O’ZBEKISTON RESPUBLIKASI
OLIY VA O’RTA MAXSUS TA'LIM VAZIRLIGI

NAMANGAN MUHANDISLIK-PEDAGOGIKA INSTITUTI

Kasb ta’limi fakulteti

Kasb ta’limi (Informatika va IT) kafedrasi

OB'EKTGA YO'NALTIRILGAN DASTURLASH TILLARI FANIDAN

MA’RUZALAR MATNI

Namangan-2012 y

1-Ma`ruza. Kirish. Dasturlarni qayta ishlash bosqichlari. Delphi va uning paydo bo'lish bosqichlari. Algoritmlar va dasturlar. Object Pascal dasturlash tili.

O'quv modul birliklari:

1. Kirish.
2. Dasturlarni qayta ishlash bosqichlari. Delphi va uning paydo bo'lish bosqichlari.
3. Algoritmlar va dasturlar. Object Pascal dasturlash tili.

Aniqlashtirilgan o'quv maqsadlari:

Talaba ushbu mavzuni to'la o'zlashtirgandan so'ng:

1. Delphi va uning paydo bo'lish bosqichlarini biladi.
2. Algoritmlar va dasturlarni Object Pascal dasturlash tili orqali o'zlashtiradi.

Kirish

Insoniyat tarixining ko`p asrlik tajribasi ezgu g`oyalaridan va sog`lom mafkuradan mahrum biron-bir jamiyatning uzoqqa bora olmasligini ko`rsatdi. SHu bois, mustaqillik tufayli mamlakatimiz o`z oldiga ozod va obod Vatan, erkin va farovon hayot barpo etish, rivojlangan mamlakatlar qatoridan o`rin olish, demokratik jamiyat qurish kabi ezgu maqsadlarni qo`ydi.

Bu esa kelajagimizni yaqqol tasavvur etish, jamiyatimizning ijtimoiy-ma`naviy poydevorini mustahkamlash ehtiyojijni tug`diradi. Demak, galdagi eng asosiy vazifa: yosh avlodni Vatan ravnaqi, yurt tinchligi, xalq faravonligi kabi olijanob tuyg`ular ruhida tarbiyalash, yuksak fazilatga ega, ezgu g`oyalar bilan qurollangan komil insonlarni voyaga etkazish, jahon andozalariga mos, kuchli bilimli, raqobatbardosh kadrlar tayyorlashdir.

«Jahon sivilizasiyasiga daxldor bo`lgan eng zamonaviy ilmlarni egallamay turib, mamalakat taraqqiyotini ta`minlash qiyin», - degan edilar prezidentimiz I. Karimov. O`zbekistonning iqtisodiy va ijtimoiy sohalarda yuqori natijalarga erishishi, jahon iqtisodiy tizimida to`laqonli sheriklik o`rnini egallay borishi, inson faoliyatining barcha jabhalarida zamonaviy axborot texnologiyalaridan yuqori darajada foydalanishning ko`lamlari qanday bo`lishiga hamda bu texnologiyalar ijtimoiy mehnat samaradorligini oshishida qanday rol o`ynashiga bog`liq.

Prezidentimiz Islom Karimovning ko`p yillik izlanishlari, asarlaridagi fikr-mulohazalariga tayanib yaratilgan «Milliy istiqlol g`oyasi: asosiy tushuncha va tamoyillar» nomli risolada ta`lim-tarbiya jarayonining ixtiyoriy bosqichida amal qilish lozim bo`lgan quyidagi mezon va talablar keltirilgan:

- o`quv mashg`ulotlarini olib borishda talabalarning yoshi, tafakkuri, dunyoqarashi va qiziqishlarini hisobga olish;
- ta`lim-tarbiyaning ilg`or, ta`sirchan vositalaridan, zamonaviy o`qitish texnologiyasi imkoniyatlaridan keng foydalanish;

- ayrim tushunchalarni haddan ziyod soddalashtirish, ta`limning eskicha uslub va tamoyillarini qo`llash natijasida fanning qadrsizlanishiga yo`l qo`ymaslik;
- ta`lim jarayonida tazyiq o`tkazmasdan ma`rifiy asosda ish tutish, yoshlarning mustaqil va erkin fikrlash, bahs-munozara yuritish ko`nikmalarini oshirishga e`tibor qaratish;
- o`qituvchi va tinglovchilar orasida o`zaro hamfikrlik va hamkorlik muhitini shakllantirish, mavzuning tushuncha va tamoyillarini sharhlashda hayotiy misollar, bugungi dunyoda ro`y berayotgan voqealar tahlilidan, matbuot materiallaridan keng foydalanish;
- yoshlarda g`oyalar, o`z ma`no-mohiyatiga ko`ra bunyodkor yoki vayronkor bo`lishi haqidagi hayotiy va haqqoniy tasavvurlarni shakllantirish;
- milliy istiqlol g`oyasining insonparvarlik mohiyatini ko`rsatish asosida mustaqillik biz uchun eng oliy qadriyat, uni asrab-avaylash esa har birimizning muqaddas burchimiz ekanini talabalarning qalbi va ongiga singdirish.

Yuqorida aytilgan mezon va talablarga rioya qilgan xolda Respublikamizda, zamonaviy hisoblash texnikasi vositalaridan samarali foydalanishni uddalay oladigan, zamonaviy kompyuterlardan amaliy ish faoliyatida keng foydalana oladigan etuk kadrlar tayyorlash dolzarb vazifalardan hisoblanadi. Shuning uchun, kadrlar tayyorlash milliy dasturining ikkinchi bosqichida yuqori malaka, raqobatbardosh kadrlar tayyorlash uchun sifatli, jahon andozalariga mos darsliklar, o`quv qo`llanmalari va ma`ruza matnlarini tayyorlab, chop ettirish masalasiga juda katta e`tibor berilgan.

Dasturni qayta ishlash bosqichlari.

Dasturlash – bu buyruqlar ketma-ketligini kiritish bo'lib, uni yaratishda quyidagi bosqichlar bosib o'tiladi:

- Qo'yilgan masalani dasturlash mumkinligini tekshirish;
- Qo'yilgan masalaning algoritmini tanlash yoki qayta ishlash;
- Buyruqlarni yozish;
- Dastur xatoliklarini tekshirish;
- Testdan o'tkazish.

Qo'yilgan masalani dasturlash mumkinligini tekshirish – kerakli bosqichlardan biri bo'lib, masalaning qo'yilishi sinchkovlik bilan tekshiriladi va natija olish uchun ma'lum bir formaga tushiriladi. Masalan, masalaning qo'yilishi bo'yicha kvadratli tenglamani umumiy ko'rinishi quyidagicha yoziladi: $ax^2+bx+c=0$ va uni quyidagi formalarga bo'lish mumkin:

- (a,b,c) noma'lumlik darajasining koeffisiyentlari dasturlashda boshlang'ich shart hisoblanadi;
- noma'lumlar albatta dialog rejimida, klaviaturadan dastur ishlayotgan vaqtda kiritilishi shart;
- chiqariladigan natijalar ma'noga ega bo'lishi kerak;

- agarda natija ma’noga ega bo’lmasa, ogohlantiruvchi so’z chiqishi kerak.

Tuzilgan dastur Windowsda ishlash uchun mo’ljallanadi, tuzilayotgan dasturga ixtiyoriy ravishda dialog oynalarini qo’shish mumkin.

Qo’yilgan masalaning algoritmini tanlash yoki qayta ishlash bosqichida natija olish uchun kerak bo’ladigan muhit tekshiriladi. Agarda masala turli usullar bilan yechiladigan bo’lsa, dasturchi eng qulay, ya’ni tez va aniq ishlaydigan usulni tanlaydi.

Bo’yruqlarni yozish – dasturga qo’yilgan talablar tekshirilganidan va algoritmi tuzilganidan so’ng, u tanlangan dasturlash tillaridan birida yoziladi.

Dastur xatoliklarini tekshirish bosqichida yaratilgan dastur ichidagi xatoliklar izlanadi. Dasturdagi xatoliklar ikki qismga bo’linadi: ***sintaktik*** (matn ichidagi xatoliklar) va ***algoritmik***. **Sintaktik** xatoliklarni (biron-bir belgilarning almashganligi, tushirib qoldirilganligi va hokazolar) oson topiladi. **Algoritm** xatoliklarini topish mushkulroq kechadi. Ma’lumotlarni kiritish bir-ikki bor takrorlanganda dastur to’g’ri ishlasa, xatoliklarini tekshirish bo’limi yakunlangan hisoblanadi.

Testdan o’tkazish bosqichi o’ta muhim bo’lib, yaratilgan dasturdan boshqalar ham foydalanishi hisobga olinadi. Bu bosqichda eng ko’p qancha ma`lumotni ko’tara olishi va unda kiritilishi mumkin bo’lgan noto’g’ri ma’lumotlar tekshiriladi.

Algoritmlar va dasturlar

Dastur yaratilishining birinchi bosqichida qo’yilgan masalani kompyuterga tushiriladi va undagi muammolar ko’rib chiqiladi. Masalan, kvadrat tenglamani ildizini hisoblash dasturi. Bunda kvadrat tenglama ildizini topish uchun kerakli ma`lumotlar kiritilishi kerak. Natijada «kvadrat tenglamani ildizi yoki tenglamaning ildizi yo’q» degan ogohlantirish chiqishi lozim.

Kvadrat tenglamaning ildizlari formula orqali hisoblanadi. Dastlab formuladan diskriminant natijasini topish kerak. So’ngra agar natija nolga teng yoki katta bo’lsa, u holda formula bo’yicha ildiz hisoblanadi.

Qo’yilgan masalani yechish uchun algoritmdan foydalaniladi.

Algoritm deb, qo’yilgan masalani yechishga qaratilgan, hisoblash jarayonini ifodalovchi, boshlang’ich ma’lumotlardan izlanayotgan natijani keltirib chiqarishga qaratilgan jarayonga aytiladi.

Shuni aniqlash kerakki, muayyan ketma-ketlik quyidagi 3 ta xossaga ega bo’lsagina algoritm hisoblanadi:

Bir qiymatlilik – masalani yechish uchun bajariladigan amallar ketma-ketligi va bajarish yo’li yagona bo’lishi;

Umumiylik – algoritm o’zgaruvchilarining turli xil qiymatlari uchun to’g’ri natija olish imkoniyatiga ega bo’lishi;

Natijaviylik – algoritm bajarilishi jarayonida aniq natija olinishi kerak.

