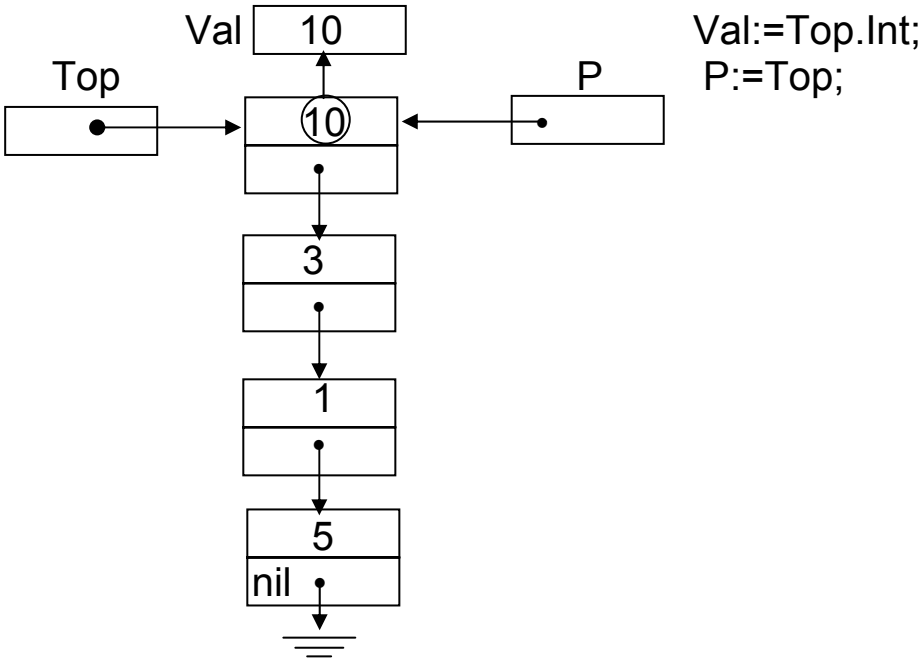
Top:= P;

Стектен элементти өчүрүү

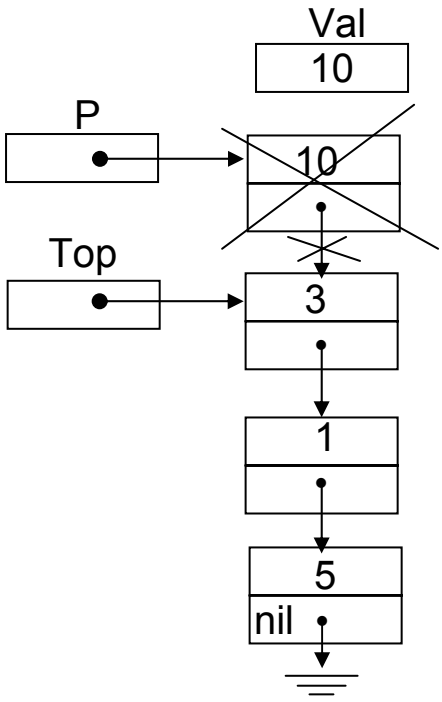
1. Баштапкы абал:



2. Стектин Top чокусунун информациялык талаасынан информацияны Val өзгөрүлмөсүнө алып чыгып берүү жана P жардамчы көрсөткүчүн стектин чокусуна коюу.



3. Стектин Top чокусуна болгон көрсөткүчтү кийинки элементке жылдыруу жана стектин «эски» чокусу ээлеп тарган эсти бошот:



Top:=P^.Link;
Dispose(P);

Мисал катары он элементтен турган стекти түзүү жана өчүрүү программасын карайбыз. Мында стек менен иштөө үчүн эки процедура пайдаланылган:

1) Push процедурасы, ал стектин абалынан көз каранды түрдө жаңы элементти түзөт же стектин чокусуна кезектеги жаңы элементти кошот;

2) Pop процедурасы, ал стектин чокусунан информацияны алып чыгат жана ага бөлүнгөн эсти бошотот.

```
program  Stack;
  uses Crt;
type
     TPtr = ^TElem;
      TElem=record
        Inf:    Real;
        Link:  TPtr
        end;
Var
    Top: TPtr;
    Value       : Real;
     i             : Byte;
Procedure  Push(val: Real);
Var
 P: TPtr;
```

```
begin
New(p);
        P^.Inf := val;
        P^.Link := Top;
       Top := P
end;
Procedure Pop( var Val: Real);
var
   P: TPtr ;
begin
  Val :=Top^.Inf;
   P := Top;
   Top := P^.Link;
    Dispose(P)
end;
begin
   ClrScr;
          {Көрсөткүчтөрдүн баштапкы коюулушу}
     Top := nil;
           {Он элементтен турган стекти түзүү}
 for i:=1  to  10  do  Push(i);
  {Элементтеринин маанилерин печатка чыгаруу}
                { менен стекти өчүрүү}
 while Top <> nil do
  begin
       Pop(Value);
       Writeln ('Value=' , Value :5 :2)
   end
end.
```

Программанын ишинин жыйынтыгы төмөндөгүдөй болот:

```
Value = 10.00
Value = 9.00
Value = 8.00
Value = 7.00
Value = 6.00
Value = 5.00
Value = 4.00
Value = 3.00
Value = 2.00
Value = 1.00
```

Белгилеп кетчү нерсе, кезек үчүн AddEl жана GetDelEl процедураларынан айырмаланып Push жана Pop процедураларында стектин биринчи элементин түзүү жана акыркы элементин өчүрүү учурларын өзгөчө бөлүп көрсөтүү талап кылынган жок, анткени бул учурларга Top = nil баштапкы маани учурунда стекке элементти кошуу жана кезектеги элементти өчүрүү учурундагыдай эле операторлордун удаалаштыгы тиешелеш келет.

2. ПРОГРАММАЛАРДЫ ТАЛДООНУН СТРУКТУРАЛЫК УСУЛИЯТЫ

Адамдын тили – адамдын оюн толук бере билгени менен, ички терең сезимди берүү жагынан абдан начар котормочу.

(Кошут)

2.1. Тарыхый маалыматтар

Өткөн кылымдын 70 - жылдарынын башында эле ийгиликтүү долбоорлоо, коддоо жана ондоп-түзөө үчүн арналган программалык долбоорлор өтө эле татаал болуп кеткендиги көрүнүп калган эле. Андан сырткары, татаал маселелерди чечүүдө программистер мындай бир проблемага туш болушкан: программадагы жолчолордун саны жана программанын көлөмү ушунчалык чонойуп кетип андан ары талдоо процессин дээрлик башкаруу мүмкүн болбой калган жана талдоочулардын эч бири түзүлгөн программалык продукт дайыма талап кылынган гана нерсени аткарат, а талап кылынбаган нерсени аткарбайт-деп ишенимдүү айта алган эмес. Ошентип чоң программалык комплекстерди түзүүгө карата болгон ыкманы түп-тамырынан бери өзгөртүү проблемасы келип чыккан.

Ушул проблемадан улам ошол учурдун эң алдынкы программисттери (Дейкстра, Вирт, Дал, Хоар, Йордан, Константин, Майерс ж.б.) тарабынан долбоорлорду түзүүнүн ***структуралык усулияты*** деп аталган катуу эреже иштелип чыгылган.

Жаңы усулияттардын пайда болушундагы башкы этап болуп 1968-1969-жылдардагы программалоо боюнча өткөн эл аралык конференциялар болду. Ушул конференциялардын экинчисинде Эдсгер Дейкстра (Dijkstra) биринчи жолу «структуралык программалоо» деген терминди колдоонуу менен пргограммаларды түзүүнүн принципиалдуу жаңы жолун сунуш кылган. Ал программаны так-даана структуралаштырууга мүмкүнчүлүк берүүчү иерархиялык абстракттык деңгээлдердин жыйындысы катары караган. Бул болсо программанын программисттер үчүн түшүнүктүүлүгүн жогорулатуу менен анын корректүүлүгүн далилдөөгө мүмкүнчүлүк берген жана ошону менен программалардын ишенимдүү иштешин жогорулатып ал программаны талдап чыгуу мөөнөтүн кыскарткан.

Программисттердин ой жүгүртүүсүнүн өзгөрүшүнө дагы бир түрткү болгон нерсе ошол кезде жарыяланган Дейкстранын илимий журналдардын биринин редакторуна жазган «Goto операторун

зыяндуу деп эсептөө керек» деп башталган каты болгон. Бул кат ошол кездеги программисттердин арасында курч талаш-тартыштарды пайда кылган. Бирок жыйынтыгында Дейкестрадан башка Цюрих техникалык университетинин профессору Вирт (Wirth) жана Оксфорд университетинин профессору Хоар да колдогон структуралык ой жүгүртүү идеясы баары бир жеңип чыккан.

Талаш-тартыштын жыйынтыктарынын бири - бул ар кандай программаны операторлордун жөнөкөй удаалаштыгын, while тибиндеги оперативдик конструкцияны жана case тандоо конструкциясын гана пайдаланып түзүүгө мүмкүн экендигинин, ал эми goto оператору структуралык программалоодо зарыл болгон башкаруучу конструкция эмес экендигинин далилдениши болгон. Тилекке каршы goto оператору боюнча талаш-тартыштардын бир жаман жагы – goto операторун пайдаланбай программа түзүү көбүнчө бүтүндөй структуралык программалоо менен тендештириле баштаган. Бирок структуралык программалоонун максаты алда канча глобалдуу болуп ойлонулган мамилени талап кылат.

2.2. Структуралык программалоонун максаттары

Программалоо дисциплинасын камсыздоо

Программалык комплекстерди түзүү процессинде программалоо дисциплинасын камсыздоо структуралык программалоонун негизги максаты болуп, анын дагы башка коюлган максаттарына жетүү үчүн башкы фактор болуп саналат. Дейкстра мындай деп аныктама берген:

«Структуралык программалоо – бул программист өзү өзүнө таңуулаган дисциплина».

Программанын окумдуулугун жакшыртуу

Эгерде төмөнкү эрежелер сакталса программанын окумдуулугу (читабельность) жакшырат.

- айкын эмес семантикалуу тилдик конструкцияларды пайдалануудан качуу.
- башкаруучу конструкциялардын аракеттерин жана берилгендердин структурасын пайдалануну локалдаштырууга умтулуу.
- башка бетке болгон башкаруучу өтүүнү кармабаган, окуганда башынан этегине чейин окууга мүмкүн боло тургандай программаны жазуу.

Программанын эффективдүүлүгүн жогорулатуу

Эгер программаны структуралоо аткарылса, анда каталарды табуу жана аларды корректирлөө женил боло тургандай кылып программаны модулдарга бөлүү менен, ошондой эле эффективдүүлүктү жогорулатуу максатында калаагандай модулдун текстин башка модулдардан көз карандысыз түрдө өзгөртүп жасоого боло тургандай кылып жасоо менен программанын эффективдүүлүгүн жогорулатуу мүмкүн.

Программанын ишенимдүү иштешин жакшыртуу

Баштан аяк тестирлөөгө мүмкүнчүлүк берип оңдоп-түзөө процесин уюштурууда эч кандай проблеманы пайда кылбаган учурда программанын абдан ишенимдүү иштешине жетишүүгө болот. Бул нерсени программаны модулдарга ажыратууда аны жакшы структуралоо жана программанын окумдуулугун жогорулатуу эрежелерин сактоо менен камсыз кылууга болот.

Программаны талдоонун убактысын жана наркын төмөндөтүү

Программалык комплекстерди талдоонун наркынын жана ага кеткен убакыттын төмөндөшү качан талдоочулар командасындагы ар-бир программист мурдагыга караганда көбүрөөк сандагы программалык кодду жазууга жана ондоп-түзөөгө жөндөмдүү болуп калган учурда болушу мүмкүн, башкача айтканда программисттин эмгек өндүрүмдүүлүгү жогорулаган учурда болот. Структуралык программалоонун эрежелерин сактоо менен буга жетишүүгө болот.

2.3. Структуралык методологиянын негизги принциптери

Абстракция принциби

Абстракция программист - талдоочуга коюлган проблеманын талап кылынган чечимин ошол замат бардык майда детальдарды эсепке албастан туруп элестетүүгө мүмкүнчүлүк берет. Абстракция принцибин пайдаланып талдоочу программаны деңгээлдер боюнча карап көрө алат. Жогорку деңгээл бизге чоң абстракцияны көрсөтүү менен долбоорго болгон көз карашты жөнөкөйлөтсө, ошол эле учурда төмөнкү деңгээл реализациялоонун майда деталдарын көрсөтөт.

Көптөгөн структуралык усулдар ушул абстракция принцибине негизделген, мисалы, программмалоонун өйдөлөөчү (входящая) жана түшүүчү (нисходящая) стратегияларын айтсак болот.

Формалдуулук принциби

Структуралык ыкманын экинчи бир фундаменталдык принциби-бул формалдуулук принциби болуп эсептелет. “Формалдуулук” сөзү катуу усулдук ыкманы көздөйт.

Формалдуулук принциби программалоону импровизациядан инженердик дисциплинага айландыруу үчүн база болуп саналат. Андан сырткары, бул принцип программанын тууралыгын далилдөө үчүн негиз болот, анткени ал программаларды (алгоритмдерди) математикалык объекттер катары окуп үйрөнүүгө мүмкүнчүлүк берет.

Кандайдыр бир процесске (техникалык же техникалык эмес) чыгармачылыкты төмөндөтөт деген негизде формалдуулук принцибин кийирүүгө каршы турушат. Бирок структуралык ыкма формалдуулукту чыгармачылык процесске анык бир дисциплинаны жана катуулукту (строгость) берүү үчүн пайдаланылып, ал болсо чечимдерди кабыл алуунун тездетилишине жана көптөгөн каталардан качууга мүмкүнчүлүк берет. Формалдуулук - программалоо искусствосун жана анын инженердик аспекттерин бириктирүүнүн ыңгайлуу жана кубаттуу жолу болуп саналат.

«Бөлүп-жар да бийлик кыл» принциби

Бул принцип Юлий Цезардын убагынан бери эле белгилүү болуп татаал проблемаларды түшүнүүгө жана чечүүгө жеңил болгон майда көз каранды эмес проблемалардын көптүгүнө бөлүп-ажыратуу жолу менен чечүү усулу болуп саналат.

Программалык камсыздандырууну талдап чыгууга колдонуу ракурсунда бул принцип программаны жөнөкөй башкарууга жана көз карандысыз оңдоп-түзөөгө жана тестирлөөгө мүмкүн болгон өзүнчө фрагменттерге (модулдарга) ажыратууну билдирет. Ал программаларды талдоочуларга бүтүндөй системаны кучагына алган абдан көп сандагы деталдарга көңүл бурбастан, ошол чоң системанын өзүнчө бир бөлүгүнүн үстүндө гана беймарал иштөөгө мүмүкүнчүлүк берет.

Иерархиялык иреттөө принциби

Бул принцип “бөлүп-жар да бийлик кыл” принциби менен тыгыз байланышта. Бөлүктөргө ажыратуунун структурасы, ушундай ажыратуу фактысынын өзүнөн кем эмес маанеге ээ. Иерархиялык иреттөө принциби адам ишмердүүлүгүнүн бардык сфераларында чоң ийгилик менен колдонулушу менен бирге көптөгөн бөлүктөрдү өз ичине камтыган системаларды башкаруунун татаал проблемаларын чечүүгө жардам берет.

Программалоодо бул принципти колдонуу программалык комплекстин модулдарынын арасындагы өз ара байланыштарды иерархиялык структуралоо талабын коет, бул болсо структуралык программалоонун жогоруда каралган максаттарына жетүүнү жеңилдетет.

2.4. Модулдук программалоо

Модулдук программалоонун негизги принциби – бул “бөлүп-жар да бийлик кыл” принциби болуп саналат. ***Модулдук программалоо*** – бул программаны, структурасы жана өзүн алып жүрүшү кандайдыр бир эрежелерге баш ийген ***модулдар*** деп аталуучу анчалык чоң эмес көз карандысыз блоктордун жыйындысы катары уюштуруу.

Бул жерде программалоо тилдеринин синтаксистик конструкция (Turbo Pascal тилинде unit), жана чоң программаны өзүнчө блокторго ажыратып бөлүү (алар процедура же функция көрүнүшүндө реализацияланышы мүмкүн) бирдиги учурундагы “модул” сөзүнүн пайдаланылышын айырмалап кароо керек экендигин байкоого болот.

Модулдук программалоону пайдалануу программаны тестирлөөнү жана каталарды табууну жөнөкөйлөтүүгө мүмкүнчүлүк берет. Аппараттык көз каранды камтылма-маселелер (подзадачи) башка камтылма маселелерден так ажыратылып, ал түзүлүүчү программалардын мобилдүүлүгүн жакшыртат.

Программанын эффективдүүлүгүн жогорулатуу процесси жөнөкөйлөйт, анткени убакыт боюнча критикалык модулдарды башка модулдардан көз карандысыз түрдө көп жолу кайра жасоого болот. Андан сырткары модулдук программаларды түшүнүү алда канча жеңил, ал эми модулдардын өзүн башка программаларда курулуш блоктору катары пайдаланууга болот.

Программалоодо «Модул» термини программаларды түзүүдө модулдук принциптердин киргизилишине байланыштуу колдонула баштаган. 70-жылдары модул дегенде анык бир эрежелерге тиешелеш жазылган кандайдыр бир процедураны же функцияны түшүнүшкөн. Мисалы, мына мындай эрежеге: «Модул-жөнөкөй, туюк (көз каранды эмес), көрүмдүү (обозримый) (50 дөн 100 жолчого чейин), маселенин бир гана функциясын реализациялоочу, бир гана кирүү чекитине жана бир гана чыгуу чекитине ээ болушу керек». Бирок жалпы макулдашылган талап болгон эмес, ошондуктан көбүнчө көлөмү 50 жолчого чейин болгон каалагандай процедураны модул деп аташкан.

Программалык модулдун негизги касиеттерин биринчи болуп Парнас (Parnas): *«Бир модулду жазыш үчүн башка модулдун чекити жөнүндөгү минималдык билим жетиштүү болуш керек»*-деп бир

кыйла так формулировкалаган. Ошентип бул аныктамага ылайык, иерархиянын (реализация деңгээлинин) эң төмөнкү деңгээлинин да, ошондой эле башка процедура-модулдарды чакыруу гана жүргүзүлүүчү эң жогорку деңгээлинин ар кандай өзүнчө процедурасы модул боло алган.

Ошентип, Парнас биринчи болуп программалоодо информацияны жашыруу (information hiding) концепциясын көтөрүп чыккан. Бирок 70-жылдардагы тилдердеги процедура жана функция сыяктуу синтаксистик конструкциялардын гана болушу информацияларды ишенимдүү жашырууну камсыз кыла алган эмес, анткени татаал программалардагы жүрүш-турушун алдын ала айтууга көбүнчө мүмкүн болбогон глобалдык өзгөрүлмүлөрдүн таасирине туш болушкан.

Бул проблеманы глобалдык өзгөрүлмөлөрдүн таасирине дуушар болбоочу жаңы синтаксистик конструкцияларды иштеп чыгуу менен гана чечүүгө мүмкүн болгон.

Мындай конструкция түзүлгөн жана ал модул деп аталган. Оболу модул, татаал программалык комплекстердин реализацияланышында анык бир камтылуучу маселенин (подзадача) реализациясынын деталдарын бириктирүүчү жана ишенимдүү жашыруучу конструкция катары, процедура жана функциялар менен бир катарда пайдаланылышы керек деп эсептелинген.

Ошентип чоң программалык комплекстеги модулдардын саны коюлган маселени көз каранды эмес камтылуучу маселелерге декомпозицияланышы менен аныкталышы керек.

Биринчи жолу модулдун синтаксистик конструкциясы 1975-жылы Н.Вирт тарабынан сунуш кылынып, анын жаңы Modula деген тилине кошулган. Ушул эле жылы Modula тилинин тажрыйбалык реализациясы жасалган. Бир канча кайра иштелип чыгылгандан кийин бул жаңы тил 1977-жылы биротоло реализацияланып Modula-2 деген атты алган. Натыйжада буга окшогон конструкциялар бир аз гана айырмачылыктары менен башка программалоо тилине да кошулган, мисалы Pascal Plus (Уэлш жана Бастард, 1979-жыл), Ada (1980) жана Turbo Pascal 4.0.

Модулдук программанын формасы

Модулдук программанын иерархиялык структурасына жакшы форманы берүү аны иштеп чыгуу процессин жакшыртат. Кандайдыр бир модул тарабынан чакырылуучу модулдардын саны жана аны чакыруучу модулдардын саны программанын татаалдыгына таасирин тийгизет. Иодан (Yourdon) берилген модул тарабынан чакырылуучу модулдардын санын модулдарды башкаруунун кулачы (размах) же

кеңдиги деп атаган. Чоң өлчөмдөгү модул менен катар модулдарды башкаруунун өтө кичине же өтө чоң кеңдиги жаман (начар) схема боюнча модулдарга ажыратуунун белгиси болуп саналат. Жалпы учурда модулдарды башкаруунун кеңдиги 10 дон ашпастыгы керек. Бул сан психологиялык абалдарга, өзгөчө информациянын «үзүмдөрү» (“chunking”-“кусок”) теориясына негизделүүчү “сыйкырдуу” 7 саны менен байланышкан. Адамдын кыска убакыттуу эси информациянын “үзүмдөрүн” сактоодо чектелген жөндөмдүүлүккө ээ. Психологиялык эксперименттер көрсөткөндөй биздин кыска убакыттуу эсибиздин жөндөмдүүлүгү 5-9 “үзүм” (орточо-7) пределинде турат экен. Ал бир убакытта 7 “үзүмгө” жакын информацияны иштете алат. Качан ушул пределден ашкандан кийин адам көбүрөөк ката кетире баштайт.

Ылайыктуу сандагы майда бөлүктөргө бөлүп информацияны кайра уюштуруу адамдын кыска убакыттуу эсин эффективдүү пайдалануу жана материалдын түшүнүктүүлүгүн жакшыртуу үчүн негизги аракет болуп эсептелет. Турмушта көпчүлүк ситуацияларда адамдар мындай кайра уюштурууну аң-сезимдүүлүк менен жасашпайт. Бирок программист модулдарды башкаруу кеңдигин 7 ден ашып кетишине жол койбоо менен өзүнө-өзү жардам бере алат.

2.5. Структуралык программалоонун стандарттары

1. Программа модулдар деп аталган көз каранды эмес бөлүктөргө ажыратылышы керек.
2. Модул-бул коду башка модулдардын кодунан физикалык жана логикалык түрдө ажыратылган көз каранды эмес блок.
3. Модул бир гана логикалык функцияны аткарат.
4. Модулдун өлчөмү 100 оператордон ашпашы керек.
5. Модул бир кирүү чекитине жана бир чыгуу чекитине ээ.
6. Модулдардын ортосундагы өз ара байланыш иерархиялык структура боюнча орнотулат.
7. Ар бир модул анын кызматын, өзгөрмөлөрдүн арналышын, бул модулга берилүүчү модулдар, жана бул модулду чакыруучу ошондой эле бул модулдун ичинен чакырылуучу модулдарды баяндоочу комментарийден башталышы керек.
8. Керексиз эн белгилерден качуу менен goto операторун модулдун кирүү же чыгуу чекитине өтүү үчүн гана колдонууга же таптакыр эле пайдаланбоого аракет кылуу керек.
9. Бардык өзгөрүлмөлөрдүн жана модулдардын идентификаторлору мааниге ээ болгондой болушу керек.

10. Тектеш (жакын) тайпадагы идентификаторлор бирдей префикс менен башталышы керек.
11. Стандарттык гана башкаруучу конструкцияларды (тандоо, цикл, чыгуу, блок) пайдалануу керек.
12. Бир жолчого бирден көп эмес оператор жазуу керек. Эгер операторду жазуу үчүн бирден көп жолчо талап кылынса, анда ар бир кийинки жолчо бош жылышкан орун таштоо (отступ) менен жазылат.
13. If операторунун камтылуучулугуну 3 деңгээлден ашып кетүүсүнө жол койбоо керек.
14. Семантикасы айкын болбогон тилдик конструкцияларды жана программисттик «трюктарды» пайдалануудан качыш керек. Мисалы С тилинде бир караганда күтүлгөн аракетке таптакыр карама-каршы келген ишти аткарган программанын болушу толук мүмкүн:

```
# include <stdio.h>
main ()
{
if (3<2<1) printf (“Бул болбогон нерсе!!!”);
else printf (“А Паскаль болсо буга жол бербейт”);
}
```

3. ПРОГРАММАЛАРДЫ ТАЛДООНУН ОБЪЕКТТИК -ОРИЕНТИРЛЕНГЕН МЕТОДОЛОГИЯСЫ

Акылдуу болгуң келсе, акылын таап сурай бил, абайлап уга бил, кыскача жооп бере бил, өзүңө бөгөт коё бил!
(И. Лафатер)

3.1. Негизги түшүнүктөр жана аныктамалар

Программаларды талдоонун объекттик-ориентирленген усулдары жаатында таанымал авторитет болгон Гради Бучтун аныктамасы боюнча объекттик-ориентирленген программалоо (ООП) - бул программаларды ар бири анык бир класстын (өзгөчө көрүнүштө-типте болгон) реализациясы болгон, ал эми класстар мурасталуучулук принциптердин негизинде иерархияны түзүүчү объекттердин жыйындысы көрүнүшүндө көрсөтүүгө негизделген программалоонун методологиясы.

Объекттик-ориентирленген методология (ОО-методология) структуралык методология сыяктуу эле чоң программалык комплекстерди талдап иштеп чыгуу процессин тартиптештирүү, ошону менен анын татаалдыгын жана наркын төмөндөтүү максатында түзүлгөн.

ОО-методология структуралык методологиядай эле максатты көздөйт, бирок ал максаттарды башкача башталыш чекитинен баштоо менен чечип, көпчүлүк учурда структуралык методологияга караганда бир кыйла татаал долбоорлорду башкарууга мүмкүнчүлүк берет.

Биз мурда белгилеп өткөндөй долбоордун татаалдыгын башкаруу принциптеринин бири декомпозиция болуп саналат. Гради Буч декомпозициянын эки түрүн көрсөтөт: алгоритмдик (мынтип ал структуралык усулдар менен жүргүзүлгөн декомпозицияны атайт) жана объекттик-ориентирленген. Анын ою боюнча булардын айырмасы төмөндөгүчө: «Алгоритмдер боюнча ажыратуу - болуп жаткан окуялардын иретине көңүлдү топтойт, ал эми объекттер боюнча ажырытуу - же аракеттерди пайда кылуучу факторлорго, же бул аракеттердин колдонулуш объекттери болгон факторлорго өзгөчө маани берет».

Башка сөз менен айтканда алгоритмдик декомпозиция көбүрөөк деңгээлде татаал проблеманын бөлүктөрүнүн ортосундагы өз ара байланыштардын ***структурасын*** эсепке алат, ал эми объекттик-ориентирленген декомпозиция өз ара байланыштардын ***мүнөзүнө*** көбүрөөк көңүл бурат.

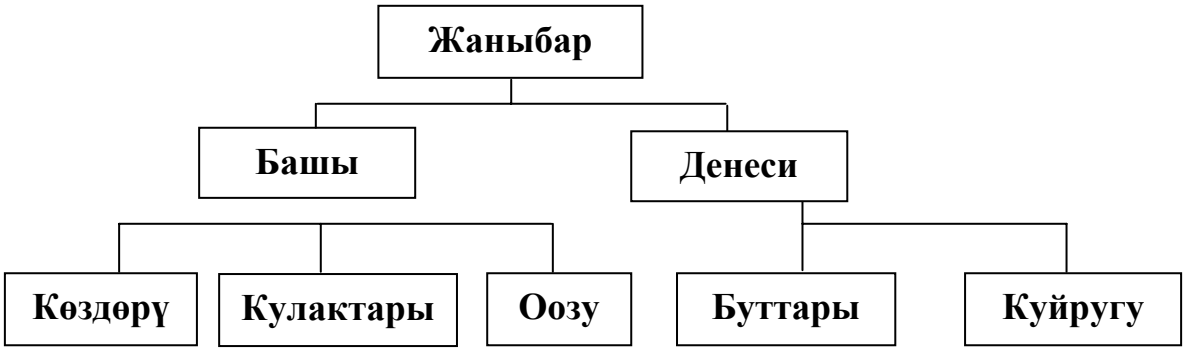
Практикада декомпозициянын бул эки түрүн тең колдонуу сунуш кылынат: чоң долбоорлорду түзүүдө адегенде объекттердин жалпы иерархиясын түзүп алуу үчүн объекттик-ориентирдик ыкманы, программалануучу маселенин мазмунун чагылдыруучу, андан соң талданылуучу программалык комплексти талдоону жана коштоону жөнөкөйлөтүү үчүн модулдарга алгоритмдик декомпозициялоо ыкмасын пайдалануу максатка ылайыктуу.

«Объект» түшүнүгүнүн бир нече аныктамасы бар экендигин белгилөөгө болот. Биз Гради Бучтун ою менен макулдукта объекттин аныктамасын мындайча беребиз: «объект-бул өзүнүн жүрүш-турушун (поведение) так көрсөтө алган реалдуу кабыл алынуучу маңыз (мазмуун)».

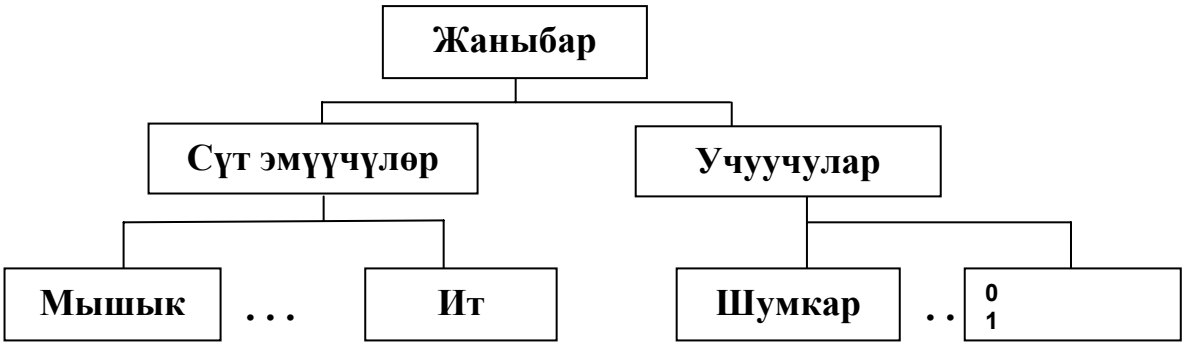
Объект өзүнүн бардык касиеттеринин жыйындысы (мисалы жаныбарлар үчүн – бул башынын, кулактарынын, көздөрүн ж.б болуусу) жана алардын учурдагы маанилери (башы-чон, кулактары-узун, көздөру-күрөң ж.б) менен да ошондой эле берилген объект үчүн уруксат берилген аракеттердин жыйындысы (тамак жей алышы, тура, отура, жүгүрө ж.б алышы) менен да мүнөздөлөт. «Материалдык» курамдык бөлүктөрдүн жана бул бөлүктөрдүн үстүнөн жүргүзулүүчү аракеттердин бир, бүтүн объектке жогоруда көрсөтүлгөндөй биригүүсү ***инкапсуляция*** деп аталат. Башка объекттерге таасир этүү менен жана өз кезегинде алардын таасирине кабылуу менен объекттер дайыма өз ара аракетте болушат.

Долбоордун татаалдыгын башкаруунун дагы бир принциби-декомпозиция процессинде алынуучу объекттерди (камтылуучу маселелерди (подзадачи)) иерерхиялык иреттөө болуп саналат.

Структуралык методология дагы, ОО-методология дагы объекттер ортосундагы өз ара байланыштардын иерархиялык дарагын тургузууну көздөйт. Бирок, эгер структуралык иерархия бүтүндү курама бөлүктөргө ажыратуу деген жөнөкөй принцип боюнча тургузулса,



анда объекттик-ориентирленген иерархияны (ОО-иерархия) түзгөн учурда ошол эле баштапкы объектке болгон башкача көз-караш кабыл алынып, иерархияда **аталык** (родительский) типтеги (жогору жайгашкан) объекттердин касиеттеринин **балалык** (дочерний) типтеги (төмөн жайгашкан) объекттерге мурасталышы сөзсүз чагылдырылат.



Аталык типтер дагы башкача жөн гана **аталар** же **бабалар** (предки) деп, ал эми балалык типтер **урпактар** (потомки) деп аталат. Эгерде А тиби тикеден тике (аралык типтерсиз) төмөн жайгашкан В тиби менен байланышкан болсо, анда А тиби В тибинин **түздөн-түз бабасы** (атасы) деп ал эми В тиби А тибинин **түздөн-түз урпагы** деп аталат.

Гради Бучтун айтканы боюнча «***мурастоо*** – бул бир объект экинчисинин структурасын жана жүрүш-турушун кайталаган кездеги объекттердин ортосундагы катнаш».

Мурастоо принциби турмушта кадам сайын жана күн сайын аракет этет. Сүт эмүүчүлөр жана учуучулар тирүү организмдин белгисин мурасташат, өсүмдүктөрдөн айырмаланып бүркүт менен таранчы учуучуларга жалпы болгон уча алуу касиетин мурасташат. Экинчи жактан, арстандар, жолборстор, кабыландар сыяктуу жаныбарлар мышыктар түркүмүнө мүнөздүү болгон «структураны» жана жүрүм-турумду мурасташат ж.у.с.

ОО-иерархиянын жогорку денгээлдериндеги типтер, эреже катары, объекттердин конкреттүү нускаларына (экземпляр) ээ болушпайт. Мисалы үчүн өзүнчө «Сүт эмүүчү» же «Учуучу» деп аталуучу тирүү организм (объект) жок. Мындай типтер ***абстракттык*** типтер деп аталат. Объекттердин конкреттүү нускаларына, эреже катары, ОО-иерархиялын төмөнкү денгээлдериндеги типтер ээ болушат: «крокодил» Гена – бул «крокодил» тибинин конкреттүү нускасы, ит Кумайык – үй ити тибиндеги объекттин конкреттүү нускасы.

ООПтун негиздөөчү түшүнүктөрүнүн дагы бири - полиморфизм. ***Полиморфизм*** – бул ар түрдүү обьетктердин бир эле аракетти өз-

өзүнчө ар түрдүүчө (өзүнүкү боюнча) аткаруу касиети болуп эсептелет. Мисалы «чуркоо» аракети көпчүлүк жаныбарларга таандык касиет. Бирок алардын ар бири (арстан, пил, коён, ташбака) бул аракетти ар түрдүүчө аткарышат.

Традициялык (объекттик-ориентирленген эмес) ыкма менен программалоодо программист конкреттүү жаныбар жана конкреттүү аракет үчүн өзүнчө камтылуучу программаны чакыруу менен бул жаныбарларды жылдырат. Объекттик-ориентирленген ыкмада программист кайсы объект үчүн ага таандык болгон аракеттердин ичинен кайсы аракетти аткаруу керек экендигин гана көрсөтөт жана (биз карап жаткан мисал үчүн) мурда баяндалган *объекттер – жаныбарлар аларга мүнөздүү болгон ыкма (жол) менен анын курамына кирген усулдарды (методдорду) пайдаланып өздөрүн-өздөрү жылдырат.*

Ошентип, биздин учурда «чуркоо» аракети полиморфиялык аракет деп, ал эми бул аракеттин ишке ашырылыш формаларынын көп түспөлдүүлүгү – ***полиморфизм*** деп аталат.

3.2. Turbo Pascal тилинин объекттик ориентирленген каражаттары

3.2.1. Объекттердин типтерин жана нускаларын (көчүрмөлөрүн) баяндоо

Объекттик типтеги өзгөрүлмөлөр – ***объекттердин нускалары*** – булар дагы жазуулар сыяктуу берилгендердин комбинацияланган (аралашма) структуралары болуп саналат. Баяндоонун сырткы көрүнүшү боюнча да алар жазууларга окшош.

Объекттерди баяндоонун жазуулардын баяндоосунан болгон ***биринчи синтаксистик айырмачылыгы*** - **record** кызматчы сөзүнүн ордуна **object** кызматчы сөзүнүн пайдаланылышы.

Мисалы, жөнөкөй учурда, шахмат тактайындагы шахмат фигурасынын абалы объекттик типтин жардамында мындайча баяндалышы мүмкүн:

```
Type
{Тактайдагы позиция}
TPozition = object
            Column: TColumns;
             Row   :  TRows;
end;
```

мындагы TColumns жана TRows типтери төмөндөгүдөй көрүнүшкө ээ:

```
Type
{Тактайдагы координаталардын типтери}
        TColumns = ‘a’ .. ‘h’;
        TRows = 1..8;
```

Объекттин жазуудан болгон экинчи айырмачылыгы – объект берилген учурда усулдар (методдор) деп аталуучу процедура жана функция көрүнүшүндө жазылган (жасалган) аракеттерди өзүнө алып турушу мүмкүн.

Өзгөчө бөлуп көрсөтүүчү нерсе, ***объекттер менен иштөөнүн жакшы стили алардын талааларына болгон түз кайрылууну жокко чыгарат.***

ОО – методология жакшы-так иштеши үчүн объекттер бардык аракеттерди өздөрүнүн усулдарынын жардамында аткарышы талап кылынат. Мындай катуу талап структуралык программалоо дисциплинасынын жалпы түшүнүгүнө туура келет. Ошентип, ***объекттин информациялык талааларынын үстүнөн аткарылуучу ар кандай аракет үчүн өзүнчө усул жазылган болушу керек.***

Мисалы, шахмат фигурасынын позициясы менен, башкача айтканда TPosition тибиндеги объект менен иштеш үчүн төмөндөгүдөй усулдарды түзүүгө болот.

- Объекттин талааларынын маанилерин коюу (установка) үчүн усул (Init процедурасы);
- Шахматтык фигуранын учурдагы колонкасын аныктоо үчүн усул (GetColumn функциясы);
- Шахматтык фигуранын учурдагы жолчосун аныктоо үчүн усул (GetRow функциясы).

Программада *усулдардын бөрктөрүн баяндоо объекттин информациялык талааларынын баяндалыштарынан кийин жайгашат* жана төмөндөгүдөй көрүнүштө болот

```
                {Тактайдагы позиция}
TPosition=object
    Column: TColumns;
    Row: TRows;
    procedure Init (Cl: TColumns; Row: TRows);
    function GetColumn: TColumns;
```

```
    function GetRow: TRows;
  end;
```

Init процедурасындагы эки Cl жана Rw формалдык параметри шахматтык фигура коюла турган позициянын координаталарынын иш жүзүндөгү маанилерин берүү үчүн арналган.

*Информациялык талааларды жана алардын үстүнөн аракеттерди (амалдарды) аткаруучу усулдардын бир объектке биригүүсү "**инкапсуляция**" деп аталуучу ОО-түшүнүгүнө туура келет.*

Жазуулар сыяктуу эле объекттик типтеги өзгөрүлмөлөрдүн талааларына квалификацияланган (такталган) идентификаторлордун жардамында да ошондой эле with операторунун жардамында да кайрылууга уруксат берилет.

Мисалы, өзгөрүлмөлөрдүн төмөндөгүдөй баяндалышында

```
Var
  WKingPos: TPosition;
  KingCol   : TColumns;
  RingRow  : TRows;
```

программанын текстиндеги төмөнкү операторлордун ордуна

```
begin
  WKingPos.Columns  :='e';
  WKingPos.Row       :=1;
  KingCol    := WKingPos.Column;
  KingRow   := WKingPos.Row;
end.
```

төмөнкү көрүнүштөгү with операторун пайдалануу мүмкүн

```
begin
   with  WKingPos  do
     begin
      Column  :='e';
      Row      : =1;
      KingCol  := Colmn;
      KingRow := Row
     end;
end.
```

Жазуулар үчүн болгон учурдагыдай эле with операторунун объекттик типтерге колдонулушу мүмкүн эле болбостон аны колдонуу сунуш кылынат.

Эске салчу нерсе, объекттер менен иштөөнүн жакшы стили объекттин информациялык талааларына болгон түз кайрылууну жокко чыгарат. Ошондуктан, катуу эрежеге ылайык келтирилген фрагменттеги төмөнкү операторлорду

```
Colmn     :='e';
Row       :=1;
KingCol   :=Colmn;
KingRow :=Row
```

объектти инициализациялоо процедурасын жана учурдагы колонканын жана учурдагы жолчонун маанилерин алуу функцияларын чакырууга алмаштыруу талап кылынат. Бул болсо, объекттин информациялык талаалары учурундагыдай эле ошондой эле кваликацияланууучу идендикаторлордун жардамында мындайча

```
begin
   WKingPos.Init('e',1,);
   KingCol   :=WKingPos.GetColmn;
   KingRow  :=WKingPos.GetRow
end;
```

же with опернаторунун жардамында мындайча

```
with WKing Fos do
   begin
     Init('e',1);
     KingCol  :=GetColmn;
     KingRow :=GetRow;
   end;
```

жазылышы мүмкүн.

Көңүл бурчу нерсе, ***объекттик типтерди баяндоонун типтер бөлүгүндө усулдардын бөрктөрү гана көрсөтүлөт. Усулдардын толук баяндалышы - алардын телосу - процедураларды жана функциларды жарыялоо бөлүгүндө көрсөтүлөт.***

Type бөлүгүндө TPosition тибин жарыялоо кийинки init, GetColumn жана GetRow толук баяндалыштары менен толукталышы керек

```
Type
   {тактайдагы координаталардын типтери}
```

```
TColumns  = 'a' . . 'h';
TRows = 1 . . 8;
 {тактайдагы позиция}
TPosition =object
    Column: TColumns;
    Row: TRows
    procedure  Init (Cl: TColumns; Rw: TRows);
    function GetColumn: TColumns;
    function GetRow: TRows;
     end;
                {усулдардын реализациясы }
                    {Tpostion  объекти}

procedure TPosition.Init (Cl: TColumns; Rw:TRows);
begin
   Column  :=Cl;
    Row      : Rw;
end;

function TPosition. GetColumn: TColumns;
begin
  GetColumn :=Column;
end;

function TPosition.GetRow: TRows;
begin
  GetRow:=Row;
end;
```

Усулдардын реализациясын баяндоонун башкы өзгөчөлүгү – бул аныкталуучу Init, GetColumn, GetRow усулдары таандык болгон Tpostion объекттик тибинин Column жана Row ***информациялык талааларына кайрылуу***, ***усулдардын оператордук блогу көрүнбөс*** with ***оператору менен курчалгандагыдай ишке ашырылат***.

Бул объекттин талааларынын жана усулдарынын аракет этүү областынын (көрүмдүүлүк сферасынын) эрежелеринен келип чыгат. Бул эрежелерге ылайык талаалар жана усулдар програманын текстинин ар түрдүү жерлеринде баяндалган болушуна карабастан бир эле аракет этүү областын бөлүшүшөт.

Эгерде ObjectVar объекттик тибинин кандайдыр бир көчүрмөсү (нускасы) ага таандык болгон Method усулун чакырса, анда объектти

жана анын усулдарын жалпы бир (бирдиктүү) аракет этүү областына байланыштыруучу айкын эмес

```
with ObjectVar do Method
```

оператору аткарылат.

Мисалы, төмөнкүдөй чакыруу болгондо

```
WKingPos.Init  ('e',1);
```

TPosition процедурасынын тексти компилятор тарабынан мындайча трактовкаланат

```
procedure TPosition.Init(Cl:TColumns; Rw: Rows);
begin
  with WKingPos do
    begin
      Column  : = Cl;
      Row     : = Rw;
    end;
end;
```

ал эми

```
Wqueen Pos.Init('e',1);
```

чакыруусу болгондо мындайча трактовкаланат

```
procedure TPosition.Init(Cl:TColumns; Rw: Rows);
begin
   with  WQueenPos do
  begin
    Column  : = Cl;
    Row     : = Rw;
  end;
end:
```

Иш жүзүндө TPosition.Init процедурасынын баштапкы тексти, көрсөтүлгөн мисалдардагыдай, чакыруудан көз каранды түрдө ***өзгөрбөй тургандыгын*** байкайбыз. Бул мисалдар болуп өтүүчү аракеттердин маанисин түшүндүрүү үчүн гана келтирилди.

Чындыгында усулдардын бөрктөрүндө усулду чакырууда иш жүзүндөгү (фактический) параметр катары объекттин конкреттүү нускасы айкын эмес түрдө берилүүчү Self деген идентификаторлуу көрүнбөс формалдык параметр ар дайым катышып турат. Self

параметри иш жүзүндө усулдун чакырылышын ишке ашыруучу объекттин нускасына болгон 32 разряддуу көрсөткүч болуп саналат.

Ар бир усулдун көрүнбөс (невидимый) баяндоолорунун структурасын мындайча көрсөтүүгө болот:

```
procedure MethodName(Self: Pointer ;башка параметрлер);
begin
   with self  do
     begin
       усулдун  операторлору
     end;
end;
```

Белгилей кетчү нерсе, Self усулда ар дайым айкын эмес төрдө катышып турганы менен, аны айкын түрдө да пайдаланууга болот.

Мисалы, TPositcon тибиндеги Init процедурасынын баяндалышында төмөндөгүчө жазууга болот

```
procedure Tposition. Init (Cl: TColumns; Tw: TRows);
 begin
         self.Column  := CL;
         self.Row :=Rw;
  end;
```

Бирок Self операторун зарыл болгон учурда гана колдонуу сунуш кылынат, мисалы, идентификаторлордун бир маанилүү эместигин жоготуу үчүн колдонсо болот.

3.3. Объекттер жана модулдар

Объекттик типтердин баяндоосунун жана алардын усулдарынын реализацисынын ажыратылып бөлүнүшү гармоникалык түрдө модул көрүнүшүндө жасоого тиешелеш келип, ал модулда да экспорттолуучу процедуралардын бөрктөрүнүн интерфейстик бөлүктөгү баяндоосу жана реализация бөлүгүндөгү алардын реализациялоо коду ажыратылып бөлүнөт.

Объекттик типтер көбүнчө ар түрдүү маселелерди чечүүдө пайдалана алууга мүмкүн болсун үчүн киргизилгендиктен алар, эреже катары, interface бөлүгүндө баяндалат.

Усулдардын реализациясынын баяндалышын идеалдуу турдө implementation бөлүгүндө жайгаштыруу туура болот, анткени объекттерди колдонуунун манызы боюнча, алардын усулдарынын конкреттүү иштери пайдаланууучудан жашырылган болушу талап

кылынат. Жалпысынан модулдар үчүн кабыл алынган өзгөрүлмөлөрдүн аракет этүү сферасынын жалпы принциптери объекттер үчүн да жайылтылат: интерфейстик бөлүктө эмнелер баяндалган болсо, ошонун баарын реализация бөлүгүндө пайдаланууга болот.

Жогоруда киргизилген TPosition объекттик тибин модул көрүнүшүндө жазуу төмөнкүдөй болот.

```
unit   ChessMod;
Interface
type
                { Тактайдагы координаталардын типтери }
                  TColumns= ’a’ .. ’h’;
                  TRows= 1 .. 8;
                  { Тактайдагы позиция }
          TPosition = object
                Column:   TColumns;
                Row:       TRows;
                procedure   Init( Cl: TColumns; Rw: TRows);
                function   GetColumn: TColumns;
                function   GetRow: TRows;
    end;
Implementation
          { Усулдардын  реализациясы}
                { TPosition  объекти}
 procedure TPosition.Init (Cl: TColumns; Rw: TRows);
    begin
       Column :=Cl;
       Row :=Rw;
 end ;
 function TPosition.CetColnmn:  TColumns;
 begin
       GetColumn:=Column;
  end;
  function  Tposition.GetRow: TRows;
  begin
       GetRow:=Row;
   end;
 begin

   end.
```

3.4. Private жана public директивалары

Объекттердин Turbo Pascal 7.0 тилинде реализацияланышы талааларын жана усулдардын аракет этүү сферасын (областын) башкаруу үчүн кошумча каражаттарга ээ. Биринчи каражат – бул private директивасы болуп ал мурдагы Turbo Pascal 6.0 версиясында эле пайда болгон, ал эми экинчиси – public директивасы – бул директива ***тилдин жетинчи версиясында гана биринчи жолу киргизилген.***

3.4.1. Private директивасы

Private *директивасы объектти баяндоо аймагында*, модулдагы implementation реализация бөлүгү аткаргандай эле кызматты аткарат. Башкача айтканда, объектти баяндоонун кээ бир деталдарын келечекте бул объекттин усулдарын пайдалана турган пайдалануучулардан жашыруу үчүн арналган. Ошентип Turbo Pascal объекттердин ичинде жабык (приваттык–private) талааларды жана усулдарды берүүгө мүнкүнчүлүк берет.

Приваттык талааларга жана усулдарга, объект баяндалган модулдун ичинде гана кайрылууга (колдонууга) болот.

Атайын көрсөтүлбөгөн учурда (по умолчанию) object резерв делген сөзүнөн кийин эле дароо баяндалган талаалар жана усулдар жалпы колдонулуучу (общедоступные) деп кабыл алынат. Приваттык талаалар жана усулдар кадимки жалпы колдонулуучу талаалардын жана усулдардын баяндалышынан кийин, private резерв делген сөзүнө улдаалаш баяндалат.

Private директивасын пайдалануу менен объектти баяндоонун структуурасын төмөнкүчө көрсөтүүгө болот

```
type
   objectType=object (Аталык объекттин тиби ).
      Жалпы колдонулуучу талаалар
      Жалпы колдонулуучу усулдар
    private
      Приваттык талаалар
      Приваттык усулдар
end;
```

Мисалы, биз карап жаткан TPosition тиби үчүн Column жана Row талааларын (шахматтык модулдун пайдалануучулары аларга

түздөн-түз эмес, усулдардын каражаттары аркылуу гана кайрылсын үчүн), ошондой эле Init усулун (анткени аны, кийинчерээк көрүнгөндөй, пайдалануучулар тарабынан түздөн-түз чакыруу талап кылынбый калат) приваттык кылуу максатка ылайыктуу.

```
{ Тактайдагы позиция }
 TPosition = object
       function GetColumn:   TColumns;
       function GetRow:      TRows;
    private
       Column:   TColumns;
        Row:      TRows;
        procedure   Init( Cl: TColumns; Rw: TRows);
   end.
```

Натыйжада, TPosition тиби модулдун интерфейстик бөлүгүндө баяндалганына карабай ChessMod модулун импортточу модулдарда Column жана Row талааларына жана Init усулуна кайрылуу мүмкүн болбой калат.

3.4.2. Public директивасы

Жогоруда айтылгандай, приваттык талаалар жана усулдар жалпы колдонумдуу талаалардан жана усулдардан кийин жайгашкан болушу керек. Жаңы Public директивасы, эгер чындыгында эле ыңгайлуу боло турган болсо, аларды тескери тартипте жайгаштырууга мүмкүнчүлүк берет.

```
type
   objectType=object (Аталык тип ).
     private
       Приваттык талаалар
       Приваттык усулдар
     public
       Жалпы колдонумдуу талаалар
       Жалпы колдонумдуу усулдар
end;
```

Ошентип, TPosition тибинин талааларынын жана усулдарынын баяндоолорунун мурда кабыл алынган удаалаштыгын бузбаш үчүн аны төмөндөгүчө жарыялоо мүмкүн:

```
{ Тактайдагы позиция }
TPosition = object
     private
          Column:   TColumns;
          Row:      TRows;
          procedure   Init( Cl: TColumns; Rw: TRows);
       public
       function GetColumn:   TColumns;
       function GetRow:      TRows;
end;
```

3.5. Мурастоо жана мурастоо эрежелери

Кандайдыр бир тип тарабынан аталык типтин мүнөздөмөлөрүн мураска алуу факты object кызматчы сөзүнөн кийин тегерек кашаалардын ичинде түздөн-түз аталык типтин атын көрсөтүү менен белгиленет.

Мисалы, TPosition тибинин урпагы (укум-тукуму) болгон жана жалпыланган (абстракттык) шахматтык фигуранын мүнөздөмөлөрүн баяндаган шахматтык иерархиянын TChessMan тибин алалы.

```
{Фигуранын түсү}
  TColour  =  (WhiteColour, BlackColour);
                  {Шахматтык фигура}
     TChessMan  =  object ( Tposition );
     private
               Colour:     TColumns;
    Present :  TColour;

  procedure   Init ( Cl: TColumns; Rw: TRows; Rw: TColour );
  procedure   Put ( Name : String );
    public
               function  GetColour :    TColour;
               function  IsPresent  :    Boolean;
               procedure Clear;
               procedure  Del;
               procedure   Display;
  end;
```

Бул тип өзүнүн TPosition аталык тиби менен салыштырмалуу Colour (түс: ак / кара) жана Present (фигуранын тактайда турушу: True / False) информациялык талаалары жана төмөнкү усулдар менен кеңейтилген:

- фигуранын түсүн берүүчү GetColour функциясы менен;

```
function  TChessMan.GetColour :   TColour;
begin
      GetColour :   TColour;
end;
```

- фигуранын тактайда бар экендигин текшерүүчү IsPresent функциясы менен;

```
function  TChessMan. IsPresent :   Boolean;
begin
      IsPresent :  =  Present;
end;
```

- Clear процедурасы менен, бул процедура фигураны жаңы позицияга жылдыруу үчүн анын экрандын мурдагы эски позициясындагы сүрөттөлүшүн өчүрөт.

```
procedure TChessMan.Clear;
begin
     Put  (‘ ‘);
     ErrorMessege (‘                    ‘);
end;
```

- Del прецедурасы менен ал Clear процедурасын чакыруунун жардамында фигуранын сүрөттөлүшүн экрандан өчүрөт жана аны тактайдан өчүрүлдү (алынып ташталды) деп белгилейт.

```
procedure TChessMan. Del;
begin
   Clear;
   Present :  = False;
end;
```

- Display прецедурасы менен ал фигуранын сүрөттөлүшүн анын учурдагы позициясында пайда кылат.

```
procedure TChessMan.Display;
begin
      Put (‘CM‘);
end;
```

• Put процедурасы менен, ал болсо шахматтык координаталарды экрандык координаталарга өзгөртүп түзүү үчүн GetGoords усулун чакырат, талаага жана фигурага тиешелеш болгон түстү коет жана фигуранын сүрөттөлүшүн анын учурдагы позициясында чыгарат.

```
procedure TChessMan.Put (Name:String);
var
    X,Y :   Word;
begin
      GetGoords(X,Y);
      GotoXY (X,Y);
      if (Row+Ord (Column)) mod 2 = 0 then
           TextBackGround (Black)
           else TextBackGround (White);
      case Colour of
            WhiteColour :   TextColor(LightRed);
             BlackColour :  TextColor(LightGreen);
      end;
             write(name);
end;
```

Бул жерде белгилеп кетчү нерсе, TPosition тибине GetGoords жана ErrorMessage усулдары кошулуп, Put усулу менен бирге шахматтык аракеттерди жана чыгарылуучу билдирүүлөрдү экрандын мониторундагы сүрөттөлүшкө «байлоо» («привязка») үчүн кызмат кылышат.

GetGoords усулу логикалык шахматтык координаталарды физикалык экрандык координаталарга которот жана аларды чакыруу чекитине кайтарат. TPosition тиби бардык шахматтык фигуралар тарабынан мураска алынгандыктан, алардын ар бири жүрүштү жосоодо өзүнүн шахматтык координаталарын экрандык координаталарга өзгөртүп түзө жана андан кийин өзүн дисплейде көрсөтө алат.

GetGoords усулун реализациялоонун мүмкүн болгон варианттарынын бири төмөнкүдөй көрүнүшкө ээ:

```
procedure TPosition.GretCoords(X,Y : Word);
const
      BegX = 20; BegY = 23;
      StepX = 7; StepY = -3;
begin
     X:=BegX+StepX*(Ord(Column)-Ord('a'));
     Y:=BegY+StepY*(Row-1);
end;
```

ErrorMessage усулу экранга каталар жөнүндөгү билдирүүлөрдү чыгарат:

```
procedure TPosition.ErrorMessage(Mes: String);
Const
      BegX = 23;
begin
      GotoXY (1,BegY);
      Text BackGround (Blue);
      TextColor(LightRed+Blink);
      write (Mes);
end;
```

Балалык типтер тарабынан алардын аталык типтеринин информациялык талааларын жана усулдарын мураска алуусу ***төмөнкү эрежелерге ылайык*** аткарылат.

1.

Аталык типтин информациялык талаалары жана усулдары анын бардык балалык типтерине иерархиянын аралык денгээлдеринин санынан көз карандысыз турдө мурасталат.

2.

Аталык типтердин талааларына жана усулдарына балалык типтин рамкасында туруп кайрылуу, ал талаалар жана усулдар балалык типтин өзүндө баяндалгандай жүргүзүлөт. Мындан төмөнкү 3-эреже келип чыгат.

3.

Балалык типтердин эч биринде аталык типтердин кандайдыр бирөөсүнүн талааларынын идентификаторлору менен дал келген талаа идентификатору пайдаланыла албайт. Ушул эле нерсе усулдардын бөлүктөрүндө көрсөтүлүүчү формалдык параметрлердин идентификаторлоруна да тиешелүү.

4.

Балалык тип керегинче каалаган сандагы өздүк усулдарды жана информациялык талааларды аныктап ала алат.

5.

Аталык усулдун текстиндеги ар кандай өзгөрүү, аны чакыруучу балалык типтердеги бардык усулдарга автоматтык түрдө таасирин тийгизет.

6.

Информациялык талааларга айтылгандарга тескеринче(3-эреже) балалык типтердеги усулдардын идентификаторлору аталык типтердеги усулдардын аттары менен дал келе алат. Бул учурда балалык типтин рамкасында балалык усул ага аты окшош (бирдей) аталык типтин усулун басат (подавляет) деп айтышат да, мындай усулдун атын көрсөткөндө аталык эмес, кудум балалык усул чакырылат.

Бирок басылган аталык усулга кайрылуу мүмкүнчүлүгүн квалификацияланууучу (такталган) идентификатордун жардамында алуу кала берет. Мисалы, мындай нерсе реализациясы төмөнкү көрүнүштө болгон TChessMan.Init усулунда жазылган:

```
procedure TChessMan.Init (Cl: TColumns; Rw: TRows; Bw:
TColour);
begin
TPosition.Init (Cl, Rw);
Colour: =BW;
Present: =True;
end;
```

Эми мисал катары хан (король) жана жөөчү (пешка) сыяктуу конкреттүү шахмат фигураларынын типтерин баяндайлы.

```
{Объекттик типтердин баяндалышы}
                { Хан (Король) }
TKing =object(TChessMan)
        procedure Init (Cl:TColumns;  Rw: Trows; Bw:TColour);
        procedure Move (Cl:TColumns;  Rw: TRows);
        function IsMoveRight (Cl:TColumns; Rw:TRows):  Boolean;
        procedure Display;
    end;
```

```
                                { Жөөчү (Пешка) }
TPawn =object (TChessMan)
        procedure Init (Cl:TColumns; Rw:Trows; Bw: TColour);
        procedure Move (Cl:TColumns; Rw:TRows);
         function IsMoveRight (Cl:TColumns; Rw: TRows):  Boolean;
         procedure Display;
    end;

                              {усулдардын реализациясы}

                                 {TKing объекти}
procedure TKing.Init(Cl:TColumns; Rw:TRows; Bw:TColour);
 begin
       TChessMan.Init(Cl,Rw,Bw);
 end;

procedure TKing.Move (Cl:TColumns; Rw: TRows);
    begin
       if  IsMoveRight (Cl,Rw) then
            begin
                Clear;
                Column:=Cl;
                Row:=Rw;
                Display;
            end
             else ErrorMessage ('туура эмес жүрүш');
     end;

function TKing.IsMoveRight (Cl:TColumns; Rw:TRows):Boolean;
  var DeltaCl, DeltaRw :  Word;
  begin
    DeltaCl:=Abs (ord(Column)-Ord(Cl));
    DeltaRw:=Abs (Row-Rw);
    If (DeltaCl>1) or (Delta Rw>1) or
(DeltaCl=0) and (DeltaRw=0)
             then IsMoveRight := False
             else IsMoveRight := True
            end;

procedure TKing.Display;
   begin
      Put ('Kg');
   end;
```

```
                                {TPawn объекти}
procedure TPawn.Init (Cl: TColumns; Rw: TRows, BW: TColour);
     begin
       TchessMan.Init (Cl, Rw, Bw);
     end;

procedure Tpawn.Move (Cl: TColumns; Rw:TRows):
   begin
      if  IsMoveRight (Cl,Rw)  then
            begin
              Clear;
              Column: = Cl;
              Row: =Rw;
              Displey:
            end
        else   ErrorMessage (‘Туура эмес жүрүш ‘);
end;

function TPawn.IsMoveRight (Cl: TColumns; Rw: TRows): Boolcan;
    var
         DeltaCl, DeltaRw:  Integer;
     begin
     DeltaCl := Abs (Ord (Column)-ord(Cl));
     DeltaRw := Rw-Row;
     IsMoveRight := False;
     Case Colour of
     WhiteColour:  if (DeltaRw=1) and (Delta Cl<2)
       or (Row=2) and (Delta Cl=0)  and  (DeltaRw=2)
            then IsMoveRight := True;
   BlackColour: if (DeltaRw = -1) and (DeltaCl < 2)
      or (Row=7) and (DeltaCl = -2)
           then IsMoveRight:=True;
     end
end;

procedure TPawn.Display;
   begin
       Put (‘P’);
    end;
```

Бул объекттик типтерде төмөнкүдөй жаңы усулдар кошулган:

- фигураны жылдыруунун Move процедурасы;
- IsMoveRight функциясы. Ал аткарылуучу жүрүштүн конкреттүү фигураны жылдыруу эрежелерине тиешелештигин текшерет.
- Display процедурасы. Ал «өзүнүн» фигурасынын сүрөттөлүшүн экранга чыгаруу үчүн (король үчүн – ‘Kg’символун, жөөчү үчүн –‘P’символун) объекттик типтеги TChessMan процедурасынын аракетин басат.

Калган шахматтык фигуралар да ушуга эле окшош баяндалышат.

Turbo Pascal тилинин 7.0 версиясынын мурастоо механизмине таандык болгон айырмалуу өзгөчөлүк - бул жаңы **inherited** *резервделген сөзү болуп саналат.*

Бул сөз берилген объекттин түздөн-түз ата тегинин (предок) атынын ордуна пайдалануу үчүн арналган. Аны MS-DOS тогу “аталык каталог “ түшүнүгүнө окшош “аталык теги” деген псевдоат катары кароого болот.

Мисалы, TChessMan тиби TKing тибинин түздөн-түз ата теги экендигин эске алып TKing.Init усулун мындайча жазууга болот.

```
                    {TKing объекти}

procedure TKing.Init (Cl:TColumns; Rw: TRows; Bw:TColour);
 begin
   inherited Init (Cl, Rw, BW);
         {мурда бул жерде ThessMan.Init (Cl, Rw, BW);}
                        {жолчосу бар эле}
 end;
```

Бул жерде биз inherited резервделген сөзү ата тектерге ээ болбогон объекттик типтердин усулдарында пайдаланыла албастыгын байкайбыз.

Мурастоого тиешелүү болгон башкы аспект – бул ***мурасталуучу усулдарды чакыруу эрежелери*** болуп эсептелет:

1. Усулду чакырган учурда компилятор оболу аты объекттин тибинин ичинде аныкталган усулду издейт. **TChessMan** аталык тибинде аныкталгандай, **TKing** тибинде **Display** аттуу усул аныкталат. Качан **TKing.Move** усулунда **Display** усулу чакырылганда компилятор чакыруу командасына **TKing** үчүн өздүк болгон **Display** усулунун адресин коёт.

