ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

АБУ РАЙХОН БЕРУНИЙ НОМИДАГИ ТОШКЕНТ ДАВЛАТ ТЕХНИКА УНИВЕРСИТЕТИ

Дастурлаштириш воситаларининг замонавий технологиялари. Delphi асослари

Маърузалар матни

(электроника ва автоматика факультети таълим йўналиши бакалаврлари учун)

Кодиров Мирхусан Мирпулатович

ТОШКЕНТ – 2008

Алгоритмик тиллар. Унинг таърифи. Дастурлаштириш воситаларининг замонавий технологиялари. Delphi. Унинг лаќжалари ва асосий фарқлари.

Компьютерда дастурлаш охирги йилларда жуда тез ривожланиб дастур тузушга қизиқувчилар сони ошиб бормоқда. 10-15 йил олдин ўз дастурларини Windows муҳитида яратиш кўпгина дастурчиларнинг орзуси эди. Delphi дастурлаш воситасининг яратилиши эса нафақат профессионал дастурчилар, балки оддий дастур тузувчилар учун ҳам кенг йўл очиб берди. Бу маруза Delphi дастурлаш воситаси буйича тулик тасниф беради.

Delphi дастурлаш воситаси Turbo Pascal тилининг ривожи бўлган Object Pascal тилини ишлатади. Ҳозирги кунда бу тилга жуда кўплаб янгиликлар киритилган унинг имкониятлари янада кенгайтирилган, шу сабаб бу тилни Delphi тили деб ҳам аташ мумкин.

Delphi тили ҳам бошқа дастурлаш тиллари каби ўз алфавитига ва белгиларига эга. У 26 бош лотин ҳарфларини, 0 дан 9 гача бўлган араб рақамларини ва қуйидаги белгиларни ишлатади: бўшлиқ белгиси; 4 та арифметик амаллар + , - , * , / ; мантиқий амалларни бажариш учун <, >, <=, >= , <>, = белгиларини ишлатади. Булардан ташқари вергул, нуқта, икки нуқта, кичик қавс, катта ва ўрта қавслар. Дастурда изоҳлар исталган жойда берилиши мумкин. Улар катта қавс ичида ёзилади.

Масалан. Program ad; { Бу дастур номи }

DELPHI ВИЗУАЛ ДАСТУРЛАШ МУҲИТИ ҲАҚИДА АСОСИЙ ТУШУНЧАЛАР

Delphi дастурлаш муҳити

Delphi -Windows операцион тизимида дастур яратишга йўналтирилган дастурлаш мухитидир. Delphiда дастур тузиш замонавий визуал лойиҳалаш технологияларига асосланган бўлиб, унда дастурлашнинг объектга йўналтирилган ғояси мужассамлашган. Delphiда дастур Turbo Pascal дастурлаш тилининг ривожи бўлган Object Pascal тилида ёзилади.

Delphi -бир неча муҳим аҳамиятга эга бўлган технологиялар комбинациясини ўзида мужассам этган:

- юқори даражадаги машинали кодда тузилган комплятор;
- объектга йўналтирилган компонента лар моделлари;
- дастур иловаларини визуал тузиш;
- маълумотлар базасини тузиш учун юқори масштабли восита.

Delphi - Windows муҳитида ишлайдиган дастур тузиш учун қулай бўлган восита бўлиб, компьютерда дастур яратиш ишларини автоматлаштиради, хатоликларни камайтиради ва дастур тузувчи меҳнатини енгиллаштиради. Delhiда дастур замонавий визуал лойиҳалаш технологияси асосида объектга йўналтирилган дастурлаш назариясини ҳисобга олган ҳолда тузилади. Delphi системаси Turbo Pascal 7.0. тилининг ривожи бўлган объектга йўналтирилган Object Pascal дастурлаш тилини ишлатади.

Маълумки дастур тузиш сермашаққат жараён, лекин Delphi тизими бу ишни сезиларли даражада соддалаштиради ва масала турига қараб дастур тузувчи ишининг 50-80%ни тизимга юклайди. Delphi тизими дастурни лойиҳалаш ва яратиш вақтини камайтиради, ҳамда Windows муҳитида ишловчи дастур иловаларини тузиш жараёнини осонлаштиради.

Delphi ўзида бир қанча замонавий маълумотлар базасини бошқариш тизимлари дастурлаш

технологияларини ҳам маълумотлар базасини яратишда ишлатади.

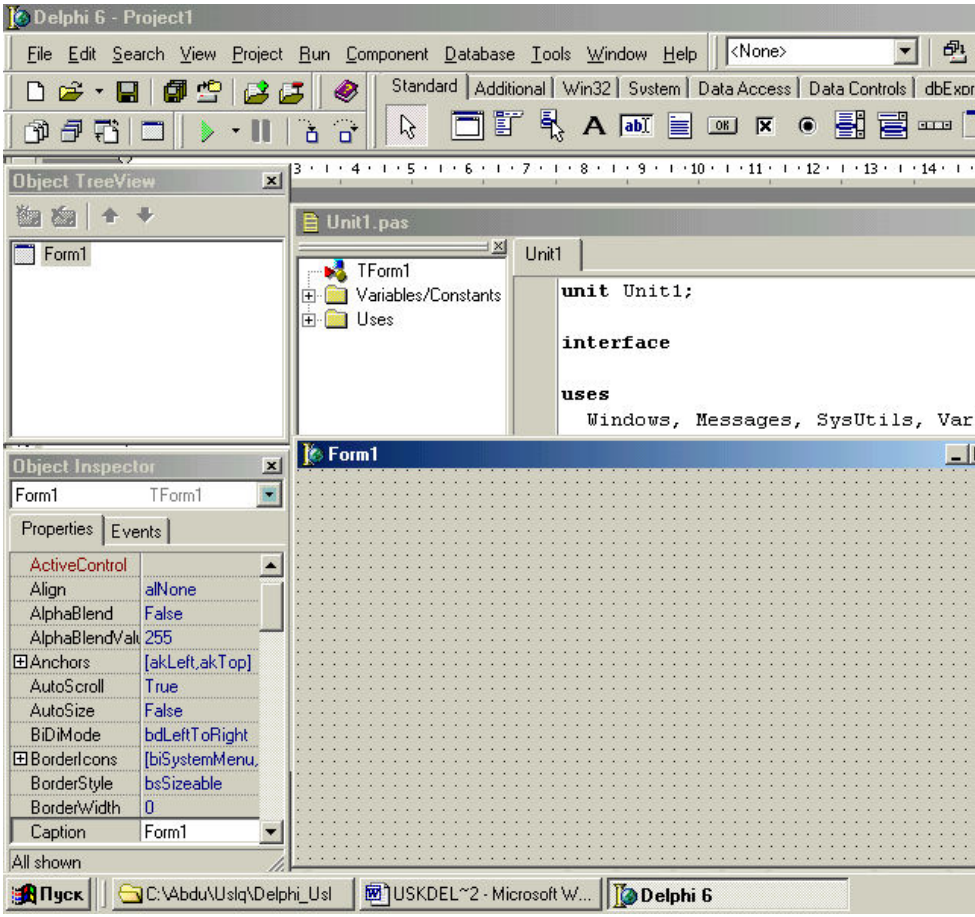
Delphi тизими ойнаси ва унинг элементлари

Delphi тизимида ишни бошлаш учун уни дастурлар менюсидан топиб ишга туширамиз.

Пуск=>Программы=>Borland Delphi=>Delphi

Delphi ойнаси кўриниши одатдагидан анча бошқачароқ бўлиб, у ўз ичига бешта ойнани олади:

- бош ойна - Delphi Project1;
- форма ойнаси - Form1;
- объект хоссаларини таҳрирлаш ойнаси-Object Inspector;
- объектлар рўйхатини кўриш ойнаси - Object tree View;
- дастур кодларини таҳрирлаш ойнаси - Unit.pas.



Бош ойна экраннинг юқори қисмида жойлашган бўлиб, унинг биринчи қаторида сарловҳа, яъни проектнинг номи жойлашган. Иккинчи қаторда буйруқлар менюси горизантал кўринишда жойлашган. Кейинги қаторнинг чап тарафида ускуналар панели ва ўнг тарафида компонеталар политраси жойлашган.

Буйруқлар менюси қуйидагиларни ўз ичига олган:

-File (файл) бўлими файллар устида иш бажариш учун керакли буйруқларни ўз ичига олган;

-Edit (тахрир) бўлими файл ичидаги маълумотларни тахрирлаш учун керакли буйруқларни ўз ичига олган;

-Seerch

-View

-Compile

-Run формани ишга тушириш.

-Options

-Tols сервис хизматидан фойдаланиш.

-Help ёрдам чақириш.

Форма ойнасида иловалар яратилади. Object Inspector ойнаси объект хоссаларини тахрирлаш учун хизмат қилади. Объект хоссалари бу - объектга берилган характеристика бўлиб, унинг кўриниш, жойлашиши ва ҳолатидир. Масалан, Width ва Height хоссалари форма ўлчамини, top ва Lift эса форманинг экрандаги ҳолати, Caption - сарловҳа матнини аниқлайди.

Визуал дастурлаш технологиясида объект деганда мулоқат ойнаси ва бошқариш элементлари (киритиш ва чиқариш майдони, буйруқ тугмалари, переключателлар ва бошқа) тушунилади.

Delphiда дастурлаш иккита ўзаро таъсир этувчи бир-бири билан боғлиқ жараён асосида ташкил қилинади:

-дастурни визуал лойиҳалаш жараёни;

-дастур кодларини киритиш (ёзиш) жараёни.

Кодларни ёзиш учун махсус код ойнаси мавжуд бўлиб, у дастур матнини киритиш ва тахрирлаш учун мўлжаллангандир. Бу кодларни ёзиш ойнасида дастурлаш Pascal тилининг ривожи бўлган ва кенгайтирилган Object Pascal тилида тузилади.

Кодларни ёзиш ойнаси бошланишда ўз ичига ҳали бўш формани акслантирувчи дастур матнини ёзиб чиқаради. Дастур лойиҳасини ишлаши мобайнида дастурчи керакли дастур операторларини киритиб, формани лойиҳа бўйича акслантиради. Delphiда дастурлаш форма ойнасини ташкил этишдан бошланади.

Оддий дастур иловасини яратиш кетма-кет File=> New=> Application буйруғини бериш билан бошланади. Бу буйруқни беришдан олдин иккита асосий ишни бажариш лозим:

-папка ташкил этиш;

-тизимни тўғрилаш.

Папка тузинг, масалан, **My_Delhp** номли. **My_Delhp** папкаси ичида яна ўз дастурнгизни сақлаш учун папка очиш, масалан Pgm_1.

Delphi муҳитининг стандарт настройкасига ўзгартириш киритиш учун Tols=>Environment Options меню буйруғини бериш ва мулоқат дарчасидан кракли ўзгаришларни бажариш лозим.

Delphi дастурлаш муҳитида ишлаш жараёнида қуйидаги кенгайтмали файллар ишлатилади:

-лойиҳа файли, кенгайтмаси **.dpr**;

-паскал модули файли, кенгайтмаси **.pas**;

-компоненталар жойлашган файл, кенгайтмаси **.dcu**;

-формалар жойлашган файл, кенгайтмаси **.dfm**;

-маълумотлар базаси файли, кенгайтмаси **.dbf**.

Тайёрланадиган Delphi дастур учта асосий этапдан ўтади:

-компиляция;

-компоновка;

-бажариш.

Компиляция этапида тайёрланган дастур матни Object Pascal тилига ўтказилади. Компановка этапида эса керакли қўшимча ёрдамчи дастурлар ва остдастурлар унга бирлаштирилади. F9 тугмасини босиш билан Save UnitAs диалог ойнаси пайдо бўлади ва сиздан Unit.pas модули учун файл номини ва жойлашадиган папкани кўрсатишингизни сўрайди. Агар жой кўрсатилмаса Delphi автоматик равишда дастурнгизни Bin папкасига жойлаштиради. Яхшиси сиз бу папкани ўз ишчи папкангиз номига алмаштиринг, масалан My_Delph. Дастур

компиляция қилиниши пайтида Delphi системаси pas, dfm ва dcu кенгайтмали модуллар тузади. .pas кенгайтмали файл кодларни ёзиш ойнасида киритилган дастур матнини, .dfm форма ойнаси ташкил этувчиларини, .dcu кенгайтмали файл эса .pas ва .dfm кенгайтмали файлларнинг биргаликдаги машина кодига ўтказилган вариантини сақлайди. Бу .dcu кенгайтмали файл комплятор тамонидан ташкил қилинади ва ягона ишчи (бажарилувчи) .exe кенгайтмали файл ташкил қилишга база яратади.

Delphi лойиҳаси структураси

Delphi дастури - бу бир неча бир бири билан боғлиқ файллардир. Ҳар қандай дастур .dpr кенгайтмали лойиҳа файли ва бир ёки бир неча .pas кенгайтмали модуллардан ташкил топади. Лойиҳа файли дастурчи тамонидан киритилмайди, у фойдаланувчининг кўрсатмалари асосида автоматик равишда Delphi системали дастури тамонидан тузилади. Лойиҳа файли матнини кўриш учун Project/View Source буйруғини бериш зарур. Лойиҳа матни умумий ҳолда қуйидагича бўлиши мумкин.

```
Program Project1;
        Uses
           Forms,
           Unit1 in 'Unit1.pas' {Form1}
        {$R *.res}
 Begin
        Application.Initialize;
        Application.CreateForm(Tform,Form1);
        Application.Run;
 End.
```

Лойиҳа номи дастурчи томонидан лойиҳа файлини сақлаш вақтида берилади, ва у Delphi муҳитида бажарилувчи файл, яъни кенгайтмаси .exe бўлган файлни

ташкил қилишни аниқлайди. Лойиҳа файлидан кейин ишлатиладиган модуллар: стандарт модуллар Forms ва Unit1 жойлашади. **{$R *.res}** директиваси компиляторга ишлатилиши керак бўлган русурс файллари, масалан дастурларни элон қилиш кераклигини билдиради. Юлдузча белгиси ресурс файлининг кенгайтмаси .res эканлигини билдиради. Бош модулнинг бажарилувчи қисми Begin .. End операторлари орасига жойлашади.

Модул - бу бирор бир дастур. Модуллар стандарт конструкцияга эга. Object Pascalда модул структураси умумий ҳолда қуйидаги кўринишда бўлади:

Unit <Модул номи>
Interfase
. . . . . .
Implementation
. . . . . .
Initialization
. . . . . .
Finalization
. . . . . .
End.

Delphi тизимини ишга туширгандан кейин модул структураси қуйидаги кўринишда бўлади.

```
Unit unit1;

Interface

Uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls,
  Forms, Dialogs;
```

```
Type
 TForm1 = class(TForm)
 Private
        { Private declarations }
 Public
        { Public declarations }
 end;

Var
 Form1: TForm1;

 Implementation
 {$R *.dfm}

End.
```

1.3.Ўзгармаслар, ўзгарувчилар ва стандарт функциялар

Ҳақиқий турдаги сонлар умумий ҳолда қуйидаги кўринишда бўлади:

s $a_1a_2...a_n$. $b_1b_2...b_k$

Бу ерда s ишора (+ ёки -) ёки буш жой; $a_1a_2...a_n$ бутун қисм; $b_1b_2...b_k$ каср қисм. Масалан: +3,147 сони +3.147 ёки 3.147

-143,03 сони -143.03
57,0 сони 57.0
0,493 сони 0.493 ёки .493

Ҳақиқий сонларнинг ўзгариш диапазони компьютернинг турига караб турлича бўлади. $10^{-38}<x<10^{+38}$ х-ихтиёрий сон. Улар экспоненциал (даражали) кўринишда ифодаланиши ҳам мумкин, яъни $\pm m10^{\pm n}$. Бундай сонлар қуйидагича ёзилади $\pm mE\pm n$. Масалан:

$0{,}43 \cdot 10^{-6}$ .43E-6
0,0003 3E-4

Бутун сонлар умумий ҳолда қуйидагича ёзилади s $a_1a_2...a_n$.
Масалан: +345 сони +345 ёки 345
-106 сони -106

Бутун сонлар ўзгариш диапозони -32768 дан +32767 гача. Агар бутун сон қиймати бу дипазондан чиқса, у ҳақиқий сон шаклида ифодаланади ёки компьютер турига қараб, у ўнолтилик саноқ системасида ифодаланиши ҳам мумкин. Белгилилар қўштирноқ ичида ёзилади. Ёзилиш диапазони 0 дан 255 тагачадир. Мисол. "Паскаль", "405.5"

Паскал тилида идентификатор тушунчаси мажуд бўлиб, дастурда объектларни номлашда ишлатилади. Ўзгармасларни, ўзгарувчиларни, белги(метка), процедура ва функцияларни белгилашда ишлатилган ном **идентификаторлар** дейилади. Идентификаторлар лотин алфавити ҳарфларидан бошланиб қолган ҳарфлари белги ёки рақам кетма-кетлигидан ташкил топган бўлиши мумкин. Масалан: xx, xx1, alfa&.

Delphi тилида дастур ишлаши мобайнида қиймати ўзгармайдиган идентификаторлар **ўзгармаслар** дейилади ва улар дастурнинг бош қисмида **Const** сўзи билан эълон қилиниб, унга аниқ қиймат тенглаштирилади.

Мисол. Const aa1=2.27;
Pi=3.14;
radius=14;

Дастур ишлаши мобайнида қийматлари ўзгариши мумкин бўлган идентификаторга **ўзгарувчилар** дейилади ва улар дастур бош қисмида **Var** сўзи билан эълон қилинади. Ўзгарувчилар номи келтирилиб, уларнинг турлари берилади. Ўзгарувчиларнинг энг кўп ишлатиладиган турлари **бутун, ҳақиқий, белгили, қатор** ва **мантиқий**дир. Улар мос равишда бутун - **Integer**, ҳақиқий - **Real**, белгили - **Char**, қатор (матн) - **String** ва мантиқий - **Boolean** деб ёзилади.

Масалан: Var a, d1, alfa : Integer;
c121, df : Real;
Etx, xx : Char;
St,Sw: String;
fl : Boolean;

Мантиқий ўзгарувчилар фақат иккита қиймат қабул қилади: "True" (чин) ва "False" (ёлғон).

Стандарт ва ностандарт математик функциялар

Функция номи	Тилда ёзилиши	Маъноси
Sinx	SIN(x)	x нинг синуси
Cosx	COS(x)	x нинг косинуси
Lnx	Ln(X)	x нинг натурал логарифми
e^x	EXP(x)	Експонента
$\sqrt{x}$	SQRT(x)	Квадрат илдиз
Arctgx	ARCTAN(x)	x нинг арктангенси
$\|x\|$	ABS(x)	x нинг модули
x^2	SQR(x)	x нинг квадрати
a^b	EXP(b*LN(a))	a нинг b чи даражаси

Ностандарт математик функциялар.

$1.Secx = \frac{1}{Sinx}$; $2.Co\sec x = \frac{1}{Cosx}$; $3.Tgx = \frac{Sinx}{Cosx}$; $4.Arcctgx = Arctg\frac{1}{x}$;

$5.Arc\sin x = Arctg\frac{x}{\sqrt{1-x^2}}$; $6.Arc\cos x = Arctg\frac{\sqrt{1-x^2}}{x}$; $7.Arc\sec x = A$

$8.\ Arc\cos ecx = Arctg\sqrt{1-x^2}$; $9.Log_a b = \frac{Lnb}{Lna}$; $10.Padian = \frac{Gradius \cdot \pi}{180}$

Ўзгартириш функциялари

Функция	Қиймати
Chr(n)	Коди n га тенг символ
IntToStr(k)	Бутун k ни тасвирловчи сатр
FloatToStr (n)	Ҳақиқий n тасвирлови сатр
FloatToStrF(n, f , k,m)	Ҳақиқий n тасвирлови сатр. Бунда: f - форма аниқлик; m - каср қисмидаги рақамлар сони
StrToInt (s)	Сатрни бутун сонга ўтказиш
StrToFloat (s)	Сатрни ҳақиқий сонга ўтказиш
Round (n)	Ҳақиқий сонни яхлитлаш
Trunc (n)	Ҳақиқий сон каср қисмини олиб ташлаш
Frac(n)	Касрли соннинг каср қисми

Int (n)	Каср соннинг бутун қисми

Дастурда арифметик ва мантиқий ифодалар ўзгарувчи, ўзгармас, стандарт функциялар, қавслар ва амал белгилари орқали ташкил қилинади.

Ифодаларда ҳисоблашлар тартиби қавслар ичидаги ифодалар бажарилгандан кейин қуйидаги тартибда бажарилади:

1.NOT амали;

2. *, /, DIV, MOD, AND;

3.+, -, OR;

4.таққослаш белгилари: <, >, <=, >=, <>, =, IN.

Ифодадаги амал натижаси қандай турда бўлиши амалларда қатнашаётган ўзгарувчиларнинг турларига боғлиқ. Агар иккита ўзгаручининг тури Integer ёки Real бўлси, амал натижаси ҳам Integer ёки Real бўлади. Агар бири Integer иккинчиси Real бўлса натижа Real бўлади. NOT, OR, AND ва таққослаш амалларининг натижалари эса Boolean турида бўлади.

Компьютер фойдаланувчи томонидан қўйилган масалани аниқ ва тушунарли кўрсатмалар берилгандагина бажара олади. Бу кўрсатмалар маълум бир маънони англатувчи сўзлардан иборат бўлиб, компьютерга қандай операцияни бажариш лозимлигини билдиради ва бу кўрсатмаларга **операторлар** дейилади. Операторлар дастур ишлаганда кетма-кет равишда бажарилади. Delphi тилида бир сатрга бир неча операторларни ёзиш умкин.

Delphi тилида дастур матни бош ва асосий бўлимдан ташкил топади. Бош бўлим дастур номи ва ўзгарувчилар, ўзгармаслар, массивлар, белгилар(меткалар), процедуралар ва функцияларни тавсифлашдан иборат бўлади. Асосий бўлим дастур танаси дейилиб, унда дастурда бажариладиган ҳамма операторлар кетма-кетлиги берилади ва у Begin (бошламоқ) сўзи билан бошланиб End (тугаш)

сўзи билан тугайди. Умумий ҳолда дастур структураси қуйидаги кўринишга эга:

Program <дастур номи>;
Uses <Фойдаланадиган библиотекалар (модуллар) рўйхати>;
Label <Ишлатиладиган белгилар(меткалар) рўйхати>;
Const <Ишлатиладиган ўзгармасларни аниқлаш>;
Type <Янги турларни аниқлаш>;
Var <Ўзгарувчиларни эълон килиш>;
<Процедура ва функцияларни аниқлаш>
Begin
<Бажариладиган операторлар кетма кетлиги>
End.

1.4.Маълумотлар турлари

Маълумотлар турларини Delphi тилида умумий ҳолда иккига ажратиш мумкин:

- стандарт турлар. Бу турлар олдиндан Delphi тили томонидан аниқланган бўлади;
- дастурчи томонидан киритиладиган (аниқланадиган) турлар.

Стандарт турлар таркибига қуйидагилар киради: бутун, ҳақиқий, белгили (символ), қатор (строк), мантиқий, кўрсатгичли ва variant.

Дастурчи турларни дастурнинг **Var** бўлимида ўзгарувчиларни тавсифлашда аниқлайди ёки махсус турларни аниқлаш учун бўлим бўлган -турларни тавсифлаш **Type** бўлимида аниқлайди.

Бу бўлим умумий ҳолда қуйидагича бўлади.

Type
<тур номи>=<турнинг тавсифи>;

Мисол:

```
Type
        TColor=(Red,Blue,Black);
Var  Color1,Color2,Color3: TColor;
```

Type бўлимида дастурчи томонидан янги Tcolor номли тур киритилмоқда ва у Red,Blue,Black мумкин бўлган қийматларни қабул қилиши мумкин.

Var бўлимида дастурчи томонидан тури аниқланган учта Color1,Color2,Color3 ўзгарувчилар тавсифланмоқда.

Бу ўзгарувчиларни тўғридан тўғри қуйидагича ҳам тавсифлаш мумкин.

```
Var  Color1,Color2,Color3: (Red,Blue,Black);
```

Стандарт турларни Type бўлимида тавсифлаш шарт эмас, уларни тўғридан тўғри Var бўлимида тавсифлаш мумкин.

Delphiда стандарт турларни қуйидагича классификация қилиш мумкин.

