TÜRKMENISTANYŇ BILIM MINISTRLIGI
TÜRKMEN POLITEHNIKI INSTITUTY

O. Nurgeldiýew

Obýekte gönükdirilen programmirleme

Hünär: Maglumatlary işläp taýýarlamagyň we dolandyrmagyň awtomatlaşdyrylan ulgamlary

Aşgabat 2010 ý.

GIRIŞ

Garaşsyz, baky Bitarap Türkmenistan döwletimizde geljegimiz bolan ýaşlaryň dünýäniň iň ösen talaplaryna laýyk gelýän derejede bilim almagy üçin ähli işler edilýär.

Hormatly Prezidentimiz döwlet başyna geçen ilkinji gününden bilime, ylma giň ýol açdy, Türkmenistan ýurdumyzda milli bilim ulgamyny kämilleşdirmek boýunça düýpli özgertmeler geçirmäge girişdi.

Türkmenistanyň Prezidenti Gurbanguly Berdimuhamedowyň «Türkmenistanda bilim ulgamyny kämilleşdirmek hakynda» 2007-nji ýylyň 15-nji fewralyndaky Permany bilim ulgamyndaky düýpli özgertmeleriň başyny başlady.

Häzirki wagtda milli bilim ulgamyndaky döwrebap özgertmeler ýaş nesliň ýokary derejede bilim almagyna we terbiýelenmegine, giň dünýägaraýyşly, edep-terbiýeli, tämiz ahlakly, kämil hünärmenler bolup ýetişmeklerine uly ýardam edýär.

Häzirki wagtda kompýuterler önümçiliginiň islendik pudagynda giňden ýaýrandyr. Şonuň üçin hem hasaplaýyş tehnikasy bilen tanyşlyk talyplaryň haýsy hünär boýunça bilim alýanlygyna garamazdan öwrenilýär.

Şu dersiň özi ýöriteleşdirilen bolup, kompýuter tehnikasy bilen işlemekde has ussat bolmaly hünärmenler üçin niýetlenen.

Dersi okatmagyň maksady – Bu dersde häzirki döwürde has giňden ýaýran Delphi ulgamynda programma ýazmaklyga seredilýär. Delphi ulgamy Windows-da işleýän programma önümlerini döretmeklige mümkinçilik berýär. Bu ulgam önümçilikde ýüze çykýan meseleleriň islendigini kompýuteriň kömegi bilen çözmeklige mümkinçilik berýär. Bu ulgamda köp sanly taýýar komponentleriň bar bolmagy, programmany has çalt çözmeklige mýmkinçilik döredýär. Häzirki wagtda giňden ulanylýan programma önümleriniň köpüsi Delphi ulgamynda

taýýarlanandyr. Delphi ulgamy öz düzüminde assembler kodlaryny hem ulanyp bilýär. Munuň ozi bolsa onuň mümkinçiliklerini has-da giňeldýär.

Häzirki wagtda Döwletimizde yglan edilen beýik galkynyşlar döwründe halk hojalygynyň hemme pudaklarynda dürli döwrebap, çylşyrymly gurluşlar satyn alnýar we giňden ornaşdyrylýär. Ol gurluşlar bolsa mikroprosessorlaryň kömegi bilen öňden taýýarlanan programmalaryň esasynda dolandyrylýar. Ol gurluşlaryň işleýşine önat düşünmek üçin, täzeçe dolandyrmak üçin Delphi ulgamyny bilmeklik orän zerurdyr.

Delphi sistemasynyň prinsipi we esasy düşünjeleri

Obýekt PASCAL algoritmik dili häzirki döwürde personal kompýuterlerde iň bir ýörgünli dilleriň biridir. Bu dil 1971-nji ýylda şweýsar professory Niklaus Wirt tarapyndan işlenip düzülýär we onda esasan strukturalaýyn programmirlemeklige uly üns berilýär. Diliň ady XVII-nji asyrda ýaşap geçen beýik fransuz filosofy, matematigi Blez Paskalyň hatyrasyna dakylýar(B.Paskal 1623-1662).

1980-nji ýylda personal (kiçi) kompýuterler üçin MS-PASCAL, PASCAL, PASCAL-80 translýatorlarynyň döredilmegi bilen PASCAL, OBJECT-PASCAL ýokary derejeli ylmy algoritmik dil hökmünde öz ornuny hasda berkitdi. Häzirki döwürde OBJECT-PASCAL dili islendik personal kompýuterlerde ulanylýan iň kämil programmirleýiş dilleriniň biri hasaplanylýar.

PASCAL diliniň esasy simwollary. Kömekçi sözler.

OBJECT -PASCAL diliniň elipbiýi aşakdaky simwollardan, kömekçi sözlerden we belgilerden durýar:

1) Latyn elipbiýiniň uly we kiçi harplary:
   A,B,C,…, X,Y,Z;
   a,b,c,… x,y,z;
2) Onluk hasaplaýyş sistemasynyň sanlary: 0,1,2,… ,9
3) Arifmetiki amallaryň belgileri:

+(goşmak), -(aýyrmak), *(köpeltmek), /(bölmek), Div(bitin sanlary bölmek bilen taşlamak), mod(bitin sanlarda bölmek bilen galyndyny kesgitlemek). Görkezilen amallar bitin san köplüginde ýerine ýetirilmek bilen degişlilikde paýy we galyndyny kesgitleýär.

Meselem:

20 div 3 = 3; 16 div 3 = 5; 25 div 4 = 6;
13 mod 5 = 3; 24 mod 5 =4; 12 mod 3 =0;

4) Gatnaşyk belgileri:

>(uly), >=(uly we deň), <(kiçi), <=(kiçi we deň), =(deň), <>(deň däl).

5) Logiki amallaryň belgileri:

not(inkär etmek), or(dizýunksiýa-logiki goşmak), and(konýunksiýa-logiki köpeltmek)

6) Eýe bolmak operatorynyň belgisi :=

7) Bir-biriniň yzyndan gelýän iki gurluşyň arasyny aýyrmak üçin ulanylýan belgiler: .(nokat); ,(otur); :(goşa nokat); ;(nokatly otur);

8) Beýleki ýörite belgiler: ‘(apostrof), ()(açylýan we ýapylýan ýay), [ ](kwadrat ýaý), !(ýüzlenme belgisi), ?(sorag belgisi), %(prosent), (boş öýjük we ş.m.

OBJECT-PASCAL algoritmik dilinde programmany ýazmak üçin aşakdaky kömekçi sözlerden peýdalanýarlar:

KÖMEKÇI SÖZLER	MANYSY
ARRAY	MASSIW
BEGIN	BAŞLANGYÇ
CASE	WARIANT,SAÝLAW
CONST	HEMIŞELIK
DO	ÝERINE ÝETIRMEK
DOWNTO	KEMELTMEK
ELSE	ÝOGSAM
END	SOŇY

FILE	FAÝL
FOR	ÜÇIN
FUNCTION	FUNKSIÝA
GOTO	GEÇMEK
IF	EGER
IN	DEGIŞLI
LABEL	BELGI
NUL	HIÇ ZAT
OF	-DAN, -DEN GOŞULMASY
PACKED	GAPLANAN
PROCEDURE	PROSEDURA
PROGRAM	PROGRAMMA
RECORD	ÝAZGY
REPEAT	GAÝTALAMAK
SET	KÖPLÜK
THEN	ONDA
WHILE	ENTEK
WITH	BILEN

Pascal dilinde jemi hasaplamak üçin programma:

```
uses crt;
Label 2;
Var S,ak, eps: real; k: longint;
Begin
clrscr;
S:=0; K:=1; eps:=1E-04;
2: ak:=(2*(K+1))/(2*K*(K+1));
{writeln('k=',k,'    ak=',ak,'  maydalawjy=',2*k*(k+1));}
S:=S+ak; K:=K+1;
if abs(ak)>EPS then goto 2;
Writeln ('s=', s:8:3,'   K=',K);
End.
{-----------------------------------------------------}
```

Berlen n natural sana çenli bar bolan ýönekeý sanlary tapmaklygyň programmasy. Netije Ys.txt faýla ýazylmaly:

```
uses crt;
var
  i,n,k:longint;
  t: text;
{-------------------------------------------------}
function ys(k:longint):boolean;
var
  i : integer;
begin
for i:=2 to round(sqrt(k)) do
if k mod i = 0 then begin ys:=false; exit; end;
ys:=true;
end;
{-------------------------------------------------}
begin
clrscr;
assign(t,'Ys.txt'); rewrite(t);
write('N='); read(n);
k:=2;  write(t,2,#9,3,#9);
for i:=4 to n do
begin
 if i mod 3 = 0 then continue;
 if ys(i) then begin
     if k mod 8=0 then writeln(t);
     inc(k); write(t,i,#9);
     end;
end;
close(t);
end.
```

```
{Perhat, lunka 27.01.2006(5)}
uses crt;
```

```
var i,j,r,n,m: integer;
   l: array [1..7] of 0..2;
   b: boolean;
{------------------------------}
procedure cap;
var i:integer;
begin
for i:= 1 to 7 do
write(l[i]:2);
writeln;
end;
{------------------------------}
procedure barlag;
var i:integer;
begin
if (l[1]=2) and(l[2]=2) and (l[3]=2) then begin readkey; halt;
end;
end;
{------------------------------}
procedure nol;
var i,j,n,m: integer;
begin
j:=0;
repeat
for i:= 1 to r do
if l[i]=0 then begin n:=i; break; end;
if l[n-1]=2 then begin
                 m:=l[n-1];
                 l[n-1]:=l[n];
                 l[n]:=m;
               end;
cap;
readkey;
until (l[n-1]=1);
end;
```

```
{------------------------------}
procedure orun;
var k,w: integer;
begin
repeat
for i:= 1 to r do
if l[i]=0 then begin n:=i; break; end;
if b=true then begin
                if l[n-1]=1 then begin
                                k:=l[n-1];
                                l[n-1]:=l[n];
                                l[n]:=k;
                                end;
                b:=false;
              end
          else begin if l[n+2]=2 then begin
                                k:=l[n+2];
                                l[n+2]:=l[n];
                                l[n]:=k;
                                end;
              b:=true;
              end;
cap;
readkey;
until l[r]=1;
end;
{------------------------------}
begin
clrscr;
b:=true;
      l[1]:=1; l[2]:=1; l[3]:=1; l[4]:=0; l[5]:=2; l[6]:=2; l[7]:=2;
cap;
r:=8;
repeat
dec(r);
```

```
orun;
barlag;
nol;
b:=true;
until r=5;
readkey;
end.
```

Obýekt pascal diliniň esaslary

Ulanyjy tarapyndan ýazylan programma ýerine ýetirilýän wagtynda öz bahasyny üýtgetmeýän ululyklara hemişelikler diýilýär. OBJECT-PASCAL algoritmik dilinde hemişelikleri aşakdakytoparlara bölmek mümkin:

1) san hemişelikleri
2) liter hemişelikleri
3) setir hemişelikleri
4) logiki hemişelikler

San hemişelikleri iki görnüşde bolup bilýär:

a) bitin san hemişelikleri
b) hakyky san hemişelikleri

Bitin san hemişeliklerиniň tipi INTEGER, hakyky san hemişelikleri bolsa REAL görnüşinde bellenilýär. Meselem, OBJECT PASCAL dilinde bitin san hemişelikleri 23, 2001, -97, -465 we ş. m. ýaly ýazmak bolýar. Bitin san hemişelikleri (-32768, 32767) san aralykda çäklendirilýär.

Hakyky sanlar kompýuteriň ýadynda ýakynlaşan, ýagny käbir takyklyk bilen ýazylýar. Meselem: 21.0, 21.000001, 20.999999 sanlaryň ählisi hakyky tipde berlen 21-i aňladýar. Hakyky sanlary iki görnüşde ýazyp bolýar:

1) fiksirlenen nokat arkaly; 2) Üýtgeýän nokat arkaly

Meselem 0.161, -19.74, -4.0 we 141.69 sanlar fiksirlenen nokat arkaly ýazylandyr. Programmada ulanylýan absolýut ululygy boýunça has kiçi we has uly sanlar üýtgeýän nokat arkaly aňladylýar. Meselem 0.00075 hakyky sany üýtgeýäň nokat arkaly 75E-05, 7.5E-04, 0.75E03 we ş.m. görnüşinde ýazmak

mümkündir. Hakyky san hemişelikleri absolýut ululygy boýunça (10^{-38}, 10^{38}) aralykda çaklendirilendir. Diliň elipbiýine girýän diňe bir sany simwoldan ybarat bolan ululyga liter hemişeligi diýilýär. Liter hemişeliklerini “ ′ “ apostrof belgisiniň arasynda ýazylýar. Meselem ‘L’ , ’*’ , ‘C’ , ‘8’ , ‘ ‘ we .ş.m. liter hemişelikleridir. Liter hemişelikleriň tipi CHAR görnüşde ýazylýar. OBJECT-PASCAL dilinde iki sany logiki hemişelik ulanylýar:

1) TRUE (çyn); 2) FALSE (ýalan).

Logiki hemişelikleriň tipi BOOLEAN görnüşde bellenilýär. Setir hemişeligi diýip diliň elipbiýine girýän harplaryň, sanlaryň we beýleki simwollaryň toplumyna aýdylýar. Setir hemişeligi hem apostrof belgisiniň içinde ýazylýar. Meselem: ‘Y=’, ‘NETIJE’, ‘JEM’ we ş.m. setir hemişeliklerine mysaldyr. OBJECT-PASCAL dilinde setir hemişeliginiň tipi STRING görnüşde ýazylýar. Setirdäki ähli simwollaryň sanyna setir hemişeliginiň uzynlygy diýilýär. Uzynlygy diňe 1-e deň bolan setir hemişeligine liter hemişeligi hökmünde seretmek mümkindir.

Üýtgeýän ululyklar

Kompýuterde programma ýerine ýetirilýän mahalynda öz bahasyny üýtgedip bilýän ululyklara üýtgeýän ululyk diýilýär. Üýtgeýän ululyklary atlandyrmak üçin ulanylýan belgilere bolsa identifikatorlar diýilýär. Identifikator üýtgeýän ululyga berilýän bahanyň kompýuteriň ýadynyň haýsy öýjüginde ýerleşýändigini görkezmek üçin hyzmat edýär. Identifikator üýtgeýän ululyga berilýän bahanyň kompýuteriň ýadynyň haýsy öýjüginde ýerleşýändigini görkezmek üçin hyzmat edýär.

Identifikatoryň birinji simwoly hökman latyn harpy bolmaly we OBJECT-PASCAL-da identifikatoryň uzynlygy çäklendirilmeýär. Meselem: OBJECT-PASCALI, FTK, FIZIK, TALYP we ş.m. identifikatorlara mysal bolup biler.

Üýtgeýän ululyklaryň atlary kömekçi sözleriň hiç birine hem gabat gelmez ýaly edip saýlap almaly. OBJECT-PASCAL dilinde programmada ulanylýan hemişelikleriň, funksiýalaryň, proseduralaryň we faýllaryň atlaryna hem identifikatorlar diýilip düşünilýär. Identifikatory standart we standart däl identifikatorlara bölmek bolýar. Standart identifikatorlar diýip diliň özünde öňden kabul edilen identifikatorlara aýdylýar. Meselem:

№	standart identifikator	manysy
1	FALSE, TRUE	standart logiki hemişelikler
2	MAXINT,PI	standart san hemişelikleri
3	ABS, SQR, COS, SIN, LN, EXP, SQRT, ARCTAN, TRUNC, ROUND, PRED, SUCC, ORD, CHR, ODD, EOF, EOLN we ş.m.	standart funksiýalar
4	GET, PUT, NEW, REWRITE, READ, PACK, UNPACK, READLN, RESET, WRITE, WRITELN, PAGE we ş.m.	standart proseduralar

Standart däl identifikator diýip programmany düzýän ulanyjynyň girizýäň identifikatoryna aýdylýar. Olar programma ýazylanda, başda tipleri boýunça kesgitlenilýär. Meselem:

VAR A100, OMEGA, K : INTEGER; Y5, X10, Z : REAL;

C,T : BOOLEAN; D10 : CHAR;

Gurluşy boýunça üýtgeýän ululyklary iki topara bölmek bolýar:

1) Ýönekeý üýtgeýän ululyklar;

2) Indeksli üýtgeýän ululyklar

Ýönekeý üýtgeýän ululyklara mysallar:
X2, Y5, Z1, Q3 we ş.m.
Olar INTEGER,REAL,BOOLEAN,CHAR,…tipleriň islendigine degişli bolup bilýärler.

OBJECT PASCAL dilinde bir tipe degişli bolan birnäçe üýtgeýän ululyklary identifikatorlar arkaly hem atlandyrmak mümkin . Bu ýagdaýda üýtgeýän ululyklaryň her birine indeksli üýtgeýän ululyklar diýilýär. Indeksli üýtgeýän ululyklar bir indeksli, iki indeksli we köp indeksli bolup bilerler: Meselem:
A[6], C5[15], XY[I], B[10]-bir indeksli üýtgeýän ululyklar;
A[3,4], B[2,3], C5[I,J], D[4,4]-iki indeksli üýtgeýän ululyklar.

```
label m1;
var i,j,n,m,x,y,x2,y2: integer;
   a: array [1..8,1..8] of char;
   b,b1: boolean;
{-----------------------------------}
procedure cap;
var i,j: integer;
begin
for i:= 1 to 8 do
begin
for j:= 1 to 8 do
write(a[i,j]:2);
writeln;
end;
end;
{-----------------------------------}
procedure ferz(x,y: integer);
var i,j,k,l,x1,y1: integer;
begin
l:=x+y; k:=x-y; x1:=x; y1:=y;
for i:= 1 to 8 do
```

```
for j:= 1 to 8 do
begin
if i+j=l then a[i,j]:='1';
if i-j=k then a[i,j]:='1';
end;
for i:= 1 to 8 do
  a[i,y]:='1';
for i:= 1 to 8 do
  a[x,i]:='1';
a[x,y]:='x';
end;
{--------------------------------------}
procedure prob(var b:boolean);
var i,j,o,p: integer;
begin
b:=true;
for i:= 1 to 8 do
if a[i,y2]='0' then begin ferz(i,y2); exit; end;
b:=false;
end;
{--------------------------------------}
procedure barlag(var b:boolean);
var i,j,h: integer;
begin
h:=0; b:=true;
for i:= 1 to 8 do
for j:= 1 to 8 do
if a[i,j]='x' then inc(h);
if h<8 then b:=false;
end;
{--------------------------------------}
begin
clrscr;
x:=1; y:=1; x2:=1; y2:=1;
x2:=1; y2:=1;
```

```
m1:
for i:= 1 to 8 do
for j:= 1 to 8 do
a[i,j]:='0';
repeat
ferz(x,y);
inc(y2);
writeln;
cap;
readkey;
prob(b1);
until y2=8;
barlag(b);
if x=8 then begin readkey; halt; end;
if b=false then begin inc(x); goto m1 end;
readkey;
end.
```

Obýekt pascal diliniň esaslary

Standart funksiýalar

Köp ulanylýan elementar funksiýalaryň bahalaryny hasaplamak üçin standart funksiýalardan peýdalanylýar. Hasaplamalarda köp duş gelýän standart funksiýalar: sinx, cosx, e^x, lnx, $\sqrt{x}$, $|x|$, x^2 we ş.m.Standart funksiýalaryň bahalaryny hasaplamak üçin nietlenen programmalar kompýuteriň ýadynda saklanylýar. Olara ýüzlenmek üçin funksiýalaryň adyny ýazyp, ýaýyň içinde argumentiň bahasy görkezilýär.Trigonometrik funksiýalaryň bahalary radianlarda hasaplanylýar. Eger burç gradiuslarda berlen bolsa, onda ony aşakdaky formula boýunça radianlara öwürmeli:

radian=gradius*PI/180

Ters triganometrik funksiýalaryny hasaplamak üçin aşakdaky formulalardan peýdalanmak bolýar:

arcsin x=arctg(x/sqrt(1-x^2);
arccos x=arctg(sqrt(1-x^2)/x);
arcctg x=arctg(1/x).
Islendik esaslar logarifmik hasaplamakda bir esasdan beýleki esasa geçmek üçin
$\log_a b=\ln b/\ln a$.
formulalardan peýdalanylýar

Standart funksiýalaryň bahasy hasaplanylanda onuň argumentini ýaýyň içide ýazmaly we açylýan-ýapylýan ýaýlaryň sany deň bolmalydyr. Meselem: $\cos^2 x^2$
funksiýanyň bahasy hasaplanylanda aňlatmany
sqr(cos(sqr(x))) görnüşde ýazyp bolýar.