2. *Эгерде объекттин тибинде чакыруу операторунда көрсөтүлгөн атка ээ болгон усул аныкталбаган болсо, мисалы, **TKing.Display** баяндалбаган болсо, анда компилятор ушундай аттуу усулду издеп жогору карай түздөн-түз аталык типке көтөрүлөт.* Эгерде талаптагыдай атка ээ болгон усул табылган болсо, анда TKing.Move балалык тибинин баштапкы кодунда TKing.Display усулунун адресинин ордуна TChessMan.Display аталык усулунун адреси коюлат. Эгерде мындай ат менен усул табылбаган болсо, анда компилятор усулду издеп, аталык объекттер боюнча жогору көтөрүлүүнү уланта берет. Эгерде компилятор эң биринчи (жогорку) объект тибине жетип калса, ал эми талап кылынган атка ээ болгон усул табылбаган болсо, анда ал бир да мындай усул аныкталбагандыгын көрсөткөн ката жөнүндөгү билдирүүнү берет.

3. Эгерде мурасталуучу усул табылган болсо жана анын адреси коюлган болсо, анда төмөндөгүнү эске алуу керек, ***чакырылуучу усул балалык тип үчүн эмес, аталык тип үчүн кандай аныкталган жана компиляцияланган болсо, ошондой иштейт***. Эгерде ушул мурасталуучу усул дагы башка усулдарды чакыра турган болсо, анда эми аталык же андан жогору жаткан усулдар чакырылат, анткени ***типтердин иерархиясы боюнча төмөн жайланышкан типтерден усулдарды чыгарууга болбойт.***

Ушул айтылгандарды эске алып, бардык каралган объекттердин типтерин жана алардын реализацияларын өз ичине алган ChessMod усулу төмөнкү көрүнүштө болот:

```
unit ChessMod;
interface
type
        {Тактайдагы координаталардын типтери}
         TColumns = ’a’ . . ’h’;
         TRows  = 1 . . 8;
                  {Фигуралардын түсү}
         TColour={WhiteColour, BlackColour};
```

```
                {Тактайдагы позиция}
      TPosition=object
         private
              Column: TColumns;
              Row: TRows;
              procedure Init(Cl: TColumns;  Rw: TRows);
              procedure GetCoords(var X,Y: Word);
              procedure ErrorMessege (Mes: String);
          public
              function GetColumn: TColumns;
              function GetRow: TRows;
          end;
                        {Шахматтык фигура}
        TChesssMan=object (TPosition)
              private
              Colour: TColour;
              Present: Boolean;
              procedure Init (Cl:TColumns; Rw: TRows; BW: TColour);
              procedure Put (Name: String);
            public
              function GetColour: TColour;
              function IsPresent: Boolean;
              procedure  Clear;
              procedure Del;
              procedure Display;
       end;

                                  { Хан (Король)}
TKing=object (TChesssMan)
            procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
            procedure Move (Cl: TColumns; Rw: TRows);
            function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
            procedure Display;
      end;

                                  {Вазир (Ферзь)}
TQueen=object (TChessMan)
          procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
          procedure Move (Cl: TColumns; Rw: TRows);
          function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
           procedure Display;
      end;
```

```
                                { Түркүк (Ладья) }
  TCastle=object (TChessMan)
          procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
          procedure Move (Cl: TColumns; Rw: TRows);
          function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
          procedure Display;
     end;

                                { Пил (Слон) }
  TBishop=object (TChessMan)
          procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
          procedure Move (Cl: TColumns; Rw: TRows);
          function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
          procedure Display;
     end;

                                { Ат (Конь) }
  TKnight=object (TChessMan)
          procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
          procedure Move (Cl: TColumns; Rw: TRows);
          function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
          procedure Display;
     end;

                                {Жөөчү (пешка)}
  TPawn=object (TChessMan)
          procedure Init (Cl:TColumns; Rw: TRows;  BW:TColour);
          procedure Move (Cl: TColumns; Rw: TRows);
          function IsMoveRight (Cl:TColumns; Rw: TRows):Boolean;
          procedure Display;
     end;

var
              {шахматтык фигуралардын өзгөрүлмөлөрү}

   WKg, BKg: TKing;                    { Хандар }
   Wa,  Ba:TQueen;                     { Ханышалар }
   WC1, WC2, BC1, BC2: TCastle;        { Түркүктөр }
   WB1, WB2, BB1, BB2:  TBishop;       { Пилдер }
   WK1,WK2, BK1, BK2: TKnight;         { Аттар }
   WP1, WP2, WP3,WP4, WP5, WP6, WP7, WP8,
   BP1, BP2, BP3, BP4, BP5, BP6, BP7, BP8: TDawn; { Жөөчүлөр }
```

```
Implementation

                    {Усулдардын реализациясы}

uses Crt, Dos;

                          {TPosition объекти}

procedure TPosition.Init (Cl:  TColumns; Rw:  TRows);
    begin
       Column := Cl;
       Row := Rw;
    end;

function TPosition.GetColumn:  TColumns;
        begin
              GetColumn : = Column;
        end;
        function TPosition.GetRow :  TRows;
         begin
             GetRow:=Row;
         end;

       procedure TPosition.GetCoords (var X,Y: Word);
       const
             BegX = 20;   BegY = 23;
             StepX = 7;    StepY = -3;
       begin
             X:= BegX+StepX*(Ord(Colunm)-Ord ('a'));
             Y:=BegY+StepY*(Row-1);
       end;

       procedure TPosition.ErrorMessage (Mes:  String);
         const
                BegY = 23;
       begin
             CotoXY (1, BegY);
             Text BackGround (Blue);
             TextColor ( LightRed + Blink );
             Write(Mes);
        end;
```

```
                                {TChessMan обекти}
procedure TChessMan.Init (Cl:TColumns; Rw: TRows;
                                                        BW: TColour);
      begin
            Inherited Init (Cl, Rw);
            Colour:=BW;
             Present := True;
      end;

  function TChessMan.GetColour: TColour;
    begin
       GetColour:=Colour
    end;

  function TChessMan.IsPresent: Boolean;
     begin
     IsPresent:=Present;
     end;

  procedure TChessMan.Clear;
     begin
            Put (' ');
            ErrorMessage('                  ');
      end;

  procedure TChessMan.Del;
       begin
            Clear;
            Present:=False;
        end;

  procedure TChessMan.Display;
        begin
            Put( 'CM');
        end;

  procedure TChessMan.Put( Name: String);
   var
         X,Y:Word;
        begin
            GetCoords( X,  Y );
            GotoXY (X, Y);
```

```
        if (Row+Ord(Column)) mod 2 =0  then
                  TextBackGround(Black)
            else TextBackGround(White);
         case Colour of
               WhiteColour: TextColor (Light Red);
               BlackColour: TextColor (Light Green);
         end;
         Write (Name);
       end;

                                        {TKing  объекти}
  procedure TKing.Init (Cl: TColmns;  Rw: TRows; BW: TColour);
       begin
             inherited Init (Cl, Rw, BW);
       end;

  procedure TKing.Move (Cl: TColnmns; Rw: TRows);
        begin
             if   IsMoveRight (Cl, Rw)    then
              begin
                Clear;
                Column:=Cl;
                Row:=Rw;
                Display;
              end
              else  ErrorMessage ('Туура эмес жүрүш!');
        end;
 function TKing.IsMoveRight (Cl: TColumns; Rw: TRows): Boolean;
    var  DeltaCl, DeltaRw: Word;
        begin
             DeltaCl := Abs (Ord(Column)-Ord(Cl));
             DeltaRw := Abs (Row-Rw);
                 if (DeltaCl >1) or (DeltaRw >1) or
                    (Delta Cl =0) and (DeltaRw=0)
                      then IsMoveRight : =  False
                   else IsMoveRight : =  True
        end;

  procedure  TKing.Display;
       begin
          Put ('Kg');
       end;
```

```
                                {TPawn объекти}
  procedure TPawn.Init(Cl: TColumns; Rw: TRows; BW: TColour);
      begin
         Inherited Init(Cl, Rw, BW):
      end;
  procedure TPawn.Move (Cl: TColumns;  Rw : TRows);
       begin
           if  IsMoveRight (Cl, Rw)  then
                begin
                 Clear;
                 Column: =Cl;
                 Row: =Rw;
                 Display;
                end
     else   ErrorMessage ('Туура эмес жүрүш!');
    end;

function TPawn.IsMoveRight (Cl: TColumns;
                                          Rw: TRows): Boolean;

   var  DeltaCl, DeltaRw: integer;
        begin
            DeltaCl := Abs (Ord(Column)-Ord(Cl));
            DeltaRw := Rw-Row;
            IsMoveRight:= False;
           case Colour of
     WhiteColour:   if (DeltaRw = -1) and  (DeltaCl < 2) or
                    (Row = 7) and (DeltaCl = 0) and (DeltaRw = -2)
                    then IsMoveRight : =  True;

   BlackColour: if (DeltaRw=1) and  (DeltaCl <2) or
                    (Row=2) and (DeltaCl=0) and (DeltaRw=2)
                    then IsMoveRight : =  True;
          end
  end;

procedure  TPawn.Display;
       begin
          Put ('P');
       end;
```

```
                                {TPawn объекти}
  procedure TQueen.Init(Cl: TColumns; Rw: TRows; BW: TColour);
      begin
         Inherited Init(Cl, Rw, BW):
      end;
  procedure TQueen.Move (Cl: TColumns;  Rw : TRows);
       begin
          if  IsMoveRight (Cl, Rw)  then
               begin
                Clear;
                Column: =Cl;
                Row: =Rw;
                Display;
               end
            else   ErrorMessage ('Туура эмес жүрүш!');
       end;
   function TQueen.IsMoveRight (Cl: TColumns;
                                          Rw: TRows): Boolean;
       begin
         IsMoveRight := True;
       end;
   procedure  TQueen.Display;
       begin
         Put ('Q');
      end;

                                  {TCastle объекти}
procedure TCastle.Init(Cl:TColumns: RW:  TRows; BW:  TColour);
   begin
        inherited Init(Cl, Rw, BW);
   end;

procedure TCastle.Move (Cl:  TColums;  Rw:   TRows);
   begin
     if   IsMoveRight ( Cl, Rw)  then
       begin
         Clear;
         Column:=Cl;  Row:=Rw;
         Display;
       end
     else  ErrorMassage ('Туура эмес жүрүш!');
   end;
```

```
function TCastle.IsMoveRight (Cl: Columns; Rw: TRows): Boolean;
    begin
       IsMoveRight:=True;
    end;

   procedure TCastle.Display;
        begin
           Put('C');
        end;

                                    {TBishop объекти}
   procedure TBishop.Init(Cl: TColumns; Rw: TRows; Bw:TColour);
        begin
             inherited Init (Cl, Rw, BW);
        end;

procedure TBishop.Move(Cl: TColumns;  Rw:  TRows);
  begin
         if IsMoveRight (Cl, Rw)   then
            begin
              Clear;
              Column := Cl;
              Row:=Rw;
              Display;
             end
          else  ErrorMassage ('Туура эмес жүрүш!');
   end;

  function TBishop.IsMoveRight (Cl:Tcolumns;  Rw:
                                                  TRows):Boolean;
      begin
           IsMoveRight:=True;
      end;

  procedure TBishop.Display;
      begin;
          Put('B');
      end;

                                    {TKnight объекти}
  procedure TKnight.Init(Cl: TColumns;  Rw:  TRows; BW: Tcolour);
      begin
          inherited Init (Cl, Rw, BW);
       end;
```

```
    procedure TKnight.Move (Cl: Tcolumns;  Rw:  TRows);
    begin
       Clear;
       Column:=cl;
       Row:=Rw;
       Display;
    end
         else Error Message (‘Туура эмес журуш!’);
    end;

function TKnight.IsMoveRight (Cl: Columns; Rw: TRows):Boolean;
  begin
     IsMoveRight:=True;
  end;

procedure TKnight.Display;
   begin
      Put ( ‘ K ’ );
   end;

procedure BoardPicture;
 Const
      BegX = 20;
      BegY = 24;
      StepX = 7;
      StepY = 3;
var BX, BY:  Byte;
          i , j :  Byte;
   procedure Cell ( X, Y,  BW: Byte);
     var j: Byte;
     beign
        Text BackGround ( BW );
        for j := Y + StepY  do
        begin
          GotoXY ( X, Y);
           Write( ‘             ‘ );
        end;
    end;
begin
      for j := 8 dounto 1  do
  begin
        for I : =1 to 8  do
        if (i+j)  mod  2 = 0  then
```

```
            Cell (BegX – 3 + (i -1)* StepX, BegnY-j*StepY+1, Blask)
        else Cell(BegX - 3+ (i-1)*StepX, BegY - j*StepY +1, White);
 end;
 GotoXY (BegX - 3, BegY+1);
 TextBaskGround (Blue);
 TextColor (Yellow);
 Write(‘  a   b   c   d   e   f   j   h  ’ );
  for j := 8 dounto 1  do
       begin
         GotoXY (BegX – 5, BegnY - j*StepY+2);
         Write( j : 1 );
       end;
 end;
procedure SetCursorSize (BegLine,  EndLine: Byte);

                                  {Курсорду коюу}
  var Regs: Regrsters;
  begin
      with Regs do
      begin
         AH := $01;
         CH := BegLine;
         CL :=  EndLine;
      end;
       Intr ($10,  Regs);
  end;
 procedure HideCursor;
                            {Курсорду бекитүү}
var BegLine, EndLine: Byte
   begin
   BegLine: = $20; {$20 - баштапкы сызыктын маани си катары}
                        {курсорду көрүнбөс кылат                    }
    EndLine: = $00;
    SetCursorSize (BegLine, EndLine);
   end;
 begin
  TextBackGround (Blue);
  ClrScr;
  HideCursor;   { Курсорду бекитүү }
  BoardPicture;
 end.
```

Шахматтык аракеттерди монитордун экранындагы сүрөттөлүшкө «байлоо» (привязка) төмөнкү камтылуучу программалардын жардамында аткарыларына көңүл бургула:

- TPozition объекттик тибинин жогоруда каралган GetCoords жана ErrorMessage усулдары жана TChessMan тибинин Put усулу менен;
- Объекттердин иерархиясынан көз каранды болбогон BoardPicture (экранда, тексттик режимде шахмат тактайчасынын жөнөкөйлөтүлгөн көрүнүшүн чиет), SetCursorSize жана HideCursor (экранга курсорду чыгарууну башкаруу) процедуралары менен.

Дагы бир белгилеп кетчү нерсе, бул келтирилген модулда Ханды жана Жөөчүнү жылдыруу эрежелери толук реализацияланган эмес. IsMoveRight функциясы жогоруда көрсөтүлгөн көрүнүштө Хандын жана Жөөчүнүн жүрүшүн аткаруу эрежелерине ылайык талап кылынган аракеттин аткарылыш мүмкүндүгүн гана текшерип, тактайдагы башка фигуралардын өз ара жайланышын эсепке албайт.

Эки Ханды баштапкы абалдарга коюуну жана алардын ар биринин бирден жүрүш аткарышын ишке ашыруучу ChessMode модулуну пайдалануучу төмөндөгүдөй элементардык программаны келтиребиз. Мында Readln процедурасы жүрүштөрдүн аткарылышын көрүү үчүн кошулган.

```
program Chess;
uses  ChessMod;
 begin
        Readln;
        WKg.Init ( ‘e’, 1, WhiteColour); WKg.Display;
        Readln;
        BKg.Init ( ‘e’, 8, BlackColour); BKg. Display;
        Readln;
        WKg.Move ( ‘e’, 2 );
        Readln;
        BKg.Move ( ‘e’, 8 );
        Readln;
  end.
```

Мурастоо эрежелерин реализациялоодогу бир кылдаттыкты (тонкость) карайбыз.

Бардык шахматтык фигуралардын баяндоосунда Move усулу TKing тибиндеги Move усулуна толук окшош (идентичный)

экендигин байкоо кыйын эмес, анткени анда жеке Хандын конкреттүү өзгөчөлүктөрү жок. Ошондуктан текстти кыскартуу жана кайталануучу усулду бир денгээл жогору турган TChessMan тибине чыгаруу керек деген табигый ой пайда болот.

Move усулунда IsMoveRight усулу чакырылгандыктан, TChessMan тибинде аны да баяндоо керек болот.

```
                                {Шахматтык фигура}
TChessMan = object (TPozition)
  private
        Colour:  TColour;
        Present: Boolean;
        procedure Init (Cl: TColumns; Rw: TRows; BW: TColour);
        procedure Put (Name: String);
  public
       function GetColour: TColour;
       function IsPresent: Boolean;
       procedure Clear;
       procedure Del;
       procedure Display;
       function  IsMoveRight (Cl: TColumns; Rw: TRows): Boolean;
       procedure Move (Cl: TColumns; Rw: TRows);
  end;

                                    { Хан (Король) }
 TKing = object (TChessMan)
       procedure  Init (Cl: TColumns; Rw: TRows; BW: TColour);
       function IsMoveRight (Cl: TColumns; Rw: TRows): Boolean;
       procedure Display;
 end;
```

Мурастоо эрежеси боюнча Move усулу, ар бир жекече фигуранын типтеринде баяндалган болгон учурдагыдай чакырылышы керек. Биз программада ушундай модификацияны жасагандан кийин, ал чындыгында эле фигураларды корректүү (корректно) жылдырып калат, бирок жаңы позицияда ак да, кара да Хандын аттарынын ордуна TChessMan.Display усулундагы экранга чыгаруу операторунда жазылган, абстракттык шахматтык фигурага тиешелеш келүүчү «СМ» символу пайда болот.

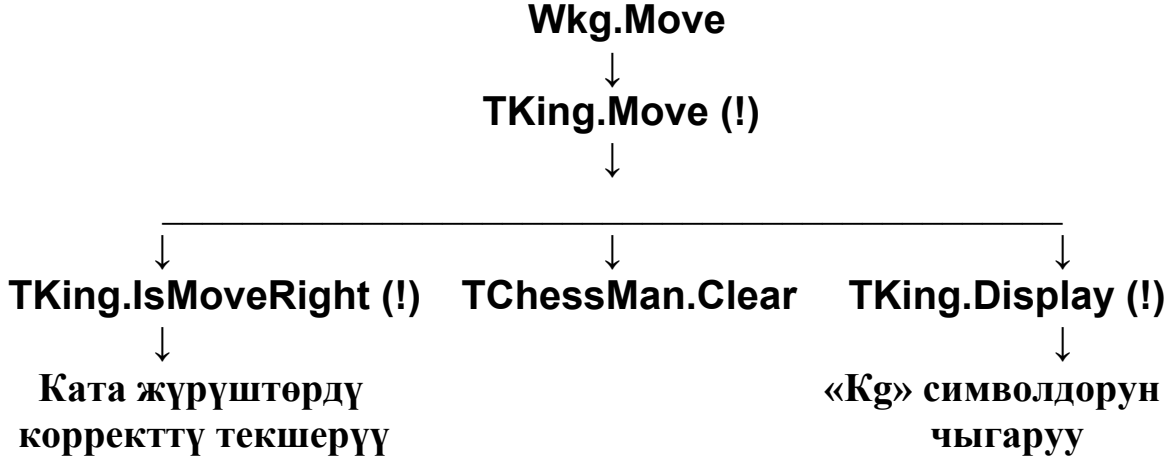
Мунун себеби, биз буга чейин ***статикалык усулдар*** деп аталган усулдарды гана пайдаландык. Мындай усулдардын адрестери объекттин нускалары менен (объекттик типтеги өзгөрүлмөлөр менен)

компиляция мезгилинде эле статикалык түрдө байланышып, программанын иши аяктаганча өзгөрбөйт.

Биринчи учурда, качан фигуралардын ар бири өзүнүн Move усулуна ээ болгон учурда, TKing тибиндеги объекттин нускасынын (экземплярынын) структурасы төмөндөгүдөй көрүнүшкө ээ болгон эле:

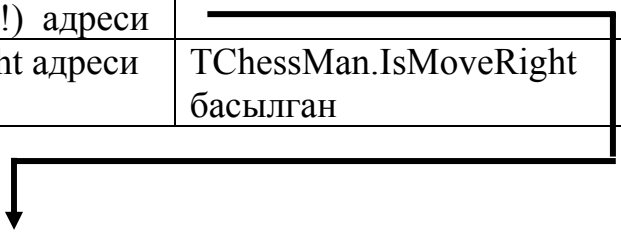
	WKg объекти	
Column	‘e’	
Row	1	
Colour	WhiteColour	
Present	True	
GetColumn	TPosition. CetColumn адреси	мурасталат
GetRow	TRosition.CetRow адреси	мурасталат
GetCoords	TPosition. CetCoords адреси	мурасталат
Error Message	TPosition.ErrorMessage адреси	мурасталат
Put	TChessMan.Put адреси	мурасталат
GetColour	TChessMan.GetColour адреси	мурасталат
IsPresent	TChessMan.IsPresent адреси	Мурасталат
Clear	TChessMan.Clear адреси	мурасталат
Del	TChessMan.Del адреси	мурасталат
Init	TKing.Init адреси	TPosition.Init, TChessMan.Init басылган (подавлены)
Display	TKing.Display адреси	TChessMan.Display басылган
Move	TKing.Move адреси	TChessMan. Move баяндалган эмес
IsMoveRight	TKing. IsMoveRight адреси	TChessMan.IsMoveRight баяндалган эмес

Бул учурда чакыруулардын төмөнкү корректүү чынжырчасы иштейт.



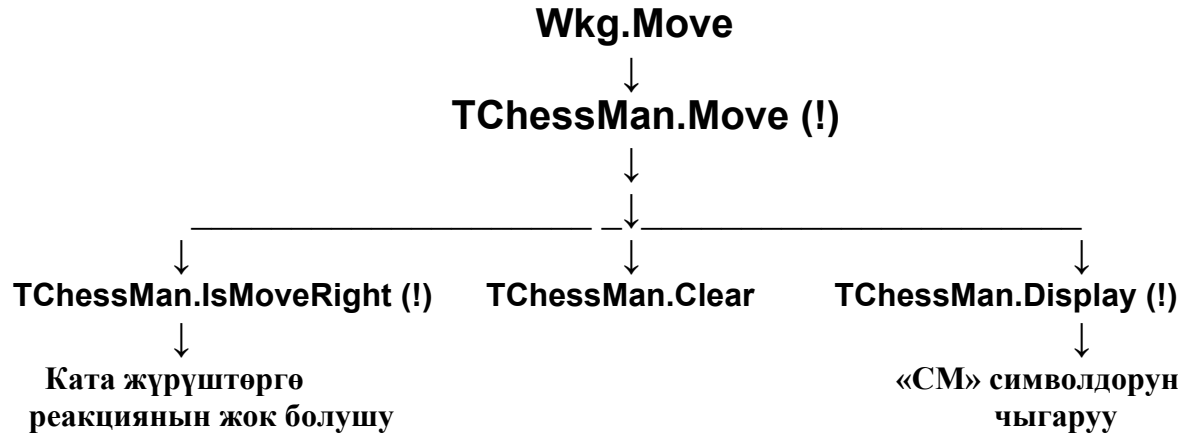
Экинчи учурда, качан бардык фигуралар, алардын TChessMan аталык тибинде баяндалган жалпы Move усулуна ээ болгон учурда TKing объекттик тибинин реализациясынын структурасы жогоркудан бир канча айырмаланат.

	WKg объекти	
Column	‘e’	
Row	1	
Colour	WhiteColour	
Present	True	
GetColumn	TPosition. CetColumn адреси	мурасталат
GetRow	TRosition.CetRow адреси	мурасталат
GetCoords	TPosition. CetCoords адреси	мурасталат
Error Message	TPosition.ErrorMessage адреси	мурасталат
Put	TChessMan.Put адреси	мурасталат
GetColour	TChessMan.GetColour адреси	мурасталат
IsPresent	TChessMan.IsPresent адреси	Мурасталат
Clear	TChessMan.Clear адреси	мурастадат
Del	TChessMan.Del адреси	мурасталат
Init	TKing.Init адреси	TPosition.Init, TChessMan.Init басылган (подавлены)
Display	TKing.Display адреси	TChessMan.Display басылган
Move	TChessMan.Move(!) адреси	
IsMoveRight	TKing. IsMoveRight адреси	TChessMan.IsMoveRight басылган



Мурасталат, бирок анын телосунда TKing.Display жана TKing.IsMoveRight эмес, TChessMan.Display жана TChessMan.IsMoveRight чакырылат.

Бул учурда чакыруулардын төмөндөгү чынжырчасы аткарылат



Көрүнүп тургандай, IsMoveRight жана Display усулдарын, керек болгон типтен эмес - TKing тибинин ордуна TChessMan тибинен чакыруу жүргүзүлөт.

Бул абалдан чыгуу болуп – Move усулун жалпы кылуу жана бардык нерсе туура иштеши үчүн полиморфиялык аракеттерди (усулдарды) пайдалануу эсептелет.

Полиморфиялык аракет – бул бардык объекттер үчүн бирдей атка ээ болгон (жаныбарлар жөнүндөгү жогоруда каралган мисалда - «чуркоо» аракети), бирок ар бир объект тарабынан өз бетинче (по-своему) (арстан тарабынан, пил тарабынан, крокодил ж.б. тарабынан) аткарылуучу аракет экендигин эске салабыз. Шахматтын учурунда ар бир фигура тарабынан өз эрежеси боюнча аткарылуучу «фигуранын жүрүшү» деген бирдей аракетке ээ болобуз. Жогоруда каралган абалда бир караганда баары туура жасалгандай көрүнөт – Move жалпы аракети ага бардык фигуралар кайрыла алсын үчүн аталык типке чыгарылган, бирок ал аракет полиморфиялык болбой калды.

Мындай жыйынтыктын себеби мына мында: статикалык усулдардын жардамында полиморфиялык аракеттерди реализациялоо мүмкүн эмес, анткени усулдардын (процедуралардын, функциялардын) адрестерин объекттердин нускаларына байлоо (привязка) статикалык түрдө болуп өтөт.

Биринчи учурда Tking.Move усулу TKing.Display жана TKing.IsMoveRight корректтүү усулдары менен статикалык байланышкан, анткени алар бир эле TKing тибинде баяндалышкан. Экинчи учурда мурастоо эрежелери боюнча чакырылган TChessMan.Move усулу TChessMan тибинде жарыялоо каражаты

аркылуу TChessMan.Display жана TChessMan.IsMoveRight усулдары менен статикалык байланышкан.

Ошентип программалоо тилинде полиморфизмди реализациялоо үчүн объекттердин нускаларын усулдар менен статикалык эмес, полиморфиялык аракеттерди ишке ашыруу үчүн динамикалык (программа аткарылып жаткан учурда) байланыштыруу талап кылынат.

3.6. Виртуалдык жана динамикалык усулдар

Turbo Pascal тилинде объекттердин нускалары менен динамикалык байланышуучу усулдардын эки түрү бар. Биринчи түрдөгү усулдар ***виртуалдык***, ал эми экинчи түрдөгү – ***динамикалык*** усулдар деп аталат. Эгерде виртуалдык усулдар тилдин мурдагы версияларындагы объекттик-ориентирленген каражаттар аркылуу белгилүү болсо, динамикалык усулдар Turbo Pascal тилинин 7.0 версиясындагы жаңы киргизилген каражат. Алардын ортосундагы негизги айырмачылык – аларды реализациялоо үчүн компилятор ар түрдүү структурадагы ички таблицаны пайдаланат: Виртуалдык усулдар үчүн – ВУТ (Виртуалдык Усулдар Таблицасы), ал эми динамикалык усулдар үчүн – ДУТ (Динамикалык Усулдар Таблицасы). Динамикалык байланыштыруу принциби болсо аларда жалпы болот.

Текст боюнча мындан ары калган баяндалуучу аракеттер виртуалдык да, динамикалык да усулдарга таандык болгон учурларда кыскалык үчүн «виртуалдык» деген терминди пайдаланабыз.

Объекттик ориентирленген терминологиянын дагы эки терминине аныктама беребиз.

Эрте байланыш деп – усулдар менен объекттердин реализацияларынын статикалык байланышуу процесси аталат.

Кеч байланыш деп – усулдар менен объекттердин реализацияларынын динамикалык байланышуу процесси аталат.

Виртуалдык жана динамикалык усулдардын баяндалыштары синтаксиси боюнча статикалык усулдардын баяндалышынан, объекттик типтерди баяндоо учурунда усулдардын бөрктөрүнүн аягында көрсөтүлүүчү virtual директивасынын турушу менен айырмаланат. Усулдун реализациясын баяндоодо бул директива коюлбайт.

Андан сырткары, эгерде кандайдыр бир аталык типте виртуалдык усул баяндалса, анда ал анын балалык типтерине төмөндөгүдөй чектөөлөрдү коет:

- Балалык типтердин аталык виртуалдык усулдар менен бир аттуу (одноименные) бардык усулдары дагы милдеттүү түрдө виртуалдык болушу керек (статикалык усул эч качан виртуалдык усулду басып кое (подавить) албайт).
- Усул виртуалдык болуп калгандан кийин, иерархиянын төмөнкү деңгээлиндеги объекттерде анын бөркү өзгөрүлө албайт. ***Бир эле виртуалдык усулдун бардык реализацияларынын бөрктөрү, параметрлердин саны жана алардын типтери менен кошо толук окшош (идентичные) болушу керек.*** (Бул чектөө статикалык усулдарга тиешеси жок экендигин байкоого болот: башка усулду кайрадан аныктоочу (берүүчү) статикалык усул параметрлердин башкача санына жана типтерине ээ болушу мүмкүн).
- Виртуалдык усулдарга ээ болгон ар бир объекттик тип милдеттүү түрдө конструкторго ээ болушу керек.

3.6.1. Конструкторлор

Конструктор – бул виртуалдык усулдардын иштөө механизми үчүн баштапкы орномолорду (установки) аткаруучу процедуранын атайын тиби болуп саналат. Мына мындай бир маанилүү учурду байкоого болот: ***кандайдыр бир объекттин каалаган виртуалдык усулун чакырууда бул объекттин конструкторун чакыруу зарыл.*** Алдын ала конструкторду чакырбай туруп виртуалдык усулду чакыруу системанын таптакыр иштебей калышына (блокировка) алып келиши мүмкүн, ал эми компиляция деңгээлинде динамикалык байланылуучу усулдарды чакыруунун корректтүүлүгүн (тууралыгын) текшерүү мүмкүн эмес. ***Объекттин ар бир жекече нускасы конструктордун өзүнчө (жекече) чакырылышы менен инициализацияланган болушу керек.*** Ыйгаруу оператору объекттердин маанисин көчүрүп гана коёт, бирок инициализациялоону аткарбайт. Ошондуктан, эгер объекттин бир нускасын инициализациялап, анан бул нусканы башка объектке ыйгарсак, анда экинчи объекттин виртуалдык усулдарын чакырууда система токтоп (блокировкаланып) калат.

Объекттердин конструкторлорунун аттары катары, инициализациялоо процедураларынын аттары сыяктуу эле init идентификаторун пайдалануу сунуш кылынарын белгилеп кетебиз.

Ошентип, эгер бизге TChessMan.Move, TChessMan.Displey, TChessMan.IsMoveRight усулдарын виртуалдык кылуу талап кылынса, анда Init инициализациалоо процедурасы конструкторго өзгөртүлүп түзүлгөн болушу керек, ошондон кийин TChessMan жана TKing типтеринин баяндалышы төмөнкү көрүнүштө болот:

```
                    { Шахматтык фигура }

  TChessMan=obgect (TPosition)
      private
          Colour:  TColour;
          Present: Boolean;
   constructor Init (Cl: TColumns; Rw:  TRows;  BW: TColour);
   procedure Put (Name:  String);
      public
          function GetColour: TColour;
          function IsPresent:  Boolean;
          procedure Clear;
          procedure Del;
{Display, IsMoveRight, Move усулдары виртуалдык кылынган!!!}
          procedure Display;                          virtual;
   function IsMoveRight(Cl; TColumns;
                          Rw; TRows): Boolean;       virtual;
   procedure Move (Cl; TColumns;  Rw;  TRows);       virtual;
end;
                          { Хан (Король) }

 TKing = object (TChessMan)
      constructor Init (Cl: TColumns; Rw:  TRows;  BW: TColour);

                  {Move   усулу жок кылынгын !!!}

      function IsMoveRight (Cl: TColumns;
                          Rw: TRows): Boolean;       virtual;
      procedure Display;                             virtual;
 end;
```

Көрүнүп тургандай Move усулу бул учурда жалпы болуп калып TChessMan тибинде аныкталган, ал эми ар бир фигура үчүн индивидуалдык болгон жүрүштүн корректтүүлүгүн текшерүүчү IsMoveRidht жана өзүнүн экрандагы сүрөтөлүшүн чыгаруучу Display усулдары, TchessMan тиби менен катар катасы бар мисалда

кылынгандай фигуралардын типтеринде (айрым учур катары TKing тибинде) баяндалган. Бирок, азыр Move.Display жана IsMoveRight усулдары виртуалдык усулдар катары баяндалып, натыйжада объекттер менен иштөөдө статикалык (эрте) эмес, динамикалык (кеч) байланышуу аткарылат.

Виртуалдык усулдарды пайдаланган учурда TKing тибиндеги объекттин нускасынын структурасы баштапкы абалда төмөнкүдөй көрүнүшкө ээ болот:

	WKg объекти	
Column		
Row		
Colour		
Present		
GetColumn	TPosition. CetColumn адреси	Мурасталат
GetRow	TRosition.CetRow адреси	Мурасталат
GetCoords	TPosition. CetCoords адреси	Мурасталат
Error Message	TPosition.ErrorMessage адреси	Мурасталат
Put	TChessMan.Put адреси	Мурасталат
GetColour	TChessMan.GetColour адреси	Мурасталат
IsPresent	TChessMan.IsPresent адреси	Мурасталат
Clear	TChessMan.Clear адреси	Мурастадат
Del	TChessMan.Del адреси	Мурасталат
Init	TKing.Init адреси	→ Консруктор
Display	TKing.Display адреси	
Move	TKing.Move(!) адреси	→ Виртуалдык
IsMoveRight	TKing. IsMoveRight адреси	усулдар

Статикалык усулдар

Байкай турган нерсе, виртуалдык усулдар үчүн статикалык түрдө адрестер үчүн гана эс бөлүнөт, ал эми адрестер өздөрү TKing.Init конструкторун чакырган учурда кеч байланыштын орнолуш процессинде ячейкаларга киргизилет.

Мына бул оператор аткарылгандан кийин

```
WKg.Init( ‘e’, 1, WhiteColour);
```

объекттин нускасынын структурасы төмөндөгүдөй көрүнүшкө ээ болот:

	WKg объекти	
Column	‘e’	
Row	1	
Colour	WhiteColour	
Present	True	
GetColumn	TPosition. CetColumn адреси	Мурасталат
GetRow	TRosition.CetRow адреси	Мурасталат
GetCoords	TPosition. CetCoords адреси	Мурасталат
Error Message	TPosition.ErrorMessage адреси	Мурасталат
Put	TChessMan.Put адреси	Мурасталат
GetColour	TChessMan.GetColour адреси	Мурасталат
IsPresent	TChessMan.IsPresent адреси	Мурасталат
Clear	TChessMan.Clear адреси	Мурастадат
Del	TChessMan.Del адреси	Мурасталат
Init	TKing.Init адреси	Консруктор
Display	TKing.Display адреси	
Move	TKing.Move(!) адреси	
IsMoveRight	TKing. IsMoveRight адреси	

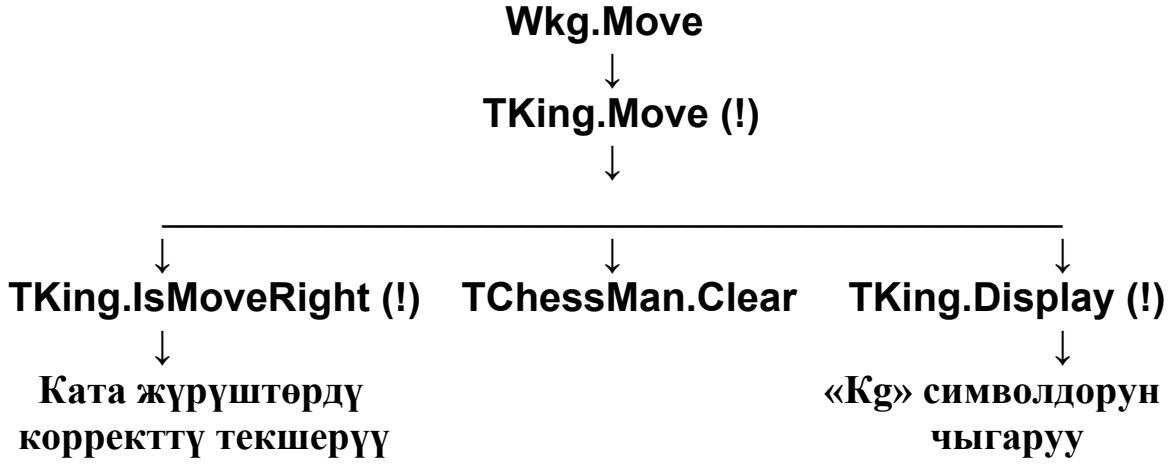
Статикалык усулдар

Виртуалдык усулдар

Сүрөттөн көрүнүп тургандай, объекттин нускасында жогоруда каралган катасы бар учурдагыдай эле, ошол эле TChessMan.Move, TKing.Display жана TKing.IsMoveRight усулдарынын адрестери турат.

Бирок, мында Move, Display *жана* IsMoveRight *усулдарынын виртуалдыгынан улам, кеч байланыштын натыйжасында* TChessMan.Move *усулу* Display *жана* IsMoveRight *усулдарын эрте байланыш учурундагыдай* TChessMan *тибинен эмес* TKing *тибинен болгон чакырууга автоматтык түрдө тууралanат (настраивается).*

Ошентип, усулдарды чакыруунун төмөнкү чынжырчасы эң биринчи каралган учурдагыдай кайрадан тура болот.



Бирок эми, биринчи учурдагыдан айырмаланып биздин программада, объекттик-ориентирленген программалоонун бир кыйла эффективдүү каражаттарынын бири болгон - полиморфизмди аракетке келтирүүчү виртуалдык усулдар каралат.

3.6.2. Виртуалдык жана динамикалык усулдардын айырмасы

Жогоруда айтылып кеткендей динамикалык усулдар Turbo Pascal 7.0 тилинде жаңы киргизилген каражат болуп ал виртуалдык усулдан, негизинен, компилиятордун ички таблицаларынын структурасы менен айырмаланат.

Тагыраак айта турган болсок, динамикалык усулдар - бул DOS то иштөөчү платформалар үчүн жаңы киргизилген нерсе, ал эми Windows платформасы үчүн андай эмес, анткени Turbo Pascal ***for*** Windows чөйрөсүндө ал эбак пайдаланылган.

Динамикалык усулдар виртуалдык усулдардан аткарылыш этабындагы диспетчерлөө ыкмасы менен айырмаланат. Динамикалык усулдар үчүн компилиятор виртуалдык усулдар таблицасынын (ВУТ) ордуна динамикалык усулдар таблицасын (ДУТ) түзөт, ДУТ - сын колдонуу объекттер менен иштөөдө колдонмо программа тарабынан пайдаланылуучу эстин өлчөмүн азайтат. Бирок, бул учурда программа жайыраак иштеп калат. Калган баардык учурларда динамикалык усулдар виртуалдык усулдар менен эквивалентүү деп эсептелет.

Синтаксиси боюнча виртуалдык усулдардан айырмаланып динамикалык усулдар үчүн алардын бөрктөрүндө түздөн-түз virtual разерделген сөзүнөн кийин жайгашуучу динамикалык усулдун индекси колдонулат. Диамикалык усулдун индекси 1 ден 656535 ке чейинки диапозондогу бүтүн сандык константа болуш керек, андан сырткары, берилген объекттик типте жана/же анын аталык типтеринде баяндалган башка динамикалык усулдардын индексинин арасында кайталанбашы керек.

Динамикалык усулдун бөркүн баяндоонун **мисалы**:

```
procedure Move (CL:TColumns ;  Rw: TRows);  virtual 77;
```

Балалык типте динамикалык усулду кайра баяндоодо анын бөркү, виртуалдык усулдардагыдай эле, атайын типтеги бөрк менен (индекс менен кошо) толук окшош болушу керек.

3.7. Динамикалык объекттер

Эң оболу баса белгилей турган нерсе, *динамикалык усулдар менен динамикалык объекттерди чаташтырбоо керек!* Динамикалык усулдар динамикалык да, статикалык да объектте катышып турган болушу мүмкүн. Буга чейин каралган мисалдарда объекттердин бардык нускалары статикалык эле, анткени объекттик типтеги WKg, Bkg өзгөрүлмөлөрү var бөлүгүндө статикалык болуп баяндалган жана алар үчүн эс программанын ишинин башында бир ирет бөлүнүп коюлган.

Turbo Pascal тилинин каражаттары объекттик типтердин динамикалык нускаларын да баяндоого мүмкүнчүлүк берет. Алар ар кандай башка типтердеги каалагандай динамикалык өзгөрүлмөлөргө окшош var бөлүгүндө аныкталышат, ал эми алар үчүн эс программанын ишинин башында бир ирет бөлүнүп коюлат.

Ошентип, мына бул статикалык объекттердин

```
var
    WKg,  BKg: TKing;
```

ордуна бирдей ийгиликте төмөнкү динамикалык объекттерди баяндоого жана пайдаланууга болот

```
var
    WKg,  BKg:  ^TKing;
```

Динамикалык объекттерди пайдаланган учурда New жана Dispose процедураларынын кенейтирилген синтаксиси менен иштөө ынгайлуу.

Синтаксистин биринчи кенейтилиши New - ну процедура катары эле эмес функция катары да чакырууга уруксут берилет дегендикти билдирет. Мисалы, төмөнкү жазуу корректүү болот.

```
        . . .
type
   PTKing = ^TKing;
var  PWkg, PBKing: PTKing;
```

```
begin
    PWKg := New(PTKing);
    PBKg  := New(PTKing);
  .  .  .
end.
```

New процедурасынын синтаксисинин экинчи кенейтилишине ылайык анын жазылышында экинчи параметрди пайдаланууга уруксат берилет. Бул параметр New процедурасында, динамикалык объект үчүн эстин областы бөлүнөрү менен ал динамикалык объектти инициализациялоочу конструктордун атын иш жүзүндөгү параметлери менен кошо көрсөтүү үчүн кызмат кылат.

Мисалы, Ханды инициялизациялоо жана аларды экранга чыгаруу үчүн төмөнкү фрагменттин ордуна

```
New (PWKg);
PWKg^.Init( ‘e’ , 1, WhiteColour));
PWKg^.Display;
New(PBKg);
PBKg^.Init( ‘e’ ,8 , BlackColour));
PBKg^.Dispay;
```

төмөндөгүдөй операторлорду жазууга болот

```
New (PWKg, Init( ‘e’ , 1, WhiteColour));
PWKg^.Display;
New (PBKg, Init( ‘e’ ,8 , BlackColour));
PBKg^.Dispay;
```

Dispose процедурасынын кенейтилиши New процедурасынын экинчи кенейтилишине окшош. Айырмасы – экинчи параметр болуп конструктордун эмес ***деструктордун*** аты келет.

```
Dispose (PWKg, Done) ;
Dispose (PBKg, Done) ;
```

3.7.1. Деструкторлор

Деструктор *– бул да конструктор сыяктуу эле усулдун атайын бир көрүнүшү болуп*, башкача дагы аны “Таштанды жыйноочу” (“Сборшик мусора”) деп аташат.

Деструкторлор procedure сөзүнүн ордуна destructor кызматчы сөзүн пайдалануу менен баяндалып динамикалык жайланышкан объекттерди жоготуу (өчүрүү) жана алар ээлеген эсти бошотуу үчүн

арналган. Эреже катары, деструктор эсти бошотууда деструкторду бир жолу чакыруу менен тазалоону аткарууга мүмкүн боло тургандай кылып бардык аракеттерди бириктирет. Ар бир объекттик тип үчүн зарыл болгон учурда, эсти тазалоону ар түрдүү жолдор менен аткаруучу бир нече деструктор түзүлгөн болушу мүмкүн.

Ар бир типте бирдей Init иденфикаторун пайдалануу сунуш кылынуучу конструкторлорго окшош эле бардык деструкторлор үчүн Done атын пайдалануу сунуш кылынат. Ошентип, TKing тиби динамикалык объекттер менен иштөө үчүн дагы бир усул – Done деструктору менен кеңейтирилген болушу керек, андан соң ал төмөнкүдөй көрүнүштү алат.

```
                                { Хан (Король) }
 TKing=object (TChessMan)
    conctructor Init (Cl:TColumns ; Rw: TRows; BW:TColour);
    function IsMoveRight (Cl: TColumns; Rw:
                                TRows): Boolean;          virtual;
    procedure Display;                                    virtual;
    destructor Done;
 end;
```

Деструкторлор статикалык да, виртуалдык да боло алышат жана аларды мурастоого уруксат берилет. Объекттердин ар түрдүү типтери үчүн көбүнчө эсти бошотуунун ар түрдүү усулдары талап кылынгандыктан, деструкторду ар дайым виртуалдык кылып жарыялоо сунуш кылынат, ошондо анын натыйжасында объекттин ар бир тиби үчүн туура деструктор аткарылат.

Ошентип, эгер бардык шахматтык фигураларды динамикалык объекттер кылып реализацияласак, анда алардын TChessMan аталык тибинде, аны виртуалдык кылуу менен алар үчүн жалпы болгон деструкторду баяндоого болот.

```
                          { шахматтык фигура }

   TChesMan=object( TPosition)
      private
       Colour: TColour;
       Present: Boolean;
       constructor Init(Cl:  TColumns; Rw:  TRows; BW: TColour);
       procedure Put (Name :  String);
      public
       function GetColour: TColour ;
       function IsPresent :  Boolean;
```

```
    procedure Clear;
    procedure Del;
    procedure Display;                                    virtual;
    function IsMove Right( Cl: TColumns ;
                          Rw: TRows): Boolean;            virtual;
    procedure Move (Cl: TColumns ; Rw:  Trows);           virtual;
    destructor Done;                                      virtual;
  end;
```

Кандайдыр бир фигура үчүн өзүнүн жеке «таштанды жыйноочусуга» ээ болушу зарыл болуп калган учурда муну бул фигуранын тибинде өзүнүн виртуалдык деструкторун баяндоо менен жасоого болот.

Деструкторлорду динамикалык объекттер менен гана пайдалануу маниге ээ болот. Динамикалык объекттер ээлеген эсти бошотууда деструкторлор байттардын туура санынын бошотулушуна кепилдик берүүчү атайын аракеттерди аткарат. Өзгөчө деструкторлордун иши полиморфиялык объекттердин эсин тазалоодо эффективдүү болот.

Кеч байланыш болгон учурда түзүлгөн объекттердин эсин бошотуу учун деструкторду Dispose процедурасынын кенейтирилген синтаксисин пайдаланып чакыруу талап кылынат. Деструкторду Dispose процедурасынан сырткары чакыруу эч кандай эсти тазалоону аткарбайт.

Деструкторлордун мына мындай блогу көбүнчө куру (пустой) болот.

```
constructor TChesMan.Done;
 begin
 end;
```

Бул учурда, деструкторду чакырган кезде ***эпилог*** деп аталган анын стандарттык системалык функциялары гана аткарылат.

Эпилогдун иши ВУТ (ДУТ) боюнча Dispose процедурасында жазылган көрсөткүч шилтенилген (ссылается) объекттин нускасы ээлеп турган эстин өлчөмүнүн маанисин издөө жана бул маанини Dispose процедурасына жөнөтүү (пересылка) болуп саналат. Ошентип Dispose процедурасы көрсөткүч аталык типтеги нускага шилтенилгенби же балалык типтеги нускага шилтенилгенби, андан көз крандысыз түрдө байттардын туура санын бошотот.

3.8. Объекттик типтердин биргелешүүчүлүгү

Объекттик типтер менен иштөөдө мурдагы традициялык типтерге салыштырмалуу жаңы болгон мурастоо механизми пайдаланылгандыктан объекттик типтер үчүн биргелешүүчүлүк эрежелери бир топ айырмачылыктарга ээ.

Объекттик типтердин биргелешүүчүлүгүнүн негизги принциби - биргелешүүчүлүк объекттердин иерархиясынын төмөнкү деңгээлдеринен баштап жогорку деңгээлдерин карай (балалык типтерден аталык типтерге карай) багытта кеңейтирилет. Башкача айтканда, ***балалык типтеги объекттер аталык типтердин ордуна эркин колдонула алат, бирок тескерисинче эмес.*** **Андан сырткары, ар кандай балалык тип өзүнүн бардык аталык типтеринин биргелешүүчүлүгүн мураска алат.**

Объекттик типтердин биргелешүүчүлүгү үч түрдүү болот:

- объекттердин нускаларынын ортосундагы;
- объекттердин нускаларына болгон көрсөткүчтөрдүн ортосундагы;
- формалдык жана иш жүзүндөгү (фактические) параметрлердин ортосундагы.

3.8.1. Объекттердин нускаларынын ортосундагы биргелешүүчүлүк

Аталык типтеги объектке анын балалык типтеринин каалаган бирөөсүнүн нускасын ыйгарууга болот. Тескери ыйгарууга уруксат берилбейт .

Мисалы, жогоруда каралган шахматтык мисал үчүн мына мындай баяндоолор болгон кезде

```
Var
   Pos  :  TPosition;
   CM   :   TchessMan;
   WKg:    Tking;
```

төмөндөгүдөй ыйгарууларга уруксат берилет

```
Pos  : =  CM;
Pos  : =  WKg;
CM   : =  WKg;
```

ал эми төмөндөгүдөй ыйгарууларга уруксат берилбейт

```
WKg  : =  CM;
WKg  : =  Pos;
CM   : =  Pos;
```

3.8.2. Объекттердин нускаларына болгон көрсөткүчтөрдүн ортосундагы биргелешүүчүлүк

Объекттердин нускаларына болгон көрсөткүчтөрдүн ортосундагы типтердин биргелешүүчүлүгү объекттердин нускаларынын ортосундагы типтердин биргелешүүчүлүгүндөй эле иштейт жана ошол эле жалпы эрежелерге баш ийет.

Балалык типтеги объекттин көрсөткүчү аталык типтеги объекттин көрсөткүчүнө ыйгарыла алат. Мурдагыдай эле тескери ыйгарууга уруксат берилбейт.

3.8.3. Формалдык жана иш жүзүндөгү (фактические) параметрлердин ортосундагы биргелешүүчүлүк

Параметрлерди берүү учурундагы типтердин биргелешүүчүлүгү жогоруда баяндалгандарга окшош эле болот.

Берилген объекттик типтеги формалдык параметр (же параметр-маани, же параметр-өзгөрүлмө) иш жүзүндөгү параметр катары өзүнүн эле тибиндеги объектти, же өзүнүн бардык балалык типтериндеги объекттерди кабыл ала алат.

Бул учурда маани боюнча берүү (параметрлер-маанилер) жана адрес боюнча берүү (параметрлер-өзгөрүлмөлөр) механизмдеринин ортосундагы айырмачылыкты эсте тутуу керек.

Формалдык параметр-өзгөрүлмө параметр катары биргелешүүчү иш жүзүндөгү объектке болгон көрсөткүч болуп эсептелет.

Объекттик типтеги формалдык параметр-маанинин тибине кирген талааларды гана өз ичине алып турган иш жөзөндөгү параметрдин

көчүрмөсү болуп саналат. Башкача айтканда, параметр-маанини берүүдө ыйгаруу операторуна окшош аракеттер болуп өтөт.

Ушуга эле окшош, эгерде формалдык параметр объекттин тибине болгон көрсөткүч болсо, анда иш жүзүндөгү параметр объекттин ушул тибине же анын каалагандай балалык тибине болгон көрсөткүч боло алат.

3.9. Объекттик-ориентирленген программалоого карата мисал-көнүгүү

Объекттик-ориентирленген программалоонун мисалы катары төмөн жакта шахмат темасындагы ойномо программа жазуу үчүн «даярдама» («загатовка») боло ала турган ChessMod модулунун текстин келтиребиз. Бул жерде примитивдик көрүнүштөгү шахмат темасындагы мисал жөн гана объекттик-ориентирленген концепцияны өз алдынча окуп үйрөнүү максатында тандалды.

Ушул сунуш кылынган шахмат «даярдамасын» андан ары өркүндөтүүгө карай алгачкы кадамдар катары төмөндөгүлөрдү сунуш кылууга болот:

- тексттик режимдин ордуна графиктик режимди пайдалануу;
- кооз фигураларды тартуу;
- ар бир фигура үчүн жүрүштүн корректтүлүгүнүн эрежелерин (IsMoveRight функциялары) жазуу;
- тактайдагы позицияларды сактоо үчүн объекттик типтердин параллел иерархиясын түзүү жана учурдагы позицияга ылайык корректтүү жүрүштү аткаруу үчүн Move жана IsMoveRight усулдарына учурдагы позицияны берүү;
- фигуранын жайгашышын эсепке алуу менен ар бир фигура үчүн IsMoveRight усулдарын текшерүүлөр менен кеңейтүү.

```
unit ChesMod;
{ Бул модулда виртуалдык методдорду колдонуу }
          { демонстрацияланат }
interface
Type
        { Доскадагы координаталардын типтери }
   TColumns = 'a'..'h';
   TRows = 1..8;
   TColour = (WhiteColour,BlackColour);
         { Тактайдагы позиция }
TPosition = object
  private
   Column : TColumns;
```

```
      Row    : TRows;
      procedure Init( Cl: TColumns; Rw: TRows );
      procedure GetCoords ( X, Y : Word );
      procedure ErrorMessage ( Mes : String );
    public
  function  GetColumn : TColumns;
  function  GetRow : TRows;
end;

          { Шахмат фигурасы }
TChessMan = object ( TPosition )
 { Move жана IsMoveRigth методдорун кошуу !!! }
 private
  Colour : TColour;
  Present : Boolean;
  constructor Init ( Cl: TColumns; Rw: TRows; BW: TColour );
  procedure Put( Name : String );
 public
  function  GetColour : TColour;
  function  IsPresent : Boolean;
  procedure Clear;
  procedure Del;

 { Display, IsMoveRigth Move методдору виртуалдык }
          { болуп жасалды!!! }
  procedure Display; virtual;
  function  IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
  procedure Move ( Cl: TColumns; Rw: TRows); virtual;
end;

            { Хан (Король) }
TKing = object ( TChessMan )
  constructor Init (Cl: TColumns; Rw: TRows; BW: TColour);

          { Move методу чыгарылып ташталды !!! }
 { Display, IsMoveRigth методдору виртуалдык турдо жасалды !!! }
  function  IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
  procedure Display; virtual;
end;

          { Вазир (Ферзь)}
TQueen = object ( TChessMan )
  constructor Init (Cl: TColumns; Rw: TRows; BW: TColour);
```

```
    { Move методу чыгарылып таштaлды !!! }
 { Display, IsMoveRight методдору виртуалдык турдо жасалды !!! }
   function  IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
   procedure Display; virtual;
end;

                { Түркүк (Ладья) }
TCastle = object ( TChessMan )
   constructor Init (Cl: TColumns; Rw: TRows; BW: TColour);

    { Move методу чыгарылып таштaлды !!! }
 { Display, IsMoveRight методдору виртуалдык түрдо жасалды !!! }
   function IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
   procedure Display; virtual;
end;

               { Пил (Слон) }
TBishop = object ( TChessMan )
   constructor Init (Cl: TColumns; Rw: TRows; BW: TColour);

    { Move методу чыгарылып таштaлды !!! }
 { Display, IsMoveRight методдору виртуалдык түрдо жасалды !!! }
   function IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
   procedure Display; virtual;
end;
            { Ат (Конь) }
TKnight = object ( TChessMan )
   constructor Init (Cl: TColumns; Rw: TRows; BW: TColour);

    { Move методу чыгарылып таштaлды !!! }
 { Display, IsMoveRight методдору виртуалдык түрдо жасалды !!! }
   function IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean; virtual;
   procedure Display; virtual;
end;

              { Жөөчү (Пешка) }
TPawn = object ( TChessMan )
   constructor Init (Cl: TColumns; Rw: TRows; BW : TColour);

    { Move методу чыгарылып таштaлды !!! }
 { Display, IsMoveRight методдору виртуалдык түрдо жасалды !!! }
   function IsMoveRight ( Cl : TColumns; Rw : TRows) : Boolean; virtual;
   procedure Display ; virtual;
end;
```

```
var
     { Шахмат фигурасынын өзгөрүлмөлөрү }
   WKg, BKg : TKing;  { Хандар }
   WQr, BQ : TQueen;   { Вазирлер ( Ферзи) }
   WCl, WC2, BCl, BC2 : TCastle; { Түркүктөр(Ладьи) }
   WB1, WB2, BB1, BB2 : TBishop; { Пилдер(Слоны) }
   WK1, WK2, BK1, BK2 : TKnight;  { Аттар (Кони) }
   WP1, WP2, WP3, WP4, WP5, WP6, WP7, WP8,
   BP1, BP2, BP3, BP4, BP5, BP6,
   BP7, BP8 : TPawn;  { Жөөчүлөр(Пешки)}

implementation

{ Методдорду реализациялоо }
uses Crt, Dos;

    { TPosition объекти }
procedure TPosition.Init ( Cl: TColumns; Rw: TRows );
 begin
   Column := Cl;
   Row := Rw;
 end;

function TPosition.GetColumn : TColumns;
 begin
   GetColumn := Column;
 end;

function TPosition.GetRow : TRows;
 begin
   GetRow := Row;
 end;

procedure TPosition.GetCoords ( X, Y : Word );
 const
   BegX = 20; BegY = 23;
   StepX = 7; StepY = -3;
 begin
   X := BegX + StepX * (Ord ( Column ) - Ord ('a'));
   Y := BegY + StepY * ( Row - 1 ) ;
 end;

procedure TPosition.ErrorMessage (Mes : String );
 const
   BegY =23;
```

```
 begin
   GotoXY ( 1, BegY );
   TextBackGround (Blue);
   TextColor ( LightRed + Blink );
   Write ( Mes );
 end;

    { TChessMan объекти }
 constructor TChessMan.Init ( Cl: TColumns; Rw: TRows; BW: TColour);
  begin
    inherited Init ( Cl, Rw );
    Colour := BW;
    Present := True;
  end;

function TChessMan. GetColour : TColour;
 begin
   GetColour := Colour;
 end;

function TChessMan.IsPresent : Boolean;
 begin
   IsPresent := Present;
 end;

procedure TChessMan.Del;
 begin
   Clear;
   Present := False;
 end;

procedure TChessMan.Display;
 begin
   Put ( 'CM' );
 end;

procedure TChessMan.Clear;
 begin
   Put ( ' ' ) ;
   ErrorMessage ( '     ' ) ;
 end;

procedure TChessMan.Put( Name : String );
var
    X, Y : Word;
```

```
 begin
  GetCoords ( X, Y );
  GotoXY (X, Y );
  if (Row + Ord(Column)) mod 2=0 then
    TextBackGround (Black) else TextBackGround (White);
   case Colour of
    WhiteColour : TextColor ( LightRed );
    BlackColour : TextColor ( LightGreen );
   end;
   Write ( Name );
 end;

procedure TChessMan.Move;
 begin
   if IsMoveRight (Cl, Rw) then
     begin
       Clear;
       Column := Cl;
       Row := Rw;
       Display;
     end
    else ErrorMessage ( ' Туура эмес жүрүш!' );
 end;

function TChessMan. IsMoveRight;
 begin
  IsMoveRight := True;
 end;

  { TKing объектиси }
constructor TKing.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
   inherited Init ( Cl, Rw, BW ) ;
 end;

function TKing.IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean;
var
     DeltaCl, DeltaRw : Word;
 begin
   DeltaCl := Abs (Ord(Column)-Ord(Cl));
   DeltaRw := Abs (Row -Rw);
   if (DeltaCl > 1) or (DeltaRw > 1) or (DeltaCl = 0)
       and (DeltaRw = 0)
     then IsMoveRight := False
     else IsMoveRight := True
 end;
```

```
procedure TKing.Display;
 begin
   Put ( 'Kg' );
 end;

   { TPawn обьекти }
constructor TPawn.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
   inherited Init ( Cl, Rw, BW ) ;
 end;

function TPawn.IsMoveRight ( Cl: TColumns; Rw: TRows) : Boolean;
var
    DeltaCl, DeltaRw : Integer;
 begin
   DeltaCl := Abs (Ord(Column) - Ord(Cl));
   DeltaRw := Rw - Row;
   IsMoveRight := False;
   case Colour of
     WhiteColour : if ( DeltaRw=1 ) and ( DeltaCl<2 )
        or ( Row=2 ) and ( DeltaCl=0 ) and ( DeltaRw=2 )
      then IsMoveRight := True;
     BlackColour : if ( DeltaRw=-1 ) and ( DeltaCl<2 )
        or ( Row=7 ) and ( DeltaCl=0 ) and ( DeltaRw=-2 )
      then IsMoveRight := True;
   end
 end;

procedure TPawn.Display;
 begin

   Put ( ' P ' );
 end;
      { TQueen обьекти}
constructor TQueen.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
  inherited Init ( Cl, Rw, BW );
 end;

function TQueen.IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean;
 begin
   IsMoveRight := True;
 end;
```

```
procedure TQueen.Display;
 begin
  Put ( ' Q ' ) ;
 end;

      { TCastle объекти }
constructor TCastle.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
   inherited Init ( Cl, Rw, BW );
 end;

function TCastle.IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean;
 begin
   IsMoveRight := True;
 end;

procedure TCastle.Display;
 begin
   Put ( ' C ' );
 end;

       { TBishop объекти  }
constructor TBishop.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
   inherited Init ( Cl, Rw, BW );
 end;
function TBishop.IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean;
 begin
   IsMoveRight := True;
 end;

procedure TBishop.Display;
 begin
   Put ( ' B1 ' );
 end;

        { TKnight обекти }
constructor TKnight.Init (Cl: TColumns; Rw: TRows; BW: TColour);
 begin
   inherited Init ( Cl, Rw, BW ) ;
 end;

function TKnight.IsMoveRight ( Cl: TColumns; Rw: TRows ) : Boolean;
 begin
   IsMoveRight := True;
 end;
```

```
procedure TKnight.Display;
 begin
   Put ( ' K ' ) ;
 end;

procedure BoardPicture;
const
      BegX = 20;
      BegY = 24;
      StepX = 7;
      StepY = 3;
var
      BX, BY : Byte;
      i, j : Byte;
      procedure Cell ( X, Y, BW : Byte );
       var
          j : Byte;
       begin
         TextBackGround (BW);
         for j := Y to Y+StepY do
           begin
             GotoXY (X,j); Write ( '     ') ;
           end;
       end;
begin
  for j:=8 downto 1 do
    begin
      for i:=1 to 8 do
        if (i+j) mod 2=0 then
          Cell (BegX-3+(i-1)*StepX, BegY-j*StepY+1, Black)
        else Cell (BegX-3+(i-1)*StepX, BegY-j*StepY+1, White);
    end;
  GotoXY ( BegX-3, BegY+1 );
  TextBackGround ( Blue );
  TextColor ( Yellow );
  Write ( '   a      b      c      d      e      f      g      h ' );
  for j:=8 downto 1 do
   begin
     GotoXY ( BegX-5, BegY-j*StepY+2 );
     Write (j:1) ;
   end;
end;
procedure SetCursorSize ( BegLine, EndLine: Byte );
```

```
            { Курсорду орнотуу }
var Regs : Registers;
 begin
   with Regs do
    begin
      AH := $01;
      CH := BegLine;
      CL := EndLine;
    end;
   Intr( $10 , Regs );
 end;

procedure HideCursor; { Курсорду жашыруу }
var BegLine, EndLine : Byte;
 begin
   BegLine := $20; { $20 башкы сызыктын маániси катарында }
                          { курсорду көрүнбөс кылат }
   EndLine := $00;
   SetCursorSize ( BegLine, EndLine );
 end;
 begin
 TextBackGround ( Blue );
 ClrScr;
 HideCursor; { Курсорду жашыруу }
 BoardPicture;

            { Ак фигуралар }
 WKg.Init ( 'e',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'd',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'a',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'h',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'b',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'g',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'c',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'f',1,WhiteColour ); WKg.Display;
 WKg.Init ( 'a',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'b',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'c',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'd',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'e',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'f',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'g',2,WhiteColour ); WKg.Display;
 WKg.Init ( 'h',2,WhiteColour ); WKg.Display;
```

```
          { Кара фигуралар }
  WKg.Init ( 'e',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'd',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'a',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'h',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'b',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'g',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'c',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'f',8,WhiteColour ); WKg.Display;
  WKg.Init ( 'a',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'b',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'c',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'd',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'e',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'f',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'g',7,WhiteColour ); WKg.Display;
  WKg.Init ( 'h',7,WhiteColour ); WKg.Display;
end.
```

```
program Chess;
uses ChesMod;
 begin
   readln;
   WP5.Move ( 'e', 4); Readln;
   WP5.Move ( 'e', 8); Readln;

   WP5.Move ( 'e', 2); Readln;
   WP5.Move ( 'd', 3); Readln;
   WP5.Move ( 'c', 3); Readln;
   WP5.Move ( 'd', 4); Readln;
   WP5.Move ( 'e', 3); Readln;
   WP5.Move ( 'f', 3); Readln;
   WP5.Move ( 'e', 2); Readln;
   WP5.Move ( 'e', 1); Readln;
end.
```

4. КЛАВИАТУРА, КУРСОР ЖАНА ҮН МЕНЕН ИШТӨӨ

Билимди алып, аны пайдаланбаган адам, жерин айдап үрөнүн сеппеген дыйканга тете.