- Оддий
  - Тартибли
    - Бутун
    - Белги
    - Мантиқий
    - Саноқли (Перечисляемый)
    - Чегараланган
  - Ҳақиқий
- Қатор
- Структура
  - Тўплам
  - Массив
  - Ёзув
  - Файл
  - Класс
  - Интерфейс
- Кўрсатгичли
- Процедурали

❖ Variant

Оддий турларга тартиблашган ва ҳақиқий турлар киради. Тартиблашган турлар шу билан характерланадики унинг ҳар бир қиймати ўзининг тартибланган номерига эга. Ҳақиқий тур қийматлари каср қисмидан иборат бўлган сонлардан иборатдир.

Тартиблашган турларга бутун, белгили, мантиқий, саноқли ва чегараланган турлар киради.

Бутун турлар. Бутун турлар бутун сонларни тасвирлаш учун ишлатилади. Жадвалда Delphi 7 да ишлатиладиган бутун турлар рўйхати келтирилган.

Тур	Ўзгариш диапазони	Ў
Integer	-2147483648..2147483647	
Cardinal	0..4294967295	
Shjrtint	-128..127	
Smallint	-32768..32767	
Longint	-2147483648..2147483647	
Int64	$-2^{63}..2^{63}-1$	
Byte	0..255	
Word	0..65535	
LongWord	0..4294967295	

Ҳақиқий турлар. Ҳақиқий турлар ҳақиқий сонларни тасвирлаш учун ишлатилади. Жадвалда Delphi 7 да ишлатиладиган ҳақиқий турлар рўйхати келтирилган.

Тур	Ўзгриш диапазони	Ў
Real	$5.0*10^{-324}..1.7*10^{308}$	
Real48	$2.9*10^{-39}..1.7*10^{38}$	
Single	$1.5*10^{-45}..3.4*10^{38}$	
Double	$5.0*10^{-324}..1.7*10^{308}$	
Extended	$3.6*10^{-4951}..1.1*10^{4932}$	

Comp	$-2^{63}+1..2^{63}-1$

Белгили турлар. Маълумотларнинг белгили турлари фақат битта белгини сақлаш учун хизмат қилади. Жадвалда Delphi 7 да ишлатиладиган белгили турлар рўйхати келтирилган.

Тур	Ўлчам (байтда)
Char	1
ANSChar	1
WideChar	2

Мантиқий турлар. Мантиқий турлар чин (True) ёки ёлғон (False) қийматнинг бирини қабул қилади. Жадвалда Delphi 7 да ишлатиладиган мантиқий турлар рўйхати келтирилган.

Тур	Ўлчам (байтда)
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Дастурчи томонидан киритилувчи турлар

Delphi тили дастурчига узининг турларини киритишга имкон беради.
Бу турлар стандарт турларга еки аввал киритилган турларга асосланган булиб куйидаги турларга тегишли булиши мумкин:

- сановчи;
- интервал;
- мураккаб тур (езув).

Саноқли турлар. Саноқли турлар тартибланган қийматлар тўпламини ишлатади.

Тур =(1 Киймат, 2 Киймат, ... ,I Киймат)

Масалан:

```
Type
   Color=(black,green,yellow blue,red,white);
   Fam=(Petrov,Sidorov,Raximov,Sobirov);
   DayOfWeek=(mon,tue,wed,thu,fri,sat,sun);
```

Бу ерда

Color саноқ тури бешта ранглар кетма кетлигини аниқлайди.

Fam саноқ тури тўртта фамилияни аниқлайди.

DayOfWeek саноқ тури ҳафта номларини аниқлайди.

Одатда Delphi тилида турлар номлари T харфидан бошланади (Type — тип сўзидан).

Янги тур таърифлангандан сунг шу турга тегишли ўзгарувчини таърифлаш мумкин, масалан:

```
type

TDayOfWeek = (MON,TUE,WED,THU, FRI,SAT,SUN) ;

var

ThisDay, LastDay: TDayOfWeek;
```

Сановчи тур таърифи қийматлар ўзаро муносабатини кўрсатади. Энг чап элемент минимал, энг унг элемент максимал ҳисобланади. Юқорида киритилган DayOfWeek тури элементлари учун қуйидаги муносабат ўринли:

MON < TUE < WED < THU < FRI < SAT < SUN

Сановчи тур элементлари орасидаги муносабат ўзгаруквчиларни бошқарувчи инструкцияларда қўллашга имкон беради, масалан:

```
if (Day = SAT) OR (Day = SUN) then

begin

{ агар кун шанба ёки якшанба бўлса бажарилсин }

end;
```

Бу инструкцияни қуйидагича ёзиш мумкин:

```
if Day > FRI then begin

{ агар кун шанба еки якшанба бўлса бажарилсин }

end;
```

Сановчи тур таърифи номланган константалар киритишнинг қисқартирилган шакли деб қараш мумкин. Мисол учун TDayOfWeek турининг таърифи қуйидаги таърифларга тенгдир:

```
const

MON=0; TUE=1; WED=2; THU=3; FRI=4; SAT=5; SUN=6;
```

Интервал (диапазон) тури. Интервал (диапазон) тури бериладиган қийматга чегара қўйади.

```
Type
    <тур номи>=<минимал>..<максимал>;
```

Масалан:

```
Type
  Color=red..green; // ранга чегара
  Digit=0..9; //бутун сонларга чегара
  Symb='A'..'Z'; //  ҳарфларга чегара
```

Ҳақиқий турларга чегара қўйилмайди.

Интервал тур таърифида номланган константалардан фойдаланиш мумкин. Қуйидаги мисолда интервал тур TIndex таърифида HBOUND номланган константадан фойдаланилган:

```
const

HBOUND=100;

type

TIndex=l..HBOUND;
```

Интервал турдан массивларни таърифлашда қулайдир:

```
type

TIndex =1 .. 100;

var

tab1 : array[TIndex] of integer; i:TIndex;
```

Бутун сон туридан ташқари асос тур сифатида сановчи турдан фойдаланиш мумкин. Қуйидаги дастур қисмида TMonth сановчи тур асосида интервал тур TSammer таърифланган:

type

TMonth = (Jan, Feb, Mar, Apr, May, Jun,

Jul, Aug, Sep, Oct, Nov, Dec);

TSammer = Jun.. Aug;

Ёзув

Дастурлаш амалиётида стандарт маълумотлардан ташкил топган мураккаб маълумотлар билан ишлашга тўғри келади. Мисол учун талаба тўғрисидаги маълумот исми шарифи, тугилган йили, адреси, курси, гуруҳи ва ҳоказолардан иборат бўлиши мумкин. Бундай маълумотларни таърифлаш учун Delphi да ёзув (record) лардан фойдаланилади.

Ёзув бу - алоҳида номланган ҳар хил турли компонеталардан иборат мураккаб турдир.

Ҳар қандай тур каби, "ёзув" type бўлимида таърифланиши лозим. Бу таъриф умумий кўриниши:

Ном = record

1_ Майдон: 1_Тип; 2_ Майдон: 2_Тип;…; K_ Майдон: K_Тип; end;

Таърифларга мисоллар:

type
TPerson = record
f_name: string[20];
l_name: string[20];

day: integer;
month: integer;
year: integer;
address: string[50]; end;

TDate = record
day: integer; month: integer; year: integer;
end;

Ёзув туридаги ўзгарувчини қуйидагича таърифлаш мумкин:
var
student : TPerson; birthday : TDate;

Ёзув элементига (майдонига) мурожаат қилиш учун ёзув номи ва нуқтадан сўнг майдон номини кўрсатиш керак. Масалан:

Writeln('Имя: ', student.f_name + #13 + 'Адрес: ', student.address);

Инструкция экранга student ўзгарувчи-ёзувнинг f_name (ном) ва address (адрес) майдонларини чиқаради.

Баъзида ўзгарувчи-ёзув тури ўзгарувчилар эълон қилиш бўлимида эълон қилинади. Бу ҳолда ёзув тури ўзгарувчи номидан сўнг кўрсатилади. Мисол учун student ёзуви var бўлимида қуйидагича таърифланиши мумкин:

student: record
f_name:string[20];
l_name:string[20];
day:integer;
month:integer;
year:integer;

```
address:string[50];
end;
```

With инструкцияси

With инструкцияси дастурда майдонлар номларини ўзгарувчи – ёзув номини кўрсатмасдан ишлатишга имкон беради. Умумий ҳолда with инструкцияси қуйидаги кўринишга эга:

```
with ном do
begin
( дастур инструкцияси} end;
```

Мисол учун дастурда қуйидаги ёзув таърифланган бўлсин

```
student: record
f_name: string[30];
l_name: string[20];
address: string[50];
end;
```

ва студентлар тўғрисидаги маълумотлар E1, E2 ва E3 ўзгарувчиларда жойлашган бўлсин. У ҳолда

```
student.f_name := El;
student.l_name := E2;
student.address := E3;
```

инструкциялар ўрнига қуйидаги инструкцияни ёзиш мумкин:

```
with student do begin
```

```
f_name := E1; l_name := E2; address := E3;
end;
```

С а в о л л а р

1.Алгоритм нима ва у қандай хоссаларга эга?
3.Алгоритмнинг қандай турлари мавжуд?
4.Қандай алгоритмик тиллар бор?
5.Дастур ва дастурлаш нима?
6.Масалани ЭҲМда ечиш қандай босқичлардан иборат?
7.Ўзгармас ва ўзгарувчилар дастурда қандай тавсифланади?
8.Ўзгарувчиларнинг қандай турлари мавжуд?
9.Мантиқий ўзгарувчилар қандай қиймат қабул қилади?
10.Қандай стандарт математик функциялар мавжуд?
11.Pascal тилида дастур қандай структурага эга?
12.With инструкцияси дастур тузишда қандай имконият яратади?
13.Ёзув қандай тур?
14.Маълумотлар турларини Delphi тилида умумий ҳолда қандай турларга ажратиш мумкин?
15.Дастурчи томонидан киритилувчи турлар қандай турлар?

М а ш қ л а р

1.Pascal тилидаги 3E-4 ва 0.2E5 сонларини оддий ёзувда ёзинг.
2.1000000 ва 0.0000001 сонларини экспоненциал сонлар шаклида ёзинг.
3.Қийматларни тасвирлаш учун қайси турдаги ўзгарувчилар ишлатилади?
a) нарсалар сонини аниқлашда;
b) тенглама коеффициентлари;

c) иккита сон катта-кичиклигини аниқлашда;
d) махсулот номлари;
e) ўртача температура;
f) бир йилдаги дам олиш кунлари.

4.Қуйидаги тавсифланган ўзгарувчилар қандай қиймат қабул қилиши мумкинлигини топинг.

Var a, b: Integer; c: Real; w: String; f: Boolean;

Ж а в о б л а р

1. 0.0003 ва 20000.0
2. 1E6 ва 1E-7
3. a)Integer; b)Real ; c)Bollean ; d)String ; e)Real; f)Integer.
4. а ва б -бутун; с -ҳақиқий; w -матн қатори; f –мантиқий.

Типик алгоритмларни дастурлаш асослари

Оддий алгоритмларни даструрлаш

2.1.Маълумотларни киритиш ва чиқариш операторлари

Бирор бир масалани ечишнинг чизиқли бўлган алгоритмига дастур тузишда алгоритмдаги келтирилган кетма-кетликлар асосида операторлар ёзилади. Бундай дастурларни тузушда асосан ўзгарувчилар қийматини киритиш, натижаларни чиқариш ва шу билан бирга ўзлаштириш операторлари ишлатилади.

Дастурдаги ўзгарувчилар қийматларини дастур ичида ўзлаштириш оператори ёрдамида ҳам бериш мумкин. Лекин дастурда ўзгарувчи қийматини ташқаридан киритиш қулайлик туғдиради ва умумийликни таъминлайди.

Read оператори ўзгарувчилар қийматларини экрандан компьютер хотирасига киритиш учун ишлатилади. У қуйидаги кўринишларга эга.

Read(c1,c2,...,cn);
Readln(c1,c2,...,cn);
Readln;

бу ерда c1,c2,...,cn - ўзгарувчилар номи; ln - қўшимчаси қийматни киритиб кейинги қаторга ўтишни билдиради.

Мисоллар: Read(Sm1,Sm2);
Readln(x1,x2,x3);
Readln;

Бу ерда биринчи оператор Sm1 ва Sm2 ўзгарувчилар қийматини экрандан киритади. Иккинчи оператор эса x1,x2,x3 ўзгарувчилар қийматини экрандан киритади ва киритишни кейинги қаторга ўтказади. Охирги оператор эса киритишни кутади ва қатор ўтказади.

Write оператори оддий маълумотларни ва ўзгарувчилар қийматларини компьютер экранига чиқариш учун ишлатилади. У қуйидаги кўринишларга эга.

Write(c1,c2,...,cn);

Writeln(c1,c2,...,cn);

Writeln;

бу ерда c1,c2,...,cn - оддий матнлар ёки ўзгарувчилар номи; ln - қўшимчаси чиқаришни кейинги қаторга ўтишни билдиради.

Мисоллар: Write(Summa);

Write('Натижа йук');

Write('Тенглама ечими x1=', x1, 'x2=', x2);

Оддий маълумотларни чиқаришда улар матн деб қаралади ва у қўштирноқ ичида ёзилади. Чиқариш оператори ёрдамида ўзгарувчилар қийматини формат кўринишда ҳам бериш мумкин:

Write(c:m:n);

бу ерда c-ўзгарувчи; m-шу ўзгарувчи қиймати узунлиги; n-қийматнинг каср қисми ва унда n-1<m бўлиши керак.

Мисол. Write(x:8:4);

Агар x=155.01021 бўлса, қуйидаги ёзув чиқади 115.0102.

Write('Махсулот сони:', kol:5);

Агар kol=15 бўлса, қуйидаги ёзув экранга чиқади,

Махсулот сони: 15

Дастур матнини тушунтириш мақсадида кўпинча дастурда изоҳлар келтирилади. Дастурда изоҳлар исталган жойда берилиши мумкин. Изоҳ катта қавс ичида ёзилади.

Масалан: { Бу матн дастурга изоҳ беради }

{ Бу жойда ечим аниқланмоқда }

Дастурда маълум ҳисоблашларнинг натижаларини бирор бир ўзгарувчида сақлаш учун ўзлаштириш (юбориш)

оператори ишлатилиб, у «**:=**» белгиси ёрдамида қиймат юборилиши керак бўлган ўзгарувчидан кейин қўйилади.

Масалан: i:=0; i-қиймати нолга тенглашади, яъни і ўзгарувчига нол юборилади деб тушунилади. Бунда машина і ўзгарувчи учун ажратилган хотирасига нол ёзиб сақлайди.

Мисол: B:=5; C:=4; A:=(B+C)/2;

Бу ерда, агар А бутун идентификатор бўлса, унинг қиймати 4 га, акс ҳолда эса 4.5 қийматга эга бўлади.

Чизиқли структурали алгоритмларни дастур шаклида ёзиш учун олдин ишлатиладиган ўзгарувчилар руйхати келтирилиб, кейин алгоритмдаги бажарилишлар кетма-кет равишда амалга оширилиши керак.

Мисол: Текисликдаги икки нуқта орасидаги масофани топиш дастури.

```
Program XY;
 Var
    x1,y1,x2,y2,d: Real;
  Begin
        Write('Нукта координаталарини киритинг:');
        Read(x1,y1,x2,y2);
        d:=Sqrt(Sqr(x1-x2)+Sqr(y1-y2));
        Writeln;
        Writeln('Нукта координаталари:',x1,y1,x2,y2);
        Writeln('Масофа=',d);
        Readln;
   End.
```

2.2.Delphiда консол иловасини яратиш

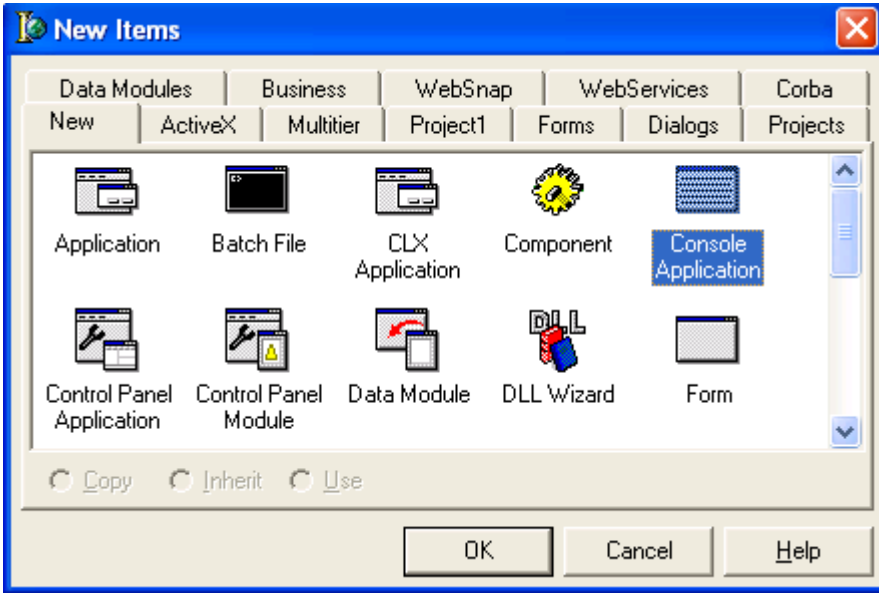
Delphiда консол иловаларини ҳар хил усулларда яратиш мумкин. Улардан энг осон усули қуйидагича:

1.Delphi муҳити ишга туширилади.

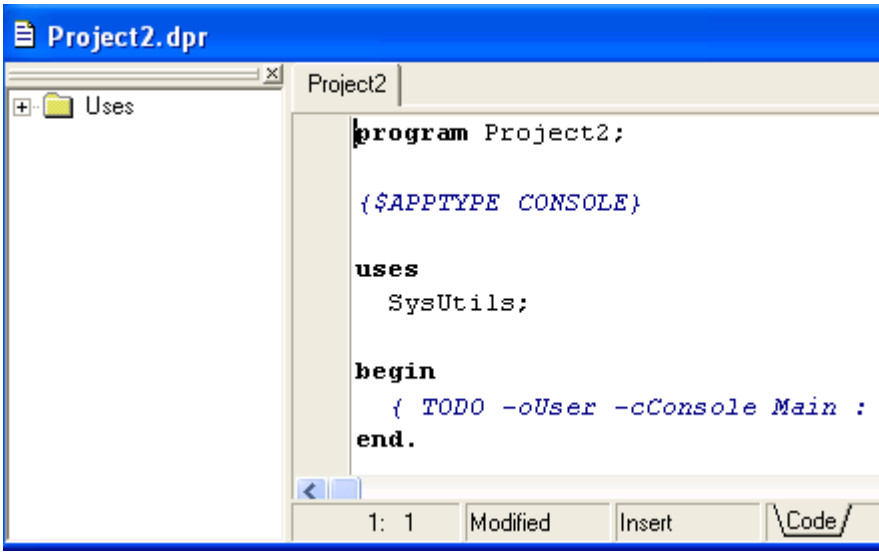
Пуск=>Программы=>Borland Delphi

2.Бош менюдан File пунктини очиб у ердан New, кейин эса Other буйруқлари берилади. **File=> New=> Other**

3.Форма ва лойиҳаларни сақлаш учун очилган махсус ойнадан (бу ойнага Delphi архив ойнаси дейилади) “Console Application” пиктограммаси танланади ва Ok тугмаси босилади.



4.Натижада экранда лойиҳа ойнаси очилади (.dpr кунгайтмали ном билан).



Begin – end ичига олинган

{ TODO -oUser -cConsole Main : Insert code here }

изоҳ ўрнига лойиҳа файлининг дастур матни киритилади.

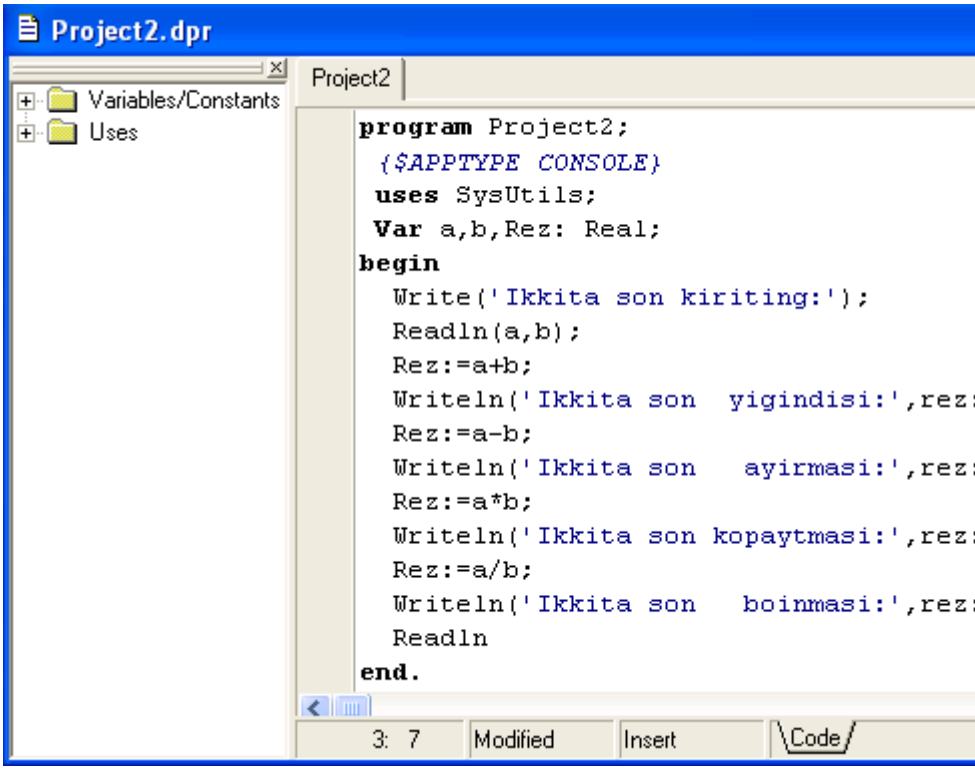
Тузилган дастурни ишга туширишдан олдин уни сақлаш керак бўлади. Уни сақлаш file=>Save All буйруғини бериш лозим. Ҳар бир лойиҳа алоҳида янги папкага сақланишни тавсия беради. Лойиҳа файлини сақлашда алохида курсатилмаган ҳолатида ProjectN.dpr номли файл номини тавсия қилади. Бу ерда N ҳар бир кетма кет номланадиган лойиҳа номери (сон, масалан 1,2,3,..). Лекин биз лойиҳа файлини исталган ном билан сақлашимиз мумкин. Масалан MyProgram.dpr. Бу ном автоматик равишда чиқади.

Лойиҳани сақлаб бўлгандан сўнг, уни бажаришга берамиз. Бунинг учун бош менюдан қуйидаги буйруқни бериш лозим: Run=>Run ёки F9 функционал тугмачасини

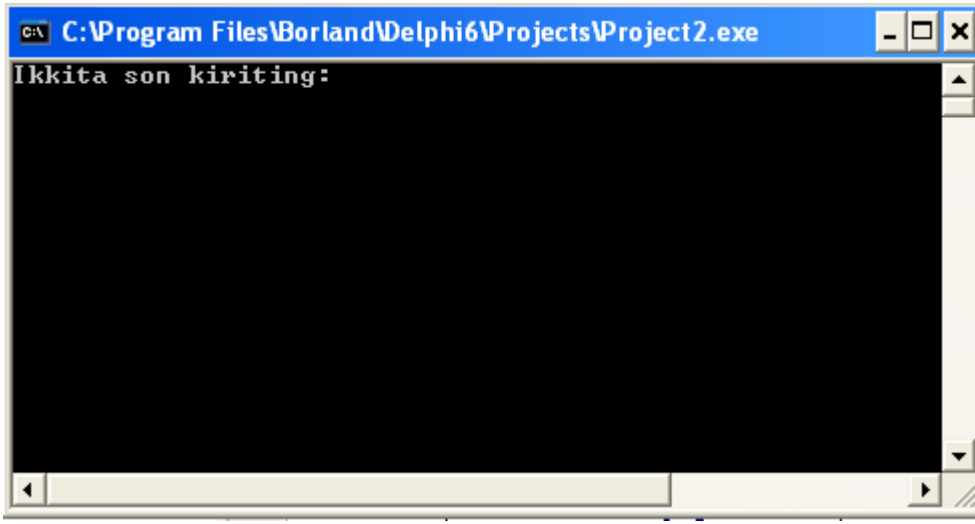
босиш керак бўлади. Дастур нармал ишга тушгандан сўнг экранда DOSнинг стандарт дастур ойнаси намаён бўлади.