Quyida kvadrat tenglamaning ildizini topish algoritmining matematik ko'rinishiga misol keltirilgan:

Kiritiladigan ma'lumotlar – tenglama koeffisiyentlari: *a, b, s.*

Topiladigan natija – *x1* va *x2* tenglama ildizlari.

Diskriminatni hisoblash formulasi: $d = b^2 - 4ac$

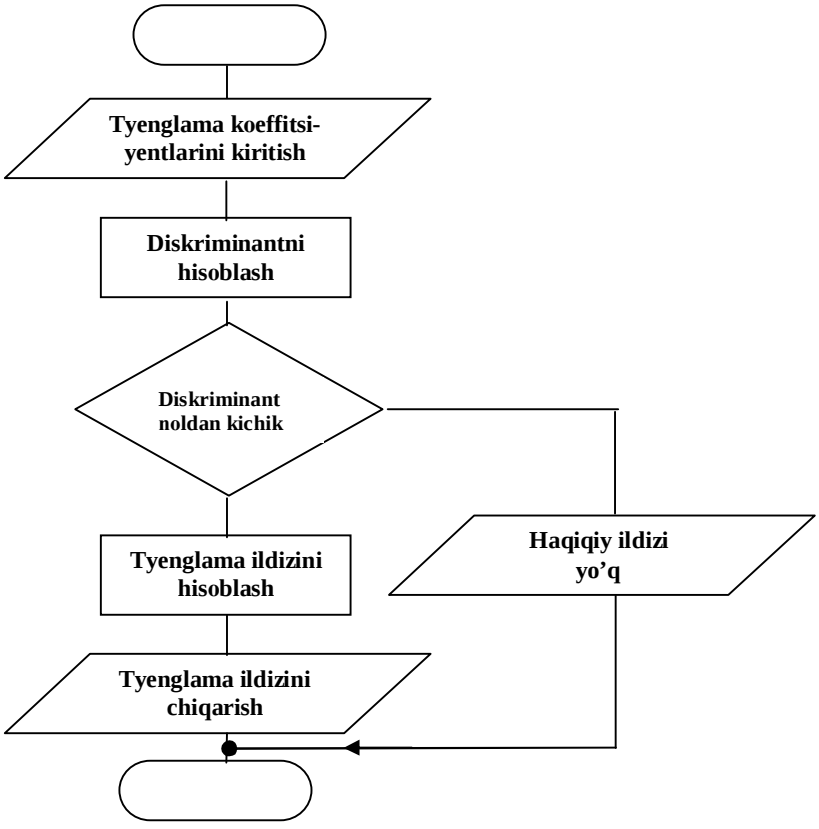
Agar diskriminant natijasi nolga teng yoki katta bo'lsa, u holda quyidagi formula bilan tenglama ildizlari topiladi: $x1 = \frac{-b - \sqrt{d}}{2a}$; $x2 = \frac{-b + \sqrt{d}}{2a}$

Agar diskriminant natijasi noldan kichik bo'lsa, bu tenglamaning xaqiqiiy ildizi yo'qligini bildiradi.

Masalani yechish algoritmi blok-sxema ko'rinishida bo'ladi. Blok-sxemada masalaning mantiqiy va turli qismi standart shakllar bilan yoziladi. Blok-sxemaning asosiy elementlari boshlash/tamom,kiritish/chiqarish, qayta ishlash va tanlash(6-rasm).

boshlash/tamom kiritish/chiqarish qayta ishlash tanlash

6-Rasm. Algoritmning blok-sxemasi uchun asosiy shakllar

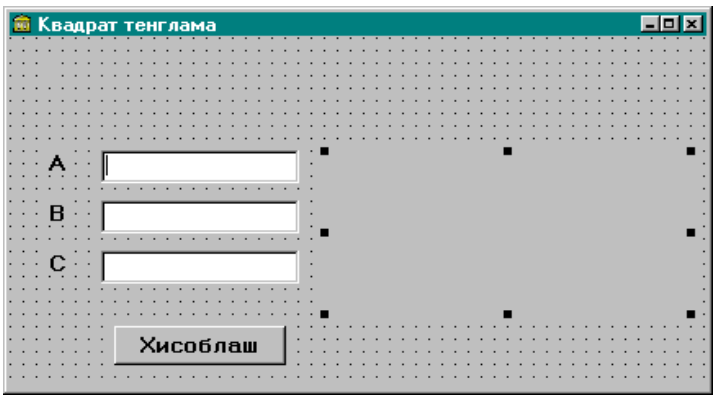


7-Rasm. Kvadrat tenglamani hisoblash algoritmining blok-sxemasi

Misol uchun, kvadrat tenglamaning ildizini topish algoritmi -rasmdagi blok-sxema ko’rinishida bo’lishi mumkin. Bu blok-sxema ko’rinishidagi algoritm dasturchiga bajariladigan barcha jarayonni aniq kuzatish va tahlil qilish uchun xizmat qiladi.

Algoritmning blok-sxemasi qurilganidan so’ng, tanlangan dasturlash tilida uning dasturini tuzish mumkin.

Kvadrat tenglama algoritmining dasturi quyidagi dastur matnida berilgan bo’lib, dialogli oynasi esa 8-rasmda ko’rsatilgan.



8-rasm. Kvadrat tenglamaning dialogli oynasi

Dastur matni

```
unit Kvadrat;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

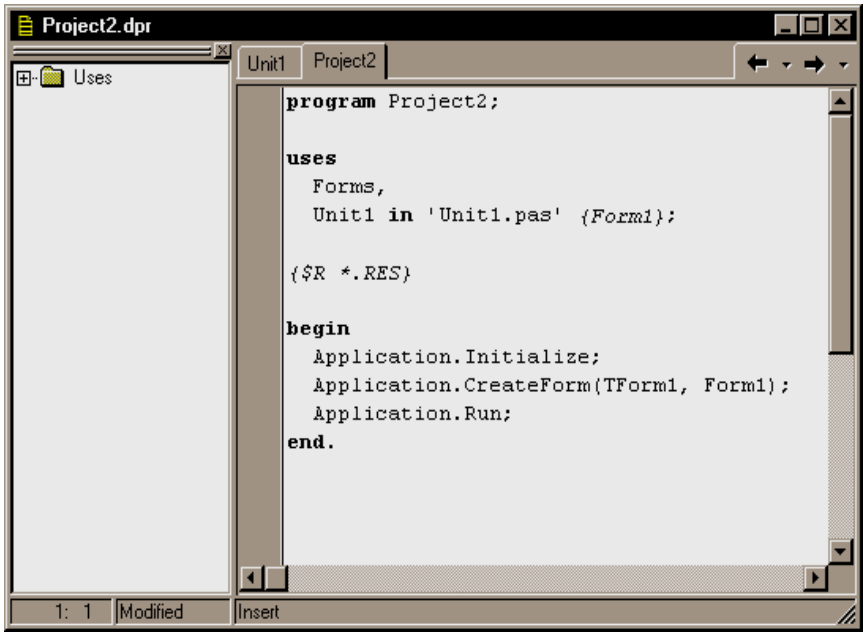
```
 var
  Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
Var
   a, b, c: Real;  { Tenglama koeffisiyentlari }
   d: Real;      { Diskriminant }
   x1, x2: Real;  { Tenglama ildizlari}
begin
   { Kerakli ma'lumotlarni kiritish }
   a := StrToFloat(Edit1.Text);
   b := StrToFloat(Edit2.Text);
   c := StrToFloat(Edit3.Text);
   { Diskriminantni xisoblash }
   d := b * b – 4 * a * c;
   If d < 0 Then
   Begin
      Label5.Caption := 'Diskriminant noldan kichik' + #13 +
      'Tenglamaning ildizi yuk.'
   End
   Else
   Begin
      { Ildizlarni xisoblash }
      x1 := (-b - Sqrt(d) ) / (2 * a);
      x2 := (-b + Sqrt(d) ) / (2 * a);
      { x1, x2 natijani chop etish }
      Label5.Caption := 'Tenglama ildizlari'
          + #13 + 'x1= ' + FloatToStr(x1)
          + #13 + 'x2= ' + FloatToStr(x2);
   End;
End;

procedure TForm1.FormActivate(Sender: TObject);
begin
   Label1.Caption:='Tenglama koeffisiyentlarini kiriting'
   +#13+'va Xisoblash tugmasini bosing';
end;
end.
```

Dastur matnidagi TForm1.Button1Click(Sender: TObject) prosedurasi tenglama yechimini hisoblaydi. Tenglamani yechish uchun Hisoblash tugmasi bosiladi.

Konsolli ilovalar

Delphida dasturchilar uchun Read, Readln klaviaturadan kiritish va Write, Writeln oynaga chiqarish operatorlaridan foydalanish imkoniyati ham yaratilgan. Bular konsolli ilovalar deb yuritiladi.

Konsolli ilovalar quyidagi ko'rinishda yaratiladi: Delphi ishga yuklanganidan so'ng, oynada yangi **Form1** formasi bo'lmasa. **File** menyusidan **New Application** (Yangi ilova) buyrug'i tanlanadi. Yangi forma xosil bo'lgandan so'ng, **Project** (Proyekt) menyusidan **View Source** (Ko'rish) tanlanadi. Natijada *Project2.dpr* deb nomlangan (9-rasm) oyna xosil bo'ladi.



9-rasm

Eslatma: *Konsolli ilovalarda kiril harflari o'rniga tushunib bo'lmas belgilar chiqib qoladi, sababi konsolli ilovalar ASCII kodida chop etiladi. Windowsda esa ANSI kodi ishlatiladi. Shuning uchun konsolli ilovalarni lotin harfida yozish talab qilinadi. Misol uchun, Writeln('A sonni kiriting').*

Quyidagi dastur matnida berilgan kilogrammni necha funt ekanligini hisoblovchi dastur ko'rsatilgan. Unda biror buyumning og'irligi foydalanuvchi tomonidan kilogrammda kiritiladi. Natijada esa uning qancha funt ekanligi chop etiladi.

Dastur matni

```
{$APPTYPE CONSOLE}
Program Project2;
Var
  k, f: Real;
Begin
  Writeln('Buyum og'irlugini kilogrammda kiriting');
  Writeln('va  <Enter> tugmasini bosing');
  Write('→');
```