(Саади)

4.1. Клавиатура

Клавиатурадагы бардык клавишаларды үч тайпага бөлүүгө болот:

1. Басканда клавиатуранын буферине ASCII - кодду жиберүүчү клавишалар жана клавишалардын комбинациялары.

Мындай клавишаларга эки регистрде тең (Shift менен жана ансыз басылган) басылуучу тамгалык-санариптик жана атайын символдордун клавишаларын, ошондой эле башкаруучу символдордун (0 – 31 коддорго ээ болгон) ASCII - коддорун иштеп чыгуучу клавишалардын ***Ctrl+АлфавиттикСанариптикКлавиша*** жана ***Ctrl+КайсылбирАтайынСимвол*** комбинациялары кирет.

2. Басканда клавиатуранын буферине кеңейтирилген кодду жиберүүчү клавиша жана клавишалардын камбиинациясы.

Мындай клавишаларга өзүнчө гана басылган, ошондой эле Shift, Ctrl, Alt клавишаларынын каалаган бирөөсү менен комбинацияда басылган функционалдык клавишалар (F1-F10); Alt клавишасы менен комбинацияда басылган тамгалык-санариптик клавишалар жана дагы кээ бир башка клавишалар жана клавишалардан комбинациялары кирет.

3. Басканда клавиатуранын буферине эч кандай кодду жибербеген клавишалар жана клавишалардын комбинациялары.

Мындай клавишаларга регистрдин Shift, Ctrl, Alt, CapsLock, NumLock, ScrollLock клавишалары ошондой эле клавишалардын кээ бир комбинацялары кирет.

ASCII-коддордун жана клавишалардын басылышы менен иштелип чыгуучу коддордун толук тизмеси 3-тиркемеде келтирилген.

Клавишалардын басылышын жогорку деңгээлде иштеп чыгуу үчүн жана пайдалануучу менен программанын ортосундагы жакшы диалогду уюштуруу үчүн Crt модулунун KeyPressed жана ReadKey функциялары пайдаланылат. Диалогдук маселелерде көп пайдаланылуучу эки циклди карап өтөбүз.

Биринчи цикл

```
{Калагандай клавишанын басылышын күтүү цикли}
        repeat until KeyPressed;
```

Бул цикл клавиатуранын буфери бош болгон шартта (б.а. KeyPressed=False) кодду иштеп чгуучу каалагандай клавишанын басылышын күтүүгө алып келет .Эгерде клавиатуранын буфери жок дегенде бир кодду кармап турса (б.а. KeyPressed=True), анда бул цикл эч кандай аракетерге алып келбейт да башкаруу андан кийинки турган операторго берилет.

Ошентип,клавишанын басылышын күтүү циклин корректтүү (туура) пайдалануу мүмкүн болсун үчүн клавиатуранын буферин пайдалануучу тарабынан кокусунан басылып кеткен клавишалардын коддунан алдын ала тазалап алуу зарыл. Ал үчүн төмөнкү көрүнүштөгү экинчи цикл колдонулат.

Экинчи цикл

```
var
    ch:  char ;
      begin
    .  .  .
    {Клавиатуранын буферин тазалоо цикли}
    while  KeyPressed   do   Ch := ReadKey ;
    .  .  .
end.
```

Алфавиттик-санариптик клавишалардын басылышын иштеп чыгуунун мисалы катары “Y”, “y” символдуу тамгалык клавишалардын жана “Esc” клавишасынын басылышына гана жооп кылуучу жөнөкөй программаны карайбыз.

```
program  KeyBoard1;
uses  Crt;
var  Ch: Char;
begin
  ClrScr;      {Экранды тазалоо}
  repeat       {«Esc» нын басылышына чейин}
   while  KeyPressed   do Ch:= ReadKey; {Клавитуранын}
                                        {буферин тазалоо}
   Writeln (‘ “Y”,“y” жана “Esc” лардан башка клавишалардын’);
   Writeln (‘басылышы эч кандай аракеттерге алып келбейт’);
   Writeln (‘“Y” же “y”   тин басылышы бул текстти’);
```

```
    Writeln (‘кайталап чыгарат’);
    Writeln (‘“Esc” нын басылышы программаны аяктайт’);
    Writeln;
  repeat          {“Y”, “y” же “Esc” нын басылышына чейин }
     Ch:= ReadKey;
    until  Ch  in  [ ‘Y’, ‘y’,  #27{Esc} ];
  until  Ch=#27;       { Esc }
    while  KeyPressed   do  Ch:= ReadKey; {Клавитуранын}
                                          {буферин тазалоо}
    Writeln (‘ Программаны аякташ үчүн каалагандай’);
    Writeln (‘ клавишаны  баскыла…’);
    repeat  until  KeyPressed;  {клавишанын басылышын күтүү}
  end.
```

Клавишалардын кеңейтилген коддорун иштеп чыгууну демонстрациялоо үчүн клавиша-жебелердин жана F1-F10 функционалдык клаыишалардын басылышына жооп кылган, ал эми башка клавишалардын басылышын жокко чыгарган (игнорирует) программаны келтиребиз.

```
program  KeyBoard2;
    uses  Crt;
    var
        Ch: Char;
        ExtendedKey:   Boolean;
begin
   ClrScr;     {Экранды тазалоо}
   while  KeyPressed   do Ch:= ReadKey;    {Клавитуранын}
                                           {буферин   тазалоо}
   Writeln (‘ “Y”,“y” жана “Esc” ден башка клавишалардын’);
   Writeln (‘басылышы эч кандай аракеттерге алып келбейт’);
   Writeln (‘ “Y” же “y”   тин басылышы бул текстти’);
   Writeln (‘кайталап чыгарат’);
   Writeln (‘кайталап чыгарат’);
   Writeln;
   Writeln;
  repeat          {“Y”, “y” же “Esc” нын басылышына чейин }
  ExtendedKey:= False;
  Ch:= ReadKey;  {}
  if   Ch = #0  then  {}
    begin
       Ch:= ReadKey; {}
```

```
          ExtendedKey:= True; {}
        end;
    if  ExtendedKey  then {}
        case   Ch   of
    #75:  Writeln (‘ “ солго”  жебеси басылган’);
    #77:  Writeln (‘ “ оңго”  жебеси басылган’);
    #72:  Writeln (‘ “ жогоруга”  жебеси басылган’);
    #80:  Writeln (‘ “ төмөнгө”  жебеси басылган’);
#59 .. #68:  Writeln (‘ “ төмөнгө”  жебеси басылган’);
        end;    { case,  end }
 until  (Ch = #68)  and  ExtendedKey;  {демек  F10  басылган}
 Writeln (‘Клавишалардын басылышын демонстрациялоо бүттү’);
```

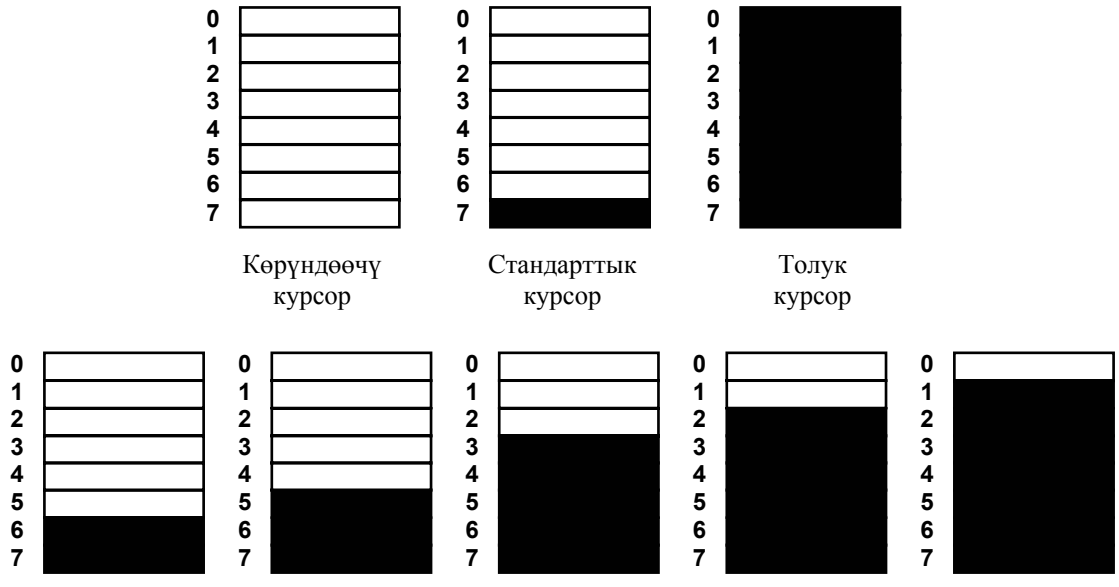
4.2. Курсор

Пайдалануучу менен праграмманы ортосунда кооз интерфейсти уюштуруу үчүн кээде курсордун формасы өзгөртүү (бир аз семизирээк, бүтүндөй символдун чоңдугунда же таптакыр эле көрүнбөс кылуу) талап кылынат.

Растрдык шрифттин ар бир символу шрифттин матрицасынын бир же бир нече жолчосу менен сүрөттөлөт . Тексттик режимдик системалык ширифттери сыяктуу анык бир 8*8 же 8*14 матрицаларынын пределинде сүрөттөлөт.

Курсор символ сыяктуу эле сканирлөө сызыктары 0 дөн баштап жогортон төмөн карай номерленет.

Ошентип, курсор, мисалы, 8*8 ширфти үчүн төмөндөгүдөй сегиз ар түрдүү формага ээ болушу мүмкүн.



Курсордук аралык жоондуктагы формалары

Курсордун формасын коюу процесордун регистрлерине кайрылууга уруксат берүүчү Registers алдын ала аныкталган тибинин жана DOS модулундагы Intr стандарттык процедурасынын жардамында аткарылат

```
Type
      Registers=record
        Case  Inrteger  of
     0: (Ax, Bx, Cx, Dx, Bp, SI, DI,Ds, Es, Flags : word);
     1: (Al, AH, Bl, BH, CH, CH, Dl, DH:  Byte);
end;
```

Курсордун формасы коюу үчүн AH регистрине $10 үзгүлтүгүнүн (прерывание) $01 функциясын киргизүү, CH регистрине сканирлөнүн баштапкы сызыгынын номерин киргизүү, CL регистирине сканирлөөнүн акыркы сызыгынын номерин киргизүү, андан кийин Intr процедурасынын жардамында $10 үзгүлтүгүн чакыруу талап кылынат.

Ошентип, бизге керек болгон процедура төмөндөгүдөй көрүнүшкө ээ болот:

```
Procedure  SetCursor Size (Begline, Endline: Byte);
 Var      Regs :  Registers ;
    Begin
        With     Regs   do
       Begin
          AH:=$01;
          CH:=Begline
          CL:=Endline    end ;  Intr ($10,Regs);   end ;
```

Стандарттык жана максималдык формадагы курсорду коюу үчүн төмөндөгүдөй процедуралар сунуш кылынышы мүмкүн.

Стандарттык курсор үчүн :

```
Procedure  Standard Cursor ;
 Var
       Begline , Endline : Byte ;
  Begin
    If  lastMode =Mono  then
  Begin
    Begline :=$0B;
    Endline :=$07;
```

```
    End;
     Else
        Begin
          Begline:=$06;
          Endline:=$07;
     End ;
Set Cursor Site(Begline,Endline);
End;
{ Максималдык курсор үчүн }
  Procedure Max Cursor ;
Var
      Begline,Endline:Byte;
Begin
     Begline :=$00
If lost Mode=Mono then Endline:=$0C  else Endline:=$07;
SetCursor Site(Begline, Endline);
End ;
```

Ал эми төмөндөгү процедура курсорду көрүнбөс кылып коёт:

```
Procedure Hide Cursor;
 Var Begline,Endline:Byte;
 Begin
    Begline :=$20; {$20 деген чоңдук баштапкы сызыктын }
                   {маниси катары курсорду көрүнбөс кылып коёт}
  Endline :=$00;
          SetCursorSite (Begline , Endline);
End ;
```

4.3. Үн

Үн эффекттерин түзүү үчүн Crt стандарттык модулуна кирген Sound, NoSound жана Delay процедуралары пайдаланылат.

```
Procedure Sound(Hz:integer);
```

Sound процедурасынын Hz параметри үндүн герцтердеги жыштыгын берет. Берилген үн (добуш) NoSound процедурасын чакыруу моментине чейин же Sound процедурасынын кийинки чакырылышына чейин уланат. Sound процедурасынын жаңы чакырылшы мурдагы үндү токтотот жана Hz параметри менен берилген жаңы жыштыктагы үндү берет.

Procedure NoSound;

NoSound процедурасы динамиктин үн чыгарышын токтотот. Эгерде программада бул процедура чакырылбаса, анда сигиналдын чыгышы программанын аякташы менен токтойт. Үн NoSound процедурасы аркылуу гана токтойт.

Procedure Delay(MSec: integer);

Delay процедурасы программанын ишин , демек тиешелүү түрдө үндүн чыгышын MSec параметиринде көрсөтүлгөн милисекундалардын санынынчалык кармап-токтотуп турат.

Sound, NoSound, Delay процедураларынын жардамында компьютердин динамигинен алууга мүмкүн болгон жөнөкөй үндөрдү демонстрациялоочу эки мисалды келтиребиз

Биринчи мисал бош эмес (занятый) телефондун үнүн берет.

```
program PhoneIsBusy;
   { бош эмес телефондун үнү }
   { туракчуулар 486DX2-66 }
   { процессору үчүн тандалып алынды }
 uses Crt;
 var
      i: Integer;
 begin
   repeat
      { үн каалагандай клавиша басылганга }
      { чейин улантылат }
     for i:=1 to 100 do
      begin
        Sound(400);
         Delay(4);
         NoSound;
      end;
      Delay(600);
   Until Keypressed;
 end.
```

Экинчи мисалда удаалаш түрдө 100 гц тен 3000 гц ке чейнки жана тескери тартиптеги жыштыктагы үндөр алынат. Натыйжада, сиренанын үнүн элестеткен үн пайда болот.

```
program Test_Sound;
  uses Crt;
  var
       i: Integer;
   begin
      repeat
       for i:=100 to 3000 do
        begin
           Sound(i);
            Delay(1);
        end;
       for i:=3000 downto 100 do
        begin
           Sound(i);
            Delay(1);
        end;
      Until Keypressed;
       NoSound;
    end.
```

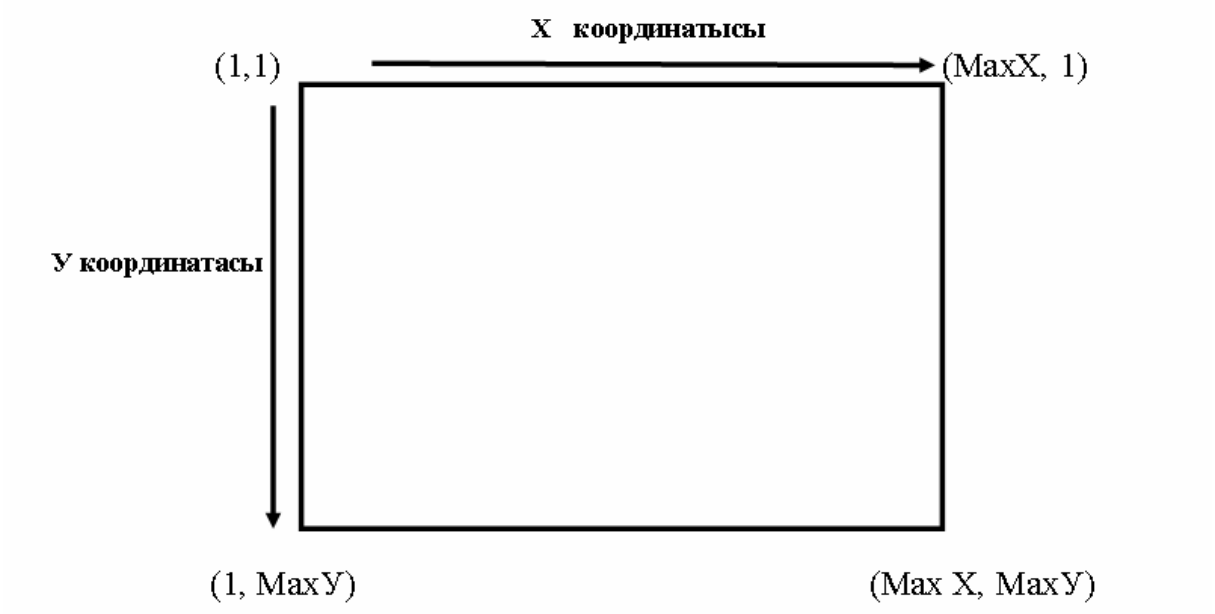
5. ТЕКСТТИК ВИДЕОРЕЖИМДЕ ИШТӨӨ

Акылдуу адам, көптү билиш үчүн окуйт, акылсыз адам – көпкө билиниш үчүн окуйт.
(Япон санжырасы)

Тексттик режимде видеоэсти башкаруу Crt стандарттык модулунун алдын ала аныкталган константаларынын, процедураларынын жана функцияларынын жардамында аткарылат. Crt модулу ошону менен эле катар үн, клавиатура жана таймер менен иштөө үчүн каражаттарды өз ичине алып турат. Бул бөлүмдө биз видеоэстин тексттик режимине тиешелүү болгон гана каражаттарды карап өтөбүз.

5.1. Тексттик режимдеги экран

Тексттик режимде экрандын өлчөмдөрү пайдаланылып жаткан компьютердеги видеоадаптердин тибинен жана берилген моменттеги тексттик режимдин пайдаланылып жаткан параметрлеринен көз каранды түрдө горизонталдык багытта 80 же 40 позициядан, ал эми вертикалдык багытта 25, 43 же 50 позициядан турушу мүмкүн. Биринчи координата (x) горизоталдык багытта солдон оңду карай өзгөрөт, ал эми экинчи координата (y) вертикалдык багытта жогортон төмөн карай өзгөрөт. Координаталардын эсептелишинин башталышы болуп (1,1) координаталарына ээ болгон жогорку сол бурч эсептелет.



5.2. Түстүн константалары

Константа	Мааниси	
Black	0	(кара)
Blue	1	(көк)
Green	2	(жашыл)
Cyan	3	(бирюза түс)
Red	4	(кызыл)
Magenta	5	(малина түс)
Brown	6	(күрөң)
LightGray	7	(ачык-боз)
DarkGray	8	(кычкыл -боз)
LightBlue	9	(ачык-көк)
LightGreen	10	(ачык-жашыл)
LightCyan	11	(ачык-бирюза түс)
LightRed	12	(ачык-кызыл)
LightMagenta	13	(ачык-малина түс)
Yellow	14	(сары)
White	15	(ак)
Blink	16	(бүлбүлдөк)

Символдор ушул келтирилген тизмедеги каалагандай түстө боло алышат, ал эми фондун түсү болсо биринчи сегиз - 0- дөн 7-ге чейинки түстө гана боло алат.

Экранга чыгарылуучу символдордун түсүн коюу TextColor процедурасынын жардамында, символдор артындагы фондун түсү TextBackGround процедурасынын жардамында, ал эми символдордун түсүнүн ачыктыгы (яркость) LowVideo, HighVideo жана NormVideo процедураларынын жардамында аткарылат

LowVideo - ачыктык битин 0 го коет.
High Video - ачыктык битин 1 ге коет.
Norm Video - ачыктык битин программа ишке салыныш алдында болгон мааниге коет.

Мисал.

```
{көк түстөгү фон}
TextBackGround(Blue);
{кызыл түстөгү символдор}
TextColor(Red);
{бүлбүлдөгөн ак түстөгү символдор}
TextColor  (White + Blink);
```

Байкалып тургандай, бүлбүлдөө белгиси символдордун түсүнө кошулуучу катары кошулат. Фондун түсү бүлбүлдөй албайт.

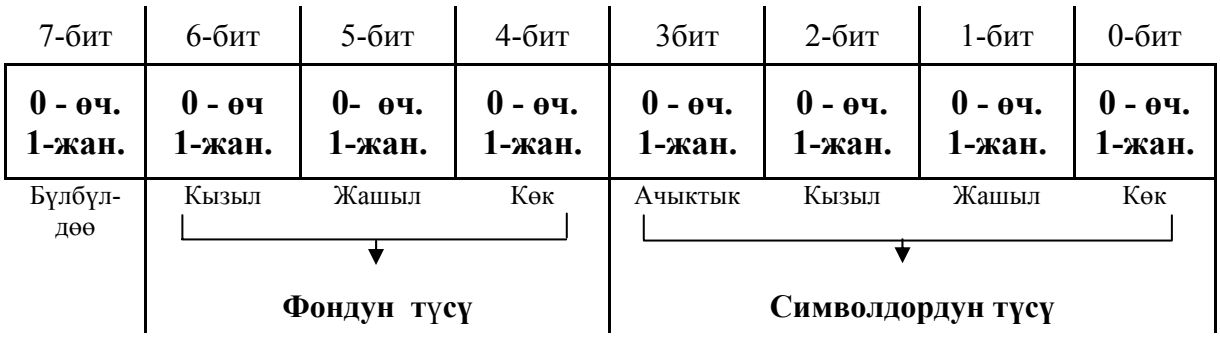
Тексттик режимде экранды тазалоо ClrScr процедурасы менен аткарылат. Бул учурда экран фондун учурдагы түсү менен боёлот (эгерде фондун түсү берилбесе, анда кара түскө боёлот).

Түстөрдүн берилишин демонстрациалоо үчүн төмөндөгүдөй программаны сунуштоо мүмкүн.

```
program TestTextColor;
uses Crt;
var  TColor, BColor:  Byte;
begin
    for BColor := Blask to LightGray do
    begin
      TextBackGround (BColor);
      ClrScr;
      for TColor := Black to White do
      begin
         TextColor (TColor);
         Writeln('Түстү текшерүү');
      end.
     TextColor (Yellow+Blink);
     Writeln ('Бүлбүлдөөнү текшерүү')
     Readln;{'Enter басылганага чейин сүрөттөлүштү}
              {экранда кармап туруу;}
    end;
    NormVideo;
end.
```

Түстөрдү башкарууда тексттик режимде жогоруда баяндалган процедуралардын ордуна Crt модулунун алдын ала аныкталган TextAttr өзгөрүлмөсүн пайдаланууга болот. Бул өзгөрүлмө Byte тибине ээ болуп, убакыттын ар бир моментинде ондук мааниге ээ болот, ал маани түстүк атрибуттардын жыйындысын (символдордун түсү, фондун түсү, бүлбүлдөө, түстүн ачыктыгы) аныктайт.

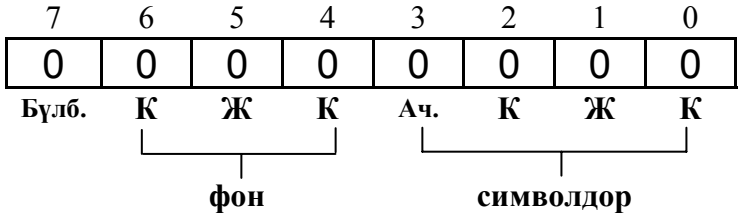
Түстүк атрибуттардын берилиш механизмин түшүнүш үчүн TextAttr өзгөрүлмөсүнүн маанисин ондук системада эмес, экилик системада кароо керек. Бул байттык өзгөрүлмөнүн сегиз битинин ар бири анык бир атрибут үчүн жооп берет. TextAttr өзгөрүлмөсүнүн ар бир битинин арналышы төмөнкү сүрөттө көрсөтүлгөн.



Символдордун түсү кичи разряддар менен аныкталгандыктан, TextAttr өзгөрүлмөсүнө төмөндөгүдөй кылып жөн гана түстүн маанисин ыйгаруу

```
TextAttr:=4; {кызыл}
```

символдордун көрсөтүлгөн маанидеги түсүнүн коюлушуна, фондун кара (нөлдүк) түсүнө, төмөнкү (нөлдүк) ачыктыкка (яркость) жана бүлбүлдөктүн жок болушуна (жетинчи биттин нөлдук мааниси) алып келет, анткени бул учурда TextAttr өзгөрүлмөсүнүн маанисинин экилик көрсөтүлүшү төмөндөгүдөй көрүнүшкө болот.



Ачыктык (яркость) атрибутуна 1000 экилик саны тиешелеш келип, ал ондук 8 санына барабар болот. Black… White алдын ала аныкталган константаларды пайдаланганда бул атрибутту тиешелүү мааниге коюу автоматтык түрдө аткарыларын байкоого болот.

Бүлбүлдөө (мерцание) атрибутуна 10000000 экилик саны тиешелеш болуп, ал 128 ондук санына барабар болот.

Фондун түсүн TextAttr өзгөрүлмөсүнүн жардамында берүүдө фондун түстүк константалары 0 дон 7 ге чейинки маанилерге, ал эми фондун түсү үчүн баштапкы биттер эмес 4-дөн 6-га чейинки биттер жооп берерин көнүлгө алуу зарыл. Ошондуктан фондун түсүнүн туура (корректтүү) берилиши үчүн керектүү константаны 16 га көбөйтүү керек, ал болсо түстүн маанисин төрт разрядка солго жылдырууга жана бул маанини 4-…6- биттерге киргизүүгө алып келет.

1-мисал.

```
        {фондун түсү  -    көк}
        {символдордун түсү -   ак}
        {бүлбүлдөө   -  өчүрүлгөн}
   {-----------------------------------------------}
   TextAttr :=  16  *  Blue + White;
                {фон} {көк}  {ак}

      же ошонун эле өзү
      TextAttr:= 16   *    1   +   15
                 {фон}{көк}  {ак}

      же ошонун эле өзү
       TextAttr :=31;
```

2-мисал.

```
        {фондун түсү   - жашыл}
        {символдун түсү -   кызыл}
        {бүлбүлдөө   -  өчүрүлгөн}
   {-----------------------------------------------}
        TextAttr  :=  16  *    Green  +     Red    +    Blink
                     {фон}{жашыл}   {кызыл}   {бүлбүлдөө}

      же ошонун эле өзү
        TextAttr    :=    16   *       2       +     4       +      128
                        {фон}   {жашыл} {кызыл} {бүлбүлдөө}

      же ошонун эле өзү
         TextAttr:=164;
```

Атрибуттарды коюунун жалпы формуласын төмөнкү көрүнүштө жазууга болот:

TextAttr:=16*Фондунтүсү+Символдордунтүсү+
Символдордунбүлбүлдөшү

5.3. Тексттик режимдеги терезелер

Turbo Pascal тили тексттик информацияны бүтүндөй экрандын рамкасында эмес анын кандайдыр бир бөлүгүндө (терезесинде) чыгарууга мүмкүнчүлүк берет. Ал үчүн Crt модулунун Window процедурасы кызмат кылат. Терезелер менен иштөөдө дагы башка WindMin жана WindMax өзгөрүлмөлөрү пайдаланылат.

Window процедурасынын бөркү төмөндөгүдөй көрүнүшкө ээ:

```
procedure Window (X1, У1, X1, X2 :  Byte);
```

Мында X1, Y1 параметрлери терезенин жогорку сол бурчунун коррдинаталары, X2, Y2 - төмөнкү оң бурчунун координаталары. Экрандын жогорку сол бурчуна (1,1) координатасы тиешелеш келет. Тексттик терезенин минималдык өлчөмү - бул бир жолчого бир мамыча. Эгерде координаталар кандайдыр бир негизде кайрылууга мүмкүн болбогон (недопустимые) координаталар болсо, анда Window процедурасына кайрылуу жокко чыгарылат.

Word тибине ээ болгон WindMin жана WindMax алдын ала аныкталган өзгөрүлмөлөрү ар дайым терезенин учурдагы аныктамасын өзүнө сактап турат. WindMin өзгөрүлмөсү учурдагы терезенин жогорку сол бурчунун координаталарын сактап турат, ал эми WinMax өзгөрүлмөсү - төмөнкү оң бурчунун координаталарын сактап турат. X координатасы ушул өзгөрүлмөлөрдүн кичине байтында, ал эми Y координатасы - чоң байтында сакталат.

Атайын көрсөтүлбөгөн учурда (по умолчанию) терезенин өлчөмү бүтүндөй экрандын өлчөмүнө барабар болуп коюулуп, видеоадаптердин режиминен көз каранды түрдө төмөндөгүлөрдүн бирөөсү болушу мүмкүн.

Режимдин константасы	Маниси	Экрандын өлчөмү
BW40	0	40х25,түстүү адаптердеги ак-кара экран
CO40	1	40ч25,түстүү адаптердеги түстүү экран
HW80	2	80ч25,түстүү адаптердеги ак-кара экран
CO80	3	80ч25,түстүү адаптердеги түстүү экран
Mono	7	80ч25, монохромдук адаптердеги ак-кара экран
Font8x8	256	EGA/VGA (43/50 жолчо) адаптерлери үчүн

BW40, CO40, BW80 жана CO80 режимдери IBM PC нин түстүү графиктик адаптерлери тарабынан уюштурулуучу төрт тексттик түстүү режим болуп эсептелет. Mono константасы IBM PC нин монохромдук адаптери тарабынан уюштурулушу мүмкүн болгон жалгыз ак - кара режим болуп саналат. Font 8x8 константасы 43 же 50 жолчолук EGA/WGA режимин кошуу үчүн арналып CO80 менен биргеликте пайдаланылат.

Видеорежим TextMode процедурасы менен коюлуп анын бөркү төмөнкүдөй көрүнүшкө ээ:

```
TextMode (режимдин константасы);
```

Window процедурасы менен терезени койгондон кийин Crt модулунун көпчүлүк процедуралары жана функциялары (Clr Eol, ClrScr, DelLine, GotoXY, InsLine, WhereX, WhereY, Read, Readln, Write, Writeln) ал терезенин координаталар системасына карата иштейт. Мисалы, түстөрдү демонстрациялоочу жогорку мисалда терезени коюну (орнотууну) кошсок, анда түс жана сөздөр берилген пределдерде гана пайда болот.

Курсорду (X, Y) координаталарына ээ болгон терезенин талап кылынган позициясына коюу үчүн GotoXY процедурасы кызмат кылып анын бөркүнүн көрүнүшү төмөндөгүдөй :

```
procedure CotoXY (X, У: Byte);
```

Ошентип дисплейдин экраны символдор киргизилүүчү позициялардын өзгөчө бир «матрицасы» болуп эсептелерин байкоо кыйын эмес. Ошондуктан көптөгөн визуалдык эффекттерди алыш үчүн дегеле матрицалар (эки өлчөмдүү) менен жакшы иштей билүү зарыл (1 - бөлүк, 9 - бапты кара).

Эки өлчөмдүү массивде биринчи координата вертикалдын маанилерин, экинчи координата горизонталдын маанилерин өзгөртөт. Ал эми ***GotoXY процедурасында - тескерисинче*** болоорун дайыма ***эсте тутуу*** зарыл.

Мисалы, мамычалар боюнча экрандын ортосунан баштап бир мезгилде эки багытта анын четтерин карай жүрүп чыгууну аткаруу менен экранды акырындап тегиз кайра боёп чыга турган жөнөкөй визуалдык эффектти алууга болот.

```
program TestGoto XY;
uses  Crt;
const
   Rows=24;            {Толтурулуучу катарлардын саны}
   Columns=80;          {толтурулуучу тилкелердин саны}
   RangeX=Coluwus div 2;   {X боюнча өзгөрүү диапазону}
   RangeУ=Rows;             {У боюнча өзгөрүү диапазону}
  Msec=0;     {кармап туру чондугу компьютердин аракет}
              {этүү тездигинен көз каранды түрдө коюлат}
     procedure NewPicture;
     var
          i, j, k:  Byte;
     begin
       for  I := RangeX downto 1 do
```

```
        begin
      k :=  Columns – i + 1;
            for  j := 1 to RangeУ do
                 begin
                   GotoXУ (i, j);
                   Write (‘ ■ ‘);
                    GotoXУ(k, j);
                    Write (‘■‘);
                    Delay (Msec)
                  end;
            end;
      end;
    begin
       TextAttr := Cyan+16*Black;
        ClrScr;
        NewPicture;
        Repeat   until    KeyPressed;
end.
```

Аракет этүү тездиги жогору болгон процессорлуу компьютерлерде экран көз ирмемде кайра боёлушу мүмкүн. Ошондуктан программадагы циклдерге Delay процедурасы кошулуп, анын параметрин өзгөртүү менен кайра боёо тездигин жөнгө салып турууга болот. Delay процедурасынын бөркү төмөндөгүдөй көрүнөшкө ээ:

```
prosedure Delay (Msec: Word);
```

Msec параметри кармап туруунун (задержка) милисекунддарынын санын берет. Ошону менен эле дагы бир эске алчу нерсе, Delay процедурасы аныk бир каталык менен иштейт, ошондуктан кармап туруу мезгили милисекунддардын берилген санына так барабар боло бербейт.

Эсте тутчу нерсе, экрандын же терезенин төмөнкү оң бурчуна символду чыгарууда сүрөттөлүш бир жолчого жогору жылып калат, ал эми курсор жаңы төмөнкү жолчонун сол жаккы позициясына өтөт.

5.4. Видеоэске түз кайрылуу

Видеоэске түз кайрылуу информацияларды экранга чыгаруу ылдамдыгын жогорулатууга мүмкүнчүлүк берет.

Эстин уячаларына (ячейкаларына) жана видеоэске түз кайрылуу айрым учур катары алдын ала аныкталган Mem, MemW, MemL массивдеринин жардамында ишке ашырылат. Mem массивинин элементтери Byte тибине, MemW массивинин элементтери Word тибине, MemL массивинин элементтери Longint тибине ээ. Бул массивдерде индекс катары адрестик константалар пайдаланылып, алардын баяндоо синтаксиси абсолюттук өзгөрүлмөлөрдүн синтаксисинин (1-бөлүк, 3-бапты кара) дал өзүндөй. Сегментти жана жылышты (смешение) көрсөтүү үчүн Seg жана Ofs стандарттык функциялары пайдаланылышы мүмкүн.

Процессордун портторуна кайрылуу үчүн Turbo Pascal да алдын ала аныкталган Byte тибиндеги Port массиви жана Word тибиндеги PortW массиви реализацияланган болуп, алардын индекси Word тибине ээ. Качан Port же PortW массивинин элементтерине маани ыйгарылганда ал тандалып алынган портко чыгарылат. Качан туюнтмаларда ушул массивдердин элементтерине шилтемелер бар болгондо портто турган маани туютмадагы массивдин жайгашкан чекитине коюлат.

IBM – биргелешкен компьютерлер үчүн түстүү тексттик режимде видеоэс $ B800:0000 адрестен, ал эми монохромдук режимде $ B000:0000 адрестен башталган уячаларда жайгашат. Сүрөттөлүштүн ар бир символу үчүн видеоэстин эки байты жооп берет:

- биринчи байт символдун ASC II – кодун кармап турат;
- экинчи байт TextAttr өзгөрүлмөсүнүн форматында түстүн атрибуттарын кармап турат.

Түстүү режим үчүн:

ASCII-код	TextAttr	ASCII-код	TextAttr	▪ ▪ ▪
$B8000:0000	$B8000:0001	$B800:0002	$B800:0003	▪ ▪ ▪

Эгерде жогоруда каралган программаны символдор түздөн-түз видеоэске жазыла тургандай кылып өзгөртсөк, анда ал төмөндөгүдөй көрүнүштө болот:

```
program ChangeScreen1;
uses Crt;
const
    Rows = 24;              {толтурулуучу катарлардын саны}
    Columns = 80            {толтурулуучу тилкелердин саны}
    RangeX = Columns div 2; {X боюнча өзгөрүү диапазану}
    RangeУ = Rows;          {Y боюнча өзгөрүү диапазану}
    Msec = 0;       {кармап туруу чондугу компьютердин }
                    {аракет этүү тездигинен көз каранды  }
                    {түрдө коюлат}
    ScreenSeg = $B800;          {түстүү режимдеги видеоэс}
                                {сегментинин адреси}
    function OffSet (X, Y: Byte): Word;
 {offset функциясы видеоэстеги сүрөттөлүш элементинин}
{удаалаш адресин экрандын матрицасынын эки өлчөмдүү}
              {координаталары боюнча эсептейт}
 begin
 OffSet := 2 * ( Columns * (Y - 1) + ( X - 1) )
 end;

procedure NewPicture (Color:  Byte);
 var
    i, j, k:  Byte;
 begin
     for I := RangeX   downto 1  do
   begin
      k:=Columns – i + 1;
      for j := 1  to  RangeУ   do
begin
    Mew[ScreenSeg: OffSet(i, j)]:=ord('▒ ');
    Mew[ScreenSeg: OffSet(i, j)+1]:= Color;
    Mew[ScreenSeg: OffSet(k,j)]:=ord('▒ ');
    Mew[ScreenSeg: OffSet(k, j)+1] := Color;
    Delay (Msec)
 end;
end;
end;
    begin
      ClrScr;
    NewPicture (Cyan);
    repeat   until    KeyPressed;
end.
```

«Жаңы» сүрөттөлүштүн элементтерин талап кылынган тартипте (иретте) жазыш үчүн видиеоэстин сегментинин конкреттүү адрестерин аныктоо OffSet жардамчы функциясы аркылуу (жардамында) аткарылат.

Эми биз, карап өткөн визуалдык эффекттин алгоритмин экран өлчөмүндө бир "сүрөттү" экинчи "сүрөт" менен алмаштырууда кандайча пайдалануу мүмкүн экендигин көрсөтөбүз.

Ал үчүн экинчи "сүрөттүн" сүрөттөлүшү алдын ала пайдаланылып жаткан графиктик режимге тиешелеш келүүчү өлчөмгө ээ болгон өзүнчө массивде даяардалат. Мисалы, 25 жолчого 80 тилке режими үчүн мындай массивдин өлчөмүн байттарда төмөнкү формула боюнча эсептөөгө болот:

ScreenSize=25 {жолчо} * 80 {тилке} * 2 {байт};

Бирок программаларда видеоэстин элементтери менен иштөө үчүн берилгендердин мына мындай структураларын пайдалануу ыңгайлуу:

```
const
   Rows =25;     {толтурулуучу катарлардын саны}
   Columns=80; { толтурулуучу тилкелердин саны}
   type
                {сүрөттөлүштүн символунун тиби}
Tpoint = record
          Symb, Color:  Byte
        end;
  { Массивдин – экрандын образынын тиби}
TypeScreen = array [ 1 . . Rows, 1 . . Columns]  of  TPoint;
  var
   {Массив – экрандын образы}
   New Screen : TypeScreen;
```

Экинчи «сурөт» NewScreen массивинде даярдалып бүткөндөн кийин анын маанилерин визуалдык эффектти түзүү үчүн талап кылынган иретте видеоэстин тиешелүү уячаларына ыйгаруу гана калат.

Жогорудагы программага салыштырмалуу жаңы программа кошумча «эски» сүрөттү экранга чыгаруу үчүн OldPicture процедурасын жана «жаңы» сүрөттөлүштү NewScreen массивинде калыптандыруу үчүн FormNewPicture процедурасын кармап турат. «Жаңы» сүрөттү экранга чыгаруучу NewPicture процедурасында «

символун ыйгарууну NewScreen массивинин тиешелүү элементтерин ыйгаруу менен алмаштыруу гана талап кылынат.

```
program ChangeSereen2;
uses Crt
const
    Rows =25;
    Columns=80;
    Range X=Columns div 2;
    Range У=Rows;
    Msec=1;
    Screen Seg=$B800; {түстүү режимде видеоэс}
                      {сегментинин адреси      }
type
    {Сүрөтөлүштүн символунун тиби}
    Tpoint = recort
                Symb, Color:  Byte
             end;
     {массивдин-экранындын образынын тиби}
    TypeScreen=array[1..Rows, 1..Columns]  of   TPoint;
var
   NewScreen : TypeScreen;   {массив-экрандын образы}
   i, j, k :  Byte;
   Ch   : char;

   function OffSet ( X, Y: Byte): Word;
    begin
     OffSet := 2 *( Columns * ( y-1) + ( x-1) )
   end;

 procedure  OldPicture;
  const
    Str: String[Columns]= 'old picture old picture old picture';
 var i:  Byte;
  procedure LastSymbol (S:  Char);
     {LastSymbol   процедурасы  S  символун экрандын}
  {төмөнкү оң бурчуна сүрөтөлүштү  жылдырбастан коёт}
 begin
  GotoXY (Columns,Rows);
 Write(S);
 GotoXY(1,1)
  Insline;
end;
```

```
      begin
          TexrBackGround(Red);
          TextColor(Greeen);
          LastSymbol(‘ ‘);
          Str := Str  +  Str;
          for i:=1  to  Rows - 1 do Write (Str);
         Delete (Str, Length(Str), 1)
        end;
 procedure FormNewpicture;
    Const
      Str :  String[Columns]  =
     ’   NEW PICTURE       NEW PICTURE      NEW PICTURE  ’;
  var
    i, j:  Byte;
  begin
      Str := Str  +  Str;
      for I := 1 to Rows do
         for j := 1  to  Columns do
           with  NewScreen[i, j] do
           begin
              Symb := Byte ( Str[j] );
              Color := Yellow + 16 * Blue;
           end;
   end;
procedure NewPicture;
var  i, j, k:  Byte;
  begin
     k := Columns – i + 1;
     for j := 1  to RangeY do
       begin
         With    NewScreen [j, i] do
          begin
             Mem [ScreenSeg: OffSet(i, j)] := Symb;
             Mem [ScreenSeg: OffSet(i, j)+1] := Color;
          end;
           with   NewScreen[j, k]   do
            begin
               Mem[ScreenSeg:  OffSet(k, j)] := Symb;
               Mem[ScreenSeg:  OffSet(k, j)+1] := Color;
            end;
             Delay(Msec)
```

```
            end;
          end;
      end;
    begin
      TextAttr:=White + 16 * Black;
      ClrScr;
      OldPicture;
       FormNewPicture;
       Ch:=ReadKey;{«эски»  сүрөттү экранда кармап туруу}
       NewPicture;
       repeat   until   KeyPressed;
 end.
```

Ушуга эле окшош сүрөттөрдү спираль боюнча (1-бөлүк, 9-баптагы программаны кара), диагональ боюнча ж.у.с алмаштыруу визуалдык эффекттерин жасоого болот.

6. ГРАФИКТИК ВИДЕОРЕЖИМДЕ ИШТӨӨ

Билип тим болуш аздык кылат,
аны колдоно билүү керек.
(В. Гёте)

Графиктик видеорежимде иштөө Graph стандарттык модулунун алдын ала аныкталган константаларынын, типтеринин, процедураларынын жана функцияларынын жардамында аткарылат.

6.1. Алдын ала аныкталган константалар

6.1.1. Түстүн константалары

Graph модулундагы түстүн константалары Crt модулундагыдай эле идентфикаторлорго жана коддорго ээ.

Константа	Мааниси	Эскертүү
SolidLn	0	Туташ сызык
DottedLn	1	Чекиттик сызык
CenterLn	2	Штрих-пунктир сызык
DashedLn	3	Пунктирдик сызык
UserBitLn	4	Программист аныктоочу сызыктын тиби
NormWidth	1	Нормалдуу жоондуктагы сызык
ThickWidth	3	Жоон сызык

Бул константалар GetLineSettings жана SetLineStyle процедуралары тарабынан пайдаланылат.

6.1.2. Боёо типтеринин константалары

Константа	Маа-ниси	Эскертүү
EmptyFill	0	Аймакты фондун түсү менен боёо.
SolidFill	1	Аймакты туташ боёо.
LineFill	2	——— сызыктары менен боёо.
ItSlashFill	3	//// сызыктары менен боёо.
SlashFill	4	//// жоон сызыктары менен боёо.
BkSlashFill	5	\\\\ жоон сызыктары менен боёо.
LtBkSlashFill	6	\\\\ сызыктары менен боёо.
HatchFill	7	Сейрек штрихтер менен боёо.
XHatchFill	8	Жыш штрихтер менен (эки багытта) боёо.
InterleaveFill	9	Үзгүлтүк сызыктар менен боёо.
WideDotFill	10	Сейрек чекиттерден турган сызыктар менен боёо.
CloseDotFill	11	Жыш чекиттерден турган сызыктар менен боёо.
UserFill	12	Программист тарабынан аныкталуучу боёо тиби.

Бул константалар GetFillSettings жана SetFillStyle процедуралары тарабынан пайдаланылат.

6.1.3. Шрифттин тибинин жана текстти түздөө константалары

Константалар	Мааниси	Эскертүү
DefaultFont TriplexFont SmallFont SansSerifFont GothicFont	0 1 2 3 4	8x8 растрдык шрифт Штрихтүү шрифт Штрихтүү шрифт Штрихтүү шрифт Штрихтүү шрифт
HorizDir VertDir	0 1	Солдон оңго карай багыт Төмөндөн жогору карай багын
UserCherSize	0	Программист тарабынан аныкталуучу символдордун өлчөмү

Бул константалар GetTextSettings жана SetTextStyle процедуралары тарабынан пайдаланылат.

Горизонтал боюнча текстти түздөө константалары

Константалар	Мааниси	Эскертүү
LeftText CenterText RightText	0 1 2	Сол жаккы чек боюнча Борбор боюнча Оң жаккы чек боюнча

Вертикал боюнча текстти түздөө константалары

Константалар	Мааниси	Эскертүү
Cottom Text Center Text Top Text	0 1 2	Төмөнкү сызык боюнча Борбор боюнча Жогорку сызык боюнча

Текстти түздөө константалары SetTextJustify проседурасы тарабынан пайдаланылат.

6.1.4. SetViewPort процедурасы үчүн константалар

Константа	Мааниси
ClipOn ClipOff	True False

6.1.5. Bar3D процедурасы үчүн константалар

Константа	Мааниси
TopOn TopOff	True False

Бул константалар Bar3D процедурасы тарабынан тартылган тик бурчтуу параллелепипеддин жогорку гранынын сүрөттөлүшүн башкаруу үчүн пайдаланылат.

6.1.6. PutImage жана SetWriteMode процедуралары үчүн константалар

Константа	Мааниси	Амал
NormalPut	0	MOV
CopyPut	0	MOV
XorPut	1	XOR
OrPut	2	OR
AndPut	3	AND
NotPut	4	NOT

Бул константалар PutImage процедурасынын жардамында сүрөттөлүштөр менен иштеген учурда логикалык амалдарды берүү үчүн пайдаланылат.

6.2. Алдын ала аныкталган типтер

6.2.1. Палитранын түстөрүн коюу тиби

```
  const
    MaxColors=15;
  type
PaletteType = record
Size:  Byte;
Colors : array[0 .. MaxColors]   of  Shortint;
      end;
```

6.2.2. Сызыктардын көрүнүшүн коюу тиби

```
LineSettingsType= record
  LineStyle    :  Word;
  Pattern      :  Word;
  Thickness    :  Word;
end;
```

6.2.3. Текстгин жасалгасын берүү тиби

```
TextSettingsType = record
    Font     :   Word;
    Direction :  Word;
     CharSize :   Word;
     Horiz      :    Word;
     Vert       :    Word;
end;
```

6.2.4. Боёо көрүнүшүн коюу типтери

```
FillPatternType = array[ 1 . . 8 ]  of   Byte;
   FillSettingsType=record
                Pattern :   Word;
                Color   :    Word;
   end;
```

6.2.5. GetViewSettings проседурасы үчүн тип

```
ViewPortType= record
    x1, y1, x2, y2 :  integer;
   Clip              :  Boolean;
end;
```

6.2.6. Arc жана Ellipse проседуралары менен иштөө үчүн тип

```
Arc CoordsType = record
   X,Y,
   Xstart,  Ystart,
   Xend,    Yend  :   integer;
end;
```

6.2.7. Экранда чекиттердин координаталарын берүү үчүн тип

```
PointType= record
     X, Y  :   integer;
 end;
```

Стандарттык графиктик процедуралар менен функциялардын кыскача баяндамалары 1- тиркемеде келтирилген.

6.3. Графиктик драйверлер жана режимдер

Graph модулунда төмөндөгүдөй графиктик драйверлерди жана режимдерди пайдаланууга болот.

6.3.1. Графиктик драйверлердин константалары

Константа	Мааниси
Current Driver	-128
Detect	0
CG A	1
MCG A	2
EG A	3
EGA64	4
EGAMono	5
IBMP514	6
HercMono	7
ATT400	8
VGA	9
PC3270	10

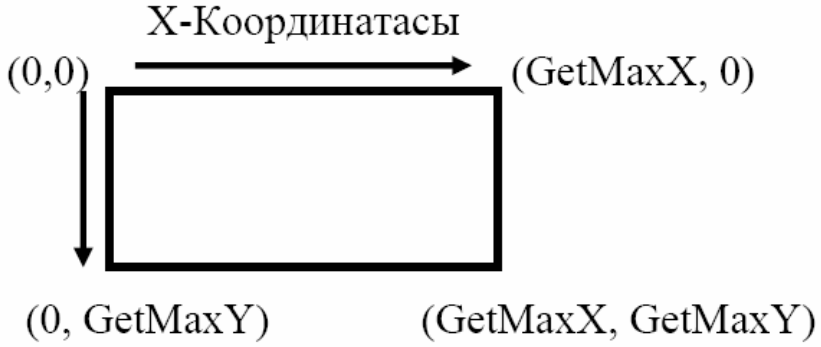
6.3.2. Графиктик режимдердин константалары

Константанын аты	Маа-ниси	Жолчо* мамыча	Па-литра	Түстөр	Беттер саны
ATT400C0	0	320*200	0	Ачык-боз, Ачык-кызыл, сары	1
ATT400C1	1	320*200	1	Ачык-бирюза түс, Ачык-малина түс, ак	1
ATT400C2	2	320*200	2	Бирюза түс, малина түс, ачык-боз	1
ATT400MED	4	640*200			
ATT400Hi	5	640*400			
CGAC0	0	320*200	0	Ачык-боз, Ачык-кызыл, сары	1
CGAC1	1	320*200	1	Ачык-бирюза түс, Ачык-малина түс, ак	1
CGAC2	2	320*250	2	Жашыл, Кызыл, Күрөң	1
CGAC3	3	320*200	3	Бирюза түс, малина түс,ачык-боз	1
CGAHi	4	640*200			
EGAlo	0	640*200		16 түс	4

EGAHi	1	640*350		16 түс	2
EGA64Lo	0	640*200		16 түс	1
EGA64Hi	1	640*350		4 түс	1
EGAMonoHi	3	640*350		Платага 64к Платага 256к	1 2
HercMonoHi	0	720*348			
IBM8514Lo	0	640*480		256-Түс	
IBM8514Hi	1	1024*76		256-Түс	
MCGAC0	0	320*200	0	Ачык-боз, Ачык-кызыл,сары	1
MCGAC1	1	320*200	1	Ачык-бирюза түс, Ачык-малина түс, ак	1
MCGAC2	2	320*200	2	Жашыл,Кызыл, Күрөң	1
MCGAC3	3	320*200	3	Бирюза түс, малина түс, ачык-боз	1
MCGAMED	4	640*200			
MCGAHi	5	640*480			
PC3270Hi	0	720*350			
VG ALO	0	640*200		16 түс	4
VGAMED	1	640*350		16 түс	2
VGAHi	2	640*480		16 түс	1

6.4. Координаталар системасы

Turbo Pascal тилинде графиктик режимдин төмөндөгүдөй координаталар системасы кабыл алынган.



GetMaxX жана GetMaxY – булар Graph модулунун стандарттык функциялары болуп, видеоадаптердин ражиминен көз каранды түрдө X,У октору боюнча тиешелеш максималдык координаталарды берет.

6.5. Учурдагы көрсөткүч

Учурдагы көрсөткүчтү башкача графиктик курсор деп да аташат. Учурдагы көрсөткүч түшүнүгү көптөгөн графиктик системаларда пайдаланылат. Ал тексттик курсорго окшош, бирок андан айырмаланып көрүнбөс (невидимый).

Көрүнбөс болуп турган учурдагы көрсөткүчтүн учурдагы координаталарын аныкташ үчүн GetX, GetY функциялары пайдаланылат, ал эми аны экранда кандайдыр бир сүрөттөлүштөрдү чыгарбастан туруп жылдыруу үчүн – MoveTo жана MoveRel процедуралары пайдаланылат. Андан сырткары, учурдагы көрсөткүчтүн абалы LineTo, LineRel, жана OutText процедуралары менен сүрөттөлүштөрдү чыгарганда автоматтык түрдө өзгөрүп, алар аны тартылган фрагменттин акыркы чекитине жылдырып коёт. Графиктик режимди инициялизациялоо жана экранды тазалоо процедуралары болгон InitGraph, GraphDefaults, ClearDevice, SetGraphMode, SetViewPort, ClearViewPort процедуралары пайдаланылган учурда да учурдагы көрсөткүчтүн абалы автоматтык түрдө өзгөрөт, алар аны координаталар башталышына алып барып коёт.

6.6. Графиктик режимди инициялизациялоо

Графиктик режимди инициализациялоо InitGraph процедурасы менен, ал эми жабуу - CloseGraph процедурасы менен аткарылат.

InitGraph процедурасынын бөркү мына мындай көрүнүшкө ээ.

```
InitGraph (Драйвер:Integer; Режим:Integer;
Драйвердин файлына карай жол: String);
```

Мындагы биринчи эки параметр графиктик драйверлердин жана режимдердин жадыбалында келтирилген константалар менен аткарылат.

Драйверлерди жана режимдерди автоматтык түрдө аныктап табуучу (автоматическое разпознавание) графиктик режимди инициялизациялоодо драйвер үчүн жооп берүүчү өзгөрүлмөгө Detect же 0 константасын ыйгаруу жетиштүү. Бул учурда режим үчүн жооп берүүчү өзгөрүлмөгө кандайдыр бир маанини ыйгаруу талап кылынбайт. String тибиндеги үчүнчү параметр жолчолук маанини алышы керек, ал маани-жолчо MS-DOS тун эрежесине ылайык графиктик драйверлердин файлдары (*.BGI) жайгашып турган каталогко болгон синтаксистик корректтүү көрсөтүлгөн жол болуп саналат.

Мисал.

```
program InitDemo;
uses Graph ; { Graph модулун пайдалануу керек}
 var
   GraphDriver,     {Графиктик  драйвер }
   GraphMode: Integer;  {Графиктик  режим    }
 begin
     GraphDriver :=  Detect; {Драйверди автоматтык          }
                             {түрдө аныктап табуу           }
                             {Бул учурда Gm  параметрин     }
                             {коюу талап кылынбайт          }
     InitGraph (GraphDriver, GraphMode, 'C:\BP\BGI\');
                          {Графиктик драйверлер C: дискинин }
{BP каталогунун, BGI камтылуучу                             }
      {каталогунда жайгашкан                                }

            {Графиктик аракеттер}
Line (0,0,GetmaxX, GetmaxY);
Readln; {акыркы сүрөттү экранда кармап туруу}
CloseGraph;    {Графиктик  режимди жабуу    }
end.
```

6.7. Графиктик режимди инициалдаштыруу каталары жана аларды иштеп чыгуу

Эгерде графиктик режимди инициалдаштыруу кезинде ката кетирилген болсо, анда Turbo Pascal тиешелүү катанын кодун генерациялайт да ал GraphResult функциясы тарабынан кайтарылып берилет. Каталардын коддору менен иштөө үчүн Graph модулунда төмөнкү константалар киргизилген:

Графика катасынын константасы	Катанын коду	Ката жөнүндөгү тиешелүү билдирүү
grOK	0	Ката жок
grNoInitGraph	-1	Графика инициялдаштырылган жок
grNoDetect	-2	Графиктик каражаттар табылган жок
grFileNotFound	-3	Файл табылган жок
grInvalidDriver	-4	Уруксат берилбеген драйвер
grNoLoadMem	-5	Драйвер жүктөлгөн эмес
grNoScanMem	-6	Эсти карап көрүүдөгү ката
grNoFLoodMem	-7	Боёо кезиндеги ката
grFontNotFound	-8	Шрифт табылган жок
grNoFontMem	-9	Шрифт эске жүктөлгөн эмес

grInvalidMode	-10	Уруксат берилбеген режим
grError	-11	Графиканын катасы
grIOError	-12	Графикалык кийирүү-чыгаруу катасы
grInvalidFont	-13	Шрифттин уруксат берилбеген файлы
grInvalidFontNum	-14	Шрифттин уруксат берилбеген номери

Кетирилиши мүмкүн болгон каталарды иштеп чыгуу менен болгон графиканы инициалдаштырууну көрсөтөбүз. Графика менен көп иштөөгө тура келген учурда инициялдаштыруу процессин өзүнчө MyGraphInit процедурасына жасап алып, кийин аны ар бир графиктик программага кошуу ыңгайлуу.

```
program InitDemo;
uses Graph ; { Graph модулун пайдалануу керек}
  var
     GraphDriver,  {графиктик драйвер}
     GraphMode, { графиктик режим }
     ErrorCode: Integer;     {катанын коду }
procedure MyGraphInit;
begin
       GraphDriver := Detected; {автоматтык аныктап табуу}
        InitGraph(GraphDriver,GraphMode,'C:\BP\BGI');
        ErrorCode:=GraphResult;  { инициялдаштыруунун
жыйынтыгы }
        If ErrorCode<> grOK then { ката болуп өттү }
        Begin
            Writeln ('Графиканын инициалдаштыруу
                          катасы;', GraphErrorMsg(ErrorCode));
            Writeln ('программанын иши үзгүлтүккө учурады');
              Halt(1);
             end;
         end;
     begin
          MyGraphInit;
      {Графиктик аракеттер}
          Line(0,0,GetMaxX, GetMaxY);
          Readln;
          CloseGraph;
     end.
```

6.8. Жөнөкөй графиктик фигуралар

Төмөндө келтирилүүчү 1-мисалда фигууралардын жөнөкөй геометриялык процедуралары жана сызыктардын учурдагы түсүн берүү процедурасы менен иштөө демонстрацияланат.

```
Program SimpleFigureDemo;
  Uses Graph ; { Graph модулун пайдалануу керек}
   Var
        GraphDriver,  {графиктик драйвер}
        GraphMode, { графиктик режим }
        ErrorCode: Integer;     {катанын коду }
        Procedure MyGraphInit;
 Begin
        GraphDriver:=Detected;   {автоматтык аныктап табу}
        InitGraph(GraphDriver,GraphMode,'D:\BP\BGI');
        ErrorCode:=GraphResult;
              { инициялдаштыруунун жыйынтыгы }
        If ErrorCode<> grOK then { ката болуп өттү }
            Begin
               Writeln ('Графиканын инициялдаштыруу
               катасы;',GraphErrorMsg(ErrorCode));
               Writeln ('программанын иши үзгүлтүккө учурады');
               Halt(1);
             End;
 End;
 Begin
        { графиктик режимди инициалдаштыруу}
        MyGraphInit;
        {-------}
        {Төрт бурчтуктун түсү-Кызыл}
        SetColor(RED);
        { Төрт бурчтук тартуу }
{ Жогорку сол бурчу координатасы(10,15) болгон чекитте, }
{төмөнкү оң бурчу (320,175) координаталуу чекитте. }
  Rectangle(10,15,320,175);
{-------}
{ Тегеректин жана диаганалдардын түсү жашыл }
SetColor(Green);
{ Төрт бурчтуктун ортосуна айлана тартуу: }
{ Биринчи эки параметр-айлананын борборунун }
{ координаталары, 40-бул айлананын радиусу}
```

```
Circle((320+10)div2, (172+15)div2,40);
{-------}

 { Төрт бурчтуктун биринчи диагоналын тартуу: }
 { (10,15)-диагоналдын биринчи учунун координаталары }
 { (320,175)-экинчи учунун координаталары }
Line(10,15,320,175);
{-------}

 { Төрт бурчтуктун экиичи диагоналын тартуу: }
 { (10,175)-диагоналдын биринчи учунун координаталары }
 { (320,15)-экинчи учунун координаталары }
Line(10,175,320,15);
{-------}

 { Сүрөттү экранда Enter клавишасын }
 { басканга чейин кармап туруу }
ReadIn;
{-------}
 { Графиктик режимди жабуу }
CloseGraph;
End;
```

Төмөндөгү экинчи программада фигураларды ар түрдүү типтеги сызыктар менен чийүү жана боё демострацияланат.

```
Program LinesAndFillDemo;
Uses Graph {Graph модулун пайдалануу керек}
Var
     GraphDriver; {графиктик драйвер}
     GraphMode; {графиктик режим }
     ErrorCode: Integer;
Proceduse MyGraphInit;
       GraphDriver,  {графиктик драйвер}
       GraphMode, { графиктик режим }
       ErrorCode: Integer;     {катанын коду }
       Procedure MyGraphInit;
   Begin
       GraphDriver:=Detected;   {автоматтык аныктап табу}
       InitGraph(GraphDriver,GraphMode,'D:\BP\BGI');
       ErorCode:=GraphResult;
            { инициялдаштыруунун жыйынтыгы }
```

```
        If ErrorCode<> grOK then { ката болуп өттү }
            Begin
                Writeln ('Графиканын инициялдаштыруу
                катасы;',GraphErrorMsg(ErrorCode));
                Writeln ('программанын иши үзгүлтүккө учурады');
                Halt(1);
    End;

Begin MyGraphInit: { Графиктик режимди инициалдаштыруу }
{-------------------------------------------------------------------------}
{Нормалду (стандарттык) жоондуктагы ак туташ сызык менен}
{кызыл түс менен боёлгон секторду тартуу}
{Түс-ак}
SetColor(white):
{Нормалдуу жоондуктагы туташ сызык}
SetfillStyle (Solidln, 0, NormWidth);
{Кызыл түс менен туташ боёо}
SetfillStyle (Solidln, Red);
{ Борбору (100,200) чекитинде болгон, баштапкы бурчу 45, }
{ бүтүү бурчу 135 жана радиусу 80 болгон боёлгон сектор}
PieStyle(100,200,45,135,80};
{-------------------------------------------------------------------------}
{Ичи сары түскө боёлгон , жоон туташ кызыл сызык }
{ менен чийилген төрт бурчтукту тартуу }
  {түс- кызыл}
Setcolor (red);
{туташ жоон сызык}
SetlineStyle(Solidline, 0, TickWidth);

 { сары түс менен туташ боёо }
SetfillStyle (solidfill , Yellow);
 { туташ жоон кызыл сызык менен төрт бурчтукту чийүү}
Rectangle(300, 10, 400, 70);
{ Төрт бурчтукту берилген түс-сары түс менен боёо }
{ (300,50}- бул боёлуучу аймактын каалагандай ички чекити }
{Red –боёлуучу аймакты чектеп турган сызыктын түсү}
 FloodFill(350, 50, Red};
{-------------------------------------------------------------------}
{Пунктирдик жоон жашыл сызык менен экрандын}
{диагоналын чийүү}
                    {түс-жашыл}
     SetColor{Green};
```

```
{ пунктирдик жоон жашыл сызык }
SetlineStyle(DashedIn, 0,ThicWidth);
{ экрандын диагоналын чийүү }
Line(0,GetMaxY, GetMax,0);
{-----------------------------------------------------------------}
{ Нормалдуу (стандарттык) жоондуктагы көк сызык менен}
{ кызыл түстөгү торчолук штрихтер менен боёлгон }
{ тегеректи чийүү}
          {түс-көк}
SetColor(Cyan);
{Нормалдуу жоондуктагы туташ сызык}
SetlineStyle(SolidIn,0,NormWidth};
{Кызыл түс менен торчолоп боёо}
SetFillStyle(XHatchFill, Red};
{ Айлананы чийүү: (500,250)-айлананын борбору}
{                      50-айлананын радиусу}
Circle (500,250,50);
{Берилген түс-кызыл түс менен боёо;     }
{ (500,250)-боёлуучу аймактын каалагандай ички чекити }
{ Cyan – боёлуучу аймакты чектеп турган сызыктын түсү }
    FloodFill (500,250, Cyan};
{-----------------------------------------------------------------}
{сүрөттү экранда кармап туруу}
Readln;
{-----------------------------------------------------------------}
       {Графиктик режимди жабуу}
        CloseGraph;
end.
```

6.9. Графиктик режимде текстти чыгаруу

Графиктик режимде текстти экранга чыгаруу OutText жана OutTextXY процедуралары менен аткарылат. Атайын көрсөтүлбөгөн учурда (по умолчанию) текст растрдык 8x8 системалык шрифти менен чыгарылат.