Мисол. Иккита соннинг йиғиндиси, айирмаси, кўпайтмаси ва бўлинмасини ҳисоблаш дастурини яратинг.

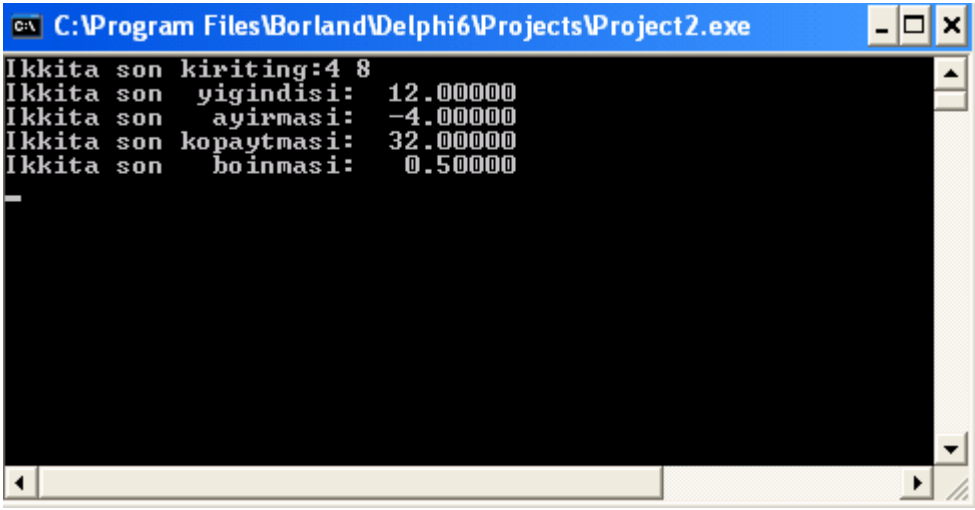
Бу мисолни ечиш учун юқорида келтирилган тўртта кетма кетликни бажарамиз ва дастур кодини киритамиз.



Дастур коди киритилгандан сўнг уни сақлаб кейин ишга туширамиз. Натижада экранда Dos ойнаси очилиб унда “Ikkita son kiriting:” сўзи чиқади. Кейин иккита сон кирииб Enter тугмасини босиш керак бўлади.



Натижада қуйидаги жавоблар чиқади.



5.Кейин Обект инспектори (Object Inspector) ва Объект дарахтлар (Object TreeView) ойналари ёпилади.

6.Бош менюдан Project=>View Sourse буйруғи берилади.

Энди Delphiда консол иловасини яратишнинг иккинчи усулини кўриб чиқамиз:

1.Delphi муҳити ишга туширилади.

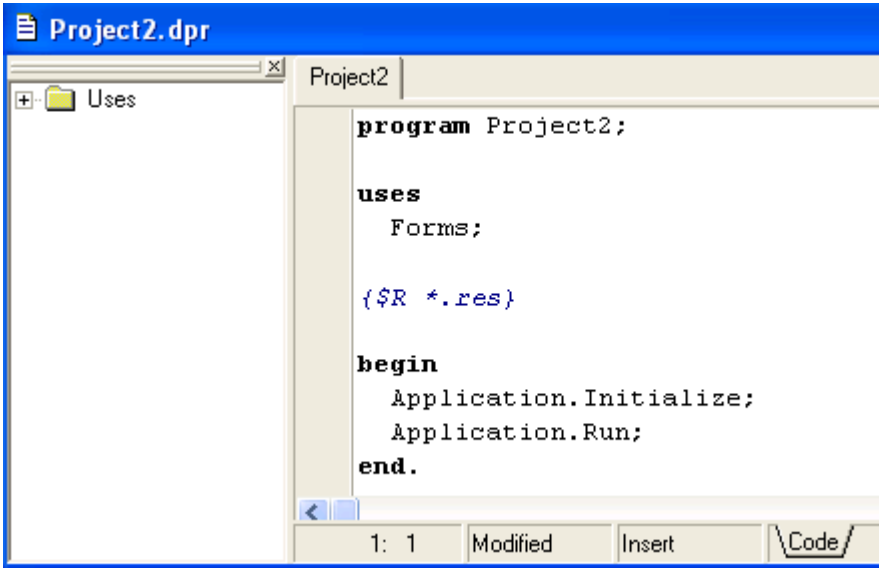
Пуск=>Программы=>Borland Delphi

2.Бош менюдан File пунктини очиб у ердан New, кейин эса Application буйруқлари берилади. **File=> New=> Application**

3.Форма ойнаси ёпилади.

4.Дастур кодини ёзиш (модул) ойнаси ёпилади. Ёпиш вақтида “Save changes to Unit1.pas?” (“Unit1.pasдаги ўзгаришлар сақлансинми?”) сўров ойнаси чиқади. У ердан “NO” (Йўқ) буйруғи берилади.

Натижада қуйидаги Project1.drp лойиҳа файли ойнаси экранга чиқади.



7.Лойиҳа файли агар лозим бўлса бошқа ном билан сақланади.

Бу ойнадан Program, Uses, Begin ва End калит сўзлари қолдирилиб бошқалари ўчирилади ва кейин дастур матн кодлари киритилади.

Агар лойиҳа файли ёзилган папка ичи қаралса унда қуйидаги файллар рўйхатини кўрамиз.

- MyProgram.dpr -лойиҳа файли (бош лойиҳа модули);
- MyProgram.exe -илова файли ёки бажарилувчи файл. Бу файл компилятор ёрдамида, яъни компиляция жараёнида, агар дастурда синтактик хатоликлар бўлмаса тузилади. Бошқача сўз билан айтганда, агар сизга ўз дастурингизни ишга тушуриш мумкин бўлса, масалан F9 тугмасини босиш билан бажарилувчи файл автоматик равишда тузилади. Бажарилувчи файл автоном файл бўлиб унинг учун бошқа файл ёки бирор дастурий система мавжуд бўлиши шарт эмас. Уни сиз ишга туширишингиз мумкин бошқа дастурлар каби, масалан Paint, Блокнот ёки ўйин дастурларини ишга тушургандай;
- MyProgram.cfg -лойиҳа конфигурацияси файли;
- MyProgram.dof -лойиҳа опция файли. Унда дастурнинг тўғри ишлаганлиги ҳақида ахборотлар сақланади,

Лойиҳа опция ва конфигурацияси файллари лойиҳа файлининг тузилиши билан бир вақтда Delphi томонидан автоматик равишда тузилади. Кўп ҳолларда юқорида келтирилган файллардан ташқари яна .dpr кенгайтмали файл ҳам тузилади. Бу файл лойиҳа файлининг (резерв файли) нусхаси бўлиб ҳисобланади. Масалан, MyProgram.-dpr. Бу файл лойиҳа файли тузилиши даврида бир вақтнинг ўзида тузиб борилади. Агар асосий лойиҳа файлида бузилиш ёки учирилиш содир бўладиган бўлса, у ҳолда уни MyProgram.-dpr файлидан тиклаш мумкин. Бунинг учун кенгайтма олдидаги “-” белгини олиб ташлаш кифоя.

Тармоқланувчи жараёнларни дастурлаш

Шартли ўтиш оператори

Тармоқланувчи жараён алгоритмида маълум жараён бажариш маълум шарт асосида у ёки бу тармоққа ажратилади, яъни маълум кейинги бажариладиган жараён танланади.

Object Pascal тилида шарт - бу мантиқий турдаги ифода бўлиб, у фақат «чин»(True) ёки «ёлғон»(False) қийматни қабул қилади.

Қуйидаги мантиқий белгилар ишлатилади: >, <,<=,>=,<>,=. Буларга муносабат амаллари дейилади.

Қуйидаги мантиқий амаллар ишлатилади:

- NOT-«инкор»;
- AND-«мантиқий ва»;
- OR-«мантиқий ёки».

Бу мантиқий амалларнинг бажарилиш натижалари қуйидагича:

A	B	A AND B	A OR B	NOT
False	False	False	False	Tru
False	True	False	True	Tru
True	False	False	True	Fals
True	True	True	True	Fals

Масалан: (5<6) AND (6<50) -мантиқий ифода рост (True),

(20>0) OR (20<0.5) -мантиқий ифода рост (True),

(10<8) AND (10<15) -мантиқий ифода ёлҚон (False),

NOT(100>3) -мантиқий ифода ёлҚон (False).

Мантиқий ифодаларни бирор бир мантиқий ўзгарувчига ўзлаштириш ҳам мумкин.

Масалан:

F:=(A<B) AND (A<C);

Бу ерда, агар иккала шарт бажарилгандагина F мантиқий ўзгарувчи “чин” (True) қийматни қабул қилади. Акс ҳолда “ёлғон” (False) қийматни қабул қилади.

Оиоусе Pascal тилида шартли ўтиш операторининг икки хил кўриниши мавжуд: тўлиқ ва қисқа.

Тўлиқ кўриниш:

If <шарт> then Begin
<шарт бажарилса бериладиган операторлар>
End
Else
Begin
<шарт бажарилмаганда бериладиган операторлар>
End;

Қисқа кўриниш:

If <шарт> then Begin
< шарт бажарилса бериладиган операторлар >
End;

Бу ерда IF -агар; then -у ҳолда; else -акс ҳолда маъносини билдирувчи хизматчи (калит) сўзлар.

Биринчи кўринишдаги шартли операторда, агар шарт бажарилса биринчи Begin ва end ичидаги операторлар кетма–кет бажарилади, акс ҳолда иккинчи Begin ва end ичидаги операторлар кетма-кет бажарилади.

Иккинчи кўринишдаги шартли оператор қуйидагича ишлайди. Агар берилган шарт бажарилса Begin ва end ичидаги операторлар кетма-кет бажарилади, акс ҳолда улар бажарилмайди.

Агар бажарилувчи операторлар сони битта бўлса Begin ва End сўзларини ёзиш шарт эмас.

Мисоллар:

1) If A>0 Then Begin C:=1; B:=C+1; End

Else Begin C:=0; B:=4; End;

2) If D=A Then D:=A Else A:=D;

Хар бир шартли ўтиш оператори ичида бошқа ички шартли операторлар жойлашиши ҳам мумкин, масалан.

If b1 then a1 else If b2 then a2 Else a3;

Мисоллар.

A:=0.5; B:=-1.7; IF A<B THEN A:=B ELSE B:=A;

Жавоб: 0.5<-1.7 ёлғон бўлганлиги сабабли B:=A оператор бажарилади, ва бунда A=0,5 ва B=0,5 экенлиги келиб чиқади.

A:=0.1; B:=0.1; C:=0.5; D:=0;

IF (A<B) OR (A>C) THEN D:=B+C ELSE

IF B=A THEN BEGIN D:=C; C:=A; END;

Жавоб: (0.1<0.1)ёки(0.1>0.5) бу мантиқий ифода ёлғон бўлганлиги сабабли B=A шарт текширилади. Бу мантиқий ифода чин бўлганлиги сабаб D=0,5 га, C=0,1 қийматларга тенг эканлиги келиб чиқади.

Шартсиз ўтиш оператори

Дастурда шундай ҳолатлар бўладики операторларнинг бажарилиш шартига қараб дастурнинг у ёки бу қисмига тўғридан-тўғри ўтишга тўғри келади. Бундай ҳолатларда шартсиз ўтиш операторидан фойдаланиш мумкин.

Шартсиз ўтиш операторининг кўриниши қуйидагича:

Goto n;

Бу ерда n -белги(метка) бўлиб идентификатор ёки бутун сон бўлиши мумкин. Goto - ўтиш маъносини билдиради.

n- белги дастурнинг бош қисмида Label сўзи ёрдамида эълон қилинган бўлиши шарт. n бошқарилиш узатиладиган жойга «n:» шаклида қўйилади.

Мисол:

.....

```
   Goto L2;
    .....
L2: C:=x*y;
    .....
```

Танлаш операторлари Case

Кўп ҳолларда барор бир параметрнинг қийматига қараб керакли операторларни бажаришга тўғри келади. Бундай ҳолларда танлаш операторини ишлатган қулай. Танлаш оператори кўриниши қуйидагича бўлади:

```
Case s of
  1: A1;
  2: A2;
  . . . . . .
  n: An;
        Else Begin
          <B1,B2,..Bn>
          End;
 End;
```

Бу ерда Case -хизматчи сўз бўлиб танлаш маъносини беради; of -«дан» маъносини беради; s-оператор селектори; 1,2,..n-оператор белгилари; A1,A2,...An ва B1,B2,...Bn-операторлар.

Case оператори тармоқланиш жараёнида берилган бир неча операторидан бирини танлаш йўли билан амалга оширади. Опреаторлар кетма-кетлигини танлаш оператор селекторининг қийматига қараб аниқланади. Оператор селектори ҳақиқий бўлмаган ўзгарувчи ёки ифода бўлиши мумкин. Агар оператор селектори қиймати оператор белгилари ўзгармас қийматига тенг бўлмаса B1,B2,...Bn-операторлари кетма-кет бажарилади.

Шартли ўтиш операторининг қуйидаги кўриниши

```
 If B Then A1 Else A2;
```

танлаш операторининг қуйидаги операторига эквивалентдир.

```
Case B of
        True: A1;
        False: A2;
    End;
```

Мисол

$ax^2+bx+c=0$ квадрат тенгламанинг илдизларини топиш дастури тузилсин.

```
PROGRAM Corni;
 Label 20;
 Var  A, B,C, D, E, F, X, X1, X2, Z:real;
   BEGIN
      Writeln ('a, b, c коэффициентлар қийматини киритинг:');
    Read (A,B,C);
   if A=0 THEN
    BEGIN
      x:=-B/C;  writeln (x);
        goto 20
       end
    else
    BEGIN
     D:=B*B-4.0*A*C;      Z:=2.0*A;     E:=-B/Z;
                    F:=sqrt(ABS(d)/Z;
    End;
       if D>=0 THEN
     BEGIN
          x1:=E+F;   x2:=E-F;   writeln (x1,x2);
     End
     else if D=0 then
        begin X:=E; writeln(x); end else writeln
('ечим йўқ');
   20: end.
```

Такрорланувчи жараёнларни дастурлаш.

Такрорланиш операторлари.

Айрим масалаларда бир ёки бир неча параметрларнинг ўзгаришига қараб маълум ҳисоблашлар бир неча марта такрорланиб бажарилиши мумкин. Масалан, y=ax+b функцияни x нинг бир неча қийматида унинг мос қийматларини ҳисоблаш керак дейлик. Бундай ҳисоблашларни компьютерда дастур тузиб бажариш учун циклик структурали дастурлар тузиш керак бўлади. Бу каби дастурларни шартли ўтиш оператори ёрдамида ҳам тузиш мумкин. Лекин Паскал тилида циклик структурали дастур тузиш учун бир неча махсус операторлар мавжуд.

Object Pascal такрорланувчи жараёнларни ташкил қилишнинг 3 та усулини тавсия этади: олд шартли, кетшартли ва параметрли такрорланиш жараёнлари. Бу турдаги жараёнларни дастурлашда мос равишда While, Repeat ва For операторларидан фойдаланилади.

Параметрли такрорланиш оператори.

For оператори такрорланишлар сони аниқ бўлган циклик жараёнларни ташкил этишда ишлатилади. Унинг умумий кўриниши қуйидагича:

For i:=m1 to m2 Do S;

Бу ерда i-цикл параметри; m1,m2 -i параметрининг бошланғич ва охирги қиймати бўлиб, улар ўзгармас сон ёки ифода бўлиши мумкин; S-цикл танаси бўлиб, бир неча операторлардан ташкил топиши мумкин. Цикл танаси i нинг m1 дан m2 гача бўлган ҳамма қийматлари учун такроран бажарилади.

Агар цикл танаси бир неча операторлардан иборат бўлса улар **Begin** ва **End** ичига олинади.

Мисол. 1,2,...10 сонлар йиғиндисини ҳисоблаш дастурсини тузинг.

```
Program S10;
        Const kn=10;
        Var  i: Integer; S: Real;
        Begin
          S:=0;
          For i:=1 to kn do S:=S+i;
          Write (‘S=’,S);   Readln;
       End.
```

Агар to сўзни **DoWnto** сўзига алмаштирилса цикл параметрининг тескари тартибда ўзгаради, яъни -1 қадам камайиб боради. У ҳолда цикл кўриниши қуйидагича бўлади.

For i:=m1 DoWnto m2 Do S;

Мисол. 10 дан 1 гача сонларни экранга чиқариш дастурини тузинг.

```
Program SP;
        Var i: Integer;
        Begin
         For i:=10 DoWnto 1 do Write (i); Readln;
        End.
```

Олдшартли такрорланиш оператори.

While цикл оператори такрорланишлар сони олдиндан аниқ бўлмаган ҳолларда такрорланишни бирор бир шарт асосида бажаради. Берилган шарт олдин текширилади ва кейин шартнинг бажарилишига қараб керакли операторлар кетма-кетлиги бажарилади. Бу операторнинг умумий кўриниши қуйидагича:

While B Do S;

Бу ерда B -мантиқий ифода; S -цикл танаси бўлиб, бир ёки бир неча операторлар кетма-кетлигидан иборат бўлиши мумкин. Мантиқий ифода ‘True’ ёки ‘False’ қиймат қабул қилади.

Агар мантиқий ифода ‘True’ қиймат қабул қилса S операторлари бажарилади, акс ҳолда бажарилмайди, яъни цикл ишлашдан тўхтайди.

Мисол. 1,2,...,10 сонлар йиғиндисини ҳисоблаш дастурини тузинг.

```
Program S10;
        Const   kn=10;
        Var  i: Integer; S: Real;
        Begin
          S:=0;  i:=0;
          While i<=kn do Begin i:=i+1; S:=S+i; End;
          Write (‘S=’,S);
           Readln;
      End.
```

Кетшартли такрорланиш оператори.

Repeat цикл оператори ҳам такрорланишлар сони олдиндан аниқ бўлмаган ҳолларда такрорланишни бирор бир шарт асосида бажаради. Бу операторнинг умумий кўриниши қуйидагича:

Repeat
S
Until B

Бу ерда B -мантиқий ифода, ‘True’ ёки ‘False’ қиймат қабул қилади; S -цикл танаси бўлиб, бир ёки бир

неча операторлар кетма-кетлигидан иборат бўлиши мумкин. Олдин цикл танасидаги операторлар кетма-кетлиги бажарилади. Берилган шарт кейин текширилади. Агар берилган шарт рост (True) бўлса, бошқарув циклдан кейинги операторни бажаришга ўтади, акс ҳолда цикл такрорланади. Агар мантиқий ифода ‘False’ қиймат қабул қилса циклда такрорланиш давом этади, акс ҳолда тўхтайди.

Мисол. 1,2,...,10 сонлар йиғиндисини ҳисоблаш дастурини тузинг.

```
Program S10;
        Const  kn=10;
        Var  i: Integer; S: Real;
        Begin
          S:=0;  i:=0;
          Repeat
                  i:=i+1; S:=S+i;;
          Until I>kn;
           Write (‘S=’,S);
           Readln;
      End.
```

Одатда WHILE оператори REPEAT операторига нисбатан кўп ишлатилади. Бунга сабаб кўпчилик масалаларда цикл тугалланиш шарти цикл бошланмасдан текшириш мақсадга мувофиқдар. Зарур бўлса циклни умуман бажармасдан ўтиш мумкин.

Кўпчилик масалаларни ечишда тузилган дастурда ичма-ич жойлашган цикллар ташкил этишга тўғри келади. Бундай цикларга мураккаб(каррали) цикллар дейилади. Мураккаб цикллар ташкил этилганда қуйидаги талаблар бажарилиши зарур.

-ички цикл ташқи цикл ичида тўлиқ ётиши керак;

-цикллар бир-бири билан кесишмаслиги керак;

-цикл ичига ташқаридан тўғридан-тўғри кириш мумкин эмас;

-цикл параметрлари бошқа-бошқа идентификаторлар билан белгила-ниши керак;

Мисол. $S = \sum_{i=1}^{10} \prod_{j=1}^{5} \frac{i+j}{\sqrt{i \cdot j}}$ ифодани ҳисоблаш дастурсини тузинг.

Бу формулада агар йиғиндини очсак у қуйидаги кўринишга келади.

$$S = \sum_{i=1}^{10} \prod_{j=1}^{5} \frac{i+j}{\sqrt{i \cdot j}} = \prod_{j=1}^{5} \frac{1+j}{\sqrt{1 \cdot j}} + \prod_{j=1}^{5} \frac{2i+j}{\sqrt{2 \cdot j}} + ... + \prod_{j=1}^{5} \frac{10+j}{\sqrt{10 \cdot j}}$$

```
Program SP;
      Var
        i,j: Integer; S: Real;
      Begin
       S:=0;
       For i:=1 to 10 do
        Begin
            P:=1;
            For j:=1 to 5 do P:=P*(i+j)/Sqrt(i*j);
          S:=S+P;
      End;  Write ('S=',S);
      End.
```

Мунтазам тоифадаги маълумотлар билан ишлаш.

Массивлар

Кўп ҳолларда жадвал ёки матрицалар кўринишидаги маълумотлар билан иш юритиш керак бўлади. Жадвалда маълумотлар жуда кўп бўлгани сабаб, уларнинг ҳар бир ячейкасидаги сонни мос равишда битта ўзгарувчига қиймат қилиб берилса улар устида иш бажариш анча ноқулайликларга олиб келади. Шу сабаб дастурлашда бундай муаммолар массивларни ишлатиш ёрдамида ҳал қилинади.

Массив - бу бир ном билан белгиланган бир тоифали қийматлар гуруҳи ёки жадвалдир. Массивнинг ҳар бир элементи массив номидан сўнг ўрта қавс ичига олинган рақам ва арифметик ифода ёзиш билан белгиланади, Қавс ичидаги рақам массив индексини белгилайди. Векторни бир ўлчовли массив, матрицани икки ўлчовли массив деб қараш мумкин.

Бир ўлчовли массивда унинг ҳар бир элементи ўзининг жойлашган ўрин номери билан аниқланади ва номери қавс ичида индекс билан ёзилади, масалан C[2]. Икки ўлчовли массив элементи ўзи жойлашган сатр ва устун номерлари ёрдамида аниқланади. Шу сабаб икки ўлчамли массив элементи иккита индекс орқали ёзилади. Масалан: A[i,j] бу ерда i-сатр номери j-устун номерини билдиради.

Массивни эълон қилиш дастурнинг бош қисмида берилиб, унинг ёзилиши умумий ҳолда қуйидагича бўлади:

<Массив номи>:Array[ўлчам, индекс тофиаси] of <элемент тоифаси>;

Масалан:

```
A,B:Array[1..100] of real;
C,A1,D:Array[1..10,1,,15] of real;
```

Бу ерда А ва В массивлари 100тадан элементга эга. C,A1,D1 массивлари эса 10x15=150 тадан элементга эга.

Массивларни эълон қилишдан мақсад массив элементлари учун компьютер хотирасидан жой ажратишдир.

Массив элементлари қийматларини киритиш учун цикл операторларидан фойдаланилади.

Мисол: For i:=1 to 10 do Read(A[i]);

Бу мисолда А массивнинг 10 та элементи қийматини экрандан кетма-кет киритиш керак бўлади. Худди шундай массив қийматларини экранга чиқариш ҳам мумкин.

Мисол: For i:=1 to 10 do Write(A[i]);

Дастурда массив элементларини ишлатганда уларнинг индекси эълон қилинган чегарадан чиқиб кетмаслиги керак.

Массивлар устида одатда, массив элементларини тартиблаш, уларнинг йиғиндиси, максимал ва минимал элементини аниқлаш каби амаллар бажарилади.

Массив элементларини тартиблаш усуллари.

Массивни тартиблаштиришнинг бир неча усуллари (алгоритмлари) мавжуд. Улардан қуйидаги усулларни қараб чиқамиз:

-танлаш усули;

-алмаштириш усули.

Танлаш усули ёрдамида массивни ўсиш бўйича тартиблаштириш алгоритми қуйидагича:

1.Массивнинг биринчи элементидан бошлаб қараб чиқилиб энг кичик элемент топилади.

2.Биринчи элемент билан энг кичик элемент жойлари алмаштирилади.