```
  Readln(k);
  F := k * 0.4095;
  Writeln(k: 10: 4,'  kilogrammq', f: 10: 4,'  funt');
  Readln;
End.
```

Yuqoridagi dasturda *{$APPTYPE CONSOLE}* qatori mavjud bo'lib, u izoh ko'rinishida yozilgan. Lekin u, dasturning konsolli ilova ekanligini bildiradi. Bunday dasturni tuzishda albatta *{$APPTYPE CONSOLE}* qatori yozilishi shart.

Dasturni ishga tushirish uchun **Run** menyusidan **Run** buyrug'i tanlanadi yoki **F9** tugmachai bosiladi.

2-Ma`ruza. Delphini o'rnatish va ishlatish. Delphi ishchi oynasining asosiy elementlari. Delphida dastlabki amallar va proektlar. Dasturni ishga yuklash va dasturda yuz beradigan xatoliklar. (2 soat)

O'quv modul birliklari:

1. Delphini o'rnatish va ishlatish. Delphi ishchi oynasining asosiy elementlari.
2. Dastur matnining umumiy ko'rinishi. Dasturni ishga yuklash.
3. Dastur bajarilayotganda yuz beradigan xatoliklar.

Aniqlashtirilgan o'quv maqsadlari:

Talaba ushbu mavzuni to'la o'zlashtirgandan so'ng:

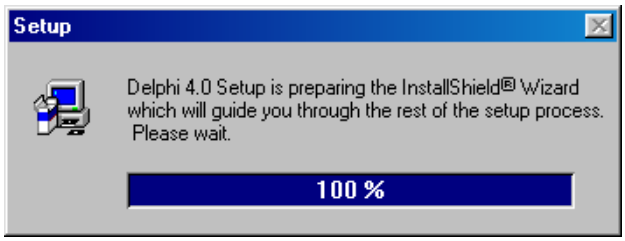
1. Delphini mustaqil o'rnata oladilar.
2. Dasturda kichik kodlarni yarata oladi.
3. Dasturlashdagi yuz beradigan xatoliklarni tushunadi va ularni yo'qotish yo'llarini biladi.

Delphini o`rnatish va ishlatish

Odatda Delphi paketini o'rnatish CD-ROM qurilmasi yordamida amalga oshiriladi. Kampak diskda barcha o'rnatuvchi initsializatsiya dasturlar va kerakli fayllar (Delphi Setup launeher) joylashgan. CD - diskovodga o'rnatuvchi diskni solishimiz bilan o'rnatiluvchi dastur avtomatik tarzda ishga tushadi.

Eslatma: Delphini o'rnatish vaqtida barcha aktiv dasturlar ishini yakunlash kerak.

O'rnatuvchi dastur ishga tushishi natijasida kompyuter oynasida **Delphi Setup launeher** (Delphini o'rnatishni boshlash) oynasi xosil bo'lib, bu oynadan turli ma'lumotlar va dasturni o'rnatish uchun tugmachalar joylangan. Dasturni o'rnatish uchun sichqoncha ko'rsatkichini Delphi satriga olib kelib chap tugmachani bosish kerak. Natijada Delphini o'rnatuvchi dastur ishga tushadi va oynada **Setup** (o'rnatish) oynasi xosil bo'ladi (1-rasm).



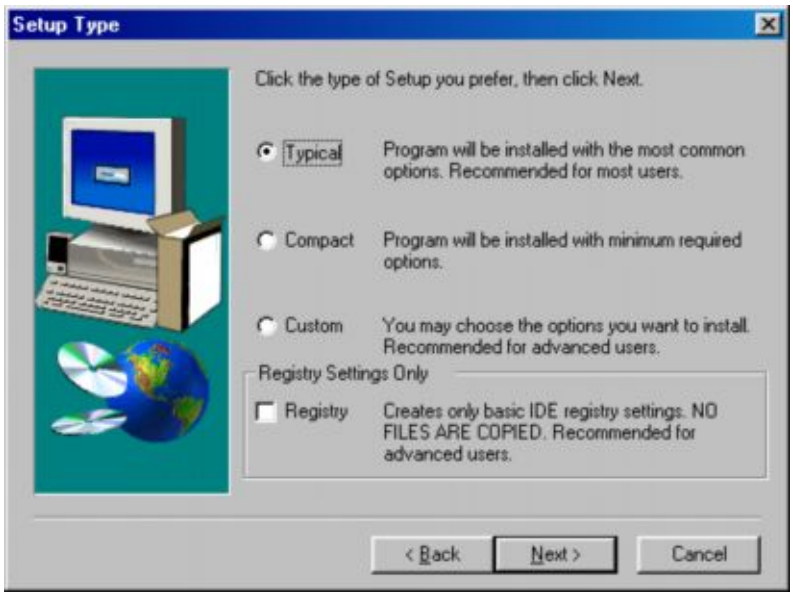
1-rasm. Delphini o’rnatish jarayonini boshlash.

Bu oynada Delphini o’rnatish uchun tayyorgarlikni bajarilishini foiz hisobida ko’rinishi chiqadi. Tayyorgarlik oxiriga yetganidan so’ng oynada **Welcome** (boshlanish) muloqot oynasi xosil bo’ladi (2-rasm).

2-rasm. Welcome dialogli oyna.

Undagi **Next**(keyingi) tugmasi bosilsa **Password Dialog** oynasi xosil bo’ladi. Bu oynada **Serial Number** (seriya raqami) va **Authorization key** (komplekt kodi)lar kiritiladi. **Next** tugmasi bosilsa **Software Lieense Agreement** (Litsyenziya kelishuvi) oynasi hosil bo’ladi (agar seriya raqami va komplekt kodi to’g’ri keltirilsa Delphini o’rnatish dastur tomonidan to’xtatiladi). Undagi matnni o’qib **Yes**(ha) tugmasi bosiladi.

O’rnatish jarayonida quyidagi oyna hosil bo’ladi (3-rasm):



3-rasm. Setup Type dialogli oyna.

Delphi- **Setup Type** (o'rnatish varianti) oynasi bo'lib, undagi keltirilgan variantlardan birini tanlash va **Next** tugmasini bosish kerak. Variantlar: **Typical** (odatdagi), **Compact** yoki **Custom** (tanlash).

Next, bosilgandan keyin **Select Component Directories** (komponentlar uchun katalog tanlash) dialog oynasi hosil bo'ladi (Custom varianti tanlanmasi). Bu oynada **Next** tugmasi bosilgandan so'ng, **Select Program Folder** (dastur papkasini ko'rsatish) oynasidan kerakli papka tanlanib, **Next** tugmasi bosiladi. Oynadagi **Start Copying Files** (fayllarni nushalashni boshlash) oynadan **Install** tugmasi bosiladi. Fayllar ko'chirib bo'lingandan so'ng S**etup Complete** oynasi hosil bo'ladi (4-rasm) va **Finish** (yakunlash) tanlanadi.

4-rasm. Setup Complete dialogli oyna.

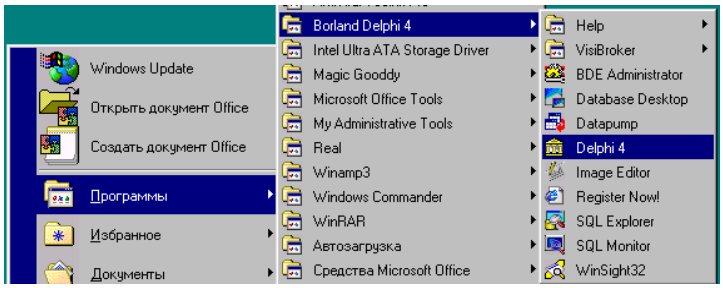
Shu bilan Delphini o'rnatish yakunlanadi va Delphi ishchi holatga tayyor holga keladi.

Delphini ishga tushirish

Asosan **Delphini** ikki usulda ishga tushirish mumkin:

“Пуск” (Start) tugmachasi bosiladi, **“Программы”** satri tanlanadi va **Borland Delphi4** satridan **Delphi4** oynasiga kirib, sirtiga sichqonchani chap tugmasini bosish bilan (5-rasm);

Ishchi stolga o’rnatilgan **Delphi4** yorlig’ini sirtiga sichqoncha ko’rsatkichini o’rnatib, chap tugmasini ikki marta bosish bilan (Yorliqni foydaluvchini o’zi yaratib olishi kerak).

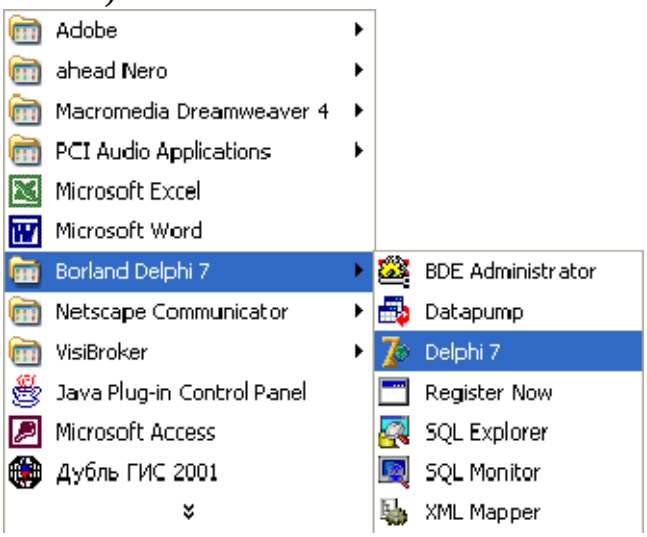


5-rasm. Windows ning bosh menyusidan Delphini yuklash.

	1. Delphini yana qaysi yo`l bilan ishga tushirish mumkin? 2. Delphi muxitida oynalardan biri yuq bo`lsa siz qanday qilib anashu oynani aktivlashtirasiz?

Ishni boshlash

Delphi oddiy usul bilan ishga tushiriladi, ya’ni, Windows da Start (Пуск) menyusidan Programs (Программы) bo’limidagi Borland Delphi 7 satridan Delphi 7 buyrug’i tanlanadi (B6-rasm).

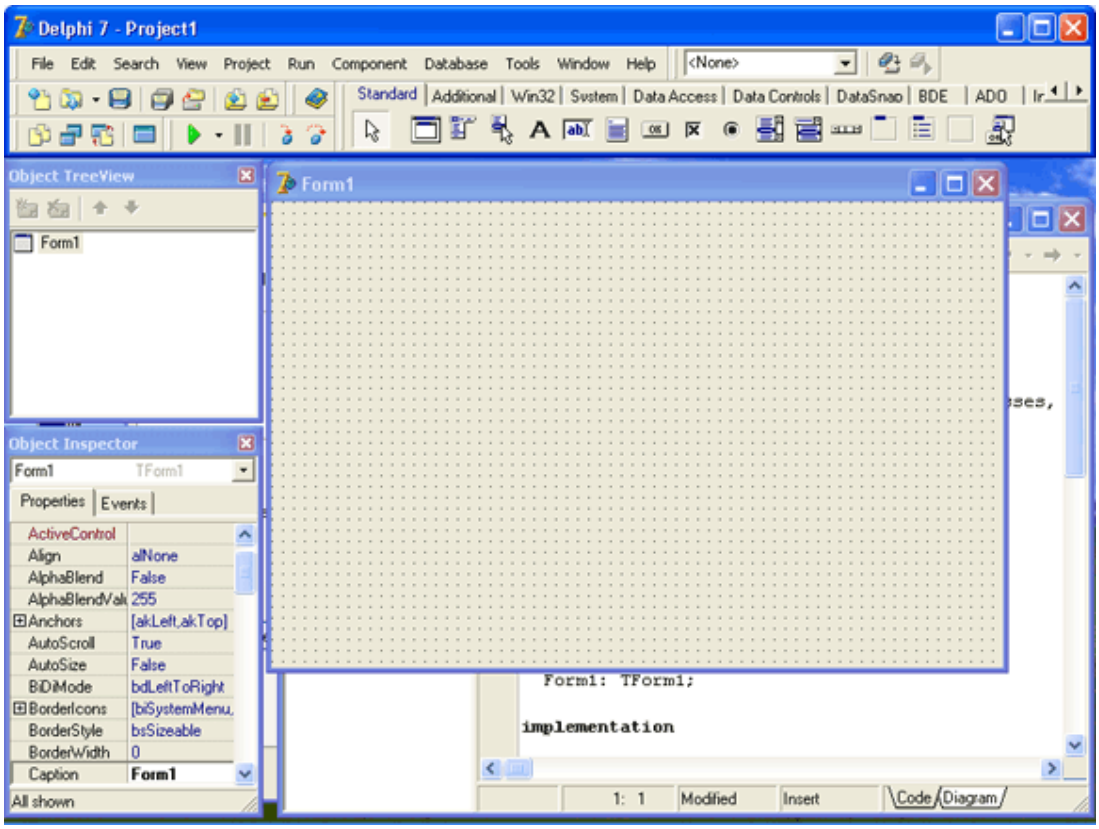


B6-rasm. Delphi ni ishga tushirish

Delphi ning ishchi muhiti beshta oynadan iborat (B7-rasm):

- Bosh oyna — **Delphi 7;**
- Bosh ishchi forma oynasi — **Form 1;**
- Ob’yekt hususiyatlarini taxrirlash oynasi — **Object Inspector;**
- Ob’yektlar ro’yxatini ko’rish oynasi — **Object TreeView;**

- Kodlarni taxrirlash oynasi — **Unitl.pas.**



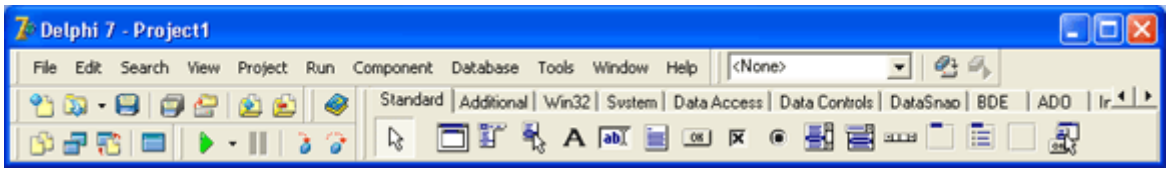
B7-rasm. Delphi ishchi muxiti

Bosh oynada (B8-rasm) buyruqlar menyusi, uskunalar paneli va komponentlar palitrasi joylashgan

Bosh ishchi forma (**Form1**) yaratiluvchi ***ilovaning*** bosh oynasi hisoblanadi.

Dasturlash ta’minoti ikkiga bo’linadi: sistemali va amaliy.

Sistemali dasturlash ta’minotini operatsion sistemani tashkil etadi. Qolgan dasturlar amaliy dasturlash ta’minoti hisoblanadi va qisqacha ilova deb ataladi.

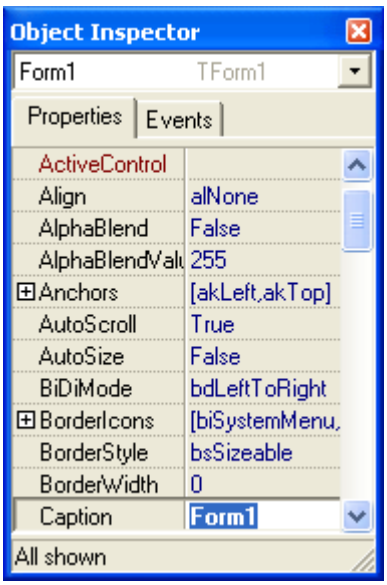


B8-rasm. Bosh oyna

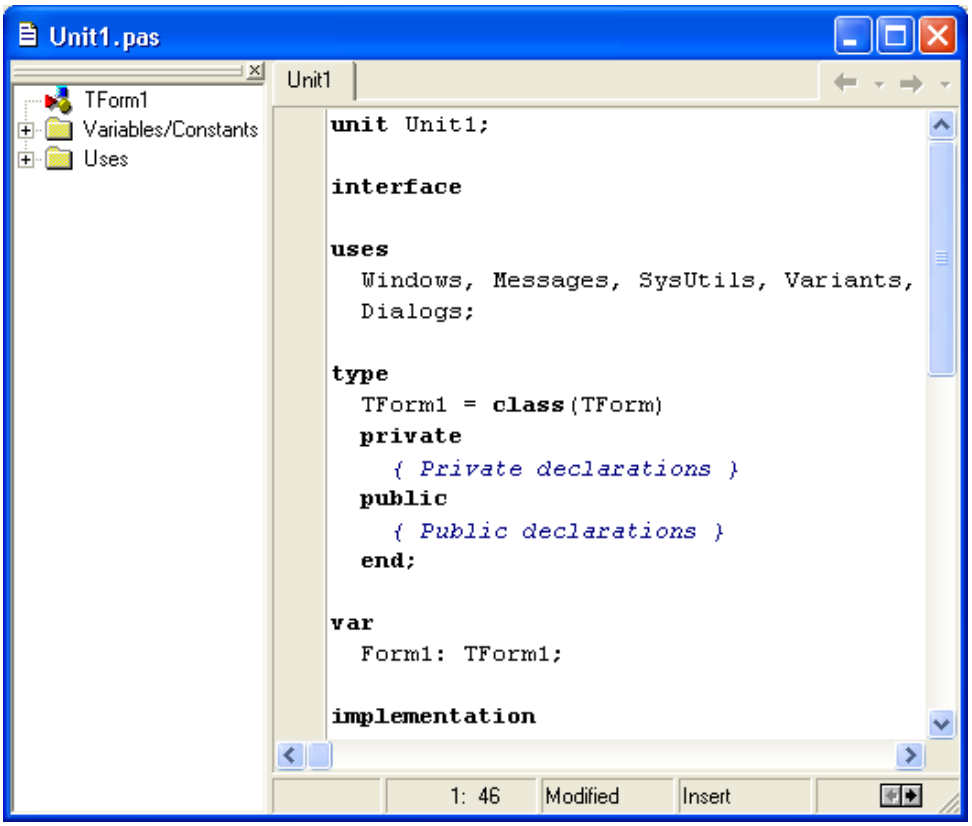
Object Inspector oynasi (B9-rasm) - ob’yektlarning xususiyatlarining qiymatlarini taxrirlash uchun mo’ljallangan.

Vizual loyihalash atamalarida *ob’yektlar* - bu muloqot oynalari va boshqaruv elementlaridir (kiritish va chiqarish maydonlari, buyruq tugmalari va boshqalar).

Ob’yekt xususiyati - ob’yektning ko’rinishi va joylashishini aniqlovchi tavsiflardir. Masalan, Width va Height xususiyatlari formaning o’lchamlari (balandligi va kengligi)ni, Top va Left xususiyatlari formaning ekranda joylashgan o’rnini, caption esa sarlavha matnini beradi.



B9-rasm. Object Inspector oynasi

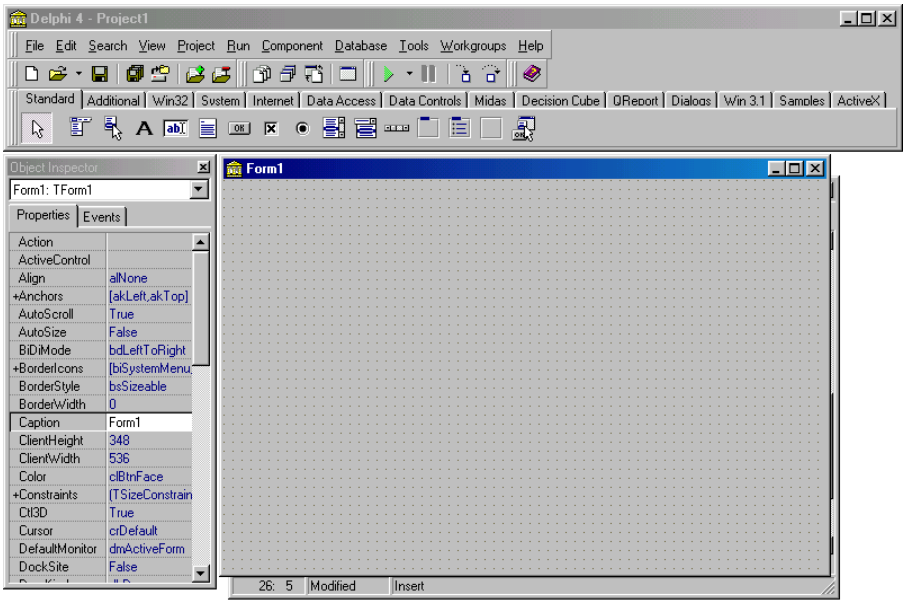


B10-rasm. Kodlarni taxrirlash oynasi

Kodlarni taxrirlash oynasida (B10-rasm) dastur matni yoziladi.

Delphi da boshlang’ich amallar va proyektlar

Delphi dasturlash tilini ishga tushirganimizda uning ishchi oyna ko’rinishini ko’ramiz (qanday ishga tushirishni avvalgi ma’ruzada ko’rib o’tilgan). U unchalik oddiy emas (10-rasm).



10-rasm.

Oynada to’rtta oyna hosil bo’lib, ular quyidagilardir: Delphi4 – bosh oynasi, Form1 – forma oynasi, Object Inspector – ob’yekt inspyektori oynasi va Unit1.pas – kodlarini tahrirlash oynasi.

Delphining bosh oynasida Delphi buyruqlar satri, buyruq tugmachalari va komponentlar palitrasi joylashgan bo’ladi.