Андан сырткары, графиктик режимде текст менен иштөө үчүн Turbo Pascal пакетинин дистрибутивинде штрихтик шрифттер файлы деп аталган жана .CHR кеңейтирилишинде ээ болгон файлдар болот.

Штрихтик шрифттердин растрдык шрифттерден болгон башкы айырмачылыгы мына мында турат: растрдык шрифттин символдорунун өлчөмүн чоңойткон кезде сүрөттөлүштүн сапаты

начарлайт, ал эми штрихтик шрифттердин символдорунун өлчөмдөрүн чоңойткон учурда сүрөттөлүштүн сапаты начарлабайт.

Штрихтик шрифттер менен болгон иш туура (корректүү) аткарылышы үчүн *. CHR файлдарынын графиктик драйверлердин файлдары (*.BGI)жайгашкан каталогдо жайгашкан болушу зарыл.

Шрифттер менен иштөөдө керек болуучу алдын ала аныкталган константалардын тизмеги ушул баптын башында келтирилген.

Штрихтик шрифттерди пайдаланууга карата орнотуулар үчүн (установки) жана өлчөмдү өзгөртүү үчүн SetTextStyle процедурасы, экранда текстти түздөө үчүн SetTextJustify процедурасы, көрсөтүлгөн жолчонун тамгаларынын бийиктигин аныктоо үчүн TextHeight функциясы, көрсөтүлгөн жолчонун кеңдигин аныктоо үчүн TextWidth функциясы колдонулат.

Көрсөтүлбөгөн учурда өлчөмү 1 болгон DefaultFont растрдык шрифти колдонулат.

Мисал.

```
Program TextInGraphModeDemo ;
{Ар түрдүү шрифттерди жана алардын өлчөмдөрүн
демонстрациялоо}
Uses Graph;
Var
     GraphDriver; {графиктик драйвер}
     GraphMode; {графиктик режим }
     ErrorCode: Integer;

Procedure MyGraphInit;
Begin
        GraphDriver:=Detect; {авто аныктап табуу}
        InitGraph (GraphDriver, GraphMode,E:/Bp/ BGI);
        ErrorCode:=GraphResult; {инициалдаштыруунун жыйынтыгы}
        If ErrorCode, < > grOk then {ката болуп өттү}
        Begin
              Writeln('графиканын инициалдаштыруу катасы: ',
                GraphErrorMsg(ErrorCode));
              Writeln('Программанын иши үзгүлтүкө учурады');
              Halt(1);
         end;
 end;

Begin
MyGraphInit; {Графиктик режимди инициалдаштыруу}
{------------ Горизанталдык чекит-----------------------}
{Көрсөтүлбөгөн учурда кабыл алынган Default }
```

```
{ растрдык шрифти аракет этет }
{Көрсөтүлбөгөн учурда тексттин сол чеги боюнча түздөө}
{Растрдык шрифт үчүн көрсөтүлбөгөн учурдагы өлчөм-1(8*8)
 {-------------------------------------------------}
   {түс-жашыл}
SetColor(Green);
OutTextXY((GetMaxX div 2)-50,0, ‘DefaultFont, Size 1’);
Readln;
{-----------------------------------------------------------}
          {түс-көк}
SetColor (Blue);
{ шрифт-Default, багыты-горизонталдык, өлчөмү-3.}
SetTextStyle(DefaultFont, HorizDir,3);
{ Чыгарылуучу тексттин сол чеги жана жолчонун символдорунун }
{ жогорку сызыгы боюнча түздөө }
SetTextJustify (leftText, TopText);
OutTextXY(0,GetMaxY div 7, ‘DefaultFont, Size 3’);
{-----------------------------------------------------------------}
Readln;
  {түс-кызыл}
SetColor (Red);
{шрифт-Triplex, багыты-горизанталдык, өлчөмү-1}
SetTextStyle(TriplexFont, HorizDir, 1);
{ чыгарылуучу тексттин борбору жана жолчонун төмөнкү }
{ сызыгы боюнча түздөө }
SetTextJustify (CenterText, BottomText);
OutTextXY(GetMaX div 2, GetMaxY div 3, ‘TriplexFont, Size 1’);
Readln;
{-----------------------------------------------------------------}
             {түс-сары};
SetColor(Yellow);
{шрифт-Triplex, багыты-горизанталдык, өлчөмү-4}
SetTextStyle (TriplexFont, HorizDir, 4);
{ Чыгарылуучу тексттин сол чеги жана жолчонун }
{ символдорунун борбору боюнча түздөө }
SetTextJustify(LeftText, CenterText);
OutTextXY(GetMaxX div 2, GetMaxY div 2, ‘TriplexFont, Size 4’);
Readln;
{-----------------------------------------------------------------}
       {түс-ак}
SetColor(White);
{ шрифт-small, багыты-горизанталдык, өлчөмү-1 }
SetTextStyle(SmallFont, HorizDir, 1);
{ Чыгарылуучу тексттин оң чеги жана жолчонун }
{үстүңкү сызыгы боюнча түздөө}
```

```
SetTextJustify(RightText, TopText);
OutTextXY (GetMaxX div 2, GetMaxY-180, ‘SmallFont, Size 1’);
Readln;
{-----------------------------------------------------------------)
        {түс-сирень түс}
SetColor(Magenta);
{шрифт-small, багыты-горизанталдык, өлчөмү-4}
 SetTextStyle (SmallFont, HorizDir,4};
{Чыгарылуучу тексттин борбору жана жолчонун символдорунун}
{төмөнкү сызыгы боюнча түздөө}
  SetTexJustify(CenterText, BottomText);
  OutTextXY(GetMaxX div 2, GetMaxY-120, ‘SmallFont, Site 4’);
Readln;
{-----------------------------------------------------------------}
{------------------ вертикалдык текст--------------------}
{-----------------------------------------------------------------}
             {түс-жашыл}
 SetColor (Green);
{шрифт-Default, багыты-вертикалдык, өлчөмү-1}
 SetTextStyle (DefaultFont, VertDir, 1);
 OutTextXY (GetMaxX div2, GetMaxY div3, ‘DefaultFont, Size 1’);
Readln;
{-----------------------------------------------------------------}
                    {түс-көк}
SetColor(Blue);
{шрифт-Default, багыты-вертикалдык, өлчөмү-4}
 SetTexStyle(DefaultFont, VertDir, 3);

{Чыгарылуучу тексттин сол чеги жана жолчонун }
{символдорунун жогорку сызыгы боюнча түздөө }
SetTextJustify(LeftText, TopText);
OutTextXY(GetMaxX div 4,10,’DefaultFont,size 3’);
Readln;

{-----------------------------------------------------------------}
                              {Түс-кызыл}
SetColor(Red);
{Шрифт-Triplex, багыты – вертикалдык,өлчөмү -1}
SetTextStyle(TriplexFont, VertDir,1);
{Чыгарылуучу тексттин борбору жана жолчонун}
{Символдорунун төмөнкү сызыгы боюнча түздөө}
SetTextJustify(CenterText, Bottomtext);
OutTextXY((GetMaxX div 2)-40, GetMaxY div 2,’TriplexFont,Size 1’);
Readln;
```

```
{-------------------------------------------------------------------}
                        {Түс- Сары}
SetColor(Yellow);
{Шрифт-Triplex, багыты –вертикалдык,өлчөмү -4}
SetTextStyle(TriplexFont, VertDir,4);

{ Чыгарылуучу тексттин сол чеги жана жолчонун }
{ символдорунун борбору боюнча түздөө }
SetTextJustify(leftText ,CenterText);
OutTextXY((GetMax X div 2)+50, GetMaxY div 2,'TriplexFont,Size 4');
Readln;

{-------------------------------------------------------------------}
                        {Түс-Ак}
SetColor(White);
{Шрифт-Small, багыты –вертикалдык,өлчөмү -1}
SetTextStyle(SmallFont, VertDir,1);
{ Чыгарылуучу тексттин оң чеги жана жолчонун }
{ символдорунун жогорку сызыгы боюнча түздөө }
SetTextJustify(RightText, TopText);
OutTextXY((GetMaxX div 2)-30, (GetMaxY div 2)+20,'SmallFont,Size 1');
Readln;

{-------------------------------------------------------------------}
                        {Түс-сирень түс}
SetColor(Magenta);
{Шрифт-Small, багыты –вертикалдык,өлчөмү -4}
SetTextStyle(SmallFont, VertDir,4);
{ Чыгарылуучу тексттин борбору жана жолчонун }
{ символдорунун төмөнкү сызыгы боюнча түздөө }
SetTextJustify(CenterText, Bottomtext);
OutTextXY(GetMaxX div 2, GetMaxY-100, 'SmallFont,Size 4');
Readln;
{-----------------------------------------------------------}
{Графикалык режимди жабуу.}
CloseGraph;
end.
```

6.10. Графиктик режимдеги кыймыл эффекти

Графикалык режимде фигуралардын жөнөкөй кыймылдарын жасоо үчүн GetImage жана PutImage процедуралары жана алар менен иштөө үчүн арналган CopyPut, XorPut, OrPut, AndPut жана NotPut константалары пайдаланылат.

Төмөнкү келтирилүүчү дагы бир мисалда экранга жебе көрүнүшүндөгү курсорду чыгаруу жана анын жебе-клавишалардын басылыштарынан көз каранды түрдө экран боюнча тиешелүү багытта жылдырылышы демонстрацияланат.

```
program GraphCursor;
uses Graph, Crt;
var
      X,Y
      Xn,Yn: Integer; {Курсор жебесинин учунун координаталары }
      PCursor:Pointer; {Курсордун образына болгон көрсөткүч}
      TheEnd: Boolean;{Программанын ишинин аяктоо желеги}
      Ch: Char;{Жардамчы өзгөрүлмө}
      GraphDriver;{Графиктик драйвер}
      GraphMode;{Графиктик режим}
      ErrorCode:Integer;{Катанын коду}

procedure MyGraphInit;
 begin
  GraphDriver:=Detect;
  InitGraph(GraphDriver, GraphMode,'D:\bp\bgi');
  ErrorCode:=GraphResult;{Инициалдаштыруунун жыйынтыгы}
   If ErrorCode<>grOk then {Ката болуп өттү}
      begin
        Writeln ('Графиканын инициалдаштыруу катасы:',
                                                GraphErrorMeg(ErrorCode));
        Writeln ('Программанын иши үзгүлтүккө учурады');
        Halt(1);
      End;
End;

procedure CursorDraw(var x,y:integer;var p:pointer);
{ CursorDraw проседурасы жебе көрүнүшүндөгү курсорду чиет }
{ жана аны адреси эстин P көрсөткүчүнө сакталып турат. }
{Ал динамикалык областында жылдыруу үчүн сакталат }
const
{ Жебе көрүнүшүндөгү көп бурчтук }
Cursor:Array[1..8] Of Point Type=((X:320;Y:175),
(X:323;Y:186),
(X:325;Y:184),
(X:332;Y:193),
(X:338;Y:187),
(X:329;Y:180),
(X:331;Y:178),
(X:320;Y:175));
```

```
Var Size:Word;
Begin
{курсорду чийүү}
DrawPoly (8,corsor);
{курсорду боёо}
FloadFill(323,178,White);
{Курсорду кармап турган экрандын областынын өлчөмү}
Size:=ImageSize(320,175,338,193);
{ Эстин адреси P көрсөткүчүнө жазылуучу }
{ динамикалык областына курсордун }
{образын сактоо үчүн буферге эсти ажыратуу.}
GetMem (P,Size);
{ Курсордун образын сактоо }
GetImage(320,175,338,193,p^);
{ Курсор жебесинин учунун координаталарын сактоочу }
{ өзгөрүлмөлөрдү коюу }
X:=320;
Y:175;
end;

procedure NewXY(varX,Y:Integer;varFlag:boolean);
const
Esc =#127;{esc клавишасы}
leftArrow =#75; {солго жебе клавишасы (←)}
RightArrow =#77; {оңго жебе клавишасы (→) }
UpArrow =#72; {жогору жебе клавишасы (↑) }
DownArrow =#80; {төмөн жебе клавишасы (↓) }
Var
  {Клавиша-жебелер жана esc клавишасы символдорунун көптүгү}
Arrows and Esc: Set of Char;
{Кеңейтилген коддун келип түшүүсүнүн желеги}
ExtendedKey:Boolean;
{Символдорду окуу үчүн кошумча өзгөрүлмө}
Ch: char ;
begin
Arrows and Esc:=[ leftArrow, RightArrow, UpArrow, DownArrow,Esc];
Repeat
 ExtendedKey:=false;
Ch:=readkey; {басылган клавишанын символун окуу}
If ch=Esc do {программанын ишин аяктоо желегин коюу}
ExtendedKey:=true;
If ch=#0 then {эгерде ооба болсо- анда код кеңейтирилген код}
```

```
begin
Ch:=ReadKey; { кеңейтирилген коддун экинчи символун окуу}
ExtendedKey:=true; { кеңейтирилген коддун келип түшүшүн
фиксирлөө}
end;

If ExtendedKey then {эгерде код кеңейтирилген код болсо}
{ басылган клавишадан көз каранды түрдө }
{ координаталарды өзгөртүү }
Case ch of
leftArrow:X:=X-10;
RightArrow:X:=X+10;
UpArrow :Y:=Y-10;
DownArrow:Y:=Y+10
End;
{Экрандын сол чекесине курсорду фиксирлөө}
If x<0 then x:=0;
{Экрандын оң чекесине курсорду фиксирлөө}
If x>Getmax-20  then x:= Getmax-20;
{ Курсорду экрандын жогорку чегинде фиксирлөө}
If y<0 then y:=0;
{ Курсорду экрандын төмөнкү чегинде фиксирлөө}
If y>Getmay-20  then y:= Getmay-20;

{Жебе-клавишасын жа Esc клавишасын басылышын күтүү}
Unit ch in Arrows and Esc;
end;

begin
{Графикалык режимди инициалдаштыруу}
MyGraphInit
{---------------------------------------------------------------------}
{(X,Y) чекитинде курсорду чийүү жана анын образын Pcursor
көрсөткүчү көрсөтүп турган буферге сактоо}
CursorDraw(X,Y, Pcursor);
Xn:=x;  Yn:=y;
{Программанын ишин аяктоо желегинин баштапкы коюлушу}
TheEnd:=false;
{клавиатуранын буферин тазалоо}
While KeyPressed do ch:=ReadKey;
Repeat {Esc клавишасын басканга чейин кайталоо }
{New XY-бул Esc клавишасы басылгандан кийин басылган }
{ клавиша жебесинен же TheEnd:=True ден кийин жаңы }
{ Xn,Yn координаталарын кайтарып берет }
New XY(Xn,Yn, TheEnd);
```

```
{курсорду «эски» ордуна бекитүү}
 PutImage(X,Y, Pcursor^,XorPut);
{курсорду жаңы орунга чыгаруу}
PutImage(Xn,Yn, Pcursor^,XorPut);
{«жаңы» координаталар «эски» болуп калат}
X:=xn;  Y:=yn;
{Жебенин сүрөттөлүшүн экранда кармап туруу}
Delay(100);
Until TheEnd; { TheEnd=True болгон учурда циклдан чыгуу}
                {Графиктик режимди жабуу}
CloseGraph;
end;
```

Контролдук тапшырмалар

1. Экрандын берилген (X,У) чекитинде борбору жайгашкан жана берилген R радиустуу айлана анын оң-жебе жана сол-жебе клавишаларын басылышынан көз каранды түрдө тиешелүү багытта жылышын ишке ашыруучу программаны жазгыла.
2. Экрандын дал ортосуна жайгашкан, циферблатындагы сааттар араб санариптери менен көрсөтүлгөн жебелүү тегерек саатты реализациялоочу программаны жазгыла. Мында сааттардын, мүнөттөрдүн жана секундалардын жебелеринин кыймылдары таймер менен өз ара макулдукта болушу керек.
3. Бир эле координаталар системасында синусоиданы, косинусоиданы ар түрдүү түстөр менен берүүчү, ар бир ийри кайсы функциянын графиги экендиги көрсөтүлгөн сүрөттөлүштү берүүчү программаны жазгыла.

7. PASCAL ЖАНА АССЕМБЛЕР

Татаал жазылганы же адамдын бардыгына жеткилең эместиги гана болбосо, түшүнбөй турган илим жок.

(А. И. Герцен)

Көптөгөн программалоо тилдеринде жазылып жаткан программанын эффективдүүлгүн жогорулатуу максатында програмисттер ассемблердик кыстырмаларды (вставки) же ассемблердик модулдарды кошууну уюуштурушат.

Turbo Pascal тили программага ассемблердик фрагменттерди кошуунун беш жолун пайдаланууга мүмкүнчүлүк берет:

- asm оператору
- assembler дирактивалуу процедура же функция
- { $ L Ат} жана external директивалары
- inline оператору
- inline дирактивасы.

Turbo Pascal тилинин курамына тиркелген ассемблер тилинде төмөндөгүдөй резервделген идентификаторлор бар:

AH	CL	FAR	SEG
AL	CS	HIGH	SHL
AND	CX	LOW	SHR
AX	DH	MOD	SI
BH	DI	NEAR	SP
BL	DL	NOT	SS
BP	DS	OFFSET	ST
BX	DWORD	OR	TBYTE
BYTE	DX	PTR	TYPE
CH	ES	QWORD	XOR
			WORD

7.1. asm оператору

asm оператору курама операторго окшош жазылып төмөндөгүдөй структурага ээ:

```
asm
  ассемблердик командалар
end;
```

asm оператору программанын операторлук бөлүгүнүн каалагандай жеринде жайгашкан болушу мүмкүн жана ассемблердик тексттин синтаксистик корректтүү фрагментерин өзүнө алып турат.

Мисал:

```
 program   Fibonacci;
{Программа Фибаначчинин сандар катарынын n-мүчөсүн
                              эсептейт}
 uses crt;
 label  L1, L2;
 var
       F_n, N:   Word;
 begin
        ClrScr;
 Write('Фибаначчи санынын иреттик номерин кийригиле:');
  Readln(N);

  asm
  MOV CX, N    {Номерди CX - эсептегич регистрине жүктөө.}
  XOR AX, AX  {Аккумуляторду тазалоо.}
  JCXZ L2       {Эгерде эсептегич = 0 болсо,анда F_h=0.     }
  MOV DX, 1     {DX ке катардын биринчи мүчөсүн киргизүү. }
L1:  ADD AX, DX    {AX те катардын кезектеги мүчөсүн алуу      }
  XCHG AX, DX {катардын акыркы эки мүчөсүнүн                }
                {орундарын  алмаштыруу                       }
  LOOP   L1      {эгерде эсептегич <> 0 болсо,анда           }
                {улантабыз                                   }
L2:   MOV F_n, AX  { жыйынтыкты F_n ге жазуу}
   end;
   Writeln ('Fib_n=', F_n)
   end.
```

asm операторун процедуралар менен функцияларда да пайдаланууга болот. Бул учурда автоматтык түрдө процедурага/функцияга кирүү (вход) жана чыгуу (выход) коду генерацияланат. Функция туура жыйынтыкты кайтарып бериши үчүн функциянын жыйынтык маанисин @Result атайын өзгөрүлмөсүнө берүү керек болот. Андан сырткары, ассемблердик коддо адаттагы паскальдык эн-белгилердин (меткалардын) ордуна @ символу менен башталуучу жарыяланбаган (баяндалбаган) эн-белгилерди пайдаланууга уруксат берилет.

```
program  Fibonachi;
uses Crt;
var
      F_h, N: Word;
function  Fib_n  (N: Word):  Word;
begin
asm
  MOV   CX, N     {CX - эсептегичине номерди кийрүү}
  XOR   AX, AX    {аккумуляторду тазалоо}
  JCXZ @2         {эгерде эсептегич=0 болсо, анда F_h=0}
  MOV  DX, 1      {DX ке катардынбиринчи номерин кийрүү}
@1:  ADD   AX, AX     {AX те катардын кезектеги мүчөсүн алуу}
  XCHG  AX, DX  {катардын акыркы эки мүчөсүнүн }
                {орундарын  алмаштыруу}
  LOOP @1       {эгерде эсептегич <> 0 болсо,}
                {анда  улантабыз}
@2:  MOV @Result, AX {жыйынтыкты @Result өзгөр-нө Берүү}
    end
end;
begin
      ClrScr;
      Write (‘N - ди киргизгиле’);
      Readln (N);
      F_n:= Fib_n   (N);
      Writeln (‘Fib_n=’, F_n)
end.
```

7.2. assembler директивасы

Бул директиваны качан процедуранын же функциянын операторлук блогу толугу менен ассемблердик командалардан туруп калган учурларда пайдаланууга болот. Мисалы, мындан мурдагы мисалда келтирилген маселени чечүү үчүн asm операторунун ордуна assembler директивасын кармап турган төмөнкү процедураны пайдаланууга болот.

```
program Fibonachi;
uses Crt;
var
    F_n, N :Word;
function Fib_n  (N: Word): Word;   assembler;
```

```
    asm
      MOV   CX, N     {CX-эсептегичине номерди кийрүү}
      XOR   AX, AX    { аккумуляторду тазалоо}
      JCXZ @2         {эгерде эсептегич=0 болсо, анда F_h=0}
      MOV  DX, 1      {DX ке катардынбиринчи номерин кийрүү}
@1:  ADD   AX, AX     {AX те катардын кезектеги мүчөсүн алуу}
      XCHG  AX, DX   {катардын акыркы эки мүчөсүнүн }
                      {орундарын  алмаштыруу}
      LOOP @1         {эгерде эсептегич <> 0 болсо,}
                      {анда  улантабыз}
@2:
     end;
     begin
        ClrScr;
        Write ('N ди киргизгиле':); Readln(N);
        F_n := Fib_n (N);
        Writeln ('Fib_n = ', F_n)
     end.
```

assembler директивасын кармап турган прецедуралар жана фунциялар бул директиваны пайдаланбаган прцедураларга жана функцияларга салыштурмалуу төмөндөгүдөй өзгөчөлүктөргө ээ:

- комплятор параметр-маанилерди локалдык өзгөрүлмөлөргө көчүрүү үчүн кодду генерациялабайт. Бул болсо жолчо тибиндеги бардык параметр-маанилерге жана өлчөмдөрү 1, 2 же 4 байтка барабар болбогон башка параметр-маанилерге таасир этет. Процедуранын же функциянын ичинде мындай параметрлер параметр-өзгөрүлмөлөр болуп калган учурдагыдай интерпретацияланышы керек.
- Компилятар функциянын жыйынтыгы үчүн эсти ажыратпайт, жана @Result идентификатаруна болгон шилтеме (ссылка) ката болуп саналат. Жолчолук жыйынтыкка ээ болгон функциялар үчүн бул эреже таасир этпейт.
- assembler директивасын пайдаланган функциялар жыйынтыкты төмөндөгүчө кайтарып бериши керек:

а) иреттик типтеги (Integer, Char, Boolean, саналуучу типтер) функциялардын жыйынтыктары 8 разряддык маанилер болгонодо AL регистрине, 16 разряддык маанилер болгондо AX регистрине же 32 разряддык маанилер болгондо регистрлердин DX, AX түгөйүнө кайтарылып берилет.

б) чыныгы типтеги (Real) функциялардын маанилери DX, BX, AX регистрлерине кайтарылат.

в) 8087 шериктеш процессорунун (сопроцессорунун) инструкцияларын пайдаланган (Single, Double, Extended, Comp) типтердеги функциялардын жыйынтыктары шериктеш процессордун ST(0) – стек регистрине кайтарылат.

г) көрсөткүчтүк типтеги функциялардын жыйынтыктары регистирлердин DX, AX түгөйүнө кайтарлып берилет .

д) жолчолук типтеги функциялардын жыйынтыктары @Result өзгөрүлмөсү көрсөтүп турган убактылуу уячага кайтарылып берилет.

7.3. {$L ат} жана external директивалары

Компилятордун {$L файылдын_аты} директивасы паскал-программага ассемблердин жардамында алынган объекттик коддогу (б.а. .OBJ кеңейтирилишиндеги файлды) камтылуучу программаны (модулду) кошууга мүмкүнчүлүк берет. Мындай модулдун кодун программага сыйлыгыштырып коюу {$L файылдын_аты} директивасы жайгашкан чекитте аткарылат.

Жогоруда каралган Фибоначчинин n-санын эсептөө мисалын ассемблерде төмөндөгүчө жазууга болот:

```
        .MODEL  Large,  PASCAL
        .CODE
        PUBLIC Fib_n
   Fib_n  proc  NEAR  N: Word
        MOV   CX, N
        XOR   AX, AX
        JCXZ   L2
        MOV DX, 1
L1:     ADD AX, DX
        XCHG AX, DX
        LOOP   L1
L2:     NOP
        RET
Fib_n  ENDP
        END
```

Андан сырткары, паскалдагы программага объекттик код кошулат жана паскаль-программада external процедуралык

директивасын сөзсүз түрдө көрсөтүү менен Turbo Pascal тилинин синтаксисине ылайык кошулуучу процедуранын, фуңкциянын бөркү баяндалган болушу керек.

Мисалы, эгерде Fib_n функциясын алдын ала Fib_n.OBJ чыгуу файлын алуу менен ассемблерлесек, анда аны пайдалануучу программа төмөндөгүдөй көрүнүшкө ээ болот.

```
program Fibonacci;
uses Crt;
var
     F_n, N:  Word;
{$ L Fib_n.OBJ}
 function Fib_n (N: Word): Word:  external;
 begin
 ClrScr;
 Write(‘ N   ди киргизгиле:’);
 Readln(N);
 F_n := Fib_n (N);
 Writeln(‘Fib_n= ’ , F_n)
 End.
```

7.4. inline оператору

Бул оператор программага түздөн-түз машиналык коддун инструкциеларын коюуга (вставлять) мүмкүнчүлүк берип, эреже катары, өлчөмү боюнча анчалык чоң болбогон ассемблердик программалардын ордуна пайдаланылат. Inline оператору inline кызматчы сөзүнөн туруп ал сөздөн кийин тегерек кашаалардын ичинде бири-биринен жантык сызык менен ажыратылган машиналык коддун жазуулары жайгаштырылат. Мындай операторлор Turbo Pascal тилинин башка операторлору менен эркин түрдө кезектешиши мүмкүн.

Төмөн жакта эки квадраттын суммасы – $(a^2 + b^2)$ ты эсептөө үчүн inline оператору пайдаланылган, ал эми функциянын жыйынтыгын кайтарып алуу үчүн asm оператору пайдаланылган программа келтирилген.

```
program inline - operatoru;
uses CRT;
var
      a, b:  Byte;
      Res:  Word;
```

```
 finction Pifagor (a,b: Byte): Word;
 begin
   Inline
   (
    $8A/$46/04/      {MOV AL, b        }
    $F6/$66/04/      {MUL  b           }
    $8B/$C8/         {MOV CX, AX       }
    $8A/$46/06/      {MOV AL, a        }
  $F6/$66/06/        {MUL a            }
    );
 asm
     MOV   @Result, AX
 end
end;
   begin
     ClrScr;
      Write ('a, b  ларды киргизгиле:');
      Readln (a, b);
      Res := Pifagor(a,b);
      Writelu('c**2=a**2+b**2= ' , Res)
    end.
```

7.5. inline директивасы

Бул директиванын жардамында компилятор тарабынан генерацияланууучу машиналык кодко анын ар бир чакырылыш чекитинде тексти коюлуучу процедура жана функциялар баяндалат.

Анын синтаксиси inline операторунун синтаксиси менен бирдей.

Жогорудагы маселенин inline директивасы менен уюштурулган функцияны пайдалануу менен чечилиши төмөндөгүдөй көрүнүштө болот.

```
 program inline-direktive;
 uses crt:
  var
       a,b: Byte;
       Res: Word;
  finction Pifagor(a,b:Byte):word;
  inline
         (
```

```
                $59/        { POP  CX      }
                $5B/        { POP  BX      }
                $8B/$C1/    { MOV  AX, CX  }
                $F7/$E1/    { MUL  CX      }
                $8B/$C8/    { MOV  CX, AX  }
                $8B/$C3/    { MOV  AX, BX  }
                $F7/$E3/    { MUL  BX      }
                $03 /$C1/   { ADD  AX, CX  }
                );
  begin
    ClrScr;
    Write (‘a,b-ларды киргизгиле:');
    Readln (a,b);
    Res:=Pifagor(a,b);
    Writeln (‘c**2=a**2+b**2=’, Res)
  end.
```

inline директивасын пайдалануу төмөндөгүдөй өзгөчөлүктөргө ээ болот:

- inline дын процедураларынын жана функцияларынын параметрлерине алардын идентификаторлору боюнча шилтеме жасоого болбойт;
- inline дын процедуралары жана функциялары иш жүзүндө макрокомандалар болуп, алар үчүн кирүү жана чыгуу инструкцияларын кармаган код автоматтык түрдө генерацияланбайт жана эч кандай башкарууну кайтаруу инструкциялары талап кылынбайт.

8. СТАНДАРТТЫК МОДУЛДАР

Акылдуу адам өз теңин табууга умтулат. Ал эми акмак адамдар адамдардын жакшы-жаманын ылгабайт.

(Б. Паскаль)

Модул – бул константаларды, берилгендердин типтеринин баяндоолорун, өзгөрүлмөлөрдү, процедураларды жана функцияларды кармап турган библиотека болуп эсептелет. Ар бир модул өзүнчө трансляцияланат жана пайдалануучунун программасында пайдаланылышы мүмкүн.

Borland Pascal 7.0 with Objects DOS реалдык режим, корголгон режим жана Windows үчүн 6 модулду өз ичине алып турат. Булардын ичинен бир кыйла көп колдонулуучу модулдар Turbo.TPL файлында кармалып турат, ал эми калгандары .TPU кеңейтилишине ээ болгон файлдарда кармалып турат.

Пайдалануучу өзүнүн программасында туруп мындай стандарттык модулдардын процедураларын жана функцияларын пайдаланышы үчүн кандай эрежелерди эсепке алуу керектиги окуу китебинин 1-бөлүгүндө каралган.

Албетте, стандарттык модулдардын ар түрдүү процедуралары жана функциялары жөнүндөгү маалыматтарды интегралданган чөйрөнүн сурап билүү (Help) кызматын пайдалануу менен да алууга болор эле. Бирок андагы маалыматтар англис тилинде болгондуктан ал маалыматтарды кыргыз тилине жана аларды колдонууга карата айрым мисалдарды түшүндүрмөлөрү менен берүүнү туура көрдүк.

8.1. SYSTEM модулунун процедуралары жана функциялары

System модулу – бул файлдык кийирүү-чыгаруу, жолчолорду иштеп чыгуу, жылма чекиттер менен болгон амалдар жана динамикалык эсти башкаруу сыяктуу тиркелген мүмкүнчүлүктөр үчүн төмөнкү деңгээлдеги программалардын аткарылышын пайдалануну (поддержка) камсыз кылат.

Бул модулдун процедуралары жана функциялары төмөнкү файлдарда жайгашкан:

- System.TPW – Windows режими үчүн;
- System.TPP – DOS тун корголгон режими үчүн;
- System.TPU – DOS тун чыныгы режими үчүн.

Бардык модулдар жана программалар System модулун пайдаланышат, ошондуктан бул модулду атайын башка модулдар сыяктуу чакыруу талап кылынбайт. Мисалы, төмөнкү программада System модулунун GetDir – берилген дисктин учурдагы каталогун чыгаруучу процедурасы колдонулду:

```
Uses Crt;
  Var S : String;
 Begin
   GetDir(0, S); {0 - Учурдагы диск}
   WriteLn('Учурдагы каталог: ', S);
end.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

Abs функциясы	
Function Abs(X)	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин абсалюттук маанисин берет. X параметри – бул бүтүн же чыныгы типтеги туюнтма. Жыйынтыктын тиби параметрдин тибине тиешелеш келет.

Addr функциясы	
Function Addr(X): Pointer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Берилген объекттин адресин берет. X параметри – бул каалагандай өзгөрүлмө же процедуранын же функциянын идентификатору. Жыйынтык болуп X ке шилтенилген көрсөткүч эсептелет.

Append процедурасы	
Procedure Append(var F: Text);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Бириктирүү (присоединение) үчүн файлды ачат. F параметри – бул Assign процедурасынын жардамында сырткы файл менен байланыштырылышы керек болгон тексттик типтеги файлдык өзгөрүлмө.

ArcTan функциясы	
Function ArcTan(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин аркстангенсин берет. X параметри – бул чыныгы типтеги туюнтма. Жыйынтык радиандардагы X тин арктангенсинин башкы мааниси болуп эсептелет.

Assign процедурасы	
Procedure Assign(var F: Name);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Сырткы файлдын атын файлдык өзгөрүлмөгө ыйгарат. F параметри – каалагандай файлдык типтеги файлдык өзгөрүлмө, ал эми Name – жолчолук типтеги туюнтма.

Assigned функциясы	
Function Assigned(var P): Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Көрсөткүч же процедуралык өзгөрүлмө nil маанисие ээ болобу,ошону текшерет. P параметри өзгөрүлмөгө болгон шилтеме же процедуралык типтеги көрсөткүч болушу керек.

BlockRead процедурасы	
Procedure BlockRead(F: File; var Buff; Count: Word; Result: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	F файлынан саны Count параметринин маанисинен ашып кетпеген жазуулар Buff буферине окулуп берилет. Толук окулуп берилген жазуулардын анык саны (<= Count) сөзсүз эмес болгон Result параметрине берилет.

BlockWrite процедурасы	
Procedure BlockWrite(F: File; var Buff; Count: Word; Res: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Buff буфердик өзгөрүлмөсүнөн саны Count параметринин маанисинен ашып кетпеген бир же андан көп жазууларды F файлына жазат. Толук жазылып берилген жазуулардын анык саны (<= Count) сөзсүз эмес болгон Res параметрине берилет.

Break процедурасы	
Procedure Break;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Break процедурасы телосунда ушул процедура жайгашкан for, while жана repeat операторлорун аяктайт.

ChDir процедурасы	
Procedure ChDir(S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Учурдагы каталогду S параметринде көрсөтүлгөн маршруттун жардамында берилген каталогко алмаштырууну аткарат.

Chr функциясы	
Function Chr(x: Byte): Char;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	ASCII кодунун көрсөтүлгөн маанисине (иреттик номерине) тиешелеш келген символду берет.

Close процедурасы	
Procedure Close(var F);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Ачылган файлды жабат.

Concat функциясы	
Function Concat(S1, S2, … Sn: String): String;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Параметрлер менен көрсөтүлгөн жолчолор удаалаштыгынын конкатенациясын аткарат. Эгерде жыйынтык жолчонун узундугу 255 символдон ашып кетсе, анда ал 255 символго чейин кесилип ташталат.

Continue процедурасы	
Procedure Continue;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	for, while же repeat операторлорунун кайсынысынын телосунда жайгашып турган болсо, ошондогу циклдин жаңы итерациясына өтүүнү аткарат.

Copy функциясы	
Function Copy(S: String; Indx: Integer; Count: Integer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Copy функциясы символдорунун саны Count параметрине тиешелеш келген жана S жолчосунун индекси Indx параметри менен берилген символу менен башталган жолчону берет.

Cos функциясы	
Function Cos(X: Real);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин косинусун берет. X параметри – бул чыныгы типтеги туюнтма. Жыйынтык болуп X тин косинусу эсептелет.

CSeg функциясы	
Function CSeg: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows

Арналышы:	CS регистринин учурдагы маанисин берет. Сөз узундугундагы жыйынтык CSeg функциясы чакырылган программанын бөлүгү үчүн сегменттин адреси болуп эсептелет.

Dec процедурасы	
Procedure Dec(var [; N: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Өзгөрүлмөнүн маанисин азайтат. X параметри саналуучу типтеги өзгөрүлмө болуп саналат, ал эми N – бул X тин мааниси канчага өзгөрүүсү керек экендигин көрсөтүүчү бүтүн сандагы туюнтма.

Delete процедурасы	
Procedure Delete(var S: String; Indx: Integer; Count: Integer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Delete функциясы S жолчосунан номери Indx параметри менен берилген символдон баштап саны Count параметри менен берилген бардык символдорду жоготот.

Dispose процедурасы	
Procedure Dispose(var P: Pointer [, Destruktor]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	P көрсөткүчү көрсөтүп турган динмаикалык өзгөрүлмөнү жоготот.

DSeg функциясы	
Function DSeg: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	DS регистринин учурдагы маанисин берет. Учурдагы сөзгө барабар болгон жыйынтык берилгендер сегменитинин адреси болуп саналат.

Eof функциясы	
Function Eof[(var F)]: Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдар үчүн «файлдын бүтүшү» (end-of-file) абалын берет. Эгерде учурдагы позиция файлдын акыркы элементинен кийин турса же файл эч кандай элементтерди кармап турбаса, анда Eof(f) функциясы True маанисин берет. Андай болбогон учурда False маанисин берет.

Eoln функциясы	
Function Eoln[(var F: Text)]: Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Тексттик файлдар үчүн «жолчонун бүтүшү» (end-of-line) абалын берет. Эгерде файлдагы учурдагы позиция жолчонун бүтүш эн-белгисинде турса, анда Eoln(f) функциясы True маанисин, андай болбогон учурда False маанисин берет.

Erase процедурасы	
Procedure Erase(var F);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	F өзгөрүлмөсү менен байланышкан сырткы файлды өчүрөт. Жабык файлдар үчүн гана пайдаланылат.

Enclude процедурасы	
Procedure Enclude(var S: set of T; I: T);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	S көптүгүнөн I параметри менен берилген элементти жоготот.

Exit процедурасы	
Procedure Exit;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Учурдагы блоктон токтоосуз чыгып кетүүнү аткарат.

Exp функциясы	
Function Exp(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин экспоненсиалдык маанисин (X даражасындагы E саны) берет.

FilePos функциясы	
Function FilePos(var F): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдагы учурдагы позицияны берет. F параметри файлдык өзгөрүлмө болуп эсептелет. Эгерде учурдагы позиция болуп файлдын башталышы эсептелсе, анда FilePos(F) функциясы 0 маанисин берет. Эгерде учурдагы позиция файлдын бүтүшү болсо, башкача айтканда Eof(F) функциясы True маанисин берсе, анда FilePos(F) функциясы тарабынан алынуучу маани FileSize(F) функциясы берүүчү маани менен (башкача айтканда файлдын өлчөмү менен) дал келет.

FileSize функциясы	
Function FileSize(var F): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдын учурдагы өлчөмүн берет. F параметри – бул файлдык өзгөрүлмө. Бул функция F теги элементтердин санын берет. Эгерде файл бош (курук) болсо, анда 0 маанисин берет. Каралып жаткан функция тексттик файл үчүн пайладаныла албайт. Файл ачылган болушу керек.

FillChar процедурасы	
Procedure FillChar(var X; Count: Word; Val);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Берилген сандагы биринин артынан бири келүүчү байтка көрсөтүлгөн маанини жайгаштырат. X – параметри бул каалагандай типтеги өзгөрүлмө болгон шилтеме, Count – бул сөз узундугундагы туюнтма болуп саналат. Бул процедура Val параметринде көрсөтүлгөн маанини саны Count параметри аныкталган биринин артынан экинчиси келген байттарга X өзгөрүлмөсү ээлеп турган биринчи байттан баштап жайгаштырат. Чек аралардын уруксат берилгендиги же берилбегендигин текшерүү аткарылбайт.

Flush процедурасы	
Procedure Flush(var F: Text);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Тексттик файлды чыгаруу үчүн ачылган буфердин мазмунун чыгарат. F параметри – бул тексттик типтеги өзгөрүлмө.

Frac функциясы	
Function Frac(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин бөлчөк бөлүгүн берет. X параметр – бул чыныгы типтеги туюнтма. Жыйынтык болуп X тин бөлчөк бөлүгү, башкача айтканда Frac(X)=X-Int(X) эсептелет.

FreeMem процедурасы	
Procedure FreeMem(P: Pointer; Size: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Берилген өлчөмдөгү динамикалык өзгөрүлмөнү жоготот. P параметри – бул алдын ала ыйгаруу GetMem процедурасынын жардамында аткарылган, же маани ыйгаруу операторунун жардамында аткарылган өзгөрүлмөгө көрсөткүчтөрдүн каалагандай тибине

	таандык) болгон көрсөткүч болуп эсептелет. Size параметри – бул жоготулуучу динамикалык өзгөрүлмөнүн өлчөмүн берүүчү сөз учундугундагы туюнтма. Анын мааниси GetMem процедурасы тарабынан мурдатан өзгөрүлмө үчүн бөлүнүп (ажыратылып) коюлган эстин байттарынын саны менен так дал келиши керек. FreeMem процедурасы P көрсөткүчү көрсөтүп турган өзгөрүлмөнү жок кылат жана динамикалык бөлүштүрүлүүчү аймакта ал ээлеп турган эстин аймагын бошотот. Эгерде P көрсөткүчү динамикалык бөлүштүрүлүүчү аймактагы эстин бөлүгүнө шилтеме жашабаса, анда процедуранын аткарылыш этабында ката пайда болот. FreeMem процедурасына кайрылгандан кийин P көрсөткүчүнүн мааниси аныкталбаган болуп калат, ал эми кийинки P^ көрүнүшүндөгү шилтемелер катага алып келет.

GetDir процедурасы	
Procedure GetDir(D: Byte; var S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Бурилген дискте учурдагы каталогду берет. D параметри – бул бүтүн типтеги туюнтма, ал эми S – жолчолук типтеги өзгөрүлмө. D параметри менен берилген дисктеги учурдагы каталогдун мааниси S өзгөрүлмөсүнө берилет. D=0 мааниси учурдагы диск салгычты (дискавод), 1 мааниси – A дискин, 2 мааниси – B дискин ж.у.с. көрсөтөт.

GetMem процедурасы	
Procedure GetMem(P: Pointer; Size: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Көрсөтүлгөн өлчөмдөгү жаңы динамикалык өзгөрүлмөнү түзөт жана блоктун адресин өзгөрүлмө-көрсөткүчкө жайгаштырат. P параметри өзгөрүлмө-көрсөткүч болуп көрсөткүчтөрдүн каалагандай тибине таандык боло алат. Size параметри сөз узундугуна ээ болот жана динамикалык өзгөрүлмө үчүн бөлүнгөн эстин өлчөмүн (байттардагы) берет. Жаңыдан түзүлгөн динамикалык өзгөрүлмөгө P^ жардамында шилтеме жасоого болот. Динамикалык бөлүштүрүлүчү эсте ажыратылышы мүмкүн болгон эң чоң блок 65521 байтты түзөт.

Halt процедурасы	
Procedure Halt[(ExtCode: Word)];	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Программанын аткарылышын токтотот жана башкарууну операциялык системага берет. ExtCode параметри (аяктоо коду) көрсөтүлүшү сөзсүз эмес болгон сөз узундугундагы туюнтма болуп, программанын аяктоо кодун берет. Параметри жок Halt процедурасы Halt(0) чакыруусуна тиешелеш келет. Аяктоо коду DOS модулунун ExitCode функциясынын жардамында пайда болуучу процесс тарабынан же DOS тун командалык файлындагы ERRORLEVEL текшерүүсүнүн жардамында текшерилиши мүмкүн.

Hi функциясы	
Function Hi(X): Byte;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин чоң байтын берет. X параметри – бул бүтүн типтеги туюнтма же сөз. Hi функциясы X тин чоң байтын белгиге ээ болбогон маани көрүнүшүндө берет.

High функциясы	
Function High(X);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин чоң байтын берет. X параметри – бул шилтемелик өзгөрүлмөнүн идентификатору. X менен белгиленген тип же X менен белгиленген өзгөрүлмөнүн тиби иреттик тип, массив же жолчолук тип болушу керек. Иреттик тип үчүн High функциясы типтин диапазонундагы чоң маанини кайтарып берет. Массив үчүн High функциясы массивдин иреттик тибинин диапазонундагы чоң маанини берет. Ачык массив же жолчолук параметр үчүн High функциясы иш жүзүндөгү параметрдеги элементтердин санын көрсөтүү менен Word тибиндеги маанини берет.

Inc процедурасы	
Procedure Inc(X [; N: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Өзгөрүлмөнүн маанисин чоңойтот. X параметри – бул саналуучу типтеги өзгөрүлмө, ал эми N – бүтүн сандык туюнтма. X тин мааниси 1 ге (же N айкын берилген учурда N ге) чоңоёт. Башкача айтканда Inc(X) деген X:=X+1 ге тиешелеш келет, ал эми Inc(X,N) деген X:=X+N ге тиешелеш келет. Inc функциясы оптималдаштырылган кодду жаратат жана чоң циклдерде өзгөчө пайдалуу.

Include процедурасы	
Procedure Include(var S: set of T; I: T);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Элементти S көптүгүнө кошот. S – бул көптүк типтеги өзгөрүлмө, ал эми I – бул S базалык тиби менен биргелешүүчү типтеги туюнтма. Берилген I элемент берилген S көптүгүнө кошулат. Include(S,I) конструкциясы S:=S+[I] ге тиешелеш келет, бирок Include процедурасы бир кыйла эффективдүү кодду генерациялайт.

Insert процедурасы	
Procedure Insert(String1: String; var S: String; Indx: Integer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жолчого камтылуучу жолчону (подстрока) коёт. String1 параметри – бул жолчолук типтеги туюнтма. S параметри – каалагандай узундуктагы жолчолук типтеги өзгөрүлмө. Indx параметри – бул бүтүн типтеги туюнтма. Каралып жаткан процедура String1 параметри менен берилген жолчонун S параметри менен берилген жолчого анын Indx параметри менен берилген позициясынан баштап коёт. Эгерде натыйжада алынган жолчо 255 символдон ашып кетсе, анда ал 255 символго чейин кесилип ташталат.

Int функциясы	
Function Int(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин бүтүн бөлүгүн берет. X параметри – бул чыныгы типтеги туюнтма. Жыйынтык болуп X тин бүтүн бөлүгүг эсептелет, башкача айтканда X нөл жакка карай тегеректелет.

IOResult функциясы	
Function IOResult: Integer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Акыркы аткарылган кийирүү-чыгаруу амалынын абалын көрсөтүүчү бүтүн маанини берет. IOResult функциясынын жардамында кийирүү-чыгаруу каталарын кармап алуу (перехват) үчүн кийирүү-чыгарууну текшерүү режими коюлган болушу керек (компилятордун {$I-} директивасы). Эгерде кийирүү-чыгарууну текшерүү коюлган болсо жана кийирүү-чыгаруу катасы болуп өтсө, анда IOResult функциясына кайрылуу аткарылып жатканда андан аркы бардык кийирүү-чыгаруу амалдары жокко чыгарылат. IOResult функциясына болгон кайрылуу анын ички ката желегин (флаг ошибки) түшүрөт.

Length функциясы	
Function Length(S: String): Integer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жолчонун динамикалык узундугун берет. S параметри – бул жолчолук типтеги туюнтма. Жыйынтык болуп S узундугу эсептелет.

Ln функциясы	
Function Ln(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин натуралдык логорифмин берет. X параметри

	бул чыныгы титеги туюнтма. Жыйынтык болуп X туюнтмасынын натуралдык логорифми эсептелет.

Lo функциясы	
Function Lo(X): Byte;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин кичине байтын берет. X параметри бул бүтүн титеги параметр же сөз узундугундагы параметр. Lo функциясы X тин кичине байтын белгиге ээ болбогон (беззнаковые) маани катары берет.

Low функциясы	
Function Low(X);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин диапазонундагы кичине маанини берет. X параметри бул типтин идентификатору же өзгөрүлмөгө болгон шилтеме (ссылка). X менен берилген тип же X менен берилген өзгөрүлмөнүн тиби иреттик тип, массив же жолчолук болушу керек. Иреттик тип үчүн Low функциясы 0 маанисин берет. ачык массив же жолчолук параметр үчүн Low функциясы 1 маанисин берет.

MaxAvail функциясы	
Function MaxAvail: Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Эстин динамикалык бөлүштүрүлүүчү аймагындагы эң чоң бош блоктун өлчөмүн берет. MaxAvail функциясы New же GetMem процедурасынын жардамында бөлүштүрүүгө мүмкүн болгон эстин динамикалык бөлүштүрүлүүчү аймагындагы эң чоң үзгүлтүксүз блоктун өлчөмүн берет. Бош динамикалык бөлүштүрүлүүчү эстин жалпы көлөмүн алуу үчүн MemAvail функциясы пайдаланылат.

MemAvail функциясы	
Function MemAvail: Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Эстин динамикалык бөлүштүрүлүүчү аймагындагы бош блоктордун санын берет. Бул сан эстин динамикалык бөлүштүрүлүүчү аймагынын көрсөткүчүнөн төмөн жайгашкан бардык бош блоктун өлчөмдөрү менен динамикалык бөлүштүрүлүүчү аймактын көрсөткүчүнөн жогору жайгашкан бош эстин көлөмүн кошуу жолу менен аныкталат. Байкай кетчү нерсе эстин динамикалык бөлүштүрүлүүчү аймагынын фрагменттелишинен улам өлчөмү алынуучу маниге тиешелеш болгон эстин блогуна кайрылуу мүмкүнчүлүгүнүн ыктымалдуулугу азыраак. Эң чоң бош блоктун өлчөмүн алуу үчүн MaxAvail функциясын пайдалануу керек.

MkDir функциясы	
Function MkDir(S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Камтылуучу каталогду (подкаталогду) түзөт. S параметри – бул жолчолук типтеги туюнтма. S маршруту боюнча кайрылууга боло турган жаңы каталог түзүлөт. Бул жазуунун акыркы элементи, бар болгон (жашап турган – сушествующий) файлдын атын бере албайт.

Move процедурасы	
Procedure Move(var Src, Dst; Count: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Үзгүлтүксүз байттардын берилген санын берилген диапазондон көрсөтүлгөн максаттык (целевой) диапазонго көчүрөт. Src жана Dst параметрлери каалагандай типтеги өзгөрүлмөгө болгон көрсөткүч болуп саналат. Count параметри – бул Word тибиндеги туюнтма. Move процедурасы байттык блокту Src дан, өлчөмү (байттардагы) Count параметри менен берилген, Dst параметри менен берилген диапазондун биринчи

	байтынан башталган блокко көчүрөт. Бул процедура менен кылдаттык менен иштөө керек, анткени эч кандай текшерүү аткарылбайт.

New процедурасы	
Procedure New(var P: Pointer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жаңы динамикалык өзгөрүлмөнү түзөт жана ага көрсөткүчтү коёт. P параметри өзгөрүлмө-көрсөткүч болуп каалагандай типтеги көрсөткүчкө таандык болот. Эстин ажыратылуучу блогунун өлчөмү P көрсөткүч турган типтин өлчөмүнө тиешелеш болот. Жаңыдан түзүлгөн өзгөрүлмөгө P^ жардамында шилтеме жасоого болот. Эгерде жаңы өзгөрүлмөгө эсти бөлүү (ажыратуу) үчүн динамикалык бөлүштүрүлүүчү эсте, бош эс жетишерлик болбосо, анда программанын аткарылыш убагында ката болуп өтөт.

Odd функциясы	
Function Odd(X: Longint): Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргумент так сан боло алышын текшерет. X параметри – бул узун бүтүн типтеги туюнтма. Жыйынтык True маанисин кабыл алат, эгерде X так сан болсо, андай болбогондо False маанисин алат.

Ofs функциясы	
Function Ofs(X): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Берилген объект үчүн жылышты (offset – смещение) берет. X параметри каалагандай өзгөрүлмө, же процедура же функциянын идентификатору болуп эсептелет. Узундугу сөзгө барабар болгон жыйынтык X үчүн жылыш болуп эсептелет.

Ord функциясы	
Function Ord(X): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Саналуучу типтеги маани үчүн иреттик номерди берет. X параметри саналуучу типтеги туюнтма болуп эсептелет. Жыйынтык узун бүтүн типке ээ болот жана анын мааниси X тин иреттик номери болот.

ParamCount функциясы	
Function ParamCount: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Командалык жолчодо берилген параметрлердин санын берет. Ажыраткычтар болуп бош символдор (пробелдер) жана табуляция символдору эсептелет.

ParamStr функциясы	
Function ParamStr(Indx): String;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Командалык жолчонун берилген параметрин кайтарып берет. Indx параметри Word тибиндеги туюнтма болуп саналат. Бул функция номери Indx менен берилген командалык жолчонун номерин берет же эгерде Indx тин мааниси нөлгө барабар же ParamCount тан чоң болсо, анда бош жолчону берет.

Pi функциясы	
Function Pi: Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	3.1415926535897932385 (Pi санынын мааниси) маанисин берет. Компилятор 8087 (80287, 80387) типтеги шериктеш процессорунун (сопроцессорунун) режиминде иштеп жатабы же программалык жабдылыштын режиминде эле иштеп жатабы ошондон көз каранды түрдө тактык өзгөрүшү мүмкүн.

Pos функциясы	
Function Pos(Sub, S: String): Byte;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жолчодо камтылып туруучу жолчону издейт. Sub жана S параметрлери – жолчолук типтеги туюнтмалар. Бул функция S жолчосундагы Sub параметри менен берилген камтылуучу жолчону издейт жана камтылуучу жолчону S жолчосуна кирген биринчи символунун позициясы болгон бүтүн маанини берет. Эгерде камтылуучу жолчо табылбаган болсо, анда функция нөл деген маанини берет.

Pred функциясы	
Function Pred(X);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин мындан мурдагы маанисин берет. X параметри – саналуучу типтеги туюнтма. Жыйынтык ошондой эле типке ээ болуп X тин мындан мурдагы мааниси болуп эсептелет.

Ptr функциясы	
Function Ptr(Seg, Offs: word): Pointer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Базалык сегмент жана жылыш көрүнүшүндө берилген адрести көрсөткүч тибиндеги мааниге өзгөртүп түзөт. Seg (сегмент) жана Offs (жылыш) параметрлери – Word тибиндеги туюнтмалар. Жыйынтк Seg жана Offs параметрлери менен берилген адреске болгон көрсөткүч болуп эсептелет. Ptr функциясынын жыйынтыгы ыйгаруу боюнча каалагандай типтеги көрсөткүчтөр менен биргелешүүчү болот.

Random функциясы	
Function Random[(Range: Word)];	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows

Арналышы:	Кокустук санын берет. Эгерде Range (диапазон) параметри берилбесе, анда жыйынтык болуп 0<=x<=1 диапазонунан кокусунан алынган x чыныгы саны болуп эсептелет. Эгерде Range параметри берилген болсо, анда ал бүтүн типтеги туюнтма болуусу керек, ал эми жыйынтык болуп 0<=x<=N, (мында N, Range параметри менен берилген маани) диапазонунан кокусунан алынган сөз узундугундагы сан эсептелет. Эгерде Range параметри нөлдөн кичине же барабар болсо, анда алынуучу маани нөлгө барабар болот.

Randomize процедурасы	
Procedure Randomize;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Кокустук сандардын тиркелген генераторун кокустук маани менен инициялизациялайт. Кокустук маани системалык таймерден алынат. Кокустук сандардын генераторунун ишинин натыйжасында алынган сан RandSeed атына ээ болгон алдын ала аныкталган өзгөрүлмөдө сакталат.

Read процедурасы (тексттик файлдар)	
Procedure Read([var F: Text;] V1, [, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Тексттик файлдан бир же андан көп маанини бир же андан көп өзгөрүлмөгө окуйт. F параметри (эгер ал көрсөтүлгөн болсо) тексттик файлга тиешелеш болгон файлдык өзгөрүлмө. Эгер ал жок болсо, анда стандарттык Input файлдык өзгөрүлмөсүн пайдалануу керек деген түшүнүлөт. Ар бир V параметри символдук, жолчолук, бүтүн же чыныгы типтеги өзгөрүлмө болуп саналат.

Read процедурасы (типтештирилген файлдар)	
Procedure Read(F V1, [, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдын элементин өзгөрүлмөгө (в перемнную) окуйт. F параметри – тексттик файлдан башка каалагандай типтеги

	файлга тиешелеш келүүчү файлдык өзгөрүлмө, ал эми ар бир V элементи F файлынын элементи ээ болгон типке ээ болгон өзгөрүлмө болуп саналат. Өзгөрүлмөгө болгон ар бир окууда файлдагы учурдагы позиция кийинки элементке жылып өтөт. Качан файлдын учурдагы позициясы файлдын аягында туруп калганда (б.а. качан Eof(F) ке барабар болгондо) файлдын кийинки элементин окууга болгон аракет ката болот. Файл ачылган болушу керек.

ReadLn процедурасы	
Procedure ReadLn([var F: Text;] V1, [, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Read функциясын аткарат, андан кийин файлдын кийинки жолчосуна өтөт. ReadLn процедурасы Read процедурасынын кеңейтилиш болуп тексттик файлдар үчүн аныкталган Read процедурасы аткарылгандан кийин ReadLn процедурасы кийинки жолчонун башталышына чейин өткөрүп жиберүүнү (пропуск) аткарат.Функция стандарттык кийирүү менен кошо тексттик файлдар үчүн гана иштейт. Файл кийрүү үчүн ачылган болушу керек.

Rename процедурасы	
Procedure Rename(var F; NewName);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Сырткы файлды кайра атайт. F параметри – каалагандай файлдык типке тиешелеш келүүчү файлдык өзгөрүлмө. NewName параметри жолчолук типтеги же PChar тибиндеги (эгер кеңейтирилген синтаксиске уруксат бар болсо) туюнтма. F өзгөрүлмөсү менен байланыштырылган сырткы файлга NewName параметри менен берилген жаңы ат ыйгарылат. F менен болгон андан кийинки амалдар жаңы аттагы сырткы файл менен аткарылат. Rename процедурасы ачылган файл үчүн пайдаланылбастыгы керек.

Reset процедурасы	
Procedure Reset(f [:File; Size: Word]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Бар болгон (сушествующий) файлды ачат. F параметри файлдын каалагандай тибине тиешелеш келүүчү файлдык өзгөрүлмө болуп эсептелет. Ал Assign процедурасынын жардамында сырткы файл менен байланыштырылган болушу керек. Size өлчөм параметри сөз узундугун ээ болгон сөзсүз талап кылынбаган туюнтма болуп F типтештирилбеген файл болгон учурда гана көрсөтүлүшү мүмкүн.

Rewrite процедурасы	
Procedure Rewrite(f [:File; Size: Word]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жаңы файлды түзөт жана ачат. F параметри файлдын каалагандай тибине тиешелеш келүүчү файлдык өзгөрүлмө болуп эсептелет. Ал Assign процедурасынын жардамында сырткы файл менен байланыштырылган болушу керек. Size өлчөм параметри сөз узундугун ээ болгон сөзсүз талап кылынбаган туюнтма болуп F типтештирилбеген файл болгон учурда гана көрсөтүлүшү мүмкүн. Эгерде бул параметр көрсөтүлбөсө, анда жазуунун өлчөмү 128 байт деп түшүнүлөт.

RmDir процедурасы	
Procedure RmDir(S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Куру камтылуучу каталогду жоготот. S параметри – бул жолчолук типтеги туюнтма. S жолчосу менен берилген кайрылуу жолуна ээ болгон камтылуучу каталог өчүрүлөт. Эгерде мындай кайрылуу жолу (путь доступа) жашабаса, бош эмес болсо же эгер, ал учурдагы каталогду берсе, анда кийирүү-чыгаруу катасы болуп өтөт.

Round функциясы	
Function Round(X: Real): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Чыныгы типтеги маанини бүтүн типтеги мааниге чейин тегеректейт. X параметр чыныгы типтеги туюнтма болуп эсептелет. Round функциясы x тин жакынкы бүтүн санга чейин тегеректелген узун бүтүн типтеги маанисин берет. Эгерде X эки бүтүн сандын туура (так) ортосунда болсо, анда жыйынтык болуп абсалюттук мааниси боюнча чоң болгон сан алынат. Эгерде тегерктелген x саны узун бүтүн типтин көрсөтүлүш диапазонуна тиешелеш келбесе, анда аткаруу учурунда ката пайда болот.

RunError процедурасы	
Procedure RunError[(ErrCode: Word)];	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Программанын аткарылышын токтотот жана аткарылуу этабынын катасын генрациялайт. RunError процедурасы Halt процедурасына окшош, бирок программанын токтошуна кошумча ал учурдагы операторду аткарылуу этабынын катасын генерациялайт. ErrCode параметри аткарылуу этабынын катасынын номерин көрсөтөт. Эгерде учурдагы модул коюлган DebugInformation (оңдоп-түзөө маалыматы) параметри менен компиляцияланган болсо жана силер программаны IDE интеративдик оңдоп-түзөөчүдөн ишке салсанар, анда Turbo Pascal тили RunError го кайрылууну кадимки аткарылуу этабынын катасы катары иштеп чыгат.

Seek процедурасы	
Procedure Seek(F, N: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдагы учурдагы позицияны берилген элементке которот. F параметри тексттик файлдан башка каалагандай файлдык өзгөрүлмө болуп эсептелет, ал эми N бүтүн типтеги туюнтма болуп эсептелет. F файлындагы

	учурдагы позиция N номерлүү элементке которулат. Файлдын биринчи элементинин номери 0 гө барабар. Файлды кеңейтириш үчүн анын акыркы элементинен кийин турган элементти издөөгө болот. Башкача айтканда оператор Seek(F, FileSize(F)) файлдагы учурдагы позицияны файлдын аягына которот. Бул процедураны тексттик файлдар үчүн пайдаланууга болбойт. Файл ачылган болушу керек.

SeekEof функциясы	
Function SeekEof[(var F: Text)]: Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файл үчүн «файлдын аягы» (end-of-file) абалын берет. Бардык пробелдерди, табуляция белгилерин жана жолчонун аяктоо эн-белгилерин өткөрүп жиберүү учурунан башка учурларда SeekEof функциясы толук Eof функциясына тиешелеш келет. Бул функцияны тексттик файлдан сандык маанилерди окуу чурунда пайдалануу ыңгайлуу жана аны тексттик файлдар үчүн гана пайдалануу керек. Файл ачылган болушу керек.

SeekEoln функциясы	
Function SeekEoln[(var F: Text)]: Boolean;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файл үчүн «жолчонун аягы» (end-of-line) абалын берет. Бардык пробелдерди, табуляция белгилерин жана жолчонун аяктоо эн-белгилерин өткөрүп жиберүү учурунан башка учурларда SeekEoln функциясы Eoln функциясына толук тиешелеш келет. Бул функцияны тексттик файлдан сандык маанилерди окуу чурунда пайдалануу ыңгайлуу жана аны тексттик файлдар үчүн гана пайдалануу керек. Файл ачылган болушу керек.

Seg функциясы	
Function Seg(X): Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Көрсөтүлгөн объект үчүн сегментти берет. X – параметри

	каалагандай өзгөрүлмө же процедуранын же функциянын идентификатору болуп эсептелет. Узундугу сөз болгон жыйынтык х үчүн сегменттин адреси болуп саналат.

SetTextBuf процедурасы	
Procedure SetTextBuf(var F: Text; var Buf [; Size: Word]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Тексттик файл үчүн кийирүү-чыгаруу буферин дайындайт. F параметри – тексттик файлга тиешелүү болгон файлдык өзгөрүлмө. Buf (буфер) параметри – каалагандай өзгөрүлмө, ал эми Size (өлчөм) параметри – сөз узундугундагы сөссуз эмес туюнтма. Бул процедураны Reset, Rewrite жана Append процедураларына болгон кайрылуудан кийин токтоосуз чакырууга мүмкүн болгону менен ал эч качан ачылбаган файл үчүн колдонулбастыгы керек. SetTextBuf процедурасын кийирүү-чыгаруу амалы аткарылып жаткан учурда ачылган файл үчүн чакыруу, буфердин алмашып кеткендигинене улам берилгендердин жоюлуп кетишине алып келиши мүмкүн.

Sin функциясы	
Function Sin(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин синусун берет. X параметри – бул чыныгы типтеги туюнтма. Жыйынтык болуп X тин синусу эсептелет. Бул маани радиандардагы бурчту берет деп эсептешет.

SizeOf функциясы	
Function SizeOf(X): Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргумент ээлеп турган байттардын санын берет. X параметри – бул өзгөрүлмөгө болгон шилтеме же типтин идентификатору. Бул функция X ээлеп турган эстин байттарынын санын берет.

SPtr функциясы	
Function SPtr: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	SP регистринин учурдагы маанисин берет. Узундугу сөзгө барабар болгон жыйынтык стек сегментинин ичинде стек көрсөткүчүнүн жылышы (смещение) болуп эсептелет.

Sqr функциясы	
Function Sqr(X): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин квадратын берет. X параметри – бул бүтүн же чыныгы типтеги туюнтма. Жыйынтык ошондой эле типке ээ болуп X тин квадратын б.а. X*X ти берет.

Sqrt функциясы	
Function Sqrt(X: Real): Real;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтен алынган квадраттык тамырды берет. X параметри чыныгы типтеги туюнтма болуп эсептелет. X тин квадраттык тамыры жыйынтык болот.

SSeg функциясы	
Function SSeg: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	SS регистринин учурдагы маанисин берет. Жыйынтык Word тибинде болуп стек сегментинин адреси болуп эсептелет.

Str процедурасы	
Procedure Str(X [: Size [: Dec]], var S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Сандык маанини анын жолчолук көрсөтүлүшүнө өзгөртүп

	түзөт. X параметри – бул бүтүн же чыныгы типтеги туюнтма. Size жана Dec параметрлери бүтүн типтеги туюнтма болушат. S параметри – бул жолчолук өзгөрүлмө. Бул функция X маанисин Size жана Dec форматтоо параметрлерине тиешелештикте анын жолчолук көрсөтүлүшүнө өзгөртүп түзөт. Жыйынтык жолчонун S параметрине сакталгандыгын эсепке албаганда, тексттик файлга жазылбастан, бул процедуранын аткарылышынын жыйынтыгы Write стандарттык процедурасына кайрылгандагынын өзүндөй эле болот.

Succ функциясы	
Function Succ(X);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин кийинки маанисин берет. X параметри – бул саналуучу типтеги туюнтма. Жыйынтык да ошол эле типке ээ болуп X тин кийинки маанисин берет.

Swap функциясы	
Function Swap(X);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Аргументтин чоң жана кичине байттарынын орундарын алмаштырат. X параметри – бул Integer же Word тибиндеги туюнтма.

Trunc функциясы	
Function Trunc(X: Real): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Чыныгы типтеги маанини бүтүн типтеги мааниге чейин тегеректейт. X параметри – бул чыныгы типтеги туюнтма. Trunc функциясы X тин нөлгө карай тегеректелген мааниси болгон узун бүтүн типтеги маанини берет. Эгерде X тин тегеректелген мааниси узун бүтүн типтин көрсөтүлүш диапазонуна тиешелеш келбесе, анда аткаруу этабынын катасы болуп өтөт.

Truncate процедурасы	
Procedure Truncate(var F);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Файлдын өлчөмүн файлдагы учурдагы позицияга чейин кесет. F параметри каалагандай типтеги файлдык өзгөрүлмө болуп саналат. F файлындагы учурдагы позициядан кийин жайгашкан бардык жазуулар өчүрүлөт жана учурдагы позиция файлдын аягы болуп калат (Eof(F) функциясы True маанисин кабыл алат). F файлы ачылган болушу керек.

TypeOf функциясы	
Function TypeOf(X): Pointer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Көрсөткүчтү объекттин виртуалдык усулдарынын таблицасына (VMT) кайтарат. X – бул объекттин идентификатору же объекттик типтеги нусканын идентификатору. Ар кандай учурда TypeOf функциясы объекттик типтин виртуалдык усулдарынын таблицасын берет. TypeOf виртуалдык усулдардын таблицасына ээ болгон гана объекттик типтерге колдонула алат. Башка бардык усулдар жыйынтыгында катаны берет.

UpCase функциясы	
Function UpCase(Ch: Char);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Символду жогорку регистирге өзгөртүп түзөт. Ch параметри символдук типтеги туюнтма болуп эсептелет. Символдук типтеги жыйынтык болуп Ch параметри менен берилген символдун жогорку регистирге өзгөртүлүп түзүлүшү болгон сивол эсептелет. a . . z диапазонунан сырткары жаткан символдор үчүн символдордун мааниси өзгөрүүсүз калат.

Val процедурасы	
Procedure Val(S String; var V; var Code: Integer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Жолчолук маанини анын сандык көрсөтүлүшүнө өзгөртүп түзөт. S параметри бул жолчолук типтеги туюнтма. V параметри болсо бүтүн же чыныгы типтеги өзгөрүлмө. Code параметри – бул белгиге ээ болгон бүткүл санды берүүчү бүтүн типтеги өзгөрүлмө. Val функциясы S жолчосун анын сандык көрсөтүлүшүнө өзгөртүп түзөт жана жыйынтыкты V га сактайт. Эгерде жолчонун кайсы бир жеринде уруксат берилбеген символ жолукса, анда анын номери Code параметринде сакталат. Андай болбогон учурда бул параметр нөлгө барабар. Алды жакта жолугуучу бош символдор (пробелдер) өчүрүлгөн болушу керек.

Write (тексттик файлдар) процедурасы	
Procedure Write([var F: Text;] V1[, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Бир же бир нече өзгөрүлмөлөрдөн бир же бир нече маанилерди тексттик файлга жазат. F параметри (эгар ал көрсөтүлсө) тексттик файлга тиешелеш болуучу файлдык өзгөрүлмө болуп эсептелет. Эгер ал жок болсо, анда стандарттык Output файлдык өзгөрүлмөсү пайдаланылат деп түшүнүлөт. Ар бир V параметри жазылып алынуучу (записываемый) параметр болуп эсептелет. Ар бир жазылып алынуучу параметр мааниси файлга жазылышы керек болгон чыгарылуучу туюнтманы өз ичине алып турат. Ар бир чыгарылуучу туюнтма символдук, бүтүн, чыныгы, жолчолук, таңгакталган жолчолук же бульдук типте болушу керек. Файл чыгаруу үчүн ачылган болушу керек.

Write (типтештирилген файлдар) процедурасы	
Procedure Write(F; V1[, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows

Арналышы:	Өзгөрүлмөнү файлдын элементине жазат. F параметри файлдык өзгөрүлмө, ал эми ар бир V параметри F файлынын элементинин тибиндей типке ээ болгон өзгөрүлмө болуп саналат. Өзгөрүлмөнү ар бир жазуудан кийин файлдагы учурдагы көрсөткүч кийинки элементке жазылат. Эгерде файлдын учурдагы позициясы файлдын аягында болуп калса (б.а. качан Eof(f) функциясы True маанисине ээ болсо), анда файл кеңейтилет.

Writeln процедурасы	
Procedure Write([var F: Text;] V1[, V2, … , Vn]);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим, Windows
Арналышы:	Write процедурасын аткарат, андан кийин файлга жолчонун аяктоо эн-белгисин (меткасын) жазат. Бул процедура Write процедурасынын тексттик файлдар үчүн кеңейтилиши болуп эсептелет. Write процедурасы аткарылгандан кийин Writeln процедурасы жолчонун эн-белгисин (каретканын кайтарылышы/жолчонун которулушу) жазат. Бул процедураны параметрлери жок чакырганда (Writeln(F)), файлга жолчонун аяктоо эн-белгиси жазылат. Файл ачылган болушу керек.

8.2. DOS модулунун процедуралары жана функциялары

DOS модулу – операциялык системанын бир кыйла көп пайдаланылуучу функцияларын жана файлдарды иштеп чыгуу функцияларын пайдалануу мүмкүнчүлүгүн берет.

Мисал катарында берилген жазуунун берилген каталогунда (же учурдагы) файлдын берилген атына жана файлдын атрибуттарынын жыйындысына тиешелеш болгон издөөнү жүргүзүүчү FindFirst жана FindNext процедураларын колдонууну карайлы:

```
Uses Dos, Crt;
  Var DirInfo: TSearchRec;
 Begin
   FindFirst('*.PAS', faArchive, DirInfo);
   While DosError = 0 Do
       Begin
         WriteLn(DirInfo.Name);
         FindNext(DirInfo);
       End;
End.
```

Эми төмөндө бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

DiskFree функциясы	
Function DiskFree(Disk: Byte): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Берилген дисктеги бош байттардын санын берет. Disk параметринин 0 деген маниси көрсөтүлбөгөн учурда (по умолчанию) аныкталуучу дискти, 1 деген маниси A дискин, 2 деген маниси B дискин ж.б.у.с. көрсөтөт. Эгерде дисктин номери аныкболбосо (недействительный), анда бул функция – 1 маанисин берет.

DiskSize функциясы	
Function (Disk: Byte): Longint;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Берилген дисктеги байттардын санын берет. Disk параметринин 0 деген маниси көрсөтүлбөгөн учурдагы (по умолчанию) аныкталуучу дискти, 1 деген маниси A дискин, 2 деген маниси B дискин ж.б.у.с. көрсөтөт. Эгерде дисктин номери аныкболбосо (недействительный), анда бул функция – 1 маанисин берет.

DosExitCode функциясы	
Function DosExitCode: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Камтылуучу процесс (подпроцесс) үчүн аяктоо кодун берет. Кичине байт аяктоо учурунда процесс тарабынан берилген кодду билдирет. Чоң байттын маанси 0 гө барабар болот, эгерде аяктоо нормалдуу болсо, Ctrl+C клавишаларын басуу менен болгон аяктоо учурунда 1 деген мааниге, түзүлүштүн катасынан улам болгон аяктоо кезинде 2 деген мааниге, эгерде Keep процедурасы тарабынан процесс аятаган болсо 3 деген мааниге ээ болот.

DosVersion функциясы	
Function DosVersion: Word;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим

Арналышы:	DOS тун версиясынын номерин берет. Жыйынтыктын кичине байты версиянын негизги номери, ал эми чоң байты – жардамчы номери болуп саналат. Мисалы, DOS 3.20 үчүн чоң байтка 3, ал эми кичине байтка 20 берилет.

EnvCount функциясы	
Function EnvCount: Integer;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	DOS операциялык чөйрөсүнүн аныктамасында кармалып турган жолчолордун санын берет. Ар бир мындай жолчо 'ӨЗГӨРҮЛМӨ=МААНИ' көрүнүшүнө ээ. Бул жолчолорду EnvStr функциясынын жардамында карап көрүүгө болот.

EnvStr функциясы	
Function EnvStr (Indx:integer): String;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	DOS операциялык чөйрөсүнүн берилген жолчосун берет. Бул жолчо 'ӨЗГӨРҮЛМӨ=МААНИ' көрүнүшүнө ээ. Биринчи жолчонун индекси болуп 1 эсептелет. Эгерде Indx бирден кичине же EnvCount ка караганда чоң болсо, анда EnvStr курук жолчону берет.

Exec процедурасы	
Procedure Exec(Pach, S: String);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Параметрлердин берилген жолчосун (командалык жолчону) берүү менен көрсөтүлгөн программаны аткарат. Программанын аты Pach параметринде, ал эми параметрдин жолчосу S параметринде кармалып турат. DOS тун ички командасын аткаруу үчүн COMMAND.COM файлын ишке салуу керек.

FExpand функциясы	
Function FExpand(Path: PathStr): PathStr;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Path маршрутунун параметри менен берилген файлдын атын файлдын толук атына чейин кеңейтет. Натыйжада алынган ат чоң тамгаларга өзгөртүлүп түзүлөт жана диск салгычтын тамгалык энбелгисин, кош чекитти, түпкү

	каталогко салыштырмалуу кайрылуу жолун жана файлдын атын кармап турат. “ . . ” жана “ . “ каталогдоруна болгон ички шилтемелер өчүрүлөт.

FindFirst процедурасы	
Procedure FindFirst(Path: String; Attr: Byte; var S: SearchRec);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын берилген атына жана файлдын атрибуттарынын жыйындысына тиешелеш болгон биринчи жазуунун берилген (же учурдагы) каталогунда издөөнү жүргүзөт. Path параметри каталогко карай жолду аныктайт. Attr параметри каралып жаткан файлдардын тизмесине атайын файлдардын (бардык кадимки файлдар менен катар) кошулушун аныктайт.

FindNext процедурасы	
Procedure FindNext(var S: SearchRec);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	FindFirst процедурасына болгон мурдагы кайрылууда көрсөтүлгөн файлдын аты жана атрибуттары менен дал келген кийинки жазууну берет. S параметри FindFirst процедурасына кайрылгандагыдай эле болушу керек (SearchRec тиби DOS модулунда баяндалат). DosError дун жардамында катанын кодун алуу мүмкүн. Мүмкүн болгон жалгыз код 18 болуп, ал файлдын жок боуп жаткандыгын көрсөтөт.

FSearch функциясы	
Function FSearch(Path: PathSize; L: String): PathSize;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Каталогдордун тизмесинде файлды издейт. Бул функция Path параметри менен берилген файлды L параметри менен берилген каталогдордун тизмесинде издөөнү жүргүзөт. Тизмедеги каталогдор үтүрлүү чекит менен ажыратылган болушу керек. Издөө дайыма учурдагы дисктин учурдагы каталогунан башталат. Алынуучу маани каталогдордун маршруттарынын бири менен файлдын атынын конкатенциясы, же файл табылбаган учурда – бош жолчо болуп эсептелет.

FSplit процедурасы	
Procedure FSplit(Path: PathSize; var Dir: DirStr; var Name: NameStr; var Ext: ExtStr;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын атын үч компонентке бөлөт. Path параметри менен берилген файлдын аты үч компонентке ажыратылат. Dir өзгөрүлмөсү үчүн дисктин тамгалык эн-белгисинин мааниси жана каталогко кайрылуунун маршруту бардык баштапкы жана акыркы тескери жантык сызык белгилери менен коюлат, Name өзгөрүлмөсү файлдын атынын мааниси, ал эми Ext өзгөрүлмөсүнө файлдын атынын кеңейтилиши алдындагы чекити менен кошо ыйгарылат. Жолчону түзгөн бул элементтердин ар бири куру болуп калышы да мүмкүн (эгерде Path тиешелүү элементти кармап турбаган болсо).

GetCBreak процедурасы	
Procedure GetCBreak(var Break: Boolean);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	DOS операциялык системасы тарабынан текшерилүүчү Ctrl+Break тин абалын берет. Өчүрүлгөн (false) абалда DOS операциялык системасы Ctrl+Break ты консольго, печаттоо түзүлүшүнө же коммуникациялык консольго чыгарууда гана текшерет. Туташтырылган учурда (True) системага болгон ар бир кайрылууда текшерүү жасалат.

GetDate процедурасы	
Procedure GetDate(var Year, M, Day, D: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Операциялык системада коюлган учурдагы күн-санакты (датаны) берет. Алынуучу маанилер: Year (жыл) – 1980..2099, M (ай) – 1..12, Day (күн) – 1..31, D (аптанын күнү) – 0..6 (0 жекшембиге туура келет) диапазондорунда болушат.

GetEnv функциясы	
Function GetEnv(Env: String): String;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Операциялык чөйрөнүн берилген өзгөрүлмөсүнүн

	маанисин берет. GetEnv функциясы берилген Env өзгөрүлмөсүнүн маанисин берет. Өзгөрүлмөнүн аты чоң тамгалар менен да көрсөтүлүшү мүмкүн, бирок ал барабардык белгисин (=) өзүнө алып турбашы керек. Эгерде операциялык чөйрөнүн берилген өзгөрүлмөсү (жашабаса) жок болсо, анда GetEnv функциясы бош жолчону берет.

GetFAttr процедурасы	
Procedure GetFAttr(var F; var Attr: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын атрибуттарын берет. F параметри файлдык өзгөрүлмөгө ыйгаруу аткарылышы керек, бирок ачылбаган болушу керек болгон файлдык өзгөрүлмөнү (типтештирилген, типтештирилбеген же тексттик файлга тиешелүү болгон) бериши керек. Attr атрибуттарын текшерүү DOS модулунда константалар көрүнүшүндө берилген алардын маскалары менен салыштыруу жолу менен аткарылат. F файлы ачылбаган болушу керек.

GetFTime процедурасы	
Procedure GetTime(var F; var Time: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын акыркы жазылыш датасын жана убактысын берет. F параметри эбак ага физикалык файл дайындалган жана ачылып коюлган файлдык өзгөрүлмө (типтештирилген, типтештирилбеген же тексттик файлга тиешелеш болгон) болушу керек. Time параметринде алынуучу убакыттын мааниси UnpackTime процедурасына кайрылуу жолу менен ачылып коюлган (распаковано) болушу мүмкүн. Каталардын коддорун DosError функциясынын жардамында алууга болот. Жалгыз мүмкүн болгон ката – бул 6 коду (файлдын уруксат берилбеген баяндагычы).

GetIntVec процедурасы	
Procedure GetIntVec(Int: Byte; var Vec: Pointer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Көрсөтүлгөн үзгүлтүктөр векторунун адресин берет. Int параметри үзгүлтүк векторунун номерин берет (0 дөн 255

	ке чейин), ал эми анын адреси Vec параметрине берилет. DOS тун корголгон режиминде жана Windows дун стандарттык жана корголгон режиминде GetIntVec берилген номердеги үзгүлтүктөр вектору менен коюлган режимдин үзгүлтүктөр векторун же корголгон режимдин өзгөчө абалдар (исключительные ситуации) векторун талап кылып алууга (запрос) болбойт.

GetTime процедурасы	
Procedure GetTime(var Hour, Min, Sec, Ssec: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Операциялык системага коюлган учурдагы убакытты берет. Алынуучу маанилер төмөнкүдөй маанилерди кабыл алышат: Hour (саат) – 0 дөн 23 ке чейин, Min (мүнөт) – 0 дөн 59 га чейин, Sec (секунда) – 0 дөн 59 га чейин жана Ssec (секунданын жүздүк үлүшү) – 0 дөн 99 га чейин.

GetVerify процедурасы	
Procedure GetVerify(var Flag: Boolean);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	DOS текшерүү желегинин абалын берет. Бул процедура DOS текшерүү желегинин абалын берет. Көтөрүлбөгөн желек (False) учурунда дискке жазууда текшерүү аткарылбайт. Көтөрүлгөн желек (True) учурунда жазуунун тууралыгын камсыз кылуу үчүн дискке болгон жазуулардын баарысы текшерилет.

Intr процедурасы	
Procedure Intr(IntNum: Byte; var Regs: Registers);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Берилген программалык үзгүлтүктү аткарат. IntNum параметри – бул программалык үзгүлтүктүн номери (0..255). Кирүүдө SP га же SS ке анык бир маанини талап кылган же чыгууда SP менен SS тин маанилерин өзгөрткөн программалык үзгүлтүктөр бул процедураны пайдалануу менен аткарыла албайт.

Keep процедурасы	
Procedure Keep(Code: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Keep процедурасы (ишти аяктоо жана эсте калтыруу) программанын ишин үзгүлтүккө учуратат жана аны резиденттик кылуу менен эсте калтырат. Программанын бүтүндөй коду, берилгендер сегменти, стек сегменти жана динамикалык бөлүштүрүлүүчү эс эсте резиденттик болуп кала берет. Ошондуктан компилятордун $M директивасынын жардамында динамикалык бөлүштүрүлүүчү эстин максималдык өлчөмүн берилгендигине ишенип коюу зарыл. Code параметри Halt стандарттык процедурасына берилүүчү аяктоо кодуна тиешелеш келет. Бул процедураны этияттык менен пайдалануу керек. Ишин аяктап эсте резиденттик болуп кала берген программалар жетишерлик татаал жана алар үчүн пайдаланыла турган башка эч нерсе каралган эмес.

MsDos функциясы	
Function MsDos(var Regs: Registers);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	DOS ту функционалдык чакырууну аткарат. MsDos процедурасына кайрылуунун жыйынтыгы Intr функциясына IntNo $21 үзгүлтүк номери менен кайрылгандагыдай эле болот. Registers параметри DOS модулунда баяндалган жазуу болуп саналат. Кирүүдө SP га же SS ке анык бир маанини талап кылган же чыгууда SP менен SS тин маанилерин өзгөрткөн программалык үзгүлтүктөр бул функцияны пайдалануу менен аткарыла албайт.

PackTime процедурасы	
Procedure PackTime(var DT: DateTime; var Time: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Дата жана убакыт жазуусун SetTime процедурасы тарабынан пайдаланылуучу дата менен убакыттын көрсөтүлүшүнүн узун бүтүн тибин конвентирлөө менен төрт байттык мааниге өзгөртүп түзөт. DateTime жазуусу DOS модулунда баяндалган. Бул жазуунун талаалары үчүн чектердин тууралыгын болгон текшерүү аткарылбайт.

SetCBreak процедурасы	
Procedure SetCBreak(Break: Boolean);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Бул процедура DOS тарабынан текшерилүүчү Ctrl+Break абалын коёт. Качан бул абал өчүрүлгөндө (выключено – False) DOS Ctrl+Break абалын консолго, печаттоо түзүлүшүнө же портко кийирүү/чыгаруу учурунда гана текшерет. Бул абал коюлган учурда (True) текшерүү ар бир системалык чыгаруу кезинде аткарылат.

SetDate процедурасы	
Procedure SetDate(var Y, M, D, Dw: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим,
Арналышы:	Операциялык системада учурдагы датаны коёт. Берилүүчү маанилер төмөнкүдөй болот: Y (жыл) – 1980..2099, M (ай) – 1..12, D (күнү, числосу) – 1..31, Dw (аптанын күнү) – 0..6 (0 жекшембиге туура келет). Эгерде дата туура көрсөтүлбөгөн болсо, анда суроо-талап (запрос) жокко чыгарылат.

SetFAtrr процедурасы	
Procedure SetDate(var Attr: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын атрибуттарын берет. F параметри – бул файлдык өзгөрүлмөгө ыйгаруу аткарылышы керек болгон, бирок айчылбаган болушу керек болгон файлдык өзгөрүлмө (типтештирилген, типтештирилбеген же тексттик файлга тиешелүү болгон) болуп саналышы керек. Атрибуттардын калыптанышы DOS модулунда константалар көрүнүшүндө берилген тиешелүү маскаларды кошуу жолу менен аткарылат. Файл ачылган боло албайт.

SetFTime процедурасы	
Procedure SetFTime(var F; Time: Longint);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Файлдын жазылышынын акыркы күнүн жана убактысын коёт. F параметри типтештирилген, типтештирилбеген же тексттик файлга тиешелүү болгон файлдык өзгөрүлмө

	болушу керек. Убакыттын Time параметрин PackTime процедурасына кайрылуунун жардамында калыптандырууга болот. Каталарды DosError функциясынын жардамында алууга болот. Болушу мүмкүн болгон жалгыз ката – бул 6 коду болуп саналат (кайрылууга мүмкүн болбогон файлдык канал). F файлы ачылган болушу керек.

SetIntVec процедурасы	
Procedure SetIntVec(IntNum: Byte; Vec: Pointer);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Берилген үзгүлтүк векторун көрсөтүлгөн адрес боюнча коёт. IntNum параметри үзгүлтүк векторунун номерин (0..255) берет, ал эми Vec анын адресин берет. Үзгүлтүктү иштеп чыгуу процедурасынын адресин алуу үчүн Vec параметри көбүкчө @ операторун пайдалануу менен берилет. DOS тун корголгон режиминде ошондой эле Windows дун стандарттык жана кеңейтирилген режиминде SetIntVec процедурасы корголгон режимдин берилген номердеги үзгүлтүк векторун коёт. SetIntVec процедурасын реалдык режимдин үзгүлтүктөр векторун модификациялоо же корголгон режимдин өзгөчө кырдаалдар векторлорун модификациялоо үчүн пайдалнууга болбойт.

SetTime процедурасы	
Procedure SetTime(var Hour, Min, Sec, Ssec: Word);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	Операциялык системада учурдагы убакытты коёт. Алынуучу параметрлер төмөнкүдөй маанилерди кабыл алат: Hour (саат) – 0 дөн 23 ке чейин, Min (мүнөт) – 0 дөн 59 га чейин, Sec (секунда) – 0 дөн 59 га чейин жана Ssec (секунданын жүздүк үлүшү) – 0 дөн 99 га чейин.

SetVerify процедурасы	
Procedure SetVerify(Verify: Boolean);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	SetVerify процедурасы DOS операциялык системасында текшерүү желегинин абалын коёт. Көтөрүлбөгөн (False) абалда дискке жазуу амалын текшерүү аткарылбайт.

	Желектин көтөрүлгөн (True) абалында дискке жазуу амалдары учурда амалдын аткарылышынын тууралыгын текшерүү аткарылат.

SwapVectors процедурасы	
Procedure SwapVectors;	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	System модулундагы SaveIntXX көрсөткүчтөрүнүн камтымдарынын жана үзгүлтүктөр векторунун учурдагы камтымдарынын орундарын алмаштырат. SwapVectors процедурасы адатта Exec ти чакыруунун алдында тикеден-тике жана чакырылгандан кийин тикеден-тике чакырылат. Ошентип Exec ти чакыруу менен аткарылуучу процесс учурдагы процесс тарабынан кошулуучу жана тескерисинче эч кандай үзгүлтүктөрдү иштеп чыгууларды пайдаланбайт.

UnPackTime процедурасы	
Procedure UnPackTime(Time: Longint; var DT: DateTime);	
Пайдаланылуучу режим:	Реалдык режим, корголгон режим
Арналышы:	GetTime, FindFirst жана FindNext процедуралары менен алынуучу датанын жана убакыттын көрсөтүлүшүнүн таңгакталган узун бүтүн тиби болуп эсептелген төрт байттык маанини ачылып (чечилип) коюлган DateTime жазуусуна өзгөртүп түзөт. DateTime жазуусу DOS модулунда баяндалган. Time убакытынын талаалары үчүн диапазондорду текшерүү аткарылбайт.

8.3. WinDOS модулунун процедуралары жана функциялары

WinDOS модулу – Windows операциялык системанын көп пайдаланылуучу функцияларын жана файлдарды иштеп чыгуу функцияларын пайдалануу мүмкүнчүлүгүн берет. Стандарттык Паскалда бул модулдун эч бир программасы аныкталган эмес.

Мисал катарында төмөнкү программаны карайлы:

```
Uses WinDos;
Var F : Text;
      H, M, S, Hund : Word; { GetTime үчүн}
      FTime : Longint; { Get/SetFTime үчүн }
      Dt : TDateTime; { Pack/UnpackTime үчүн }
Function LeadingZero(W : Word) : String;
 Var S : String;
Begin
  Str(W:0, S);
   If Length(S) = 1 Then S:='0'+S;
   LeadingZero:=S;
End;
Begin
  Assign(F, 'TEST.TXT');
  GetTime(H, M, S, Hund);
   ReWrite(F); { Жаңы файл түзүү}
   GetFTime(F, FTime); { Файл түзүлгөн датаны алуу}
   WriteLn('Файл', LeadingZero(H), ' түзүлгөн:',
   LeadingZero(M), ':', LeadingZero(S));
   UnpackTime(FTime, Dt);
  With Dt Do
    Begin
      WriteLn('Файлдын түзүлгөн убакыты ',
      LeadingZero(Hour), ':', LeadingZero(Min), ':',
      LeadingZero(Sec));
Hour:=0;
Min:=1;
Sec:=0;
PackTime(Dt, Ftime);
WriteLn('Файлдын жаңы атрибуту 00:01:00 орнотулду');
Reset(F); { Файлды окууга карата ачабыз }
{ (Close убакыт атрибутун жанылайт) }
SetFTime(F, FTime);
```

```
End;
 Close(F);{ Файлды жабуу }
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

CreateDir процедурасы	
Procedure CreateDir(Dir: PChar);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Жаңы камтылуучу каталогду (подкаталог) түзөт. Түзүлүүчү камтылуучу каталог Dir параметри менен берилет. Каталар DosError до билдирилет. SYSTEM модулундагы MkDir процедурасы CreateDir ге окшош функцияны аткарат, бирок аргумент катары нөл менен аяктоочу жолчону эмес String-жолчону кабыл алат.

DosVersion функциясы	
Function DosVersion:Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	DOS тун версиясынын номерин берет. Жыйынтактын кичине байты версиянын негизги номери, ал эми чоң байты – жардамчы номери болуп саналат.

FileExpand функциясы	
Function FileExpand(Dest, Name: Pchar): Pchar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Name параметри менен берилген файлдын атын файлдын толук атына чейин кеңейтет. Жыйынтыкта алынган ат жогорку регистрдин тамгаларына өзгөртүп түзүлөт жана диск салгычтын эн-белгисин (меткасын), кош чекитти, түпкү каталогко салыштырмалуу маршрутту жана файлдын атын кармап турат. “. .” жана “.” каталогдоруна болгон ички шилтемелер жоготулат. Файлдын атынын компоненттери жана кеңейтилиши тиешелүү түрдө 8 жана 3 символго чейин кесилип ташталат. Алынуучу маани Dest те сакталат.

FileSearch функциясы	
Function FileSearch(Dest, Name, List: Pchar): Pchar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	List параметри менен берилген каталогдордун тизмесинде Name деген атка ээ болгон файлды издейт. List тизмесиндеги каталогдор DOS тун PATH командасында берилген каталогдорго окшош үтүрлүү чекит менен ажыратылган болушу керек. Издөө дайыма учурдагы дисктин учурдагы каталогунан башталат. Эгерде файл табылган болсо, анда Search функциясы Dest ке каталогдордун маршруту менен файлдын атынын конкатенциясын жазат. Андай болбогон учурда Dest ке бош жолчону жазат. Алынуучу маани Dest те болот. Dest жана Name бир эле нерсеге шилтеме жасабаган болушу керек.

FileSplit функциясы	
Function FileSplit(Path, Dir, Name, Ext: Pchar): Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Path параметри менен берилген файлдын толук атын үч компонентке ажыратат. Dir дисктин тамгалык эн-белгисин жана аягында келүүчү тескери жантык сызыгы менен каталогдун маршрутун, Name – файлдын атын, ал эми Ext чекит менен башталуучу файлдын атынын кеңейтилишин кармап турат. Эгерде кайсы бир параметр nil маанисин кармап турса, анда маршруттун ошол тиешелүү бөлүгү жазылбайт. Эгерде маршрут берилген компонентти кармап турбаса, анда компоненттин кайтарылып алынуучу жолчосу (возвращаемая строка) бош болот. Dir, Name жана Ext парметрлеринде жолчолордун максималдык узундуктары fsDirectory, fsFileName жана fsExtension константалары менен аныкталат.

FindFirst процедурасы
Procedure FindFirst(Path: String; Attr: Byte; var S: SearchRec); **{DOS}**
Procedure FindFirst(Path: String; Attr: Byte; var S: TSearchRec); **{Windows}**

Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Файлдын берилген атына жана файлдын атрибуттарынын жыйындысына тиешелеш биринчи жазуунун берилген (же учурдагы) каталогунда издөөнү жүргүзөт. Path параметри издөөнүн маршрутун аныктайт. Мисалы, “*.*”

FindNext процедурасы	
Procedure FindNext(var S: SearchRec); {DOS} **Procedure FindNext(var S: TSearchRec); {Windows}**	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	FindFirst процедурасына болгон мурдагы кайрылууда көрсөтүлгөн файлдын аты жана атрибуттары менен дал келген андан кийинки жазууну кайтарып берет. S параметри FindFirst процедурасына кайрылгандагыдай эле болушу керек (SearchRec тиби DOS модулунда, ал эми TSearchRec тиби Windows модулунда баяндалат). DosError дун жардамында катанын кодун алууга болот. Болушу мүмкүн болгон жалгыз код – бул 18 болуп, ал файлдын жок боуп жаткандыгын көрсөтөт.

GetArgCount функциясы	
Function GetArgCount: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Командалык жолчодо программага берилген параметрлердин санын берет.

GetArgStr функциясы	
Function GetArgStr(Dest:PChar; Index:Integer;MaxLent: Word):PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Командалык жолчонун Index менен берилген параметрин берет. Эгерде Index нөлдөн кичине же GetArgCount тан чоң чоң мааниге ээ болсо, анда каралып жаткан функция бош жолчону кайтарат. Кайтарылуучу жолчонун максималдык узундугу MaxLen параметри менен аныкталат.

GetCurDir функциясы	
Function GetCurDir(Dir: PChar; Drive: Byte): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Берилген дисктеги учурдагы каталогду берет. Dir де алынуучу жолчо ар дайым дисктин тамгалык эн-белгисинен, кош чекиттен жана тескери жантык сызыктан башталат. Drive=0 деген учурдагы дискти, 1-деген A дискин, 2-деген B дискин ж.б.у.с. көрсөтөт. Алынуучу маани Dir ге жазылат. Каталар DosError до берилет.

GetDate процедурасы	
Procedure GetDate(var Year, M, Day, D: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Операциялык системада коюлган учурдагы датаны берет. Алынуучу маанилер: Year (жыл) – 1980..2099, M (ай) – 1..12, Day (күн) – 1..31, D (аптанын күнү) – 0..6 (0 жекшембиге туура келет) диапазондорунда болушат.

GetEnvVar функциясы	
Function GetEnvVar(Env: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	GetEnvVar функциясы операциялык чөйрөнүн берилген өзгөрүлмөсүнүн маанисине көрсөткүчтү кайтарат. Мисалы, VarName менен берилген операциялык чөйрөнүн жазуусунда барабардык (=) белгисинен кийин турган биринчи символго көрсөткүчтү кайтарат. Өзгөрүлмө жогорку же төмөнкү регистрде жазылган болушу мүмкүн, бирок барабардык символун өзүнө алып турбашы керек. Эгерде операциялык чөйрөнүн берилген өзгөрүлмөсү жок болсо (жашабаса), анда GetEnvVar функциясы nil маанисин берет.

GetFTime процедурасы	
Procedure GetFTime(var F; Var Time: Longint);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows

Арналышы:	Файлдын акыркы жазылыш күнүн (датасын) жана убактысын берет. F параметри физикалык файлга эбак дайындалган жана алынып коюлган файлдык өзгөрүлмө (типтештирилген, типтештирилбеген же тексттик файлга тиешелеш келүүчү) болуп эсептелет. Time параметринде алынуучу убакыттын мааниси UnpackTime процедурасына кайрылуу жолу менен да алынган болушу мүмкүн. Каталардын коддорун DosError функциясынын жардамында алууга болот. Болушу мүмкүн болгон жалгыз ката – 6 коду (файлдын уруксат берилбеген баяндагычы).

GetVerify процедурасы	
Procedure GetVerify(var Flag: Boolean);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бул процедура DOS тун текшерүү желегинин абалын берет. Желек түшүп турганда (False) дискке жазуу учурунда текшерүү аткарылбайт. Желек көтөрүлгөн учурда (True) жазуунун тууралыгын камсыз кылуу үчүн бардык дискке жазуу амалдары текшерилет.

Intr процедурасы	
Procedure Intr(IntNum: Byte; var Regs: Registers); {DOS} **Procedure Intr(IntNum: Byte; var Regs: TRegisters); {Windows}**	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Берилген программалык үзгүлтүктү аткарат. IntNum параметри – бул программалык үзгүлтүктүн номери (0..255). TRegisters болсо WinDos модулунда аныкталган жазуу болуп эсептелинет. SP же SS тин анык бир маанилерин өзгөртүү талап кылынган программалык үзгүлтүктөр бул процедураны пайдалануу менен аткарыла алат.

RemoveDir процедурасы	
Procedure RemoveDir(Dir: PChar);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бош (куру) камтылуучу каталогду (подкаталогду) өчүрөт. Dir параметри менен берилген маршрут боюнча

	камтылуучу каталогду өчүрөт. Каталар жөнүндө (жашабаган каталог же куру эмес каталог дегенге окшогон) маалыматтар DosError өзгөрүлмөсүндө берилет.

SetCurDir процедурасы	
Procedure SetCurDir(Dir: PChar);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Учурдагы каталогду Dir параметри менен берилген маршрутка өзгөртөт. Эгерде Dir дисктин тамгалык эн-белгисин берсе, анда учурдагы диск да өзгөрөт. Каталар DosError до билдирилет.

SetFTime процедурасы	
Procedure SetFTime(var F, Time: Longint);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Файлдын акыркы жазылыш күнүн (датасын) жана убактысын коёт. F параметри эбак ачылып коюлган типтештирилген, типтештирилбеген же тексттик файлга тиешелеш болгон файлдык өзгөрүлмө болушу мүмкүн. Time убакыттын параметрин PackTime процедурасына кайрылуунун жардамында да алууга болот. Каталарды DosError дун жардамында алуу мүмкүн. Бул жерде болушу мүмкүн болгон жалгыз ката – бул 6 коду (уруксат берилбеген (недопустимый) файлдык канал).

SetIntVec процедурасы	
Procedure SetIntVec(IntNum: Byte; Vec: Pointer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Көрсөтүлгөн адрес боюнча берилген үзгүлтүк векторун берет. IntNum параметри үзгүлтүк векторунун номерин (0..255) берет, ал эми Vec анын адресин берет. Үзгүлтүктөрдү иштеп чыгуу процедурасынын адресин алуу үчүн Vec параметри көбүнчө @ операторун пайдалануу менен берилет.

UnPackTime процедурасы	
Procedure UnPackTime(Time: Longint; var DT: DateTime);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Күн (дата) менен убакыттын таңгакталган (упакованный) узун бүтүн типтеги төрт байттык көрсөтүлүшүн (GetTime, FindFirst жана FindNext процедуралары менен алынган) DateTime (дата/убакыт) чечилген (распакованный) жазуусуна өзгөртүп түзөт.

8.4. CRT модулунун процедуралары жана функциялары

Crt (Cathode-Ray Tube – электрондук-нур түтүкчөсү) модулу экрандын иштөө режимдерин башкаруу, клавиатуранын кеңейтирилген коддорун окуу, түстөрдү пайдалануу, терезе жана үн менен иштөө камтылуучу программаларын кармап турат. Ал чыгаруунун экранга багытталган, түздөн-түз BIOS ко багытталган же видеоэске багытталган программаларын жазууга мүмкүнчүлүк берет. Бул модулду IBM PC же аны менен биргелешкен компьютерлерде иштөө үчүн арналган программаларда гана пайдаланууга болот.

Бул модул Crt.TPU жана Crt.TPP файлдарында жайгашкан ушул модулду чакырып иштетүүнүн мисалы катарында төмөнкү программаны келтиребиз.

```
Uses Crt;
Var
     X, Y: Byte;
 Begin
   TextBackground(Black); {арткы түстү карартуу}
   ClrScr; {экранды тазалоо}
 Repeat
  X:=Succ(Random(80));
  Y:=Succ(Random(25)); {кокустук сандар}
   Window(X, Y, X + Random(10), Y + Random(8));
   TextBackground(Random(16)); {арткы түстү берүү}
   ClrScr;
 Until KeyPressed;
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

AssignCrt процедурасы	
Procedure AssignCrt(var F: text)	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Тексттик файлды CRT түзүлүшү (терминал) менен байланыштырат. AssignCrt процедурасы стандарттык Assign проседурасынын так өзүндөй иштейт, болгону файылдын аты көрсөтүлбөйт. Анын ордуна тексттик файл Crt түзүлүшү (терминал) менен байланыштырылат.

ClrEol процедурасы	
Procedure ClrEol;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Курсордун позициясынан баштап жолчонун акырына чейинки бардык символдорду өчүрөт. Курсор бул учурда ордунан которулбайт. Символдордун бардык позициялары бош символдор (пробелдер) менен толтурулат. Мында тексттик атрибуттардын учурдагы аныктамасы пайдаланылат. Ошентип, эгерде TextBackGround процедурасында кара түс берилбеген болсо, анда курсордон баштап оң жаккы чекке чейинки позицияларда экран фондун түсүн алат.

ClrScr процедурасы	
Procedure ClrScr;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Экранды тазалайт жана курсорду экрандын сол жогорку бурчуна жайгаштырат. Символдордун бардык позициялары бош символдор (пробелдер) менен толтурулат. Мында тексттик атрибуттардын учурдагы аныктамасы пайдаланылат. Ошентип, эгерде TextBackColor процедурасы үчүн кара түс берилбеген болсо, анда бүтүндөй экран үчүн фондун түсү коюлат. Ушул нерсе ClrEol, InsLine жана DelLine процедураларынын жардамында тазаланган символдордун позицияларына да, экранды баракталанда түзүлүп калуучу бош жолчолорго да тиешелүү.

Delay процедурасы	
Procedure Delay(Msec: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Миллисекунддардын берилген санына чейинки убакытка кармап турууну (задержка) аткарат. Msec параметри күтүү интервалындагы миллисекунддардын санын берет.

DelLine процедурасы	
Procedure DelLine;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Курсордун позициясында турган жолчону өчүрөт. Курсор жайгашып турга жолчо өчүрүлөт. Бул учурда берилген жолчодон төмөн жайгашкан бардык жолчолор бир жолчого жогору жылышат (бул үчүн кийирүү-чыгаруунун базалык системасынын экранды баракто программасы пайдаланылат). Экрандын төмөн жагына жаңы жолчо кошулат.

HighVideo процедурасы	
Procedure HighVideo;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Символдор үчүн жогорулатылган ачыктыкты (повышенная яркость) коёт. Crt модулунда TextAttr байттык өзгөрүлмөсү бар болуп ал сүрөттөлүштүн учурдагы атрибуттарын сактоо үчүн пайдаланылат. HighVideo процедурасы символдордун түсү үчүн TextAttr өзгөрүлмөсүндө ачыктык битин коёт. Ошентип 0..7 түстөрү 8..15 түстөрүнө чагылдырылат.

InsLine процедурасы	
Procedure InsLine;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Курсордун позициясынан баштап бош жолчону сыйлыгыштырып коёт (вставляет). Кошулган жолчодон төмөн жайгашкан бардык жолчолор бир жолчого төмөн жылышат, ал эми эң төмөңкө жолчо экрандан жоголот. (Мында кийирүү-чыгаруунун базалык системасынын экранда сүрөттөлүштү тегеретүү (прокрутка) программасы пайдаланылат).

GotoXY процедурасы	
Procedure GotoXY(X, Y: Byte);	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсорду координаталары берилген чекитке позициялайт. Курсорду учурдагы терезенин X жана Y координаталары менен берилген позициясына жылдырат. (X – мамычаны, Y – жолчону берет). Жогорку сол бурч (1,1) координаталары менен берилет. Эгерде уруксат берилбеген (берүүгө болбогон) координаталар берилип калса, анда процедурага кайрылуу жокко чыгарылат.

KeyPressed функциясы	
Function KeyPressed: Boolean;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Эгерде клавиатурада клавиша басылган болсо True маанисин, андай болбогондо False маанисин кайтарат. Символ (символдор) клавиатуранын буферинде калат. Бул функция Shift, Alt, NumLock ж.б.у.с. регистрди которуу клавишаларын тааныбайт (не распознает). Клавишаны ReadKey функциясынын жардамында окууга болот.

LowVideo процедурасы	
Procedure HighVideo;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Символдор үчүн басаңдатылган ачыктыкты (пониженную яркость) коёт. Crt модулунда TextAttr байттык өзгөрүлмөсү бар болуп ал сүрөттөлүштүн учурдагы атрибуттарын сактоо үчүн пайдаланылат. LowVideo процедурасы TextAttr өзгөрүлмөсүнүн символдорунун түсү үчүн жарыктандыруу (подсветка) битин тазалайт. Ошентип 8..15 түстөрү 0..7 түстөрүнө чагылдырылат.

NormVideo процедурасы	
Procedure NormVideo;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Программаны жүктөгөн жана ишке салган учурда курсордун позициясы үчүн текстин кадимки атрибутун тандайт. Crt блогунда учурдагы атрибутту сактоо үчүн

	пайдаланылуучу байттык өзгөрүлмө (TextAttr) бар. Аталган процедура TextAttr үчүн программа ишке салынганга чейин ээ боло турган маанини калыбына келтирет.

NoSound процедурасы	
Procedure NoSound;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Ички динамикти өчүрөт.

ReadKey функциясы	
Function ReadKey: Char;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Клавиатурадан символду окуйт. Окулуучу символ экранда чагылдырылбайт. Эгерде ReadKey функциясына кайрылуунун алдында KeyPressed функциясы True маанисине ээ болсо, анда символ токтоосуз окулат, антпесе функция клавишанын басылышын күтөт.

Sound процедурасы	
Procedure Sound(Herz: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Ички динамикти жандырат (включает). Herz параметри генерацияланууч символдун герцтердеги жыштыгын берет. Бул сигнал NoSound процедурасына кайрылуу менен айкын түрдө өчүрүлмөйүнчө чыгып тура берет.

TextBackGround процедурасы	
Procedure TextBackGround(Color: Byte);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Фондун түсүн коёт. Color параметри 0..7 диапазонундагы бүтүн типтеги туюнтма болуп, ал түстүн биринчи сегиз константасынын бирине тиешелеш келет.

TextColor процедурасы	
Procedure TextColor(Color: Byte);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Символдордун түсүн тандайт. Color параметри 0..15

	диапазонундагы бүтүн типтеги туюнтма болуп, ал Crt модулунда аныкталган түстүк константалардын бирөөсүнө тиешелеш келет.

TextMode процедурасы	
Procedure TextMode(Mode: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Көрсөтүлгөн тексттик режимди тандайт. TextMode процедурасы чакырылган кезде учурдагы терезе ташталынып жиберилип (сбрасывается), учурдагы терезе катары бүтүндөй экран каралат. DirectVideo үчүн True маани си, CheckShow үчүн да True маани си коюлат жана учурдагы тексттик атрибут NormVideo процедурасына кайрылууга тиешелеш келген нормалдык абалга келтирилет. Ал эми учурдагы видеорежим LostMode да сакталат. Программанын инициализацияланышы учурунда LostMode учурдагы видеорежимдин маанисин алат.

WhereX функциясы	
Function WhereX: Byte;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Курсордун учурдагы позициясы үчүн учурдагы терезеге салыштырмалуу X координатасын берет.

WhereY функциясы	
Function WhereY: Byte;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Курсордун учурдагы позициясы үчүн учурдагы терезеге салыштырмалуу Y координатасын берет.

Window процедурасы	
Procedure Window(X1,Y1,X2,Y2: Byte);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Экранда тексттик терезени аныктайт. X1,Y1 параметри терезенин жогорку сол бурчунун, ал эми X2,Y2 терезенин төмөнкү оң бурчунун координаталары болуп эсептелинет. Экрандын жогорку сол бурчу (1,1) координатасына туура келет. Тексттик терезенин минималдык өлчөмү – бир жолчо, бир мамычага ээ болот. Эгерде координаталар

	уруксат берилбеген (недопустимый) мааниде болуп калса анда Window процедурасына кайрылуу жокко чыгарылат. 80 символдук режимде көрсөтүлбөгөн учурда (по умолчанию) (1,1,80,25) терезеси, ал эми 40 символдук режимде (1,1,40,25) терезеси аныкталып, ал бүтүндөй терезеге тиешелеш келет.

8.5. WinCRT модулунун процедуралары жана функциялары

Төмөндө WinCRT модулунун жалпы процедура жана функциялары кыскача баяндалат.

AssignCrt процедурасы	
Procedure AssignCrt(var F: text)	
Пайдаланылуучу режим:	Windows
Арналышы:	Тексттик файлды CRT түзүлүшү менен байланыштырат. AssignCrt процедурасы Assign стандарттык проседурасынын так өзүндөй эле иштейт, болгон айрымасы – файылдын аты көрсөтүлбөйт. Анын ордуна тексттик файл Crt түзүлүшү менен(терминал менен) байланыштырылат. Кийинки Write жана Writeln операциалары иштетилген учурда файл Crt терезеси чыгарылат, ал эми Read жана Readln операциалары CRT терезесинен файлды окуйт.

ClrEol процедурасы	
Procedure ClrEol;	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсор турган орундан баштап жолчонун акырына чейинки бардык символдорду өчүрөт. Бул учурда курсор ордунан жылбайт.

ClrScr процедурасы	
Procedure ClrScr;	
Пайдаланылуучу режим:	Windows
Арналышы:	Экранды тазалайт, аны фондун түсү менен толтурат жана курсорду экрандын сол жогорку бурчуна орнотот. Символдордун бардык позициялары пробелдер менен

	толтурулат. Мында тексттик атрибуттардын учурдагы аныктамасы пайдаланылат.

CursorTo процедурасы	
Procedure CursorTo(X,Y: Integer);	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсорду виртуалдык экрандын берилген X жана У кординаталарына жылдырат. Жогорку сол бурч (0, 0) кординаталарына тиешелеш келет.

DoneWithCrt процедурасы	
Procedure DoneWithCrt;	
Пайдаланылуучу режим:	Windows
Арналышы:	CRT терезесин жок кылат (эгер ал жашаса). Программанын аяктоосу алдында DoneWithCrt процедурасын чакыруу CRT терезесин активдүү эмес абалга өтүшүн алдын алат, ошону менен пайдалануучуга бул терезени жабуу талап кылынбайт.

GotoXY процедурасы	
Procedure GotoXY(X, Y: Byte);	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсорду берилген координаталар боюнча алынуучу чекитке позициялайт. Жогорку сол бурч (1,1) координаталары менен берилет. Cursor өзгөрүлмөсү бул учурда (x-1, y-1) чекитине коюлат, анткени анда (1,1) чекитине эмес (0,0) чекитине салыштырмалуу позиция сакталып турат.

InitWinCrt процедурасы	
Procedure InitWinCrt;	
Пайдаланылуучу режим:	Windows
Арналышы:	Crt терезесин түзөт (Эгер ал түзүлө элек болсо). Read, ReadLn, Write же WriteLn процедуралары Crt га ыйгарылган файл менен автоматтык түрдө InitWinCrt процедурасын чакырышат.

KeyPressed функциясы	
Function KeyPressed: Boolean;	
Пайдаланылуучу режим:	Windows
Арналышы:	Эгерде клавиатурада клавиша басылган болсо True маанисин, андай болбогондо False маанисин кайтарат. Символ (символдор) клавиатуранын буферинде кала берет. Бул функция регистрди алмаштыруу клавишаларын – Shift, Alt, NumLock ж.б.у.с. клавишаларын тааныбайт (не распознает). Мындай клавишаны ReadKey функциясынын жардамында окууга болот.

ReadBuf функциясы	
Function ReadBuf(Buffer: PChar; Count: Word): Word;	
Пайдаланылуучу режим:	Windows
Арналышы:	Crt терезесинен жолчону окуйт. Buffer параметри Count ко чейинки символдор жайгаштырылышы мүмкүн болгон жолчо буферин көрсөтүп турат. Бардыгы болуп Count-2 ге чейин символдорду киргизүүгө болот, ал эми пайдалануучу тарабынан Enter клавишасы басылганда автоматтык түрдө жолчонун бүтүш маркери кошулат (#13 жана #10). Эгерде CheckEOF=True болсо, анда пайдалануучу жолчонун кийрилишин Ctrl+Z клавишасын басуу менен аяктай алат. Бул учурда жолчого файлдын бүтүш маркери (#26) кошулат. Кайтарылып берилүүчү маани болуп файлдын бүтүш маркерин кошо эсептегендеги окулуучу символдордун саны эсептелет.

ReadKey функциясы	
Function ReadKey: Char;	
Пайдаланылуучу режим:	Windows
Арналышы:	Клавиатурадан символду окуйт. ReadKey функциясы ASCII стандарттык коддору менен гана иштейт. Клавишалардын кеңейтирилген коддору иштетиле албайт.

ScrollTo процедурасы	
Procedure ScrollTo(X, Y: Integer);	
Пайдаланылуучу режим:	Windows

Арналышы:	Жогорку сол бурчтун (X, Y) координаталарына ээ болгон виртуалдык экрандын сүрөттөлүшүн чыгаруу үчүн Crt терезесин түрүү аракетин (прокручивание) аткарат.

TrackCursor процедурасы	
Procedure TrackCursor;	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсордун көрүнүмдүүлүгүн (видимость) камсыз кылуу үчүн Crt терезесин түрөт.

WhereY функциясы	
Function WhereY: Byte;	
Пайдаланылуучу режим:	Windows
Арналышы:	Учурдагы терезенин координаталар башталышына салыштырмалуу курсордун учурдагы позициясынын Y координатасын берет. Бул маани 1 ден баштап эсептелинген жана Cursor.Y+1 маанисине тишелеш болгон координата болот.

WriteBuf функциясы	
Function WriteBuf(Buffer: PChar; Count: Word): Word;	
Пайдаланылуучу режим:	Windows
Арналышы:	Символдор блогун Crt терезесине жазат. Buffer параметри блоктун биринчи символун көрсөтөт, ал эми Count жазылуучу символдордун санын аныктайт.

WriteChar процедурасы	
Procedure WriteChar(Ch: Char);	
Пайдаланылуучу режим:	Windows
Арналышы:	Курсордун учурдагы позициясындагы жалгыз символду Crt терезесине жазат.

8.6. WinPRN модулунун процедуралары жана функциялары

WinPrn модулу колдонуучунун программасынан туруп анын тандоосуна жараша печатка жиберүүнү уюштурат.

Ошондой эле бул модул Windows программасында туруп текстти печаттоого мүмкүнчүлүк берет. Кандайдыр бир файлды принтерге жиберүүчү төмөнкү программаны карайбыз:

```
Uses WinPrn, WinCrt;
Var
      Source, Prn : Text;
      Name : Array [0..80] Of Char;
      Line : String;
 Begin
{ Файлдын атын окуу }
  Write('Введите имя распечатываемого файла : ');
  ReadLn(Name);
  Assign(Source, Name);
  Reset(Source);
  AssignDefPrn(Prn); { Файлды стандарттык принтер
менен байланыштыруу }
  TitlePrn(Prn, Name);
  ReWrite(Prn);
 WriteLn('Печатаю файл : ',Name); {Файлды печаттоо}
 While Not Eof(Source) Do
Begin
ReadLn(Source, Line);
WriteLn(Prn, Line);
{ Эгер ESCбасылса, анда печатты токтотуу }
If KeyPressed And (ReadKey = #27) Then
Begin
AbortPrn(Prn);
Break;
End;
End;
 Close(Source); { Файлды жабуу }
 Close(Prn);
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

AbortPrn процедурасы	
Procedure AbortPrn(var F: Text);	
Пайдаланылуучу режим:	Windows
Арналышы:	Текстти печаттоону токтотот. AbortPrn процедурасы тексттин бардык печатталбаган бөлүктөрүн таштап жиберет. Печатталуучу файл AssignPrn же AssignDefPrn процедураларынын жардамында дайындалат.

AssignDefPrn процедурасы	
Procedure AssignDefPrn(var F: Text);	
Пайдаланылуучу режим:	Windows
Арналышы:	Атайын көрсөтүлбөгөн учурда (по умолчанию) пайдаланылуучу принтерге печаттоо үчүн файлды дайындайт. AssignDefPrn процедурасы AssignPrn процедурасын nil маанисине коюлган Devise, Driver жана Port параметрлери менен чакырат. Бул Windows да атайын көрсөтүлбөгөн учурда пайдаланылуучу принтерди F файлы менен байланыштырат. Эгерде Windows дун печаттоо менеджери активдүү болсо, анда F ке жазылган ар кандай файлдын тексти печаттоо менеджери тарабынан эске сакталынат жана файлды жапкан кезде печатталат.

AssignPrn процедурасы	
Procedure AssignPrn(var F: Text; Devise, Driver, Port: PChar);	
Пайдаланылуучу режим:	Windows
Арналышы:	F тексттик файлына Devise, Driver жана Port параметрлери менен баяндалган принтерди дайындайт. Бул маанилер win.ini (түзүлүштөр секциясындагы) файлында берилген түзүлүшкө тиешелеш келиши керек.

SetPrnFont функциясы	
Function SetPrnFont(var F: Text; Font: HFont):HFont;	
Пайдаланылуучу режим:	Windows
Арналышы:	Берилген шрифтти пайдалануу менен файлды печаттоону чакырат. Принтер тарабынан берилген учурда пайдаланылып жаткан шрифттин баяндагычы кайтаралылып берилет.

TitlePrn процедурасы	
Procedure TitlePrn(var F: Text; Title: PChar);	
Пайдаланылуучу режим:	Windows
Арналышы:	Печатталуучу файлдын бөркүн түзөт. TitlePrn процедурасы PeWrite тан мурда чакырылуусу керек.

8.7. GRAPH модулунун процедуралары жана функциялары

Graph модулу – бул тез иштөөчү кубаттуу камтылуучу программалардын библиотекасы болуп графика менен иштөө үчүн универсалдык арналышка ээ. Бул модул IBM-биргелешкен компьютерлерде бир кыйла таркалган адаптерлерди пайдалануучу аппараттык көз каранды эмес графиктик иштеп чыгууларды (обработчики) кармап турат.

Модулдун иши үчүн графиктик адаптерлердин автономдук драйверинин файлдарынын тобу BGI файлдар, ал эми штрихтик шрифтерди пайдалануунда – ушул шрифттердин файлдары (CHR файлдар) талап кылынат жана модулдун өзү Graph.TPU файлында кармалган.