3.Иккинчи элементидан бошлаб қараб чиқилиб энг кичик элемент топилади.

4.Иккинчи элемент билан энг кичик элемент жойлари алмаштирилади.

5.Бу процесс битта охирги элементгача такрорланади.

Бу алгоритм дастурси қуйидагича бўлади:

```
Program Sort;
  Const  Size=5;
  Var i,j,min,k,buf: Integer;
   a: Array[1..Size] of Integer;
  Begin
       Writeln (‘Массивни тартиблаштириш’);
       Write (Size:3,’та массив элементини киритинг’);
       For k:=1 to Size Do Read(a[k]);
     Writeln (‘Тартиблаштириш’);
```

```
      For i:=1 to Size-1 Do
              Begin
                { кичик элементни топиш }
                min:=i;
                For j:=i+1 to Size Do
                 Begin
                     If a[j]<a[min] then min:=j;
                     buf:=a[i];      a[i]:=a[min];
a[min]:=buf;
                     For k:=1 to Size Do  Write (a[k],' ');
                     Writeln;
                 End;
              End;
         Writeln('Массив тартиблаштирилди.');
    End.
```

Дастур натижаси:

```
Массивни тартиблаштириш
 5 та массив элементини киритинг
 12 -3 56 47 10
Тартиблатириш
 -3 12 56 47 10
 -3 10 56 47 12
 -3 10 12 47 56
 -3 10 12 47 56
Массив тартиблаштирилди.
```

Алмаштириш усули ёрдамида массив элементларини ўсиб боришда тартиблаштириш алгоритми қуйидагича:

1.Массивнинг биринчи элементидан бошлаб кетма-кет ҳамма қўшни элементлар бир-бири билан солиштирилиб, агар биринчиси иккинчисидан кичик бўлса улар жойи алмаштирилиб борилади.

2.Бу процесс давомида кичик қийматли элементлар массив бошига катта элементлар эса охирига силжитилиб борилади. Шу сабаб бу усул «пузирка» усули ҳам дейилади.

3.Бу процесс массив элементлар сонидан битта кам марта такрорланади.

Масалан:

3 2 4 5 1 бунда 3 билан 2 ва 5 билан 1 алмаштирилади.

2 3 4 1 5 бунда 4 билан 1 алмаштирилади.

2 3 1 4 5 бунда 3 билан 1 алмаштирилади.

2 1 3 4 5 бунда 2 билан 1 алмаштирилади.

1 2 3 4 5

Бу алгоритм дастури қуйидагича бўлади:

```
Program Sort;
      Const  Size=5;
   Var i,j,min,k,buf: Integer;
    a: Array[1..Size] of Integer;
   Begin
      Writeln (‘Массивни пузирек(купикча) усулида тартиблаштириш’);
      Write (Size:3,’та массив элементини киритинг’);
      For k:=1 to Size Do Read(a[k]);
     Writeln (‘Тартиблатириш’);
     For i:=1 to Size-1 Do
              Begin
                 For k:=1 to Size-1 Do
                  Begin
                       If a[k]>a[k+1] then
                         Begin
                              buf:=a[k];     a[k]:=a[k+1];
a[k+1]:=buf;
                  End;
                  End;
```

```
            For k:=1 to Size Do  Write (a[k],' ');
                    Writeln;
          End;
        Writeln('Массив тартиблаштирилди.');
    End.
```

Дастур натижаси:

Массивни пузирек усулида тартиблаштириш
5 та массив элементини киритинг
3 2 4 1 5
Тартиблаштириш
2 3 4 1 5
2 3 1 4 5
2 1 3 4 5
1 2 3 4 5
Массив тартиблаштирилди.

Массивда энг кичик ёки энг катта элементни излаш алгоритми маълумки биринчи элемент энг кичик (катта) деб олиниб кейин бошка элементлар билан кетма-кет солиштирилиб чиқилади. Солиштирилиш охирги элементгача бажарилади.

Қуйида бу алгоритм дастурси келтирилган:

```
Program MinMax;
  Var i,min: Integer;
      a: Array[1..10] of Integer;
   Begin
        Writeln ('Массивдан энг кичик элементни излаш');
        Write (' 10-та массив элементини киритинг');
        For i:=1 to 10 Do Read(a[i]);
          min:=1;
        For i:=2 to 10 Do
            If a[i]<a[min] Then min:=i;
```

Writeln(‘Изланаётган энг кичик элемент:’,a[min]);

Writeln(‘Элемент номери’,min);

End.

Динамик массив

Динамик массив таърифланганда унинг узунлигини олдиндан эълон қилиш шарт эмас.

Массив узунлигини ўрнатиш учун SetLength функциясидан фойдаланиш мумкин. Унинг икки параметри мавжуд:

1. Динамик массив тоифасидаги ўзгарувчи.
2. Массив узунлиги.

High(r) функцияси массив элементлари сонини аниқлайди.

Мисол:

```
r:array of integer;
i:Integer;
begin
SetLength(r,10);
for i:=0 to High(r)-1 do
begin
r[i]:=i*i;
writeln (IntToStr(i)+' квадрати ='+IntToStr(r[i]));
end;
```

IntToStr функцияси сонни сатрга айлантиради.

Delphi тилининг процедура ва функциялари.

Қисм дастурлар

Дастурлаш жараёнида шундай ҳолатлар бўладики, бир хил операторлар кетма-кетлигини дастурнинг бир неча жойларида такроран ёзишга тўғри келади. Бундай такрорланишни йўқотиш мақсадида дастурлашнинг

кўпгина тилларида қисм дастур тушунчаси киритилган. Такрорланадиган операторлар кетма-кетлигини мустақил дастур бўлаги -қисм дастур кўринишида бир маротаба ёзилади ва бу дастур бўлаги керак бўлган жойларда эса, унга мурожаат қилинади халос. Паскаль тилида қисм дастур - процедура ёки функция кўринишида берилади

Айрим масалаларни ечишда маълум параметрларнинг ҳар хил қийматларида бир хил ҳисоблашларни бажаришга тўғри келади. Бундай ҳолларда дастур ҳажмини кичирайтириш мақсадида процедура ёки функциялар ташкил қилиш зарур. Процедура ёки функцияга мурожат қилиш дастурда унинг номини кўрсатиш орқали амалга оширилади. Керакли параметрлар шу номдан кейин берилади. Процедура ёки функциялар ташкил қилинганда улар дастурнинг бош қисмида берилади. Уларга мурожаат қилиш эса дастурнинг асосий қисмининг керакли жойида берилади. Асосий дастур билан процедура орасида ўзгарувчилар қиймат алмашуви формал ва фактик параметрлар ёрдамида амалга оширилади. Процедура ёки функцияга мурожаат қилинганда бошқарилиш қаердан узатилса яна шу жойга қайтиб келади. Процедура ичида яна бир неча процедура ёки функция ишлатилиши мумкин. Дастурда эълон қилинган ўзгарувчилар, шу дастурдаги процедура ва функцияларга нисбатан глобал дейилади. Процедура ва функцилар ичида эълон қилинган ўзгарувчилар локал дейилади. Уларнинг таъсир доираси шу процедура ва функцияларнинг ичида бўлади халос.

Процедураларни эълон қилиш дастурнинг бош қисмида келтирилади ва у қуйидагича бошланади.

Procedure <проц.номи> (<формал параметрлар>);

М: **Procedure AB (x,y);**

Формал параметрларни шу процедура бош қисмида ёки сарлавҳада эълон қилиш мумкин.

М. **Procedure AB (x,y: Real);**

Ҳар қандай процедурани кичик бир дастур деб қараш мумкин. Процедура ҳам дастурга ўхшаб бош ва асосий қисмлардан тошкил топади. Бош қисмда процедура номи ва унинг параметрлари эълон қилинади. Асосий қисм операторлар кетма-кетлигидан ташкил топган бўлиб, улар Begin - End ичига олинади. Процедура номи фойдаланувчи тамонидан берилади.

Мисол.

```
Procedure Dr(Var x,h1,h2,z1,z2 : Real);
    Var h,z: Real;
    Begin
        h:=h1/z1+h2/z2;
        z:=z1/z2;
        x:=(h+z)/2;
    End;
```

Бу процедурада h1,z1,h2,z2 параметрлар қиймати процедурага мурожат қилинганда аниқланган бўлиши керак. Натижани эса x- параметр узатади. h ва z ўзгарувчилар ички ўзгарувчилардир. Бу процедурага дастурдан қуйидагича мурожаат қилинади.

```
Dr(x,h1,h2,z1,z2);
```

Процедурага мурожаат қилинганда мос параметрлар қиймати бир бирига узатилади. Бериладиган формал ва фактик параметрлар сони тенг ва улар турлари бир хил бўлиши шарт. Лекин параметрлар номлари ҳар хил бўлиши мумкин.

Функциялардан фойдаланиш ва уларни ташкил қилиш худди процедура каби бўлиб, уларни эълон қилиш дастурнинг бош қисмида келтирилади ва у қуйидагича бошланади:

Function <ф-я номи>(<формал параметрлар>):<ф-я тури>;

М. Function Min (x,y:Real): Real;

Функция номи фойдаланувчи тамонидан берилади. Функцияга мурожаат қилиш унинг номи орқали берилади.

Функция ҳам процедурага ўхшаб бош ва асосий қисмлардан тошкил топади. Функциянинг процедурадан фарқи, унга мурожат қилинганда натижа фақат битта бўлиб, у шу функция номига узатилади.

Мисол 1. Јуйидаги ҳисоблашни функцияни ишлатган ҳолда дастурсини тузинг.

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

```
Program Kol;
 Var ncm,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
        Read(n,m); l:=n-m;
        ncm:=Fact(n)/Fact(m)/Fact(l);
        Write('ncm=',ncm);
  End.
```

Мисол 2. Қуйидаги ҳисоблашни процедурани ишлатган ҳолда дастурини тузинг.

$$z = \frac{tha - th^2(a-b)}{\sqrt{th(a^2-b^2)}}, \qquad thx = \frac{e^{2x}-1}{e^{2x}+1}.$$

```
Program Fun1;
 Var a,b,z,c,d,t1,t2,t3: Real;
  Procedure Th(Var x,r: Real);
   Var c: Real;
```

```
  Begin
        c:=exp(2.0*x);
        r:=(c-1)/(c+1);
  End;
Begin
      Read(a,b);   th(a,t1);
      c:=a-b;      th(c,t2);
      d:=Sqr(a)-Sqr(b);   th(d,t3);
      z:=(t1+Sqr(t2))/Sqr(t3);
      Write('z=',z:10:3);
End.
```

Олдиндан эълон қилиш

Бизга икки процедура A ва B берилган бўлиб A процедура B процедурага мурожаат қилсин ва A таърифи B таърифидан олдин келсин. Масалан:

```
Procedure A (i : integer);
begin
B (i);
Writeln(i);
end;
Procedure B (var j : integer) ;
begin
j:=j*2;
end;
```

Бундай таъриф хатоликка олиб келади. Чунки A процедура хали таърифланмаган процедурага мурожаат қилмокда. Бу ҳолда B процедурани қуйидагича олдиндан эълон килиш лозим:

```
Procedure B (var j : integer); Forward;
Procedure A (i : integer);
begin
B (i);
Writeln(i);
end;
```

```
Procedure B (var j : integer) ;
begin
j:=j*2;
end;
```

Олдиндан эълон қилишда процедура танаси стандарт директива Forward билан алмаштирилади.

Белги ва қаторлар билан ишлаш махсус функциялари

Паскаль тилида бир қанча махсус процедура ва функциялар мавжуд бўлиб, улар қуйидаги гуруҳларга бўлинади:

-қаторни қайта ишлаш;

-файллар билан ишлаш;

-динамик ўзгарувчилар учун хотирани бошқариш;

-арифметик функциялар;

-экран билан ишлаш.

Уларнинг айримларини кўриб чиқамиз:

Halt- дастурни бажаришдан тўхтатиш;

Odd(i)- I-тоқ бўлса “True” акс ҳолда “False” қиймат олади;

Edit- бажарилаётган блокдан чиқиш;

Random- 0 дан 1 гача бўлган сонни тасоддифан олиш;

Int(x)- соннинг бутун қисмини олиш;

Frac(x)- соннинг каср қисмини олиш;

Round(x)- берилган сонни яхлитлаб бутун олиш;

GotoXY(x,y)- курсорни кўрсатилган жойга қўйиш;

ClrScr- экранни тозалаб, курсорни экран бошига қўйиш;

Trunc- аргументнинг бутун қисми:

Str(I;Var S:String)- рақамни символга ўтказиш (I-ифода ёки ўзгарувчи);

Val(S:String; Var P;ko:Integer)- символни рақамга ўтказиш (P-ўзгарувчи);

Length (S:String)- қатор узунлигини аниқлаш.

Pos(st1,st) - қатордаги қатор қисми ҳолатини аниқлаш.

Мисол: st:=’Тошкент’; st1:=’кент’; p:=pos(st1,st);

Жавоб: p=4. Агар изланаётган қатор қисм қаторда йўқ бўлса қиймат нулга тенг бўлади.

Copy(st,m,n) - қатордан фрагмент кесиб олади.

Мисол: st:=’Тошкент’; p:=Copy(st,4,4);

Жавоб: p=’кент’.

Delete(st,m,n) - қатордан фрагмент кесиб олиб ташлайди.

Мисол: st:=’Тошкент’; p:=Delete(st,4,4);

Жавоб: p=’Тош’.

Махлумотларнинг аралаш тоифаси.

Езув.

Дастурлаш амалиётида стандарт маълумотлардан ташкил топган мураккаб маълумотлар билан ишлашга тўғри келади. Мисол учун талаба тўғрисидаги маълумот исми шарифи, туғилган йили, адреси, курси, гуруҳи ва ҳоказолардан иборат бўлиши мумкин. Бундай маълумотларни таърифлаш учун Delphi да маълумотларнинг аралаш тоифаси - ёзув (record) лардан фойдаланилади.

Ёзув бу - алоҳида номланган ҳар хил турли компоненталардан иборат мураккаб тоифадир.

Ҳар қандай тоифа каби, "ёзув" type бўлимида таърифланиши лозим. Бу таъриф умумий кўриниши:

<ёзув тоифаси номи> = record
1_ Майдон: 1_тоифа;
2_ Майдон: 2_тоифа;
. . . . . .;
K_ Майдон: K_тоифа
end;

Var

<ёзув номи > : <ёзув тоифаси ном> **;**

Таърифларга мисоллар:

type
TPerson = record
f_name: string[20];
l_name: string[20];
day: integer;
month: integer;
year: integer;
address: string[50]; end;

TDate = record
day: integer; month: integer; year: integer;
end;

Ёзув туридаги ўзгарувчини қуйидагича таърифлаш мумкин:
var
student : TPerson; birthday : TDate;

Ёзув элементига (майдонига) мурожаат қилиш учун ёзув номи ва нуқтадан сўнг майдон номини кўрсатиш керак. Масалан:

Writeln('Имя: ', student.f_name + #13 + 'Адрес: ', student.address);

Инструкция экранга student ўзгарувчи-ёзувнинг f_name (ном) ва address (адрес) майдонларини чиқаради.

Баъзида ўзгарувчи-ёзув тоифаси ўзгарувчилар эълон қилиш бўлимида эълон қилинади. Бу ҳолда ёзув тоифаси ўзгарувчи номидан сўнг кўрсатилади. Мисол учун student ёзуви var бўлимида қуйидагича таърифланиши мумкин:

```
Var
student: record
f_name:string[20];
l_name:string[20];
day:integer;
month:integer;
year:integer;
address:string[50];
end;
```

With инструкцияси

With инструкцияси дастурда майдонлар номларини ўзгарувчи – ёзув номини кўрсатмасдан ишлатишга имкон беради. Умумий ҳолда with инструкцияси қуйидаги кўринишга эга:

with <ёзув номи> do

begin
(дастур инструкцияси} end;

Мисол учун дастурда қуйидаги ёзув таърифланган бўлсин

```
student: record
f_name: string[30];
l_name: string[20];
address: string[50];
end;
```

ва студентлар тўғрисидаги маълумотлар E1, E2 ва E3 ўзгарувчиларда жойлашган бўлсин. У ҳолда

```
student.f_name := El;
student.l_name := E2;
student.address := E3;
```

инструкциялар ўрнига қуйидаги инструкцияни ёзиш мумкин:

```
with student do begin

f_name := E1; l_name := E2; address := E3;
end;
```

Маълумотларнинг файлли тоифаси.

Файллар билан ишлаш функция ва процедуралари

Файл бу хотиранинг номланган сохаси. Хар бир файл ўз номига эга. Одатда файлда бир тоифали маълумотлар сақланади. Яратилаётган файлнинг узунлиги олдиндан эълон қилинмайди.

Object Pascalда файлларнинг учта тури белгиланган:

1. Тоифалашган файллар.

2. Матнли файллар.
3. Тоифалашмаган файллар.

Киритиладиган ва чиқариладиган маълумотлар сони кўп миқдорда бўлса уларни файлда сақлаш дастурда қулайлик туғдиради. Бу маълумотлар оддий матн (текст) файлларида сақланади. Файл ўзгарувчиси дастур бош қисмида эълон қилинади, яъни

```
Type f=text;
Var
    fx:f;
```

бу ерда f -файл типи, оддий матн файлни билдиради;

fx-файл ўзгарувчиси.

Керакли файлдан маълумотларни ўқишга тайёрлаш учун Assign ва Reset функциялари ишлатилади.

Assign-файл ўзгарувчиси билан ососий файл орасида алоқа ўрнатади. Assign (fx,'c:\a\f1.txt');

Reset-файлни топиб уни ишга тайёрлайди. **Reset (fx);**

Бу ерда fx- файл ўзгарувчиси;

'c:\a\f1.txt'-c: дискнинг А каталогидаги fx.txt файлдан ўқишни билдиради.

Файлдаги маълумотларни ўқиш учун Read функцияси ишлатилади.

Read (<файл ўзгарувчиси>, <ўзгарувчилар,массивлар>);

Мисол. Read (fx, x,y,z,A[i],B[I]);

Файлдан ўзгарувчилар ёки массивлар қийматларини ўқиб бўлгандан кейин файл ёпилади. Файлни ёпиш қуйидаги функция ёрдамида бажарилади. Close (файл ўзгарувчиси);

Мисол. Close (fx);

Мисол 1. C: дискдаги AA каталогдаги AB файлдаги маълумотларни ўқиб A ва B массивларга жойлаштиринг. Файл маълумотлари структураси қуйидагича.

15.2
20.5
20.1 2
5
.
5
20.2
5
0
.
2
26.8 5
2
.
4
. . .
. . .
.

яъни файл структураси икки устундан иборат маълумотлар тўпламидан иборат.

```
Program FAB;
  Type
       f=text;
  Var
       A,B: Array[1..100] of Real;   m: Integer;
       fxz: f;
  Begin
       Assign(fxz,'c:\AA\AB.txt');   Reset(fxz);
       m:=0;
       While not eof(fxz) do
         Begin    m:=m+1;    Readln(fxz,A[m],B[m]);
End;
```

Close (fxz);

End.

Массив қийматларини бирон матн файлига ёзиб қўйиш учун Assign, Rewrite, Append, Write ва Close функциялари ишлатилади.

Assign- файл ўзгарувчиси билан асосий файл ўртасида аълоқа ўрнатади ва у қуйидагича ёзилади.

Assign(файл ўзгарувчиси, дискдаги файл жойи ва номи);

Append- Файлдан ёзиш учун жой тайёрлайли.

Append(файл ўзгарувчиси);

Write- ўзгарувчи қийматини файл ўзгарувчисига узатади ва файлга ёзади. Write(файл ўзгарувчиси, ўзгарувчилар);

Close- очилган файлни ёпади. Close (файл ўзгарувчиси);

Мисол 2. Юқоридаги C: дискдаги AA каталогдаги AB файлдаги маълумотларни A ва B массивларга ўқиб шу массив мос элементларини қўшиб C массивни ташкил қилинг ва A,B,C массивларини ABC номли файлга жойлаштиринг.

Файл маълумотлари структураси қуйилагича бўлсин.

i	A	B	C
1	15.2	20.5	45.7
2	20.1	25.5	45.6
. . . . .			

```
Program FABC;
  Type
      f=text;
  Var
      A,B,C: Array[1..100] of Real;  i,m: Integer;
```

```
        fax: f;
    Begin
        Assign(fax,'c:\AA\AB.txt');   Reset(fax);
        m:=0;
        While not eof(fax) do
          Begin        m:=m+1;
  Readln(fax,A[m],B[m]);  End;
          Close (fax);
        Assign(fax,'c:\AA\ABC.txt');   Rewrite(fax);
        Append(fax);
        For i:=to m do
         Begin                Write(fax,i,A[i],b[i],c[i]);
Writeln(fax);  End;
        Close(fax);
  End.
```

Модуллар

Object Pascal системасида жуда кўп махсус тайёр процедура ва функциялар мавжудки улар ҳар қайси ўз вазифасига эга бўлиб унга библиотека модуллари дейилади. Ҳар бир библиотека функция ва процедуралардан ташкил топган бўлиб маълум бир турдаги масалани ечишга мулжалланган.

Модул деб процедура ва функциянинг алоҳида компиляция қилиниб махсус .tpu кенгайтмали файл шаклидаги ифодаланган дастурсига айтилади. Модулдан ихтиёрий дастур ичида фойдаланиш мумкин. Модулдан фойдаланиш яъни уни активлаштириш учун дастурнинг бош қисмида қуйидагиларни келтириш зарур.

Uses <Модул1 номи, модул2 номи, . . . ,модулn номи>;

Мисол:

```
 Program SS;
        Uses Crt,Graph;
```

Турбо-Паскаль системасида ҳар бир фойдаланувчи ўз модулини яратиши учун яратиладиган модул структурасини қуйидагича ташкил килиш зарур.

Uses <модул номи>;

Interface

. . . . .

{Интерфейс қисм- очиқ (ёзувлар) қисми}

. . . . .

Implementation

. . . . .

{Ёпиқ (ёзувлар) қисми}

. . . . .

Begin

. . . . .

{Модулнинг асосий қисми}

. . . . .

End.

Бу ерда

Unit - модулнинг сарлавҳаси;

Interface - модулнинг интерфейси, яъни дастур ва бошқа модуллар учун очиқ (кўринарли) қисмининг бошланишини билдиради. Бу қисмда ўзгармаслар, катталиклар типлари, процедура ва функциялар аникланиб курсатилган бўлади, лекин уларнинг бутун кўриниши кейинги ёпиқ қисмда берилади.

Implementation - модулнинг дастур ва модуллар учун ёпиқ, яъни кўринмайдиган қисмининг бошлани-шини билдиради. Бу ерда интерфейс қисмда аниқланган процедура ва функциялар яна бир марта кўрсатилиши шарт (уларнинг сарлавҳалари бир хил бўлиши керак).

Инициализация қисми Begin ёзувидан кейин бошланади, агар бу қисм мавжуд бўлмаса Begin ҳам

бўлмайди. Бу қисмда бошқарувни асосий програмага ўткизишгача қадар бажарилиши керак булган операторлар руйхати жойлашади.

Мисол тариқасида икки соннинг энг катта ва энг кичигини топиш модулини яратиш дастурсини қараймиз. Қуйидаги дастур Min(x,y) ва Max(x,y) функцяларини ўз ичига олган.

```
Unit Study;
  Interface {Интерфейс кисм}
            Function Min(x,y:Integer):Integer;
            Function Max(x,y:Integer):Integer;
Implementation {Ёпик кисм}
  Function Min(x,y:Integer):Integer;
Begin
          If x<=y Then Min:=x Else Min:=y;
 End;
            Function Max(x,y:Integer):Integer;
Begin
          If x>=y Then Max:=x Else Max:=y;
 End;
{Инициализация қисми мавжуд эмас}
End.
```

Бу модул компиляция қилиниб Stadu.tpu файл номга эга бўлиши керак. Ундан дастурда фойдаланиш учун дастур бош кисмида Uses Study қаторини ёзиш керак бўлади.

Турбо Паскаль системасида қуйидаги библиотека модуллари мажуд:

System - стандарт процедура ва функцияларни ўз ичига олган бўлиб, бу модул автоматик равишда активлаштирилган бўлади.

Dos - Ms Dos операцион система имкониятларидан фойдаланувчи процедура ва функцияларни ўз ичига олган.