Tablisada beýan edilişi ýaly , TRUNC standart funksiýasy hakyky sanyň galyndy bölegini kesip taşlaýar.
Meselem:
TRUNC(-12.75)=-12 ýa-da TRUNC (46.69)=46
ROUND standart funksiýasy bolsa argumenti iň ýakyn sana çenli tegelekleyär.

Meselem:
ROUND(-9.5)=-10.0 ýa-da ROUND(35.49)=35.0 PRED we SUCC standart funksiýalary INTEGER, CHAR we BOOLEAN tipli argumentler üçin degişlilikde berlen elementiň öň ýanyndaky we yz ýanyndaky elementini kesgitleýär, Meselem:
PRED(6)=5; SUCC(9)=10 (Argument INTEGER tipe degişli bolmalydyr.);
PRED(Y)=X; SUCC(Y)=Z (argument CHAR tipe degişli bolmalydyr).

Funksiýanyň ady	Argumentiň tipi	Netijäniň tipi	Funksiýanyň manysy
ABS(X)	bitin/hakyky	argumentin-äki ýaly	X-iň absolýut bahasy
Pi	-	hakyky	Pi sanyň bahasy
SIN(X)	Hakyky	hakyky	X radian-dan alnan sinus
COS(X)	Hakyky	hakyky	X radian-dan alnan cosinus
ARCTAN(X)	Hakyky	hakyky	Xradiandan alnan arktangens
SQRT(X)	bitin/hakyky	argumentin-äki ýaly	X-dan alnan kwadrat kök
SQR(X)	bitin/hakyky	argumentin-äki ýaly	X-iň kwadraty
EXP(X)	Hakyky	hakyky	e-niň derejesindäki X
LN(X)	Hakyky	hakyky	X-iň natural logarifmi
TRUNC(X)	Hakyky	Longint	X-iň bitin bölegi
FRAC(X)	Hakyky	hakyky	X-iň drob bölegi
INT(X)	Hakyky	hakyky	X-iň bitin bölegi
ROUND(X)	-	Longint	X-i iň ýakyn bitin sana çenli tegeleklemek
RANDOM	-	hakyky	(0…1) aralykdan tötänleýin sanlar
RANDOM(X)	Word	Word	(0…X) aralykdan tötänleýin sanlar
ODD(X)	Bitin	Logiki	Eger X-iň bahasy täk bolsa nrtije TRUE

Obýekt pascal dilinde ulanylýam esasy standart funksiýalar

Standart funksiýalara pragrammanyň islendik ýerinde eýe bolmak operatorlarynyň sag tarapyndaky aňlatmadan ýüzlenip bolýar. Meselem: $Y:=a^3*SIN^2(X)$ OBJECT-PASCAL algoritmik dilinde şu aşakdaky standart funksiýa ulanylýar:

ORD we CHR standart funksiýalar bir-birine ters funksiýalar bolup, olara başgaça özgerdiji funksiýalar hem diýärler.

ORD(X) standard funksiýa X simwola degişli bolan tertip nomeri kesgitleýär. Meselem:
ORD(‘0’)=48; ORD(‘A’)=65;
ORD(‘ ‘)=32; ORD(‘B’)=66; we ş.m.
CHR(I) standart funksiýa nomer boýunça oňa degişli bolan simwoly kesgitleýär. Meselem: CHR(48)=0; CHR(65)=A we ş.m.
ODD(X) standart funksiýa X bitin sanyň jübit-täkligini kesgitleýär:

Eger x san täk bolsa, onda ODD(x) standart funksiýa ,,TRUE’’ baha eýe bolýar. Meselem: ODD(11)=TRUE; ODD(19)=TRUE
Eger x san jübit bolsa, onda ODD x standart funksiýa ,,FALSE’’ baha eýe bolýar. Mselem ODD(8)=FALSE; ODD(36)=FALSE.
ODD(x) standart logiki funksiýalara degişlidir.EOLN(x)-logiki funksiýa x-faýlda setiriň soňuny kesgitlemek üçin ulanylýar.
EOF (x)-x-faýlyň soňuny kesgitlemek üçin ulanylýar.

Aňlatmalar

OBJECT-PASCAL algaritmik dilinde aňlatmalar hemişeliklerden, ýönekeý we indeksli üýtkeýän ululyklardan, standart funksiýalrdan, amallaryň belgilerinden we ýaýlardan ybarat bolup bilerler.
Aňlatmanyň bahasy INTEGER ýa-da REAL tipe degişli bolsa, onda onuň ýaly aňlatmalar arifmetiki aňlatmalar diýilýär. Meselem: (1*25 *sin(x)+PI)/SQR(A+B)/x.
Arifmetiki aňlatmalarda amallar aşakdaky tiplerde ýerine ýetirilýär:

1) ilki ýaýyň içi ýerine ýetirilýär.
2) eger ýaý özünde ýenede bir näçe faýlary saklaýan bolsa onda hasaplama iň kiçi ýaýyň içinden başlanýar;
3) ýaýyň içinde ilki standart funksiýalaryň bahalary hasaplanylýar;

4) soňra amallar: *, /, DIW, MOD, +, -, tirtipde ýerine ýetirilýär.

OBJECT-PASCAL dilinde derejä götermek amaly LN(x) we EXP(x) standart funksiýalar arkaly aňladylýar. Meselem:

$1,6+2,75*x^{50}$ aňlatmany OBJECT-PASCAL-da

1.6+2.75*EXP(100*LN(x))

görnüşde ýazmak bolar.

Aňlatmanyň bahasy hökmünde ,,TRUE'' ýa-da ,,FALSE'' logiki hemişelikler ulanylan bolsa onda onuň ýaly aňlatmalara logiki aňlatmalar diýilýär. Iki sany arifmetiki aňlatmanyň arasynda <, >, <=, >=, =, gatnaşyk belgileriniň birden birini ulanmak bilen ýönekeý logiki aňlatmany ýazmak bolýar. Meselem:

Y1+3.95<=X2+SQR(A);

6.25+4.75<>17.75

Has çylşyrymly logiki aňlatmalar ýönekeý logiki aňlatmalardan NOT, AND, OR, XOR logiki amalar bilen tapawutlanýar. Meselem:

(B>0.85) AND (I<4.5)

Logiki amallar aşakdaky hkykatlyk tablisasy bilen kesgitlenilýär.

A	B	A AND B	A OR B	A XOR B
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F
T	T	T	T	F

A	NOT A
T	F
F	T

Bu ýerde A we B islendik logiki aňlatmalar; T we F degişlilikde amallar NOT, AND, OR, XOR we OBJECT-PASCAL-da ýerine ýetirilýär. Logiki aňlatmalar programmada köplenç şertli geçiş operatorynda şert hökmünde ulanylýar.

```
var
  a : array[1..10000] of longint;
  k,n,i,j:longint;
```

```
  l : boolean;
  f : text;
begin
clrscr;
      assign(f,'ys01.txt'); rewrite(f);
write('n='); read(n);
k:=2;
a[1]:=2; a[2]:=3; write(f,a[1]:8,a[2]:8);
for i:=3 to n do
 begin{1}
 if (not odd(i)) or (i mod 3 = 0) then continue;
 l:=true;
for j:=1 to trunc(k) do
  if i mod a[j]=0 then begin l:=false; break; end;

  if l then begin inc(k); a[k]:=i; write(f,a[k]:8); end;

 end;{1}
close(f);
{for i:=1 to k do write(a[i]:8);
writeln(#10#13'k=',k);
}
readln;readln;
end
```

Maglumatlaryň täze tipini kesgitlemek

OBJECT-PASCAL algoritmik dilinde programa söz başyndan we blokdan ybarat bolýar. Blogyň soňunda ‘’ . ‘’-nokat belgisi goýulýar. Programmanyň söz başysy umumy görnüşde aşakdaky ýaly ýazylýar:

PROGRAM M<programanyň ady>;

Bu ýerde <programmanyň ady>-ulanyjy tarapyndan girizilýän islendik at.

Blok umumy ýagdaýda 6 sany bölümden ybarat bolýar:

1) Belgiler bölümi;

2) Hemişelikler bölümi;
3) Tipler bölümi;
4) Üýdegeýän ululyklar bölümi;
5) Funksiýalar we proseduralar bölümi;
6) Operatorlar bölümi.

Belgiler bölümi umumy ýagdaýda

LABEL n1,n2,…,n_k ;

gornüşde ýazylýar. Bu ýerde LABEL-bölümiň ady bolup ,,belgi'' diýen manyny aňladýar; n1,n2,…,n_k-programmada ulanylýan belgileriň sanawy. OBJECT-PASCAL algoritmik dilinde belgi hökmünde atlary we 0-dan 9999-a çenli islendik bütin polojitel sany ulanmak mümkin. Belgileriň uzynlygy 4-sandan geçmeli däl we onuň öňünde alamat goýulmaýar. Belgileriň sanawy tertip boýunça ýazmak hökman däl, ol isilendik tertipde ýazylyp biliner. Programma ýazylanda belgi bilen islendik operatoryň arasynda ,, : ‘‘-goşa nokat belgisi goýulýar. Meselem:

```
LABEL 10;
……
10:X:=X+1 ;
…….
```

Eger programmada belgi ulanylmaýan bolsa, onda bu bölüm ýazylmaýar. Hemişelikler bölümi umumy ýagdaýda

CONST a1=c1; a2=c2, ..., an=cn;

görnüşde ýazylýar. Bu ýerde CONST – bölümiň ady a1, a2, ..., an – programmada ulanylýan hemişelikleriň atlary; c1, c2, ..., cn – degişlilikde a1, a2, ..., an hemişelikleriň eýe bolýan bahalary.

Her bir hemişelik öz bahasyna eýe bolandan soň hökman ‘‘ ; ‘‘ – nokatly otur belgisi goýulmagy zerurdyr. Hemişelik bilen onuň bahasynyň arasynda ‘‘ = ‘‘ – deňdir belgisi goýulýar.

Meselem:

CONST NMIN = 1; NMAX = 100; PI = 3.141592;

Hemişelikleriň tipi olaryň eýe bolýan bahalarynyň tipleri boýunça kesgitlenilýär.

Meselem:

NMIN we NMAX - INTEGER tipe, PI bolsa REAL tipe degişlidir. Eger programmada hiç hili hemişelik girizilmedik bolsa, onda bu bölüm ýazylmaýar.

Tipler bölümi umumy ýagdayda aşakdaky ýaly ýazylýar :

TYPE T1=<tipiň mazmuny >; T2=<tipiň mazmuny>; ..., Tn=<tipiň mazmuny>;

Bu ýerde TYPE – bölümiň ady bolup "tip" diýen manymy aňladýar; T1, T2,...,

Tn – kesgitlenýän tipleriň atlary.

Meselem:

```
TYPE
        TOMUS=(IÝUN, IÝUL, AWGUST);
        MAŞGALA=(KAKA, EJE, DOGAN, JIGI);
VAR
        A: TOMUS; B=MAŞGALA;
```

Üýtgeýän ululyklar umumy ýagdaýda:

VAR X11, X12,..., X1n: T1; X21, X22,..., X2n: T2;..., Xm1, Xm2,...,Xmn: Tm

görnüşde ýazylýar. Bu ýerde VAR - bölümiň ady bolup "üýtgeýän ululyk" diýen manyny aňladýar. X1i (i=1, n) T1 tipe degişli bolan üýtgeýän ululyklaryň atlary;

X2i (i=1, n) T2 tipe degişli bolan üýtgeýän ululyklaryň atlary; Xmi (i=1,n) Tm tipe degişli bolan üýtgeýän ululyklaryň atlary.

Meselem:

```
VAR
        I, J, K: INTEGER; S, T: REAL; H: CHAR; B1,
    B2 : BOOLEAN;
```

Ýazgy I, J, K – üýtgeýän ululyklaryň diňe bitin tipe degişli bolan bahalary; S, T- üýt-

geýän ululyklaryň diňe hakyky tipe degişli bahalary; H-üýtgeýän ululygyň diňe liter tipe degişli bolan bahalary we B1, B2 – üýtgeýän ululyklaryn bolsa diňe logiki tipe degişli bolan bahalary kabul edip bilýändigini aňladýar.

Funksiýalar we proseduralar bölüminde programmada ulanylýan standart däl proseduralar beýan edilýar.Funksiyanyň sözbaşysy aşakdaky ýaly ýazylyar:

FUNCTION<funksiýanyň ady>(q1:T1; q2:T2; ...; qn:Tn): T;

Bu ýerde qi(i=1,n) formal parametler; Ti(i=1,n) degişlilikde qi(i=1,n) parametrleriň tipleri; T- funksiýanyň ýerine ýetirilmeginde alynyan netijäniň tipi . Meselem:

FACTORIAL n!-i hasaplamak üçin düzülen funksiýa bolsa, onda ony funk-
siýalar we proseduralar bölüminde

FUNCTION FACTORIAL (K:INTEGER): INTEGER;

sözbaşy görnüşde ýazylýar. Eger-de programmada prosedura ulanylýan bolsa, onda ol

PROCEDURE <proseduranyň ady> (formal parametrleriň sanawy); görnüşdäki sözba
şy bilen ýazylýar. Formal parametrleriň sanawynda giriş-çykyş parametrleri we olaryň
tipleri görkezilýär.

Meselem:

KWU- kwadrat deňlemäniň hakyky köklerini tapmagyň prosedurasy bol-
sa, onda ol funksiýalar we proseduralar bölüminde aşakdaky ýaly ýazylýar:

PROCEDURE KWU(a, b, c: REAL; VAR X1, X2: REAL);

parametrleri. Çykyş pa-
rametrleri üýtgeýän ululyk hökmünde yglan edilýär. Şonuň üçin olaryň öňünden VAR sözi goýulýar. Proseduranyň adynyň tipi kesgitlenilmeýär, sebäbi ol bu ýerde hiç hili baha eýe bolmaýar.

Operatorlar bölümi BEGIN sözi bilen başlanýar we END sözi bilen gutarýar.Bloguň ahyryny görkerzýän END sözünden soň hökman “.”- nokat belgisi goýulmaly. Bu bölüm programmanyň ýerine ýetirilýän bölümi bolup, ol arasynda “ ; “ –nokatly otur belgisi goýulýan operatorlaryň yzygiderliginden durýar. Operatorlar özleriniň gurluşy boýunça ýönekeý we strukturaly operatorlara bölünýär.

Özünde diňe bir sany operator saklaýan operatorlara ýönekeý operatorlar diýilýär. Meselem: şertsiz geçiş operatory ; eýe bolmak operatory we ş.m.

Özünde birden köp – birnäçe operatory saklaýan operatorlara strukturaly operatorlar diýilýär. Meselem: şertli geçiş operatory ; saýlaw operatory ; sikl operatory we ş.m. OBJECT-PASCAL algoritmik dilinde birnäçe operatorlary birikdirip bir operator hökmünde hem ýazmak bolýar we olara düzümli operatorlar diýilýär. Düzümli opera-tor umumy ýagdaýda

BEGIN P1; P2; ...;Pn END.

görnüşde ýazylýar. Bu ýerde P1; P2; ...;Pn – OBJECT-PASCAL diliniň islendik ope-ratory; hususy ýagdaýda

BEGIN P END ýa-da P;

Başgaça BEGIN we END sözleriň jübütine operatorlar ýaýy hem diýilýär, BEGIN – açylýan ýaýy, END – ýapylýan ýaýy aňladýar. Meselem:

BEGIN Y: =SQR(x)/2; x:=x+1 END

BEGIN K: =5; BEGIN J:=0; AK: =0 END; END

Operatorlar bölümiň özüni hem düzümli operator hökmünde seretmek mümkindir.

Object-pascal dilinde tipler

Skalýar tipler

Tip düşünjesi ähli programmirleýiş dillerinde hem esasy düşünjeleriň biri hasaplanylýar.Tip – dilinde obýektleriň kabul edip biýän bahalarynyň köplügine we şol koplükde kesgitlenen amallaryň toplumyna düşünilýär. OBJECT-

PASCAL algoritmik dilinde tipleri iki uly topara bölmek mümkin: 1) skalýar tipler; 2) strukturaly tipler. Eger obýekt diňe bir sany komponentden ybarat bolsa, onda oňa skalýar tipe degişli diýilýär. Birden köp-birnäçe komponentden ybarat bolan obýektler strukturaly tipe degişlidir.