Мисал катарында бул модулдун бир нече процедура жана функцияларын колдонуу менен түзүлгөн программалык кодду карайлы:

```
Uses Graph,Crt;
 Var Gd, Gm : Integer; Color : Word;
       Pal : PaletteType;
Begin
   Gd:=Detect;
    InitGraph(Gd, Gm, 'C:\Bp\Bgi');
    If GraphResult <> grOk Then Halt(1);
    Randomize;
    GetPalette(Pal);
   Repeat
     Color:=Succ(GetColor);
     If Color>Pal.Size - 1 Then Color:=0;
     SetColor(Color);
     LineTo(Random(GetMaxX), Random(GetMaxY));
   Until KeyPressed;
  CloseGraph;
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

Arc процедурасы	
Procedure Arc(X, Y: Integer; Angle1, Angle2, R: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	R радиустуу айлананын Angle1 баштапкы бурчунан Angle2 бурчуна чейинки жаасын сызат. (X, Y) чекити – айлананын борбору.

Bar процедурасы	
Procedure Bar(X1, Y1, X2, Y2: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Боёнун стандарттык тибин жана түсүн пайдалануу менен (X1, Y1) жана (X2, Y2) кординаталары менен берилген мамычаны сызат.

Bar3D процедурасы	
Procedure Bar3D(X1, Y1, X2, Y2: Integer; L: Word; S: Boolean);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Боёнун учурдагы тибин жана түсүн пайдалануу менен (X1, Y1) жана (X2, Y2) кординаталары, L – терендиги, S – жогорку тегиздигинин чагылышы белгиси менен үч өлчөмдүү параллелепипедди сызат.

Circle процедурасы	
Procedure Circle(X, Y: Integer; R: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	R радиустуу айлананы сызат. (X1, Y1) чекити – айлананын борбору деп эсептелет.

ClearDevice процедурасы	
Procedure ClearDevice;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик экрандын учурдагы абалын алып салат (сбрасывает) жана аны берилгендерди чыгаруу үчүн даярдайт.

ClearViewPort процедурасы	
Procedure ClearViewPort;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Көрүү терезесинин аймагын тазалайт.

CloseGraph процедурасы	
Procedure CloseGraph;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик системанын ишин токтотот

DetectGraph процедурасы	
Procedure DetectGraph;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Тиешелүү аппараттык каражаттардын бар экендигин текшерет жана кайсы графикалык режимди, драйверди пайдалануу керектигин аныктайт.

DrawPoly процедурасы	
Procedure DrawPoly(N: Word; Var S);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Сызыктын учурдагы тибин жана учурдагы түстү пайдалануу менен көп бурчтуктун контурун сызат. S параметри көп бурчтуктагы ар бир кесилиштин координаталарын кармап турган типтештирилбеген параметр. N параметри S теги координаталардын санын берет. Координата эки сөздөн турат: X маанiси жана Y маанiси.

Ellipse процедурасы	
Procedure Ellipse(X, Y: Integer; SStart, SEnd: Word; RadX, RadY: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	(X, Y) чекитин борбор катары, ал эми RadX менен RadY ти горизонталдык жана вертикалдык октор катары пайдалануу менен SStart баштапкы бурчунан SEnd бурчуна чейинки элиптикалык жааны сызат.

FillEllipse процедурасы	
Procedure FillEllipse(X, Y: Integer; XRad, Yrad: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	(X, Y) чекитин борбор катары, ал эми XRad менен YRad параметрлерин горизонталдык жана вертикалдык октор катары пайдаланып боёлгон эллипсти сызат.

FillPoly процедурасы	
Procedure FillPoly(N: Word; Var S);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Боёлгон көп бурчтукту сызат. S – параметри көп бурчтуктагы ар бир кесилиштин координаталарын кармап турган типтештирилбеген параметр. N параметри S теги координаталардын санын берет. Координата эки сөздөн турат: X мааниси жана Y мааниси.

FloodFill процедурасы	
Procedure FloodFill(X, Y: Integer; S: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы толтургучту пайдалануу менен чектелген аймакты боёйт. (X, Y) чекити – бул толтурулуучу аймактын каалагандай ички чекити. S параметри менен аныкталган түс менен чектелген аймакты толтуруу үчүн боёонун учурдагы үлгүсү пайдаланылат.

GetArcCoords процедурасы	
Procedure GetArcCoords(Var Coords: ArcCoordsType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Пайдалануучуга Arc процедурасына акыркы кайрылуунун координаталары жөнүндөгү суроо-талапты (запрос) берүүгө мүмкүнчүлүк берет.

GetAspectRatio процедурасы	
Procedure GetAspectRatio(Var Xk, Yk: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	(Xk, Yk) – салыштырма узартуу коэффициентин эсептөөгө мүмкүнчүлүк берүүчү графиктик экрандын аракети этип

	турган мүмкүнчүлүк берүү жөндөмдүүлүгүн (разрешающая способность) берет.

GetBkColor функциясы	
Function GetBkColor: Wnd;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Фондук түстүн учурдагы маанисин берет.

GetColor функциясы	
Function GetColor: Wnd;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	SetColor процедурасына болгон мурдагы (алдыңкы) ийгиликтүү кайрылуу кезинде коюлган (орнотулган) негизги түстүн учурдагы маанисин кайтарып берет.

GetDefaultPalette функциясы	
Function GetDefaultPalette(var Palette:PaletteType):PaletteType;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	InitGraph процедурасын чакырган кезде драйвер аркылуу инициализанцияланган Palette палитрасын кармап турган PaletteType тибиндеги палитранын баяндалыш жазуусун берет.

GetDriverName функциясы	
Function GetDriverName: String;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы драйвердин атын кармап турган жолчону берет.

GetFillPattern процедурасы	
Procedure GetFillPattern(var Pattern: FillPatternType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	SetFillPattern процедурасына болгон акыркы кайрылууда берилген толтургучтун учурдагы түсүн берет. Pattern параметри – бул толтургучтун үлгүсү.

GetFillSettings процедурасы	
Procedure GetFillSettings(var Inf: FillSettingsType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	SetFillStyle жана SetFillPattern процедуралары менен коюлган толтургучтун тиби жана түсү жөнүндөгү суроо-талапты (запрос) чыгарууга мүмкүнчүлүк берет.

GetGraphMode функциясы	
Function GetGraphMode: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы графиктик режимди берет. Режимдин мааниси 0 дөн 5 ке чейинки диапазондогу бүтүн сан болуп учурдагы драйверден көз каранды болот.

GetImage процедурасы	
Procedure GetImage(X1, Y1, X2, Y2: Integer; var BitMap);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Экрандын берилген областынын экилик образын буферге сактайт. X1, Y1, X2, Y2 параметрлери экрандын тик бурчтуу аймагын аныктайт. BitMap параметри – бул экрандын аймагы үчүн бөлүнгөн эстин өлчөмү менен кошулууда 4 төн кичине болбогон типтештирилбеген параметр болот. BitMap тын биринчи эки сөзү экрандын аймагынын кеңдиги жана бийиктигин сактоо үчүн пайдаланылат, үчүнчү сөзү бош болот.

GetLineSettings процедурасы	
Procedure GetLineSettings(var LineInfo: LineSettingsType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	SetLineStyle процедурасы менен коюлган сызыктын учурдагы стилин, сызыктын үлгүсүн жана анын жоондугун берет.

GetMaxColor функциясы	
Function GetMaxColor: Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	SetColor процедурасы менен берүүгө мүмкүн болгон түстүн эң чоң маанисин берет.

GetMaxMode функциясы	
Function GetMaxMode: Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы жүктөлгөн драйвер үчүн режимдин номеринин максималдык маанисин берет.

GetMaxX функциясы	
Function GetMaxX: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы графиктик режим жана драйвер үчүн эң оң жаккы мамычанын координатасын (X боюнча мүмкүнчүлүк берүүчү) берет.

GetMaxY функциясы	
Function GetMaxY: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы графиктик режим жана драйвер үчүн эң төмөңкү жолчонун (Y боюнча мүмкүнчүлүк берүүчү) координатасын берет.

GetModeName функциясы	
Function GetModeName(ModeNumber: Integer): String;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Берилген Mode графикалык режиминин атын кармап турган жолчону берет.

GetModeRange процедурасы	
Procedure GetModeRange(GraphDriver: Integer; var LoMode, HiMode: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Берилген драйвер үчүн эң чоң жана эң кичине графиктик режимиди берет.

GetPalette процедурасы	
Procedure GetPalette(var Palette: PaletteType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы Palette палитрасын жана анын өлчөмүн берет.

GetPaletteSize функциясы	
Function GetPaletteSize: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Түстөрдүн палитрасынын жадыбалынын өлчөмүн берет. Бул функция учурдагы графиктик режим үчүн палитранын канча жазууларын берүүгө мүмкүн экендигин көрсөтөт. Мисалы, түстүү режимди жана EGA адаптерин пайдалнууда функция 16 маанисин берет.

GetPixel функциясы	
Function GetPixel(X, Y: Integer): Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	X, Y чекитиндеги сүрөттөлүш элементинин түстүк маанисин берет.

GetTextSettings процедурасы	
Procedure GetTextSettings(var TextInfo: TextSettingsType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Тексттик шрифттин тибин, SetTextStyle жана SetTextJusttify процедураларынын жардамында берилген анын багытын, өлчөмүн жана түздөлүшүн берет.

GetViewSettings процедурасы	
Procedure GetViewSettings(var ViewPort: ViewPortType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Пайдалануучуга экрандын карап-көрүлүп жаткан учурдагы бөлүгү жана “кесилип алынган” сүрөттөлүштүн параметрлери жөнүндө суроо-талапты берүүгө мүмкүнчүлүк берет..

GetX функциясы	
Function GetX: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы көрсөткүчтүн X-координатасын берет.

GetY функциясы	
Function GetY: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы көрсөткүчтүн Y-координатасын берет.

GraphDefaults процедурасы	
Procedure GraphDefaults;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик каражаттар үчүн коюлган (орнотулган) параметрлерди алып салат (сбрасывает). Учурдагы көрсөткүчтү нөлдүк чекитке коёт жана графкитик система үчүн атайын көрсөтүлбөгөн учурдагы (по умочанию) параметрлерди (карап-көрүү аймагын, палитраны, фонду жана негизги түстү, сызыктын тибин жана үлгүсүн, толтургучтун тибин, толтургучтун үлгүсүн жана түсүн) орнотот.

GraphErrorMsg функциясы	
Function GraphErrorMsg(ErrorCode: Integer): String;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Катанын берилген коду (ErrorCode) үчүн ката жөнүндөгү билдирүү жолчосун кайтарып берет.

GraphResult функциясы	
Function GraphResult: Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Акыркы графиктик амал үчүн катанын кодун берет.

ImageSize функциясы	
Function ImageSize(X1, Y1, X2, Y2: Integer): Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим

Арналышы:	Экрандын тик бурчтуу аймагын сактоо үчүн зарыл болгон байттардын санын берет. X1, Y1, X2, Y2 параметрлери экрандын тик бурчтуу аймагын аныктайт. ImageSize функциясы экрандын берилген аймагын сактоо үчүн GetImage функциясы үчүн зарыл болгон байттардын санын аныктайт. Экрандын аймагынын экилик образынын өлчөмү бир нече сөздөн турган эстин өлчөмүн өзүнө алып турат. Биринчи сөздө аймактын кеңдиги, экинчи сөздө бийиктиги сакталат. Кийинки бир нече сөз аймактын өзүнүн атрибуттарын кармап турат. Акыркы сөз резервде болот.

InitGraph процедурасы	
Procedure InitGraph(var GraphDriver: Integer; var GraphMode: Integer; PathToDriver: String)	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик системаны инициализациялайт жана аппаратураны графиктик режимге которот.

InstallUserDriver функциясы	
Function InstallUserDriver(Name: String; AutoDetectPtr: Pointer): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Borland фирмасынын BGI түзүлүштөр драйверлеринин жадыбалына башка фирмалар тарабынан иштелип чыгылган драйверлерди кошот.

InstallUserFont функциясы	
Function InstallUserFont(FontFileName: String): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Borland фирмасынын системасында (BGI) каралбаган жаңы шрифтти орнотот.

Line процедурасы	
Procedure Line(X1, Y1, X2, Y2: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Тиби SetLineStyle процедурасы менен, ал эми түсү SetColor процедурасы менен коюлган (X1, Y1) чекитинен

	(X2, Y2) чекитине чейинки сызыкты сызат. Учурдагы көрсөткүч өзүнүн абалын өзгөртпөйт.

LineRel процедурасы	
Procedure LineRel(Dx, Dy: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы көрсөткүчтөн баштап (Dx, Dy) салыштырмалуу аралык менен берилген чекитке түздү жүргүзөт.

LineTo процедурасы	
Procedure LineTo(X, Y: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы көрсөткүч турган чекиттен (X, Y) чекитине чейинки түз сызыкты жүргүзөт. Сызыктын жоондугу жана тиби SetLineStyle жана SetColor процедуралары менен берилет. Учурдагы көрсөткүч акыркы чекитке которулат.

MoveRel процедурасы	
Procedure MoveRel(Dx, Dy: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы көрсөткүчтү анын учурдагы позициясынан салыштырмалуу координаталар менен берилген аралыкка жылдырат.

MoveTo процедурасы	
Procedure MoveTo(X, Y: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы графиктик көрсөткүчтү (X, Y) чекитине жылдырат.

OutText процедурасы	
Procedure OutText(TextString: String);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Жолчону чыгаруу түзүлүшүн учурдагы көрсөткүч турган жерине жиберет.

OutTextXY процедурасы	
Procedure OutTextXY(X, Y: Integer; TextString: String);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Жолчону чыгаруу түзүлүшүнө айдайт (персылает). S параметри менен берилген жолчого (X, Y) чекитинде чыгарылат. Эгерде жолчо абдан узун болуп экрандын пределинен чыгып кетсе же карап-көрүү (просмотр) аймагына батпаса, анда ал кесилип ташталат.

PieSlice процедурасы	
Procedure PieSlice(X, Y: Integer; SStart, SEnd, R: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Тегеректин секторун чийет жана түс менен толтурат. (X,Y) чекити тегеректин борбору катары пайдаланылып, секто баштапкы (SStart) бурчтан акыркы (Send) бурчуна чейин R радиусу менен чийилет.

PutImage процедурасы	
Procedure PutImage(X, Y: Integer; var Mass; Oper: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Экрандын берилген аймагынын экилик образын буферден экранга чыгарат. (X, Y) координаталуу чекит экрандын тик бурчтуу аймагынын жогорку сол бурчун аныктайт. Mass параметри экрандын аймагынын кеңдигин жана бийиктигин аныктоочу позитивдүү эмес параметр болуп эсептелет. Oper параметри аймактын экилик образын экранга чыгаруу үчүн кайсы экилик амал пайдаланыларын акныктайт.

PutPixel процедурасы	
Procedure PutPixel(X, Y: Integer; Pixel: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	(X, Y) чекитинде түсү Pixel параметри менен аныкталуучу сүрөттөлүштүн элементин тургузат.

Rectangle процедурасы	
Procedure Rectangle(X1, Y1, X2, Y2: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы түстү жана сызыктын тибин пайдалануу менен тик бурчтукту чийет. (X1, Y1) чеити тик бурчтуктун жогорку сол бурчун, ал эми (X2, Y2) төмөнкү оң бурчун аныктайт (0<=X1<=X2<=GetMaxX жана 0<=Y1<=Y2<=GetMaxY).

RegisterBGIDriver функциясы	
Function RegisterBGIDriver(Driver: Pointer): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик системада пайдалануучу тарабынан жүктөлгөн же программа менен компановка болгон BGI (Borland фирмасынын форматы) форматындагы драйверди каттайт (регистрирует).

RegisterBGIFont функциясы	
Function RegisterBGIFont(Font: Pointer): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик системада пайдалануучу тарабынан жүктөлүүчү же программа менен компановка болгон BGI (Borland фирмасынын форматы) форматындагы шрифтти каттайт. Ката болгон учрда алынуучу маани 0 дөн кичине болот. Андай болбогондо шрифттин ички номери алынат.

RestoreCrtMode процедурасы	
Procedure RestoreCrtMode;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Экрандын режимин баштапкы абалга кайтарат (графиканы инициялизациялоого чейинки абалга).

Sector процедурасы	
Procedure Sector(X, Y: Integer; SStart, SEnd, XRed, YRed: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Элиптикалык секторду сызат жана толтурат. Процедура (X, Y) чекитин борбор катары пайдаланат, XRed жана

	YRed параметрлери тиешелүү түрдө горизонталдык жана вертикалык радиустарды аныктайт. Сектор баштапкы SStart бурчунан SEnd бурчуна чейин сызылат. Сектор учурдагы түс менен чийилип SetFillStyle жана SetFillPattern процедуралары менен берилген боёнун жана түстүн үлгүлөрүн пайдалануу менен боёлот.

SetActivePage процедурасы	
Procedure SetActivePage(Page: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик чыгаруу үчүн активдүү бетти коёт. Page параметри менен берилген беи активдүү болуп саналат. Бүткүл графиктик чыгаруу эми ушул бетке багытталат.

SetAllPalette процедурасы	
Procedure SetAllPalette(var Palette);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Палитранын бардык түстөрүн көрсөтүлгөн түстөргө өзгөртөт. Palette параметри типтештирилбеген болуп эсептелет. Анын биринчи сөзү палитранын узундугу болуп саналат. Кийинки n байт палитранын учурдагы түстөрүн өзгөртөт. Ар бир түс -1 ден 15 ке чейинки маанилерди кабыл алат. -1 мааниси жазуунун мурдагы маанисин өзгөртпөйт.

SetAspectRatio процедурасы	
Procedure SetAspectRatio(Xasp, Yasp: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Салыштырмалуу узартуу коэффициентинин атайын көрсөтүлбөгөн учурда (по умолчанию) алынуучу маанисин өзгөртөт.

SetBkColor процедурасы	
Procedure SetBkColor(Color: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Палитраны пайдалнуу менен учурдагы фондук түстү коёт.

SetColor процедурасы	
Procedure SetColor(Color: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Палитраны пайдалнуу менен сүрөттүн түсүн Color параметринин маанисине ылайык коёт.

SetFillPattern процедурасы	
Procedure SetFillPattern(Pattern: FillPatternType; Color: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Пайдаланууучу тарабынан берилген боёо үлгүсүн тандайт.

SetFillStyle процедурасы	
Procedure SetFillStyle(Pattern: Word; Color: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Боёонун тибин жана анын түсүн коёт.

SetGraphBufSize процедурасы	
Procedure SetGraphBufSize(BufSize: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Боёо үчүн пайдаланылуучу буфердин өлчөмүн өзгөртүүгө мүмкүнчүлүк берет.

SetGraphMode процедурасы	
Procedure SetGraphMode(Mode: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Система үчүн графиктик режимди коёт жана экранды тазалайт.

SetLineStyle процедурасы	
Procedure SetLineStyle(SType: Word; Pattern: Word; S: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Сызык үчүн S жоондугун (семиздигин) жана SType тибин коёт.

SetPalette процедурасы	
Procedure SetPalette(N: Word; Color: Shortint);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	N (түстүн номери) жана Color (түс) параметрлерине тиешелеш түрдө палитранын түстөрүнүн бирөөсүн өзгөртөт.

SetRGBPalette процедурасы	
Procedure SetRGBPalette(N, RedValue, GreenValue, BlueValue: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	IBM 8514 жана VGA драйверлери үчүн палитранын жазуусун өзгөртүүгө мүмкүнчүлүк берет. N параметри жүктөө керек болгон палитра жазуусун берет, ал эми RedValue, GreenValue, BlueValue – палитранын жазуусун түзүүчү түстөр. SetRGBPalette процедурасын VGA графикалык адаптерин жана IBM 8514 драйвери болгон учурда гана пайдаланууга болот.

SetTextJustify процедурасы	
Procedure SetTextJustify(Horiz, Vert: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	OutText жана OutTextXY процедуралары тарабынан пайдаланылуучу текстти түздөө маанилерин коёт.

SetTextStyle процедурасы	
Procedure SetTextStyle(Font: Word; S: Word; Size: CharSizeType);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Font учурдагы шрифттин S тибин жана Size символдун өлчөмүнүн коэффициентин коёт. Процедура OutText жана OutTextXY процедуралары тарабынан аткарылуучу бүткүл тексттик чыгарууга таасир этет.

SetUserCharSize процедурасы	
Procedure SetUserCharSize(MultX, DivX, MultY, DivY: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим

Арналышы:	Вектордук шрифттер үчүн символдун бийиктигин жана туурасын өзгөртүүгө пайдалануучуга мүмкүнчүлүк берет. MultX:DivX катышы активдүү шрифт үчүн кадимки кеңдикке көбөйтүлгөн катышты берет. MultY:DivY катышы – активдүү шрифт үчүн нормалдуу бийиктикке көбөйтүлгөн катыш. Мисалы, шрифтти эки эсе чоң кылыш үчүн MultX үчүн 2 деген маанини, ал эми DivX үчүн 1 ге барабар маанини (2 div 1=2) берүү керек.

SetViewPort процедурасы	
Procedure SetViewPort(X1, Y1, X2, Y2: Word; Clip: Boolean);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Графиктик чыгаруу үчүн учурдагы көрүү (параметр) аймагын же терезесин коёт.

SetVisualPage процедурасы	
Procedure SetVisualPage(Page: Word);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Сүрөттөлүүчү (отображаемый) графиктик беттин номерин берет.

SetWriteMode процедурасы	
Procedure SetWriteMode(Mode: Integer);	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Сызыкты чийүүдө Mode жазуу режимин коёт.

TextHeight функциясы	
Function TextHeight(S: String): Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Учурдагы шрифттин өлчөмүнүн негизинде жолчонун сүрөттөлүш элементтердеги бийиктигин берет.

TextWidth функциясы	
Function TextWidth(S: String): Word;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим
Арналышы:	Жолчонун сүрөттөлүш элементтердеги кеңдигин (ширина) берет.

8.8. OVERLAY модулунун процедуралары жана функциялары

Overley модулу – тилдин драйверлерин башкаруу программалары пайдалануучу процедураларды, функцияларды жана өзгөрүлмөлөрдү кармап турат. Ал DOS тун реалдык режиминде иштөөчү (аткарылуучу) процедуралар үчүн талап кылынуучу эстин көлөмүн азайтууга мүмкүнчүлүк берет. Программа иштеген мезгилде анын бир бөлүгү гана эсте тургандыктан иш жүзүндө ал машинада реалдуу уруксат берилген эске караганда эсти көп талап кылган програмаларды жазууга уруксат берет.

Бул модул Overlay.TPU файлында жайгашкан. Мисал катарында төмөнкү программаны карайбыз:

```
Uses Overlay;
Begin
  OvrInit('EDITOR.OVR');
   If OvrResult <> ovrOk Then
 Begin
  Case OvrResult Of
  ovrError : WriteLn('Программада оверлей жок.');
  ovrNotFound: WriteLn('Оверлей файлы табылган жок.');
  End;
 Halt(1);
 End;
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

OvrClearBuf процедурасы	
Procedure OvrClearBuf;	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Оверлейлик буферди тазалайт. Процедураны чакыруунун алдында берилген моментте жүктөлгөн бардык оверлейлер оверлейлик буферден чыгарылат. Бул болсо оверлейлик программаларга кийинки кайрылууларда оверлейлердин оверлейдик файлдан (же EMS эсинен) жүктөлүшүнө алып келет. Эгерде OvrClearBuf процедурасы оверлейден чакырылса, анда OvrClearBuf тан кайтуунун алдында бул оверлей токтоосуз кайра жүктөлөт.

OvrCectBuf функциясы	
Function OvrCectBuf:Longint;	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Оверлейлик буфердин учурдагы өлчөмүн берет. Оверлейлик буфердин өлчөмү OvrSetBuf процедурасына кайрылуунун жардамында коюлат. Алгач оверлейлик буфер эң чоң оверлейдин өлчөмүнө тиешелеш келүүчү минималдык мүмкүн болгон өлчөмгө ээ болот. Оверлейлик программа аткарылганда мындай өлчөмдөгү буфер автоматтык түрдө бөлүнүп алынат. Буфердин баштапкы өлчөмү 64 К дан ашып кетиги мүмкүн, анткени өзүнө эң чоң оверлейдин кодун да, о.э. анын корректирлөө (информацияларын) маалыматтарын да алып турат.

OvrGetRetry функциясы	
Function OverGetRetry: Longint;	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Маниси OvrSetRetry процедурасы менен коюлуучу оверлейлик буфердеги сынама аймактын (пробная область) учурдагы өлчөмүн берет.

OvrInit процедурасы	
Procedure OvrInit(FileName: String);	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Оверлейлерди башкаруучу камтылуучу системаны (подсистема) инициализациялайт жана оверлейлик файлды ачат. Эгерде файлдын атынын FileName параметринде дисктин эн-белгиси (меткасы) же камтылуучу каталог берилбеген болсо, анда оверлейди башкаруучу камтылуучу система файлды EXE файлдарды кармап турган учурдагы катологдон (DOS тун 3.x версиясында иштеген учурда) жана операциялык чөйрөнүн PATH өзгөрүлмөсүнүн жардамында берилген каталогдордон издейт. Болуп калышы мүмкүн болгон каталар операциялык чөйрөнүн IoResult өзгөрүлмөсүнүн жардамында алынат. OvrOk мааниси ийгиликтүү аяктоону, OvrError оверлейлик файл туура эмес форматка ээ экендигин, же программа оверлейди кармап турбагандыгын көрсөтөт. OvrNotFound мааниси оверлейлик файлдын табылбагандыгын билдирет.

OvrInitEMS процедурасы	
Procedure OvrInitEMS;	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Эгер бул мүмкүн болсо, анда оверлейлик файлды EMS эсине жүктөйт. Эгерде кеңейтирилген эстин драйвери ушундай эстин жетишерлик көлөмү бар болгон болсо, анда бул процедура бардык оверлейлерди кеңейтирилген эске жүктөйт жана оверлейлик файлды жабат. Оверлейлердин кийинки жүктөлүштөрү жөн гана алардын эстен эске тез жиберилишине келтирилет. Ошондой эле бул процедура программанын иши аяктаган кезде бөлүнүп коюлган EMS эсти автоматтык түрдө бошотуучу чыгуу процедурасын орнотот.

OvrSetBuf процедурасы	
Procedure OvrSetBuf(BufSize: Longint);	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Оверлейлик буфердин өлчөмүн коёт. BufSize параметри оверлейлик буфердин баштапкы өлчөмүнөн чоң же барабар жана MemAail+OvrGetBuf тан кичине же барабар болушу керек. Оверлейлик буфердин баштапкы өлчөмү – бул OvrSetBuf ка кайрылуунун алдында OvrGetBuf процедурасы тарабынан берилүүчү өлчөм болуп саналат. Эгерде берилген өлчөм учурдагы өлчөмдөн ашып кетсе, анда динамикалык бөлүштүрүлүүчү эстин башынан кошумча эс бөлүнүп алынат, ошентип олтуруп эстин динамикалык бөлүштүрүлүүчү областынын өлчөмү азаят. Эгерде берилген өлчөм учурдагы өлчөмдөн кичине болсо, анда ашыкча аймак динамикалык бөлүштүрүлүүчү эске берилет.

OvrSetRetry процедурасы	
Function OverSetRetry(Size: Longint);	
Пайдаланылуучу режим:	Реалдык режим
Арналышы:	Оверлейлик буфердеги сынама аймактын (пробная область) өлчөмүн коёт. Эгерде оверлейди буферге жайгаштыргандан кийин буфердин акырына чейин Size байт калган болсо, ал автоматтык түрдө сынама аймакка жайгаштырылат. Оверлейлик буфердеги ар кандай бош

	мейкиндик сынама аймактын кандайдыр бир бөлүгү катары каралат. Мурдагы версиялары менен биргешелүүчүлүктү камсыз кылуу максатында (оверлейлер администраторунун) көрсөтүлбөгөн учурда (по умолчанию) сынама аймак нөл өлчөмүнө ээ болот, ал сынама кайталоолор механизмине тыюу салууга алып келет.

8.9. STRINGS модулунун процедуралары жана функциялары

Strings модулунун процедуралары жана функциялары PChar тибиндеги нөл менен аяктоочу жолчолор (ASCII – жолчолор) менен иштөө жана бул жолчолорду String тибиндеги String – жолчолор менен эки жактуу өзгөртүп түзүү үчүн арналган.

Мисал катарында төмөнкү программаны карайбыз:

```
Uses Strings, WinCrt;
  Var
       S : Array [0..15] Of Char;
 Begin
  StrCopy(S, 'Turbo Pascal');
  WriteLn(S);
End.
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

StrCat функциясы	
Function StrCat(Dest, Source: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бир жолчонун көчүрмөсүн экинчи жолчонун аягына улайт жана ушундай конкатенциянын жыйынтыгын берет. StrCat фунцкциясы Dest ке Source жолчосунун көчүрмөсүн кошот жана Dest ти берет. Бул функция узундукту текшерүүнү аткарбайт. Биз өзүбүз Dest параметри менен берилген буферде StrLen(Dest)+StrLen(Source)+1 символ үчүн жетишерлик орун болушун камсыз кылышыбыз керек. Эгерде узундукту текшерүүнү аткаргыбыз келсе анда StrLCat функциясын пайдаланышыбыз керек.

StrComp функциясы	
Function StrComp(Str1, Str2: PChar): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Эки жолчону салыштырат. Бул функция эки Str1 жана Str2 жолчолорун салыштырат. Чыгарылуучу маани кичине нөл болот эгерде Str1<Str2 болсо, нөлгө барабар болот эгерде Str1=Str2 болсо, нөлдөн чоң болот эгерде Str1>Str2 болсо.

StrCopy функциясы	
Function StrCopy(Dest, Source: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бир жолчону экинчи жолчого көчүрөт. Source жолчосун Dest жолчосуна көчүрөт жана жыйынтыкты Dest ке берет. Бул функция да узундукту текшерүүнү аткарбайт. Dest параметри менен берилген буферде StrLen(Dest)+1 символ үчүн жетишерлик орун болушун өзүбүз камсыз кылышыбых керек. Эгерде узундукту текшерүүнү аткаргыбыз келсе, анда StrLCopy функциясын пайдаланышыбыз керек.

StrDispose функциясы	
Function StrDispose(Str: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Эстин динамикалык бөлүштүрүлүүчү аймагындагы жолчону жок кылат. StrDispose функциясы мурда StrNew функциясынын жардамында бөлүштүрүлгөн жолчону жок кылат. Эгерде str параметри nil маанисине ээ болсо, анда StrDispose эч кандай аракет аткарбайт.

StrECopy функциясы	
Function StrECopy(Dest, Source: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Көрсөткүчтү жыйынтык-жолчонун акырына кайтаруу менен бир жолчону экинчи жолчого көчүрөт. StrECopy функциясы Source жолчосун Dest ке көчүрөт жана

	StrEnd(Dest) ти берет. Dest параметри менен берилген буферде StrLen(Dest)+1 символ үчүн жетишерлик орун болушун өзүбүз камсыз кылышыбыз керек. Жолчолордун узундугунун конкетанциясы үчүн StrECopy нин камтылма (вложенные) чакырууларын пайдалануу мүмкүн.

StrEnd функциясы	
Function StrECopy(Str: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Көрсөткүчтү жолчонун аягына кайтарат. StrEnd функциясы көрсөткүчтү Str жолчосун аяктоочу нөл символуна кайтарат.

StrIComp функциясы	
Function StrIComp(Str1, Str2: PChar): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Символдордун регистрлерин айырмалабастан эки жолчону салыштырат. StrIComp функциясы Str1 жолчосун Str2 жолчосу менен символдордун регистирин эске албастан салыштырат.Алынуучу маани нөлдөн кичине болот эгерде Str1<Str2 болсо, нөлгө барабар болот эгерде Str1=Str2 болсо, нөлдөн чоң болот эгерде Str1>Str2 болсо.

StrLCat функциясы	
Function StrLCat(Dest, Source: PChar; MaxLen: Word): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бир жолчонун көчүрмөсүн экинчи жолчонун аягына бириктирет (улайт) жана ушундай конкатенциянын жыйынтыгын берет. StrLCat фунцкциясы Dest ке Source жолчосунун көчүрмөсүн кошот (MaxLen-StrLen(Dest) символдон көп эмес) жана жыйынтыкты Dest ке берет. MaxLen параметрин аныктоо үчүн SizeOf функциясын пайдаланууга болот.

StrLComp функциясы
Function StrLComp(Str1, Str2: PChar; MaxLen: Word): Integer;

Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Эки жолчону берилген максималдык узундукка чейин салыштырат. StrLComp функциясы Str1 жолчосун Str2 жолчосу менен MaxLen максималдык узундугуна чейин салыштырат.Алынуучу маани нөлдөн кичине болот эгерде Str1<Str2 болсо, нөлгө барабар болот эгерде Str1=Str2 болсо, нөлдөн чоң болот эгерде Str1>Str2 болсо.

StrLCopy функциясы

Function StrLCopy(Str1, Str2: PChar; MaxLen: Word): PChar;

Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Символдорду бир жолчодон экинчижолчого көчүрөт. StrLCopy функциясы MaxLen ден ашпаган символдорду Source жолчосунан Dest жолчосуна көчүрөт жана жыйынтыкты Dest ке берет. MaxLen параметрин аныктоо үчүн SizeOf стандарттык функциясын пайдаланууга болот.

StrLen функциясы

Function StrLen(Str: PChar): Word;

Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Str жолчосундагы символдордун санын берет. StrLen функциясы Str жолчосундагы аяктоочу нөлдү кошпогондогу символдордун санын берет.

StrLIComp функциясы

Function StrLIComp(Str1, Str2: PChar; MaxLen: Word): Integer;

Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Эки жолчону максималдык узундукка чейин символдордун регистирлерин айырмалабастан салыштырат. StrLIComp функциясы Str1 жолчосун Str2 жолчосу менен символдордун MaxLen максималдык узундугуна чейин символдордун регистирлерин айырмалабастан салыштырат. Алынуучу маани нөлдөн кичине болот эгерде Str1<Str2 болсо, нөлгө барабар болот эгерде Str1=Str2 болсо, нөлдөн чоң болот эгерде Str1>Str2 болсо.

StrLower функциясы	
Function StrLower(Str: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Жолчону төмөнкү регистрге өзгөртүп түзөт. StrLower функциясы Str жолчосун төмөнкү регистрге өзгөртүп түзөт жана жыйынтыкты Str ге берет.

StrMove функциясы	
Function StrMove(Dest, Source: PChar; Count: Word): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Бир жолчодон экинчи жолчого символдорду көчүрөт. StrMove функциясы Source жолчосунан туптуура Count символду Dest ке көчүрөт жана жыйынтыкты Dest ке берет. Source жана Dest бирин-бири каптап (прекрывать) калышы мүмкүн.

StrNew функциясы	
Function StrNew(Str: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Динамикалык бөлүштүрүлүүчү аймакта жолчо үчүн эсти бөлөт. StrNew функциясы динамикалык бөлүштүрүлүүчү аймакта Str жолчосу үчүн эсти бөлөт. Эгерде Str мааниси nil ге барабар болсо же көрсөткүч бош жолчону көрсөтүп турса, анда StrNew функциясы Nil маанисин берет жана динамикалык аймакта эсти бөлбөйт. Андай болбогон учурда StrNew функциясы Str дин көчүрмөсүн түзөт, GetMem ди чакыруу аркылуу эсти алуу менен, жана көрсөткүчтү жолчо-дубликатка кайтарат. Бөлүнгөн эс StrLen(Str)+1 байт өлчөмүнө ээ болот.

StrPas функциясы	
Function StrPas(Str: PChar): String;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Нөл менен аяктоочу жолчону String-жолчого өзгөртүп түзөт. StrPas функциясы ASCIIZ-жолчо тибиндеги Str жолчосун String-жолчого өзгөртүп түзөт.

StrPCopy функциясы	
Function StrPCopy(Dest: PChar; Source: String): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	String-жолчону нөл менен аяктоочу ASCIIZ-жолчого өзгөртүп түзөт. StrPCopy функциясы Source String-жолчосун Dest ке көчүрөт жана жыйынтыкты Dest ке берет. Dest параметри менен берилген буферде StrLen(Source)+1 символ үчүн жетишерлик орун болушун өзүбүз камсыз кылышыбых керек.

StrPos функциясы	
Function StrPos(Str1, Str2: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Жолчонун башка жолчого биринчи киришине көрсөткүчтү кайтарат. StrPos функциясы көрсөткүчтү Str2 нин Strt1 деги биринчи нускасына которот (кайтарат). Эгерде Str2 жолчосу Strt1 ден кармалып турбаса, анда бул функция nil маанисин берет.

StrRScan функциясы	
Function StrRScan(Str: PChar; Chr: Char): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Көрсөткүчтү символдун акыркы кирүүсүнө кайтарат. StrRScan функциясы көрсөткүчтү Chr символунун Strt жолчосуна акыркы кирүүсүнө (последнее вхождение) кайтарат. Эгерде Chr символу Str де кармалып турбаса (жок болсо), анда nil маанисин берет. Аяктоочу нөл символу жолчонун бөлүгү болуп эсептелет.

StrScan функциясы	
Function StrScan(Str: PChar; Chr: Char): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Көрсөткүчтү символдун жолчого биринчи киришине кайтарат. StrScan функциясы Chr символунун Strt

	жолчосуна биринчи кирүүсүнө кайтарат. Эгерде Chr символу Str де кармалып турбаса, анда nil маанисин берет. Аяктоочу нөл символу жолчонун бөлүгү болуп эсептелет.

StrUpper функциясы	
Function StrUpper(Str: PChar): PChar;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Жолчону жогорку регистрге өзгөртүп түзөт. StrUpper функциясы Strt жолчосун жогорку регистрге өзгөртүп түзөт жана жыйынтыкты Str ге берет.

8.10. WinAPI модулунун процедуралары жана функциялары

WinAPI модулу DOS тун корголгон режиминдеги Borland дын кеңейтирилиштери менен иштөөгө мүмкүнчүлүк түзөт. Тактап айтканда, ресурстук файлдарды, модулдарды, жана селекторлорду башкаруу, экилик коддогу DLL файлдары менен иштөө, эсти башкаруу функцияларын колдонуу ж.б.у.с. көптөгөн API функцияларын колдонууга болот.

Windows режиминде иштөө учурунда WinAPI модулунда кармалган процедура, функциялар, подпрограммалар динамикалык компоновкалануучу KERNEL.DLL жана USER.DLL библиотекаларында жайгашат. Ал эми DOS тун корголгон режиминде бул DLL файлдары талап кылынбайт.

Бул модулдун процедура жана функцияларын реализациялоону баяндоо үчүн төмөнкү программалык кодду карайлы.

Бул мисалда H – THandle тибиндеги өзгөрүлмө, ал эми P – көрсөткүч:

```
H:=GlobalAlloc(gmem_Moveable, 8192); { блокту бөлүп алуу }
  if H<>then { эгер эс белгиленип алынган болсо }
     begin
       P:=GlobalLock(H); { блокту блокировкалоо }
       .
                  { блокко P аркылуу кайрылууга уруксат алуу }
       .
        GlobalUnlock(H);   { блокту кайра ачуу }
        GlobalFree(H);     { блокту бошотуу }
     end;
```

Эми бул модулдун жалпы процедура жана функцияларын кыскача баяндап кетели.

AccessResource функциясы	
Function AccessResource(Instance, ResInfo: THandle): Integer;	
Пайдаланылуучу режим:	Корголгон режим, реалдык режим, Windows
Арналышы:	Instance параметри менен берилген ресурс файлын ачат, жана файл көрсөткүчүн ResInfo менен берилген позицияга кайтарат. Алынуучу маани ресурсту жүктөө үчүн андан аркы файлдан окуу амалдарында пайдалануучу файлдын баяндагычы болуп эсептелет. Эгерде ресурс табылбаса, анда 1 деген маани кайтарылып алынат. Instance параметри – бул System модулундагы HInstance өзгөрүлмөсүнөн адатта алынуучу же LoadLibrary менен түзүлүүчү модулдун нускаларынын (экземплярынын) баяндагычы болот. ResInfo параметри FindResource менен түзүлгөн ресур жөнүндөгү маалымат болушу керек.

AllocDStoCSAlias функциясы	
Function AllocDStoCSAlias(Selector: Word):Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Берилгендер сегментинин селекторун код сегментинин селекторуна чагылдырат. Берилгендер сегментинин аталган селектору шилтенилген сегменттин өзүнө эле шилтенилген код сегментинин селекторун түзөт жана кайтарат. Эгерде функция жаңы селекторду бөлүп көрсөтө албаса, анда нөлдүк маани алынат. Качан селектор-псевдоним керек болбой калганда, колдонулуучу программа аны FreeSelector дун жардамында бошотушу керек.

AllocSelector функциясы	
Function AllocSelector(Selector: Word):Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Selector параметри менен берилген селектордун так көчүрмөсү болгон жаңы селекторду бөлүп көрсөтөт. Эгерде Selector нөлгө барабар болсо, анда инициалдаштырылбаган жаңы селекторду берет. Эгерде функция жаңы селекторду бөлүп ала албаса, анда

	кайтарылып алынуучу маани нөлгө барабар болот. Качан селектор керек болбой калган учурда колдонмо программа аны FreeSelector функциясынын жардамында бошотуш керек.

ChangeSelector функциясы	
Function ChangeSelector(SourceSel, DestSel: Word):Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Көрсөтүлгөн берилгендер селекторуна тиешелеш болгон код селекторун же көрсөтүлгөн код селекторуна тиешелеш болгон берилгендер селекторун генерациялайт. Бул функция селектордун маанисин эмес, анын атрибуттарын гана өзгөртөт.

DOS3Call процедурасы	
Procedure DOS3Call;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	DOS тун 21H функциясы боюнча чакырылуучу үзгүлтүгүн берет. Бул процедуранын чакырылышы INT21H инструкциясынын аткарылышына тиешелеш келет. DOS3Call процедурасы Windows дун башкарылуусу астында тиешелеш DOS тогу INT21H программалык үзгүлтүгүнө караганда тезирээк иштейт. Анткени DOS3Call параметрлердин борбордук процессордун регистрлеринде берилишин талап кылат, ал ассемблер тилиндеги камтылуучу программаларда пайдаланышы мүмкүн.

FatalExit процедурасы	
Procedure FatalExit(Code: Integer);	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Фоталдык каталар жөнүндөгү билдирүүлөр үчүн оңдоп түзөөлөрдү (отладка) жүргүзүү максатында пайдаланылат.

FindResource функциясы	
Function FindResource(Instance: THandle; Name, ResType: PChar): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Instance параметри менен берилген ресурстар файлында

	Name атынын параметрлери жана ResType ресурстар тиби менен берилген ресурсту табат жана ресурс жөнүндөгү информациянын баяндагычын берет. Эгерде ресурс табылбаса нөлдүк маанини берет.

FreeLibrary процедурасы	
Procedure FreeLibrary(LibModule: THandle);	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Динамикалык жүктөлүүчү библиотканы бошотот. LibModule параметри LoadModule функциясы менен түзүлгөн модулдун нускасынын баяндагычы болушу керек. FreeLibrary берилген библиоткалар үчүн шилтемелердин эсептегичин 1 ге азайтат жана эгерде эсептегичтин мааниси 0 гө барабар болуп калса, анда библиотка ээлеп турган эс бошотулат.

FreeResource функциясы	
Function FreeResource(ResData: THandle):Bool;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Ал ээлеп турган эсти бошотуу менен жүктөлгөн ресурсту эстен жоготот.

FreeSelector функциясы	
Function FreeSelector(Selector: Word): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Алгач AllocSelector же AllocDStoCSAlias функциясы менен түзүлөн селекторду бошотот. FreeSelector ду чакыргандан кийин селектор уруксат берилбеген (недопустимый) болуп калат жана ал башка пайдаланыла албайт. Бул функция ийгиликтүү аткарылган учурда нөлгө барабар болгон маанини берет, андай болбогондо селектордун маанисин берет.

GetDOSEnvironment функциясы	
Function GetDOSEnvironment(var Year, M, Day, D: Word);	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Маселенин операциялык чөйрөсүнүн учурдагы жолчосуна көрсөткүчтү кайтарат. Колдонмо (прикладная) программадан айырмаланып динамикалык

	компоновкалануучу библиотка операциялык чөйрөнүн жолчосунун көчүрмөсүнө ээ эмес жана жолчону алуу үчүн ушул функцияны пайдаланышы керек.

GetFreeSpace функциясы	
Function GetFreeSpace(Flag: Word): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Динамикалык бөлүштүрүлүүчү аймакта бош эстин жалпы көлөмүн берет. Flag параметри жокко чыгарылат (игнорируется) жана 0 гө барабар болушу керек. Ал реалдык режим менен болгон биргелешүүчүлүк үчүн гана керек. Бош эстин көлөмүнүн маниси байттарда алынат.

GetModuleFileName функциясы	
Procedure GetModuleFileName(Module: Thandle; FileName: PChar; Size: Integer): Integer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Модулдун аткарылуучу файлынын атын кайтарып берет. Бул функция Dos ко ылайык берилген модул жүктөлгөн модулдун аткарылуучу файлынын толук атын берет. Module параметри – бул System модулундагы HInstance өзгөрүлмөсүнөн адатта алынуучу же LoadLibrary аркылуу түзүлгөн модулдун нускасынын көчүрмөсү болуп эсептелет. FileName жана Size параметрлери нөл менен аяктоочу жолго көчүрүлө турган буфердин адресин жана өлчөмүн берет. Алынуучу маани буферге көчүрүлүүчү жолчонун иш жүзүндөгү узундугуна барабар болот.

GetModuleHandle функциясы	
Function GetModuleHandle(ModuleName: PChar): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Берилген модулдун баяндагычын берет. ModuleName параметри модулдун атын билдирет. Жолчо файлдын аты боло албайт. Башкача айтаканда ал маршрутту же кеңейтирилишти кармап тура албайт. Кайтарылып берилүүчү мани берилген модулдун баяндагычы болот же эгерде берилген моментте ушундай модул жүктөлбөгөн болсо анда 0 болот.

GetModuleUsage функциясы	
Function GetModuleUsage(Module: THandle): Integer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Модулга болгон шилтемелер эсептегичинин маанисин берет. Module параметри – бул адатта System модулунун HInstance өзгөрүлмөсүнөн алынган же LoadLibrary функциясы менен түзүлүүчү модулдун нускасынын жолчосу.

GetProcAddress функциясы	
Function GetProcAddress(Module: THandle; ProcName: PChar): TFarProc;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Экспорттолуучу процедуранын же функциянын адресин табат. Бул функция Module модулунда берилген ProcName параметри аркылуу процедураны же функцияны издөөнү аткарат жана көрсөткүчтү процедурага же функцияга кирүү чекитине кайтарат. Module параметри LoadLobrary функциясы арыкылуу түзүлгөн модулдун нускасынын баяндагычын бериши керек. ProcName параметри же 0 менен аяктоочу жолчону көрсөтөт, же иреттик маанини берет. Акыркы учурда ProcName дин чоң сөзү 0 гө барабар, ал эми кичине сөзү иреттик маанин кармап турат. GetProcAddress функциясын DLL ге таандык болгон экспорттолуучу процедуралардын жана функциялардын адрестерин издөө үчүн гана пайдаланууга болот.

GetSelectorBase функциясы	
Function GetSelectorBase (Selector: Word): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Селектордун базалык адресин кайтарып берет. Selector базалык адреси алынышы керек болгон селекторду берет.

GetSelectorLimit функциясы	
Function GetSelectorLimit(Selector: Word): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Selector параметри менен берилген селектордун пределдик маанисин кайтарып берет.

GetVersion функциясы	
Function GetVersion: Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Windoews дун жана DOS тун версияларынын учурдагы номерин берет. Алынуучу маанинин кичине сөзү Windows дун функциясын кармап турат. Чоң байт версиянын кичине номерин берет (эки орундуу ондук номер көрүнүшүндө). Мисалы, Windoews 3.1 үчүн кичине номер болуп 10 эсептелет. Кичине байт версиянын негизги номерин берет.

GetWinFlags функциясы	
Function GetWinFlags: Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Алынуучу GetWinFlags маанисинин биттерин текшерүү менен wf_xxxx желектеринин жардамында алууга мүмкүн болгон эс жана системанын конфигурациясы жөнүндөгү информацияны кайтарып берет. DOS тун корголгон режиминде wf_Standart, wf_Enchanced, wf_LargeFrame жана wf_SmallFrame желектери дайыма нөлгө барабар.

GlobalAlloc функциясы	
Function GlobalAlloc(Flags: Word; Bytes: Longint): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин глобалдык динамикалык бөлүштүрүлүүчү аймагында эстин блогун бөлүп алат. GlobalAlloc функциясы Flags жана Byte параметрлери менен берилген атрибуттары жана өлчөмү менен блокту бөлүштүрөт жана блоктун баяндагычын кайтарып берет.

GlobalAllocPtr функциясы	
Function GlobalAllocPtr(Flags: Word; Bytes: Longint): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогун бөлүп алат жана аны жабат (блокирует). Бул функция GlobalAlloc жана GlobalLock функцияларын бир амалга комбинациялайт жана жүктөлүүчү блоктордун бөлүштүрүлүшү үчүн пайдаланыла албайт.

GlobalCompact функциясы	
Function GlobalCompact(MinFree: Longint): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Глобалдык динамикалык бөлүштүрүлүүчү эсти таңгактайт (упаковывет). GlobalCompact эстин которулуучу блокторун кайрадан иреттөө менен аны чыгарылып ташталуучу (выгружаемый) блоктордон тазалоо менен өлчөмү MinFree байттан кем болбогон бош эстин блогун түзүүгө аракет жасайт. Кайтарылып алынуучу маани – бул таңгакталгандан жана блокторду чыгарып салгандан кийинки эстин эң чоң блогундагы байттардын саны. Ал MinFree ден кичине болуп калышы да мүмкүн.

GlobalDiscard функциясы	
Function GlobalDiscard(Mem: THandle): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Глобалдык эстин блогун чыгарып салат (выгружает). Бул функция Mem параметри менен берилген эстин блогун чыгарып салат. Эстин блогу чыгарылып салынуучу (выгружаемый) (б.а. gmem_Discardable желеги менен бөлүнүп көрсөтүлүүчү) болушу керек, ал эми блок эсптегичи ал үчүн 0 гө барабар болушу керек.

GlobalDOSAlloc функциясы	
Function GlobalDOSAlloc(Bytes: Longint): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Реалдык режимде уруксат берилген (доступный) эстин блогун бөлүп түзөт. Бул функция реалдык режимде да корголгон режимде да уруксат берилген глобалдык эстин блогун бөлүп алат. Bytes параметри блоктун байттардагы өлчөмүн берет. Сызыктуу адрестик мейкиндиктин биринчи мегабайтында блоктун бөлүнүп алынышын камсыз кылат.

GlobalDOSFree функциясы	
Function GlobalDOSFree(Selector: Word): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Реалдык режимде уруксат берилген эстин блогун бошотот. Бул функция мурда GlobalDosAlloc

	функциясынын жардамында бөлүнүп коюлган глобалдык эстин блогун бошотот.

GlobalFix процедурасы	
Procedure GlobalFix(Mem: THandle): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогун жабат (длокирует). Бул процедура Mem параметри менен берилген сызыктуу эстеги блокту жабат жана блокту жабуу эсептегичин (счетчик блокировки) 1 ге чоңойтот. Блок жабылган (эсте фиксирленген) болуп кала берет, качан жабуу эсептегичи GlobalFix ти чакыруу аркылуу нөлгө чейин азаймаынча.

GlobalFlags функциясы	
Function GlobalFlags(Mem: THandle): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогу жөнүндөгү маалыматты берет. Алынуучу маанин чоң байты төмөнкү желектер жөнүндө маалыматтарды кармап турат: gmem_DDEShare, gmem_Discardable, gmem_Discarded, gmem_Not_Banked.

GlobalFree функциясы	
Function GlobalFree(Mem: THandle): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогун кайтарып берет. GlobalFree функциясы Mem параметри менен берилген эстин блогун бошотот жана эстин блогунун баянжагычын уруксат берилбеген (недопустимый) кылып коёт. Эгерде функция ийгиликтүү аткарылса, анда нөлдүк маани кайтарылып алынат. Андай болбогон учурда Mem ге барабар болгон маани алынат. Эгерде эстин объекти блоктонуп (жабылып) коюлса башкача айтканда 0 дөн чоң болгон блоктоо эсептегичине ээ болсо, анда GlobalFree объекти бошотуу үчүн пайдалныла албайт.

GlobalFreePtr функциясы	
Function GlobalFreePtr(P: Pointer): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Блокту ачып салат жана эстин блогун бошотот. Бул

	функция GlobalUnLock жана GlobalFree функцияларын бир амалга комбинациялайт.

GlobalHandle функциясы	
Function GlobalHandle(Mem: Word): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогунун баяндагычын кайтарып берет. Бул функция селектору Mem параметри менен берилген эстин блогунун баяндагычын кайтарып берет. Алынуучу маанинин кичине сөзү блоктун баяндагычын кармап турат, ал эми чоң сөзү блоктун селекторун кармап турат. Эгерде Mem эстин блогун идентификациялабаса, анда алынуучу маани 0 гө барабар болот.

GlobalLock функциясы	
Function GlobalLock(Mem: THandle): Pointer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогун жабат (блокирует). Бул функция Mem параметри менен берилген эстин блогун жабат (блокирует) жана көрсөткүчтү блоктун биринчи байтына которот. Эгерде блок чыгарылып салынуучу (выгружаемый) болбосо, анда анын блоктоо эсептегичи 1 ге чоңоёт жана анын блоктоо эсептегичи 0 гө чейин азаймайынча (GlobalUnLock ту чакыруу менен) блоктун эсте сакталып калышы камсыздалат.

GlobalLRUNewest функциясы	
Function GlobalLRUNewest(Mem: THandle): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Бул функция берилген объекттин анын чыгарылып салыныш (выгрузка) ыктымалдуулугун азайтуу менен эң акыркы “жаңы” позицияга (LRU) жылдырат.

GlobalLRUOldest функциясы	
Function GlobalLRUOldest(Mem: THandle): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Бул функция берилген объекттин анын кийинки чыгарылып салынуучу (выгружаемый) объект кылуу менен эң акыркы “эски” позицияга (LRU) жылдырат.

GlobalNotify функциясы	
Function GlobalNotify(NotifyProc: TFarProc): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Бул функция учурдагы маселе үчүн маалымдоо (уведомление) процедурасын коёт. Эстин администратору эстин маалымдоо процедурасын качан gmem_Notify менен бөлүштүрүлгөн эстин блогу чыгарылып салынышы керек болгондо чакырат.

GlobalPageLock функциясы	
Function GlobalPageLock(Selector: THandle): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Selector параметри менен берилген эстин блогу үчүн бетти (страница) блоктоо эсептегичин азайтат. Блок мурдатан GlobalPageLock функциясынын жардамында жабылган (блоктолгон) болушу керек. Бул функция Windows дун кеңейтирилген режиминде гана мааниге ээ болот. Dos тун корголгон режиминде аталган функция GlobalFix функциясына тиешелеш келет.

GlobalPtrHandle функциясы	
Function GlobalPtrHandle(P: Pointer): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Адреси P параметри менен берилген эстин блогунун баяндагычын кайтарып берет. Эгерде P эстин блогун идентификациялабаса, анда нөлдүк маани кайтарылып алынат.

GlobalReAlloc функциясы	
Function GlobalReAlloc(Mem: THandle; Bytes: Longint; Flags: Word): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Mem параметри менен берилген эстин блогунун өлчөмүн Bytes байттар санына чоңойтот же азайтат. Flags параметри блокту кандай кылып кайра бөлүштүрүү керек экендигин аныктайт. Алынуучу маани блоктун жаңы баяндагычы болуп саналат же 0 болот, эгерде блок кайра бөлүштүрүлбөсө.

GlobalReAllocPtr функциясы	
Function GlobalReAllocPtr(Mem: THandle; Bytes: Longint; Flags: Word): Pointer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогунун өлчөмүн же атрибуттарын өзгөртөт. Бул функция GlobalUnLock, GlobalReAlloc жана GlobalLock функцияларын бир амалга комбинациялайт.

GlobalSize функциясы	
Function GlobalSize(Mem: THandle): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Mem параметри менен берилген эстин блогунун өлчөмүнүн байттардагы маанисин кайтарып берет. Эгерде Mem баяндагычы уруксат берилбеген (недопустимый) болсо, же эстин блогу чыгарылып ташталган болсо, анда нөл алынат.

GlobalUnfix функциясы	
Function GlobalUnfix(Mem: THandle): Bool;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогунун блоктолушун ачат. Бул функция Mem параметри менен берилген эстин блогунун блоктоо эсептегичинин (счетчик блокировки) маанисин азайтат. Эгерде блоктоо эсептегичинин мааниси 0 болуп калса, анда блок ачылат (блоктолушу алынып салынат) жана объект которулат же чыгарылып салынат. Эгерде блоктоо эсептегичинин мааниси нөл болуп калса, анда алынуучу маани False ге, андай болбогондо True ге барабар болот.

GlobalUnlock функциясы	
Function GlobalUnlock(Mem: THandle): Bool;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Эстин блогунун блоктолушун ачат. Бул функция Mem параметри менен берилген эстин блогунун блоктоо эсептегичинин маанисин азайтат. Эгерде блок чыгарылып салынуучу (выгружаемый) болсо, анда анын блоктоо эсептегичинин мааниси 1 ге азаят. Эгерде бул эсептегичтин мааниси нөлгө барабар болуп калса, анда

	алынуучу маани False ге, андай болбогондо True ге барабар болот.

HiByte функциясы	
Function HiByte(A: Word): Byte;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Word тибиндеги А параметринин маанисинин чоң байтын кайтарып берет.

HiByte функциясы	
Function HiWord(A: Longint): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	HiWord функциясы Longint тибиндеги А параметринин чоң сөзүн кайтарып берет.

LoadLibrary функциясы	
Function LoadLibrary(LibFileName: PChar): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Библиотекалык модулду жүктөйт. LibFileName параметри DLL дин жүктөлүүчү файлынын аты менен болгон нөл менен аяктоочу жолчону көрсөтүп турушу керек. Алынуучу маани – бул функция ийгиликтүү аткарылган учурда жүктөлгөн библиотеканын баяндагычы болот, андай болбогондо катанын коду болот.

LoadResource функциясы	
Function LoadResource(Instance: THandle; ResInfo: ThaHdle): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Instance параметри менен берилген аткарылуучу файлдан (из файла) ResInfo параметри менен көрсөтүлгөн ресурсту эске жүктөйт жана глобалдык эстин блогунун ресурсун кармап турган баяндагычты кайтарып берет. Instance параметри – бул адатта System модулунун HInstance өзгөрүлмөсүнөн алынган же LoadLibrary функциясы менен түзүлгөн модулдун нускасынын баяндагычы. ResInfo параметри ресурс жөнүндөгү маалыматтын FindResource камтылуучу программасы аркылуу түзүлгөн баяндагычы болушу керек.

LoadString функциясы	
Function LoadString(Instance: THandle; ID: Word; Buffer: PChar; BufferMax: Integer): Integer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Instance параметри менен берилген аткарылуучу файлдан ID параметри менен берилген жолчолук ресурсту эске жүктөйт. Instance параметри – бул адатта System модулунун HInstance өзгөрүлмөсүнөн алынган же LoadLibrary функциясы менен түзүлгөн модулдун нускасынын баяндагычы. ID параметри жүктөлүүчү жолчонун бүтүн сандык баяндагычы. Buffer жана BufferMax параметрлери нөл менен аяктоочу жолчо көчүрүлүп жаткан буфердин адресин жана өлчөмүн берет. Алынуучу маани буферге көчүрүлүүчү символдордун саны болот же эгерде ресурстар жолчосу жашабаса анда 0 болот.

LoByte функциясы	
Function LoByte(A: Word): Byte;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Word тибиндеги A параметринин маанисинин кичине байтын берет.

LockResource функциясы	
Function LockResource(ResData: THandle): Pointer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	ResData параметри менен берилген ресурсту бошотот (жабат, блокирует) жана көрсөткүчтү ресурстун биринчи байтына кайтарат. Эгерде ресурс чыгарылып салынган (выгруженный) болсо, анда анын блоктоо эсептегичи 1 ге чоңоёт жана бул эсептегич UnLockResource функциясын чакыруунун жардамында 0 гө чейин азаймайынча ресурс эсте кала берет.

LockSegment функциясы	
Function LockSegment(Segment: Word): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Чыгарылып салынуучу (выгружаемый) сегментти

	блоктойт. Segment параметри блоктонуучу сегменттин селекторун берет. Эгерде сегмент чыгарылып салынуучу болсо, анда блоктоо эсептегичи 1 ге чоңоёт жана UnLockSegment чакыруу аркылуу 0 гө барабар болуп калмайынча сегмент эсте кала берет. Чыгарылып салынуучу сегменттер үчүн LockSegment аракет этпейт. LockSegment функциясы ийгиликтүү аткарылган учурда нөл эмес маанини берет, ал эми эгер сегмент чыгарылып салынган же ката болуп өткөн болсо, анда алынуучу маани нөлгө барабар болот.

LoWord функциясы	
Function LoWord(A: Longint): Byte;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Longint тибиндеги A параметринин маанисинин кичине байтын берет.

MakeLong функциясы	
Function MakeLong(A, B: Word): Longint;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Word тибиндеги эки мааниден Longint тибиндеги маанини түзөт. A жана B параметрлери алынуучу маанинин кичине жана чоң сөздөрүн берет.

MessageBox функциясы	
Function MessageBox(WndParent: Thandle; Txt, Caption: PChar; TextType: Word): Integer;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Билдирүүлөр терезесин экранга чыгарат жана аны тейлейт. Бул функция берилген бөрктү жана билдирүүнү кармап турган диалогдук терезени чыгарат жана аны тейлейт, ошондой эле mb_xxx константалары менен баяндалган алдын ала аныкталган кнопкаларды чыгарат.

SetSelectorBase функциясы	
Function SetSelectorBase(Selector: Word; Base: Longint): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Base параметри менен берилген сызыктуу адрес үчүн Selector параметри менен берилген селектордун базалык

	адресин коёт (установливает). Функция ийгиликтүү аткарылган учурда алынуучу маани Selector го барабар болот, андай болбогондо 0 гө барабар болот.

SetSelectorLimit функциясы	
Function SetSelectorLimit(Selector: Word; Base: Longint): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Limit параметринин маанисине барабар болгон Selector параметри менен берилген селектордун пределдик маанисин коёт (орнотот). Limit мааниси $10000 (64 Кб) дан ашпастыгы керек. алынуучу маани дайыма 0 гө барабар.

SizeOfResource функциясыSelectorResourc	
Function SizeOfResource(Instance, ResInfo: THandle): Word;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Instance жана ResInfo параметрлери менен берилген ресурстун өлчөмүн берет. Instance параметри – System модулунун HInstance өзгөрүлмөсүнөн алынуучу же LoadLibrary функциясы менен түзүлүүчү модулдун нускасы. ResInfo параметри FindResource камтылуучу программасы менен алынган ресурс жөнүндөгү маалымат болушу керек. Эгерде ресурс табылбаса, анда алынуучу маани 0 гө барабар болот.

UnLokcResource функциясыSelectorResourc	
Function UnLokcResource(ResData: THandle): Bool;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	ResData параметри менен берилген ресурска коюлган блокту ачат (разблокирует). ResData параметри LoadResource менен түзүлгөн ресурстун баяндагычы болушу керек. Эгерде ресурс чыгарылып салынуучу болсо, анда анын блоктоо эсептегичи 1 ге азаят. Эгерде блоктоо эсептегичи 0 болуп калса, анда ресурска коюлган блок алынат жана ал чыгарылып салынат (выгружается). Эгерде ресурстун блоктоо эсептегичи 0 гө чейин азайтылган болсо, алынуучу маани False болот, андай болбогондо True мааниси алынат

UnlockSegment функциясы	
Function UnlockSegment(Segment: Word): THandle;	
Пайдаланылуучу режим:	Корголгон режим, Windows
Арналышы:	Сегментке коюлган блокту ачат (разблокирует). Segment параметри блоктоо ачыла турган сегменттин селекторун берет. Эгерде ресурс чыгарылып салынуучу болсо, анда анын блоктоо эсептегичи 1 ге азаят. Эгерде блоктоо эсептегичи 0 болуп калса, анда ресурска коюлган блок алынат жана ал чыгарылып салынат (выгружается). Эгерде ресурстун блоктоо эсептегичи 0 гө чейин азайтылган болсо, алынуучу маани False болот, андай болбогондо True мааниси алынат. Чыгарылып салынбоочу (невыгружаемый) сегменттер үчүн UnlockSegment функциясы аракет этпейт.

9. ОБЪЕКТТИК-ОРИЕНТИРЛЕНГЕН ПРОГРАММАЛООНУН ПРОБЛЕМАЛАРЫ. ВЕЗЕНСПРОГРАММАЛОО

Көп тил билүү – демек бир кулпуга түшө турган көп ачкычтуу болуу.
(Ф. Вольтер)

9.1. Программалоо тилдеринин тарыхы

Жалпыга маалым болгондой, алгачкы программалоо тилинин пайда болушунан бери өтө деле көп убакыт өтпөсө да программалоо тилдеринин теориясында бир топ технологиялар иштелип чыгылып, ушул технологияларды реализациялоо үчүн ар түрдүү программалоо тилдери сунуш кылынып программистердин сыноосуна коюулуп келе жатат. Программалоо тилдеринин көбөйүп кетиши, алардагы реализацияланган ар түрдүү каражаттар кандайдыр бир деңгээлде чаташууну пайда кылчудай.

Бул жерде программалоо тилдеринин кайсыл абалда турганлыгын иликтөөгө аракет кылып көрөлү. Төмөнкү жадыбалда программалоо тилдеринин тарыхындагы негизги окуялар келтирилген.

Жыл-дар	**Окуялар**	**Тилдин мүнөздөмөлөрү**
1941	Германияда Конрад Цузе электромеханикалык релеге негизделген Z3 машинасын курган	
1943	Англияда Collosus деп аталган биринчи ЭЭМ иштелип чыккан	
1944	Джон Эккерт эсте сакталуучу программа концепциясын баштаган	
1949	Алгачкы Ассемблер тили ойлонуп табылган	Ассемблер
1951	Грейс Хоппер программа үчүн ыңгайлуу алгебралык формада жазылуучу алгачкы	

	трансляторду иштеп чыккан	
1955	Fortran	Процедуралык
1958	ALGOL 58	Процедуралык
1959	LISP	Функционалдык
1961	GPSS	Логикалык (Деклоративдик)
1967	Simula 67	Императивдик тилдин объекттик-ориентирленген кеңейтилиши
1970	Prolog	Логикалык (Деклоративдик)
1971	Pascal	Процедуралык
1972	C	Процедуралык
1981	Биринчи персоналдык компьютер (IBM PC)	
1986	C++	Императивдик тилдин объекттик-ориентирленген кеңейтилиши
1986	Object Pascal	Императивдик тилдин объекттик-ориентирленген кеңейтилиши
1988	CLOS	LISP тин объекттик-ориентирленген кеңейтилиши
1996	Java	Императивдик тилдин объекттик-ориентирленген кеңейтилиши
2002	C#	Императивдик тилдин объекттик-ориентирленген кеңейтилиши

Бул жерден биз көрүп тургандай бүгүнкү күндө пайдаланылып жаткан программалоо концепциялары 1967-жылга чейин эле пайда болгон. Азыркы персоналдык компьютерлерге караганда миллион эсе жай иштеген, тез-тез бузулуп жана бүтүндөй бир чоң бөлмөнү ээлеп турган ENIAC машинасы пайда болгондон 22 жылдан кийин эле бүгүнкү күндө пайдаланылып жаткан программалоо концепциялары калыптанган, жана ушул идеяларды реализациялоочу тилдер үчүн компиляторлор жазылган. Кийинки кырк жылда радикалдуу жаңы концепциялар алынып чыгылган жок (программалоо тилдери

жөнүндө сөз болуп жатат), программистердин санынын жана компьютерлердин өндүрүмдүүлүгүнүн алда канча өскөндүгүнө карабастан болгону бар болгон концепциялар гана өркүндөтүлүп келди.

9.2. Айрым объекттик-ориентирлеген тилдер боюнча баяндама

ООП тун азыркы абалын баалоо үчүн негизги объекттик-ориентирлеген тилдерди кыскача карайбыз.

Simula

Биринчи объекттик-ориентирлеген тил – Simula67 тили 1967-жылы пайда болгон. Ал Норвегия эсептөө борборунда (Nask Regnesentrol) жана Осло университетинде Кристен Нигард (Kristen Nygaard) жана Оле-Йохан Дал (Ole-Jahon Dahl) тарабынан түзүлгөн. 1986-жылы Simula 67 тили Simula тили деп кайра аталган. Simula 67 – 60-жылдары дискреттик окуяларды моделдештирүү үчүн түзүлгөн Simula 1 тилинин мураскери. Simula 1 ден айырмаланып Simula 67 – бул көп максаттуу ОО (объекттик-ориентирлеген) тил. Эки тил тең ошол кездеги эң популярдуу болгон ALGOL60 тилинин кеңейтирилиши болуп саналат.

Simula тилинде класс талааларын, усулдардан жана класстын телосунан турат.

Класстын телосу кандайдыр бир деңгээлде класстын конструкторун алмаштырат. Объектти түзгөн учурда класстын телосунуну операторлору аткарылат.

Simula тилинде ООП тун бардык түшүнүктөрү – мурастоо, полиморфизм, абстракттык класстар реализацияланган.

Андан сырткары, Simula да шериктеш программа (сопрограмма) түшүнүгү бар. Шериктеш программалар класстардын негизинде реализацияланган. Алардын жардамында талдоочулар иштин параллелдүүлүгүнүн аналогуна жетишишкен (чындыгында шериктеш программалар – бул параллелизм эмес, анткени алар шериктеш программалардын бири-биринен болгон катуу көз карандылыгына карата иштейт). Кийинчерээк, качан мультиагымдык (мультипотоковый) операциялык системалар пайда болгондон кийин параллелдүүлүк менен иштөөнү ошого которушкан.

Бардык ОО-тилдер Simula67 нин урпактары болуп саналат, ошондуктан ал тилдин маанисин чындыгында абдан чоң.

Simula али өлүк тилге айлана элек, ал ушул күнгө чейин программалык камсыздандырууну иштеп-талдап чыгууда колдонулуп келе жатат (негизинен Скандинавияда).

Smalltalk

Экинчи ОО-тил Smalltalk болуп калды. ООП тун базалык концепцияларын Smalltalk тили Simula дан алган, бирок ошол эле учурда бул тилде көптөгөн жаңы түшүнүктөр киргизилген:

- Smalltalk та бардык өзгөрүлмөлөр – бул объекттер (ал тургай сандар да);
- Базалык класс – object бар;
- Класстардын өздөрү – метакласстардын объекттери;
- Класстардын менчик усулдары бар;
- Класстардын талаалары жабык талаалар гана боло алышат.

Smalltalk тилинде типтерди статикалык текшерүү жок, ошондуктан бардык зарыл болгон амалдардын колдонумдуулугун тешерүүлөр программанын иши учурунда жүргүзүлөт. Дал мына ушул кырдаал Smalltalk та жазылган программалардын өндүрүмдүүлүгүнүн (производительность) салыштырмалуу төмөн болушуна алып келген. Бирок бул нерсе ООП үчүн Smalltalk тын маанисин түшүрбөйт: метакласстар жана класстардын бирдиктүү иерархиясы – эң маанилүү киргизилген жаңылык болуп саналат. Ушул түшүнүктөрдүн жардамында программалардын жалпылыгына жеңил эле жетишүүгө болот.

Delphi (Object Pascal)

Pascal үчүн көптөгөн объекттик-ориентирлеген чечимдердин бар экендиги белгилүү. Биз бул жерде Delphi үчүн гана кыскача баа берип өтөлү.

Delphi тили Turbo Pascal тилинин кеңейтирилиши катары түзүлгөн жана алгач Object Pascal деп аталган. Ошондуктан Delphi программалык кодду процедуралык да, ОО-тилде да жазууга мүмкүнчүлүк берет.

Delphi көп нерселерди Smalltalk тан алган. Бардык класстардын TObject тен келип чыгышы – бул практикалык мааниде да акыл-эстүүлүк менен кабыл алынган нерсе, анткени ар кандай класстардын объекттери үчүн жалпы болгон усулдарды жазууга мүмкүнчүлүк берет, концептуалдык мааниде да, акыл-эстүүлүк менен алынган нерсе, анткени ар кандай учурда ар бир объект жок дегенде өзүнө болгон жана класстын дескрипторуна болгон шилтемени кармап турат, ошондуктан классты анын обьеттери ушул талааларды жана

дагы бир нече базалык усулдарды кармагандай кылып класстарды жасоо – бул жетишээрлик акылдуу чечим.

Шилтемелерди пайдалануу процедуралык программалоодо пайдаланылуучу көрсөткүчтөрсүз иш кылууга мүмкүнчүлүк берет.

Delphi де ООП тун бардык базалык мүмкүнчүлүктөрү бар.

- Талааларга private, public, protected деген идентификаторлорду берүү менен болгон инкапсуляция;
- Бирдик (одиночное) мурастоо;
- Динамикалык байланыштыруунун жардамында реализацияланган полиморфизм.

Delphi де объекттин тиби жөнүндөгү маалыматты аткарылып жаткан учурда алуу мүмкүнчүлүгү (RTTI – Run Time Type Information) жетишерлик табийгий түрдө реализацияланган – TObject базалык классында класстын дескрипторуна болгон шилтемелерди алуу мүмкүн. Метакласстар жана is, as операторлору объекттердин эмес класстардын денгээлинде аракет этүүгө мүмкүнчүлүк берет, ошондуктан программалардын жалпылыгына жеңил эле жетишүүгө болот.

Интерфейстер көптүктүк мурастоону пайдаланбастан туруп (C++ тилинин чөйрөсүн карагыла) класстардын бир кыйла структураланган иерархиясын жасоого мүмкүнчүлүк берет.

Касиеттер жетишээрлик чоң болуп реализацияланган, бирок анткен менен идея жаман эмес.

Эми жетишпестиктери жөнүндө айталы.

1. Ошол эле модулда реализацияланган башка класстарда класстын private – мүчөлөрүнө кайрылууга уруксаттын болушу. Бул Delphi нин модулдук өтмүшүнүн саркындысы. Бул жетишпестиктен кутулуу үчүн Delphi 2005 те strict private, strict protected сыяктуу элементтердин көрүнүмдүүлүгүнүн аныктагычтарын киргизишкен.
2. Деструктор башка усулдардан өзгөчө эч нерсеси менен айырмаланбайт, ошондуктан кошумча кызматчы сөзсүз эле иштей берүүгө болот.
3. TObject классындагы базалык класстардын чоң жыйындысынын болушу: алардын көпчүлүгү башка усулдардын камтылуучу усулу (подметод) болуп, өз алдынча жетишээрлик сейрек учурларда керек болот.
4. ООП тун проблемаларын талдаган айрым адистер Delphi тилин иштеп чыккандар анын Turbo Pascal менен толук биргелешүүчүлүгүн камсыз кылууга умтулбашы керек эле дешет. Ошондон улам Delphi тили goto операторун жана эскирип калган object тибин, себеби класстар киргизилген,

өзүнө алган дешет. Ошондой эле “типтештирилген константа” түшүнүгүн эске түшүрсөк болот, чындыгында const бөлүгүндө баяндалганы менен ал деле өзгөрүлмө. Эмне үчүн маани менен инициалдаштырылган өзгөрүлмөнүн типтештирилген константа деп аталган түшүнүксүз. Ушундай жетишпестиктирдин бардыгынан улам көпчүлүк программистердин түзүүчүлөрдү күнөөлөгөнүн туура эле деп эсептөөгө болот.

5. Класс өзгөрүлмөлөрү жок.

Дагы бир татаал суроо көрсөткүчтөрдүн жана тиркелген Ассемблердин кереги барбы. Бир жагынан Delphi де класстардын иерархиясы TObjec тен башталгандыктан жана обьетер менен иштөө шилтемелердин гана жардамында жүргүзүлгөндүктөн ОО-тиркемелерди түзүү үчүн көрсөткүчтөрдүн кереги жок. Экинчи жактан, көрсөткүчтөр жана тиркелген ассемблер абдан жардам бериши мүмкүн эле, эгер тиркемелердин жогорку эффективдүүлүгүнө жетишүү, мисалы, компиляторлорду жазуу үчүн, талап кылынса.

Көпчүлүк адистердин көз карашы менен макулдукта процедуралык жана ОО-ыкмалар бири-бирине жолтоо болбогондон кийин көрсөткүчтөрдү жана тиркелген ассемблерди алып салуунун кереги жок. Көрсөткүчтөрдүн кереги жок болсо, демек аларды колдонбой эле коюу керек. Колдонгула деп эч ким зордуктабайт.

Delphi 2005 те болсо .Net платформасына ылайыктоо максатында эч кандай жаңы нерселерди алып келбеген каражаттардын тобу киргизилген, болгону тилге көп жүк артылып, көлөмү чоңоюп кеткен. Ошону менен түзүүчүлөр канчалык аракет кылгандары менен кереги жок болгон жаңы киргизилген каражаттардын саны боюнча C++ тан аша алган эмес.

C++

C++ тили – бул 1986-жылы Бьярн Страуструп тарабынан түзүлгөн бир кыйла популярдуу ОО-тилдеринин бири. Ал 1972-жылы Денис Ритчи тарабынан Unix операциялык системасын талдап иштеп чыгуу үчүн түзүлгөн C тилинин кеңейтирилиши болуп эсептелет.

C тили системалык программалоо үчүн түзүлүп, ал Ассемблерде кодду азыраак түзгөнгө мүмкүнчүлүк берүү менен программистерди сүйүнткөн. C тилинде таптакыр процедуралык программалоого тиешеси болбогон механизмдер бар – мисалы, макростор. TP да ал жок, ошондуктан ушул макросторду кеңирирээк карап өтөлү.

Макростор

Макростор – бул параметрлештирилген символдук константалар. Компиляциянын башталышы алдында компилятор макростун

кирүүсүн (катышуусун) ага параметрлерди коюу менен жолчого алмаштырат.

Макросторду #define директивасынын жардамында түзүүгө болот. Төмөнкү мисалда үч макрос – aPb1, aPb2, aPb3 макростору киргизилген.

Мисал: C++ тилиндеги макростор.