Crt - монитор экрани ва клавиатура билан ишлаш имкониятини яратувчи процедуралар тўпламини ўз ичига олган.

Graph - ҳар хил монитор видеоадаптерларини қуллаган ҳолда компьютер график имкониятларидан фойдаланувчи кўплаб процедуралар тупламини ўз ичига олади.

Printer - принтер билан ишловчи кичик модул.

Динамик боғланувчи библиотекалар (DLL)

Таърифи

Динамик богланувчи библиотекалар дастурда бошка тилларда яратилган процедура ва функциялардан фойдаланишга имкон беради. Динамик библиотекалар билан оддий модуллар орасида жуда кўп ўхшашликлар мавжуд ,лекин икки жихатдан фарқ қилади.

Биринчидан динамик библиотекада эълон қилинган ўзгарувчилар ва константалардан асосий дастурда фойдаланиб бўлмайди.

Иккинчидан модуллар статик усулда, яъни компиляциянинг компоновка босқичида боғланади. Динамик библиотекалар динамик яъни дастур бажарилиш жараёнида боғланади. Агарда икки дастур оддий модулга мурожаат қилса шу модул ишлатилаётган қисмининг икки нусхаси хотирада яратилади. Динамик библиотеканинг икки дастур мурожаат қилаетган қисми фақат бир нусхада яратилади.

Динамик библиотеканинг ўзгариши дастурни қайтадан компиляция килишга олиб келмайди.

Яратилиши

DLL яратиш учун махсус Library сузи ишлатилади.

DLL эълонлар булими Exports сузидан бошланиб, экспорт килинаетган подпрограммалар рўйхатини ўз ичига олади:

Library MyLibrary;

Function MyFunc (...):...;

begin
end;
Procedure MyProc;
begin
end;
Exports
MyFunc, MyProc;
begin
end.

Подпрограмма номидан ташқари DLL га унинг тартиб номери жойлаштирилади: биринчи подпрограмма номери 0, кейингиси - 1 ва хоказо. Дастурчи бу индексацияни ўзгартириши ва 0 дан 32767 гача номер қўйиши мумкин:

Expots
    MyFunc index 1, MyProc index 2;

Дастурчи экспорт қилинаётган подпрограмма учун ташқи ном бериши мумкин:

Exports
    MyFunc index I name 'NEWFUNC';

Чақираётган дастур экспорт қилинаётган подпрограмма ташқи номи ёки индекси бўйича чақириши мумкин.

Мисол

Мисол тариқасида cmplx модулини курамиз.

Library Cmplx;
uses
    SysUtils, Classes;
*{$R *.RES}*
type
TComplex = **record** Re, Im: Real;
end;
function AddC(x, y: TComplex): TComplex; **stdcall;**
begin
Result.Im := x.Im + y.Im;
Result.Re := x.Re + y.Re **end;**
function SubC(x, y: TComplex): TComplex;

```
stdcall;
begin
Result.Im := x.Im - y.Im;
Result.Re := x.Re - y.Re
end;
function MulC(x, y: TComplex): TComplex;
stdcall;
begin
        Result.Re := x.Re * y.Re + x.Im * y.Im;
Result.Im := x.Re * y.Im - x.Im * y.Re
end;
function DivC(x, y: TComplex): TComplex;
stdcall;
var
        z: Real;
begin
        z := sqr(y.Re) + sqr(y.Im);
        try
        Result.Re := (x.Re * y.Re + x.Im * y.Im)/z;
        Result.Im := (x.Re * y.Im - x.Im * y.Re)/z
        except
        Result.Re := le+309;
        Result.Im := le+309
        end
        end;
Exports
AddC index 1 name 'ADDC' resident,
SubC index 2,
MulC index 3,
        DivC index 4;
begin
end.
```

Бу ерда stdcall бу DLL га Delphi дан бошқа тиллардан мурожаат қилишга имкон беради. Агар библиотекага факат

Delphi тилидаги дастурдан мурожаат қилинса бу сўзнинг кераги йўқ.

Подпрограммаларга мурожаат қилиш учун уларни куйидагича эълон қилиш лозим:

Procedure MyProc; **External** 'MyDLL';

Агар индекс бўйича чақириш лозим булса:

Procedure MyProc; **External** 'MyDLL' index 2;

Бундан ташқари дастурчи ташқи номни ўзгартириши хам мумкин:

Procedure MyProc; **External** 'MyDLL' Name 'ExtName';

Фойдаланиш

Статик юклаш

Дастурда Cmpix библиотекаси қуйидагича эълон қилиниши лозим.

```
type
TComplex = record Re, Im: Real;
        end;
function ADDC(x, y: TComplex): TComplex; stdcall; External 'Cmplx' ;
function SubC(x, y: TComplex): TComplex; stdcall; External 'Cmplx' ;
function MulC(x, y: TComplex): TComplex; stdcall; External 'Cmplx' ;
function DivC(x, y: TComplex): TComplex; stdcall; External 'Cmplx';
```

Интерфейсли модуль

DLL-подпрограммаларни чақиришда ёзув каби мураккаб турдаги маълумотларни узатишга тўғри келиши мумкин. Агар бирор DLL га куп мурожаат қилинса мураккаб тур эълон килинган интерфейсли модулдан фойдаланиш қулайдир. Масалан:

```
Unit Complx;
Interface
type
        TComplex = record Re, Im: Real;
```

```
end;
function AddC(x, y: TComplex): TComplex; stdcall;
External 'Cmplx' index 1;
function SubC(x, y: TComplex): TComplex; stdcall;
External 'Cmplx' index 2;
function MulC(x, y: TComplex): TComplex; stdcall;
External 'Cmplx' index 3;
function DivC(x, y: TComplex): TComplex; stdcall;
External 'Cmplx' index 4;
Implementation end.
```

Бошқа тилда ёзилган DLL процедураларига мурожаат қилинганда уларнинг ташқи номи Delphi қоидасига тўғри келмаслиги мумкин. Масалан C++ тили идентификаторларда “@” символидан фойдаланишга рухсат беради. Бу холда Delphi қоидасига мос ном бериб name сузидан сунг асл номини кўрсатиш керак. Масалан:

```
function MyFunction: WordBool; stdcall;
external 'MyDLL' name '_MyFunction@12'
```

DELPHI ВИЗУАЛ ДАСТУРЛАШ МУҲИТИ КОМПОНЕНТАЛАРИ БИЛАН ИШЛАШ ТЕХНОЛОГИЯЛАРИ

Label, Edit, Memo матн компонентлари ва Button тугмачаси

Label белгиси. Белги тушунтиришлар, номлар, мавзулар ва бошқа ҳар хил турдаги матнли маълумотларни экранга жойлаштириш учун ишлатилади. Белги учун **Caption** асосий хоссалардан бири бўлиб, унда экранга чиқариладиган матн жойлашади.

Матнни экранга жойлаш учун Delphiнинг **Standart** палитрасидан (ускуналар панелидан) “**A”** пиктограммаси белгиланиб форма устига келинади ва сичқонча тугмачасини босган ҳолда матн жойлаштирилиши лозим

бўлган жой ажратилади. Натижада **Label1** матн майдони ҳосил қилинади ва **Caption** хоссасига кирилиб керакли матн терилади.

Матнга ишлов бериш учун (масалан, катталаштириш ёки кичиклайтириш; курсив ёки қалин қилиш ва бошқа) яъни унга ўзгартириш киритиш учун керакли хосса танланиб улар ўзгартирилади. Маслан, киритилган матнни катталаштириш ёки кичиртиш учун олдин матн майдони ажратилиб, кейин **Font** хоссасига кирилади ва мулоқот дарчасидан шрифт, унинг ўлчами ва ранги танланиб Ок тугмаси босилади.

Label компонентаси нафақат маълумотларни экранга жойлаш-тириш учун хизмат қилади, балки дастур натижаларини чиқаришда ҳам ишлатиш мумкин. Бунинг учун дастурда **Label5.caption:='Дастур натижаси';** буйруғи берилиши керак. Мисол, **Label5.caption:='Ечим='+s;** бу ерда **s:String** ўзгарувчиси.

Edit киритиш қатори. Edit киритиш қатори матнни бир қатордан киритиш ва уни таҳрирлаш учун ишлатилади.

Матн киритиш қаторини экранга жойлаш учун Delphiнинг **Standart** палитраси (ускуналар панели) дан "**ab"** пиктограммаси белгиланиб форма устига келинади ва сичқонча тугмачасини босган ҳолда матн киритилиши лозим бўлган жой ажратилади. Натижада **Edit1** матн киритиш майдони ҳосил қилинади. Матнни киритиш дастур ишчи ҳолатига ўтилганда бажарилади.

Матн қаторига киритилган маълумот фақат матн, яъни String (қатор) бўлиб ҳисобланади. Edit киритиш қаторида киритилган маълумотни дастурда ўқиб ва уни рақамга ўтказиш учун кўп ҳолларда Val функциясидан фойдаланилади. Бу функция Турбо Паскалда қуйилагича ёзилади. **Val(Edit1.Text,a,cod) -** бу ерда a: Real; - ўзгаручиси бўлиб, Edit1.Text майдонидаги маълумотни рақам қилиб ўзлаштиради. **cod: Integer;** деб эълон қилинади.

Memo матн чиқариш қатори. Memo матнларни бир неча қатор қилиб чиқариш учун ишлатилади.

Memo матн чиқариш қаторини экранга жойлаш учун Delphiнинг **Standart** палитраси (ускуналар панели) дан "**ab"** пиктограммаси ёнидаги Memo тугмаси белгиланиб форма устига келинади ва сичқонча тугмачасини босган ҳолда матн чиқарилиши лозим бўлган жой ажратилади. Натижада **Memo1** матн чиқариш майдони ҳосил қилинади. Бу матн чиқариш майдони дастурда натижаларни чиқаришда қўл келади. Натижани чиқаоишда у дастур ичида қуйидагича ишлатилади. **Memo1.Lines.add('Ечим='+S);**

Memo майдонини тозалаш эса натижани чиқаришдан олдин модулда **Memo1.Clear;** буйруғини бериш билан амалга оширилади.

Button тугмачаси. Button тугмачаси босилиши натижасида кутилиши лозим бўлган жараёнлар (масалан, ҳисоблашлар ёки бажарилиши лозим бўлган операциялар) бажарилишга туширилади.

Button тугмачасини экранга жойлаш учун Delphiнинг **Standart** палитраси (ускуналар панели) дан "**Ok"** пиктограммаси белгиланиб форма устига келинади ва сичқонча тугмачасини босган ҳолда тугмача қўйилиши лозим бўлган жой ажратилади. Натижада **Bottom1** тугмачаси ҳосил қилинади. Тугмача номини ўзгартириш **Caption** хосасига кирилиб ўзгартирилади.

Дастурдаги ҳисоблаш жараёнлари, киритиш ва чиқариш операциялари ҳосил қилинган тугмачани икки марта тез-тез босиш билан "собитияни қийта ишлаш" дарчасига ўтилиб, у ердан модул ичига керакли операторларни ёзиш билан амалга оширилади.

Бошланғич форма иловасини яратиш

Delphiда бошланғич формани тузиш форма Form1 хоссаларини ўзгартириш билан бошланади. Форма

хоссаларини унинг ташқи кўринишини, яъни унинг ўлчами, экранда жойлашиши, ойнаси кўриниши ва сарловҳа матнини аниқлаб беради. Унинг хоссалари Object Inspector ойнасида келтирилган бўлиб, ойнанинг чап устунида хосса номлари ва ўнг устунида унинг қийматлари берилган.

Формага янги компонентаналарни жойлаштириш унча ката қийинчилик туғдурмайди. Бирор бир компонентани жойлаштириш учун уни компонентаналар политрасидан белгилаб олиб, кейин уни бир марта чиқиллатиш керак ва кейин хоссаларини ўзгартириш учун унинг формадаги кўринишини сичқончада яна бир марта чиқиллатиш зарур.

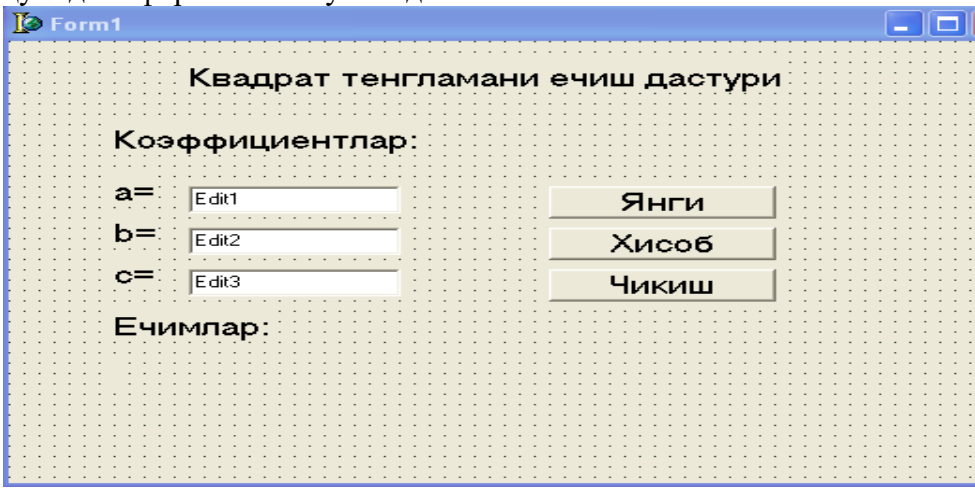
Масалан, Label (метка) компонентасини формага жойлаштириш учун уни Standart компонентаналар политрасидан топиб у сичқончада бир марта чиқилатилади, натижада формада у Label1 ном билан жойлашади. Бу компонента формага ҳар хил матнларни жойлаштириш учун хизматқилади. Унинг бошланғич ҳолатини ва хоссаларини ўзгартириш учун уни формадан белгилаймиз. Унинг Label1 стандарт номини ўзгартириш учун объект инспектори ойнасидан Caption хоссасига кирамиз ва Label1 номни ўчириб ўрнига, масалан, “Менинг биринчи дастурим” деган сўзни ёзамиз. Ёзилган матн ранги ва ўлчамини ўзгартириш эса Font хоссасига кирилиб Size ва Color параметрларини ўзгартириш билан амалга оширилади.

Яна бир компонентани - Botton (тугмача) компонентасини формага жойлаштирайлик. Бу компонента дастурчи томонидан берилган (киритилган) дастур кодларини ишга тушириш учун мўлжалланган. Унга ходисаларни қайта ишловчи (On Click - обработчик события) дейилади.

М и с о л 1.

Delphiнинг имкониятларини ва унинг визуал лойиҳалаш воситаси технологиясини намойиш этиш учун мисол тариқасида квадрат тенглама ечимларини топиш дастурини яратишни қарайлик. Бунинг учун лойиҳанинг бошланғич элементларини яратишдан бошлайлик. Delphi фойдаланувчига Form1 номли стандарт формани таклиф этади.

Формага янги, ҳисоб ва чиқиш тугмачаларини жойлаштириш учун стандарт компонентлар палитрасидан форманинг керакли жойларига қўйилади ва кейин улар номлари, яъни қийматлари хоссадан аниқланади. Натижада қуйидаги формага эга бўлинади.



Ҳодиса ва уни қайта ишлаш. Яратилган форма илованинг қай тарзда ишлашини кўрсатиб беради. Формадаги буйруқ тугмачалари бирор иш бажариши учун улар сичқончада кўрсатилиб чиқиллатилади. Сичқончада тугмачани чиқиллатиш (босиш) ходисага мисол бўлиб, у илованинг ишлаш жараёнида ҳосил бўлади. Бу ерда ҳодиса сўзини юз берадиган жараён деб тушиниш керак.

Ҳодисаларга жавоб Delphiда уларнинг қайта ишловчи процедуралар кўринишида ташкил қилинади.

Turbo Pascal тилида ёзиладиган бу процедуралар ҳодиса қайта ишловчиси (“обработчик”) деб аталади.

Delphi автоматик равишда қайта ишловчига иккита қисмдан иборат ном беради. Биринчи қисм ном формани, объектга кирувчиларни ўз ичига олиб, иккинчи қисм ном эса айнан объект ўзини ва қайта ишловчини акс эттиради. Бизнинг мисолимизда форма номи - Form1, биринчи буйруқ тугмаси номи “ҳисоб” - Button1, қайта ишловчи номи эса - Click. Энди Begin ва End орасига қайта ишловчи бажарувчи Паскаль тилидаги операторларни қуйидаги процедурада киритиш мумкин. Бу процедура “ҳисоб” тугмасини икки марта тез-тез чиқиллатиш билан экранга чақирилади.

```
Procedure Tform1.Button1click(Sender:Tobject);
 Var
   A,B,C:Real;
   D:Real;
   X1,X2:Real;
   S1,S2:String[7];
  Code:Integer;
 Begin
       Val(Edit1.Text,a,Code);
       Val(Edit2.Text,b,Code);
       Val(Edit3.Text,c,Code);

          If a=0
              Then
                  Label6.Caption:=’Хато! ’+Chr(13)
                      +’Ноъмалум иккинчи даражаси коэффиценти’
                      +Chr(13)+’нольга тенг’

              Else Begin
                  d:=b*b-4*a*c;
```

```
            x1:=(-b+Sqrt(d))/(2*a);
            x2:=(b+Sqrt(d))/(2*a);
            Str(x1:7:3,S1);
            Str(x2:7:3,S2);
            Label6.Caption:='Тенглама илдизлари:'
                  +Chr(13)+'x1='+S1
                  +Chr(13)+'x2='+S2;
    End;
  End;
```

Келтирилган дастур матнида Turbo Pascalнинг оддий Read ва Write (Киритиш ва чиқариш) операторлари ишлатилмаган. Ўзгарувчилар қийматини киритиш тахрирлаш майдонидан Text хоссасига мурожаат қилиш билан амалга оширилади. Киритилган ўзгарувчилар қиймати матн бўлгани учун улар Val функцияси ёрдамида рақамга ўтказилади. Квадрат тенгламанинг илдизлари x1 ва x2 лар қийматлари Str функцияси орқали мос равишда s1 ва s2 ўзгарувчиларга матнли қилиб ўзатилади. Натижани экранга матн кўринишида бериш учун Label6. Caption меткасига қиймат қилиб юборилади.

Худди шундай "янги" ва "чиқиш" тугмачалари учун ҳам қайта ишловчи процедураларини ташкил қилиш керак. Улар матнлари қуйидаги кўринишга эга.

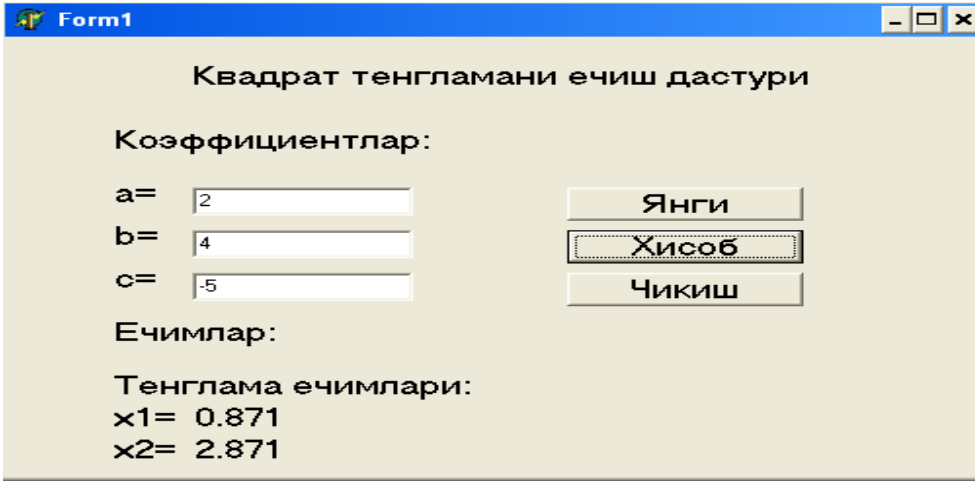
```
Procedure Tform1.Button2Click(Sender:Tobject);
  Begin
   Edit1.Text:=' ';
   Edit2.Text:=' ';
   Edit3.Text:=' ';
   Label2.Caption:=' ';
   Edit1.SetFocus;
  End;

Procedure Tform1.Buton3click(Sender: Tobject);
```

```
Begin
 Form1.Close;
End;
```

Лойиҳани сақлаш. Иловани компиляция қилиш ва ишга тушириш. Лойиҳани сақлашда Delphi бир неча файл ташкил қилади. Айримлари лойиҳа бутун лойиҳани тавсифлашни, бошқалари форма ва дастур модулини тавсифлашни ўз ичига олади. Агар ҳали сақланмаган лойиҳа бўлса Файл (File) менюсидан Сохранить проект (Save Project) буйруғи берилади ва кейин дастур модули ва проект номи берилади.

Лойиҳани боғлаб бўлгандан сўнг Compile менюсидан compile (Компилировать) буйруғи берилади. Агар дастурда синтаксик хато бўлмаса экранда компиляция тўғри ўтганлиги ҳақида хабар берилади. Агар компиляция дастурда қандайдир хатони топса хато ҳақида экранга маълумот беради. Компиляциядан тўғри ўтган дастур учун махсус - .exe кенгайтмали файл тузиб беради ва у файлни Delphi тизимисиз ишлатиш мумкин.

Delphi тизимидан чиқмасдан туриб иловани ишга тушириш мумкин, бунинг учун Run менюсининг Run буйруғини ёки F9 тугмачасини босиш кифоя бўлади. Юқоридаги мисол учун илова ишга туширилиб a, b ва c қийматлари киритилиб “хисоб” тугмаси босилса дастур қуйидаги натижани экранга чиқади.



Процедура TForm1.Button2Click “янги” тугмачасини сичқончада чиқиллатиш билан ишлайди ва тахрирлаш майдонига курсорни коэффициент қийматларини киритиш учун олиб келиб қўйади.

Процедура TForm1.Button3Click “тамом” тугмачасини сичқончада чиқиллатиш билан ишлайди ва формани ёпади.

Танлаш тугмаларини ўрнатиш

RadioGroup гурухли танлаш тугмалари иловалар яратишда бир неча вариантлардан бирини танлаш учун ишлатилади. Бу компонента Standart компонентаплар палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Унинг асосий хоссаси Items бўлиб, у тугмалар номлари рўйхатини ўзида сақлайди. Тугмалар номлари рўйхатини киритишда олдин RadioGroup тугмаси учун формадан жой ажратилади ва кейин Items хоссаси кўрсатилиб, ундан уч нуқтали тугмача босилади, натижада StringList Editor ойнаси очилади. Бу ойнадан танлаш тугмалари номларининг ҳар қайсиси янги қатордан

киритилади ва кейин Ok тугмаси босилади. Формага RadioGroup гуруҳли танлаш тугмаси жойлаштирилганда у RadioGroup1 ном билан ёзилади. Бу номни бошқа мос номга алмаштириш Caption хоссасига кириб амалга оширилади.

CheckListBox компонентаси рўйхатдан бир нечтасини танлаш имконини беради. **CheckListBox** компонентаси **Additional** палитрасида жойлашган. RadioGroup боғлик переключателларни, **CheckListBox** эса боғлиқ бўлмаган переключателларни бирлаштиради. Бунда ключател уч хил ҳолатда бўлиши мумкин:

-ёқилган (включен) -тўғри белгиси;

-ўчирилган (выключен) -бўш белгиси;

-нейтрал ҳолат -кўкиш рангда тўғри белгиси.

CheckListBox нинг асосий хоссалари:

AllowGryer -учинчи нейтрал ҳолат вариантини ишлатишни тақиқлайди;

Items –Танлаш тугмалари номлари рўйхатини сақлайди.

М **ис о л 2**.

Edit киритиш қаторида терилган матн ҳолатини ўзгартирувчи илова яратиш.

Е ч и ш

1.Янги илова яратамиз.

2.Формага Standart компоненталар палитрасидан RadioGroup компонентасини уч марта RadioGroup1, RadioGroup2, RadioGroup3 номлар билан ўрнатамиз ва унинг caption хоссаси қийматига мос равишда “ёзув”, “ўлчам” ва “ранг” қийматларни берамиз

3.Items хоссасига ўтамиз. Ундан уч нуқтали тугмачани босиб, StringList Editor ойнасига кирамиз ва бу ойнадан

ключателлар номларини ҳар қайсисини янги қатордан киритамиз:

RadioGroup1 -компонентаси учун
- обычный
- курсив
- полужирный
- полужирный курсив

RadioGroup2 -компонентаси учун
- 8
- 10
- 12
- 14

RadioGroup3 -компонентаси учун
- черный
- зеленый
- красный
- синий

Ҳар қайсиси учун киритишни тугатгандан сўнг Ok тугмаси босилади.