Şkalýar tipleri hem standart we standart däl skalýar tiplere bölmek bolýar. Standart skalýar tipe INTEGER, BYTE, REAL, CHAR we BOOLEAN tipler degişlidir. Bitin tip (-32768, 32767) aralyga degişli bolan ähli bitin sanlaryň köplügi bilen kesgitlenilýär. Bitin tip

< at, ... > : INTEGER;

görnüşde yglan edilýär. Meselem:

```
VAR
 BAHA, SANAW_N : INTEGER;
 X_KOORD,                    Y_KOORD,
Z_KOORD: INTEGER;
```

Bu ýazgy BAHA, SANAW_N, X_KOORD, Y_KOORD, Z_KOORD – 5 sany üýtgeýän ululyga diňe (-32768,32767) aralykda bolan bitin bahalara eýe bolup bilýändigini aňladýar. Mysal üçin: BAHA:=4.45 eýe bolmak operatorynyň ýazylyşy nädogry. SANAW_N:=295 operator dogry ýazylan. Bitin ululyklar köplenç parametrli sikl operatorlarynda gaytalanmady gurnamak üçin we massiwiň element- lerini indekslemek üçin ulanylýar. Bitin tipdäki bahalar bilen işlenilende amallaryň netijeleri (-32768,32767) diapozonda çykmaly däl. Meselem: 3000 * 6000/3000 görnüşli aňlatmanyň bahasyny hasaplamak üçin, ony 3000*(6000/3000) görnüşinde ýazmalydyr .

BYTE(byte)-tip edil bitin tip ýaly kesgitlenilýär we oňa (0;255) aralykdaky bitin sanlar degişlidir. Bu tipdäki ululyklar üýtgeýän ululyklar bölüminde aşakdaky yaly yglan edilýär:

VAR <ady,…>:BYTE;

Meselem:

```
VAR JEM : BYTE;
MIN,MAX,INDEKS : BYTE;
```

Bu ýazgylarda JEM,MIN,MAX weINDEKS atly ululyklara diňe (0;255) aralykdaky bitin bahalary berip bolyandygyny aňladýar. Arifmetiki aňlatmalarda BYTE we INTEGER tipli ululyklaryň utgaşyp gelmegine-de rugsat edilýär.

Ähli (10^{-38}; 10^{38}) aralykdaky položitel sanlar, olara gapma-garşylyky bolan otrisatel sanlar we 0 san hakyky tipe degişli hasap edilýär. Hakyky tipe degişli bolan üýtgeýän ululyklar aşakdaky görnüşde yglan edilýär:

VAR <ady>: REAL ;

Meselem:

VAR JEM, NETIJE : REAL ;

Ýazgy JEM we NETIJE ütgeýän ululyklaryň diňe hakyky tipe degişli bolan bahalara eye bolup bilýändigini aňladýar. Arifmetiki aňlatmalarda, BYTE , INTEGER we REAL tipli ululyklaryň gelmegine hem rugsat edilýär. REAL tipli ütgeýän ululyklary massiwiň indeksi we köplügiň elementi hökmünde ulanmak bolmayar. Logiki tipe degişli bolan üýtgeýän ululyklar aşakdaky ýaly yglan edilýär:

VAR <ady> : BOOLEAN ;

Meselem:

VAR S, T : BOOLEAN ;

Ýazgy S we T üýtgeýän ululyklaryň logiki tipe degişlidigini aňladyar. Logiki tipe degişli bolan üýtgeýän ululyklar diňe “TRUE” (çyn) ýa-da “FALSE” (ýalan) bahalaryň birden-birini kabul edip bilýär. Meselem:

Eger S: = 12 < 17; we T: = 10 > 14;

Bolsa, onda S- “TRUE”, T- “FALSE” baha eýe bolyar. OBJECT-PASCAL algoritmik dilinde TRUE we FALSE standart logiki hemişelik hökmünde hem ulanylýar.

Liter tipe degişli bolan üýtgeýän ululyklar aşakdaky ýaly yglan edilýär:

VAR <ady> : CHAR ;

Meselem:

VAR HARP, SIMWOL : CHAR ;

Ýazgy HARP we SIMWOL üýtgeýän ululyklaryň diňe liter bahany kabul edip bilýändigini aňladýar.

OBJECT-PASCAL- da liter baha diýlende diliň elipbiýine girýän islendik simwola - latyn harplaryna, onluk sanlara, amallaryň belgilerine, ýörite belgilere we ş.m. düşünilýär. Meselem:

HARP: = ‘ M ‘ ;
SIMWOL : = ‘ * ‘ ;

Ýa-da

SIMWOL : = ‘ ‘ – boş öýjük we ş.m.

CHAR tipe degişli bolan üýtgeýän ululyklary arifmetiki aňlatmalarda ulanmak rugsat edilmeýär. OBJECT-PASCAL algaritmik dilinde CHAR tipe degişli bolan üýtgeýän ululyklary aşakdaky ýaly deňeşdirmek bolýar:

‘k‘ > ‘b’ – “TRUE” baha eýe, sebäbi maşyn elipbiýinde ‘k’ harpy ‘b’ – deň soň gelýär. Şoňa göräde liter ululyklaryň üstinde CHR, ORD, PRED we SUCC fuksiýany ýerine ýetirip bolýar.

OBJECT-PASCAL algoritmik dilinde ulanyjynyň özüne hem tip kesgitlemäge rugsat edilýär. Onuň ýaly tiplere standart däl tipler ýa-da ulanyjynyň tipi diýilýär. Standart däl tipleriň elementi sanalyp geçilýän we çäklendirilen tipleri bolup olarda elementleriň sany 256- dan geçmeli däl.

Standart däl skalýar tipleri girizmeklik ýazylýan programmany sadalaşdyrýar we maşynyň ýadyny tygşytlamaga mümkinçilik berýär.

Elementleri sanalyp geçilýän tipi aşakdaky ýaly yglan edilýär:

TYPE < tipiň ady > = (< 1- nji element, 2- nji element, … , n- nji element >) ;

VAR < ady > : < tipiň ady > ;

Meselem :

```
TYPE
    HEPDE = ( 61, 62, 63, 64, 65, 66, BAZAR);
```

FAKULTET = (FIZ, MAT, HIM, TAR, BIOL, GEOGR) ;

TOPAR = (G101, G102, G103, G104) ;

Beýle ýagdaýda onuň tipi girizilýär, emma ady kesgitlenilmeýär. Bu ýerde ýaýyň içinde ýazylan bahalara görkezilen tipe degişli bolan hemişelikler hökmünde hem seretmek bolýar. Şonuň üçin hem bu bahalar ulanylanda hemişelikler bilen işlemegiň düzgüninden ugur almaly. Elementleri sanalyp geçilýän tip tertipleşdirilen tiplere degişlidir, ýagny onyň elementlerini deňeşdirip bolýar. Meselem:

G102 < G104 – "TRUE" baha eýe, G103 < G101 bolsa "FALSE" baha eýedir. Diýmek, bu tipe degişli bolan bahalary SUCC, PRED, ORD standart funksiýalarda hem ulanyp bolýar.

Bellik: Elementleri sanalyp geçilýän tipe degişli bolan bahalary giriş – çykyş operatorlarynda ulanmak bolmaýar.

Çäklendirilen tipli üýtgeýän ululyklar aşakdaky ýaly yglan edilýär :

TYPE < tipiň ady > = m1 … m2 ;

VAR < üýt. ulul. ady > : < tipiň ady > ;

Bu ýerde m1 we m2 hemişelikler, degişlilikde berlen tipe degişli bolan ütgeýän ululyklaryň kabul edip biljek iň kiçi we uly bahalaryny görkezýärler. M1 we M2 hemişelikler REAL tipe degişli bolmaly däl, CHAR, BOOLEAN, INTEGER tipleriň isledigine degişli bolup biler. Meselem:

```
TYPE    DAY = 1.. 31 ;
        Harp = ‘ a ‘ .. ‘ e ‘ ;
VAR
        WORKING-DAYS, FREE- DAYS ;
        X1, X2 : harp ;
```

Ýazgy WORKING-DAYS, FREE-DAYS- atly üýtgeýän ululyklaryň diňe 1-den 31-e çenli aralykdaky bitin bahalara eýe bolýandygyny, X1 we X2 üýtgeýän ululyklaryň bolsa, diňe ‘ a ‘, ‘ b ‘, ‘ c ‘, ‘ d ‘ we ‘ e ‘ simwollaryň birden-birine eýe

bolýandygyny aňladýar. Köpleç çäklendirilen tip aşakdaky görnüşde yglan edilýär:

```
CONST MIN = 1; MAX = 31;
TYPE
        DAY : = MIN .. MAX ;
VAR
        WORKING- DAYS, FREE- DAYS :
DAY ;
```

Beýan edilen ýagdaýda interwalyň çäklerini çalyşmak üçin diňe hemişelikler bölümini üýtgetmek ýeterlik bolýar.

Maglumatlaryň täze tipini kesgitlemek
Strukturaly tipler barada düşünje

Strukturaly tipler skalýar tipe degişli bolan üýtgeýän ululyklaryň toplumy hökmünde kesgitlenilýär. Strukturaly tip özüniň komponentleriniň tipleri bilen häsiýetledirilýär. OBJECT-PASCAL algoritmik dilinde ulanylýan strukturaly tiplere: setir tipi; köplükleri kombinirlenen tipi; massiwleri; faýllary degişli etmek mümkin.

Diliň elipbiýine girýän simwollaryň tükenikli yzygiderligine setir hemişeligi diýilýär. Setir hemişelikleri hem edil liter hemişlikleri ýaly “ ‘ “- anastrof belgisiniň içinde ýazylýar. Meselem:

‘OBJECT-PASCALI’, ‘Y=’, ‘OBJECT-PASCAL’ we ş.m.

Baha hökmünde diňe setir hemişelikleri kabul edip bilýän üýtgeýän ululyklara setir üýtgeýän ululyklary diýilýär.

Setir tipi TYPE bölüminde STRING kömekçi sözüniň kömegi bilen yglan edilýär.

Massiw diýlende şol bir tipden bolan birnäçe üýtgeýän ululyklyklaryň top-
lumyna düşünilýär. Massiwi düzýän üýtgeýän ululyklaryň yzygiderligi tükenikli hem-de tertipleşdirilen bolmaly.

Massiwiň her bir elementine ayratyn indeks degişli edilýär we oňa şol indeks boýunça ýüzlendip bolýar.

Massiw ARRAY kömekçi sözi arkaly yglan edilýär. Her bir setir üýtgeýän ululyga elementleri CHAR tipden bolan massiw hökmünde seretmek mümkin.

Köplük diýlende bir bitewi zat hökmünde seredip bolýan şol bir häsiýet, nyşanlar boýunça ýygnalan obýektleriň toplumyna düşünilýär. OBJECT-PASCAL algoritmik dilinde köplüge matematikada ulanylýan köplüge seredeňde has dar manyda düşünilýär. Mysal üçin, OBJECT-PASCAL-da köplügiň elementi hökmünde REAL tipe degişli bolan bahalary ulanyp bolmaýar; köplügiň elementleriniň sany 256-dan geçmeli däl; we ş.m. Köplükler TYPE bölüminde SET kömekçi sözi arkaly yglan edilýär.

Kombinirlenen tip - bu dürli tiplerden bolan üýtgeýän ululyklaryň toplumydyr.

Bu tipi yglan etmek RECORD(ýazgy) kömekçi sözi bilen başlanýar we END sözi bilen hem gutarýar. Kombinirlenen tip yglan edilende onuň her bir komponentiniň
ady we tipi görkezilýär.

Faýl - bu şol bir tipden bolan komponentleriň toplumydyr. Faýl massiwden tapawutlanyp, onuň komponentleriniň sany öňünden görkezilmeýär. OBJECT-PASCAL-da köplenç faýlyň komponenti hökmünde kombinirlenen tipli bahalar ulanylýar. Faýl FILE kömekçi sözüň kömegi bilen yglan edilýär. Adatça faýlyň komponentleri magnit disklerinde saklanylýar we gerek wagty ondan operatiw ýada çagyrylýar.

16-lyk hasaplaýyş ulgamyndaky sany 2-lik ulgamyna geçirýän programma

```
uses crt;
label 1;
var
  fi,fo : text;
```

```
  c : char;
begin
clrscr;
      assign(fi,'texthex.txt'); reset(fi);
      assign(fo,'textbin.txt'); rewrite(fo);
      while not eof(fi) do
       begin(*1*)
           read(fi,c);
{            if (c=' ') then goto 1;}
             if (c=' ') then writeln(fo);
           case c of
             '0': write(fo,'0000');
             '1': write(fo,'0001');
             '2': write(fo,'0010');
             '3': write(fo,'0011');
             '4': write(fo,'0100');
             '5': write(fo,'0101');
             '6': write(fo,'0110');
             '7': write(fo,'0111');
             '8': write(fo,'1000');
             '9': write(fo,'1001');
             'A': write(fo,'1010');
             'B': write(fo,'1011');
             'C': write(fo,'1100');
             'D': write(fo,'1101');
             'E': write(fo,'1110');
             'F': write(fo,'1111');
           end;
 1:
       end;(*1*)
    close(fi); close(fo);
end.
```

16-lyk hasaplaýyş ulgamyna geçirýän programma

```
uses crt;
const mas : array[0..15] of char=
     ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');
label 0,1,2,3;
var
  a : array[0..15] of byte;
  n   : longint;
  m,i,j : integer;
  ch : char;
  b : byte;
  f : text;
BEGIN
clrscr;
      assign(f,'texthex.txt'); append(f); writeln(f);
0:    ch:=readkey; write(ch);
      b:=ord(ch);
      if b=13 then begin writeln; writeln(f); goto 0; end;
      if b<=31 then goto 3;
      n:=b;
      if n<=15 then begin write(mas[n]); goto 2; end;
      i:=0;
1:    i:=i+1;
      a[i]:=n mod 16;
      m:=n div 16;
      if m>=16 then begin n:=m; goto 1; end;
      i:=i+1; a[i]:=m;
   for j:=i downto 1 do write(f,mas[a[j]]);
2:
{    write(f,' ');}
   goto 0;
3:
   close(f);
END.
```

Geçiş operatorlary

Eger-de algoritmde käbir şertiň ýerine ýetýändigine ýa-da ýetmeýändigine baglylykda hasaplaýyş prosesi iki ýa-da birnäçe şaha bölünýän bolsa, onda onyň ýaly algoritmlere şahalanýan gurluşly aigoritmler diýilýär.

Programmanyň içinde geçiş operatorlary esasan şahalanýan gurluşy algoritmleri programmirlemek üçin hyzmat edýärler. Başgaça aýdylanda, geçiş operatorlary programmadaky operatorlaryň ýerine ýetiriliş tertibini dolandyrmak üçin hyzmat edýärler.

TURBO-PASCAL algoritmik dilinde geçiş operatorlarynyň iki görnüşi ulanylýar :

1) şertsiz geçiş operatory;
2) şertli geçiş operatory;

Şertsiz geçiş operatory programmanyň bir böleginden beýleki bölegine birnäçe operatordan sowulyp geçmäge mümkinçilik berýär. Ol umumy ýagdaýda aşakdaky görnüşde ýazylýar:

GOTO <belgi>

Bu ýerde GOTO-operatoryň ady bolup , “geçmek’’ diýen manyny aňladýar; <belgi>-salgylanylýan operatoryň belgisi, ol LABEL bölüminde yglan edilýän bolmaly. Meselem:

```
…….
LABEL 20;
…...GOTO 20;
…….
20:  X:=X+1;
…….
```

Sertli geçiş operatory özünde käbir logiki aňlatmany saklap, onuň “TRUE” ýa-da “FALSE” bahalaryň haýsysyny kabul edýändigine baglylykda iki şahadan birini saýlap almaga mümkinçilik berýär.

TURBO – PASCAL algoritmik dilinde şertli geçiş operatory aşakdaky görnüşde ýazylýar:

a) IF < şert > THEN S1 ELSE S2;
b) IF < şert > THEN S;

Bu ýer-de IF – operatoryň ady bolup “eger” diýen manyny aňladýar; < şert > - käbir logiki aňlatma; S1, S2, S – TURBO-PASCAL algoritmik diliniň islendik operatorlary.

Bu operatorlar ýerine ýetirilende ilki bilen <şert>-logiki aňlatmanyň bahasy hasaplanylýar. Eger logiki aňlatma “TRUE” baha eýe bolsa, onda a)-ýagdaýda S1-operator; b)-ýagdaýda bolsa S1-operator ýerine ýetirilýär. Onda sňra IF operatoryň yz ýanyndaky ilkinji operatora geçilýär. Eger-de logiki aňlatma “FALSE” baha eýe bolsa, onda a)-ýagdaýda S2-operator ýerine ýetirilýär we ondan soňra If operatorynyň yzyndaky ilkinji operatora geçilýär; b)-ýagdaýda bolsa, hiç-hili operator ýerine ýetirilmezden, gönüden-göni IF operatorynyň yzyndaky ilkinji operatora geçilýär. TURBO-PASCAL algoritmik dilinde bir If operatorynyň düzüminde başga-da IF operatorynyň ulanylmagyna rugsat edilýär. Bu şahalanmanyň içinde ýene-de şahalanma ulanylyan ýagdaýynda ýüze çykýar. Meselem

```
. . .
IF X<15 THEN IF X>10 THEN WRITELN (‘ 10<X<15’)
                ELSE WRITELN (‘X<=10’)
```

IF operatoryny biri-biriniň içinde ulanmaklyk programma düşünmekligi kynlaşdyrýar. Şoňa görä-de IF operatoryny bir-biriniň içinde iki-üç gezekden köp ulanmak maslahat berilmeýär.

Mysallar.

1. Funksiýanyň bahasyny hasaplamak üçin programma düzmelili:

a) $Y = \begin{cases} \sqrt{X} \; egerX \geq 0 \\ X^3 \; egerX \prec 0 \end{cases}$

Programmasy:

```
Program funksiya;
Uses crt;
```

```
Var x,y:real;
Begin
Clrscr;
Write('x=');
Readln(x);
If x>=0 then y:=sqrt(x)
Else y:=sqr(x)*x;
Writeln('y=',y:8:3);
End.
```

b) $S=\begin{cases} at+b, & \text{eger } at<1; \\ \cos(at), & \text{eger } at=1; \\ e^{-at}\cos(at), & \text{eger } at>1; \end{cases}$

Programmasy:

```
Program funk1;
Uses crt;
Var a,b,t,s:real;
BEGIN
Clrscr;
Write('a,b,t bahasyny giriz');
Readln(a,b,t);
IF A*T<1 then S:=A*T+B
Elseif  A*T=1 then S:=cos(A*T)
Else S:=exp(-A*T)*cos(A*T);
Writeln('S=',S:8:3);
End.
```

Bellik:

Birnaçe IF operatory biri-biriniň içinde ulanylanda her bir ELSE kömekçi sözi özüniň öň yanyndaky THEN kömekçi sözüne degişli edilyär.