```
#include <iostream.h>; // iostream библиотекасын кошуу
# define aPb1(a,b) (a*b)
# define aPb2(a,b) a*b
# define aPb3(a,b) ((a)*(b))
void main() // негизги программа
{
  double c=(double) 10/ aPb1(1+2, 4); // c=10/(1+2*4)
  double d=(double) 10/ aPb2(1+2, 4); // d=10/1+2*4
  double e=(double) 10/ aPb3(1+2, 4); // e=10/((1+2)*4)
  cout << “c=” << c << “d=” << d << “e=” << e;
```

Мисалы, aPb1 макросунун аныкталышын талдап көрөлү:

#define aPb1(a,b) (a*b)
aPb1(a,b) – бул макростун бөркү
(a*b) – бул кийин макрос алмаштырыла турган маани.

Айрым учур катары, aPb1(1+2, 3+4) учуру (1+2*3+4) дегенге өзгөртүлүп түзүлөт. Программанын телосунда баяндалган үч макростун ортосундагы айырмачылыктар демонстрацияланат. Ушул үч макростун ичинен aPb3 гана сандарды көбөйтүүнү туура реализациялайт.

(double) – бул типтерди келтирүү (приведение). C тилинде эки бүтүн сандын катышы – бүтүн сан, ошондуктан титерди келтирүүсүз ишке ашыруу мүмкүн эмес.

Бул келтирилген мисал – жетишээрлик эле жөнөкөй. Мисалы, төмөнкү туюнтмада компилятор алмаштырууларды кандай тартипте аткарат:

aPb3(aPb3(1,2),3) ?

Жооп мындай: адегенде жолчо камтылган макростун ордуна коюлат, андан кийин тышкы макрос алмаштырылат, башкача айтканда жыйынтык төмөнкүдөй болот:

aPb3(aPb3(1,2),3) → aPb3(((1)*(2)), (3))

Ошондуктан макростор менен иштөө деген эмне экендигин билүү жетишсиз, андан сырткары кошумча эрежелердин, алардын бири – ордуна коюулардын кезекчилигин билүү керек.

Иш жүзүндө макростор камтылуучу программалардын алтернативасы болуп эсептелет. Алардын артыкчылыгы мына мында: камтылуучу программанын телосун макростун жардамында программалык кодго түздөн-түз киргизип жайлаштырууга болот, башкача айтканда, камтылуучу программаны чакырууга жана параметрлерди берүүгө убакыт кетирүүнүн кереги жок. Бирок, макростор өзүнүн ишин компиляция кезинде аткаргандыктан, алар өзгөрүлмөлөрдүн маанисин текшере албайт, а бул болсо макростордун колдонулушуна чоң чектөөлөрдү коюп коёт (айрым учур катары, алардын жардамында рекурсияны жасоого болбойт). Андан сырткары, камтылуучу программа чоң болсо, анда аны чакырууга кеткен коромжу аз болот, ал эми аны программалык кодго коюу анын көлөмүнүн чоңоюп кетишине алып келет.

Ошентип, макростор анчалык чоң эмес камтылуучу программаларды алмаштырууга гана кызмат кыла алат. Бирок бул учурда да алардан кутулууга болот: компиляторду ал бардык анчалык чоң эмес камтылуучу программаларды кодго тизип жайгаштыра тургандай кылып жазуу же атайын директиваны киргизүү (айтмакчы C++ та мындай директива бар ал – inline) керек болот, ал болсо компиляторго камтылуучу программанын телосун программалык кодго жайгаштыруу керекпи же жокпу ошону айтып турушу керек. Балким системалык программалоодо макростор кээде керек болуп калышы мүмкүн, бирок ОО-тилинде алар ачыктан ачык ашыкча.

C++ тилинин базалык ОО-мүмкүнчүлүктөрү

Delphi ден айырмаланып C++ тилинде объекттерге болгон шилтемелер эмес, объекттердин өздөрү түзүлөт. Ошондуктан, эгерде силер динамикалык эсте объектти түзгүңөр келсе, анда көрсөткүчтөрдү пайдаланышыңар керек. Бул болсо процедуралык жана ОО-тилдердин аралашып кетишине эле алып келбестен, дагы кошумча маселелер пайда болот. Мисалы, объекттерди келтирүү (приведение) жана көрсөткүчтөрдү объекттерге келтирүү түшүнүктөрүн өз-өзүнчө ажыратып так аныктап алуу керек.

C++ та негизги түшүнүк болуп класс түшүнүгү эсептелет. C++ та класстын элементтерине кайрылууларды чектөө мүмкүнчүлүгү бар. Ал үчүн private, public, protected сөздөрү кызмат кылат.

Private – бул элементке класстын ичинде гана кайрылууга болот дегенди билдирет. А бул болсо модулдун ичинде private – элементке кайрылууга мүмкүнчүлүк берүүгө караганда акыл эстүү чечим.

C++ тилинде статикалык функциялар (Delphi деги класстын функциясынын өзүндөй эле) жана статикалык өзгөрүлмөлөр бар. Delphi де статикалык өзгөрүлмөлөр жок. Статикалык

өзгөрүлмөлөрдүн жардамында, мисалы, берилген класстын түзүлгөн объекттеринин санын контролдоого болот. Муну статикалык өзгөрүлмөлөрсүз аткаруу кыйын.

C++ та жалгыздап мурастоо (одиночное наследование), динамикалык байланыштыруу жана полиморфизм бар.

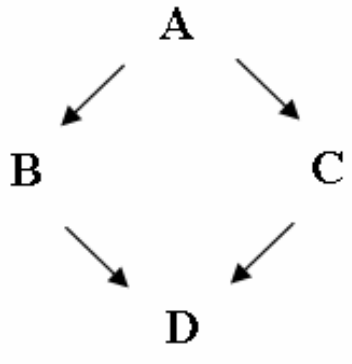
Бирок C++ та Delphi деги TObject тин аналогу болгон базалык класс жок. Бул болсо проблемаларды пайда кылат: базалык класстын болушу типсиз көрсөткүчтөрдү пайдалануунун зарылчылыгынан кутулууга мүмкүнчүлүк берет, башкача айтканда базалык класска болгон шилтеменин өзү зарыл болгон жалпылыкты берет. Натыйжада жалпы процедураларды жазуу үчүн башкача жол менен кетүүгө туура келет.

C++ тилинин кеңейтирилген мүмкүнчүлүктөрү

Мурастоо (жалгыздап – одиночное), инкапсуляция, полиморфизм Simula 67 тилинен баштап бардык ОО-тилдерде бар. Ушул жерде C++ тилиндеги ООП тун кээ бир өзгөчөлүктөрүн карап кетели. Зарылдыкка жараша кеңири мисалдарсыз жана түшүндүрмөлөрсүз факттарды айтуу менен гана чектелебиз.

1. Көптүктүк мурастоо

C++ тилинде кадимки мурастоодон башка дагы көптүктүк мурастоо (множественное наследование) бар. Бул учурда мурастоочу класстардын жок дегенде бирөөсүндө puplic жана protected катары жарыяланган бардык талаалар жана усулдар класс-мураскорго өтөт. Өзүнө тартып турганы менен көптүктүк мурастоонун проблемалары андан алынган реалдуу пайдадан ашып кетет.



Ромб көрүнүшүндөгү мурастоо

Талдап иштеп чыгуучуларга ромб көрүнүшүндөгү мурастоонун проблемаларынан качуу үчүн виртуалдык базалык класстар механизмин кошууга туура келген.

2. Амалдарга жана операторлорго ашыкча жүк артылышы (перегрузка)

C++ та операторлорго кошумча жүк артууга болот. Мисалы, блоктор менен иштөө үчүн классты долбоорлоп (проектирлеп) жатып аларды бөлүктөр үчүн да пайдаланууга мүмкүн болсун үчүн кайра аныктоого болот.

Албетте амалдарга ашыкча жүк артуу механизмин пайдалануу көбүнчө программаларды бир кыйла кооз кылууга мүмкүнчүлүк берет.

Айрым адистердин ою боюнча [11] амалдарга кошумча жүктүн артылышы - бул ашыкча эле нерсе: ал математикалык объекттерди моделдөөчү класстар менен иштөө үчүн жакшы, ал эми калган учурларда кошумча жүк артуу программанын маңызын тескерисинче түшүнүксүз кылып коюшу мүмкүн.

Андан сырткары, ашыкча жүк артуу менен иштөөдө кыйынчылыктар бар: ашыкча жүк артуу мүмкүн болсун үчүн талдоочуларга дос функция (дружественная функция) түшүнүгүн киргизүүгө туура келген. Дос функция – бул башка класстын функциясы болуу менен берилген класстын жабык элементтерине кайрылуу укугуна ээ болгон функция. Бул учурда демек private түшүнүгүнүн маниси жоголот да инкапсуляция эрежеси бузулат.

3. Жалпыланган программалоо

Жалпыланган программалоо дегенде конкреттүү эмес, каалагандай класс үчүн пайдаланууга боло турган программаларды жазууну түшүнүшөт. Процедуралык деңгээлде жалпылык типсиз көрсөткүчтөрдү колдонуу менен жетишилет.

Delphi де ООП тун деңгээлиндеги жалпылыкка үч фундаменталдык идеянын эсебинен жетишилет.

1. Бардык класстар – TObject тин мураскорлору;
2. Ар бир объект үчүн анын тибин программа аткарылып жаткан убакта билүү мүмкүн;
3. Метакласстар бар.

Бул идеялар жөнөкөй да, кооз да болуп реалдуу абалга тиешелеш келет, анткени ар бир объект класска болгон шилтемени кармап турат.

Бирок C++ та базалык класс да, метакласстар да жок, анын лрдуна объекттин тибин билиш үчүн бир нече жардамчы операторлор кригизилген. Класстарды берүү үчүн (для передачи) шаблондор киргизилген (бул болсо макростор идеясынын өркүндөтүлүшү).

C++ тили жөнүндө дагы башка толгон токой нерселерди айта берүүгө болор эле, анда керектүү болгон нерселер жетишээрлик көп, ошол эле учурда керексиз нерселер да аз эмес.

Java

Эми Java жөнүндө сөз кылалы. Бул тил 1996-жылы Sun Microsystems фирмасы тарабынан иштелип чыгылган.

Тилдин синтаксиси C тилинен алынган, ошондуктан Java тилин көп учурда C тилинин кеңейтилиши «тазартылган, жакшыртылган C++» деп аташат.

Иш жүзүндө ООП жаатындагы бөлүктөрү боюнча Java тили Delphi ге көп жагынан окшош.

Java да, Delphi да :

- Объекттердин өздөрү эмес, объекттерге болгон шилтемелер пайдаланылат;
- Object – базалык классы бар;
- Мурастоо – жалгыздап гана;
- Интерфейстер бар;
- Аткарылуу убагында объекттин тибин аныктоого мүмкүнчүлүк берүүчү RTTI (Run Time Type Information) механизми бар;

Java да метакласстар жок, ошондон улам жалпыланган программалоо жаатындагы Java нын мүмкүнчүлүктөрү бир топ төмөндөп кеткен.

Java таза ОО-тил катары түзүлгөн, ошондуктан:

1. Java да таптакыр көрсөткүчтөрдү пайдаланууга болбойт;
2. Макростор жок;
3. Goto оператору жок.

Бирок, Java да бир нече циклден дароо чыгып кетүүгө мүмкүнчүлүк берүүчү (б.а. break операторунун аналогу, бирок көп деңгээлдүү) структуралык оператор киргизилген. Андан сырткары синтаксиси жетишээрлик баш айлантма (канткен менен Java C нин кеңейтилиши да) болуп, ал бир топ көлөмдүү (чоң) болуп калган (C ге салыштырмалуу).

Java тилинин ОО-мүмкүнчүлүктөрү болжол менен Delphi тилиндей эле, бирок процедуралык программалоого тиешелүү болгон каражаттардан баш тартуу, Java тилинин колдонулуш аймагын чектеп койгон.

9.3. Компиляция же интерпретация

Программалык кодду машиналык тилге которуучу программалардын эки тиби бар: компиляторлор – алар бүтүндөй программалык кодду машиналык тилге которот, жана интерпретаторлор – алар жогорку деңгээлдеги тилдин кезектеги операторун машиналык инструкцияга которот жана аткарат, андан кийин кийинки операторго өтөт.

Эгерде программалык код компиляциялана турган болсо, анда биз кандайдыр бир процессордун машиналык тилиндеги даяр программаны алабыз. Бул программа ушундай процессору бар каалагандай компьютерде аткарылышы мүмкүн жана ага баштапкы код да, компилятордун өзү да керек болбой калат. Бирок, эгер компьютердеги процессор башка болсо, анда бул программа бул процессордо аткарыла албайт (тагыраак айтканда ал программа бул процессор үчүн дегеле программа боло албайт, анткени программа түшүнүгү процессордо аткарылышы мүмкүн болгон командалардын удаалаштыгы катары аныкталат).

Проблеманы мындайча чечүүгө болор эле: баштапкы кодду берип (эч ким окуй албасы үчүн кандайдыр бир жол менен коддолгон болушу да мүмкүн), андан кийин долбоорду ушул долбоор аткарылышы керек болгон машинада компиляциялоо менен, бирок, чоң долбоорду компиляциялоо көп убакытты алат, ошондуктан бул жол эң жакшы жол эмес.

Эгерде программалык кодду машиналык инструкцияга которуу үчүн интерпретатор пайдаланылса, ал интерпретатор проект аткарыла турган машинада орнотулган болушу керек. Ошондо баштапкы кодду ушул машинага берүүгө мүмкүн болуп, андан соң операторлор биринин артынан экинчиси интерпретатордун жардамында аткарылат. Программанын иштөө ылдамдыгы компиляцияланган кодду пайдалануудагыга караганда төмөн болушу мүмкүн, бирок анткен менен баштапкы код каалагандай процессорлуу машинада аткарылышы мүмкүн болот (бирок, компьютердин ар бир тиби үчүн өзүнүн интерпретатору керек болот).

Интерпретация механизмин калтыруу жана ошол эле учурда программанын аткарылышын жогорку ылдамдыгына жетишүү үчүн комбинацияланган мехинизмдер пайдаланылат.

Мисал катары JVM (Java Virtual Machine) механизмин карайбыз.

Java тилиндеги баштапкы код байт-коддогу – аралык төмөнкү деңгээлдеги тилдеги программага компиляцияланат. Бул программалык код JVM дин байт-коддун интерпретаторунун

жардамында аткарылат. Java да жазылган программанын компьютерде аткарыла алышы үчүн анда JVM орнотулган болушу керек.

Албетте Java дагы программалар созсүз байт-кодго өзгөртүлүп түзүлүшү, андан кийин интерпретацияланышы керек деп эсептөөгө болбойт – чындыгында Java тили үчүн компилятор да жазса болмок.

Sun фирмасы жөн гана башынан баштап эле тилдин интерпретациланууучулугуна басым жасаган.

9.4. Объекттик-ориентирлеген тилдердин жетишпестиктери

Өзү уюшулуучу системаларды баяндоо

Модулдук программалоого салыштырмалуу ООП тун негизги артыкчылыгы мына мында: программист класстардын жана объекттердин терминдеринде ой жүгүртө алат, а бул болсо жөн гана камтылуучу программалардын (подпрограммы) топтому (коллекциясы) катары каралаган модулдар менен ой жүгүртүүгө караганда алда канча реалдуулукка жакын. Анан дагы, ООП тун принциптери Prolog жана Lisp сыяктуу логикалык жана функционалдык тилдердин негизинде ишке ашырыла алат.

Бирок ООП да бардык жетишпестиктерден арылган эмес. Объекттер бири-биринин усулдарын тоскоолдуксуз чакыра алат дегендин өзү биз реалдуу турмушта көрүп турган нерселер менен көбүнчө карама-каршы келет.

Качан жансыз объекттерди моделдештирүү керек болгон учурда класстар аздыр көптүр жарамдуу, бирок, өзү уюшулуучу (самоорганизующиеся системы) системаларды баяндаган учурда проблемалар пайда болот. Класстардан турган система ар бири башкаларга тикеден тике таасир эте ала тургандай камтылуучу механизмдерден (подмеханизмы) турган механизмге окшош. Бирок, ички уюмга ээ болгон ар кандай объект анык бир өз алдынчалуулукка ээ болуп өзүнүн функцияларын сырттан чакырууларга мүмкүнчүлүк бербейт – ал маалыматты ала алат, андан кийин аны өзү ойлогондой иштеп чыгат, жана эгер кааласа кимдир-бирүүгө жооп кайтара алат. Аны бир нерсени берип кой деп талап кылууга өзгөчө сырттан аны толук жетектөгө болбойт. Программисттердин тилинде муну мындайча айтса болот: *жандыкта ачык талаалар жана усулдар таптакыр жок*. Анан дагы, организм өзүнүн курамдык бөлүктөрүнө кайрылууга толук мүмкүнчүлүк ала албайт: мен буттарымды жана башымды кыймылдата алам, мурдум менен дем ала алам, өзүмдү кандай сезип жаткандыгымды жана андагы өзгөрүүлөрдү аны бир

деңгээлде сезип биле алам, бирок клеткалардагы биохимиялык процесстерге таасир этүү жөндөмдүүлүгүм жок. *Жандык өзүнүн ички абалына таасир эте алат, бирок ар-бир ички процесстин үстүнөн тикеден-тике контрол кыла албайт*.

Объекттик-ориентирленген программалоо өзү уюшуучу системаларды тандоо үчүн эч кандай каражаттарды бербейт. Бул багыттагы жалгыз жана анчалык чоң эмес кадам – бул касиеттерди кийирүү: мен объекттен бир нерсени ала жана бере алам, бирок бул учурда буга жооп (реакция) кыла турган анын ички механизмдери ишке салынат.

Тилдик деңгээлдеги объекттер жана аткаруу убактысынын объекттери

Программист качан программа жазып жатканда ал тилдик деңгээлдеги объекттерди пайдаланат. ООП тун принциптерин аркалаган программист программалык кодду аны бир катар класстардын баяндоосуна ажыратуу менен мүмкүн болушунча жакшы структуралоого аракет кылыш керек. Бирок, императивдик тилдер (ошонун ичинде алардын ОО-кеңейтирилиштери дагы) операторлордун жөн гана удаалаштыгы болуп эсептелген негизги (башкы) программа түшүнүгүн сактап калган. Албетте, эми анда глобалдык объеттердин усулдары чакырылат, бирок негизги программа баары бир башкы классты эмес – объекттер менен иш жүргүзгөн негизги камтылуучу программаны гана (подпрограмманы) билдирет. Андан дагы, программист операциялык системанын эч кандай объекттерге таандык болбогон функцияларын пайдаланышы керек. Delphi тилиндеги операциялык система менен өз ара аракет этүү милдетин алган Application классынын жана дагы кошумча тиркелген класстардын бүтүндөй библиотекасынын болушу силерди алардын мазмунуна карата чаташтырууга алып барбашы керек, алардын бардыгы операциялык системанын функцияларын пайдаланышы керек.

Биз программалык кодду талкууладык, эми аткарылып жаткан убактагы программаны карайлы. Программа компиляцияланып, андан кийин аткарылуу үчүн ишке салынары менен ал операциялык система тарабынан класстардын жыйындысы катары эмес, агымдардын тобунан турган өзүнүн адрестик мейкиндиги, стек сегменти ж.у.с. бар процесс катары түшүнүлөт (Windows да агымдар дагы волокнолорго бөлүнөт). Кыскасы, программанын жашоо циклине тикеден-тике тиешеси болгон бир катар түшүнүктөр пайда болот.

Аткаруу убактысынын объекттери (АУО) – программанын ишин баяндоочу түшүнүк.

Айрым учур катары, процесс, агым, волокно – бул АУО.

Программанын операциялык система менен өз ара аракеттениши

Операциялык системанын аныктамасын мындайча берсек болот:

ОС – бул аппараттык камсыздандыруу менен колдонмо (прикладдык) программалардын арасындагы орто катмар болуп эсептелген жана колдонмо программаларга аппараттык жабдылыш менен иштөөгө жана бири-бири менен өз ара аракет этүүгө мүмкүнчүлүк берүүчү функцияларды берген программа.

Башкача айтканда ОС – бул кадимки колдонмо программага караганда алда канча чоң нерсе.

“Жаңы техногияларды” колдоо үчүн гана кызмат кылуучу билдирүүлөр механизми, аткаруу убактысынын көптөгөн объекттери, кошумча түшүнүктөр тилди абдан көлөмдүү кылып жиберет. Мунун баары Simula жана Smalltalk заманынан бери өнүктүрүлбөй келе жаткан программалоо принциптери өзгөртүлүшү керек дегенди билдирет.

Программалардын ОС тен көбүрөөк көз каранды эмес кылуу маселеси көп коюлуп келет. Бул маселенин коюлушунун өзү ОС кадимки колдонмо программага караганда бир нерсеси менен айырмаланарын билдирет. Азыркы бар болгон программалоо концепцияларынын аймагында ОС деген эмне дегенге канааттандырарлык аныктама берилиши мүмкүн эмес – бул түшүнүк ар дайым сырткы түшүнүк болуп кала берген.

9.5. Биологиядан алынган бир нече терминдер

Объекттик-ориентирлеген программалоо адамдардын дүйнөнү таанып билүүгө болгон умтулуусунан улам пайда болгондуктан (Simula 67 тили) ООП тун көпчүлүк терминдеринин биологиядан келгендиги таң калыштуу эмес. Ал тургай функция түшүнүгү да математикалык мааниде эмес (процедуралык программалоодогудай) биологиялык мааниде түшүнүлө баштаган.

- Функция – бул жаныбар же өсүмдүк организминин, анын органдарынын, ткандарынын жана клеткаларынын спецификалык ишмердиги.
- Полиморфизм – 1. Кээ бир кристалдык заттардын шарттардан көз каранды түрдө заттын составынын өзгөрүүсүсүз, кристалдык функциясы жана башка физикалык касиеттери боюнча бир топ айырмаланган модификацияны түзө алуу жөндөмдүүлүгү (графит-алмаз).

2. Бир нече түрдөгү жаныбарлардын же өсүмдүктөрдүн аймагында бири-биринен дароо айырмаланган өкүлдөрдүн болушу: жыныстык, куракты, сезондук жана муундардын

кезектелиши менен байланышкан айырмачылыктарды айтууга болот, өсүмдүктөрдө болсо полиморфизм деп бир эле өзүндө анын кайсы бир органынын ар түрдүү формада болушу аталат.

Биотерминдерди колдонуу традициясын улантуу менен дагы бир нече терминдерди эске түшүрүп кетебиз:

- Генетикалык код – мурастык маалыматтардын кодондордун удаалаштыгы көрүнүшүндө жазылышы;
- Кодон – бул үч азоттук негизден (нуклеотиддер) турган генетикалык коддун бирдиги. Кодон белоктун синтезделүүчү молекуласына анык бир аминокислотанын кошулушун алдын ала камсыздайт;
- Фенотип – организмдин өздүк (индивидуадык) өнүгүү процессинде калыптануучу бардык белгилердин жана касиеттердин жыйындысы.

Программисттин көз карашында жандыктын (существо) генетикалык коду – бул программа, кодон – бул элементардык команда.

Бул түшүнүктөр бизге жандык (существо – Wesen) түшүнүгүн кийиргенибизде керек болот.

9.6. Везенспрограммалоо (Wesensprogramming)

Негизги түшүнүктөр

ООП тун негизги түшүнүгү болуп класс түшүнүгүг болгон эле. Инкапсуляция объекттердин эмес, класстардын деңгээлинде реализацияланган: А классынын объекти өзүнүн усулдарында А классынын башка каалагандай объектинин private – талааларынын маанилерин өзгөртө алат.

Объект-туунду түшүнүк: объект – класстын нускасы Везенспрограммалоонун (ВП) базалык түшүнүгү болуп жандык (существо – Wesens) түшүнүгүг эсептелет. Жандык объекттин аналогу. Анын баяндоосун (класстын аналогу) – Metawesens деп атайбыз.

Wesen үч бөлүктөн турат:

1. Hirn (мээ)
2. Inwelt (ички дүйнө)
3. Zutritte (кирүүлөр)

Ар бир жандык өзүнүн өздүк (жеке) агымында жашайт.

Metawesen мындайча баяндалат:

```
type
   MW=Metawesen
      Hirn: THirn
      Invelt:
      . . . . . .
    end;
```

- Hirn – бул объект, башкача айтканда жандыктагы башкаруу борбору;
- Invelt – бул жандыктын ички дүйнөсү. Жандыктын ички дүйнөсү объекттерден да, ошондой эле башка жандыктардан да турушу мүмкүн;
- Zutritte (кирүү) – жандыктагы башка жандыктардан маалыматтарды кабыл алуучу жалгыз каражат. Объекттер жандыктарга эч нерсе жибере албайт, алар – болгону берилгендердин булагы (ООПтогудай).

Башка жандыктарга маалыматтарды жиберүү үчүн чакыруу (Ruf) түшүнүгү киргизилет. Маалыматтарды кабыл алыш үчүн жандык кирүүлөрдү даярдашы керек (“толкунга тескөө - настроить на волну”). Эгер жандык тыштан маалыматтарды алгысы келбесе, анын кирүүлөрү блоктолот (жабылып салынат).

Жандык алуучу маалымат Hirn дин ичинде берилгендердин кандайдыр бир структурасында сакталып турушу мүмкүн.

Ар бир жандыкта Inwelt бар болуп, анда башка жандыктар жашай алат, өзүнүн ичинде жашап турган жандыктарга жандык чакыруулардын жардамында гана кайрыла алат. Экинчи жактан, ар бир жандык дагы кандайдыр бир башка жандыктын Inwelt инде жашап турат.

Жандык алган маалымат Hirn дин ичинде кандайдыр бир берилгендер структурасында сактала алат. Зарылчылыкка жараша Wesen алынган маалыматты анализдей алат.

- Эгерде А жандыгы В жандыгынын Inwelt инде турган болсо, анда В жандыгы А жандыгынын үстөк жандыгы (надсущество - overwesen) деп аталат. Үстөк жандыктын Inwelt и анын ичинде турган жандыктар үчүн Umwelt (курчап турган дүйнө) деп аталат. Эгерде ар бир жандык кандайдыр бир үстөк жандыктын ичинде жашап турат деп эсептесек, анда үстөк жандыктардын удаалаштыгы алынмак. Мындай болушу мүмкүн эмес, ошондуктан эч бир үстөк жандыктын Inwelt ине кирбеген башкы (негизги) жандык (Houptwesen, HW) бар. Бул башкы жандык – силер байкап тургандай, илгертен ОС деп аталып жүргөн нерсенин өзү.
- Байкай кетчү нерсе, башкы жандык түшүнүгү везенспрограммалоо үчүн сырткы түшүнүк эмес жана ошону менен ал эски ОС түшүнүгүнө толук тиешелеш келет. Башкы жандыктын Inwelt инде

жашаган бардык жандыктар апаратурага кайрылууга уруксатты түздөн-түз ала албайт, демек, башкы жандык апаратураны жашырып турат жана ага кайрылуу мүмкүнчүлүгүн анын функцияларын пайдалануу менен берет.

- Башкы жандыктын Inwelt инде жашаган жандыктар бири-бири менен алакалаша алышат (башкы жандык аркылуу).
- Ар бир жандык өзүнүн агымында (свой поток) жашагандыктан, демек башкы жандык алардын ортосунда ресурстарды бөлүштүрүп берип турат.

Ошентип, башкы жандыктын жашашы (бар болушу) кай жактан алынгандыгы түшүнүксүз болгон кандайдыр бир сырткы түшүнүк эмес, везенспрограммалоо концепциясынын зарыл натыйжасы болуп эсептелет.

Жандыктын өмүрү

Демек, ар бир жандык аны курчап турган мейикндикте (Umwelt) жашап турат. Андан сырткары, ар бир жандыктын ички мейкиндиги (Inwelt) бар.

Жандык өзүнүн ички дүйнөсүндө объекттердин каалаган бирөөсүнө айлана алат, ал эми андагы жашаган жандыктарга чакыруулардын жардамында айлана алат. Жандык өзүнүн камтылуучу жандыктарын (подсущество) түзө жана жогото алат.

Umwelt те жандыктын укуктары бир кыйла чектелген: ал башка жандыктарга аларга болгон чакыруулардын жардамында кайрыла алат. Ал эми Umwelt тин объекеттерине кайрылууну өзүнүн Inwelt инде «өлүк материяны» жетектөөчү үстөк жашоого болгон чакыруу аркылуу жүргүзүүгө болот.

Сырткы объекттерге болгон кайрылуулардын кандайдыр бир деңгээлде көлөмүнүн чоңоюп кетиши силерди тынчсыздандырбашы керек. Биринчиден жандыктардын деңгээлинде бүтүн сандардын же кандайдыр бир жөнөкөй объекттердин учураш ыктымалдуулугу абдан аз. Жандык иштей терган объекттер, албетте, кандайдыр бир берилгендер базалары, файлдар ж.у.с. болот. Биз качан программаларда файлга кайрылганыбызда баари бир ОС тун функцияларын пайдаланабыз. Ошондуктан үстөк жашоого болгон кайрылуу – бул жөн гана пайдаланылып жүргөн механизмдердин жалпыланышы болуп эсептелет.

Ошол эле учрда ортомчу деген нерсени кийирүү жандыкты чөйрөдөн бир топ көз карандысыз кылууга мүмкүнчүлүк берет. Бул өтө маанилүү моментти кийинчерээк талкуулайбыз, азыр болсо белгилеп коё турган нерсе, биз жандыктар жөнүндө кандайдыр бир тирүү нерсе катары сөз кылып жатабыз, анткени алар өзүнүн агымында жана өзүнүнүн тиричилиги менен жашайт.

Эми биз эмнеге жетишкендигибизди жыйынтыктайлы

- өзгөчөлөнгөндүккө (өзүнүн агымында аткаруу);
- өз алдынчалуулукка (башка жандыктардан жана сырткы объекттерден көз каранды болбостук);
- толук инкапсуляцияга (жандыкка чакыруулардын жардамында гана таасир этүүгө болот, болгондо да эгер ал өзү ошону кааласа).

Эми ВП программистти кантип кезексиз терминдерден куткарарын карап көрөлү.

Керексиз техникалык терминдерди алып салабыз.

Программист качан программалоонун традициялык методологиясына ылайык программа жазганда (б.а. ВП ны пайдаланбай), ал жаратылышта түздөн түз аналогу болбогон көптөгөн түшүнүктөрдү пайдаланат: процесс, агым, программа, операциялык система деген сыяктуу.

“Процесс” жана “агым” сөздөрүнүн маанилери программалоодо жана турмуштук тилде айырмаланары даана көрүнүп турат.

ВП да абал башкача: жандык (wesen) курчап турган чөйрө менен өз ара маалымат алмашат жана ал маалыматты ушул жандыктын мээси (Hirn) иштеп чыгат. Бардык ушул түшүнүктөр ВП дагы, реалдуу турмуштагы түшүнүктөр менен толук тиешелеш келет, ошондуктан чоюп алып келген техникалык терминдерге караганда алда канча ачык айкын. Ал тургай программа түшүнүгү эми мындайча туюнтулушу мүмкүн:

- программа – бул жандыктын генетикалык коду;
- генетикалык код – бул болсо аткарылуучу ажырагыс командалардын удаалаштыгы.

Тирүү жандыктар үчүн код клеткада аткарылат, ал эми информациялык жандыктар үчүн – процессордо аткарылат.

Жандык Metawesen дескрипторуна шилтеме жасайт, ал дескриптор ушул Metawesen дин бардык жандыктарында бар болгон усулдарга шилтеме жасайт. Metawesen ди баяндоочу программалык код – бул жандыктын генетикалык кодун баяндоо.

Жандыктын «генетикалык кодунун» өзү – бул Metawesen дин дескриптору жана мээнин Hirn баяндоосу менен бирге жандыктын өзү үчүн ажыратылган командалардын удаалаштыгы болот. Байкай кетчү нерсе, ал тургай программа-командалардын удаалаштыгы түшүнүгү ВП да «жандыктын генетикалык коду» менен алмашат.

Жандыктын фенотиби жандык иш учурунда пайдалануучу (кошумча эс) генотип+өзгөрүлмөлөр болот.

Бул учурда процесс – бул ОС те жашаган жандыктын фенотиби болуп эсептелет.

Кээ бир эски жана жаңы түшүнүктөрдү салыштыра турган төмөндөгүдөй таблицаны келтирели.

	Эски түшүнүктөр	Жаңы түшүнүктөр
1.	программа	жандыктын генотиби
2.	процесс	жандыктын фенотиби
3.	агым (поток)	пайдаланылбайт
4.	операциялык система (ОС)	Hauptwesen (HW)
5.	программа жазуу	жандыкты баяндоо
6.	программаны ишке салуу	жандыкты чөйрөгө кошуу
7.	билдирүүлөр	чакыруулар (вызовы)

Агым түшүнүгү айкын түрдө дегеле пайдаланылбайт, анткени агым жандык менен ажырагыс байланышта. Албетте, кээде жандыкка кыларга иш жок болуп калат, анда ага «чээнге крип кетүүгө» мүмкүнчүлүк берүү керек. Андан чыгуну ар түрдүүчө реализациялоого болот: Мисалы, жетишээрлик сандагы кийрилүүчү берилгендердин саны топтолоору менен ал ойгонушу жана кийирилүүчү маалыматтарды иштеп чыгышы керек.

Окурмандардын айрымдары везенспрограммалоону кабыл алуу менен биз программисти эски түшүнүктөрдөн куткарганыбыз менен алардын ордуна жаңыларын кийирет экенбиз да деп каршы чыгышы мүмкүн. Бирок, биз иш жүзүндө жаңы терминдерди кийрибейбиз, болгону илгертен белгилүү болгон биологиялык түшүнүктөрдү маалыматтар дүйнөсүнө карай кеңейтебиз, ал болсо, албетте, алда канча табийгийраак жана тууралаак болот.

Жандыктардын ар түрдүү тилдерде баяндалышы

Компьютерди ток булагына туташтыруу менен автоматтык түрдө Hauptwesen - жандык жаралат. Анын Invelt и алгачкы учурда бош болушу мүмкүн. Иштөө учурунда ал (Hauptwesen) өзүнүн Invelt ине башка жандыктарды (өзүнүн демилгеси же пайдалануучунун буйругу боюнча) кошо алат. Бул жандыктардын ар бири жыйынтыгында өзүнүн Invelt ин ар түрдүү жандыктар менен толтура алат.

Эгерде программист жаңы жандыкты баяндагысы келсе (мурда жаңы программа жазгысы келсе деп айтар элек), анда Metawesen ди жазат. Аны компиляциялагандан кийин Metawesen ди баяндоочу код менен файл түзүлөт. Башка бир жандыктын Invelt ине жаңы жандыкты кошуу амалынын мааниси мына мында турат: жандыкты түзүүчү усул (аны Leven - өмүр (жизнь) деп атайбыз) ишке салынган

болушу керек, ал эми Metawesen коду жандыктын Invelt ине кошулат (поключается).

Жандыктар сырткы дүйнө менен (ички жандыктар менен да) чакыруулар каражаттары менен байланышкандыктан жандыктарды ар түрдүү тилдерде баяндоо мүмкүн болсун үчүн төмөндөгүлөр гана керек:

1. чакырууларды иштеп чыгуунун жалпы механизми;
2. жандыкты чөйрөгө кошуунун жалпы механизми;

Ушул шарттар аткарылган учурда жандыктарды каалагандай программалоо тилдеринде баяндоого болот.

Бул эки шарт тең программалардын аткарылышына карата традициялык ыкмалар үчүн да керек (алар бардык программалар ОС тун башкаруусу алдында аткарылуу абалында болуш керектигинен көрүнөт).

Жалпы көрүнүш

Ар бир компьютерде Hauptwesen – башкы жандык бар. Тармакка туташтырылган компьютерлер мейкиндикти (Inforaum) түзүп анда жандыктар жашайт жана ал жандыктар бири-бири менен маалымат алмаша алышат. Глобалдык тармак информациялык (маалыматтык) мейкиндикти берет, анда бири-бири менен жана алардын Invelt териндеги жандыктар менен маалымат алмаша алган жандыктар жашайт. Информациялык материя же тирүү же өлүк болот. Тирүүсү – жандыктар менен, ал эми өлүк материя объекттер менен көрсөтүлөт.

Эгерде программалоого ВП нын көз карашында карасак, анда “программалоо – бул программаларды жазуу” деген аныктаманы кайра кароо керек болот. Эми программисттин милдети – эмне кылуу керек экендигине карата процессорго көрсөтмө берүү эмес, Inforaum да болуп жаткан кубулуштарды баяндоо.

Көрүнүп тургандай везенспрограммалоо концепциясы жогоруда каралган үч *проблеманы* чечет.

9.7. Биз жасабаган дагы эмне калды

Биз азыркы программалоо тилдеринин дагы бир жетишпестигин караган жокпуз, ал – мурастоонун чектелгендиги. Мурастоо бар болгон (жашап турган) класстардын негизинде жаңы класстарды түзүүгө мүмкүнчүлүк берет. Бирок ал үчүн бир гана жол сунуш кылынат: бир нерсени кошумча жазуу же бир нерсени өзгөртүү (башкача айтканда усулду кайра атоо). Ушундан улам айрым

авторлор [11] мурастоону өсүү (об ростание) деп атоо керек эле деп эсептешет.

Андан сырткары, жалгыздап мурастоо ар дайым эле жетиштүү боло бербейт, ал эми көптүктүн мурастоону реализациялоодо проблемалар пайда болушу мүмкүн. Албетте класска бир нерселерди кошуу же анан кандайдыр бир бөлүктөрүн өзгөртүү менен эле чектелбеген, классты каалагандай кылып кайра жасоого мүмкүнчүлүк берүүчү, болгондо да мындагы иштин көпчүлүгүн программист эмес компьютер өзү жасай тургандай универсалдык механизм болгондо сонун болмок. Азырынча мындай механизм жок. Демек, чечиш керек болгон маселе али бар.

1 – ТИРКЕМЕ. КАТАЛАР

Жаңылбас жаак,
мүдүрүлбөс туяк болбойт.
(Эл макалы)

Биз баарыбыз өзүбүздүн турмуштук тажрыйбабыздан көрүп жүргөндөй каталар кадам сайын кезигет. Алар ар төрдүү болот: критикалык каталар болот - ситуцияны оңдош үчүн токтоосуз бир нерселерди жасоо зарылдыгын талап кылган каталар, эскертүү иретиндеги каталар болот - ага жооп кылып жоюп койсоң азаматсың, жооп кылбый койсоң - учуру келгенче эч кандай чара колдонбой койсо болчудай, а балким билинбей өтүп кетээр …

Программаларды талдоодо деле так мына ушундай – себеби программа деле адам ишмердигинин бир жыйынтыгы. Программаларда каталар, кадимки турмуштагыдай эле, дароо болбосо да, ар кандай эле пайдалануучу үчүн болбосо да жетишээрлик жакшы көрүнөт. Пайдалануучулар каталарды байкаар замат же программаны талдап-иштеп чыккандарды шыбай башташат, же аны жакшыраак башка программа менен алмаштырууну көздөшөт, же эгер убактысы жана каалоо болгон болсо ушундай калтыс жерлерди айланып өтүүгө аракет жасашат – канткен менен иштеш керек да.

Ошон үчүн программаны жазууга чейин жана жазып жатканда бул программа эмне кылышы керек, каталар кетирилип калчу кандай ситуациялар болушу мүмкүн экендигин терең ойлонуштуруп койгон өтө чоң мааниге ээ. А бирок реалдуу турмушта тескерсинче - оболу программаны жазып анан кийин ойлонобуз.

Профессионалдар эмне деп ойлошот, окуп көрөлү. **Чоң программалык система көп жылдык тестирлөөдөн жана пайдалануудан кийин деле эч качан аягына чейин оңдолуп-түзөлүп бүтпөйт**. *Аягына чейин кичинекей программалар гана оңдолуп-түзөлүшү мүмкүн, андан ары чоңураак жана татаалыраак кылганга болбойт. Программанын жалпы алгоритми үчүн өтө эле көп жолдорду караштырса болот, киргизилүүчү берилгендердин же пайдалануучулардын аракеттеринин варианттары да өтө эле көп. Жүздөгөн жылдарга созулган тестирлөө да, - эгер, албетте ушундай кылуу мүмкүн болсо - чоң, татаал программанын мүмкүн болгон бардык бутактарын текшерип чыгууга жетмек эмес. А бирок ага карабастан, «каталардан арылган программалык камсыздандыруу» жөнүндө асыресе сөз кылган адамдар табылат*, - Джозеф Фокс, Демек мындан, эмне үчүн Microsoft Windows, Microsoft Visual Studio сыяктуу чоң программалык системалар ар убак жаңыланып чыгып тургандыгы түшүнүктүү. Бирок прогресс ордунда турган жок:

программистерге жардамга алардын өздөрү тарабынан шаблондор (Templates), программаларды долбоорлоо жана талдоонун визуалдык каржаттары, программалоонун жаңы технологиялары иштелип чыгылып жатат.

Кантсе да негизги каталарды алдын ала көрө билсе болот, ал үчүн алар кандай болоороруп билүү жана али программанын жазылыш стадиясында андай каталарга жол койбоо керек. Жакшысы, даро эле катасыз, жакшы программаларды жазууга көнүү керек – бул жалпыбыз үчүн акырында жеңил болот (арзанга түшөт).

Программалардагы каталардын категориялары

Белгилүү программисттер Крис Х. Паппас жана Уильям Х. Мюррей тарабынан сунуш кылынган каталардын классификациясын пайдаланабыз.

- **Синтаксистик каталар.** Мындай каталар компиляция этабында программисттин программалоо тилинин кандайдыр бир конструкциясын билбегендигинен же жөн эле кош көңүлдүгүнөн улам пайда болот. Мындай каталар менен компиляторлор өздөрү жакшы иш алып бара алат. Ошон үчүн мындай каталар бир аз дүүлүгүүнү пйда кылганы менен андай эле коркунучтуу деп эсептелинбейт. Андан сырткары компиляторлор *эскертүүлөрдү* чыгарып бере алат, аларды көптөгөн тажрыйбалуу программисттер логикалык каталардын потенциалдуу коркунучу катары кароого ыкташат. Өтө өркүндөтүлгөн компиляторлор болот, алар өздөрүнүн эскертүүлөрүндө өтө сейрек адашат (мисалы, Microsoft Visual C++ 6.0, Borland C++ 5.02).
- **Компоновщиктин каталары.** Мындай каталар көбүнчө ар түрдүү модулдарды туташтыруу (стыковка) максатында программист тарабынан интерфейстин тура эмес уюштурулганынан улам пайда болот. А эгерде система тилдердин бирөөсүндө модулдар менен иштеп жаткан болсо мындай каталардын себептеринин эң ыктымалдуусу – библиотекалардын жана башка кошулуучу файлдардын директорияларынын туура эмес көрсөтүлүшү болот.
- **Аткаруу убактысынын каталары.** Мындай каталар программанын мөөнөтүнөн мурда авариялык токтошуна же циклденип калышына алып барат:
  - *аппараттык аныкталуучу каталар:* «Нөлгө бөлүү», «Нөлдүн же терс сандын логарифмин эсептөөгө болгон аракет», эстин корголушунун бузулушу, түзүлүштөрдүн каталары ж.б.
  - *системалык каталар:* туура болбой калган (неудачная) файлдык амал, билдирүүлөр кезегинин толуп ашып кетиши.

- *нагыз программалык каталар:* индекстин массивдин чегинен чыгып кетиши, бош кезектен элементти жоготуу (алып салуу), идея боюнча эч качан пайда болбошу керек болгон чечимдин бутагына чыгуу ж.б.
- *тиркемелер үчүн спецификалык болуп эсептелген каталар:* мисалы, чыгаруунун тура эмес форматы.

♦ **Логикалык каталар**. Бул каталардын эң татаал түрү, анткени алар дароо эле байкала койбойт, аларды алдын ала көрө билүү түздөн-түз программисттин квалификациясынан, маданиятынан жана ошондой эле буйрутмачы тарабынан маселенин тура коюлушунан көз каранды. Так мына ушундай каталарды издеп табуу жана локалдаштыруу үчүн оңдоп-түзөө (отладка) процесси жашайт.

Төмөндө Turbo Pascal тилинин чөйрөсүндө алынышы мүмкүн болгон каталардын тизмегин алардын түшүндүрмөлөрү менен келтиребиз.

Каталар жөнүндө билдирүүлөр

Компилятордун билдирүүлөрү

1.	Out of memory – Эстин чегинен сырткары чыгып кетүү
2.	Identifier expected – Идентификатор көрсөтүлгөн эмес. Бул жерде идентификатор турушу керек.
3.	Unknown identifier – Белгисиз идентификатор. Идентификатор баяндалган эмес.
4.	Duplicate identifier – Кайталанган идентификатор. Бул идентификатор учурдагы блокто эбак баяндалган программанын, костантанын, модулдун, өзгөрүлмөнүн, типтин, процедуранын же функциянын атын көрсөтөт.
5.	Syntax error – Синтаксистик ката. Баштапкы текстте тура эмес символ табылды.
6.	Error in real constant – Чыныгы константада ката бар.
7.	Error in integer constant – Бүтүн константада ката бар.
8.	String constant exceeds line – Жолчолук константа жолчонун өлчөмүнөн ашып кетти.
9.	Too many nested files – Камтылма файлдар өтө эле көбөйүп кетти.
10.	Unexpected end of file – Файлдын тура эмес бүтүшү.
11.	Line to long – Жолчо өтө эле узун. Жолчонун узундугу 126 символдон ашып кетти.
12.	Type identifier expected – Типтин идентификатору керек. Бул жерде идентификатордун тиби турушу керек.
13.	Too many open files – Ачылган файлдар өтө көп.

14.	Invalid file name – Файлдын аты туура эмес. Файлдын аты туура эмес же жок эле жолду көрсөтүп турат.
15.	File not found – Файл табылган жок.
16.	Disk full – Диск толуп калды.
17.	Invalid compiler directive – Компилятордун туура эмес директивасы.
18.	Too many files – Файлдардын саны өтө эле көп. Программанын же программалык модулдун компиляциясында катышкан файлдардын саны өтө эле көп.
19.	Undefined type in pointer definition – Көрсөткүчтү аныктоодо аныкталбаган тип бар. Баяндалбаган көрсөткүчтүк типке шилтеме (ссылка) иштетилген.
20.	Variable identifier expected – Өзгөрүлмөнүн идентификатору керек.
21.	Error in type – Типти аныктоодогу ката. Типтин аныктамасы мындай символ менен баштала албайт.
22.	Structure too large – Өтө эле чоң структура. Структуралык тип үчүн максималдык уруксат берилген өлчөм 65535 байт.
23.	Set base type out of range – Көптүктүн базалык тиби үчүн чек-ара бузулган. Көптүктүн базалык тиби – 0 дөн 255 ке чейин болгон кесинди жа маанилеринин саны 256 дан ашпаган саналуучу тип болушу керек.
24.	File components may not be files or objects – Файлдын компоненттери файлдар же объекттер боло албайт.
25.	Invalid string Length – Жолчонун узундугу тура эмес. Жолчонун баяндалуучу максималдык узундугу 1 ден 255 ке чейинки аралыкта болушу керек.
26.	Type mismatch –Типтердин тиешелеш келбестиги.
	Invalid subrange base type – Тип кесиндисинин туура эмес базалык тиби. Бардык иреттик типтер уруксат берилген базанын типке ээ болушу керек.
28.	Lower bound greater than upper bound – Төмөнкү чек жогорку чектен чоң.
29.	Ordinal type expected – Иреттик тип керек.Чыныгы, жолчолук, структуралык жана иреттик типтерге, бул учурда уруксат берилбейт.
30.	Integer constant expected – Бүтүн копетанта керек
31.	Constant expected – Константа керек.
32.	Integer or real constant expected – Бүтүн же чыныгы константа керек.
33.	Pointer type identifier expected – Көрсөткүчтүк типтеги индентификатор керек. Бул индентификатор көрсөткүчтүк типти бере албайт.

34.	Invalid function result type – Функиянын жыйынтыгынын тиби туура эмес. Функциянын жыйынтыгынын туура тиби болуп бардык жөнөкөй, жолчолук жана шилтемелик жолчолук типтер эсептелет.
35.	Label indentifier expected – Эн-белгинин (метканын) идентификатору керек. Эн-белги идентификатор менен белгиленген эмес.
36.	BEGIN expected – BEGIN болушу керек.
37.	END expected – END болушу керек.
38.	Integer expression expected – Integer тибиндеги туюнтма керек.
39	Ordinal expression expected – Иреттик типтеги туюнтма керек. Туюнтма иреттик типке ээ болушу керек.
40.	Boolean expression expected – Boolean тибиндеги туюнтма керек. Туюнтма Boolean тибине ээ болушу керек.
41.	Operand types do not match operator – Операндардын типтери операторго тура келбейт.
42.	Error in expression – Туюнтмада ката бар.
43.	Illegal assignment – Ыйгаруу тура эмес. Типтештирилбеген файлдарга жана өзгөрүлмөлөргө маанилерди ыйгарууга болбойт.
44.	Filed identifier expected – Талаа идентификатору керек. Бул идентификатор мурда келген (предшествующей) жазуу тибиндеги өзгөрүлмө үчүн талааны бере албайт.
45.	Object file too large – Объекттик файл өтө элө чоң. Turbo Pascal өлчөмү 64К дан ашкан .OBJ – файлды компановка кыла албайт.
46.	Undefined external – Тышкы процедура аныкталган эмес. Объекттик файлда тышкы процедура же функция тиешелүү Public аныктамасына ээ эмес.
47.	Invalid object file record – Объекттик файл тура эмес жазылган. .OBJ – файл тура эмес объекттик жазууну кармап турат.
48.	Code segment too large – Код (командалар) сегменти өтө эле чоң. Программанын же программалык модулдун максималдык өлчөмү 65520 байтка барабар.
49.	Data segment too large – Берилгендер сегменти өтө эле чоң. Программада берилгендер сегментинин максималдык өлчөмү пайдаланылуучу программалык модулдарда баяндалган берилгендер менен кошо эсептегенде 65520 байтка барабар.
50.	Do expected – Do кызматчы сөзү керек.
51.	Invalid PUBLIC definition – PUBLIC туура эмес аныкталган.
52.	Invalid EXTPN definition – EXTPN туура эмес аныктылган.
53.	Too many EXTPN definition – EXTPN аныктамалары өтө көп. Tyrbo Pascal 256дан көп EXTPN аныктамалары болгон учурда OBJ-файлды иштете албайт.
54	OF expected – OF талап кылынат.
55.	INTERFACE expected –Интерфейстик секция талап кылынат.

56.	Invalid relocatable reference – уруксат берилбеген аралашма шилтеме.
57	THEN expected – THEN талап кылынат. THEN кызматчы сөзү жок болуп жатат.
58.	TO or DOWNTO expected – TO же DOWNTO талап кылынат. TO же DOWNTO кызматчы сөзү жок болуп жатат.
59.	Undefined forward – Мурда келүүчү (опережающее) баяндама аныкталган эмес.
61.	Invalid typecast – Типтин өзгөртүп түзүлүшү туура эмес.
62	Division by zero – Нөлгө бөлүү болуп жатат.
63.	Invalid file type – Туура эмес файлдык тип. Бул файлдык тип файлдарды иштетүү процедурасы тарабынан тейленбейт.
64.	Cannot Read or Write variables of this type – Мындай типтеги өзгөрүлмөлөрдү окууга же жазууга болбойт.
65.	Pointer variable expected – Өзгөрүлмө-көрсөткүчтү пайдалануу керек.
66.	String variable expected – Жолчолук өзгөрүлмө керек.
67.	String expression expected – Жолчолук типтеги туюнтма керек.
68.	Circular unit reference – Модулга болгон циклдик шилтеме бар. Interface секциясында эки модул бири-бирине шилтеме жасай албайт.
69.	Unit neme mismatch – Программалык модулдардын аттарынын тиешелеш келбестигин .TPU, .TPW же .TPP файлында табылган программалык модулдун аты uses операторунда көрсөтүлгөн атка туура келет.
70.	Unit version mismatch – Программалык модулдардын версияларынын туура келбестиги. Бул программада пайдалануучу бир же бир нече программалык модулдар алардын компиляциясынан иштеп өзгөргөн.
71.	Internal stack overflow – Ички стектин толуп-ашышы. Операторлордун камтылуучулугунун өтө эле чоң деңгээлинен улам компилятордун ички стеги толук иштетилип бүткөн.
72.	Unit file format error – Программалык модулдун файлынын форматы ката. .TPU, .TPW же .TPP файлы (платформадан көз каранды түрдө) жараксыз.
73.	Implementation expected – Ишке ашыруу (реализация) секциясы керек. Implementation кызматчы сөзү жок.
74.	Constant and case types do not match – Константалардын типтери жана case операторунун туюнтмасынын тиби бири-бирине тура келбейт.
75.	Record variable expected – Жазуу тибиндеги өзгөрүлмө керек.
76.	Constant out of range – Константа чек араны бузуп жатат.
77.	File variable expected – Файлдык өзгөрүлмө керек.

78.	Pointer expression expected – Көрсөткүч тибиндеги туюнтма керек, туюнтма көрсөткүчтүк типке ээ болушу керек.
79.	Integer or real expression expected – Real же Integer тибиндеги туюнтма керек. Туюнтма Real же Integer тибине ээ болушу керек.
80.	Label not within current block – Эн-белги (метка) учурдагы блоктун ичинде жайгашкан эмес. Goto оператору учурдагы блоктон сырткары жаткан эн-белгиге шилтеме жасай албайт.
81.	Label already defined – Эн-белги эбак аныкталган. Бул эн-белги менен оператор белгиленген.
82.	Undefined label in processing statement part – Операторлордун иштетилип жаткан бөлүгүндө аныкталбаган эн-белги (метка) бар.
83.	Invalid @ argument – @ операторунун аргументи жараксыз. Жарактуу аргумент болуп өзгөрүлмөлөргө болгон шилтемелер жана процедуралардын же функциялардын идентификаторлору эсептелет.
84.	UNIT expected – UNIT кызматчы сөзү керек.
85.	«;» expected – «;» керек.
86.	«:» expected – «:» керек.
87.	«,» expected – «,» керек.
88.	«(» expected – «(» керек.
89.	«)» expected – «)» керек.
90.	«=» expected – «=» керек.
91.	«:=» expected – «:=» керек.
92.	«[» or «(.» expected – «[» же «(.» керек.
93.	«]» or «.)» expected – «]» же «.)» керек.
94.	«.» expected – «.» керек.
95.	«..» expected – «..» керек.
96.	Too nany variables – Өзгөрүлмөлөр өтө эле көп.
97.	Invalid FOR control variable – FOR операторунун башкаруучу өзгөрүлмөсү тура эмес. FOR опреаторунун башкаруучу өзгөрүлмөсү учурдагы камтылуучу программанын баяндоолор бөлүгүндө аныкталган саналуучу типтеги өзгөрүлмө болушу керек.
98.	Integer variable expected – Бүтүн типтеги өзгөрүлмө керек.
99.	Files are not allowed here – Бул жерде файлдарга уруксат жок. Типтештирилген константа файлдык типке ээ боло албайт.
100.	String length mismatch – Узундктун тиешелеш келбестиги. Жолчолук константанын узундугу символдук массивдин элементтеринин санына тура келбейт.
101.	Invalid ordering of fields – Талаалардын ирети тура эмес. Жазуу тибиндеги константада талаалар алардын баяндалыш тартибинде жазылышы керек.
102.	Sting constant expected – Жолчолук типтеги константада.
103.	Ordinal or real variable expected – Integer же Real тибиндеги өзгөрүлмө керек.

104.	Ordinal variable expected – Иреттик типтеги өзгөрүлмө керек.
105.	INLINE error – INLINE операторунда ката бар. < операторун өзгөрүлмөлөргө болгон аралашма (перемещаемый) шилтемелер менен айкалыштырууга уруксат жок.Мындай шилтемелер дайыма сөз өлчөмүнө ээ.
106.	Character expression expected – Мындан мурда келүүчү туюнтма символдук типке ээ болушу керек.
107.	Too many relocation items – Аралашма элементтер өтө эле көп (Реалдык режим үчүн гана) .EXE файылдын которулуш табилицасы бөлүгүнүн өлчөмү 64K тан ашып кеткен.
108.	Overflow in arithmetic operation – Арфиметикалык амалда толуп-ашуу (переполнение) бар, амалдын жыйынтыгы Longint(-2147483648 .. 2147483647) аралыгында жатпай калды.
109.	No enclosing FOR, WHILE or REPEAT statement – Иштетүүчү FOR, WHILE же REPEAT оператору жок. Break жана Continue стандарттык процедуралары for, while же repeat операторлорунан сырткары пайдаланыла албайт.
112.	CASE constant out of range – Case тин константасы диапазондон сырткары жатат.Бүтүн сандык Case операторлору үчүн константалар -32768 дан 32767ге чейинки аралыкта жатышы керек.
113.	Error in statemend – Операторду ката бар. Бул индентифи- катор операторду баштай албайт.
114.	Cannot call an interrupt procedure – Үзгүлтүк процедурасын чакырууга мүмкүнчүлүк жок. Үзгүлтүк процедурасын түздөн түз чакыруу мүмкүн эмес.
116.	Must be in 8087 mode tp compile this – Компиляция үчүн 8087 режими зарыл. Бул конуструкция {$N+} режими учурунда гана копиляциялана алат.
117.	Target address not found – Баруу адреси (Адрес назначения) табылган жок.
118.	Include files are not allowed here – Мындай ситуацияда кошулуучу файлдарга уруксат берилбейт. Ар бир операторлор бөлүгү бүтүн бойдон бир файлдын ичинде жайгашышы керек.
119.	No inherited methods are accessible here – Бул жерде мурасталуучу усулдарга уруксат жок. **I**nherited кызматчы сөзү усулдан сырткары же текке ээ болбогон усулда же объекттик типте пайдаланылып жатат.
121.	Invalid gualifier – Квалификатор туура эмес.
122.	Invalid variable reference – Өзгөрүлмөгө болгон уруксат берилбеген шилтеме. Мурда келген конструкция өзгөрүлмөгө болгон шилтеменин синтаксисин канаатандырат, бирок ал эстин адресин көрсөтпөй жатат.

123.	Too many symbols – Идентификаторлор өтө эле көп. Программа же программалык модул 64К дан ашык идентификаторлорду баяндап жатат.
124.	Statement part too large – Операторлор бөлөгү өтө эле чоң. Turbo Pascal операторлор бөлөгүнүн өлчөмүн болжолдуу түрдө 24К чоңдугуна чейин чектейт.
126.	Files must be var parameters – Файлдар var параметрлерине ээ болушу керек. Файлдык типтин параметрлери параметр-өзгөрүлмөлөр болушу керек.
127.	Too many conditional symbols – Шарттуу идентификаторлор өтө көп. Шарттуу идентификаторлорду аныктоо үчүн орун жок.
128.	Misplaced conditional directive – Шарттуу директива калып кеткен. Компилятор тиешелүү {$IFDEF}, {$IFNDEF} же {$IFORT} директиваларысыз {$ELSE} же {$ENDIF} директивасы бар экендигин тапты.
129.	ENDIF directive missing – ENDIF директивасы калып кеткен. Баштапкы файлда {$IFxxx} жана {$ENDIF} директивалар барабар санда болушу керек.
130.	Error in initial conditional defines – Баштапкы шарттуу аныктамаларда ката бар.
131.	Header does not match previous definition – Бөрк мурдагы аныктамага тиешелеш келбейт. Процедуранын же функциянын интерфейстик секцияда же forward баяндоосунда көрсөтүлгөн бөркү процедуранын же функциянын өзүнүн бөркүнө тиешелеш келбейт.
132.	Cannot evaluate this expression – Берилген туюнтманы эсептөөгө бобьойт. Туюнтма-константада же оңдоп-түзөө туюнмасында (отладочное выражение) колдонууга мүмкүн болбогон каражаттарды пайдаланууга болгон аракет бар.
134.	Expression incorrectly terminated – Туюнтманын корректтүү эмес (тура эмес) аякташы. [Тиркелген оңдоп-түзөгүч (отладчик) үчүн гана]. Turbo Pascal бул жерде туюнтманын бүтүшүн же амалды күтүп жатат, бирок алардын бирөөсүн да таппай жатат.
135.	Invalid format specifier – Форматтын тура эмес спецификатору. [Тиркелген оңдоп-түзөгүч (отладчик) үчүн гана]. Форматтын тура эмес спецификатору пайдаланылган же форматтын спецификаторунун сандык аргументи уруксат берилген чектен чыгып кеткен.
136.	Invalid indirect reference – Уруксат берилбеген кыйыр шилтеме. Оператор уруксат берилбеген кыйыр шилтемени ишке ашырууга аракет кылып жатат.
137.	Stuctured variable are not allowed here – Бул жерде структуралык өзгөрүлмөнү пайдаланууга уруксат жок.

138.	Cannot evaluate without System unit – System блогу жок эсептөөгө болбойт. [Тиркелген оңдоп-түзөгүч (отладчик) үчүн гана]. Оңдоп-түзөгүч туюнтманы эсептей алышы үчүн .TPL файлында System модулу кармалып турушу керек.
139.	Cannot access this sumbol – Бул идентификаторго кайрылууга уруксат жок. [Тиркелген оңдоп-түзөгүч (отладчик) үчүн гана]. Программа компиляцияланып бүтөөрү менен андагы бардык идентификаторлордун көптүгүнө кайрылууга мүмкүн болуп калат. Бирок, айрым бир идентификаторлорго программа ишке салынмайынча кайрылууга болбойт.
140.	Invalid floating-point operation – Уруксат берилбеген жылма чектиттүү амал. Эки чыныгы маанилердин үстүнөн болгон амалда ашып кетүү (переполнение) же нөлгө бөлүү болуп калды.
141.	Cannot compile overlay to memory – Оверлейлерди эске компиляциялоону аткарууга болбойт. [Реалдык режим]. Оверлейлерди пайдалануучу программа дискке компиляцияланышы керек.
142.	Procedure or function variable expected – Процедуралык же функционалдык өзгөрүлмө пайдаланылышы керек. Assigned стандарттык процедурасы өзгөрүлмө-көрсөткүч тибиндеги же процедуралык типтеги аргументти талап кылат.
143.	Invalid procedure or function reference – Процедурага же функцияга болгон уруксат берилбеген шилтеме.
144.	Cannot overlay this unit – Бул модул оверлейлик модул катары пайдаланыла албайт. [Реалдык режим]. {$O+} директивасы менен компиляция болбогон модулду оверлейлик модул катары пайдаланууга аракет болуп жатат.
145.	Too many nested scopes – Камтылуучулуктун деңгээли өтө чоң. Камтылуучулуктун деңгээлине uses сүйлөмүндөгү ар бир unit, камтылуучулукка ээ болгон ар бир жазуу, with операторлорунун камтылуучулугу таасир этет.
146.	File access denied – Файлга кайрылууга болбойт. Файлды ачууга же түзүүгө болбойт.
147.	Object type expected – Объекттик тип керек. Идентификатор об’екттик типти аныктабай жатат.
148.	Local object ture are not allows – Локалдык об’ екттик типтерди баяндоого уруксат жок.
149.	VIRTUAL expected – VIRTUAL талап кылынат. VIRTUAL кызматчы сөзү жок.
150.	Method identifier expected – Усулдун идентификатору керек. Идентификатор усулдун идентификатору эмес.
151.	Virtual constructor are not allowed – Виртуалдык конструкторго уруксат жок. Конструктордун усулу статикалык болушу керек.

152.	Constructor identifier expected – Конструктордун идентификатору керек. Идентификатор конструктордун идентификатору эмес.
153.	Destructor identifier expected – Деструктордун идентификатору керек. Идентификатор деструктордун идентификатору эмес.
154.	Fail only allowed within constructors – Fail ге конструкторлордун ичинде гана уруксат берилет. File стандарттык процедурасы конструктордун ичинде гана пайдаланыла алат.
155.	Invalid combination of opcode and operands - Амал коду менен операндардын уруксат берилбеген комбинациясы. Ассемблердеги амал коду операндардын мындай айкалышын кабыл албай жатат.
156.	Memory reference expected – Эске болгон шилтеме талап кылынат. Ассемблердин операнды, бул жерде талап кылынгандай, эске болгон шилтеме боло албайт.
157.	Cannot add or sbstract relocatable symbols - Аралашма (перемещаемые) идентификаторлорду кошууга же кемитүүгө болбойт. Ассемблердин операндында аралашма (перемещаемые) идентификаторлордун үстүнөн аткарууга уруксат берилген жалгыз амал - бул константаны кошуу же константаны кемитүү.
158.	Invalid register combination – Регистрлердин уруксат берилбеген айкалышы. Индекстик регистрлердин уруксат берилген айкалышы: [BX], [BP], [SI],[DI],[BX+SI],[BX+DI],[BP+SI] жана [BP+DI]
159.	286/ 287 Instructions not allowed - 286/ 287 процессорлорунун инструкцияларына уруксат жок. Бул көрсөтүлгөн процессорлордун амалдар коддоруна уруксат берүү үчүн компилятордун {$G+} директивасын пайдалануу керек, бирок жыйынтыктоочу код 8086 жана 8088 процессорлуу машиналарда иштей албайт.
160.	Invalid symbol reference – Идентификаторго болгон уруксат берилбеген шилтеме. Берилген идентификаторго ассемблердин операнды үчүн уруксат жок.
161.	Code generation error – Кодду генерациялоо катасы. Оператордун мурда келген бөлүгү керек болгон эн-белгиге (меткага) жете албай жаткан LOOPNE, LOOPE, LOOP же JCXZ инструкцияларын кармап турат.
162.	ASM expected – ASM кызматчы сөзү керек.
163.	Duplicate dynamic method index – Динамикалык усулдун индекси кайталанып жатат. Динамикалык усулдун бул индекси эбак башка усул тарабынан пайдаланылган.
164.	Duplicate resource identifier – Ресурс идентификаторунун кайталанышы. [Windows же корголгон режим үчүн гана]. Берилген файл башка ресурс үчүн эбак пайдаланылган атка же идентификаторго ээ болгон ресурсту кармап турат.

165.	Duplicate or invalid export index – Экспорттун кайталанган же уруксат берилбеген индекси. [Windows же корголгон режим үчүн гана]. Index операторунда берилген иреттик номер 1 .. 32767 диапазонунда жатпайт же ал номер башка экспорттолуучу камтылуучу программа тарабынан эбак пайдаланылган.
166.	Procedure or function indentifier expected – Процедуранын же функциянын идентификатору керек. [Windows же корголгон режим үчүн гана]. Export оператору процедураларды жана функцияларды гана экспорттоого урусат берет.
167.	Cannot export this symbol – Бул идентификаторду экспорттоого болбойт. [Windows же корголгон режим үчүн гана]. Процедура же функция, эгерде ал Export процедурасынын операторунда баяндалбаган болсо, анда экспорттоло албайт.
168.	Duplicate export name – Экспорттолуучу аттын кайталанышы. [Windows же корголгон режим үчүн гана]. Name операторунда берилген ат башка экспорттолуучу камтылуучу программа үчүн эбак пайдаланылган.
169.	Executable file header too large – Аткарылуучу файлдын бөркү өтө эле чоң. [Windows же корголгон режим үчүн гана]. EXE файлдын генерацияланууучу бөркү 64К (компоновщик үчүн жогорку предел) дан ашып кеткен.

Аткаруу убактысынын каталары

Аткаруу убактысынын каталары төмөндөгүдөй форматка ээ.

Runtime error nnn at xxxx:yyyy – xxxx: yyyy адреси боюнча аткаруу убактысынын **nnn** катасы.

DOS тун каталары

1.	Invalid function number – Уруксат берилбеген функциянын номери. DOS тун жок (жашабаган) функциясына кайрылуу.
2.	File not found – Файл табылган жок. Эгер файлдык өзгөрүлмөгө ыйгарылган ат жок (жашабаган) эле файлды көрсөтүп турса, анда Reset, Append, Rename же Erase процедуралары тарабынан ушул ката генерацияланат.
3.	Path not found – Маршрут табылган жок. Эгер файлдык өзгөрүлмөгө ыйгарылган ат жараксыз болсо (недействительный) же жок (жашабаган) камтылуучу каталогду (подкаталогду) көрсөтүп турса, анда Reset, Append, Rwrite же Erase процедуралары тарабынан ушул ката генерацияланат. Ошондой эле маршрут жараксыз болгон (недействительный) же жок (жашабаган) камтылуучу каталогду (подкаталог) көрсөтүп турган учурда ChDir, MkDir же RmDir процедуралары тарабынан ушул ката генерацияланат.

4.	Too many open files – Ачылган файлдардын саны өтө көп.
5.	File access denied – Файлга кайрылууга уруксат жок.
6.	Invalid file handle – Файлды баяндоодо мындай баяндагычка уруксат жок. Бул ката DOS тун системалык чакыруусуна файлдын урукста берилбеген баяндагычы берилгенде генерацияланат.
12.	Invalid file access code – Файлдарга кайрылуу коду жараксыз. Бул ката типтүү же типсиз файлдарды ачууда FileMode маани жараксыз болгон учурда Reset жана Append процедуралары тарабынан генерацияланат.
15.	Invalid drive number – Диск салгычтын (дисководдун) мындай номерине уруксат жок. Бул ката диск салгычтын номери тура эмес болгон учурда GetDir процедурасы тарабынан генерацияланат.
16.	Cannot remove current directory – Учурдагы каталогду жоготууга (өчүрүүгө) болбойт. Эгерде маршрут учурдагы каталогду көрсөтүп төрган болсо, анда RmDir процедурасы тарабынан ушул ката генерацияланат.
17.	Cannot rename across drives – Кайра атоодо ар түрдүү диск салгычтарды көрсөтүүгө болбойт. Эгерде эки файл тең бир эле дискте турбаса, анда Rename процедурасы тарабынан ушул ката чыгат.
18.	No more files – Файлдар жок. Мындай билдирүү качан FindFirst же FindNest чакыруулары берилген атка же атрибуттардын жыйынына туура келген файлдарды таппай калган учурда Dos модулундагы DosError өзгөрүлмөсү аркылуу чыгарылат.

Кийрүү-чыгаруу каталары

Эгерде {$I+} директивасы менен компиляцияланган операторлордун кайсы биринде кийрүү-чыгаруу катасы болуп жатса, анда ал программанын аткарылышынын токтошуна алып келет. Ал эми эгерде кийрүү-чыгаруу катасы бар оператор {$I-} директивасы кошулган (включенный) абалда компиляцияланса, анда программанын аткарылышы улана берет дагы, катанын номерин IOResult функциясынын жардамында алууга болот.

100.	Disk read error – Дискти окуу катасы.
101.	Disk write error – Дискке жазуу катасы. Эгерде диск толуп калган болсо, анда Close, Write, Writeln, же Page процедуралары тарабынан ушул ката генерацияланат.
102.	File not assigned – Файлга ат берилген эмес.
103.	File not open – Файл ачылган эмес. Эгерде файл ачылбаган болсо, анда Close, Read, Write, Seec, Eof, FilePos, FileSize, Flush, BlockRead же BlockWrite процедуралары тарабынан ушул ката генерацияланат.

104.	File not open for input – Кийрүү үчүн файл ачылган эмес. Эгерде файл кийрүү үчүн ачылбаган болсо, анда Read, Readln, Eof, Eoln, SeecEof же SeecEoln процедуралары тарабынан тексттик файлда ушул ката генерацияланат.
105.	File not open for output – Чыгаруу үчүн файл ачылган эмес. Эгерде файл чыгаруу үчүн ачылбаган болсо, анда Write, Writeln же Page процедуралары тарабынан тексттик файлда ушул ката генерацияланат.
106.	Invalid numeric format – Туура эмес сандык формат. Тексттик файлдан окулган сандык маани туура сандык форматка тиешелеш келбеген учурда Read же Readln процедуралары тарабынан ушул ката генерацияланат.

Критикалык каталар

Критикалык каталар реалдык же корголгон режимде пайда болушу мүмкүн.

150.	Disk is write protected – Диск жазуудан корголгон.
151.	Unknown unit – Белгисиз модул.
152.	Drive not ready – Диск салгыч «даяр эмес» абалда.
153.	Unknown command – Таанылбаган (неопознанная) команда.
154.	CRC error in data – Берилгендерде ката бар.
155.	Bad drive requiest structure length – Дискке кайрылууда структуранын туура эмес узундугу көрсөтүлгөн.
156.	Disk seek error – Дискте бөркчөлөрдү (головкаларды) коюу (установка) амалы учурундагы ката.
157.	Unknown media type – Ташып жүрүүчүнүн белгисиз тиби.
158.	Sector not found – Сектор табылган жок.
159.	Printer out of paper – Печатточу түзүлүштө кагаз түгөндү.
160.	Device write fault – Берилген түзүлүшкө жазуудагы ката.
161.	Device read fault – Берилген түзүлүштөн окуудагы ката.
162.	Hardware failure – Аппаратуранын таштап жиберүүсү (сбой). Биргелешип кайрылуу бузулган учурда же ар түрдүү тармактык каталар болгондо DOS ушул ката жөнүндөгү билдирүүнү чыгарат.

Фаталдык каталар

Фаталдык каталар программанын ишинин дароо токтошуна алып келет.

200.	Division by zero – Нөлгө бөлүү. Программада /, mod же div амалы учурунда санды нөлгө бөлүү аракети болуп жатат.
201.	Range chek error – Чек араларды текшерүүдөгү ката.
202.	Stac overflow error – Стектин ашып кетиши (переполнение).

203.	Heap overlow error – Эстин динамикалык бөлүштүрүлүүчү аймагынын ашып кетиши (переполнение).
204.	Invalid pointer operation – Жараксыз (недействительная) шилтеме амалы.
205.	Floating point overflow – Жылма чекиттүү (с плавающей точкой) амалды аткарууда ашып кетүү болду. Жылма чекиттүү амал ашып кетүүгө алып келди.
206.	Floating point underflow – Жылма чекиттүү амалды аткарууда иреттин (тартиптин) жоголуп кетиши. Жылма чекиттүү амал иреттин (тартиптин) жоголуп кетишине алып келди. Атайын көрсөтүлбөгөн учурда иреттин (тартиптин) жоголуп кетиши нөлгө барабар болгон жыйынтыкты кайтарып берет.
207.	Invalid floating point operation – Уруксат берилбеген жылма чекиттүү амал.
208.	Overlay manager not installed – Оверлейлерди башкаруучу камтылуучу система (подсистема) орнотулган эмес. [Реалдык режим үчүн гана].
209.	Overlay file read error – Оверлейлелик файлды окуу катасы. [Реалдык режим үчүн гана]. Оверлейлерди башкаруучу камтылуучу система оверлейлелик файлдан оверлейди окууга аракет кылганда окуу катасы пайда болду.
210.	Objec not initialized – Объект инициализацияланган эмес. Диапазонду текшерүү кошулганда объект конструкторду чакыруунун жардамында аныкталганга чейин обекттин виртуалдык усулуна кайрылуу болуп өттү.
211.	Call to abstract method – Абстракттык усулду чакыруу.
212.	Stream registration error – Агымды регистрациялоонун катасы.
213.	Collektion index out of range – Топтомдун индекси диапазондон сырткары. TCollection усулуна бериллүүчү индекс диапазондун чегинен чыгып кеткен.
214.	Colektion overlow error – Топтомдун ашып кеттүү (переполнение) катасы. Мындай ката TCollection тарабынан кеңейтүүгө мүмкүн болбогон топтомго элементти кошууга аракет жасалганда чыгат.
215.	Arithmetic overflow error – Арифметикалык ашып кеттүү (переполнение). Бул ката {$G+} абалында компиляцияланган операторлор тарабынан арифметикалык амал ашып кеттүүгө алып келген учурда чыгарылат.
216.	General Protection fault – Коргоонун жалпы бузулушу. [Корголгон режим үчүн гана].

Корголгон режимдеги (DPMI) DOS интефейсинин каталары

Биз бул жерде программа корголгон режимде аткарылып жаткан учурда алынышы мүмкүн болгон каталар жөнүндөгү билдирүүлөрдү келтиребиз. Мындай каталар 4 категорияга бөлүнөт: орнотуу каталары (ошибки установки), фиктивдик модулдун каталары (фиктивного модуля), аткаруу этабынын администраторунун каталары жана сервердин каталары.

Орнотуу каталары (DPMIINST)

A20 line already enabled, so test is meaningless – A20 тармагы эбак аракетке келтирилген, текшерүү мааниге ээ эмес.

Фиктивдик модул каталары.

Фиктивдик модул – эгерде жок болгон болсо DPMP – сервердик жана аткаруу этабынын администраторунун жүктөлүшбнө жооп берет. Фиктивдик модулдун каталары жөнүндөгү билдирүүлөр төмөндөгүдөй форматка ээ:

Stub error (xxxx): xxxx

мында Stub error ката фиктивдик модул тарабынан генерациялангандыгын көрсөтөт, (xxxx) – ката жөнүндөгү билидирүүнүн номери, ал эми xxxx – билидирүүнүн тексти.

Stub error (0001):	**needs at least 286** – Фиктивдик модулдун катасы: 286 дан кем болбогон процессор керек.
Stub error (2002):	**can`t find rtm.exe -** Фиктивдик модулдун катасы: rtm.exe файлы табылган жок. Аткаруу этабынын администраторунун файлы - rtm.exe маршрут боюнча же учурдагы каталогдо жатышы керек.
Stub error (2003):	**can`t find DPMI16BI.OVL -** Фиктивдик модулдун катасы: **DPMI16BI.OVL** файлы табылган жок. **DPMI16B1.OVL** файлы маршрут боюнча же учурдагы каталогдо жатышы керек.
Stub error (0012):	**file not found -** Фиктивдик модулдун катасы: файл табылган жок. Фиктивдик модул колдонмо программанын файлын таба алган жок.
Stub error (0013):	**path not found -** Фиктивдик модулдун катасы: маршрут табылган жок.
Stub error (0015):	**file access denied -** Фиктивдик модулдун катасы: кайрылууга уруксат берилбеген файл.

Stub error (0018):	**not enough memory to load file -** Фиктивдик модулдун катасы: файлды жүктөө үчүн эсте орун жетишпейт.
Stub error (001A):	**invalid environment -** Фиктивдик модулдун катасы: уруксат берилбеген операциялык чөйрө. DOS системанын чөйрөсүнүн спецификасы бузулган.
Stub error (001B):	**invalid file -** Фиктивдик модулдун катасы: Файлга кайрылууга уруксат жок.
Error:	**no DOS extensions in DPMI server –** Ката: **DPMI** серверде DOS тун кеңейтирилиштери жок. Пайдаланылуучу **DPMI -** сервер тармактык стандарттарды кармабайт.
Error:	**heeds DOS 3.x or higher -** Ката: DOS тун 3.х же андан жогорку версиясы талап кылынат.
Error:	**failed to locate DPMI-server(DPMI16BI.OVL) - DPMI –** серверди табууга мүмкүн болбой жатат. **DPMI16BI.OVL** (корголгон режимдеги колдонмо программа катары) көрсөтүлгөн маршрут боюнча жайгашкандыгына ишенүү керек.

Аткаруу этабы администраторунун каталары

Аткаруу этабы администраторунун каталарынын көпчүлүгү эстин жетишпегендигинен улам пайда болот.

Мындай типтеги каталар төмөндөгүдөй форматка ээ:

Loader error (xxxx): xxxx
(Жүктөөчүнүн катасы (xxxx): xxxx)

мында **Loader error** ката фиктивдик модулдун жүктөөчүсү тарабынан генерациялангандыгын көрсөтөт, **(xxxx)** - катанын номери, ал эми **xxxx –** билдирүүнүн тексти.

Loader error (0001):	**out of memory -** Жүктөөчүнүн катасы: эс жок.
Loader error (0002):	**out of selectors -** Жүктөөчүнүн катасы: селекторлор жок.
Loader error (0003):	**out of internal tables -** Жүктөөчүнүн катасы: ички таблицалар жетишпейт. Жүктөөчү ички таблицалар дын пределдик маанисин жогорулатып жиберди.
Loader error (0020):	**invalid dunamic link-** Жүктөөчүнүн катасы: уруксат берилбеген динамикалык компоновка. DLL ден импорттун уруксат берилбеген шилтемеси.

Loader error (0022):	**could't open file-** Жүктөөчүнүн катасы: файлды ачууга мүмкүн эмес. Файл же файл тарабынан импорттолуучу DLL табылган жок же ачууга болбой жатат.
Loader error (0023):	**invalid exe format -** Жүктөөчүнүн катасы: .exe – файлдын уруксат берилбеген форматы. Файл же файл тарабынан импорттолуучу DLL табылган жок же уруксат берилбеген форматка ээ.
Loader error (0024):	**wrong version -** Жүктөөчүнүн катасы: туура эмес версия. **DPMI16.OVL** файлы корректтүү маршрут боюнча жайгашкан жана жүктөөгө уруксат берилген биринчи файл болушу керек.
Loader error (0025):	**cannot initialize -** Жүктөөчүнүн катасы: инициализа циялоого мүмкүн эмес.
Loader error (0026):	**DLL initialization error -** Жүктөөчүнүн катасы: DLL ди инициализациялоо катасы. Инициализациялоо камтылуучу программаларынын бирөөсү тура эмес аяктады (башкача айтканда катанын нөл эмес кодун кайтарып берди).
Error:	**error the environment string –** Ката: операциялык чөйрөнүн жолчосундагы ката. "RTM" операциялык чөйрөсүнүн жолчосундагы корректтүү эмес параметрлер.
Runtime error:	**invalid entry point called –** Аткаруу этабынын катасы: кирүүнүн уруксат берилбеген чекитин чакыруу. Колдонмо программа жок эле атка же модулдардын бирөөсүнүн иреттик номерине шилтеме жасап жатат.
Application errors:	**Application load & execute error 0001; Application load & execute error FFF0 -** Колдонмо программанын жүктөлүш жана аткарылыш катасы. Корголгон режимдеги колдонмо программаны жүктөө үчүн эс жетишпейт.

DPMI - сервердин каталары

DPMI-сервердин каталары жөнүндөгү билдирүүлөр Borland DPMI-сервер тарабынан гана генерацияланып экранда төмөндөгүдөй формата пайда болот:

DPMI error (xxxx): xxxx

мында **DPMI error -** ката DPMI – сервер тарабынан генерацияланган-дыгын көрсөтөт, **(xxxx)** - катанын номери ал эми **xxxx –** билдирүүнүн тексти.

DPMI error (4001):	**insufficient memory for initialization –** DPMI нин катасы: инициализациялоо үчүн эс же тишпейт. Серверди ишке салуу (запуск) үчүн эс жетишпейт.
DPMI error (4002):	**memory manager does not support DPMI or VCPI–** DPMI нин катасы: эстин администратору DPMI ни же VCPI ни кармабай жатат (не поддерживает). DPMI же VCPI табылган жок.
DPMI error (4004):	**unrecognized hardware, run DPMINST -** DPMI нин катасы: тааныш эмес аппаратура, **DPMIINST** ти ишке салгыла. DPMI-сервер сиз пайдаланган аппаратураны тааныбай жатат.
DPMI error (4005):	**unrecognized environment parameters -** DPMI нин катасы: операциялык чөйрөнүн таанылбаган параметрлери. DPMIMEM операциялык чөйрөсүнүн өзгөрүлмөсүнүн параметрлери корректтүү эмес (тура эмес).
DPMI error (4007):	**bad A20 off parameter -** DPMI нин катасы: туура эмес A20 пармаетри. DPMI-сервер A20 параметри менен корректтүү (туура) иштей албайт. HIMEM.SYS эс администратору туура (корректно) орнотулган болушу керек.
DPMI error (4008):	**bad A20 on parameter -** DPMI нин катасы: туура эмес A20 пармаетри.
DPMI error (4009):	**bad switch parameter -** DPMI нин катасы: туура эмес пармаетр-кайра туташтыргыч.XMMDOS, XIMEM.SYS же QEMM драйверинин тура эмес иштеши.
DPMI error (4009):	**insufficient extended memory -** DPMI нин катасы: кеңейтирилген эс жетишпейт. Сервердин иши үчүн кеңейтирилген эс жетишпейт.
DPMI error (400C):	**cannot create linear address space -** DPMI нин катасы: сызыктуу адрестик мейкиндикти түзүү мүмкүн эмес.
DPMI error (400D):	**cannot create system address space -** DPMI нин катасы: системалык сызыктуу мейкиндикти түзүү мүмкүн эмес.
DPMI error (400E):	**cannot copy kernel to high memory -** DPMI нин катасы: ядрону эстин чоң адрестерине көчүрүү мүмкүн эмес.

DPMI error (400F):	**undefined error -** DPMI нин катасы: аныкталбаган ката.
DPMI error (4010):	**unable to copy DOSX –** DPMI нин катасы:**DOSX** ти көчүрүп алуу мүмкүн эмес.
DPMI error (4011):	**unable to copy IDT -** DPMI нин катасы: **IDT** тини көчүрүп алуу мүмкүн эмес.
DPMI error (4012):	**unable to create int chain table -** DPMI нин катасы: үзгүлтүктөр чынжырчасынын таблицасын түзүү мүмкүн эмес
DPMI error (4013):	**unable to create PM stack -** DPMI нин катасы: корголгон режим үчүн стекти түзүү мүмкүн эмес
DPMI error:	**Bad environment params –** Операциялык чөйрөнүн тура эмес параметрлери. Корректтүү синтаксисти пайдаланып DPMIMEM операциялык чөйрөсүнүн згөрүлмөсүнүн маанисин кайрадан бергиле.
DPMI error:	**Machine not in database (run DPMIINST) –** Машина берилгендер базасында жок болуп жатат, DPMIINST ти аткаргыла. DPMI-сервер ядронун берилгендер базасында издөөнү аткарып жатат, сиздин машинаңыз жөнүндө информацияны таппай жатат.
DPMI error :	**Not enough memory for PM init –** Корголгон режимди инициализациялоо үчүн эс жетишпейт. DPMI-сервер корголгон режимди инициализация-лашы үчүн эс жетишпей жатат.
DPMI error:	**V86 task without vcpi – VCPI** жок **V86** маселеси. DPMI - сервердин корголгон режимге өтүшүнө мүмкүнчүлүк бербей жаткан башка маселе аткарылып жатат.

2 – ТИРКЕМЕ. КОМПИЛЯТОРДУН ДИРЕКТИВАЛАРЫ

Turbo Pascal тилинин чөйрөсүндө программалоодо компилятордун директиваларын сабаттуу пайдалана билүү башкы маанеге ээ болот. Тилдин башка элементтери менен компилятордун директиваларынын ортосунда принципиалдуу айырмачылыктар бар. Директивалар компилятордун же компоновщиктин (микропроцессордун эмес) иштөөсүн башкарат. Алар компиляторго, мисалы, берилгендерди түздөө (выравнивание), пайдаланылуучу процедуралар менен функцияларды чакыруу тибин ж.б. берет. Мындай директивалардын бир топторун биз карап өткөн мисалдарда пайдаландык. Төмөндө Turbo Pascal тилинин компиляторунун директиваларынын тизмегин алардын арналыштары менен кошо келтиребиз.

{$A+} же {$A-} директивасы

Атайын көрсөтүлбөгөн учурда {$A+} директивасы аракет этет.

Бул директива өзгөрүлмөлөр менен типтештирилген константаларды сөздүн чегине түздөөдөн байттын чегине түздөөгө жана тескерисинче өтүүгө мүмкүнчүлүк берет.

{$A+} абалында өлчөмү боюнча бир байттан ашып кеткен өзгөрүлмөлөр менен типтештирилген константалар машиналык сөздүн (жуп манилүү адрестер) чегине түздөлөт.

{$A-} абалында түздөөгө байланышкан эч кандай амал каралбайт.

{$B+} же {$B-} директивасы

Атайын көрсөтүлбөгөн учурда {$B+} директивасы аракет этет.

Бул директивалар бульдук **and** жана **or** амалдары үчүн кодду генерациялоонун эки ар түрдүү түрүн ишке ашырат.

{$B+} абалында компилятор бульдук туюнтманы толук эсептөө үчүн кодду генерациялайт.

{$B-} абалында компилятор бульдук туюнтманы кыска схема боюнча эсептөө үчүн кодду генерациялайт.

{$C атрибут атрибут…}

Көрсөтүлбөгөн учурда {$C MOVEABLE DEMANDLOAD DISCARDABLE}

Бул директива код сегментинин атрибуттарын (Windows жана корголгон режим үчүн гана)башкаруу үчүн пайдаланылат.

Колдонмо программадагы же библиотекадагы ар бир код сегменти аны эске жүктөгөн учурда ал сегмент өзүн кандай алып жүрөөрүн аныктай турган атрибуттардын жыйындысына ээ болот.

$C директивасы өзү жайгашкан программалык модулдун (модулдун, программанын же библиотеканын) ичиндеги код сегментине гана таасир эте алат.

{$D+} же {$D-} директивасы

Атайын көрсөтүлбөгөн учурда {$D+} директивасы аракет этет.

Бул директива оңдоп-түзөө (отладка) үчүн информацияны генерациялоону берет же алаып салат.

Эгерде программа же модул **{$D+}** абалында компиляцияланса, анда Turbo Pascal дын тиркелген оңдоп-түзөгүчү бизге бул модулду кадамдап аткарууга мүмкүнчүлүк берет жана ал модулда аяктоо чекитин коңт.

{$DEFINE **ат**} директивасы

$DEFINE директивасы берилген атка ээ болгон шарттуу идентификаторду аныктайт. Идентификатор баштапкы компиляцияланууучу коддун калган бөлүгү үчүн же ал {$UNDEFINE **ат** } директивасында пайда болгонго чейин аныкьалган болот. Эгерде «ат» эбак аныкталган болсо, анда {$DEFINE **ат**} директивасы башка эч кандай аракеттерди пайда кылбайт.

{$D **текст**} директивасы

$D директивасы пайдалануучу тарабынан берилген текстти .EXE же .DLL файлынын бөркүндөгү файлдын баяндалыш жазуусуна сыйлыгыштырат.

{$ELSE} директивасы

Атайын көрсөтүлбөгөн учурда {$C+} **директивасы аракет этет.**

ELSE директивасы акыркы {$IFxxx}менен чектелген жана {$ENDIF}тен кийинки баштапкы текстти компиляциялоого же четтетүүгө алып келет.

{$E+} же {$E-} директивасы

Атайын көрсөтүлбөгөн учурда {$E-} **директивасы аракет этет.**

Эмуляция 8087 процессорлошу (соопроцессору) жок болгон учурда анын ишин эмуляциялоочу аткаруучу системанын библиотекасы менен компоновкага уруксат берет же тыйуу салат.

{$N+ E-}абалында Turbo Pascal тили 8087 процессорлошу болгон учурда гана пайдаланылышы мүмкүн болгон программаны түзө алат.

{**$**ENDIF} **директивасы**

ENDIF директивасы акыркы **{$IFxxx}** директивасы мнен башталган шарттуу компиляцияны аяктайт.

{$F+} **же** {$F-} **директивасы**

Атайын көрсөтүлбөгөн учурда {$F+} **директивасы аракет этет.**

Бул директива удаалаш компиляциялануучу процедуруларды жана функцияларды чакыруу тибин башкарат. {$F+} абалында компиляцияланган процедуралар жана функциялар дайыма чакыруунун алыскы тибин пайдаланышат. {$F-} директивасын көрсөткөн учурда Turbo Pascal тили автоматтык түрдө кайрылуунун тиешелүү тибин тандайт: эгерде процедура же функция программалык бирдиктин интерфейс бөлүгүндө баяндалган болсо анда алыскы (far) тибин, андай болбогон учурда жакынкы (near) тибин тандап алат.

{$G+} **же** {$G-} **директивасы**

Атайын көрсөтүлбөгөн учурда:
(реалдык режим) {$G-}
(корголгон режим жана windows) {G+**}**
директивасы аракет этет

$G директивасы 80286 процессору үчүн кодду генерациялоого уруксат берет же тыйуу салат. **{$G-}** абалында 8086 процессорунун инструкцияларя гана генерацияланат, жана ошондой эле ушул абалда генерацияланган программалар 80x86 тобундагы каалаган башка процессордо аткарыла алат.

{ $G **модулдун аты, модулдун аты**…} **директивасы**

$G директивасы бизге компоновщик бир сегменттин ичинде жайгаштырышы керек болгон модулдардын тайпаларын берүүгө мүмкүнчүлүк берет. (Windows жана корголгон режим үчүн гана).

Ар бир G директивасы модулдардын тайпасын берет. $G директивасына программанын же библиотеканын uses операторунда гана уруксат берилет.

Код сегментинин атрибуттарын $C директивасынын жардамында башкарууга болот. Сегменттин уруксат берилген өлчөмү $S директивасы менен коюлат.

{ $IFDEF **индентификатор** } **директивасы**

IFDEF директивасы эгерде “идентификатор” аты аныкталган болсо, анда кийин келүүчү баштапкы текстти компиляциялайт.

{ $IFNDEF индефикатор } **директивасы**

IFNDEF директивасы эгерде “идентификатор” аты аныкталбаган болсо, анда кийин келүүчү баштапкы текстти компиляциялайт.

{$IFORT **кайра_туташтыргыч} директивасы**

Эгерде берилген маалда “кайра_туташтыргыч” көрсөтүлгөн абалда турса, анда IFORT директивасы кийин келүүчү баштапкы текстти компиляциялайт. Кайра_туташтыргыч(директива) өзүнөн кийин плюс (+) же минус (-) белгиси келүүчү директива - кайра_туташтыргычтын атынан турат.

{$I **файлдын_аты**} **директивасы**

Бул директива компиляторго компиляциялоого аталган файлды кошуу зарылдыгын билдирет. Иш жүзүндө файл компиляцияланган текстке түздөн түз {$I **файлдын_аты**} директивасынан кийин сыйлыгыштырылып коюлат.

{$I+} же {$I-} **директивасы**

Атайын көрсөтүлбөгөн учурда {$i+} директивасы аракет этет.

Бул директива кийрүү-чыгаруу процедурасына кайрылуунун жыйынтыгын текшерүү кодун генерациялоону берет же алып салат.

{$K+} же {$K-} **директивасы**

Атайын көрсөтүлбөгөн учурда {$K+} директивасы аракет этет

$K директивасы тиркеме тарабынан экспорттолуучу процедураларды жана функцияларды эффективдүү чакыруу генерациясын башкарат (Windows үчүн гана). Эгерде колдонмо (прикладдык) программа {$K-} абалында генерацияланса, анда чакырылуучу API Windows камтылуучу программаларын түзүүдө ал MakeProcInstance жана FreeProcInstance камтылуучу программаларын пайдаланышы керек. Атайын көрсөтүлбөгөн учурдагы {$K+} абалында колдонмо (прикладдык) программа өзү экспорттолгон кийрүү чекиттерин чакыра алат жана MakeProcInstance жана

FreeProcInstance камтылуучу программаларын пайдалануу зарылчылыгы жок болот.

{$L **файлдын_аты** } **директивасы**

Бул директива компиляторго көрсөтүлгөн файлды компиляцияланууучу программа же программалык модул менен компоновкалоого көрсөтмө берет. $L директивасы сырткы (**external**) камтылуучу программа катары баяндалган камтылуучу программалар үчүн ассемблер тилинде жазылган кодду компоновкалоо үчүн пайдаланылат.

{$L+} же {$L-} **директивасы**

Атайын көрсөтүлбөгөн учурда {$L+} директивасы аракет этет.

Бул директива локалдык идентификаторлор жөнүндөгү информацияны генерациялоону аракетке келтирет же алып салат.

(реалдык режим)
{$M стек_өлчөмү, динам_обл_мин_өлчөмү, динам_обл_макс_өлчөмү}
(корголгон режим)
{$M стек_өлчөмү}
(Windows)
{$M стек_өлчөмү, динам_обл_өлчөмү } директивалары

Атайын көрсөтүлбөгөн учурда

(реалдык режим)
{$M 16384,0, 655360 }
(корголгон режим)
{$M 16384}
(Windows)
{$M 8192,8192}

Бул директива программанын эсин бөлүштүрүү параметрлерин көрсөтөт.

$M директивасы модулдарда пайдаланылса анда ал четтетилет.

{$N+} же {$N-} **директивасы**

Атайын көрсөтүлбөгөн учурда {$N-} директивасы аракет этет.

Бул директива Turbo Pascal да бар болгон жылма чекиттүү эсептөөлөрдүн кодун генерациялоодогу эки моделдин бирөөсүн тандоону ишке ашырат.

{$N-} директивасын көрсөткөн учурда Turbo Pascal аткаруучу системасынын библиотекасынын программаларын чакыруу

жардамында программалык камсыздандыруудагы бардык real тибиндеги эсептөөлөрдү аткаруу үчүн кодду генерациялайт.

{$N+} директивасын көрсөткөн учурда бардык real тибиндеги эсептөөлөрдү 8087 математикалык процессорлоштун (соопроцессордун) жардамында аткаруу үчүн код генерацияланат.

{$O+} же {$O-} директивасы

Атайын көрсөтүлбөгөн учурда {$O-} директивасы аракет этет.

$O директивасы оверлейлик кодду генерациялоого уруксат берет же тыюу салат (реалдык режим үчүн гана). Turbo Pascal модул {$O+} директивасы менен компиляцияланган болгондо гана аны оверлейлик модул катары пайдаланууга уруксат берет.

Компилятордун {$O+} директивасы дээрлик дайыма {$F+} директивасы менен бирге көрсөтүлүп ал чакыруулардын алыскы (far) тибин пайдаланууга карата оверлейлер администраторунун талаптарын аткарууга мүмкүнчүлүк берет.

{$O модулдун_аты} директивасы

{$O модулдун_аты} директивасы анны модулда пайдаланган кезде ал аракет этпейт, ал эми программаны компиляциялаган учурда .EXE файлдын ордуна программадагы пайдаланылуучу кайсы модулдарды .OVR файлга жайгаштыруу керек экендигин берет (реалдык режим үчүн гана).

{$O модулдун_аты} директивалары программанын **uses** операторунан кийин көрсөтүлүшү керек.

{$P+} же {$P-} директивасы

Атайын көрсөтүлбөгөн учурда {$P-} директивасы аракет этет.

$P директивасы **string** кызматчы сөзүнүн жардамында баяндалган параметр-өзгөрүлмөлөрдүн маанисин башкарат. {$P-} абалында **string** кызматчы сөзү менен баяндалган параметр-өзгөрүлмөлөр нормалдуу параметрлер болушат, ал эми {$P+} абалында алар ачык жолчолук параметрлер болушат.

{$Q+} же {$Q-} директивасы

Атайын көрсөтүлбөгөн учурда {$Q-} директивасы аракет этет.

$Q директивасы ашып кетүүнү (переполнение) текшерүү кодун генерациялоону башкарат. {$Q+} абалында айрым бир арифметикалык амалдар (+, -,*, Abs, Sqr, Succ, Жана Pred) ашып кетүүгө текшерилет.

{$Q+} директивасы стандарттык Inc жана Des директиваларына таасир этпейт. Бул процедуралар эч качан ашып кетүүгө текшерилбейт.

Адата $Q кайра туташтыргычы диапазонду текшерүү кодун генерациялоого уруксат берүүчү же тыйуу салуучу $R кайра туташтыргычы менен айкалышта пайдаланылат.

{$R+} же {$R-} директивасы

Атайын көрсөтүлбөгөн учурда {$R-} директивасы аракет этет.

Бул кайра туташтыргыч чек араны текшерүү кодун генерацялоону аракетке келтирет же алып салат. {$R+} директивасы көрсөтүлгөн учурда индекстелген жолчолуу бардык туюнтмалардын жана массивдердин көрсөтүлгөн чек аранын ичинде жайгашуусу текшерилет, ал эми камтылуучу диапазондордун скалярдык чоңдуктарына жана өзгөрүлмөлөрүнө болгон бардык ыйгаруу операторлорунун берилген чек арада жатышы текшерилет.

Эгерде $R директивасы кошулган болсо (включена), анда виртуалдык усулдарга болгон бардык кайрылуулар чакырууну аткаруучу об’ектин экземпляры үчүн инициализациялоо абалына текшерилет.

Мындай чек араны текшерүүгө жана виртуалдык усулдарга уруксат берүү программалардын аткарылышын секиндетет жана алардын өлчөмдөрүн бир топ чоңойтуп жиберет ошондуктан {$R+} директивасын программаны оңдоп-түзөө (отладка) үчүн гана пайдалануу керек.

{ $R **файлдын_аты** } директивасы

$R директивасы колдонмо программага же библиотекага кошулушу керек болгон ресурстар файлынын атын берет (Windows жана корголгон режим үчүн гана). Көрсөтүлгөн файл Windows дун ресурстар файлы болушу керек. Көрсөтүлбөгөн учурда ал .RES кеңейтирилишине ээ болот.

{$S **сегменттин_өлчөмү**} директивасы

Атайын көрсөтүлбөгөн учурда {$S 16384} директивасы аракет этет.

$S директива-параметрине негизги программада же библиотекада гана уруксат берилет (Windows жана корголгон режим үчүн гана). Бул директива модулдарды тайпалоо үчүн код сегментинин ылайыктуу деп эсептелген өлчөмүн берет. Көрсөтүлгөн өлчөм 0..65535 диапазонунда жатышы керек. Мындай өлчөмдөн ашып кеткен модулдар өздөрүнүн менчик код сегменттеринде жайгашышат.

$S директивасы эч качан каталар жөнүндөгү эскертүүлөрдү же билдирүүлөрдү бербейт. Эгерде модулдарды башка модулдар менен бирге код сегментинде жайгаштырууга болбосо, анда ал автоматтык түрдө өзүнчө сегментке жайгашат.

{$S+} же {$S-} директивасы

Атайын көрсөтүлбөгөн учурда {$S+} директивасы аракет этет.

Бул директива стектин ашып кетишин текшерүү менен кодду генерациялоону аракетке келтирет же алып салат.

{$S+} директивасы көрсөтүлгөн учурда компилятор ар бир процедуранын же функциянын башталышында стекте локалдык өзгүрүлмөлөр үчүн жетишерлик орун бөлүнгөнбү же жопу ошону текшерүүчү кодду генерациялайт.

{$T+} же {$T-} директивасы

Атайын көрсөтүлбөгөн учурда { $T- } директивасы аракет этет.

$T директивасы@ амалы тарабынан генерацияланууучу көрсөткүчтөрдүн маанилеринин типтерин башкарат.

{$T-} абалында @ амалынын жыйынтыгынын тиби ар дайым башка бардык көрсөткүчтөрдүн типтери менен биргелешүүчү типтештирилбеген көрсөткүч болот. Качан {$T+} абалында өзгөрүлмөгө болгон шилтемеге @ амалы колдонулганда жыйынтыктын тиби ^T болот, мында T өзгөрүлмөнүн тибине болгон көрсөткүчтөр менен гана биргелешүүчү.

{ $UNDEF ат } директивасы

$UNDEF директивасы мурда аныкталган шарттуу идентификаторду алып салат.

{$V+} же {$V-} директивасы

Атайын көрсөтүлбөгөн учурда { $V+ } директивасы аракет этет.

$V директивасы жолчолорду параметр – өзгөрүлмөлөр катары берүүдө типти текшерүүнү башкарат.

{$V+} абалында формалдык жана иш жүзүндөгү параметрлер окшош (идентичные) жолчолук типтерге (***string***) ээ болушу талап кылынган типти катуу текшерүү аткарылат.

{SV-} абалында иш жүзүндөгү параметр катары жолчолук типтеги каалагандай өзгөрүлмөнү пайдаланууга уруксат берилет, бул ал өзгөрүлмөнүнүн баяндалган узундугу тиешелүү формалдык параметрдин узундугу менен дал келбеген учурда да тура болот.

{$W+} же {$W-} директивасы

Атайын көсөтүлбөгөн учурда {SW+} директивасы аракет этет.

SW директивасы Windows үчүн спецификалык болуп эсептелген чакыруунун алыскы тибине (far) ээ болгон процедуралар жана функциялар үчүн кирүү жана чыгуу кодун генерациялайт (Windows үчүн гана). {SW+} абалында **far** процедуралар жана функциялар үчүн коддун жана чыгуунун атайын жазуусу генерецияланат. Анын натыйжасында Windows дун реалдык режиминин эсин башкаруучу камтылуучу система (подсистема) код сегменти же берилгендер сегменти жылышкандан кийин чакыруулардын чынжырчасына карата тууралоодо (настройка) стектин алыскы кадрларын корректтүү (туура) идентификацялай алат.

{SW-} абалында кирүүнүн же чыгуунун кошумча жазуусу генерацияланат.

{$X+} же {$X-} директивасы

Атайын көрсөтүлбөгөн учурда {SX+} директивасы аракет этет.

Компилятордун $X директивасы Turbo Pascal дын кеңейтирилген синтаксисине уруксат берет же тыюу салат.

{$X+} режимиминде функцияларды чакырууларды операторлор катары пайдаланууга болот, башкача айтканда функциянын жыйынтыгы таштalып жиберилиши мүмкүн.

{$X+} директивасы тиркелген функцияларга колдонулбайт (б.а. System модулунда аныкталган функцияларга).

{$Y+} же {$Y-} директивасы

Атайын көрсөтүлбөгөн учурда {$Y+} директивасы аракет этет.

$Y директивасы шилтемелик информациялардын идентификаторлору үчүн генерацияга уруксат берет же тыюу салат. Программа же модул {$Y+} абалында компиляцияланган учурда Turbo Pascal дын тиркелген карап чыгуу (просмотр) каражаты ушул модул үчүн идентификатордун аныктамаларын жана шилтемелик информацияны чыгара алат.

Адатта $Y директивасы оңдоп-түзөө информацияларын жана локалдык өзгөрүлмөлөр жөнүндөгу информацияларды генерациялоону башкаруучу **$D** жана **$L** кайра туташтыргычтары менен айкалышта пайдаланылат. Эгерде **$D** жана **$L** директиваларына уруксат берилбесе, анда $Y директивасы таасир этпейт.

3 – ТИРКЕМЕ. ASCII КОДУНУН СИМВОЛДОР ТАБЛИЦАСЫ

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	0		32	20		64	40	@	96	60	`
1	1		33	21	!	65	41	A	97	61	a
2	2		34	22	“	66	42	B	98	62	b
3	3		35	23	#	67	43	C	99	63	c
4	4		36	24	$	68	44	D	100	64	d
5	5		37	25	%	69	45	E	101	65	e
6	6		38	26	&	70	46	F	102	66	f
7	7		39	27	‘	71	47	G	103	67	g
8	8		40	28	(	72	48	H	104	68	h
9	9		41	29	)	73	49	I	105	69	i
10	A		42	2A	*	74	4A	J	106	6A	j
11	B		43	2B	+	75	4B	K	107	6B	k
12	C		44	2C	,	76	4C	L	108	6C	l
13	D		45	2D	D	77	4D	M	109	6D	m
14	E		46	2E	.	78	4E	N	110	6E	n
15	F		47	2F	/	79	4F	O	111	6F	o
16	10		48	30	0	80	50	P	112	70	p
17	11		49	31	1	81	51	Q	113	71	q
18	12		50	32	2	82	52	R	114	72	r
19	13		51	33	3	83	53	S	115	73	s
20	14		52	34	4	84	54	T	116	74	t
21	15		53	35	5	85	55	U	117	75	u
22	16		54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	\	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F		63	3F	?	95	5F		127	7F	
128	80	А	160	A0	а	192	C0		224	E0	р
129	81	Б	161	A1	б	193	C1	⊥	225	E1	с

(уландысы)

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
130	82	В	162	A2	в	194	C2	┬	226	E2	т
131	83	Г	163	A3	г	195	C3	├	227	E3	у
132	84	Д	164	A4	д	196	C4	─	228	E4	ф
133	85	Е	165	A5	е	197	C5	┼	229	E5	х
134	86	Ж	166	A6	ж	198	C6	╞	230	E6	ц
135	87	З	167	A7	з	199	C7	╟	231	E7	ч
136	88	И	168	A8	и	200	C8	╚	232	E8	ш
137	89	Й	169	A9	й	201	C9	╔	233	E9	щ
138	8A	К	170	AA	к	202	CA	╩	234	EA	ъ
139	8B	Л	171	AB	л	203	CB	╦	235	EB	ы
140	8C	М	172	AC	м	204	CC	╠	236	EC	ь
141	8D	Н	173	AD	н	205	CD	═	237	ED	э
142	8E	О	174	AE	о	206	CE	╬	238	EE	ю
143	8F	П	175	AF	п	207	CF	╧	239	EF	я
144	90	Р	176	B0	░	208	D0	╨	240	F0	Ё
145	91	С	177	B1	▒	209	D1	╤	241	F1	ё
146	92	Т	178	B2	▓	210	D2	╥	242	F2	Ґ
147	93	У	179	B3	│	211	D3	╙	243	F3	ґ
148	94	Ф	180	B4	┤	212	D4	╘	244	F4	Є
149	95	Х	181	B5	╡	213	D5	╒	245	F5	є
150	96	Ц	182	B6	╢	214	D6	╓	246	F6	І
151	97	Ч	183	B7	╖	215	D7	╫	247	F7	і
152	98	Ш	184	B8	╕	216	D8	╪	248	F8	Ї
153	99	Щ	185	B9	╣	217	D9	┘	249	F9	ї
154	9A	Ъ	186	BA	║	218	DA	┌	250	FA	Ў
155	9B	Ы	187	BB	╗	219	DB	█	251	FB	ў
156	9C	Ь	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	Э	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Ю	190	BE	╛	222	DE	▐	254	FE	■
159	9F	Я	191	BF	┐	223	DF	▀	255	FF	

4 – ТИРКЕМЕ. КЛАВИАТУРАНЫН КОДДОР ТАБЛИЦАСЫ

Клавиатуранын кеңейтирилген коддору

Код	Мааниси	Код	Мааниси
16	Alt+Q	90	Shift+F7
17	Alt+W	91	Shift+F8
18	Alt+E	92	Shift+F9
19	Alt+R	93	Shift+F10
20	Alt+T	94	Ctrl+F1
21	Alt+Y	95	Ctrl+F2
22	Alt+U	96	Ctrl+F3
23	Alt+I	97	Ctrl+F4
24	Alt+O	98	Ctrl+F5
25	Alt+P	99	Ctrl+F6
30	Alt+A	100	Ctrl+F7
31	Alt+S	101	Ctrl+F8
32	Alt+D	102	Ctrl+F9
33	Alt+F	103	Ctrl+F10
34	Alt+G	104	Alt+F1
35	Alt+H	105	Alt+F2
36	Alt+J	106	Alt+F3
37	Alt+K	107	Alt+F4
38	Alt+L	108	Alt+F5
44	Alt+Z	109	Alt+F6
45	Alt+X	110	Alt+F7
46	Alt+C	111	Alt+F8
47	Alt+V	112	Alt+F9
48	Alt+B	113	Alt+F10
49	Alt+N	114	Ctrl+PrtScr
50	Alt+M	115	Ctrl+ ¬
59	F1	116	Ctrl+ ®
60	F2	117	Ctrl+End
61	F3	118	Ctrl+PgUp
62	F4	119	Ctrl+Home
63	F5	120	Alt+1
64	F6	121	Alt+2
65	F7	122	Alt+3
66	F8	123	Alt+4
67	F9	124	Alt+5
68	F10	125	Alt+6
71	Home	126	Alt+7

(уландысы)

Код	Маани си	Код	Маани си
72		127	Alt+8
73	PgDn	128	Alt+9
75	¬	129	Alt+0
77	®	130	Alt+ -
79	End	131	Alt+ =
80	¯	132	Ctrl+PgDn
81	PgUp	133	F11
82	Ins	134	F12
83	Del	135	Shift+F11
84	Shift+F1	136	Shift+F12
85	Shift+F2	137	Ctrl+F11
86	Shift+F3	138	Ctrl+F12
87	Shift+F4	139	Alt+F11
88	Shift+F5	140	Alt+F12
89	Shift+F6		

Клавиатуранын сурамжылоо (опрос) коддору

Клавиша	Сурамжы-лоо коду	Клавиша	Сурамжы-лоо коду	Клавиша	Сурамжы-лоо коду
Esc	01	CTR	1D	Пробел	39
! 1	02	A	1E	Caps Lock	3A
@ 2	03	S	1F	F1	3B
# 3	04	D	20	F2	3C
$ 4	05	F	21	F3	3D
% 5	06	G	22	F4	3E
^ 6	07	H	23	F5	3F
&7	08	J	24	F6	40
* 8	09	K	25	F7	41
(9	OA	L	26	F8	42
)0	OB	;	27	F9	43
_ -	OC	”	28	F10	44
+=	OD	~ `	29	F11	D9
BS	OE	Сол Shift	2A	F12	DA
Tab	OF	/	2B	Home	47
Q	10	Z	2C	↑	48

(уландысы)

Клавиша	Сурамжы-лоо коду	Клавиша	Сурамжы-лоо коду	Клавиша	Сурамжы-лоо коду
W	11	X	2D	Pg Up	49
E	12	C	2E	Серый-	4A
R	13	V	2F	←	4B
T	14	B	30	5	4C
Y	15	N	31	→	4D
U	16	M	32	Серый+	4E
I	17	< ,	33	End	4F
O	18	> .	34	↓	50
P	19	? /	35	Pg Dn	51
{[	1A	Оң Shift	36	Ins	52
]}	1B	Prtscr	37	Del	53
Enter	1C	Alt	38	Num Lock	45

АДАБИЯТТАР

1. Абрамов В.Г., Трифонов Н.П. Трифонова Г.Н. Введение в язык Паскаль. – М.: Наука Гл. ред. физ.- мат. лит., 1988. – 320 с.
2. Абрамов С.А., Г.Г. Гнездилова, Е.Н.Капустина, М. И.Селюн. Задачи по программированию. – М.: Наука. гл. ред. физ-мат. лит., 1988. – 224 с.
3. Бердиев А., А.Исраилов, Р.Табышев, Э.Өсөрөв. Компьютер: колдонуучу, программалоо (DOS, BASIC, Turbo Pascal) Бишкек, 2000. – 316 б.
4. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. – М.:МИР, 1998. – 360 с.
5. Гордеев А.В., А.Ю.Молчанов. Системное программное обеспечение. Учебник, Санкт-Петербург-Москва-Харьков-Минск, 2002. – 700 с.
6. Епанешников А., Епанешников М. Программирование в среде Turbo Pascal 7.0 – 4-е изд., испр. и дополн. – М.: Диалог-МИФИ, 2000. – 367 с.
7. Климова Л.М., Практическое программирование. Решение типовых задач. Pascal 7.0, «Кудиц-Образ» Москва, 2000. – 515 с.
8. Марченко А.И., Марченко Л.А. Программирование в среде Turbo Pascal 7.0, Под. ред. Тарасенко В.П. – К.: ВЕК+, М.: Бином Универсал, 1998. – 496 с.
9. А. С. Мироненко, Императивное и объектно-ориентированное программирование на Turbo Pascal и Delphi, Одесса: ВМВ, 2007. – 408 с.
10. Немнюгин С.А., Turbo Pascal. Учебник, Санкт-Петербург-Москва-Харьков-Минск, 2000. – 491 с.
11. Осмоналиев А.Б., Аркабаев Н.К. Borland Pascal 7.0. Программалоонун негиздери. Окуу китеби: I бөлүк. 2008, – 256 б.
12. Өмүралиев А., Маалыматтар технологиясы, Бишкек, КТМУ, 2002. - 293 б.

Осмоналиев А.Б.
Аркабаев Н.К.

Borland Pascal 7.0

Программалоонун негиздери

2-БӨЛҮК

Терүүгө 09.09.2010-ж. берилди
Басууга 28.10.10 ж. кол коюлду.
Көлөмү 60x84 1/16. 19 басма табак. Нускасы 500.
Буюртма № *987* .
Ош ш. Курманжан Датка – 209. «Ошоблбасмакана» АК.