4. **CheckListBox** компонентасини **Additional** палитрасидан олиб формага ўрнатамиз ва унинг Items хоссасига кириб выключателлар номларини киритамиз.
- Зачеркнутый
- Подчеркнутый

5.**Edit** компонентасини **Standart** палитрасидан олиб формага ўрнатамиз ва унинг Text хоссасига “Компьютер” қийматини киритамиз.

6.Мос равишда **CheckListBox** ва **Edit1** компонентаси рамкалари юқорисига Label1 ва label2 меткаларини ўрнатамиз ва уларнинг Coption хоссасига “Атрибутлар” ва “Образецлар” қйиматини берамиз.

7.RadioGroup1 компонентаси майдонини икки марта тез босамиз ва пайдо бўлган кодларни киритиш майдонига қуйидаги кодларни киритамиз.

Case RadioGroup1.ItemIndex of

```
  0: Edit1.Font.Style:=[ ];
  1: Edit1.Font.Style:=[FsItalic];
  2: Edit1.Font.Style:=[FsBold];
  3: Edit1.Font.Style:=[FsItalic,FsBold];
 End;
```

8.RadioGroup2 компонентаси майдонини икки марта тез тез босамиз ва пайдо бўлган кодларни киритиш майдонига қайидаги кодларни киритамиз.

```
 Case RadioGroup2.ItemIndex of
  0: Edit1.Font.Size:=8;
  1: Edit1.Font.Size:=10;
  2: Edit1.Font.Size:=12;
  3: Edit1.Font.Size:=14;
 End;
```

9.RadioGroup3 компонентаси майдонини икки марта тез босамиз ва пайдо бўлган кодларни киритиш майдонига қайидаги кодларни киритамиз.

```
 Case RadioGroup3.ItemIndex of
  0: Edit1.Font.Color:=ClBlack;
  1: Edit1.Font.Color:=ClGreen;
  2: Edit1.Font.Color:=ClRed;
  3: Edit1.Font.Color:=ClBlue;
 End;
```

10. CheckListBox компонентаси майдонини икки марта тез босамиз ва пайдо бўлган кодларни киритиш майдонига қайидаги кодларни киритамиз.

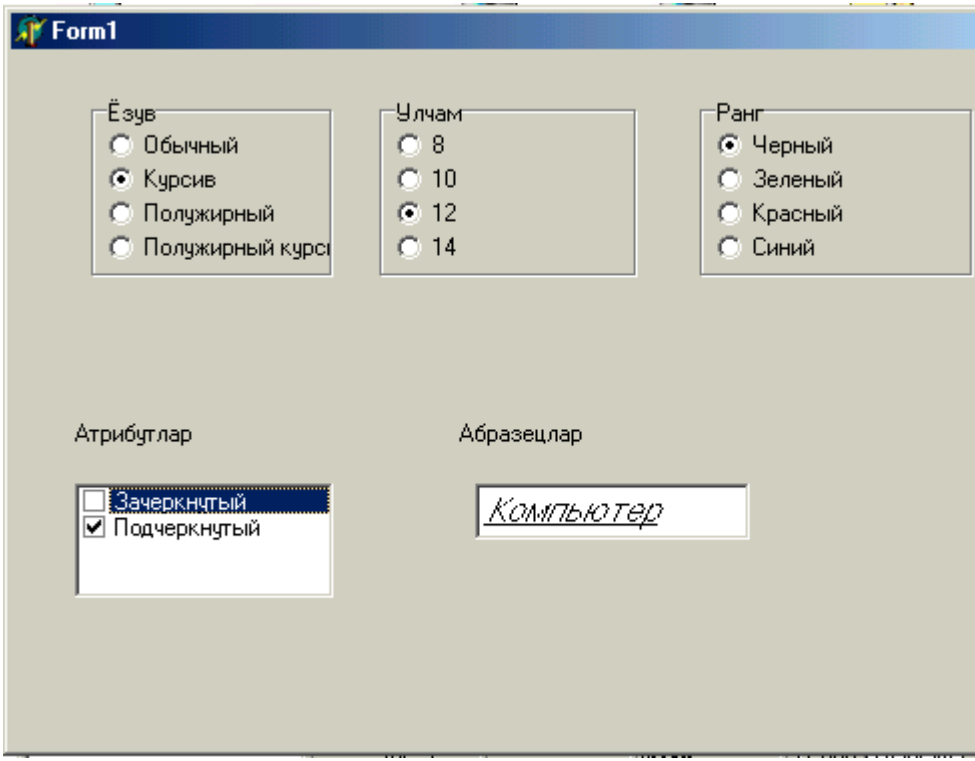
```
 If CheckListBox1.Checked[0]
  Then Edit1.Font.Style:=Edit1.Font.Style+[FsStrikeOut]
  Else Edit1.Font.Style:=Edit1.Font.Style-[FsStrikeOut];
```

```
If CheckListBox1.Checked[1]
 Then Edit1.Font.Style:=Edit1.Font.Style+[FsUnderLine]
 Else Edit1.Font.Style:=Edit1.Font.Style-[FsUnderline];
```

11.Тузилган лойиҳа (проект) яъни Project1 ва Unit1 стандарт модул номларини мос номлар билан алмаштириб сақлаймиз.

12.Янги ном билан сақланган лойиҳа (проект), яъни илова F9 тугмачасини босиш билан ишга тушурилади.

Илова ишга туширилганда унинг қуйидаги кўриниши экранда намоён бўлади. Энди илова билан ишлаш мумкин.



Ташкил қилинган модулнинг тўлиқ кўринишини келтирамиз.

```
unit pxx2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, CheckLst;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    CheckListBox1: TCheckListBox;
    Label1: TLabel;
    Label2: TLabel;
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    RadioGroup3: TRadioGroup;
    procedure RadioGroup1Click(Sender: TObject);
    procedure RadioGroup2Click(Sender: TObject);
    procedure RadioGroup3Click(Sender: TObject);
    procedure CheckListBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}
procedure TForm1.RadioGroup1Click(Sender: TObject);
```

```
begin
  Case RadioGroup1.ItemIndex of
    0: Edit1.Font.Style:=[ ];
    1: Edit1.Font.Style:=[FsItalic];
    2: Edit1.Font.Style:=[FsBold];
    3: Edit1.Font.Style:=[FsItalic,FsBold];
   End;
{    CheckListBox1ClickCheck(Self);}
end;

procedure TForm1.RadioGroup2Click(Sender: TObject);
begin
  Case RadioGroup2.ItemIndex of
    0: Edit1.Font.Size:=8;
    1: Edit1.Font.Size:=10;
    2: Edit1.Font.Size:=12;
    3: Edit1.Font.Size:=14;
   End;
end;

procedure TForm1.RadioGroup3Click(Sender: TObject);
begin
Case RadioGroup3.ItemIndex of
    0: Edit1.Font.Color:=ClBlack;
    1: Edit1.Font.Color:=ClGreen;
    2: Edit1.Font.Color:=ClRed;
    3: Edit1.Font.Color:=ClBlue;
   End;
end;

procedure TForm1.CheckListBox1Click(Sender: TObject);
begin
  If CheckListBox1.Checked[0]
   Then Edit1.Font.Style:=Edit1.Font.Style+[FsStrikeOut]
   Else Edit1.Font.Style:=Edit1.Font.Style-[FsStrikeOut];
```

```
 If CheckListBox1.Checked[1]
  Then Edit1.Font.Style:=Edit1.Font.Style+[FsUnderLine]
  Else Edit1.Font.Style:=Edit1.Font.Style-[FsUnderline];
 end;

end.
```

ListBox ва ComboBox компоненталари

ListBox компоненти рўйхат ва массив кўринишдаги маълумотларни экранга акс этдиришда ишлатилади. Маълумотларни киритишда эса Edit компонентасидан фойдаланилади. ListBox компонентаси Standart компоненталар палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формадан рўйхат учун жой ажратилади ва кейин эса хоссалари аниқланади.

Унинг айрим асосий хоссалари:

Items -рўйхат элементларини беради;

Sorter -рўйхат элементларини алфавит бўйича автоматик равишда тартиблайди;

Clear -барча рўйхат элементларини ўчиради.

ComboBox компоненти рўйхат ва массив кўринишдаги маълумотларни экрандан киритиш учун ишлатилади. У ListBox ва Edit компоненталарининг биргаликдаги ишини бир ўзи бажаради. Ташқи кўринишдан бу компонент оддий Edit киритиш қаторини эслатади. Унинг ўнг қисмида пастга белгиси бўлиб, киритилеётган маълумотларни кўриб бориш мумкин. Бу компонента Standart компоненталар палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формадан рўйхат учун жой

ажратилади ва кейин эса хоссалари аниқланади. Унинг айрим асосий хоссалари:

DropDownCount -рўйхатдаги экранга чиқадиган маълумотлар сонини аниқлайди. Бу хоссанинг бошланғич қиймати 8 га тенг бўлади. Агар экранга чиқадиган маълумотлар сонини 10 та бўлсин десак, унда унинг қийматини 10 га ўзгартириш керак бўлади. Агар киритилган маълумотлар ундан ортиқ бўлса у ҳолда пастга ва юқорига силжитиш тугмачаси автоматик равишда пайдо бўлади;

Style -рўйхатдаги маълумотнинг кўринишини тасвирлайди;

Text -рўйхатдаги киритилган маълумот матн эканини балдиради.

М и с о л 3.

Бутун қийматли A(10) массив элементлари ичидан энг ката ва энг кичиклари топилсин. Иловада Listbox компонентасини ишлатинг.

Е ч и ш

1.Янги илова яратамиз.

2.Формага Standart компоненталар палитрасидан Listbox компонентасини ListBox1 ном билан, Edit компонентасини Edit1 ном билан ва иккита Botton1 ва Botton2 тугмаларини ўрнатамиз.

3.Edit компонентасинининг text хоссасига кириб Edit1 қийматини бўш қатор қилиб берамиз.

4.Botton1 ва Botton2 тугмачаларининг Caption хоссасига кириб уларни “Киритиш” ва “Ечиш” қийматига тенглаштирамиз.

5.”Ечиш” тугмаси пастига “Memo” компонентасини “Memo1” ном билан ўрнатамиз.

6.Форма устига сичқончада икки марта босиб, кодларни ёзиш ойнасига ўтамиз ва қуйидагиларни киритамиз:

```
i:=0;
ListBox1.Clear;
```

Inteface бўлимига массив ва ишлатиладиган ўзгарувчиларни Var сўзидан кейин тавсифлаймиз.

```
a:Array[1..10] of integer;
k,i,maxx,minn: Integer;
s1,s2: String;
```

7.”Киритиш” тугмасини активлаштириш учун уни икки марта тез-тез босиб, кодларни ёзиш ойнасига ўтамиз ва қуйидагиларни киритамиз:

```
ListBox1.Items.Add(Edit1.text);
i:=i+1;
a[i]:=StrToInt(Edit1.text);
Edit1.SetFocus;
```

8.”Ечиш” тугмасини активлаштириш учун уни икки марта тез-тез босиб, кодларни ёзиш ойнасига ўтамиз ва қуйидагиларни киритамиз:

```
minn:=a[1];
maxx:=a[1];
For k:=1 to 10 do
  Begin
    If minn>a[k] Then Minn:=a[k];
    If maxx<a[k] Then maxx:=a[k];
  End;
Str(maxx:5,S1);
Str(minn:5,S2);
```

```
Memo1.Clear;
Memo1.Lines.Add('Энг каттаси='+s1);
Memo1.Lines.Add('Энг кичиги ='+s2);
```

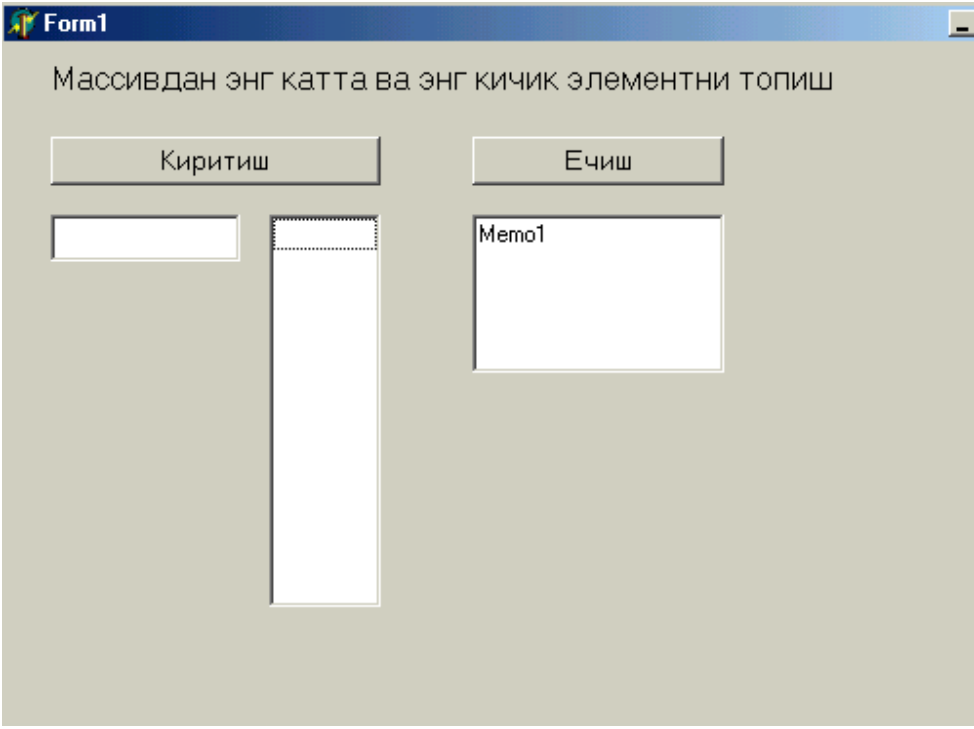
9.Киритиш фокуси Edit1 киритиш қаторида туриши учун уни икки марта босиб кодларни ёзиш ойнасига ўтамиз ва қуйидагини киритамиз:

```
If key=13 Then Button1.SetFocus;
```

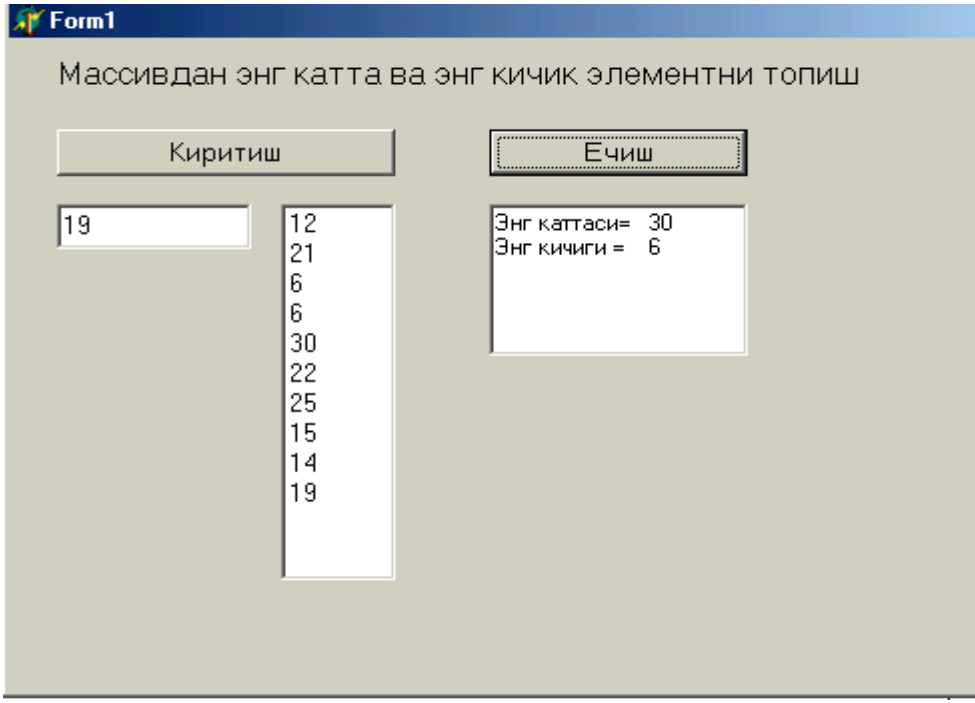
10.Тузилган лойиҳа (проект) яъни Project1 ва Unit1 стандарт модул номларини мос номлар билан алмаштириб сақлаймиз.

11.Янги ном билан сақланган проект, яъни илова F9 тугмачасини босиш билан ишга тушурилади.

Илова ишга туширилганда унинг қуйидаги кўриниши экранда намаён бўлади.



Массив элементи қийматларини киритиб, “Ечиш” тугмасини босамиз ва натижада қуйидаги натижага эга бўламиз.



Ташкил қилинган модулнинг тўлиқ кўринишини келтирамиз.

```
unit pxx3;

interface

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

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Button1: TButton;
    Button2: TButton;
```

```
    Edit1: TEdit;
    Memo1: TMemo;
    Label1: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Edit1KeyDown(Sender: TObject;var key:
Word;
      Shift: TshiftState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  a:Array[1..10] of integer;
  k,i,maxx,minn:Integer;
  s1,s2:String;
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
  ListBox1.Items.Add(Edit1.text);
  i:=i+1;
  a[i]:=StrToInt(Edit1.text);
  Edit1.SetFocus;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  i:=0;
  ListBox1.Clear;
```

```
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  minn:=a[1];
  maxx:=a[1];
  For k:=1 to 10 do
   Begin
       If minn>a[k] Then Minn:=a[k];
       If maxx<a[k] Then maxx:=a[k];
   End;
   Str(maxx:5,S1);
   Str(minn:5,S2);
   Memo1.Clear;
   Memo1.Lines.Add('Энг каттаси='+s1);
   Memo1.Lines.Add('Энг кичиги ='+s2);
end;

procedure TForm1.Edit1KeyDown(Sender: TObject;var
key: Word;
     Shift: TshiftState);
  Begin
    If key=13 Then Button1.SetFocus;
  End;
end.
```

М и с о л 4.

Бутун қийматли A(10) массив элементлари ичидан энг ката ва энг кичиклари топилсин. Иловада ComboBox компонентасини ишлатинг.

Е ч и ш

1.Янги илова яратамиз.

2.Формага Standart компонентaлар палитрасидан **ComboBox** компонентасини ComboBox1 ном билан, Memo компонентасини Memo1 ном билан ва иккита Botton1 ва Botton2 тугмаларини ўрнатамиз.

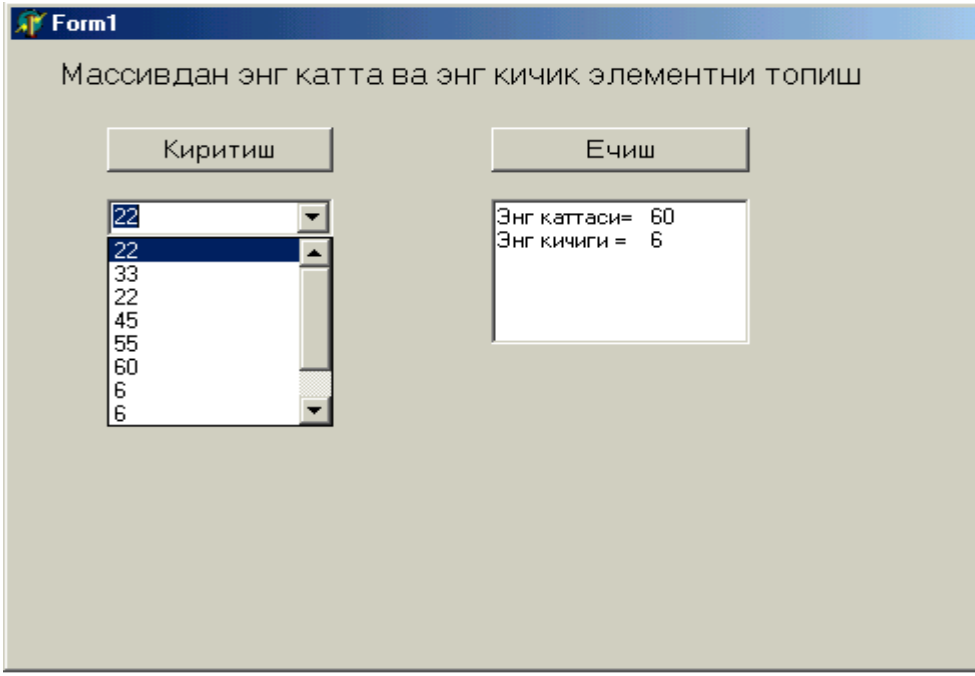
3.Олдинги мисол каби бу компонентaларнинг хоссаларини ҳам ўрнатамиз ва дастур кодларини ҳам киритамиз. Ҳамма дастур кодлари “киритиш” тугмасига боғлиқ, яъни Botton1 модули кодлари қуйидагича бўлади.

```
ComboBox1.Items.Add(Combobox1.text);
i:=i+1;
a[i]:=StrToInt(ComboBox1.text);
ComboBox1.SetFocus;
```

4.Тузилган лойиҳа (проект) яъни Project1 ва Unit1 стандарт модул номларини мос номлар билан алмаштириб сақлаймиз.

5.Янги ном билан сақланган проект, яъни илова F9 тугмачасини босиш билан ишга тушурилади.

Илова ишга туширилганда унинг қуйидаги кўриниши экранда намаён бўлади.



Ташкил қилинган модулнинг тўлиқ кўринишини келтирамиз.

```
unit s1p;

interface

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

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Memo1: TMemo;
```

```
    Label1: TLabel;
    ComboBox1: TComboBox;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
   private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  a:Array[1..10] of integer;
  k,i,maxx,minn:Integer;
  s1,s2:String;
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
 begin
  ComboBox1.Items.Add(ComboBox1.text);
  i:=i+1;
  a[i]:=StrToInt(ComboBox1.text);
  ComboBox1.SetFocus;
 end;

procedure TForm1.FormCreate(Sender: TObject);
 begin
  i:=0;
  ComboBox1.Clear;
 end;

procedure TForm1.Button2Click(Sender: TObject);
 begin
  minn:=a[1];
```

```
 maxx:=a[1];
 For k:=1 to 10 do
  Begin
     If minn>a[k] Then Minn:=a[k];
     If maxx<a[k] Then maxx:=a[k];
  End;
  Str(maxx:5,S1);
  Str(minn:5,S2);
  Memo1.Clear;
  Memo1.Lines.Add('Энг каттаси='+s1);
  Memo1.Lines.Add('Энг кичиги ='+s2);
end;

End.
```

StringGrid жадвал компонентаси

StringGrid жадвал компонентаси икки ўлчовли маълумотларни, масалан матрица элементлари қийматини экранда жадвал кўринишда тасвирлаш, улар қийматини киритиш ва таҳрирлаш учун ишлатилади. Жадвал қатор ва устун нумерлари нулдан бошланади. Жадвал устун ва қаторлар сонини кераклича ўзгартириш мумкин. Бу унинг хоссаси ёрдамида аниқланади. Жадвалнинг ҳар бир кесишган устун ва сатри ячейка дейилиб, унга киритилган маълумот символ қатори бўлиб аниқланади. Масалан, (3,5) ячейка тўртинчи устун ва олтинчи қаторда жойлашган.

StringGrid жадвал компонентасининг асосий хоссалари:

ColCount -жадвалдаги устунлар сонини аниқлайди;

RowCount -жадвалдаги сатрлар сонини аниқлайди;

FixedCols -фиксирланган устунлар сонини аниқлайди;

FixedRows -фиксирланган сатрлар сонини аниқлайди;

Options –жадвад ҳолатини аниқлайди (аниқлаш унинг памаметрларига асосан бажарилади, масалан GoEditing параметр true қийматга эга бўлса ячейкани таҳрирлаш мумкин, акс ҳолда мумкин эмас. Бу параметрларни аниқлаш учун Options хоссасига ўтиб у икки марта тез-тез босилади);

ColWidths -жадвалдаги ҳар бир устун кенглигини аниқлайди;

DefaultColWidth -жадвалнинг бошланғич устунлар кенглигини аниқлайди;

DefaultRowHeight -жадвал сатрининг бошланғич баландлигини аниқлайди;

FixedColor -фиксирланган ячейка рангини аниқлайди;

RowHeights -жадвал сатри баландлигини аниқлайди;

Cells -символ қаторли икки ўлчамли массивни аниқлайди.