2) Kwadrat deňlemäniň hakyky köklerini tapmak üçin programma düzmelili:

$AX^2+BX+C=0;\ A=0.$

Programmasy ;

```
PROGRAM KWADRATD;
Uses CRT;
Var A,B,C,D,X1,X2 : real;
BEGIN
CLRSCR;
Write('A,B,C bahasyny giriz');
Readln(A,B,C);
D:=SQR(B)-4*A*C;
IF D<0 then begin Writeln('hakyky köki yok'); END
ELSE begin
X1:=(-B+SQRT(D))/(2*A);
X2:=(-B-SQRT(D))/(2*A);
Writeln('X1=',X1,'X2=',X2);
end;
END.
```

2-niň n-nji derejesini hasaplamak üçin programma; bu ýerde n ululyk 1<=n<=1000 aralykdaky bahany kabul edip bilýär:

```
(*2-nin 1000-njiderejesi
     11.10.2005(2)                *)
uses crt;
var
  x : array[1..1000] of 0..9;
  i,j,k,p,xb,b,n : integer;
begin
clrscr;
      write('n='); read(n);
      x[1000]:=1; xb:=1000;
      for k:=1 to n do
       begin
           p:=0;
           for i:=1000 downto xb do
```

```
            begin
                b:=2*x[i];
                b:=b+p;
                x[i]:=b mod 10;
                p:=b div 10;
            end;
        if p<>0 then
                begin
                    dec(xb); x[xb]:=p;
                end;
        end;
       for i:=xb to 1000 do write(x[i]);
readln; readln;
end.
```

Kömekçi programmalar

Käbir meseleler çözülende parametrleriň dürli bahalarynda şol bir algoritmi birnäçe gezek gaýtalap ýerine ýetirmeli bolýar. Beýle ýagdaýda programmanyň şol algoritme degişli bölegini aýratyn bölüp alyp, oňa gerek wagty programmanyň islendik böleginden ýüzlenmek mümkindir.

Programmanyň özbaşdak programma birligi hökmünde ulanyp bolýan bölegine bölek programma diýilýär. OBJECT-PASCAL algoritmik dilinde bölek programmanyň iki görnüşi ulanylýar:

1. Funksiýalar; 2). Proseduralar

Eger hasaplama ýerine ýetirilenden soň alnan netije diňe bir sany baha bolsa, onda onuň ýaly hasaplamany funksiýa görnüşinde ýazyp bolýär. Funksiýa hem üç bölekden durýar:

1. Sözbaşy 2. Yglan ediş bölümi 3. Funksiýanyň göwresi

Funksiýa umumy görnüşde aşakdaky ýaly ýazylýar:

FUNCTION F(q1:T1;q2:T2;…,qn:Tn):T; {Funksiýanyň sözbaşy)
<Yglan ediş bölümi>
BEGIN
P1;
P2;
…{Funksiýanyň göwresi}
F:=…
END;

Bu ýerde F-funksiýanyň ady; $q_i\,(i=1,n)$ - formal parametrleriň atlary; $T_i\,(i=1,n)$-degişlilikde olaryň tipleri; T- ol F funksiýanyň adynyň tipi ýa-da başgaça netijäniň tipi; P1, P2,…-funksiýanyň göwresini emele getirýän operatorlar. Yglan ediş bölüminde diňe funksiýanyň çäginde ulanylýan parametrler we olaryň tipleri görkezilýär. Funksiýa özünde hemişelikler bölümini, belgiler bölümini, tipler bölümini, üýtgeýän ululyklar bölümini, kömekçi funksiýalary we proseduralary saklap biler. Eger formal parametrleriň birnäçesi şol bir tipe degişli bolsa, onda olary toparlar, tipi diňe bir gezek ýazmaklyga rugsat edilýär. Meselem:

FUNCTION FAKT(K1,K2,K3 : INTEGER) : REAL;

Funksiýa programmanyň islendik ýerinden ýüzlenmek üçin eýe bolmak operatorynyň sag tarapyndaky aňlatmada onuň adyny we ýaýyň içinde formal parametrleriň iş ýüzündäki bahalaryny görkezmek ýeterlikdir.
F(b1,b2,…,bn);
bu ýerde F-funksiýanyň ady; b1,b2,…,bn – degişlilikde q1,q2,…,qn – formal parametrleriniň iş ýüzündäki bahalary. Formal parametrleri bilen onuň iş ýüzündäki bahalarynyň sany deň bolmaly we olar degişlilikde tipleri boýunça gabat gelmeli.
Mysal:

$C_n^m = \frac{n!}{m!(n-m)} \{n, m \varepsilon N, n > m\}$ formula boýunça utgaşmalaryň sanyny hasaplamak üçin programma düzmelili. Faktorial hasaplamagy bölek programma – funksiýa görnüşinde ýazmaly.

```
PROGRAM UTGSANY;
USES CRT;
VAR CNM : REAL;
      N,M,L : INTEGER;
      FUNCTION FACT(K : INTEGER;) : INTEGER;
                  VAR P,I : INTEGER;
BEGIN
   P:=1;
   FOR I:=1 TO K DO P:=P*I;
   FACT:=P
   END;
BEGIN
   CNM:=FACT(N)/(FACT(M)*(FACT(N-M));
   WRITE('Utgaşmanyň sany=',CNM:8:2);
END.
```

Eger-de algoritm ýerine ýetirilenden soň alynýan netije birden köp – birnäçe baha bolsa, onda bölek programmanyň prosedura görnüşinden peýdalanylýar. Proseduralar umumy görnüşde aşakdaky ýaly ýazylýar:

```
PROCEDURA F(formal parametrleriň sany);
{Proseduranyň sözbaşy}
<Yglan ediş bölümi>
{bu bölümde lokal parametrler yglan edilýär}
BEGIN
   P1;
   P2;
   …{Operatorlar bölümi}
   Pn;
```

END;

Bu ýerde F-proseduranyň ady, funksiýanyň adyndan tapawutlanyp, ol bu ýerde hiç-hili baha eýe bolmaýar. Şoňa görä-de onuň üçin tip kesgitlenmeýär; Formal parametrleriň sanawynda prosedura ýüzlenmek üçin gerek bolan parametrler we olaryň tipleri görkezilýär; P1,P2,…,Pn – proseduranyň göwresini emele getirýän operatorlar. Prosedura aşakdaky ýaly ýüzlenilýär.

F(b1,b2,…,bn);

Bu ýerde b1,b2,…,bn –formal paametrleriň iş ýüzündäki bahalary; edil funksiýadaky ýaly, bu ýerde hem formal parametrler bilen onuň iş ýüzündäki bahalarynyň sany gabat we olar degişlilikde şol bir tipe degişli bolmalydyr.

Mysallara seredeliň:

1) Sh $x=e^{x}-e^{-x}/2$ prosedurany ulanyp $z= Sh^{2}a+ Sh(a-b)/Sha+SQR(a^{2}-b^{2})$ hasaplamak üçin programma düzmelili.

Programmasy:

```
PROGRAM PROSEDUR;
USES CRT;
VAR  A,B, Z,T1, T2, T3:REAL;
 PROSEDURE  SH (X: REAL; VAR R: REAL );
BEGIN
        R:= (EXP(X) – EXP(-X))/2.0
END;
BEGIN CLRSCR;
                WRITE(‘A we B bahasyny giriz ‘);
                READ (A, B);
                SH (A, T1);
                SH ( A-B,T2);
                SH(A*A-B*B,T3);
                Z:=(T1*T1+T2)/(T1+SQRT(T3));
                WRITE(‘Z=’,Z:12:4);
        END.
```

Eger prosedurada we baş programmada şol bir atly parametrler ulanylỳan bolsalar, onda prosedura yglan edilende formal parametrleri görkezmek hökman däl.

2) Goỳ tekizlikde N-sany nokat özleriniň gönüburçly koordinatalary bilen berlen bollsun:

(Xi,Yi),Xi>0, i=1, N

Bu nokatlaryň polỳar koordinatalaryny kesgitlemegi parametrsiz görnüşde ỳazmaly:

$R=(x^2+y^2)$;
Tg(f) =y/x;
F=arctg(y/x).

Programmasy:

```
PROGRAM POLKORD;
USES CRT;
    VAR X,Y,R,F:REAL;
    PROCEDURE POLAR;
    BEGIN
        R:=SQRT(SQR(x)+SQR(y));
        F:=ARCTAN(Y/X);
        END;
        BEGIN
        CLRSCR;
        WRITELN('N=');
        READ(N);
        FOR I:=1 TO N DO
        BEGIN
        WRITELN('X,Y bahasyny giriz');
        READ(x,y);
        POLYAR;
        WRITELN('R=',R:8:3,'  F=', F:8:3);
        END;
    END.
```

Funksiýada we prosedurada formal parametrleri islendik tertipde ýerleşdirmek mümkin.Bölek programma ýüzlenilen mahalynda formal parametrler nähili tertipde ýerleşdirilen bolsa, olaryň iç ýüzündäki bahalary hem şol tertipde ýerleşdirilen bolmaly.

Prosedurada formal parametrleriň iki görnüşini bir-birinden tapawutlandyrmak gerek:

1) Baxa hökmünde ulanylýan parametrler;
2) Üýtgeýän ululyk hökmünde ulanylýan parametrler;

Baha hökmünde ulanylýan parametrler proseduranyň ýerine ýetirilmegi üçin zerur bolan başlsngyç maglumatlary-berlenleri kesgitlemek üçin hyzmat edýärler.Olar formal parametrleriň sanawynda:

(q1:t1;q2:t2;…) ýa-da (q1,q2:t;…)

görnüýazulýar. Bu yerde q1,q2,…-parametrleriňbahalary, hemişelik, üytgeýän ýa-da aňlatma görnüşde berlip bilner.

Üýtgeýän ululyk hökmünde ulanylýan parametrler adatça prosedura ýerine ýetirilenden soň, alnan netijelere eýe bolmak üçin hyzmat edýärler. Proseduradan alnýan maglumatlar şol ütgeýän ululyklar arkaly esasy programma berilýär.Ütgeýän ululyk hökmünde ulanylýan parametrler formal parametrleriň sanawynda asakdaky görnüşde ýazylýar:

(…; VAR q1 : T1, q2 : T2, …, q_n : T_n) ;

Bu ýerde q1, q2,…,q_n- üýtgeýän ululyk hökmünde ulanylýan parametrleriň atlary; T1, T2,…,T_n- degişlilikde olaryň tipleri: OBJECT-PASCAL algoritmik dilinde bir bölek programmanyň içinde ýene-de şol bölek programmanyň özüne ýüzlenmäge rugsat edilýär. Bölek programma ýüzlenmekligiň bu

görnüşine rekursiw ýüzlenme diýilýär. Köplenç metematiki formulalary rekursiýa görnüşde ýazyp bolýar

Mysal :

Rekursiw ýüzlenmäni ulanyp k!-y hasplamak üçin bölek programma düzmelili. FUNCTION FAKTOR (K: INTEGER): INTEGER;
BEGIN.
IF K=1 THEN FAKTOR:=1 ELSE FAKTOR:=FAKTOR (K-1) *K END;
Programmanyň islendik ýerinde n! hasaplamak üçin
nfaktor:= fakt (n);
operatory ýazmak ýeterlikdir.

Kä ýagdaýlarda bir programmada beýleki bir entek yglan edilmedik bölek programmany çalyşmaly bolýar. Onuň üçin şol çagyryljak bölek programmanyň söz başyçynyň yzyndan FORMARD-kömekçi sözüni ýazmaly. Bu ýagdaýy aşakdaky mysalyň üsti bilen düşündürmek mümkin:

```
PROGRAM  A5;
…
VAR  X,Y: real;
PROCEDURE P1(A:REAL); FORWARD;
PROCEDURE P2(B:REAL);
BEGIN
…
P1(X)
END;
        PROCEDURE P1;
        …
        BEGIN
        P2(Y)
        END;
```

```
BEGIN
P2(X);
P1(Y);
…
END.
```

Prosedura bölek programma – funksiýadan tapawutlanyp, ol birden köp-birnäçe netijäni dolandyryp, kä ýagdaýlarda bolsa, hiç-hili netijäni dolandyrman hem biler. Meselem:

```
PROGRAM  A6;
ESES CRT;
VAR I: BYTE;
        PROCEDURE LINIA (N: INTEGER);
        VAR
                I: INTEGER;
        BEGIN
        CLRSCR;
                FOR J=1 TO N DO
                WRITE( '-') END;
        BEGIN
        FOR  I:=1 TO 6 DO
        BEGIN  LINIA (I);  WRITELN; END
        END.
```

Görnüşi ýaly, bu ýerde her gezek bölek programma ýüzlenilende yzyna netijä dolanylmaýarda, diňe käbir operatorlaryň toplumy ýerine ýetirilýär. Programmada WRITELN operatory her gezek täze setire geçmek üçin hyzmat edýär. Prosedura siklde 6-gezek ýüzlenilende aşakdaky netije alynýär:

```
—
— —
— — —
— — — —
— — — — —
```

— — — — — —

Küşt tagtasynda at-yň her öýe bir gezek barmak bilen bütün küşt tagtasyny aýlanyp çykýan programmasy. At-yň başlangyç ýagdaýy girizilýär:

```
type m = array[1..64] of byte;
const
     x : m = (1, 3, 5, 7, 8, 7, 8, 6, 4, 2, 1, 3, 1, 3, 5, 7,
              8, 7, 8, 6, 4, 2, 1, 2, 3, 1, 2, 1, 2, 4, 6, 8,
              7, 8, 7, 5, 6, 8, 6, 4, 2, 1, 2, 1, 3, 5, 7, 8,
              6, 5, 7, 5, 3, 4, 5, 3, 4, 6, 4, 6, 5, 4, 3, 2 );

     y : m = (1, 2, 1, 2, 4, 6, 8, 7, 8, 7, 5, 6, 7, 8, 7, 8,
              6, 4, 2, 1, 2, 1, 3, 5, 7, 8, 6, 4, 2, 1, 2, 1,
              3, 5, 7, 8, 6, 7, 8, 7, 8, 6, 4, 2, 1, 2, 1, 3,
              4, 6, 5, 4, 3, 5, 3, 4, 6, 5, 4, 3, 5, 3, 5, 3 );
Var
a:array[1..8,1..8] of byte;
t,x0,y0,k,i,j:integer;
begin
clrscr;
write('x,y=');readln(x0,y0);
k:=0;
for i:=1 to 64 do
if (x[i]=x0) and(y[i]=y0) then begin t:=i; break; end;
  for i:=t to 64 do begin k:=k+1; a[x[i],y[i]]:=k; end;
  for i:=1 to t-1 do begin k:=k+1; a[x[i],y[i]]:=k; end;

WRITELN('Atyn gochumleri');
for i:=1 to 8 do  begin writeln;
for j:=1 to 8 do write(a[i,j]:4,'  ');
   end;
readln;
```

```
end.

{   LABIRINT    programmasy }
Uses crt;
Label 1,2,3,4,5;
Var
b,a:array[0..100,0..100] of integer;
wariant,m,s,i1,j1,r,t,k,l,n,i,j:integer;
x,y:array[1..100] of integer;
Function Polojenie(i,j:integer):boolean;
Var
i1,j1,p:integer;
begin
p:=0;
for i1:=i-1 to i+1 do
for j1:=j-1 to j+1 do
if (a[i1,j1]=1)and((i1<>i)or(j1<>j)) then p:=p+1;
if p=0 then Polojenie:=False else Polojenie:=True;
end;

begin
clrscr;
randomize;
write('n='); read(n);
readln(Wariant);

if wariant=1 then
for i:=1 to n do
for j:=1 to n do     begin
write('a[',i,',',j,']='); read(a[i,j]); end
                                  else

for i:=1 to n do
for j:=1 to n do
a[i,j]:=random(2);
```

```
for i:=0 to n+1 do a[i,0]:=-5;
for i:=0 to n+1 do a[i,n+1]:=-5;
for j:=0 to n+1 do a[0,j]:=-5;
for j:=0 to n+1 do a[n+1,j]:=-5;

read(k,l);

for i:=1 to n do begin   writeln;
for j:=1 to n do begin b[i,j]:=a[i,j]; write('  ', a[i,j]); end;end;
                  writeln;

a[k,l]:=2;
t:=2;
m:=1;
1: for i:=1 to n do
if a[i,1]=t then  begin x[1]:=i; y[1]:=1;goto 5; end;

for i:=1 to n do
if a[i,n]=t then  begin x[1]:=i; y[1]:=n;goto 5; end;

for j:=1 to n do
if a[1,j]=t then begin x[1]:=1; y[1]:=j; goto 5;end;

for j:=1 to n do
if a[n,j]=t then begin x[1]:=n; y[1]:=j; goto 5;end;
                  {  Переход путника}
r:=0;
for i:=1 to n  do
for j:=1 to n  do
if (a[i,j]=t)and polojenie(i,j) then
for i1:=i-1 to i+1 do
for j1:=j-1 to j+1 do
if (a[i1,j1]=1)and((i1<>i)or(j1<>j)) then begin r:=r+1;
```

```
a[i1,j1]:=t+1; end;

if r>0 then begin t:=t+1; goto 1 end else begin write('Çykalga
ýok'); goto 3;end;


 m:=1;

5:writeln('Çykalga ýok');
While  m<=t-2  do   begin

for i:= x[m]-1 to x[m]+1 do
for j:= y[m]-1 to y[m]+1 do
if (a[i,j]=t-m)and((i<>x[m])or(j<>y[m])) then begin m:=m+1;
 x[m]:=i; y[m]:=j;  goto 4;
 end;
           4:       end;
textcolor(White);
for i:=1 to n do begin   writeln;
for j:=1 to n do
 if (i=k)and(j=l) then  begin textcolor(Green);
             write('  ',b[i,j]); textcolor(White);  end
             else
 begin
         s:=0;
         for i1:=1 to t-2 do
         if (i=x[i1])and(j=y[i1]) then s:=s+1;
        if s=0 then write('  ',b[i,j]) else begin textcolor(Red);
             write('  ',b[i,j]); textcolor(White);  end;
             end;

             end;
readln;
readln;
3:end.
```

Uly programmalar adatça kömekçi programmalara dargadylyp ýazylýar. Her bir kömekçi programma belli bir işi ýerine ýetirmek üçin niýetlenendir.

Kömekçi programmanyň özüniň-özüne ýüzlenmek hadysasyna rekursiýa diýilýär. Rucursiýa kyn meseleleri ýeňillik bilen çözmeklige mümkinçilik berýär. Ýöne stack bilen baglanyşykly bolany üçin örän häzir bolup ulanylmalydyr.