Object Inspector oynasi yordamida ob’yektlar hususiyatlarini o’zgartirish mumkin: formalar, tugmalar, kiritish maydonlari va hokazolarni.

Buyruq menyusi: Delphining menyu satridan quyidagilar joy olgan: File, Edit, Search, View, Project, Run, Component, Database, Tools, Help.

Bularning barchasida ost menyular mavjuddir. File ning ost menyusida bir necha buyruqlar bo’lib ular yordamida yangi proyekt, formalarni ochish va ularni saqlash mumkin. Shu bilan birgalikda ochilgan proyektni yopish, Delphi dan chiqish va shularga o’xshash fayllar bilan ishlash imkoniyatlari bor:

- Edit menyusi ost menyudan foydalanib kodlarni tahrirlash, umuman kodlar sirtida turli xil amallarni bajarish mumkin;
- View yordamida esa Delphi ishchi muhiti ko’rinishini o’zgartirish mumkin;
- Run menyusi yordamida dasturni ishga tushirishni turli yo’llari amalga oshiriladi;
- Database menyusida ma`lumotlar ba’zasini tashkil qilish mumkin;
- Help menyusi esa Delphi va unda dasturlash haqidagi barcha ma’lumotlarni olish imkoniyatini yaratadi.

Buyruqlar tugmachasi: Buyruqlar tugmachasi yordamida yangi formalar yaratish, mavjud faylni ochish, dasturni saqlash, yangi forma yaratish va shunga o'xshash amallar tez bajariladi.

Komponentlar palitrasi: Bu yerda standart yoki dasturchilar tomonidan yaratilgan komponentlar mavjud bo'lib, ulardan tez va sifatli dasturlar yaratishda foydalaniladi.

Object Inspector oynasi: Object Inspector oynasi quyidagi ob'yektlarning holatini o'zgartiradi: formalar, buyruqlar tugmachasi, kodlar maydoni va boshqalar.

Dastur formasi: Dastur tuzishda ishlatiladigan barcha komponentlar dastur formasiga joylanadi va ana shu yerdan ularga o'zgartirish kiritilishi mumkin. Dastur ishga tushirilgandan so'ng, barcha amallar dastur formasi yordamida bajariladi.

Proyekt: Dephi proyekti – bu kompilyator tomonidan, dastur yaratilganidan so'ng, yaratilgan dasturga tegishli bo'lgan fayllar to'plamidir. Proyekt, bir yoki bir nechta proyekt fayllarini va modullarni o'z ichiga oladi (Unit moduli).

Proyekt fayli *.dpv kengaytmasiga ega bo'lib, proyektning umumiy holatini o'zida saqlaydi. Proyekt moduli fayli esa *.pas kengaytmali bo'lib, ishchi faylini yaratishda kompilyatorga kerak bo'luvchi prosedura, funksiya matnlari, tiplarni tavsifi va boshqa ma'lumotlarni o'zida saqlaydi.

Dastur kodi: Dastur kodi forma orqasiga yashiringan bo'lib, u yerga dastur matnlari kiritiladi. U oynaga F12 yoki Ctrl+F12 tugmalari yordamida o'tish mumkin.

Delphi kodlar muhitida dastur buyruqlarini kiritish va ularni qayta ishlash mumkin. Shuni ham ta'kidlash lozimki, Delphi kodlar muhiti avtomatik tarzda Object Pascal dasturlash tilidagi kalit so'zlarni (begin, end, procedure, const, var va bosh.) qalin harflar bilan belgilaydi.

Ma`lumot yozilgan satr(dastur izohi)ni belgilash uchun figurali qavslardan foydalaniladi. Qavs ochilsa undan keyin turgan kodlar ko'rinishi o'zgaradi. Kerakli joyda qavs berkitilsa ko'rinishi o'zgargan kodlar faqat qavs oralig'idagina qoladi va dastur ishlash jarayonida shu oraliq ishlatilmaydi.

Delphi kodlar muhitining imkoniyatlaridan yana biri shuki, u yerga biror funksiyani masalan: «**StrToFloat**» ni yozib, qavs ochsak satr ostida kichik oyna hosil bo'ladi. Bu oynada qavs ichidagi o'zgaruvchi tipi ko'rsatilgan bo'ladi, yoki biror operatorni masalan: **Label1** ni yozib nuqta qo'yilsa satr ostida nuqtadan keyinga yozish mumkin bo'lgan operatorlar ro'yhati chiqadi va ulardan kerakligini tanlab qo'yishimiz mumkin.

Kodlar oynasida biror operator sirtiga kursorni olib borib Ctrl+F1 tugmalari teng bosilsa shu operator haqidagi yordam oynasi hosil bo'ladi. U yerdan kerakli axborotni olish mumkin. Agar kursorni bo'sh joyga olib kelib, F1 bosilsa umumiy yordam ma'lumotlari chiqadi.

Kodlar oynasida tahrirlash oddiy matn muharrirlari kabi amalga oshiriladi. Ya'ni belgilangan (blokka olingan) kod nusxasini olish, qirqib olish va kerakli joyga qo'yish mumkin. Undan tashqari kodlar ichidan kerakli belgini izlab topish va almashtirish, **Delete** tugmasi yordamida kursordan keyin turgan belgini, **Backspace** yordamida esa kursordan oldin turgan belgi yoki belgilarni o'chirish mumkin.

Ctrl+→, Ctrl+← tugmachalari yordamida bir so’z keyinga va oldinga, **PgDn, PgUp** tugmachalari yordamida esa bir oyna pastga va yuqoriga o’tiladi.

Dastur bajarilayotganda yuz beradigan xatoliklar.

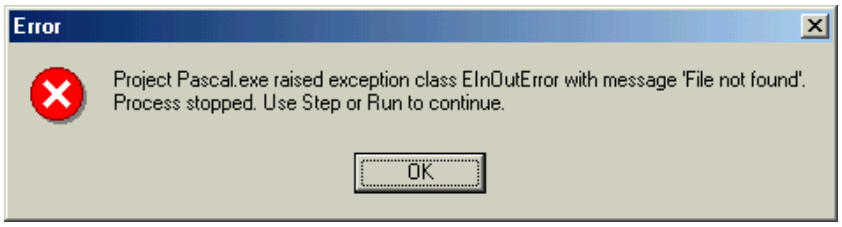
Odatda dastur tuzilayotganda ba’zi kamchilik yoki xatoliklarga yo’l qo’yamiz, dasturni ishga tushirgan vaqtimizda esa bu xatoliklar to’g’risida axborot beruvchi oyna hosil bo’ladi va bu xatoliklar quyidagilarga bo’linadi: Sintaktik; Dastur bajarilish vaqtidagi xatoliklar; Algoritmik.

Sintaktik xatoliklar operator, prosedura-funksiya, o’zgaruvchi nomi va boshqalarni notug’ri yozishdan kelib chiqadi.

Sintaktik xatoliklarni kompilyatsiya vaqtida chiquvchi xatoliklar ham deb yuritiladi (Compile-time error). Bu xatoliklarni topish boshqa xatoliklarni topishga nisbatan ancha yengil hisoblandi. Ularni kompilyator topadi va kursoni ana shu xatolik sirtiga olib kelib qo’yadi, dasturchi esa uni to’g’rilab, dasturni boshqattan ishga yuklaydi.

Dastur bajarilish vaqtidagi xatoliklar dastur ishga yuklanganidan so’ng yoki uni Testdan o’tkazayotgan vaqtda xosil bo’ladi. Masalan bu xatoliklar, massiv chegarasidan chiqib ketish, proseduraga yoki funksiyaga noto’g’ri bog’lanish, fayllar bilan noto’g’ri ishlash va boshqalar.

Xatolik yuzaga kelgan vaqtda dastur o’z ishini to’xtatadi va Delphi **“Error”** oynasida (xatolik) xatolik xaqidagi ma’lumotlarni chiqaruvchi dialogli oyna xosil bo’ladi. Bu oynadagi ma’lumotdan kerakli xulosa chiqariladi. Xatoliklar oynasi quyidagicha ko’rinishga ega bo’lishi mumkin:



Dastur bajarilish vaqtidagi xatolik.

Dastur ishini davom ettirish uchun **Error** dialogli oynadagi **OK** tugmachasi bosiladi, so’ngra **Run** menyusidan **Program Reset** buyrug’i tanlanadi.

Algoritmik xatoliklarni aniqlash ko’pchilik hollarda qiyinchilik tug’dirishi mumkin. Dastur ishga yuklanganida xatolik xaqida xech qanday ma’lumot oynaga chiqarilmaydi, lekin natija noto’g’ri chiqadi. Bu turdagi xatoliklarni aniqlash uchun dasturchi dasturni qadamlab bajartirishi va har bir qadamdagi natijani tekshirib borsa maqsadga muvofiq bo’ladi.

Dasturni qadamlab bajartirish uchun birinchi navbatda **Run** menyusidagi **Add watch** (**Ctrl+F5**) buyrug’i tanlanadi va o’zgaruvchi qiymatlarini o’zida saqlaydigan (**Watch List**) oyna xosil bo’ladi. Unga xatolikni tekshirish jarayonida zarur bo’lgan o’zgaruvchilar kiritiladi. So’ngra kursor noto’g’ri bajarilayotgan algoritm sirtiga olib boriladi va **Run** menyusidan **Run to Cursor** (**F4**) buyrug’i tanlanadi. Yuqoridagi amalni bajarganimizdan so’ng, dastur ishga yuklanadi va dasturdagi ketma-ketlik kursor turgan joyga kelganida dastur o’z ishini vaqtinchalik to’xtatadi. Dasturning