М и с о л 5.

Бутун қийматли A(4,4) массив элементлари йиғиндиси ва ўрта армфметик қиймати топилсин.

Е ч и ш

1.Янги илова яратамиз.

2.Формага Additional компоненталар палитрасидан **StrinGrid** компонентасини StrinGrid1 ном билан, Standart компоненталар палитрасидан Memo компонентасини Memo1 ном билан ва Botton1 тугмаларини ўрнатамиз.

3.StrinGrid компонентасининг хоссаларини ўрнатамиз.

FixedCols -0,

FixedRows -0,

ColCount -4,

RowCount -4.

Демак, ҳосил қилинадиган жадвал 4 та устун ва 4 та сатрга эга.

Option хоссасига кирамиз ва уни икки марта тез-тез чиқиллатамиз. У ердан GoEditing параметрини True қийматга тенглаштирамиз.

4.Botton1 тугмасининг Coption хоссасига кириб унинг номини “Ечиш” номига ўзгартирамиз.

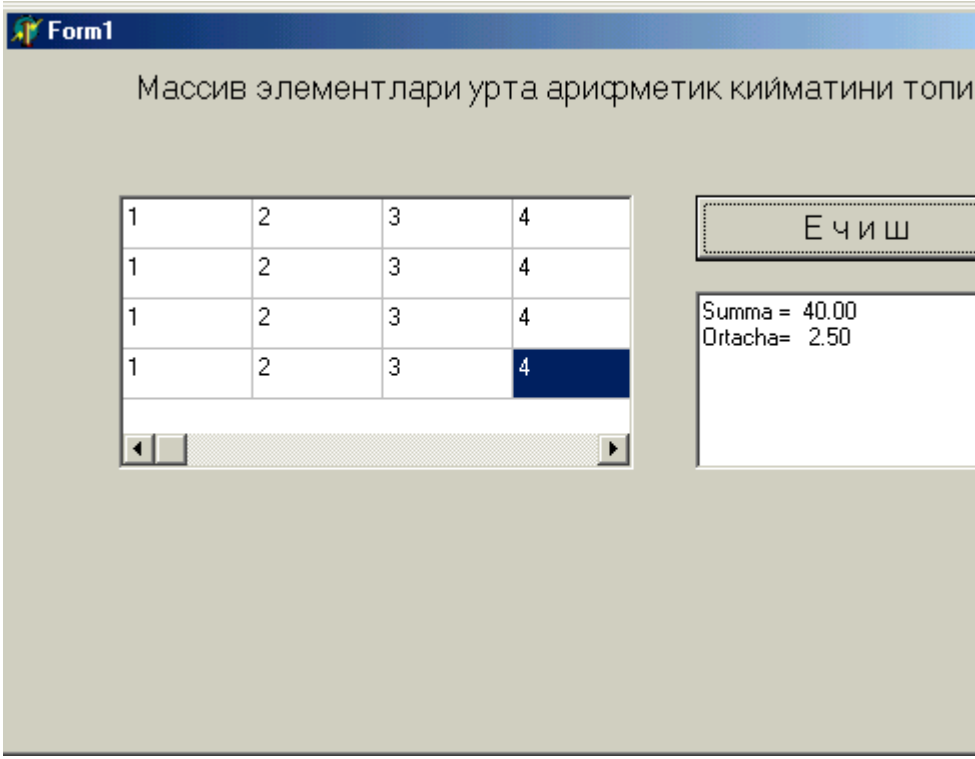
5.”Ечиш” тугмасини активлаштирамиз, яъни уни икки марта тез-тез босиб дастур кодларини ёзиш ойнасига ўтамиз ва қуйидаги кодларни киритамиз.

```
Var i,j,cod:integer;
    A:array[1..4,1..4] of Real;
    S:real; s1:String;
begin
  For i:=1 to 4 do
   For j:=1 to 4 do
    Val(StringGrid1.cells[i-1,j-1],a[i,j],cod);
     S:=0;
    For i:=1 to 4 do
     For j:=1 to 4 do
      s:=s+a[i,j];
      Str(s:7:2,s1);
      Memo1.Clear;
      Memo1.Lines.add('Summa ='+s1);
      s:=s/4/4;
      Str(s:7:2,s1);
      Memo1.Lines.add('Ortacha='+s1);
end;
```

4.Тузилган лойиҳа (проект) яъни Project1 ва Unit1 стандарт модул номларини мос номлар билан алмаштириб сақлаймиз.

5.Янги ном билан сақланган проект, яъни илова F9 тугмачасини босиш билан ишга тушурилади.

Илова ишга туширилганда унинг қуйидаги кўриниши экранда намаён бўлади.



Ташкил қилинган модулнинг тўлиқ кўринишини келтирамиз.

```
unit j1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, Grids, StdCtrls;

type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
```

```
    Button1: TButton;
    Label1: TLabel;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
  Var i,j,cod:integer;
      A:array[1..4,1..4] of Real;
      S:real; s1:String;
  begin
    For i:=1 to 4 do  For j:=1 to 4 do
      Val(StringGrid1.cells[i-1,j-1],a[i,j],cod);
       S:=0;
      For i:=1 to 4 do  For j:=1 to 4 do  s:=s+a[i,j];
        Str(s:7:2,s1);
        Memo1.Clear;
        Memo1.Lines.add('Summa ='+s1);
        s:=s/4/4;  Str(s:7:2,s1);
        Memo1.Lines.add('Ortacha='+s1);
  end;

end.
```

Delphi график имкониятлари

Delphi дастурчига график дастурлар схема, чертеж, иллюстрациялар яратишга имкон беради. Дастур графикани объект (форма еки Image компонентаси) юзасига чиқаради. Объект юзасига canvas хоссаси мос келади. Объект юзасига график элемент (тўғри чизиқ, айлана, туртбурчак ва ҳоказо), чиқариш учун бу объектнинг canvas хоссасига мос усул қўллаш лозим. Мисол учун Form1.anvas. Rectangle (10,10,100,100) инструкцияси дастур ойнасида туртбурчак чизади.

Чизиш соҳаси

Юкорида кўрилган canvas хоссаси -TCanvas типидаги объектдир. График примитивларни чиқариш усуллари Canvas хоссасини абстракт чизиш соҳаси деб қарайди. Чизиш соҳаси алоҳида нуқталар - пикселлардан иборат. Пиксел ҳолати унинг горизонтал (X) ва вертикал (Y) координаталари билан аниқланади. Чап юқори пиксел координаталари (0,0). Координаталар юқоридан пастга ва чапдан ўнгга қараб ўсиб боради.

Соҳа ўлчовларини image **компонентасининг** Height и width хоссалари ва **форманинг** ClientHeight ва Clientwidth хоссалари орқали аниқлаш мумкин.

Қалам

Қалам геометрик фигураларни чизиш учун ишлатилади. Чизиқ кўриниши Треп объектининг қуйидаги жадвалда кўрсатилган хоссалари орқали аниқланади.

Треп (қалам) хоссалари.

Хосса	***Таърифи***

Color	*Чизиқ ранги*
Width	*Чизиқ калинлиги*
Style	*Чизиқ кўриниши*
Mode	*Акслантириш режими*

Қуйидаги жадвалда color хоссаси қиймати сифатида берилувчи номланган константалар санаб ўтилган.

Color хоссаси қийматлари.

Константа	*Ранг*	*Константа*	*Ранг*
clBlack	*Қора*	*clSilver*	*Серебристый*
clMaroon	*Каштановый*	*clRed*	*Қизил*
clGreen	*Яшил*	*clLime*	*Салатный*
clOlive	*Оливковый*	*clBlue*	*Кўк (зангори)*
clNavy	*Тим-кўк*	*clFuchsia*	*Ярко-розовый*
clPurple	*Розовый*	*clAqua*	*Бирюзовый*
clTeal	*Зелено-голубой*	*clWhite*	*Оқ*
clGray	*Кул ранг*		

Чизиқ қалинлиги width хоссаси орқали пикселларда берилади.
Чизиқ турини style хоссаси белгилайди. Қуйидаги жадвалда чизиқ турини белгиловчи номланган константалар санаб ўтилган.
Style хоссаси кийматлари.

Константа	*Чизиқ кўриниши*
psSolid	*Узлуксиз чизиқ*

psDash	*Пунктир чизиқ, узун штрихлар*
psDot	*Пунктир чизиқ, қисқа штрихлар*
psDashDot	*Пунктир чизиқ, узун ва қисқа штрихлар кетлиги*
psDashDotDot	*Пунктир чизиқ, битта узун ва иккита қисқа ш кетма кетлиги*
psClear	*Чизиқ акс этмайди*

Mode хоссаси чизиқ рангининг фон рангига муносабатини кўрсатади. Одатда чизиқ ранги Pen.Color хоссаси қиймати билан белгиланади.
Дастурчи чизиқ учун фон рангига нисбатан инверс ранг бериши мумкин. Бу ҳолда ҳатто чизиқ ва фон ранги бир хил берилган бўлса ҳам чизиқ ажралиб туради.
Қуйидаги жадвалда Mode хоссаси қиймати сифатида ишлатиш мумкин бўлган константалар берилган.
Mode хоссаси қийматлари

Константа	*Чизиқ ранги*
pmBlack	*Қора, Pen. Color хоссаси қийматига боғлиқ эмас*
pmWhite	*Оқ, Pen. Color хоссаси қийматига боғлиқ эмас*
pmCopy	*Чизиқ ранги Pen. Color хоссаси қийматига боғлиқ*
pmNotCopy	*Чизиқ ранги Pen. Color хоссаси қийматига инверс*
pmNot	*Чизиқ ранги соҳанинг мос нуқтаси рангига инверс*

Муйқалам

Муйқалам (Canvas.Brush) ёпиқ соҳаларни чизиш ва соҳа ичини бўяш учун мўлжалланган усуллардан фойдаланилади. Муйқалам объект жадвалда кўрсатилган икки хоссага эга.
TBrush (муйқалам) хоссалари.

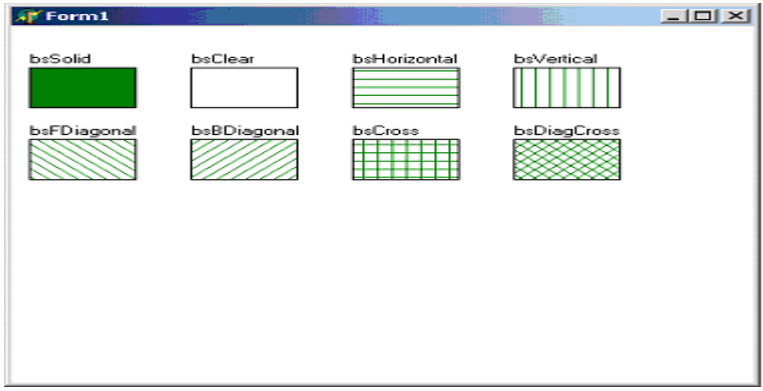
Хосса	***Таърифи***
Color	***Епиқ соҳани бўяш ранги***
Style	***Соҳани тўлдириш услуби***

Контур ичидаги соҳа бўялиши ёки штрихланиши мумкин.
Соҳани тўлдириш усулини белгиловчи константалар қуйидаги жадвалда берилган.
Brush.style хоссаси қийматлари.

Константа	***Соҳа бўяш услуби***
bsSolid	***Узлуксиз бўяш***
bsClear	***Соҳа бўялмайди***
bsHorizontal	***Горизонтал штрихлаш***
bsVertical	***Вертикал штрихлаш***
bsFDiagonal	***Диагонал штрихлаш, олдинга оғиш***
bsBDiagonal	***Диагонал штрихлаш, орқага оғиш***
bsCross	***Катакли горизонтал-вертикал штрихлаш***
bsDiagCross	***Катакли диагонал штрихлаш***

Мисол тариқасида соҳаларни бўяш усуллари дастурини келтирамиз.

Соҳани бўяш усуллари дастури ойнаси



Соҳани бўяш усуллари дастури матни

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation
```

```
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
const
bsName: array[1..8] of string =
('bsSolid','bsClear','bsHorizontal',
'bsVertical','bsFDiagonal','bsBDiagonal',
'bsCross','bsDiagCross');
var
x,y: integer;
w,h: integer;
bs: TBrushStyle;
k: integer;
i,j: integer;

begin
button1.visible:=false;
w:=60; h:=40;
y:=80;
for i:=1 to 2 do
begin
x:=10;
for j:=1 to 4 do
begin
k:=j+(i-1)*4;
case k of
1: bs:= bsSolid;
2: bs:= bsClear;
3: bs:= bsHorizontal;
4: bs:= bsVertical;
5: bs:= bsFDiagonal;
6: bs:= bsBDiagonal;
7: bs:= bsCross;
8: bs:= bsDiagCross; end;
Canvas.Brush.Color := clGreen;
```

```
Canvas.Brush.Style := bs;
Canvas . Rectangle (x, y, x+w, y-h) ;
Canvas.Brush.Style := bsClear;
Canvas.TextOut(x, y-55, bsName[k]);
x := x+w+30;
end;
y:= y+h+30;
end;
end;
end.
```

Матнни чиқариш

График объект юзасига матн чиқариш учун TextOut усули қўлланилади. Бу усулни чақириш инструкцияси қуйидаги кўринишга эга:

Объект.Canvas.TextOut(x, y, Текст)

Матн шрифти Font хоссаси қиймати билан аниқланади. Font хоссаси TFont типидаги объекдир. Қуйидаги жадвалда TFont объекти хоссалари келтирилган.

TFont объекти хоссалари

Хосса	***Таърифи***
Name	***Шрифт номи, масалан Arial***
Size	***Шрифт пунктларда катталиги***
Style	***Символлар чиқариш услуби. Қуйидаги константл орқали берилади: fsBold (полужирный), fsItalic (кур fsUnderline (подчеркнутый), fsStrikeOut (перечеркнуты***
	Бу хосса бир неча услубларни комбинациясини ол имкон беради. Масалан: Объект. Canvas . Font : = [fsBo Italic]

Color ***Символлар ранги.***

Матн чиқариш соҳаси муйқалам жорий рангига буялади. Шунинг учун матн чиқаришдан олдин Brush. Color хоссасига bsClear қийматини ёки соҳа рангига мос қийматни бериш лозим.

Мисол:

```
with Form1.Canvas do begin
Font.Name := 'Tahoma';
Font.Size := 20;
Font.Style := [fsltalic, fsBold] ;
Brush.Style := bsClear;
TextOut(0, 10, 'Borland Delphi 6');
end;
```

Textout услуби орқали матн экранга чиқарилгандан сўнг қалам матн чиқариш соҳасининг юқори ўнг бурчагига келтирилади.
Агар матн узунлиги маълум бўлмаса, чиқарилган матн ўнг чегараси координаталарини PenPos хоссасига мурожаат қилиб аниқлаш мумкин.

Мисол:

```
with Form1.Canvas do begin
TextOut(0, 10, 'Borland ') ;
TextOut(PenPos.X, PenPos.Y, 'Delphi 6');
end;
```

График примитивларни чизиш усуллари

Чизиқ

Тўғри чизиқ LineTo усули орқали амалга оширилади.

Компонент.Canvas.LineTo(x,y)

LineTo усули қалам жорий позициясидан берилган координатали нуқтагача тўғри чизиқ чизади. Бошлангич нуқтани керакли нуқтага кўчириш учун MoveTo усулидан фойдаланиш мумкин.

Туташган чизиқ

Ўзаро туташган кесмалардан иборат шаклни чизиш учун polyline усулидан фойдаланилади. Бу усул параметри TPoint типли массивдан иборат.

Polyline усулига мисол тариқасида маълум қиймат ўзгариши графигини чизувчи процедурасини келтирамиз:

```
procedure TForml.Button1Click(Sender: TObject);
var
gr: array[1..50] of TPoint;
x0,y0: integer;
dx,dy: integer;
i: integer; begin
x0 := 10; y0 := 200; dx :=5; dy := 5;
for i:=l to 50 do begin
gr[i].x := x0 + (i-l)*dx;
gr[i].y := y0 - Data[i]*dy;
end;
with forml.Canvas do begin
MoveTo(x0,y0); LineTo(x0,10);
MoveTo(x0,y0); LineTo(200,y0);
Polyline(gr);
end;
end;
```

Polyline усули ёрдамида ёпиқ кўпбурчак чизиш учун массивнинг биринчи ва охирги элементи бир нуқтанинг координаталаридан иборат бўлиши керак.

Айлана ва эллипс

Айлана ёки эллипс чизиш учун Ellipse усули чақирилади. Усулни чақириш инструкцияси умумий кўриниши:

Объект.Canvas.Ellipse(x1,y1, x2,y2).

Бу ерда x1, y1, x2, y2 – эллипсни ўз ичига олган минимал туртбурчак координаталари. Агар туртбурчак квадрат бўлса айлана чизилади.

Ей

Ёйни чизиш учун Arc усули қўлланилади ва у қуйидаги умумий кўринишга эга:

Объект.Canvas.Arc(x1,y1,x2,y2,x3,y3,x4,y4)

Бу ерда:

- x1, y1, x2, y2 - ёйга тегишли бўлган эллипс ёки айлана параметрлари;
- x3, y3 - ёй бошланғич нуктаси параметрлари;
- x4, y4 - сўнги нуқтаси параметрлари.

Ёй соат милига тескари тартибда чизилади.

Тўртбурчак

Тўртбурчак Rectangle усули билан чизилиб, бу усулни чақириш инструкцияси умумий кўриниши қуйидагича:

Объект.Canvas.Rectangle(x1, y1,x2, y2)

Бу ерда x1, y1 ва x2, y2 — чапги юқори ва ўнгги пастги бурчаклар координаталари.

RoundRec усули бурчаклари юмалоқ тўртбурчак чизишга имкон беради. RoundRec усулини чакириш инструкцияси куйидаги куринишга эга:
Объект.Canvas.RoundRec(x1,y1,x2, y2, x3, y3)
Бу ерда:

- x1, y1, x2, y2 – туртбурчак параметрлари;
- x3 и y3 — чорак кисми юмалок бурчак чизиш учун ишлатилдадиган эллипс катталиги.

Яна икки усул муйқаламдан фойдаланиб тўртбурчак чизишга имкон беради. FillRect усули ичи бўялган тўртбурчак чизади, FrameRect - фақат контур. Бу усулларда фақат битта параметрга эга -TRect типидаги структура. Қуйидаги мисолда FillRect ва FrameRect усуллари орқали форма юзасига қизил туртбурчак соҳа ва яшил контурли тўртбурчак чизувчи процедура келтирилган.

```
procedure TForm1.Button1Click(Sender: TObject);
var
r1, r2: TRect;
begin
r1 := Rect(20,20,60,40);
r2 := Rect(10,10,40,50);
with form1.Canvas do begin
Brush.Color := clRed;
FillRect(r1);
Brush.Color := clGreen;
FrameRect(r2};
end;
end;
```

Кўпбурчак

Polygon усули кўпбурчак чизишга мўлжалланган бўлиб, параметри TPoint типидаги массивдир. Массивнинг

ҳар бир элементи (x,y) майдонлари кўпбурчак учи координаталаридан иборат бўлган ёзувдир.
Қуйида polygon усули ёрдамида учбурчак чизиш процедураси келтирилган:

```
procedure TForm1.Button2Click(Sender: TObject);
var
pol: array[1..3] of TPoint;
begin
pol[1].x := 10;
polf1].y := 50;
pol[2].x := 40;
pol[2].y := 10;
pol[3].x := 70;
pol[3].y := 50;
Form1.Canvas.Polygon(pol);
end;
```

Сектор

Эллипс ёки айлана сектори pie усули билан чизилиб, чақириш инструкцияси қуйидаги умумий кўринишга эга:

Объект. Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4)

Бу ерда:

- x1, y1, x2, y2 - эллипс ёки айлана параметрлари;
- x3, y3, x4, y4 - сектор чегарасини ташкил қилувчи тўғри чизиқлар охирги нукталари координаталари**.**

Нуқта

Canvas объектининг pixels хоссаси типидаги икки ўлчовли массив бўлиб ҳар бир соҳа нуқтасининг ранги ҳақидаги маълумотни ўз ичига олади. Pixels хоссасидан фойдаланиб ихтиёрий нуқта рангини ўзгартириш, яъни нуқта чизиш мумкин. Мисол учун

Form1.Canvas.Pixels[10,10]:=clRed

Инструкцияси соҳа нуқтасини қизил ранга бўяйди.

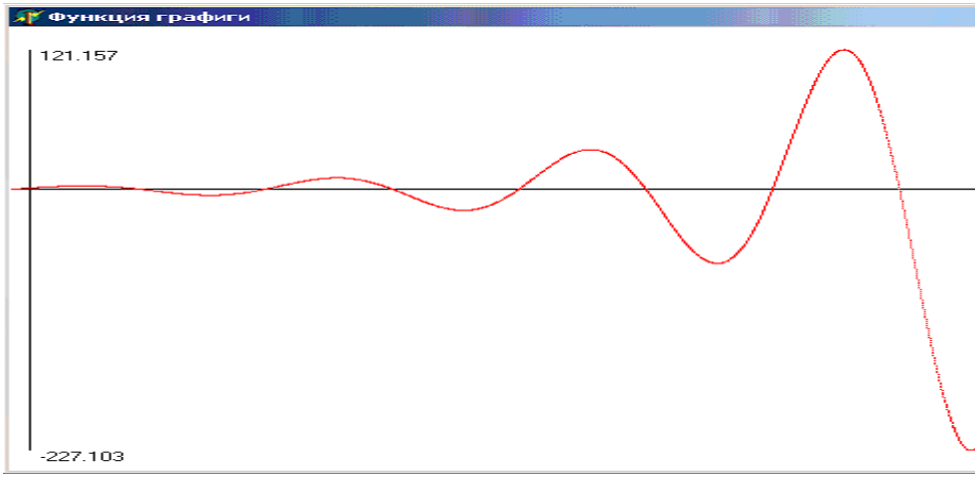
Қуйида келтирилган дастур pixels хоссасидан фойдаланиб, $y = 2 \sin(jc)\, e^{*/5}$ функцияси графигини чиқаради.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
     Function f(x:real):real;
begin
f:=2*Sin(x)*exp(x/5) ;
end;
procedure GrOfFunc;
var
x1,x2:real;
y1,y2:real;
x:real;
y:real;
dx:real;
```

```
 l,b:integer;
w,h:integer;
mx,my:real;
x0,y0:integer;
begin
l:=10;
b:=Form1.ClientHeight-20;
h:=Form1.ClientHeight-40;
w:=Form1.Width-40;
x1:=0;
x2:=25;
dx:=0.01;
y1:=f(x1);
y2:=f(x1);
x:=x1;
repeat
y := f (x);
if y < y1 then y1:=y;
if y > y2 then y2:=y;
x:=x+dx; until (x >= x2);
my:=h/abs(y2-y1);
mx:=w/abs(x2-x1);
x0:=1;
y0:=b-Abs(Round(y1*my)) ;
with form1.Canvas do
begin
// оси
MoveTo(l,b);LineTo(l,b-h);
MoveTo(x0,y0);LineTo(x0+w,y0);
TextOut(l+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
TextOut(l+5,b,FloatToStrF(y1,ffGeneral,6,3));
x:=x1; repeat
y:=f(x);
Pixels[x0+Round(x*mx),y0-Round(y*my)]:=clRed;
x:=x+dx;
```

```
until (x >= x2);
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
Button1.Visible:=false;
GrOfFunc;
end;
end.
```

Асосий вазифани GrOfFunc процедураси бажаради. Аввал [x1,x2] ораликда функциянинг максимал (y2) ва минимал (yl) қийматлари ҳисобланади. Сўнгра координаталар у ки бўйича масштаб ҳисобланади. Шундан сўнг процедура графикни қуради.



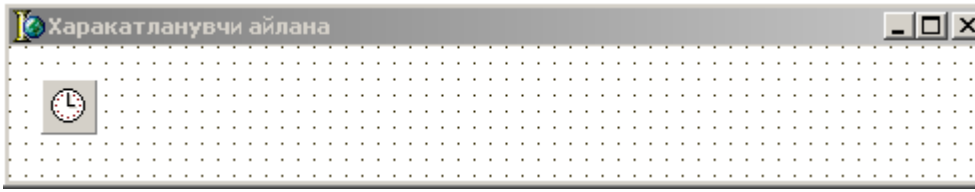
GrOfFunc процедураси томонидан қурилган график.