Recursiýany ulanmak bilen girizilen 10-lyk sany ikilik sana öwürýän programma:

```
var
  x : array[1..10] of 0..1;
  i,k,n : integer;
begin
clrscr;
     write('n='); read(n);
     k:=0;
     repeat
          inc(k); x[k]:=n mod 2;
          n:=n div 2;
     until n=0;
     for i:=k downto 1 do write(x[i]);
readkey;
end.
```

Recursiýany ulanmak bilen faktorialy hasaplamaklygyň programmasy:

```
uses crt;
{-----------------------------------}
function f(k:integer):integer;
begin
   if k=0 then f:=1
         else f:=k*f(k-1);
```

```
end;
{-----------------------------------}
BEGIN
clrscr;
      write(f(2));
readln;
END.
```

Aşakdaky programma rekursiýany ulanmak bilen “Salam” sözüni üç gezek çapa çykarýar:

```
uses crt;
{-----------------------------------}
procedure p(k:integer);
begin
    if k>0 then p(k-1);
    writeln(k,'. Salam');
end;
{-----------------------------------}
BEGIN
clrscr;
      p(3);
readln;
END.
```

Merseniň sanyny hasaplaýan programma:

```
var
  n,i,p : word;
{-----------------------------------}
function ys(m : word):boolean;
var i : word;
begin
     ys:=true;
     for i:=2 to m div 2 do
      if m mod i = 0 then ys:=false;
```

```
end;
{-----------------------------------}
BEGIN
CLRSCR;
        write('n='); read(n);
        for i:=2 to n do
          if ys(i) then
            begin
                 for p:=1 to n do
                  if (ys(p)) and (p<37) then
                              if trunc(exp(p*ln(2))-1) = i
                                 then writeln('Chislo Mersana=',i,';
p^=',p);
             end;
READLN; READLN;
END.
```

Kömekçi programmalara ýüzlenilende adatça olara belli bir maglumatlar berilýär we netijesi çagyrýan programma gaýtarylyp berilýär. Ol maglumat alşylyp çalşygy parametrleriň üsti bilen amala aşyrylýar. kömekçi programmanyň özi hem parametr görnüşinde ulanylyp biliner. Aşakda getirilýän programmada functiony parametr görnüşinde ulanmaklyga seredilýär:

```
uses crt;
var
  a,b : integer;
{---------------------------------------}
{iki sanyň jemini tapýan function}
function jem(p1,p2 : integer):integer;
begin
   jem:=p1+p2;
end;
{---------------------------------------}
{iki sanyň tapawudyny tapýan function}
```

```
function tap(p1,p2 : integer):integer;
begin
    tap:=p1-p2;
end;
{---------------------------------------}
{iki sanyň köpeltmek hasylyny tapýan function}
function kop(p1,p2 : integer):integer;
begin
    kop:=p1*p2;
end;
{---------------------------------------}
begin
clrscr;
       write('a='); read(a);
       write('b='); read(b);
{Bu ýerde parametr görnüşinde functionlaryň atlary ulanylýär}
       writeln(kop(jem(a,b),tap(a,b)));
readkey;
end.
```

Islendik yylyn kalendaryny hasaplaýan programma:

```
{ Oraz_E_N, 25.09.2005(0)}
uses crt,dos;
var ay,a,b,n : array[1..12] of byte;
   y : longint;
   i : byte;
{---------------------------------------}
function f(k:longint) : boolean;
begin
  f:=(k mod 4=0) and (k mod 100 <> 0) or (k mod 400=0);
end;
{---------------------------------------}
function c0(m:byte; y:longint) : byte;
(* islendik yylyn islendik ayynyn ilkinji gununin hepdanin
haysy
```

```
   gununе dushyanligini kesgitleyan programma*)
var
i : longint; s : byte;
begin
 s:=0;
 for i:=1 to y-1 do
  if f(i) then s:=(s+366) mod 7 else s:=(s+365) mod 7;
 for i:=1 to m-1 do s:=(s+ay[i]) mod 7;
 s:=(s+1) mod 7;
     if s=0 then s:=7;
     c0:=s;
end;
{----------------------------------------}
procedure c(x,y,h,m,m1 : byte);
(* x,y-koordinata, y-yyl,  *)
var
    mas1 : array[1..49] of byte;
    mas2 : array[1..7,1..7] of byte;
    mas3 : array[1..7] of string[3];
    mas4 : array[1..12] of string[18];
    a,b,i,j,k : byte;
    s,ss : string;
begin
        mas3[1]:='Mo'; mas3[2]:='Tu'; mas3[3]:='We';
mas3[4]:='Th';
        mas3[5]:='Fr'; mas3[6]:='Sa'; mas3[7]:='Su';

        mas4[1]:='January';   mas4[2]:='February';
        mas4[3]:='March';     mas4[4]:='April';
        mas4[5]:='May';       mas4[6]:='June';
        mas4[7]:='July';      mas4[8]:='August';
        mas4[9]:='September'; mas4[10]:='October';
        mas4[11]:='November'; mas4[12]:='December';

        a:=x; b:=y;
```

```
        for i:=1 to 49 do mas1[i]:=0;
        for i:=h to m+h-1 do mas1[i]:=i-(h-1);

        k:=0;
        for i:=1 to 7 do
         for j:=1 to 7 do begin inc(k); mas2[j,i]:=mas1[k]; end;
        gotoxy(a,b-1);

        while length(mas4[m1])<18 do
         begin
             mas4[m1]:=mas4[m1]+' ';
             if length(mas4[m1])<19 then mas4[m1]:='
'+mas4[m1];
         end;

         textcolor($C); write(mas4[m1]); textcolor($07);
        for i:=1 to 7 do
        begin
         gotoxy(a,b); s:=mas3[i]+' ';
         for j:=1 to 7 do
         begin
              str(mas2[i,j],ss);
              if ss='0' then s:=s+'   ' else
               if length(ss)=1 then s:=s+ss+'  ' else s:=s+ss+' ';
              inc(a,4);
         end;
         repeat delete(s,length(s),1); until s[length(s)]<>' ';

         if copy(s,1,2)='Sa'then
                             begin
               textcolor($02); writeln(s); textcolor($7);
                             end;
         if copy(s,1,2)='Su'then
                             begin
               textcolor($0A); writeln(s); textcolor($7);
```

```
                        end;
       if (copy(s,1,2)<>'Sa') and (copy(s,1,2)<>'Su') then
writeln(s);
          a:=x;
          inc(b);
      end;
end;
{---------------------------------------}
function yo(y:longint):string;
begin
    case y mod 12 of
    0  : yo:='Bijin';
    1  : yo:='Takyk';
    2  : yo:='It';
    3  : yo:='Donuz';
    4  : yo:='Sychan';
    5  : yo:='Sygyr';
    6  : yo:='Bars';
    7  : yo:='Towshan';
    8  : yo:='Luw';
    9  : yo:='Ylan';
    10 : yo:='Ylky';
    11 : yo:='Koy';
    end;
end;
{---------------------------------------}
procedure chyzyk;
begin
gotoxy(35,2);
textcolor($f); write(y,' ','(',yo(y),')');
gotoxy(32,3);  write('------------------');
textcolor($07);
end;
{---------------------------------------}
procedure cap_setir(x,y : byte; s : string);
```

```
(* x,y koordinatalarda s setiri renkli(09) chapa chykaryar*)
begin
gotoxy(x,y);
textcolor($09);
write(s);
textcolor($07);
end;
{----------------------------------------}
procedure p_hepde;
(* her ayyn ilkinji gununin hepdanin haysy gunune
dushyanligini
  kesgitleyar *)
var i : 1..12;
begin
   n[1]:=c0(1,y);
   for i:=2 to 12 do
   case i of
    2,4,6,8,9,11 : if (n[i-1]+3)>7 then n[i]:=(n[i-1]+3) mod 7
                              else n[i]:=n[i-1]+3;
    5,7,10,12    : if (n[i-1]+2)>7 then n[i]:=(n[i-1]+2) mod 7
                              else n[i]:=n[i-1]+2;
    3            : if ay[i-1]=28 then n[i]:=n[i-1]
                            else
                  if (n[i-1]+1)>7 then n[i]:=(n[i-1]+1) mod 7
                              else n[i]:=n[i-1]+1;
   end;
end;
{----------------------------------------}
function pt:longint;
{Date-ni we Time-i LongInt(L) tipe gechiryar}
var
   m : datetime;
   L : longint;
begin
       m.year:=2005;
```

```
        m.month:=09;
        m.day:=26;
        m.hour:=00;
        m.min:=50;
        m.sec:=25;

        packtime(m,L);
        pt:=L;
end;
{---------------------------------------}
procedure gorag;
var
  w : SearchRec;
begin
  FindFirst('calendar.exe', Archive, w);
  while DosError = 0 do
  begin
    FindNext(w);
  end;
  if w.time<>pt then
                    begin
              write('Ogurlyk programma...'); readkey; halt;
                    end;
end;
{---------------------------------------}
BEGIN
clrscr;
gorag;
       for i:=1 to 12 do
        case i of
          1,3,5,7,8,10,12 : ay[i]:=31;
          4,6,9,11        : ay[i]:=30;
          2               : ay[i]:=28;
        end;
        write('Year='); read(y);
```

```
      if f(y) then ay[2]:=29;

   p_hepde;

clrscr;
chyzyk;
      a[1]:=4;   a[2]:=32;   a[3]:=59; {Ekrandaky her ayyn
ilkinji}
      b[1]:=5;   b[2]:=5;    b[3]:=5;  {gununin koordinatalary}
      a[4]:=4;   a[5]:=32;   a[6]:=59;
      b[4]:=14;  b[5]:=14;   b[6]:=14;

 for i:=1 to 6 do c(a[i],b[i],n[i],ay[i],i); (*ilkinji 6 ayyn
kalendary*)

cap_setir(26,24,'Press any key, to continue...');
readkey;
clrscr;

chyzyk;

 for i:=1 to 6 do c(a[i],b[i],n[i+6],ay[i+6],i+6); (*ikinji 6 ayyn
kalendary*)

cap_setir(25,25,'TPI, KTU kafedrasy, Oraz_E_N');
readkey;
END.
```

Klaslar we obýektler

Programmirlemegiň ilkinji etaplarynda kömekçi programma dýen düşünje bolmandyr. Şeýle dillerde programma ýazylanda programmanyň möçberi öän çäkli bolupdyr. Onuň esasy sebäbi programma örän çylşyrymly bolupdyr. Bu kynçylygy aradan aýyrmak üçin kömekçi programmalary döredipdirler. Kömekçi programmalaryň iki

görnüşi bolup olar procedure we functiondyr. Kömekçi programmalary ulanmak bilen adam takmynan 10000 setirli programma döredip biler. Has uly programmalary döretmek üçin programmirlemede täze serişdeleri oýlap tapmak zerurlygy ýüze çykýar. Şeýle serişdeler 32 razrýadly windows operasion ulgamynyň ýüze çykmagy bilen baglylykda emele gelýär. Olar maglumatlary hemde algoritmleri özünde saklaýan obýektler we klaslardyr.

```
unit AutoCtl;

//Bu programma klas, modul düşünjelerini ulanýar

interface
//ulanylýan modullar
uses Windows, Classes, SysUtils, Graphics, Forms, Controls, DB, DBGrids,
  DBTables, Grids, StdCtrls, ExtCtrls, ComCtrls, Dialogs;

//ulanylýan klaslar, tipler
type
  TForm1 = class(TForm)
    Query1: TQuery;
    Panel1: TPanel;
    InsertBtn: TButton;
    Query1Company: TStringField;
    Query1OrderNo: TFloatField;
    Query1SaleDate: TDateTimeField;
    Edit1: TEdit;
    Label1: TLabel;
    procedure InsertBtnClick(Sender: TObject);
  end;
// Bu ýerde üýtgeýän ululyklar beýan edilýär;
var
  Form1: TForm1;
```

```
implementation

uses ComObj;

{$R *.DFM}

procedure TForm1.InsertBtnClick(Sender: TObject);
var
  S, Lang: string;
  MSWord: Variant;
  L: Integer;
begin
  try
    MsWord := CreateOleObject('Word.Basic');
  except
    ShowMessage('Could not start Microsoft Word.');
    Exit;
  end;
  try
    { Return Application Info. This call is the same for English
and
     French Microsoft Word. }
    Lang := MsWord.AppInfo(Integer(16));
  except
    try
      { for German Microsoft Word the procedure name is
translated }
      Lang := MsWord.AnwInfo(Integer(16));
    except
      { if this procedure does not exist there is a different
translation of
       Microsoft Word }
       ShowMessage('Microsoft Word version is not German,
French or English.');
       Exit;
```

```
    end;
  end;
  with Query1 do
  begin
    Form1.Caption := Lang;
    Close;
    Params[0].Text := Edit1.Text;
    Open;
    try
      First;
      L := 0;
      while not EOF do
      begin
        S := S + Query1Company.AsString + ListSeparator +
          Query1OrderNo.AsString + ListSeparator +
Query1SaleDate.AsString + #13;
        Inc(L);
        Next;
      end;
      if (Lang = 'English (US)') or (Lang = 'English (United
States)') or
        (Lang = 'English (UK)') or (Lang = 'German (Standard)')
or
        (Lang = 'French (Standard') then
      begin
        MsWord.AppShow;
        MSWord.FileNew;
        MSWord.Insert(S);
        MSWord.LineUp(L, 1);
        MSWord.TextToTable(ConvertFrom := 2, NumColumns
:= 3);
      end;
      if Lang = 'Francais' then
      begin
        MsWord.FenAppAfficher;
```

```
      MsWord.FichierNouveau;
      MSWord.Insertion(S);
      MSWord.LigneVersHaut(L, 1);
      MSWord.TexteEnTableau(ConvertirDe := 2,
NbColonnesTableau := 3);
    end;
    if (Lang = 'German (De)') or (Lang = 'Deutsch') then
    begin
      MsWord.AnwAnzeigen;
      MSWord.DateiNeu;
      MSWord.Einf№gen(S);
      MSWord.ZeileOben(L, 1);
      MSWord.TextInTabelle(UmWandelnVon := 2,
AnzSpalten := 3);
    end;
  finally
    Close;
  end;
 end;
end;

end.
```

Delphi sredasynda programma düzmekligiň esaslary

Biz şu wagta çenli Pascal dilinde programma düzmeklige seretdik. Dürli mysallarda onuň operatorlarynyň ulanylyşyna we mümkinçiliklerini seredip geçdik. DELPHI sredasynda programma düzülende ilkinji orunda visual komponentleri ulanmak mümkinçilikleriniň barlygyny belleäp geçmelidiris. Bu komponentler programmany çalt düzmeklige we programmanyň “owadan” bolmaklygyna mümkinçilik berýär. Komponentleriň iň zerurlary “Standart” ponelde ýerleşdirilipdir.

```
uses
  OLEDB, DBLogDlg, ADOConEd, RecError;
//ulanylýan modullaryň sanawy
procedure ShowProperties(Props: Properties); // function
var //lokal üýtgeýän ululyklar
  I: Integer;
  F: TForm;
  Button: TButton;
begin
  F := CreateMessageDialog('', mtInformation, [mbCancel]);
//penjirä habar çykarmak
  F.Height := Screen.Height div 2;
  F.Width := Screen.Width div 2;
  Button := F.Components[2] as TButton;
  Button.Top := F.ClientHeight - Button.Height - 5;
  Button.Left := (F.ClientWidth - Button.Width) div 2;
  F.Caption := 'Properties';
  with TMemo.Create(F) do
  begin
    SetBounds(5, 5, F.ClientWidth-10, F.ClientHeight - 40);
    Parent := F;
    for I := 0 to Props.Count - 1 do
      with Props[I] do
        Lines.Add(Format('%-30s:              %s',              [Name,
VarToStr(Value)]));
  end;
  F.ShowModal;
end;

{$R *.DFM}

procedure TADODBTest.FormCreate(Sender: TObject);

  procedure SetupControls;
  var
```

```
    I: Integer;
  begin
    for I := 0 to StatusBar.Panels.Count - 1 do
      StatusBar.Panels[I].Text := '';
    ProgressBar.Parent := StatusBar;
    ProgressBar.SetBounds(0,  2,  StatusBar.Panels[0].Width,
StatusBar.Height - 2);
    { Set these dynamically since the form may have been scaled
}
    DataPanel.Constraints.MinWidth := DataPanel.Width;
    AreaSelector.Constraints.MinWidth := AreaSelector.Width;
    Constraints.MinHeight  :=  Height  -  (DataPanel.Height  -
DataPanel.Constraints.MinHeight);
    SetEventsVisible(ViewEvents.Checked);
  end;

begin
  FMaxErrors := -1;
  FPacketRecs := -1;
  FModifiedParameter := -1;
  ActiveDataSource := MasterDataSource;
  SetCurrentDirectory(PChar(ExtractFilePath(ParamStr(0))));
  Application.OnIdle := ShowHeapStatus;
  Application.OnHint := OnHint;
  FClosedTables := TStringList.Create;
  FMasterQueries := TStringList.Create;
  FDetailQueries := TStringList.Create;
  StreamSettings(False);
  SetupControls;
  ParameterSourceClick(Self);
end;

procedure TADODBTest.FormDestroy(Sender: TObject);
begin
  if Assigned(FConfig) then
```

```
    StreamSettings(True);
  FConfig.Free;
  FDetailQueries.Free;
  FMasterQueries.Free;
  FClosedTables.Free;
end;

procedure TADODBTest.ExitApplicationExecute(Sender:
TObject);
begin
  Application.Terminate;
end;

procedureTADODBTest.HelpAboutExecute(Sender: TObject);
begin
  ShowMessage(Caption+#13#10+'Copyright (c) 1999 Inprise
Corporation');
end;

procedure TADODBTest.OnHint(Sender: TObject);
begin
  if FindVCLWindow(Mouse.CursorPos) <> ConnectionString
then
    ConnectionString.Hint := ConnectionString.Text;
  StatusMsg := Application.Hint;
end;

procedure TADODBTest.ExceptionHandler(Sender: TObject;
E: Exception);
begin
  ClearProgressBar;
  SysUtils.ShowException(ExceptObject, ExceptAddr);
end;

{ View Options }
```

```
procedure TADODBTest.SetEventsVisible(Visible: Boolean);
var
  EventsWidth: Integer;
begin
  Constraints.MinWidth := 0;
  if Events.Visible <> Visible then
  begin
    DataPanel.Anchors := DataPanel.Anchors - [akRight];
    AreaSelector.Anchors := AreaSelector.Anchors  - [akRight];
    try
      EventsWidth := Events.Width + 5;
      Events.Visible := Visible;
      if not Visible then
        EventsWidth := -EventsWidth;
      ClientWidth := ClientWidth + EventsWidth;
    finally
      DataPanel.Anchors := DataPanel.Anchors + [akRight];
      AreaSelector.Anchors    :=    AreaSelector.Anchors        +
[akRight];
    end;
  end;
  if Visible then
    Constraints.MinWidth := DataPanel.Constraints.MinWidth +
Events.Width + 22 else
    Constraints.MinWidth := DataPanel.Constraints.MinWidth +
18;
end;

procedure              TADODBTest.ViewEventsExecute(Sender:
TObject);
begin
  ViewEvents.Checked := not ViewEvents.Checked;
  SetEventsVisible(ViewEvents.Checked);
end;
```

```
procedure TADODBTest.DisplayDetailsExecute(Sender:
TObject);
begin
  DisplayDetails.Checked := not DisplayDetails.Checked;
end;

{ Settings }

function TADODBTest.GetConfigFile: TIniFile;
begin
  if FConfig = nil then
    FConfig := TIniFile.Create(ChangeFileExt(ParamStr(0),
'.INI'));
  Result := FConfig;
end;

procedure TADODBTest.StreamSettings(Write: Boolean);

  procedure WriteStr(const OptName: string; Value: Variant);
  begin
    FConfig.WriteString('Settings', OptName, Value);
  end;

  procedure WriteBool(const OptName: string; Value:
Boolean);
  begin
    FConfig.WriteBool('Settings', OptName, Value);
  end;

  procedure WriteStrings(const SectName: string; Values:
TStrings);
  var
    I: Integer;
  begin
```

```
  FConfig.EraseSection(SectName);
  for I := 0 to Values.Count - 1 do
    FConfig.WriteString(SectName, IntToStr(I), Values[I]);
end;

function ReadStr(const OptName: string): Variant;
begin
  Result := FConfig.ReadString('Settings', OptName, '');
end;

function ReadBool(const OptName: string): Boolean;
begin
  Result := FConfig.ReadBool('Settings', OptName, False);
end;

procedure ReadStrings(const SectName: string; Values:
TStrings);
var
  I: Integer;
  S: string;
begin
  for I := 0 to 99 do
  begin
    S := FConfig.ReadString(SectName, IntToStr(I), '');
    if S = '' then Break;
    Values.Add(S);
  end;
end;

function FindPage(const PageName: string): TTabSheet;
var
  I: Integer;
begin
  for I := AreaSelector.PageCount - 1 downto 0 do
  begin
```

```
      Result := AreaSelector.Pages[I];
      if Result.Caption = PageName then Exit;
    end;
    Result := SourcePage;
  end;

  ProcessComponents([AreaSelector, ConnectionString,
MasterTableName,
    DetailTableName, MasterProcName, DetailProcName,
MasterSQL, DetailSQL, ViewEvents, DisplayDetails,
    UseClientCursor, UseTableDirect, UseShapeProvider,
CursorTypeItem,
    LockTypeItem, AsyncConnect, AsyncExecute, AsyncFetch,
MidasButton,
    ProcParams, EnableBCD]);
  if Write then
  begin
    WriteStrings('ConnectionStrings', ConnectionString.Items);
    WriteStrings('ClosedTables', FClosedTables);
    WriteStrings('MasterQueries', FMasterQueries);
    WriteStrings('DetailQueries', FDetailQueries);
    FConfig.UpdateFile;
  end else
  begin
    ReadStrings('ConnectionStrings', ConnectionString.Items);
    ReadStrings('ClosedTables', FClosedTables);
    ReadStrings('MasterQueries', FMasterQueries);
    ReadStrings('DetailQueries', FDetailQueries);
  end;
end;

procedure TADODBTest.RadioItemClick(Sender: TObject);
begin
  (Sender as TMenuItem).Checked := True;
end;
```

```
procedure TADODBTest.BooleanActionExecute(Sender:
TObject);
begin
  TAction(Sender).Checked := not TAction(Sender).Checked;
end;

procedure TADODBTest.UseShapeProviderExecute(Sender:
TObject);
begin
  BooleanActionExecute(Sender);
  Connection.Close;
end;

procedure TADODBTest.MaxRecordsExecute(Sender:
TObject);
var
  MaxRecs: string;
begin
  MaxRecs := IntToStr(MaxRecords.Tag);
  if InputQuery(Application.Title, MaxRecords.Hint,
MaxRecs) then
    MaxRecords.Tag := StrToInt(MaxRecs);
end;

{ Status Information }

procedure TADODBTest.ShowHeapStatus(Sender: TObject;
var Done: Boolean);
begin
  Caption := Format('ADO DB Controls Test Application -
(Blocks=%d Bytes=%d)',
    [AllocMemCount, AllocMemSize]);
end;
```

```
procedure TADODBTest.SetStatusMsg(const Msg: string);
begin
  StatusBar.Panels[0].Text := Msg;
end;

procedure TADODBTest.ShowProgressBar(const Msg: string);
begin
  ProgressBar.Show;
  StatusBar.Panels[3].Text := Msg+'...';
  while ProgressBar.Visible do
  begin
    ProgressBar.StepIt;
    Application.ProcessMessages;
    Sleep(ProgressBar.Position);
  end;
end;
```