qolgan qismini **F7** tugmachasi yordamida qadamlab bajartirish mumkin va har bir qadamdan so’ng, **Watch List** oynasidagi o’zgaruvchi qiymatlari tekshirib boriladi.

	Edit.Text ga qiymatning noto`gri kiritilishi qaysi xatolik turiga kiradi?

3-Ma`ruza. Komponent tushunchasi. Standart komponentlar. Delphida xususiyat va hodisa tushunchalari. Ma`lumotlar turi, o’zgaruvchilar, konstantalar, amallar va ularning yozilishi. (2 soat)

O’quv modul birliklari:

1. Komponent tushunchasi. Standart komponentlar.
2. Forma xususiyatlari. Forma hodisalari.
3. O’zgaruvchilar. Ma`lumotlar turi.
4. O’zgaruvchilar, konstantalar, amallar va ularning yozilishi.

Aniqlashtirilgan o’quv maqsadlari:

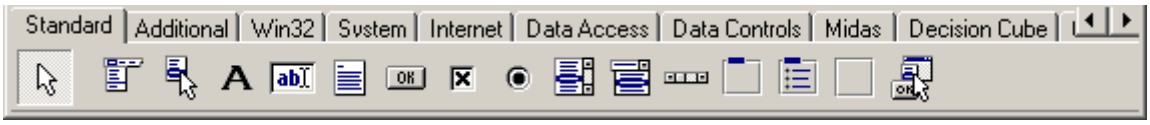
Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. O’zgaruvchilar va ma`lumotlar turi haqida kengroq tushunchaga ega bo’ladi.
2. Forma xususiyatlari va hodisalarni tushunadilar
3. O’zgaruvchilar va ma`lumotlar turlarini biladilar
4. O’zgaruvchilar, konstantalar, amallar va ularning yozilishini o’zlashtiradilar.

Delphida komponentlar

Komponentlar palitrasi bizga kerakli ob`ekt(komponent)ni tanlab undan foydalanish imkoniyatini yaratadi. Komponentdan foydalanish uchun dastlab sichqonchaning chap tugmasi kerakli **komponent**, so`ngra **forma**ning sirtida bosiladi. Formadagi komponentning turgan joyini o`zgartirish uchun sichqonchadan foydalanish mumkin.

Komponentlar palitrasi guruhlar bo`yicha sahifalarga bo`lingan bo`lib, sahifalar soni 14 tadir. Komponentlar palitrasida **Standard, Additional, Dialogs** kabi sahifalar mavjud. Istalgan sahifa sirtida sichqonchaning chap tugmasi bosilsa, tanlangan sahifadagi komponentlar ro`yhati chqarib beriladi. Quyidagi rasmda komponentlar palitrasining ko`rinishi keltirilgan:

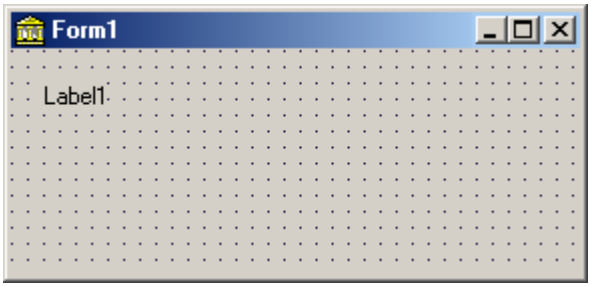


Tanlangan komponent “Objekt Inspector” oynasi yordamida tahrirlanadi.

Xususiyat

Delphida barcha ob`ektning o`ziga hos xususiyati (Properties) bo`lib, bu xususiyatlar dastuchi tomonidan o`zgartirilishi mumkin. Barcha ob`ektlarning xususiyatlari bir-biriga juda o`xshash bolib, biz misol sifatida “Label” ob`ektining xususiyatlari bilan tanishamiz.

Ob`ekt xususiyatini o`zgartirish uchun “Object Inspector” oynasining “Properties” bo`limidan foydalanamiz. Dastlab **Label** komponentini formaga joylaymiz. Dasturning forma ko`rinishi quyidagicha bo`ladi:



Label komponentining xususiyatlari quydagilardir:

Align – Ob`ektning joylashish o`rnini o`zgartiradi. Unda **alButton, alClient, alLeft, alNone, alRight, alTop** ko`rsatkichlari mavjud bo`lib, ular quyidagi vazifalarni bagaradi:

Ko`rsatkich	Vazifasi
alNone	**Formaning *ko`rsatilgan* qismida**
alTop	**Formaning *yuqori* qismida**
alBottom	**Formaning *quyi* qismida**
alLeft	**Formaning *chap* qismida**
alRight	**Formaning *o`ng* qismida**
alClient	**Formaning *barcha* qismida**

Alignment – O`bekt tarkibidagi matn o`rnini gorizontal yo`nalish bo`yicha o`zgartirish. Undagi ko`rsatrichlarning vazifasi quyidagicha:

Ko`rsatkich	Vazifasi
taLeftJsirtify	**Matn ob`ektning *chap* qismida**
taRightJsirtify	**Matn ob`ektning *o`ng* qismida**
taCenter	**Matn ob`ektning *o`rta* qismida**

Anchors – forma hajmi o`zgarishiga qarab ob`ektning turgan joyini o`zgartirish (ko`rsatkich **True** bo`lgandagina aktivlashadi). Undagi ko`rsatrichlarning vazifasi quyidagicha:

Ko`rsatkich	Vazifasi
taTop	**Ob`ektning turgan joyi formani *yuqori* qismi bo`yicha o`zgarmaydi**
taLeft	**Ob`ektning turgan joyi formani *chap* qismi bo`yicha o`zgarmaydi**
taRight	**Ob`ektning turgan joyi formani *o`ng* qismi bo`yicha o`zgarmaydi**
taButton	**Ob`ektning turgan joyi formani *ostki* qismi bo`yicha o`zgarmaydi**

AutoSize – Ob`ekt hajmini o`zgaruvchan yoki o`zgarmas holiga keltirish. Ko`rsatkich ***True*** bo`lganida ob`ekt hajmi o`zgaruvchan, aks holda o`zgamas bo`ladi.

BiDiMode – Ikki tomonlama rejimdan foydalanish imkoniyatini yaratadi.

Caption – foydalanuvchi uchun izox vazifasini bajaradi. Undagi yozilgan matn ob`ekt sirtida o`z aksini topadi. Ampersand (&)qaysi belgining oldingi qismiga qo`yilsa ana shu belgi ob`ektni aktivlashtiruvchi hisoblanadi. Alt+<belgi> bosilsa kerakli ob`ekt aktivlashadi. Misol uchun:

Label1.Caption := ‘F&ile’;

bu erda **Alt+I** tugmalari teng bosilsa **Label1** ob`ekti aktivlashadi.

Color – Ob`ekt sirtining rangini tanlash imkonini beradi.

Constraints – ob`ekt chegaralarini aniqlash. Undagi ko`rsatrichlarning vazifasi quyidagicha:

Ko`rsatkich	Vazifasi
MaxHeight	**Ob`ektning maksimum balandligi**
MaxWidth	**Ob`ektning maksimum uzunligi**
MinHeight	**Ob`ektning mimimum balandligi**
MinWidth	**Ob`ektning mimimum uzunligi.**

Cursor – sichqoncha ko`rsatkichini ob`ekt sirtida o`zgartirish.

Enabled – ob`ektni ishch holatga keltirish. **Enabled** ko`rsatkichi ***true*** bolsa ob`ekt ishchi holatda bo`ladi. Aks holda (***False***) ishchi holatda emas. Ko`pchilik hollarda vaqtinchalik bajarilmaydigan ob`ektlar ishchi holdan chiqarib turiladi (ob`ekt.Enabled := False).

Font – ob`ektning sriftini o`zgartirish.

Height – ob`ektning bo`yi. (ob`ekt.Height := 150; ob`ekt bo`yi 150 ta pikselga teng).

Hint – kichik yordam. Sichqoncha ko`rsatkichi ob`ekt sirtiga olib borilganda kichik yordam chiqariladi. (ob`ekt.Hint :=’Bu Label komponenti’;).

Layout – ob`ekt tarkibidagi matnni vertikal yo`nalish bo`yicha chop etish. Undagi ko`rsatrichlarning vazifasi quyidagicha:

Ko`rsatkich	Vazifasi
tlTop	**Matn ob`ektning *yuqori* qismida**
tlCenter	**Matn ob`ektning *o`rta* qismida**
tlBottom	**Matn ob`ektning *ostki* qismida**

Left – ob`ektni formaaning chap qismiga nisbatan joylashish o`rni (piksellarda).

Name – ob`ektning nomi. Ob`ekt nomining birinch xarfi lotin belgisi bilan boshlanishi shart.

ShowHint – kichik yordamni chiqarishga ruhsat berishni ta`minlaydi (***True***).

Tag – qo`shimcha xususiyat.

Top - ob`ektni formaganing yuqori qismiga nisbatan joylashish o`rni (piksellarda).

Transpared – ob`ekt fonini olib tashlash (***True***).

Visible – ob`ektni korsatish (***True***) yoki yashirish (***False***).

Width - ob`ekt uzunligi. (ob`ekt.Width := 15; ob`ekt uzunligi 150 pikselga teng).

Xodisalar

Delphida dasturchilar uchun dasturni soddalastirish maqsadida xodisa (Events) yo`lga qo`yilgan.

Xodisa – bu ob`ekt qismida bajariladigan amallar turdir. Misol uchu, ob`ekt sirtida Inter (Enter) tugmasining bosilishi, sichqonchaning chap yoki o`ng tugmasining bosilishi xodisadir.

Xodisa ro`yhatlari **Object Inspector** oynasining **Events** sahifasida joylashgan bo`ladi. U yerda **OnClik, OnDBLClik, OnMouseMove** kabi xodisalarni ko`rishimiz mumkin.

Xodisadan foydalanish uchu kerakli xodisaning o`ng qismida sichqonchaning chap tugmasi ikki marta bosiladi. Masalan **Forma** ob`ekti bosilganida qandaydir amal bajarish uchun. **Form** ob`ekti tanlanadi va **Object Inspector/Events/OnClik** ning sirtida sichqonchaning chap tugmasi ikki marta bosiladi. Yuqoridagi amallar ketma-ketligi bajarilganidan so`ng Delphining kodlar oynasida quyidagi prosedura hosil bo`ladi:

```
procedure TForm1.FormClick(Sender: TObject);
begin

end;
```

Biz unga quyidagi o`zgartirishni kiritamiz:

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

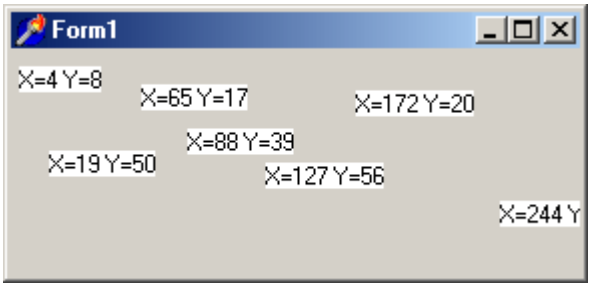
```
 begin
  MessageDlg('Salomlar !!!', mtInformation, [mbOk], 0);
end;
```

Yuqoridagi amallar bajarilgach, dastur ishga yuklanidan so`ng formaning istalgan qismida sichqonchaning chap tugmasi bosilsa quyidagi oyna hosil bo`ladi:

Yuqoridagi misoldan ko`rinib turibdiki, qandaydir xodisa ro`y berganida javob olish mumkin ekan. Ko`pchilik dasturchilar bu vaqtda qanday jarayon bo`layotganligini tushunmay dastur tuzadilar. Shuni aytish mumkinki, har-bir xodisa ro`y berganida operatsion sistema bu xodisani aniqlaydi va dasturga xodisaning turi xaqidagi xabarni uzatadi. Misol qilib forma sirtida sichqonchaning chap tugmasi bosilganida ro`y beradigan xodisani ko`rishmuz mumkin. Buning uchu xodisa sahifasidan OnMouseDown xodisasi tanlanadi va dasturga quyidagi o`zgartirish kiritiladi:

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  Canvas.TextOut(X, Y, 'X='+IntToStr(X)+' Y='+IntToStr(Y));
end;
```