Келтирилган дастур универсал характерга эга. Ўзга функция графигини чизиш учун f (x), танасини ўзгартириш етарли.
Дастур тўғри ишлайди, агар функция ҳам мусбат, ҳам манфий кийматларни қабул қилса.

Мультипликация

Мультипликация дейилганда ҳаракатланувчи расм тушунилади. Расмни ҳаракатлантириш учун аввал у экранга чизилади, маълум вақтдан сўнг расмни учириб янги жойга чизилади.
Қуйидаги дастур, айлананинг чапдан ўнгга ҳаракатини кўрсатади.

Ҳаракатланувчи айлана дастури формаси



Ҳаракатланувчи айлана ***дастури матни***

```
unit Unit1;

interface

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

type
```

```
  TForm1 = class(TForm)
  Timer1: TTimer;
    procedure FormActivate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
   private
{ Private declarations }
public
{ Public declarations }
end;
 procedure Ris;
var
  Form1: TForm1;
   x,y: byte;

implementation

{$R *.dfm}

procedure Ris;
begin
form1.Canvas.Pen.Color:=form1.Color;
form1.Canvas.Ellipse(x,y,x+10,y+10);
x:=x+5;
form1.Canvas.Pen.Color:=clBlack;
form1.Canvas.Ellipse(x,y, x+10, y+10) ;
end;

  procedure TForm1.FormActivate(Sender: TObject);
begin
x:=0;
y:=10;
timer1.Interval:=50;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
```

begin
Ris;
end;
end.

График компоненталар

Image компонентаси формага расмларни жойлаштириш учун ишлатилади. Жойлаштирилиши лозим бўлган расмлар битли файллар (кенгайтмалари .Bmp), пиктограммали (кенгайтмалари .Ico), метафайллар (кенгайтмалари .wmf) бўлиши керак.

Image компонентаси Additional палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формадан расм учун жой ажратилади ва кейин эса хоссалар бўлимидан Picture хоссаси танланиб, у ердан уч нуқтали тугмача босилади. Натижада экранда расмни аниқлаш ва жойлаш учун мулоқат дарчаси очилади. Мулоқат дарчаси қуйидаги тугмачаларга эга:

Load -файлдан расмни чақириш;
Save -расмни файлга сақлаш;
Clear -танланган расмни олиб ташлаш;
Ok -танланган расмни ажратилган жойга ёзиш;
Cancel -қилинган ўзгартиришларни бекор қилиш.

Shape компонентаси формага айлана, тўртбурчак, эллипс ва бошқа шаклларни жойлаштириш учун ишлатилади. Унинг қуйидаги хоссалари мавжуд:

Brush -шаклни бўяш учун чўткача;
Pen -шакл четини чизиш учун қалам;
Shape -экранга чиқадиган шаклни аниқлайди:
StRectangle -тўртбурчак;
StSquare -квадрат;
StRoundRect -четлари айлана тўртбурчак;
StRoundSquare -четлари айлана квадрат;

StEllipse -эллипс;
StCircle -айлана.

Shape компонентаси ҳам Additional палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формадан шакл учун жой ажратилади ва кейин эса хоссалар бўлимидан Shape хоссасига кирилиб керакли шакл танланади.

PaintBox компонентаси формага чегараланган майдонда шаклларни чизиш имконини беради.

PaintBox компонентаси System палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формадан шакл учун жой ажратилади ва кейин эса хоссалар бўлимидан Shape хоссасига кирилиб керакли шакл танланади.

Timer визуал бўлмаган компонента бўлиб, формада бажариладиган маълум бир операцияларни вақт бўйича бошқаради.

Timer компонентаси System палитрасида жойлашган бўлиб, у  кўринишдаги пиктограммага эга. Бу тугмачани босиб формага олиб келиб қўйилади.

У қуйидаги хоссаларга эга:

Enabled -true қиймати ўрнатилган бўлса у бўладиган жараёнга таъсир қилади;

Interval -миллисекундларда вақт интервалини аниқлайди ва жараённинг экранга чиқишига таъсир кўрсатади. Тегмаган ҳолда 1000 (1 секунд) кўрсатади.

М и с о л.

Илова учун “заставка” яратиш.

Е ч и ш

Заставка график тасвирлар кўринишида бўлиб, программалар ишга туширилганда бир неча секунддан сўнг экранда пайдо бўлади. Унда программа номи ва унинг авторлари ҳақида маълумот бўлиши мумкин.

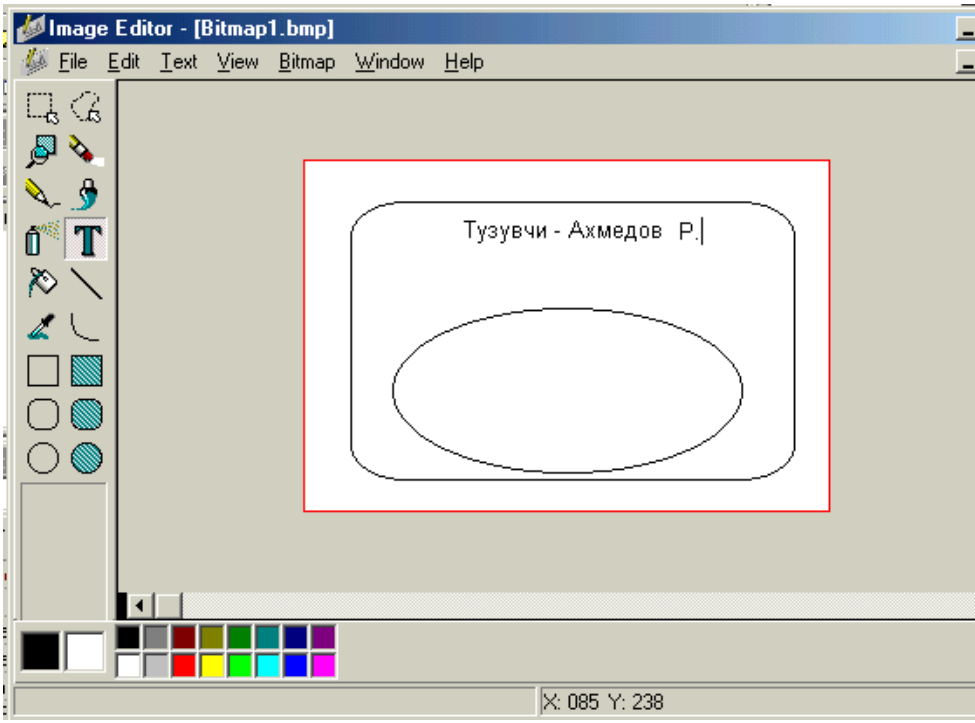
График тасвирни, яъни .bmp кенгайтмага эга бўлган файлни график муҳаррири ёрдамида тайёрлаймиз. Delphi системасини ишга туширишдан аввал тузиладиган иловани сақлаш учун ўзимизга папка ташкил қиламиз.

1.Олдин тузилган бирор бир иловани очамиз ёки янги илова ташкил этамиз.

2.Бош менюдан график муҳаррирни ишга туширамиз: Tols=>Image Editor. (Бу Delphi график мухарири оддий Paint график муҳарриридан унча катта фарқ қилмайди)

3.Delphi график мухарири Image Editor менюсидан File=>New=>Bitmap File(.bmp) буйруғи берилади. Натижада экранда расм парамертларини бериш учун мулоқат дарчаси пайдо бўлади. Мулоқат дарчасидан керакли параметрлар танланиб Ok тугмаси босилади. Тайёр мавжуд расм фойлларидан ҳам фойдалашиш мумкин.

4.График муҳарири ойнасидан ажратилган жойга ихтиёрий расм чизилиб, у сақланади. Масалан, айлана ва унга ташқи чизилган расм чизиб, ичига “Тузучи – Ахмелов Р.” сўзи ёзиб қўйилсин. Матнни ёзиш учун ускуналар панелининг “T” (Text) тугмачасидан фойдаланилади.



5.График файли сақланади ва ундан чиқилади.

6.System палитрасидан Timer компонентасининг тугмачани босиб формага олиб келиб қўйилади ва у Timer1 ном олади. Interval хоссасини 3000 га тенглашириб оламиз.

7.Additional палитрасидан Image компонентаси тугмачасини босиб формадан расм учун жой ажратилади ва кейин эса хоссалар бўлимидан Picture хоссаси танланиб, у ердан уч нуқтали тугмача босилади. Натижада экранда расмни аниқлаш ва жойлаш учун мулоқат дарчаси очилади. Мулоқат дарчасидан Load буйруғи берилиб, сақланган расм файлимиз танланади ва Ok тугмаси босилади. Расм тўлиқ формага жойлашиши учун Autosize хоссасига True қийматини ўрнатамиз.

8.Timer1 компонентини активлаштирамиз, яъни уни икки марта тез-тез босамиз ва кодларни ёзиш ойнасига қуйидаги қора ёзилган кодларни киритамиз.

```
procedure TForm1.Timer1Timer(Sender: TObject);
  begin
        Image1.Free;
        Timer1.Free;
  end;
```

Бу шуни билдирадики программа ишга тушгандан сўнг 3000 миллисекунддан ўтиши билан Image1 ва Timer1 компоненталари компьютер хотирасидан ва мос равишда экрандан ўчирилади.

9.Тузилган лойиҳа (проект) яъни Project1 ва Unit1 стандарт модул номларини мос номлар билан алмаштириб сақланади.

10.Янги ном билан сақланган проект, яъни илова F9 тугмачасини босиш билан ишга тушурилади.

Илова ишга туширилганда экранда юқоридаги 4 пунктдаги расм “заставка” кўринишида намоён бўлади.

Ташкил қилинган модулнинг тўлиқ кўринишини келтирамиз.

```
unit px1;

interface

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

type
  TForm1 = class(TForm)
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

 implementation
 {$R *.dfm}

procedure TForm1.Timer1Timer(Sender: TObject);
 begin
   Image1.Free;
   Timer1.Free;
 End;

end.
```

Яратилган “заставка”миз программа ишлаш давомида ўчиб-ёниб туриши учун қуйидаги OnTimer ходисасини қайта ишлаш кодини ёзишимиз керак бўлади.

```
procedure TForm1.Timer1Timer(Sender: TObject);
 begin
    If Image1.visible=true then Image1.hide
                    Else Image1.Show;
 End;
```

Delphi мультимедиа имкониятлари

Windows муҳитида фойдаланувчи дастурларининг кўпчилиги мультимедия дастурларидир. Бундай дастурлар видеоролик ва мультипликация кўриш, мусиқа ва нутқни эшитишга имкон беради. Мультимедияли дастурларга ўйинлар ўргатувчи дастурлар мисол бўла олади.

Delphi да мультимедияли дастурлар яратиш учун икки компонентадан фойдаланиш мумкин:

- Animate -оддий анимация яратиш учун (масалан файллардан нусха олишда фойдаланувчи кўрадиган анимация);
- MediaPlayer -мураккаб вазифаларни бажариш учун, видеороликларни кўриш, товушли анимация.

Animate Компонентаси

Animate компонентаси белгиси, Win32 қаторда жойлашган бўлиб, кадрлари AVI-файлда жойлашган содда анимация кўришга имкон беради.
AVI-файлдаги анимация товушли бўлиши мумкин бўлса ҳам Animate компонентаси фақат тасвирни акс эттиришга имкон беради.
Animate компонентаси формага оддий усулда кўшилади. Компонента формага қўшилгандан сўнг унинг хоссалаприни ўрнатиш лозим. Animate хоссалари жадвалда келтирилган:

Animate компонентаси хоссалари.

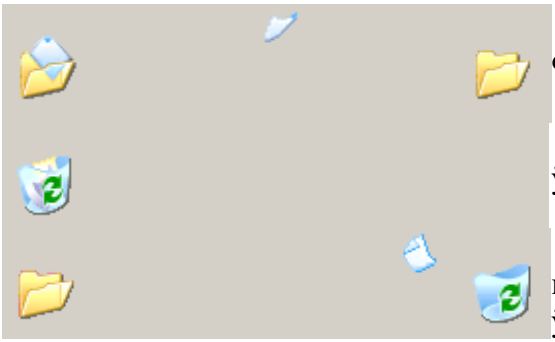
Хосса	Таърифи
Name	Компонента номи.
FileName	Анимация жойлашган AVI-файл номи
StartFrame	Анимацияни биринчи кадри номери
stopFrame	Анимация охирги кадри номери
Activate	Анимацияни акс эттириш жараенини активла белгиси
Color	Компонента фони ранги

Transparent	Анимация акс эттиришда шаффоф рангдан фой режими
Repetitions	Анимацияни қайтариш сони

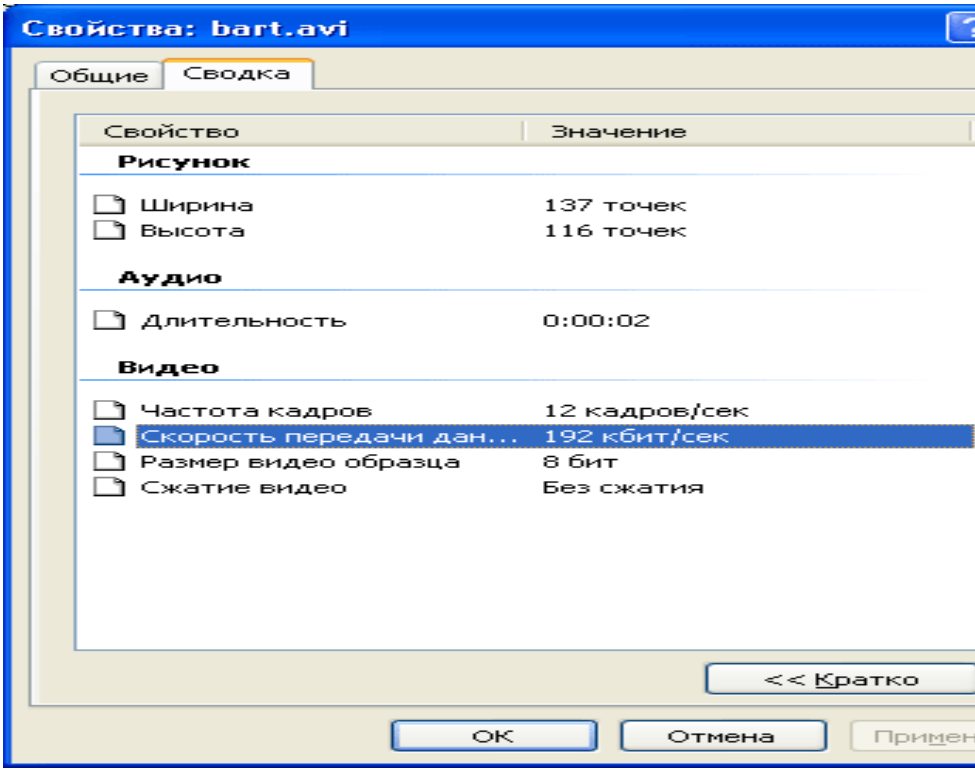
Агар FileName хоссасига товушли файл номи ёзилса, Delphi файлни очиш мумкин эмаслиги ҳақида маълумот чиқаради (Cannot open AVI). AVI-файлда анимация ва товуш ёки фақат анимация ёзилганлигини аниқлаш учун Windowsда керакли папкани очиб, AVI-файлни белгилаш ва контекстли менюдан Свойства командасини танлаш лозим. Натижада Свойства, ойнаси очилиб, Сводка қаторида файл ҳақида тўлиқ маълумот берилади.

Animate компонентаси дастурчига Windows стандарт анимациясидан фойдаланишга имкон беради. Анимация тури CommonAVI хоссаси қиймати билан белгиланади. Хосса қиймати номланган константалар орқали берилади. Қуйидаги жадвалда константалар қийматлари, анимация тури, жараён таърифи берилган.

comonAVI хоссаси қийматлари.

Қиймат	Анимация	Жараён
aviCopyFiles		Файлдан нусх олиш
AviDeleteFile		Файлни ўчириш
aviRecycleFile		Файли корзинага ўчириш

Қуйидаги дастур, Animate компонентасидан фойдаланишга мисол бўлади. Дастур формаси кўриниши расмда, Animatel компонентаси хоссалари қийматлари жадвалда берилган.



Сводка бўлимида AVI-файл ҳақида маълумот акс этади.

Анимация қуриш дастури формаси

Animate1 хоссалари қийматлари

Хосса	Киймат
FileName	D:\music\ms\COMMON\GRAPHICS\AVIS\SEARCH.AVI
Active	False
Transparent	True

Дастур ишга туширилгандан сўнг формага биринчи анимация кадри чиқарилади. Дастур анимацияни кўришнинг икки режимини таъминлайди:

- узлуксиз;
- кадрли.

Button1 тугмаси анимацияни куриш жараёнини инициализация қилиш ёки тухтатиб туриш учун ишлатилади. Анимацияни узлуксиз акс этиши Пуск тугмасининг Onclick ҳодисасини қайта ишлаш процедурасида Active хоссасига True кийматини бериш орқали инициализация қилинади. Бу процедура Button1 тугмасидаги Пуск сўзини Стоп сўзига алмаштиради. Анимацияни кўриш режими RadioButton1 ва RadioButton2 тугмалари орқали танланади.
Анимация кўриш дастури матни

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ComCtrls, ExtCtrls;
type
```

```
TForm1 = class(TForm)
Animate1: TAnimate;
Button1: TButton;
 Button2: TButton;
Button3: TButton;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
private
{ Private declarations } public
{ Public declarations }
 end;
var
Form1: TForm1;
CFrame: integer;
implementation {$R *.DFM}
procedure TForm1.Button2Click(Sender: TObject);
begin
if CFrame = 1 then Button2.Enabled := True;
if CFrame < Animate1.FrameCount then begin
CFrame := CFrame + 1;
Animate1.StartFrame := CFrame;
Animate1.StopFrame := CFrame;
Animate1.Active := True;
if CFrame = Animate1.FrameCount
then Button2.Enabled:=False;
end;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
if CFrame = Animate1.FrameCount
```

```
then Button2.Enabled := True;
if CFrame > 1 then begin
CFrame := CFrame - 1;
Animate1.StartFrame := CFrame;
Animate1.StopFrame := CFrame;
Animate1.Active := True;
if CFrame = 1
then Form1.Button3.Enabled := False;
end;
end;
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
Button1.Enabled:=True;
Button3.Enabled:=False ;
Button2.Enabled:=False;
end;
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
Button2.Enabled:=True;
Button3.Enabled:=False;
Button1.Enabled:=False; end;
procedure TForm1.Button1Click(Sender: TObject);
begin
if Animate1.Active = False
then begin
Animate1.StartFrame:=1;
Animate1.StopFrame:=Animate1.FrameCount;
Animate1.Active:=True;
Button1.caption:='Тухташ';
RadioButton2.Enabled:=False;
end
else
begin
Animate1.Active:=False;
Button1.caption:='Бошлаш';
```

```
RadioButton2.Enabled:=True;
end;
end;
end.
```

График режим (TPrinter объекти)

График режимда босмага чиқариш учун форманинг Print усулини қўллаш мумкин. Лекин махсус Printer объектидан (TPrinter синфига тегишли) фойдаланиш қулайроқдир. Бу объектдан фойдаланиш учун Printers модулини дастурга қўшиш яъни uses бўлимида эълон килиш лозим. Printer объекти Canvas хоссасидан фойдаланишга имкон беради.

Printer хоссалари:

Aborted – мантиқий тур; фойдаланувчи Abort усули билан босмани тўхтатганини курсатади.

Canvas – графика чиқариш сохаси.

Fonts – шрифтлар рўйхати.

Handle - Windows API чақирилганда фойдаланилади.

Orientation – сахифа йўналиши, вертикал еки горизонтал.

PageWidth, PageHeight, PageNumber – сахифа кенглиги, баландлиги ва номери.

Printers принтерлар

PrinterIndex жорий принтер номери. Ўрнатилган принтер номери -1.

Printing – мантиқий тур, босмага чиқариш бошланганлигини кўрсатади (BeginDoc усули билан).

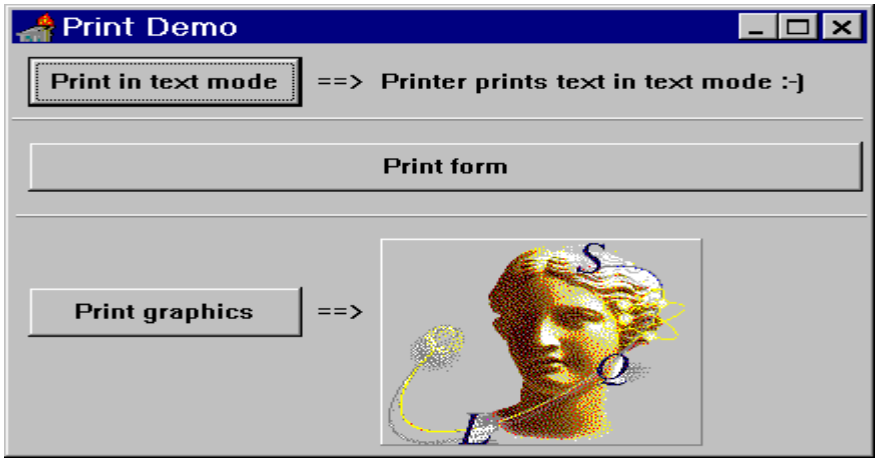
Title - Print Manager учун сарлавха.

Printer усуллари:

Abort –босмани тўхтатиш
BeginDoc – сохага чизишдан олдин чақирилади.
EndDoc – сохага чизиб бўлингандан сўнг чақирилади. Шундан сўнг, принтер ишга тушади.
NewPage –янги сатрга ўтиш.
Шундай қилиб график маълумотни босмага чиқариш қуйидаги тартибда амалга оширилади:

- BeginDoc усули чақирилади
- Сохага (Canvas) чизилади
- Агар бир неча сахифага ажратиш зарур булса NewPage усули чақирилади
- EndDoc усули билан график маълумот принтерга юборилади

Қуйидаги мисолда босмага чиқаришнинг учта усули кўрсатилган.
Дастур формаси кўриниши



Дастур матни

```
 unit Pri_form;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, Printers;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Bevel1: TBevel;
    Button2: TButton;
    Bevel2: TBevel;
    Button3: TButton;
    Label3: TLabel;
    Panel1: TPanel;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  To_Prn : TextFile;
begin
  Button1.Enabled:=False;
  AssignPrn(To_Prn);
  Rewrite(To_Prn);
  Writeln(To_Prn, Label2.Caption);
  CloseFile(To_Prn);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Button2.Enabled:=False;
  Form1.Print;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Button3.Enabled:=False;
  with Printer do begin
    BeginDoc;
    Canvas.Draw(0,0,Image1.Picture.Bitmap);
    NewPage;
    Canvas.StretchDraw(Rect(0,0,300,300),
Image1.Picture.Bitmap);
    EndDoc;
  end;
```

```
end;

end.
```

С а в о л л а р

1.Соҳа ўлчовлари қайси хоссалар ёрдамида аниқланади?
2.Матн шрифти қайси хоссалар орқали аниқланади?
3.График объект юзасига матн чиқариш учун қайси усул қўлланилади ва бу усулни чақириш инструкцияси кўринишини келтиринг.
4.Тўғри чизиқ қайси усул орқали амалга оширилади ва бу усулни чақириш инструкцияси кўринишини келтиринг.
5.Айлана ёки эллипс чизиш учун қайси усул чақирилади ва бу усулни чақириш инструкцияси кўринишини келтиринг.
6.Ёйни чизиш учун қайси усул қўлланилади ва у қандай умумий кўринишга эга.
7.Тўртбурчак чизиш учун қайси усул қўлланилади ва у қандай умумий кўринишга эга.
8.Кўпбурчак чизиш учун қайси усул қўлланилади ва у қандай умумий кўринишга эга.
9.Эллипс ёки айлана секторини чизиш учун қайси усул қўлланилади ва у қандай умумий кўринишга эга.
10.Image компонентаси формага нимани жойлаштиришда қўлланилади?
11.Shape компонентаси формага нималарни жойлаштиришда қўлланилади?
12.Мультипликация дейилганда нима тушунилади?