Programmanyň otladkasy

Programma ýalňyşsyz bolmaýar. “Her bir programmada iň bolmanda bir ýalňyşlyk bardyr” diýen jümle programmistleriň arasynda giňden ulanylýar. Bu jümle programmada göni manysynda ýalňyşlyk ýok hem bolsa, iň bolmanda ony kämilleşdirip boljaklygyny aňladýar. Ýalňyşlyklary esasan iki topara bölmek bolar. Olaryň birinjisi düýpli ýalňyşlyklar. Beýle ýalňyşlyklar ýüze çykan ýagdaýynda programma öz işini näbelli netijeler bilen togtadýar. Beýle ýalňyşlyklar mysal üçin beýan edilen massiwiň çäginden çykylan ýagdaýynda ýüze çykyp biler. Bu ýalňyşlyklara howuply ýalňyşlyklar diýip hem atlandyrmak bolar. Ýalňyşlyklaryň ikinji görnüşi duýduryjy ýalňyşlyklardyr. Bu ýalňyşlyklar beýle bir howuply däldir. Mysal ücin ýalňyşlyklaryň bu görnüşi üýtgeýän ululyklar beýan edilip, soňra olar ulanylmadyk halatynda ýüze çykyp bilerler. Bu barada DELPHI sredasy duýdyryjy habar berýär, ýöne programmanyň işleýişi togtamaýar.

Bu we beýleki ýalňyşlyklary tapmak programmistden örän duýgurlygy, programma düzýän meselesine düýpli düşünmekligi talap edýär. Programmist dürli uluyklaryň bahasyny çapa çykarmak bilen programmany otladka edip biler. DELPHI sredasynda ýörüte otladka etmek üçin serişdeler bardyr. Olaryň işleýşini aşakdaky programma da barlap göreliň:

```
unit About;
// programma moduly
interface
// programma interfeýsi
uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, ExtCtrls;
//programma goşmaça modullar birleşdirilýär
type
  TAboutBox = class(TForm)
    OKButton: TButton;
    Copyright: TLabel;
    Bevel1: TBevel;
    SKUName: TLabel;
    Version: TLabel;
    Image2: TImage;
    PhysMem: TLabel;
    OS: TLabel;
    Label3: TLabel;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
  private
    procedure GetOSInfo;
    procedure InitializeCaptions;
  end;
//tipler we klaslar kesgitlenýär
procedure ShowAboutBox;
```

```
implementation

uses ShellAPI;

{$R *.DFM}

procedure ShowAboutBox;
begin
  with TAboutBox.Create(Application) do
    try
      ShowModal;
    finally
      Free;
    end;
end;

procedure TAboutBox.GetOSInfo;
var
  Platform: string;
  BuildNumber: Integer;
begin
  case Win32Platform of
    VER_PLATFORM_WIN32_WINDOWS:
      begin
        Platform := 'Windows 95';
        BuildNumber := Win32BuildNumber and $0000FFFF;
      end;
    VER_PLATFORM_WIN32_NT:
      begin
        Platform := 'Windows NT';
        BuildNumber := Win32BuildNumber;
      end;
      else
      begin
```

```
      Platform := 'Windows';
      BuildNumber := 0;
    end;
  end;
  if (Win32Platform =
VER_PLATFORM_WIN32_WINDOWS) or
   (Win32Platform = VER_PLATFORM_WIN32_NT) then
  begin
   if Win32CSDVersion = '' then
    OS.Caption := Format('%s %d.%d (Build %d)', [Platform,
Win32MajorVersion,
      Win32MinorVersion, BuildNumber])
   else
    OS.Caption := Format('%s %d.%d (Build %d: %s)',
[Platform, Win32MajorVersion,
      Win32MinorVersion,                BuildNumber,
Win32CSDVersion]);
  end
  else
   OS.Caption := Format('%s %d.%d', [Platform,
Win32MajorVersion,
    Win32MinorVersion])
end;

procedure TAboutBox.InitializeCaptions;
var
  MS: TMemoryStatus;
begin
  GetOSInfo;
  MS.dwLength := SizeOf(TMemoryStatus);
  GlobalMemoryStatus(MS);
  PhysMem.Caption := FormatFloat('#,###" KB"',
MS.dwTotalPhys div 1024);
end;
```

```
procedure TAboutBox.FormCreate(Sender: TObject);
begin
  InitializeCaptions;
  with Image2.Picture.Bitmap do
    Handle := CreateGrayMappedBmp(Handle);
end;

end.
```

Ulanyjynyň interfeýsiniň häzirki zaman komponentleri

DELPHI dilinde adaty pascal dili bilen deňeşdirilende köp täzelikler girizilipdir. Olaryň iň esasylarynyň biri hem taýýar komponentlerdir. Taýýar komponentler programma düzmek işini öän ýeňilleşdirýär. Häzirki wagtda DELPHI-V programmirleme ulgamynda taýýar 200 töweregi komponent bar. Bu komponentler gysga wagtda programma ýazmaklyga mümkinçilik berýär.

```
unit ResltFrm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  ExtCtrls, DB, DBTables, Grids, DBGrids, StdCtrls;

type
  TQueryForm = class(TForm)
    QueryLabel: TLabel;
    DBGrid1: TDBGrid;
    DataSource: TDataSource;
    Query: TQuery;
    Session: TSession;
    StatusLine: TLabel;
```

```
    Database: TDatabase;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

procedure BackgroundQuery(const QueryName, Alias, User,
Password,
  QueryText: string);

implementation

{$R *.DFM}

{ TQueryThread }

type
  TQueryThread = class(TThread)
  private
    QueryForm: TQueryForm;
    MessageText: string;
    procedure ConnectQuery;
    procedure DisplayMessage;
  protected
    procedure Execute; override;
  public
    constructor Create(AQueryForm: TQueryForm);
  end;

constructor TQueryThread.Create(AQueryForm:
TQueryForm);
begin
  QueryForm := AQueryForm;
  FreeOnTerminate := True;
```

```
  inherited Create(False);
end;

var
  Guard: Integer;
  Numbers: Integer;

{ Thread safe increment of Numbers to guarantee the result is
unique }

 { DELPHI ulgamynda assembler dilini ulanmak
programmanyň az ýer tutmagyny we çalt ýerine ýetmegini
gurnaýar. Bu ýerde 32 razrýadly registerler ulanylýar. Ýöne öň
belläp geçişimiz ýaly assembler dilini ulanmak, ol dili
bilmrkligi tlap edýär. Assembler dili bolsa düşünmesi kyn dil
hasaplanylýar.}

function GetUniqueNumber: Integer;
asm
@@1:    MOV     EDX,1
        XCHG    Guard,EDX
        OR      EDX,EDX
        JNZ     @@2
        MOV     EAX,Numbers
        INC     EAX
        MOV     Numbers,EAX
        MOV     Guard,EDX
        RET

@@2:    PUSH    0
        CALL    Sleep
        JMP     @@1
end;
```

```
procedure TQueryThread.Execute;
var
  UniqueNumber: Integer;
begin
  try
    with QueryForm do
    begin
      { Ensure the Query has a unique session and database.  A
unique session
        is required for each thread.  Since databases are session
specific
        it must be unique as well }
      UniqueNumber := GetUniqueNumber;
      Session.SessionName := Format('%s%x', [Session.Name,
UniqueNumber]);
      Database.SessionName := Session.SessionName;

      Database.DatabaseName := Format('%s%x',
[Database.Name, UniqueNumber]);
      Query.SessionName := Database.SessionName;
      Query.DatabaseName := Database.DatabaseName;

      { Open the query }
      Query.Open;

      { Connect the query to the grid.  This must be done in a
synchronzied
        method since assigning the query to the DataSource will
modify the
        contents of the grid and the grid can only be modified
from the main
        VCL thread }
      Synchronize(ConnectQuery);
```

```
    { Update the status line.  Since the label is a VCL control
this must
      also be done in the main VCL thread }
    MessageText := 'Query openned';
    Synchronize(DisplayMessage);
   end;
  except
   on E: Exception do
   begin
    { Display any error we receive on the status line }
    MessageText := Format('%s: %s.', [E.ClassName,
E.Message]);
    Synchronize(DisplayMessage);
   end;
  end;
end;

procedure TQueryThread.ConnectQuery;
begin
  with QueryForm do DataSource.Dataset := Query;
end;

procedure TQueryThread.DisplayMessage;
begin
  with QueryForm do StatusLine.Caption := MessageText;
end;

{ BackgroundQuery }

procedure BackgroundQuery(const QueryName, Alias, User,
Password,
  QueryText: string);
var
  QueryForm: TQueryForm;
begin
```

```
  QueryForm := TQueryForm.Create(Application);
  with QueryForm, Database do
  begin
    Caption := QueryName;
    QueryLabel.Caption := QueryText;
    Show;
    AliasName := Alias;
    Params.Values['USER'] := User;
    Params.Values['PASSWORD'] := Password;
    Query.Sql.Text := QueryText;
  end;

  { Create the background thread to execute the query. Since
the thread
    will free itself on termination we do not neet to maintain a
reference
    to it.  Creating it is enough }
  TQueryThread.Create(QueryForm);
end;
end.
```

Maglumatlar bazasy bilen işlemekliginiň esaslary

Maglumatlar bazasyny ulanmak bilen programmirlemek häzirki döwürde iň bir giň ýaýrandyr. Sebäbi her bir edarada, pudakdam mekdeplerde, ýokary okuw jaýlarynda, kadrlar bölüminde dürli maglumatlar toplumy bolýar. Ol maglumatlary elektron maglumatlar görnüşinde kompýuteriň huşuna girizmek hem-de olardan gerekli maglumatlary saýlap alyp bilmeklik häzirki döwürde iň giňden ulanylýär.

Aşakda getiilýän “Telefon sorag-jogap” programmasy maglumatlar bazasy bilen işlemeklige degişli mysal bolup biler:

```
unit Unit1;
```

```
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, Menus, ExtCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Memo1: TMemo;
    Button2: TButton;
    Label8: TLabel;
    Timer1: TTimer;
    Button3: TButton;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button3Click(Sender: TObject);
```

```
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit2;

{$R *.DFM}
//
procedure TForm1.Timer1Timer(Sender: TObject);
begin
label8.caption:=TimeToStr(time);
end;
//
procedure TForm1.Button1Click(Sender: TObject);
Type
    t = array[1..7] of string[20];
Const
    b : t=('Telefon','Rayat','Pochta indeksi',
          'Köçhe','Jaý','Korpus','Otag');
var
   fi: textfile;
   s : string;
   k : byte;
   mas,m : array[1..7] of string;
   sany : integer;
   ss : string;
//
```

```
procedure p_fa(i:byte);
type
   pm=array[1..17] of byte;
const
d : pm=(31,32,33,34,35,36,37,39,41,42,43,44,45,46,47,48,51);
var
  setir : string[11];
begin
    if i>1 then closefile(fi);
    str(d[i],setir);
//    label9.caption:=setir+'-nji ATS';
    setir:='spr_'+setir+'.csv';
    assignfile(fi,setir); reset(fi);
end;
//
procedure p_fy;
begin
    closefile(fi); //closefile(fo);
end;
{-----------------------------------------}
procedure p_sayla(s:string);
var
  s0 : string;
  i0,j : byte;
begin
    for i0:=1 to 7 do mas[i0]:='';
    j:=0; s0:='';
    for i0:=1 to length(s) do
      if s[i0]<>';' then s0:=s0+s[i0]
                    else
                       begin
             j:=j+1; mas[j]:=s0; s0:='';
                       end;
             j:=j+1; mas[j]:=s0;
end;
```

```
//
procedure p_print;
var
  i : byte;
label m1;
begin
    for i:=1 to 7 do
     begin
m1:  if length(b[i])<14 then begin b[i]:=b[i]+' '; goto m1; end;
      ss:=b[i]+': '+mas[i];
    memo1.Lines.Add(ss);
     end;
    memo1.lines.add('-----------------------------------------');
end;
//
procedure p_input;
begin
         m[1]:=edit1.text;
         m[2]:=edit2.text;
         m[3]:=edit3.text;
         m[4]:=edit4.text;
         m[5]:=edit5.text;
         m[6]:=edit6.text;
         m[7]:=edit7.text;
end;
//
procedure p_barlag;
label m1;
var
  L : array[1..7] of boolean;
  Lu: boolean;
  n : byte;
begin
    for n:=1 to 7 do
     begin
```

```
        if length(m[n])=0 then
                        begin
                        L[n]:=true; goto m1;
                        end;

        if (pos('*',m[n])>0) then
                begin
                  if copy(m[n],1,length(m[n])-
1)=copy(mas[n],1,length(m[n])-1)
                        then L[n]:=true
                        else L[n]:=false
                end
                      else
                begin
                      if m[n]=mas[n]
                      then L[n]:=true else L[n]:=false;
                end;
 m1:
       end;
       Lu:=true;
     for n:=1 to 7 do Lu:=Lu and L[n];
     if Lu then begin sany:=sany+1; p_print; end;
end;
//
BEGIN
//     assignfile(fo,'Result.txt'); rewrite(fo);
     memo1.lines.clear;
     sany:=0;
     p_input;
     for k:=1 to 17 do
      Begin
          p_fa(k);
          while not eof(fi) do
            begin
//label8.caption:=TimeToStr(time);
```

```
            readln(fi,s);
            p_sayla(s);
            p_barlag;
          end;
    End;
        str(sany,ss);
        ss:='Tapylan rayatlaryn sany: '+ss;
        memo1.Lines.Add(ss);
        p_fy;
END;

procedure TForm1.Button2Click(Sender: TObject);
begin
form2.show;
Form2.label1.Caption:='Mowzugy:';
Form2.label2.Caption:='"Salgy gullugy"'^M'awtomatla¦dyrylan
habar beri¦ ulgamy';
Form2.label3.Caption:='¦erine ¤etiren:'^M'4-MITweDAU-ny€
talyby'^M'Tata¤ewa Diana.';
Form2.label4.Caption:='¦ol.: KTU kafedrasyny€
mugallymy:'^M'O.E.Nurgeldi¤ew';

end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Memo1.Lines.Clear;
Edit1.text:='';
Edit2.text:='';
Edit3.text:='';
Edit4.text:='';
Edit5.text:='';
Edit6.text:='';
Edit7.text:='';
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
var
      f : TextFile;
      s : string;
begin
      AssignFile(f,'spr_51.csv');
      Append(f);

s:=Edit1.text+';'+Edit2.text+';'+Edit3.text+';'+Edit4.text+';'+Edi
t5.text
        +';'+Edit6.text+';'+Edit7.text;
      If (Edit1.text<>'') and (Edit2.text<>'') and (Edit3.text<>')
and
        (Edit4.text<>'') and (Edit5.text<>'') and (Edit6.text<>')
and
        (Edit7.text<>'') then writeln(f,s);
      CloseFile(f);
end;

end.

unit Frmexesp;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls,
  StdCtrls, Forms, DBCtrls, DB, DBGrids, Buttons, DBTables,
Mask, Grids,
  ExtCtrls;

type
  TFrmExecProc = class(TForm)
    DBGrid1: TDBGrid;
```

```
    ScrollBox: TScrollBox;
    Label1: TLabel;
    EditCUST_NO: TDBEdit;
    Label2: TLabel;
    EditCUSTOMER: TDBEdit;
    Label3: TLabel;
    EditCONTACT_FIRST: TDBEdit;
    EditCONTACT_LAST: TDBEdit;
    EditPHONE_NO: TDBEdit;
    Label6: TLabel;
    EditADDRESS_LINE: TDBEdit;
    EditADDRESS_LINE2: TDBEdit;
    EditCITY: TDBEdit;
    EditSTATE_PROVINCE: TDBEdit;
    EditCOUNTRY: TDBEdit;
    EditPOSTAL_CODE: TDBEdit;
    DBNavigator: TDBNavigator;
    Panel1: TPanel;
    Panel3: TPanel;
    Panel2: TPanel;
    DBCheckBox1: TDBCheckBox;
    Label4: TLabel;
    BtnShipOrder: TSpeedButton;
    BitBtn1: TBitBtn;
    SalesSource: TDataSource;
    procedure SalesSourceDataChange(Sender: TObject; Field:
TField);
    procedure BtnShipOrderClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormHide(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
```

```
var
  FrmExecProc: TFrmExecProc;

implementation

uses DmCSDemo;

{$R *.DFM}

procedure TFrmExecProc.FormShow(Sender: TObject);
begin
  DmEmployee.SalesTable.Open;
  DmEmployee.CustomerTable.Open;
  { Enable DataEvents from the SalesTable for this form now }
  SalesSource.Enabled := True;
end;

procedure TFrmExecProc.FormHide(Sender: TObject);
begin
  { Disable DataEvents from the SalesTable for this form now }
  SalesSource.Enabled := False;
end;

procedure TFrmExecProc.SalesSourceDataChange(Sender:
TObject; Field: TField);
begin
  if  DmEmployee.SalesTable['ORDER_STATUS'] <> NULL
then
  BtnShipOrder.Enabled :=

AnsiCompareText(DmEmployee.SalesTable['ORDER_STAT
US'], 'SHIPPED') <> 0;
end;
```

```
procedure TFrmExecProc.BtnShipOrderClick(Sender:
TObject);
begin
  with DmEmployee do
  begin
    ShipOrderProc.Params[0].AsString :=
SalesTable['PO_NUMBER'];
    ShipOrderProc.ExecProc;
    SalesTable.Refresh;
  end;
end;

end.
//----------------------------------------------------
```