Dasturni ishga yuklab forma sirtida sichqonchaning chap tugmasi bosilsa quyidagiga o`xshash natijani ko`rish mumkin:



Ko`rinib turibdiki, xodisalar bilan ishlash unchalik murakkab emas ekan. Yana bir misol sifatida quyidagi dasturni ko`rishimiz mumkin (**OnKeyDown** (tugmacha bosilganida) xodisasi):

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
 begin
  MessageDlg(Chr(Key), mtInformation, [mbOk], 0);
end;
```

Dasturni ishga tushirib qanday natija chiqishini bilib olarsiz.

Yuqoridagi yozuvlardan foydalanib xodisa nima ekanligini bilb oldik. Endi yangi xodisa yaratishni ko`rib chiqamiz. Xodisa yaratishning umumiy ko`rinishi quyidagicha bo`ladi:

procedure Handler_Name(**var** Msg : MessageType); message WM_XXXXX;

bu erda

Handler_Name – uslub nomi;
Msg – uzatiluvchi parametr nomi;
MessageType - xabarga mos keluvch qandaydir bir tip;
message – xizmatchi so`zi joriy uslub xabarlarni qayta ishlovchi ekanligini bildiradi;
WM_XXXXX – konstanta yoki amal, yani Windows xabarini aniqlovchi nomer.

Xabarlarni qayta ishlashni misol yordamida ko`rib chiqamiz. Misol uchun sichqonchaning o`ng tugmasi forma sirtida bosilganida qandaydir xabarnoma chiqsin. Dastlab yangi proyekt yaratiladi (File/NewApplication). So`ngra kodlar oynasiga o`tiladi(F2). U yerda quyidagi dasturni ko`rishimiz mumkin:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
end.
```

Uni quyidagicha o`zgartiramiz:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
  private
    { Private declarations }
   procedure WMRButtonDown(var Msg: TWMMouse); message
WM_RBUTTONDOWN;
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.WMRButtonDown(var Msg : TWMMouse);
begin
  MessageDlg('Sichqonchaning ung tugmasi bosildi.', mtInformation, [mbOK], 0);
end;
end.
```

Dastur ishga yuklanib forma sirtida shichqonchaning o`ng tugmasi bosilsa, quyidagi dialog oynasi hosil bo`ladi:

	Xodisaning shunday turini tanlanki. Uning yordamida formadagi tugmachalarning necha marta boslganligini bilish mumkin bo`lsin.

Ma’lumotlar tipi

Object Pascal dasturlash tilida ma’lumotlarni qayta ishlash uchun turli tiplar mavjud bo’lib, ular butun sonli, haqiqiy sonli, belgili, satrli va mantiqiy tiplardir.

Butun tip

Object Pascal dasturlash tili yetti xil butun tiplarni o’z ichiga oladi va ular quyidagilar: ShortInt, SmalInt, LongInt, Byte, Word, Integer va Cordinal.

ShortInt, SmalInt, LongInt, Byte va Word ma’lumot tiplari asosiy (fundamental) toifaga kiradi. Asosiy toifadagi tiplarning formati va chegarasi protsessor razryadiga va ishlayotgan operatsion sistemaga bog’liq emas.

Quyidagi jadvalda butun tiplarning asosiy toifasi keltirilgan:

Tip	Chegara	Format
Shortint	-128..127	Belgili, 8 bit
Smallint	-32768..32767	Belgili, 16 bit
Longint	-2147483648..2147483647	Belgili, 32 bit
Byte	0..255	Belgilisiz, 8 bit
Word	0..65535	Belgilisiz, 16 bit

Integer va Cardinal ma’lumot tiplari umumiy (fundamental) toifaga kiradi. Umumiy toifadagi tiplarning formati va chegarasi protsessor razryadiga va ishlayotgan operatsion sistemaga bog’liq bo’ladi.

Quyidagi jadvalda butun tiplarning umumiy toifasi keltirilgan:

Tip	Chegara	Format
Integer	-32768..32767	Belgili, 16 bit
Cardinal	0..65535	Belgilisiz, 16 bit
Integer	-2147483648..2147483647	Belgili, 32 bit
Cardinal	0..21474883647	Belgilisiz, 32 bit

Haqiqiy tip

Object Pascal dasturlash tili to’rt xil haqiqiy tiplarni o’z ichiga oladi va ular quyidagilar: Real, Single, double, extended. Bu tiplar bir biridan sonlarini qabul qilish chegarasi va aniqlik darajasi bilan farq qiladi. Ular quyidagi jadvalda keltirilgan:

Tip	Chegara	Aniqlik darajasi	Bayt
Real	$2,9*10^{-39}..1,7*10^{+38}$	11-12	6
Single	$1,5*10^{-45}..3,4*10^{+38}$	7-8	4
Doble	$5,10^{-324}..1,7*10^{+308}$	15-16	8
Extended	$3,4*10^{-4932}..1,1*10^{+4932}$	19-20	10

Belgili tip

Object Pascal dasturlash tilida uch xil belgili tiplar mavjud. Ular AnsiChar, WideChar va Char. Belgili tiplar ham butun tiplar kabi asosiy va umumiy toifalarga bo'linadi. Asoiy toifaga AnsiChar va WideChar tiplari kiradi.

AnsiChar tipi ANSI belgilarini o'z ichiga oladi. Ular chop etiluvchi va ishchi belgilar bo'lib, 0 dan 255 gacha kodlanadi.

WideChar tipi Unicode belgilarini qabul qiladi va ular 0 dan 65535 gacha kodlanadi.

Char tipi umumiy toifaga kiradi va o'zida ANSI belgilarning chop etiluvchi va ishchi qismini mujassamlashtirgan.

Satrli tip

Object Pascal dasturlash tili uch xil satrli tipni o'z ichiga olgan bo'lib, ular ShortString, LongSting va WideString lar.

ShortString tipi 0 dan 255 tagacha belgilarni qabul qiladi.

LongString tipi kompyuter tezkor xotirasining bo'sh qismi qancha bo'lsa, unga shuncha belgi sig'adi.

WideString tipi kompyuter tezkor xotirasining bo'sh qismi qancha bo'lsa, unga shuncha belgi sig'adi. LongString tipidan farqi shundaki, uning har bir belgisi Unicode belgisidan tashkil topgan.

Eslatma: Turbo Pascal da ishlatiluvchi String tipi Object Pascal da ham ishlatiladi. Uning xususiyati ShorString tipi bilan bir xil.

Mantiqiy tip

Object Pascal dasturlash tilida Boolean mantiqiy tipi bo'lib, u True (rost) va False (yolg'on) qiymatlariga ega. Mantiqiy tipli qiymatlar ham tartiblangan, ya'ni False<True.

Asosan quyidagi uchta mantiqiy amaldan ko'proq foydalaniladi: not - rad etmoq, and - mantiqiy ko'paytirish, or - mantiqiy qo'shish.

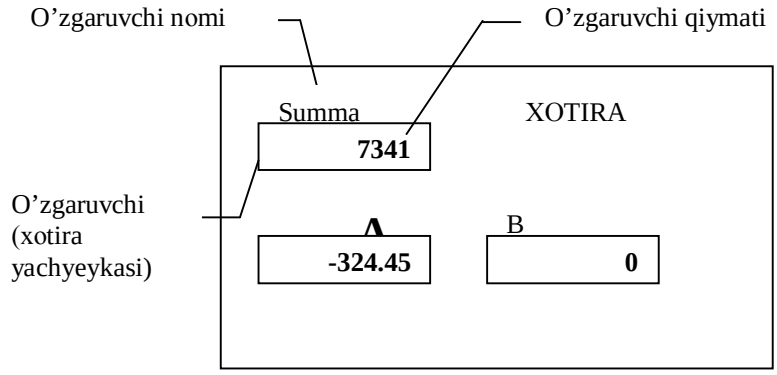
Bu amallarni faqat mantiqiy o'zgarmaslar sirtidagina ishlatish mumkin va natijada yana mantiqiy o'zgarmas hosil bo'ladi. Quyida mantiqiy o'zgarmaslar sirtida amallar jadvali ko'rsatilgan:

Mantiqiy ko'paytirish	**Mantiqiy qo'shish**	**Mantiqiy rad etmoq**
True and true = true	true or true = true	not true = false
True and false = false	true or false = true	not false = true
False and true = false	False or true = true	
False and false = false	False or false = false	

O’zgaruvchilar

O’zgaruvchilar nima ekanligini tushunish dasturlashda katta ahamiyatga ega. O’zgaruvchining qiymatlarini o’zida saqlay oladigan qurilmaga o’xshatish mumkin. Misol uchun sonlarni. Dastur bajarilayotgan vaqtda bu qurilma qiymatlari o’zgarishi yoki boshqa qurilma qiymatlarini qabul qilishi mumkin.

Amaldagi barcha dasturlarda hisoblash ishlarini olib borish uchun turli qiymatlar kompyuter xotirasida saqlanadi. Masalan, kvadrat tenglamani yechish dasturida koeffisiyentlari, diskriminant va tenglama ildizlari uchun o’zgaruvchilar kerak bo’ladi.



11-rasm

O’zgaruvchiga shunday ta’rif berish mumkin: **O’zgaruvchi** – bu kompyuter xotira(yachyeyka)sidagi maydondir (11-rasm).

Dasturda qatnashadigan har bir o’zgaruvchiga alohida nom berilishi shart. O’zgaruvchini nomlashda lotin alfaviti, son va bir nechta ishchi belgilardan foydalaniladi. O’zgaruvchining birinchi harfi lotin bo’lishi kerak. O’zgaruvchini e’lon qilishda yoki undan foydalanishda bo’sh joy (Space) belgisini qo’yish mumkin emas. Undan tashqari Object Pascal buyruqlarini ham o’zgaruvchi nomi sifatida ishlatish mumkin emas (Begin, End, Private, For,…).

Object Pascal dasturlash tilida ishlatilishi lozim bo’lgan har bir o’zgaruvchi e’lon qilinishi shart. Bunda nafaqat o’zgaruvchi borligi eslatiladi, balki o’zgaruvchi uchun tip ham beriladi.

<O’zgaruvchi nomi> : <Tip>;
<O’zgaruvchi nomi> - E’lon qilingan o’zgaruvchi nomi
<Tip> - Object Pascal dasturlash tilining tiplaridan biri.
Misol uchun:
A: Real;
B: Real;
I: Integer;

Ko’rsatilgan misolda ikkita Real tipli, bitta Integer tipli o’zgaruvchi e’lon qilingan.

Bir xil tipli bir nechta o’zgaruvchilarni e’lon qilish uchun ularning orasiga vergul (,) qo’yib yoziladi va o’zgaruvchilar nomini kiritish tugaganidan so’ng ikki nuqta (:) qo’yiladi va tip nomi beriladi. Masalan:

A, b, c: Real;

X1, x2: Real;

Konstantalar

Object Pascal dasturlash tilida konstantalarning ko’rinishi ikki xil bo’lib, ular oddiy va nomlangan turlarga bo’linadi.

Oddiy konstanta – bu butun, haqiqiy, satrli, belgili yoki mantiqiy ifoda bo’lishi mumkin.

Dastur matnida sonli konstantalar matematikada qanday yozilsa shunday yoziladi.

Masalan:

123
0,0
-524,03
0

Satrli va simvolli konstantalar apostrof (‘) ichiga olib yoziladi. Masalan:

‘Object Pascal dasturlash tili’
‘Delphi 4’
‘2.4’
‘d’

Amallar va ularning yozilishi

Butun yoki haqiqiy tipli, sonli natija beruvchi ifodani (odatda bunday ifodani arifmetik ifoda deb ataladi) hisoblash uchun arifmetik o’zlashtirish operatoridan foydalaniladi. Arifmetik ifodada qatnashuvchi barcha o’zgaruvchilar haqiqiy yoki butun tipli bo’lishi kerak. Arifmetik ifoda: sonlar, o’zgarmaslar, o’zgaruvchilar va funksiyalardan tashkil topadi, hamda +, -, *, /, div, mod kabi amallar yordamida yoziladi. Arifmetik amallarni bajarilishi quyidagi tartibda bo’ladi : *, /, div, mod, +, - .

Ifodani bajarilishidagi bu tartibni o’zgartirish uchun kichik qavslardan foydalaniladi. Ifodaning qavslar ichiga olib yozilgan qismlari mustaqil holda birinchi galda bajariladi.

Sanab o’tilgan arifmetik amallarning vazifalari bizga matematika kursidan ma’lum. Lekin, bu ro’yhatdagi div va mod amallari bilan tanish emasmiz.

Div – butun bo’lishni anglatadi, bo’linmani butun qismi qoldirilib, qoldiq tashlab yuboriladi. Misol:

7 div 2 = 3
5 div 3 = 1
-7 div 2 = -3
-7 div –2 = 3
2 div 5 = 0
3 div 4 = 0

Mod – butun sonlar bo’linmasining qoldig’ini aniqlaydi. *m mod n* qiymat faqat *n>0* dagina aniqlangan. Agar *m≥0* bo’lsa, *m mod nqm-((m div n)*n), m<0* bo’lsa m mod nqm-*((m div n)*n)+n, m mod n* ning natijasi doim musbat sondir. Misol:

7 mod 2 = 1
3 mod 5 = 3
(-14) mod 3 = 1
(-10) mod 5 = 0

Objekt Paskal tilida ham boshqa algoritmik tillar kabi arifmetik standart funksiyalar mavjud. Bu funksiyalarni matematik yozilishi va Objekt Paskal tilida ifodalanishi quyidagi jadvalda keltirilgan:

Funksiyalar	**Paskal tilida ifodalanishi**
sin x	sin(x)
Cosx	cos(x)
Arctg x	Arctan(x)
Lnx	ln(x)
e^x	exp(x)
$\sqrt{x}$	Sqrt(x)
$\|x\|$	Abs(x)
x^2	sqr(x)
Argumentning kasr qismini topish funksiyasi	Frac(x)
Argumetning butun qismini topish funksiyasi	Int(x)

Arifmetik ifodaga doir misollar :

2 * 5 – 4 * 3,

9 div 4/2,
45 / 5 / 3,
a + b / 2 * 7.2 - sqrt(7),
exp(2 - a) * 9.7 - 6.1 * 6.1

Object Pascal dasturlash tilida darajaga ko'tarish amali yo'q, shuning uchun, bu amalni bajarishda logarifmlash qoidasidan foydalanamiz.

Misol: $y=a^n$, $a>0$ ifodani hisoblashni ko'rib chiqaylik. Tenglikni ikkala tomonini logarifmlaymiz:

$Lny=lna^n$, logarifm xossasiga ko'ra

$Lny=nlna$, bu tenglikdan "u" ni aniqlaymiz,

$u=ye^{nlna}$ - bu tenglikni Paskal tilida quyidagicha yozish mumkin: $y=exp(n*ln(a))$.

Endi sal murakkabroq arifmetik ifodalarni Paskal tilida yozilishini ko'rib chiqaylik.

Matematik yozuvi	**Paskal tilidagi yozuvi**
$\frac{A+b}{C+d}$	(a + b) / (c + d)
$\frac{a\cdot(a+b)}{bc}$	A * (a + b) / (b * c)
$\frac{1}{1-\frac{1}{1-\frac{1}{x}}}$	1 / (1 – 1 / (1 – 1 / x))
$2-(x-b)^2-e^{ax}+\sin CX$	2 - sqr(x – b) - exp(a * x) + sin(c * x)
$x\cdot\frac{e^{x^2+y^2}-1}{\sqrt{\lvert x^2+y^2\rvert}}$	X * (exp(x * x + y * y) - 1) / sqrt(abs(x * x + y * y))

Endi arifmetik o'zlashtirish operatoriga doir misollar ko'rib chiqamiz:

X := 0;
C := sqrt(a * a + b * b);
Y := 2 * pi * r; I := I + 1; I := 5 / 4; x := a – b / 2;

O'zlashtirish operatorining o'ng tomonidagi ifodada qatnashuvchi o'zgaruvchilar, albatta, bu operatordan oldin o'zining qiymatlariga ega bo'lishi kerak. Aks holda, o'zlashtirish operatori o'z ishini bajara olmaydi. Dastur tuzishda ko'pchilik yo'l qo'yadigan xatolikni quyidagi misolda tahlil qilib ko'ring:

To'g'ri tuzilgan dastur **Noto'g'ri tuzilgan dastur**