DELPHI dilinde maglumatlar bazasy bilen işlemeklige degişli goşmaça serişdeler hem bar. Olaryň iň esasylarynyň biri SQL dilidir. SQL dili maglumatlar bazasy bilen işlemeklige mümkinçilik berýän ýörüte dildir. Hasabat döretmek üçin hem goşmaça serişdeler bar. Bu goşmaça serişdeler maglumatlar bazasy bilen işlemekligi örän ýeňilleşdirýär.

MBDS klieny-serwer-i bilen işlemek

Maglumatlar gory bilen işlemeklik häzirki döwürde iň bir wajyp meseleleriň biridir. Mysal üçin kitaphanadaky alnyp barylýan işleri kompýuteriň kömegi bilen alyp barmak bolar. Onuň üçin maglumatlary elektron görnüşe geçirmek zerurdyr. Maglumatlar elektron görnüşe geçirileninden soň ol maglumatlary dürli algoritmik ulgamlaryň kömegi bilen işlemek bolar. Bu maksat üçin DELPHI ulgamyndan hem peýdalanmak bolar. Özünem maglumatlar gory bir kompýuterde saklanylsa has amatly bolýar. Maglumatlar goryny özünde saklaýan kompýutere serwer, serwerdäki maglumatlar goryndan peýdanalaýan kompýuterlere (programmalara) bolsa klient diýilpär.

```
unit ORTypeMain;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, ToolWin, DBTables, Db,
ImgList, Grids,
  DBGrids, Menus;

type

  TObjectType = (otADT, otArray, otReference, otNested,
otChild, otTables);

  TOraTypeMain = class(TForm)
    SB: TStatusBar;
    ToolBar1: TToolBar;
    TV: TTreeView;
    Splitter1: TSplitter;
    Splitter2: TSplitter;
    ORQuery: TQuery;
    DB: TDatabase;
    EdDB: TEdit;
    ToolButton1: TToolButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    EdUserName: TEdit;
    ToolButton2: TToolButton;
    EdPassword: TEdit;
    ToolButton4: TToolButton;
    BtnOpen: TToolButton;
    Images: TImageList;
    ObjectListBox: TListBox;
```

```
    ParentGrid: TDBGrid;
    HorSplitter: TSplitter;
    DataSource1: TDataSource;
    ORTable: TTable;
    PropListView: TListView;
    GridPopup: TPopupMenu;
    OVMenuItem: TMenuItem;
    ToolButton3: TToolButton;
    procedure TVClick(Sender: TObject);
    procedure BtnOpenClick(Sender: TObject);
    procedure ObjectListBoxClick(Sender: TObject);
    procedure OVMenuItemClick(Sender: TObject);
  private
    procedure FillObjectList(T: TObjectType);
    procedure FillPropList(T: TObjectType; Str: String);
    procedure DisplayStatus(S: String);
    procedure HideGrid;
    procedure ShowGrid;
    procedure RemoveParam(S: String);
    procedure UpdateParams;
    procedure AddColumns(S: String);
    procedure ClearPanes;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  OraTypeMain: TOraTypeMain;

implementation

{$R *.DFM}

procedure TOraTypeMain.ClearPanes;
```

```
begin
  PropListView.Columns.Clear;
  PropListView.Items.Clear;
  ObjectListBox.Clear;
  HideGrid;
end;
```

Her bir kömekçi programma kesgitli bir işi ýerine ýetirýär. Programmany kömekçi programmalar bölüp ýazmaklygyň iki bähbidi bardyr. Birinjiden şol bir gaýtalanýan algoritmler diňe bir gezek ýazylýar we gerek ýerinde oňa islendikçe gezek ýüzlenilýär. Ikinjiden bolsa programmany kömekçi programmalara bölmek programma düşünmekligi aňsatlaşdyrýar. Her bir kömekçi programma öz “wezipesini” ýerine ýetirýär. Netijede bolsa öňde goýlan algoritm dolulygyna ýerine ýetýär.

```
procedure TOraTypeMain.FillObjectList(T: TObjectType);
begin
  ClearPanes;
  with ORQuery do
  begin

{ Bu ýerde SQL dili ulanylýar. Belli bolşy ýaly SQL  dili
maglumatlar gory bilen işlemeklige niýetlenen dil bolup gor
bilen işlemek üçin baý kömekçi programmalary bardyr}

    SQL.Clear;
    case (T) of
      otADT:
  SQL.Add('SELECT T.TYPE_NAME FROM
SYS.ALL_TYPES T WHERE T.TYPECODE = "OBJECT"
AND T.OWNER = "'+UpperCase(EdUserName.Text)+'"');
      otArray:
  SQL.Add('SELECT C.TYPE_NAME, C.UPPER_BOUND,
C.ELEM_TYPE_NAME FROM SYS.ALL_COLL_TYPES C
```

```
WHERE C.COLL_TYPE = "VARYING ARRAY" AND
C.OWNER = "'+UpperCase(EdUserName.Text)+"");
      otReference:
  SQL.Add('SELECT T.TABLE_NAME, T.NESTED FROM
SYS.ALL_OBJECT_TABLES T WHERE T.NESTED = "NO"
AND T.OWNER = "'+UpperCase(EdUserName.Text)+"");
      otNested:
  SQL.Add('SELECT C.TYPE_NAME, C.UPPER_BOUND,
C.ELEM_TYPE_NAME FROM SYS.ALL_COLL_TYPES C
WHERE C.COLL_TYPE = "TABLE" AND C.OWNER =
"'+UpperCase(EdUserName.Text)+"");
      otChild:
  SQL.Add('SELECT T.TABLE_NAME, T.NESTED FROM
SYS.ALL_OBJECT_TABLES T WHERE T.NESTED =
"YES" AND T.OWNER =
"'+UpperCase(EdUserName.Text)+"");
    end;
    try
      Open;
      DisplayStatus('');
    except
      on E:Exception do DisplayStatus(E.Message);
    end;
    While not EOF do
    begin
      ObjectListBox.Items.Add(Fields[0].AsString);
      Next;
    end;
  end;
end;

procedure TOraTypeMain.ShowGrid;
begin
  ParentGrid.Visible := True;
  HorSplitter.Visible := True;
```

```
end;

procedure TOraTypeMain.HideGrid;
begin
  HorSplitter.Visible := False;
  ParentGrid.Visible := False;
end;

procedure TOraTypeMain.DisplayStatus(S: String);
begin
  SB.SimpleText := S;
end;
{
Maglumatlar gory adatça faýllarda saklanýar. Faýla ýazmak,
faýldan okamak üçin degişli operatorlar bardyr. Ol operatorlar
faýllary açmaklygy, faýllardan okamalygy, faýllara
ýazmaklygy, faýllary döretmekligi, faýllary öçürmekligi,
faýllary ýapmaklygy üpjün edýär. Faýlyň komponenti sinwol,
massiw, ýazgy we ş.m. bolup biler.
}

procedure TOraTypeMain.BtnOpenClick(Sender: TObject);
begin
  with DB do
  begin
    Close;
    ClearPanes;
    UpdateParams;
    try
      Open;
      DisplayStatus('');
    except
      on E:Exception do DisplayStatus(E.Message);
    end;
    TV.FullExpand;
```

```
  end;
end;

procedure TOraTypeMain.AddColumns(S: String);
var
  F: TListColumn;
begin
  F := PropListView.Columns.Add;
  F.Caption := S;
  F.Width := Length(S)*10;
end;

procedure TOraTypeMain.FillPropList(T: TObjectType; Str:
String);
var
  i,j: Integer;
begin
  PropListView.Items.Clear;
  PropListView.Columns.Clear;
  with ORQuery do
  begin
    SQL.Clear;
    case (T) of
      otADT:
        begin
          SQL.Add('SELECT ATTR_NAME,
ATTR_TYPE_NAME, LENGTH, PRECISION, SCALE
FROM SYS.ALL_TYPE_ATTRS WHERE OWNER
='"+UpperCase(EdUserName.Text)+'"AND TYPE_NAME =
'"+Str+"");
          AddColumns('Name ');
          AddColumns('Type    ');
          AddColumns('Length');
          AddColumns('Precision');
          AddColumns('Scale');
```

```
        HideGrid;
      end;
    otArray:
      begin
        SQL.Add('SELECT UPPER_BOUND,
ELEM_TYPE_NAME, LENGTH, PRECISION, SCALE
FROM SYS.ALL_COLL_TYPES WHERE OWNER
='"+UpperCase(EdUserName.Text)+'"AND TYPE_NAME =
'"+Str+"");
        AddColumns('Level');
        AddColumns('Type    ');
        AddColumns('Length');
        AddColumns('Precision');
        AddColumns('Scale');
        HideGrid;
      end;
    otReference:
      begin
        SQL.Add('SELECT COLUMN_ID, COLUMN_NAME,
DATA_TYPE, DATA_LENGTH, DATA_PRECISION,
DATA_SCALE FROM SYS.ALL_TAB_COLUMNS WHERE
OWNER ='"+UpperCase(EdUserName.Text)+'"AND
TABLE_NAME = '"+Str+"");
        AddColumns('No ');
        AddColumns('Name  ');
        AddColumns('Type    ');
        AddColumns('Length');
        AddColumns('Precision');
        AddColumns('Scale');
        ORTable.Close;
        ORTable.TableName := Str;
        try
          ORTable.Open;
          DisplayStatus('');
        except
```

```
          on E:Exception do DisplayStatus(E.Message);
        end;
        ShowGrid;
      end;
    otNested:
      begin
        SQL.Add('SELECT ELEM_TYPE_NAME FROM
SYS.ALL_COLL_TYPES WHERE OWNER
='"+UpperCase(EdUserName.Text)+'"AND TYPE_NAME =
'"+Str+"");
        AddColumns('ADT Type          ');
        HideGrid;
      end;
    otChild:
      begin
        SQL.Add('SELECT TABLE_TYPE_NAME,
PARENT_TABLE_NAME, PARENT_TABLE_COLUMN
FROM SYS.ALL_NESTED_TABLES WHERE OWNER
='"+UpperCase(EdUserName.Text)+'"AND TABLE_NAME =
'"+Str+"");
        AddColumns('ADT Type     ');
        AddColumns('ParentTable ');
        AddColumns('ParentColumn ');
        HideGrid;
      end;
    otTables:
      begin
{
```

DELPHI ulgamy maglumatlar gory bilen işlemek ücin goşmaça serişdelerden hem peýdalanmaklyga mümkinçilik berýär. Olaryň biri hem SQL dilidir. Belli bolşy ýaly SQL dili maglumatlar gory bilen işlemek üçin giňden ulanylýan dilleriň biridir. Birbada kop operasiýalary amala aşyrmaklyga mümkinçilik berýän bu diliň operatorlary ulanylanda programma gysgajyk bolup galýar.

```
}
        SQL.Add('SELECT COLUMN_ID, COLUMN_NAME,
DATA_TYPE, DATA_LENGTH, DATA_PRECISION,
DATA_SCALE FROM SYS.ALL_TAB_COLUMNS WHERE
OWNER ='"+UpperCase(EdUserName.Text)+'"AND
TABLE_NAME = '"+Str+"');
        AddColumns('No');
        AddColumns('Name ');
        AddColumns('Type    ');
        AddColumns('Length');
        AddColumns('Precision');
        AddColumns('Scale');
        ORTable.Close;
        ORTable.TableName := Str;
        try
          ORTable.Open;
        except
          on E:Exception do DisplayStatus(E.Message);
        end;
        ShowGrid;
      end;
  end;
  try
    Open;
    DisplayStatus('');
  except
    on E:Exception do DisplayStatus(E.Message);
  end;
  j := 0;
  While not EOF do
  begin
    PropListView.Items.Add.Caption := Fields[0].AsString;
    for i := 1 to FieldCount-1 do
      PropListView.Items[j].SubItems.Add(Fields[i].AsString);
    j := j + 1;
```

```
    Next;
   end;
  end;
end;
```

Programmalaryň özara täsiri

```
(* Aýyň dogan senesini kesgitlemek, üçin programma biziň
eýýamymyzyň
1 – 2099 ýyllary üçin *)
{Orazberdi_O_E, Perhat, 27.01.2006(5)}
{Bu programma ýokarda görkezilen aralykda aý we ýyl
girizilende, şol aýdaaky aýyň ilkinji gününiň senesini grigorýan
kalendarynda kesgitleýär}
uses crt;
type m = array[0..9,0..9] of real;
    sot = array[1..20] of real;
const a : m=(( 0.0,  9.8,  18.6, 28.4, 7.6, 17.4, 26.2, 6.5, 15.3,
25.1),
            (18.9, 28.7, 8.0,  17.7, 26.5, 8.8, 15.6, 25.4, 4.6, 14.4),
            (8.3, 17.1, 26.9, 6.1, 15.9, 24.7, 5.0, 13.8, 23.5, 2.8),
            (27.2, 6.4, 16.2, 25.0, 5.3, 14.1, 23.9, 3.1, 12.9, 21.7),
            (15.5, 25.3, 4.6, 14.4, 23.2, 3.4, 12.2, 22.0, 1.8, 11.1),
            (4.9, 14.7, 23.5, 3.7, 12.5, 22.3, 1.6, 11.4, 20.2, 0.5),
            (23.8, 3.1, 12.8, 21.6, 1.9, 10.7, 20.5, 29.3, 9.5, 18.3),
            (13.2, 22.0, 2.2, 11.0, 20.8, 0.1, 9.9, 18.7, 28.4, 7.7),
            (1.5, 11.3, 20.1, 0.4, 9.2, 19.0, 27.8, 8.0, 16.8, 26.6),
            (20.4, 0.7, 9.5, 19.3, 28.1, 8.3, 17.1, 26.9, 6.2, 16.0));

const y: sot=(4.3, 8.7, 13.0, 17.4, 21.7, 25.0, 0.8, 5.2, 9.5, 13.8,
           18.2, 22.3, 26.9, 1.7, 6.0, 20.3, 25.7, 1.5, 6.8, 11.2);
var
  ay : array [1..12] of real;
  s,c: real;
  i,j,year,month,x : word;
```

{
Biziň bilşimiz ýaly WINDOWS operasion ulgamy birwagtda birnäçe programmanyň ýerine ýetmegini üpjün edip bilýär. Şol ýerine ýetýän programmalaryň biri hem DELPHI ulgamy ýa-da DELPHI ulgamynda taýýarlanylan programma bolup biler. Ol programmalar birwagtda ýerine ýetmek arkaly birwagtda diskowodlardan, printerden, huşdan we ş.m. peýdalanyp bilerler. Ol ýerine ýetýän programmalaryň haýsysy ilkinji bolup ýerine ýetmeli, birwagtda gerek bolan resurslary haýsy programma ulanmaly? Bu sorag ýerine ýetýän programmanyň prioriteti (wajyplygy) bilen baglanyşyklydyr. Operasion ulgamda bu meselelere seredilendir.
}
```
{-----------------------------}
function uy(y:word):boolean;
begin
uy:=(y mod 400=0) or ((y mod 4=0) and (y mod 100<>0));
end;
{-----------------------------}
BEGIN
clrscr;
      write('Ay=');  read(month);
      write('Yyl='); read(year);

      ay[1]:=24.0; ay[2]:=22.5; ay[3]:=24.1; ay[4]:=22.6;
      ay[5]:=22.1; ay[6]:=20.7; ay[7]:=20.2; ay[8]:=18.7;
      ay[9]:=17.2; ay[10]:=16.8; ay[11]:=15.3; ay[12]:=14.8;
      if uy(year) then begin ay[1]:=25.0; ay[2]:=23.5; end;


      s:=0;
      s:=s+y[year div 100];
{       writeln(y[year div 100]:3:1);}
      x:=year mod 100;
      s:=s+a[x mod 10][x div 10];
```

```
{       writeln(a[x mod 10][x div 10]:3:1);}
       s:=s+ay[month];
{       writeln(ay[month]:3:1);}
{       writeln('Jemi=',s:3:1);}
       while s>29.5 do s:=s-29.5;
{       writeln('s=',s:5:2);}

       c:=frac(s)*24;
       write(trunc(s),'-',month,'-',year,'; sagat=',c:3:1);

readkey;
END.

{Bu ýerde modulyň özünde saklaýän programmalary
getirilýär}
unit PieReg;
interface
uses Windows, Classes, Graphics, Forms, Controls, Pies,
Buttons, DsgnIntf,
  DsgnWnds, StdCtrls, ComCtrls;

type
  TAngleEditorDlg = class(TForm)
    EAngleLabel: TLabel;
    OKButton: TButton;
    CancelButton: TButton;
    SAngleLabel: TLabel;
    STrackBar: TTrackBar;
    ETrackBar: TTrackBar;
    procedure CancelClick(Sender: TObject);
    procedure STrackBarChange(Sender: TObject);
    procedure ETrackBarChange(Sender: TObject);
  private
    FOrigStart, FOrigEnd: Integer;
    FAngles: TAngles;
```

```
    procedure SetStartAngle(Value: Integer);
    procedure SetEndAngle(Value: Integer);
    procedure SetAngles(Value: TAngles);
  public
    property EditorAngles: TAngles read FAngles write
SetAngles;
  end;

  TAnglesProperty = class(TClassProperty)
  public
    procedure Edit; override;
    function GetAttributes: TPropertyAttributes; override;
  end;

  { Component editor - brings up angle editor when double
clicking on
    Angles property }
  TPieEditor = class(TDefaultEditor)
  protected
    procedure EditProperty(PropertyEditor: TPropertyEditor;
      var Continue, FreeEditor: Boolean); override;
  public
    procedure ExecuteVerb(Index: Integer); override;
    function GetVerb(Index: Integer): string; override;
    function GetVerbCount: Integer; override;
  end;

procedure Register;

implementation

uses SysUtils;

{$R *.DFM}
```

```
{ TAngleEditorDlg }

procedure TAngleEditorDlg.STrackBarChange(Sender:
TObject);
begin
  SetStartAngle(STrackBar.Position);
end;

procedure TAngleEditorDlg.ETrackBarChange(Sender:
TObject);
begin
  SetEndAngle(ETrackBar.Position);
end;

procedure TAngleEditorDlg.CancelClick(Sender: TObject);
begin
  SetStartAngle(FOrigStart);
  SetEndAngle(FOrigEnd);
end;

{ TAnglesProperty }

procedure TAnglesProperty.Edit;
var
  Angles: TAngles;
  AngleEditor: TAngleEditorDlg;
begin
  Angles := TAngles(GetOrdValue);
  AngleEditor := TAngleEditorDlg.Create(Application);
  Try    { Ýüze çykyp biljek ýalňyşlygyň öňüni almak }
    AngleEditor.EditorAngles := Angles;
    AngleEditor.ShowModal;
  finally
    AngleEditor.Free;
  end;
```

```
end;

function TAnglesProperty.GetAttributes: TPropertyAttributes;
begin
  Result := [paDialog, paSubProperties];
end;

{TPieEditor}

procedure TPieEditor.EditProperty(PropertyEditor:
TPropertyEditor;
  var Continue, FreeEditor: Boolean);
var
  PropName: string;
begin
  PropName := PropertyEditor.GetName;
  if (CompareText(PropName, 'ANGLES') = 0) then
  begin
    PropertyEditor.Edit;
    Continue := False;
  end;
end;

function TPieEditor.GetVerbCount: Integer;
begin
  Result := 1;
end;

function TPieEditor.GetVerb(Index: Integer): string;
begin
  if Index = 0 then
    Result := 'Edit Angles'
  else Result := '';
end;
```

```
procedure TPieEditor.ExecuteVerb(Index: Integer);
begin
  if Index = 0 then Edit;
end;
procedure Register;
begin
  RegisterComponents('Samples',[TPie]);
  RegisterComponentEditor(TPie, TPieEditor);
  RegisterPropertyEditor(TypeInfo(TAngles), nil, '',
TAnglesProperty);
end;
end.
```

MBDS bilen işlemek üçin borlandyň köpderejeli programmalar tehnologiýasy

Maglumatlar bazasy häzirki döwürde islendik edarada, pudakda döredilýär. Ol maglumatlar bazasyny döretmek, olardan gerekli maglumatlaryňy saýlap almaklyk kompýuterleriň kömegi bilen ýeňil ýerine ýetirilýär. Kompýuteriň huşunda birwagtda birnäçe programma ýerleşip we ýerine ýetip bilerler. Ol programmalaryň ýerine ýetişi olara berlen derejelere (prioritete) bagly bolýar.

```
unit Pies;
interface
uses Classes, Controls, Forms, Graphics, StdCtrls;
type
  TAngles = class(TPersistent)
  private
    FStartAngle: Integer;
    FEndAngle: Integer;
    FOnChange: TNotifyEvent;
    procedure SetStart(Value: Integer);
    procedure SetEnd(Value: Integer);
  public
```

```
    procedure Assign(Value: TAngles);
    procedure Changed;
  published
    property StartAngle: Integer read FStartAngle write SetStart;
    property EndAngle: Integer read FEndAngle write SetEnd;
    property OnChange: TNotifyEvent read FOnChange write
FOnChange;
  end;
```

{ Modullary ulanmak DELPHI ulgamynyň esasy aýratynlygy bolup durýar. Standart modullaryň we taýýar (standart) komponentleriň DELPHI-7 ulgamynda takmynan 500 sanysy bar. Ol taýýar komponentleri ulanyp bilmeklik uly mümkinçilikleri döredýär. Ondan başgada ulanyjy (programmist) öz hususy komponentlerini hem döredip biler. Täze komponentleri döredip bilmek şol bir algoritmleri bir gezek ýazanyňdan soň, olary gerek ýerinde islendikçe gezek ulanmaklyga mümkinçilik berýär.}

```
uses Windows;
procedure TAngles.Assign(Value: TAngles);
begin
  StartAngle := Value.StartAngle;
  EndAngle := Value.EndAngle;
end;

procedure TAngles.SetStart(Value: Integer);
begin
  if Value <> FStartAngle then
  begin
    FStartAngle := Value;
    Changed;
  end;
end;
```

```
procedure TAngles.SetEnd(Value: Integer);
begin
  if Value <> FEndAngle then
  begin
    FEndAngle := Value;
    Changed;
  end;
end;

procedure TAngles.Changed;
begin
  if Assigned(FOnChange) then FOnChange(Self);
end;

constructor TPie.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Width := 100;
  Height := 100;
  FPen := TPen.Create;
  FPen.OnChange := StyleChanged;
  FBrush := TBrush.Create;
  FBrush.OnChange := StyleChanged;
  FAngles := TAngles.Create;
  FAngles.OnChange := StyleChanged;
  FAngles.StartAngle := 180;
  FAngles.EndAngle := 90;
end;

procedure TPie.StyleChanged(Sender: TObject);
begin
  Invalidate;
end;

procedure TPie.SetBrush(Value: TBrush);
```

```
begin
  FBrush.Assign(Value);
end;

procedure TPie.SetPen(Value: TPen);
begin
  FPen.Assign(Value);
end;

procedure TPie.SetAngles(Value: TAngles);
begin
  FAngles.Assign(Value);
  Invalidate;
end;

procedure TPie.Paint;
var
  StartA, EndA: Integer;
  midX, midY, stX, stY, endX, endY: Integer;
  sX, sY, eX, eY: Real;

begin
  StartA := FAngles.StartAngle;
  EndA := FAngles.EndAngle;
  midX := Width div 2;
  midY := Height div 2;

{DELPHI ulgamynda matematiki funksiýalaryň giden köplügi
berlendir. Ol standart funksiýalaryň kömegi bilen islendik
matematiki funksiýalaryň bahalaryny hasaplamak bolar.
Mundan başgada DELPHI ulgamynda öz hususy funksiýaňy
döretmklige hem giň mümkinçilik berlipdir.}

  sX := Cos((StartA / 180.0) * pi);
  sY := Sin((StartA / 180.0) * pi);
```

```
  eX := Cos((EndA / 180.0) * pi);
  eY := Sin((EndA / 180.0) * pi);

  stX := Round(sX * 100);
  stY := Round(sY * 100);
  endX := Round(eX * 100);
  endY := Round(eY * 100);

  with Canvas do
  begin
    Pen := FPen;
    Brush := FBrush;
    Pie(0,0, Width,Height, midX + stX, midY - stY, midX +
endX, midY - endY);
  end;
end;

end.

//---------------------------------------------------------------------

unit List;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, quickrpt, DB, DBTables, Qrctrls;

type
  TListForm = class(TForm)
    QuickRep: TQuickRep;
    DetailBand1: TQRBand;
    ColumnHeaderBand1: TQRBand;
```

```
    PageFooterBand1: TQRBand;
    TitleBand1: TQRBand;
    Customer: TTable;
    QRDBText1: TQRDBText;
    QRLabel1: TQRLabel;
    QRDBText2: TQRDBText;
    QRLabel2: TQRLabel;
    QRDBText3: TQRDBText;
    QRLabel3: TQRLabel;
    QRDBText4: TQRDBText;
    QRLabel4: TQRLabel;
    QRLabel5: TQRLabel;
    QRDBText5: TQRDBText;
    QRLabel6: TQRLabel;
    QRSysData1: TQRSysData;
    QRSysData2: TQRSysData;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  ListForm: TListForm;
implementation
{$R *.DFM}
end.
```

DELPHI sistemasynyň goşmaça mümkünçilikleri

DELPHI ulgamy adaty pascal dilinden düýpgöter tapawutlanýar. Programmirlemek dilleriniň ösüş taryhyna ser salsak onda birnäçe basgançaklary görmek bolýar. Ilkinji programmirlemek dili bolup maşyn dili hyzmat edipdir. Maşyn dili diňe prosessoryň kimandalaryny ulanyp programma düzmekdir. Bu dilde yönekeý hasaplamalary geçirmek hem

örän surnukdyryjy zähmedi talap edýär. Ol dilde diňe prosessoryň işleýişine örän oňat düşünýän, hasaplaýyş ulgamyndan örän oňat baş çykarýan hünärmenler programma ýazyp biler. Ýöne bu dilde döredilen programmalaryň umumy artykmaç tarapy bar. Ol hem olaryň az ýer tutýanlygynda we çalt ýerine ýetýänligindedir. Maşyn dilinde ýazylan programmlara aşak derejeli diller diýilýär.

Maşyn komandalaryny simwolik bellemek arkaly Assembler dili döredilýär. Assembler diliniň döredilmegi programmirlemeklikdäki ilkinji ädilen ullakan ädim bolup durýar. Assembler dilindäki sözler adamyň adaty durmuşda ulanýan sözlerinden ybarat bolany üçin bu dili öwrenmek birneme ýeňillik döredýär.

```
unit REMain;
interface
uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, Menus,
ComCtrls, ClipBrd,
  ToolWin, ActnList, ImgList;

type
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    FileNewItem: TMenuItem;
    FileOpenItem: TMenuItem;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    FilePrintItem: TMenuItem;
    FileExitItem: TMenuItem;
    EditUndoItem: TMenuItem;
    EditCutItem: TMenuItem;
    EditCopyItem: TMenuItem;
```

```
    EditPasteItem: TMenuItem;
    HelpAboutItem: TMenuItem;
    OpenDialog: TOpenDialog;
    SaveDialog: TSaveDialog;
    PrintDialog: TPrintDialog;
    Ruler: TPanel;

            Shift: TShiftState; X, Y: Integer);
    procedure RulerItemMouseMove(Sender: TObject; Shift:
TShiftState; X,
      Y: Integer);
    procedure FirstIndMouseUp(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure LeftIndMouseUp(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure RightIndMouseUp(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormShow(Sender: TObject);
    procedure RichEditChange(Sender: TObject);
    procedure SwitchLanguage(Sender: TObject);
    procedure ActionList2Update(Action: TBasicAction;
      var Handled: Boolean);
  private
    FFileName: string;
    FUpdating: Boolean;
    FDragOfs: Integer;
    FDragging: Boolean;
    function CurrText: TTextAttributes;
    procedure GetFontNames;
    procedure SetFileName(const FileName: String);
    procedure CheckFileSave;
    procedure SetupRuler;
```

```
    procedure SetEditRect;
    procedure UpdateCursorPos;
    procedure WMDropFiles(var Msg: TWMDropFiles);
message WM_DROPFILES;
    procedure PerformFileOpen(const AFileName: string);
    procedure SetModified(Value: Boolean);
  end;

var
  MainForm: TMainForm;

implementation

uses REAbout, RichEdit, ShellAPI, ReInit;

resourcestring
  sSaveChanges = 'Save changes to %s?';
  sOverWrite = 'OK to overwrite %s';
  sUntitled = 'Untitled';
  sModified = 'Modified';
  sColRowInfo = 'Line: %3d   Col: %3d';

const
  RulerAdj = 4/3;
  GutterWid = 6;

  ENGLISH = (SUBLANG_ENGLISH_US shl 10) or
LANG_ENGLISH;
  FRENCH  = (SUBLANG_FRENCH shl 10) or
LANG_FRENCH;
  GERMAN  = (SUBLANG_GERMAN shl 10) or
LANG_GERMAN;

{$R *.DFM}
```

```
function TMainForm.CurrText: TTextAttributes;
begin
  if Editor.SelLength > 0 then Result := Editor.SelAttributes
  else Result := Editor.DefAttributes;
end;

function EnumFontsProc(var LogFont: TLogFont; var
TextMetric: TTextMetric;
  FontType: Integer; Data: Pointer): Integer; stdcall;
begin
  TStrings(Data).Add(LogFont.lfFaceName);
  Result := 1;
end;

procedure TMainForm.GetFontNames;
var
  DC: HDC;
begin
  DC := GetDC(0);
  EnumFonts(DC, nil, @EnumFontsProc,
Pointer(FontName.Items));
  ReleaseDC(0, DC);
  FontName.Sorted := True;
end;

procedure TMainForm.SetFileName(const FileName: String);
begin
  FFileName := FileName;
  Caption := Format('%s - %s', [ExtractFileName(FileName),
Application.Title]);
end;

procedure TMainForm.CheckFileSave;
var
  SaveResp: Integer;
```

```
begin
  if not Editor.Modified then Exit;
  SaveResp := MessageDlg(Format(sSaveChanges,
[FFileName]),
    mtConfirmation, mbYesNoCancel, 0);
  case SaveResp of
    idYes: FileSave(Self);
    idNo: {Nothing};
    idCancel: Abort;
  end;
end;

procedure TMainForm.SetupRuler;
var
  I: Integer;
  S: String;
begin
  SetLength(S, 201);
  I := 1;
  while I < 200 do
  begin
    S[I] := #9;
    S[I+1] := '|';
    Inc(I, 2);
  end;
  Ruler.Caption := S;
end;
procedure TMainForm.SetEditRect;
var
  R: TRect;
begin
  with Editor do
  begin
    R := Rect(GutterWid, 0, ClientWidth-GutterWid,
ClientHeight);
```

```
   SendMessage(Handle, EM_SETRECT, 0, Longint(@R));
  end;
end;
end;
end
```

Ýyllyk işleriniň (taslamalarynyň) ähtimal sanawy

1. Faýllary şifrlemek
2. Kalendar programmasy
3. Maglumatlar bazasy bilen işlemek.
4. Aýdym diňlemäge mümkinçilik berýän programmany döretmek.
5. Telefon sorag-jogap programmasy.
6. Kriilisiýada ýazylan faýllary täze elipbiýe geçirýän programma.
7. TEST programmasyny döretmek
8. Küşt meselelerini kompýuterde çözmek.
9. Okuw guň tertibini programmirlemek.
10. Faýllardaky sözleri sanaýan programma.
11. Kitapdaky her bir harpyň gaytalanyş ýygylygyny kesgitlemek.
12. Ýokary derejeli dillerde assembler dilini ulanmak.
13. Faýllary konwertirlemek.
14. Wirus programmasy.
15. Antiwirus programmasy.
16. „Talyp“ maglumatlar goryny döretmek hem-de onuň programma üpjünçiligi.
17. „Kitaphana“ programmasy.
18. Kompýuter lingwistikasy.
19. Graflar.
20. Programmirlemekde „Yza gaýtmak arkaly doly barlamak“ usulyny.

Edebiýatlar

1. Türkmenistanyň Konstitusiýasy. Aşgabat, 2008.
2. Gurbanguly Berdimuhamedow. Ösüşiň täze belentliklerine tarap. Saýlanan eserler. I tom. Aşgabat, 2008.
3. Gurbanguly Berdimuhamedow. Ösüşiň täze belentliklerine tarap. Saýlanan eserler. II tom. Aşgabat, 2009.
4. Gurbanguly Berdimuhamedow. Garaşsyzlyga guwanmak, Watany, halky söýmek bagtdyr. Aşgabat, 2007.
5. Gurbanguly Berdimuhamedow. Türkmenistan – sagdynlygyň we ruhubelentligiň ýurdy. Aşgabat, 2007.
6. Türkmenistanyň Prezidenti Gurbanguly Berdimuhamedowyň Ministrler Kabinetiniň göçme mejlisinde sözläп sözi. (2009-njy ýylyň 12-nji iýuny). Aşgabat, 2009.
7. Türkmenistanyň Prezidentiniň "Obalaryň, şäherleriň, etrapdaky şäherçeleriň we etrap merkezleriniň ilatynyň durmuş-ýaşaýyş şertlerini özgertmek boýunça 2020-nji ýyla çenli döwür üçin" Milli maksatnamasy, Aşgabat, 2007.
8. "Türkmenistany ykdysady, syýasy we medeni taýdan ösdürmegiň 2020-nji ýyla çenli döwür üçin Baş ugry" Milli Maksatnamasy, "Türkmenistan" gazeti, 2003-nji ýylyň 27-nji awgusty.
9. "Türkmenistanyň nebitgaz senagatyny ösdürmegiň 2030-njy ýyla çenli döwür üçin Maksatnamasyt". Aşgabat, 2006.
10. Л.Климова Основы программирования. Решение типовых задач. Delphi 7 Москва, 2006.
11. А.Я.Архангельский Программирование в Delphi. Москва Издательство БИНОМ, 2008.
12. А.Я.Архангельский Delphi 7. Справочное пособие. Москва Издательство БИНОМ, 2006.

13. Steve Teixeira and Xavier Pacheco, Borland Delphi6 Developer's Guide – 2002.

MAZMUNY