```
Var
    a, x, y: Real;
Begin
    A := 2.3;
    X := 3.1;
    Y := a * x;
   Label1.Caption := IntToStr(y);
End;
```

```
Var
    a, x, y:Real;
Begin
    A := 2.3;
    Y := a*x;
    Label1.Caption := IntToStr(y);
End.
{o'zlashtirish operatorining o'ng
tomonidagi "X" o'zgaruvchining qiymati
aniqlanmagan}
```

1. Xizmatch so`zlarni yozing va ularning vazifalarini tushuntiring.
2. O`zgaruvch va o`zgarmaslarning farqini aniqlang.

4-Ma`ruza. Standart funktsiyalar. Buyruqlar. (If Case, For, While, Repeat,Goto). Belgilar va satrlar. (2 soat)

O'quv modul birliklari:

1. Standart funktsiyalar.
2. Buyruqlar. (If Case, For, While, Repeat,Goto)
3. Belgilar va satrlar.

Aniqlashtirilgan o'quv maqsadlari:

Talaba ushbu mavzuni to'la o'zlashtirgandan so'ng:

1. Stndart funktsiyalar o'rganadi va ular asosida dastur tuzishni o'zlashtiradi.
2. Tarmoqlanuvchi va takrorlanuvchi jarayonlar uchun dastur yazo oladi.
3. Belgilar va satrlar to'g'risida tushunchaga ega bo'ladi va ular asosida kichik dasturlar yarata oladi.

Shart

Kunlik hayotda shartlar **Ha** yoki **Yo'q** deb javob berish mumkin bo'lgan savol ko'rinishidagi shaklda bo'ladi. Masalan:

- Qarshilik nolga tengmi?
- Javob to'g'rimi?
- Harid qiymati 300 so'mdan ko'pmi?

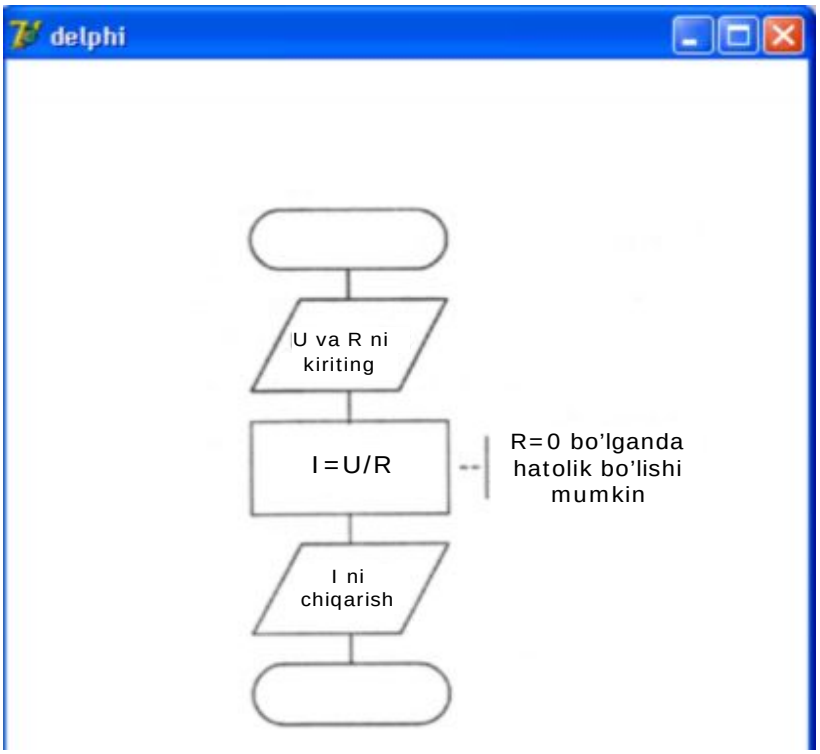
Dasturda shartlar mantiqiy tipdagi (Boolean) ifoda bo'lib, **True** (Rost) yoki **False** (Yo'lg'on) qiymatlaridan birini qabul qiladi.

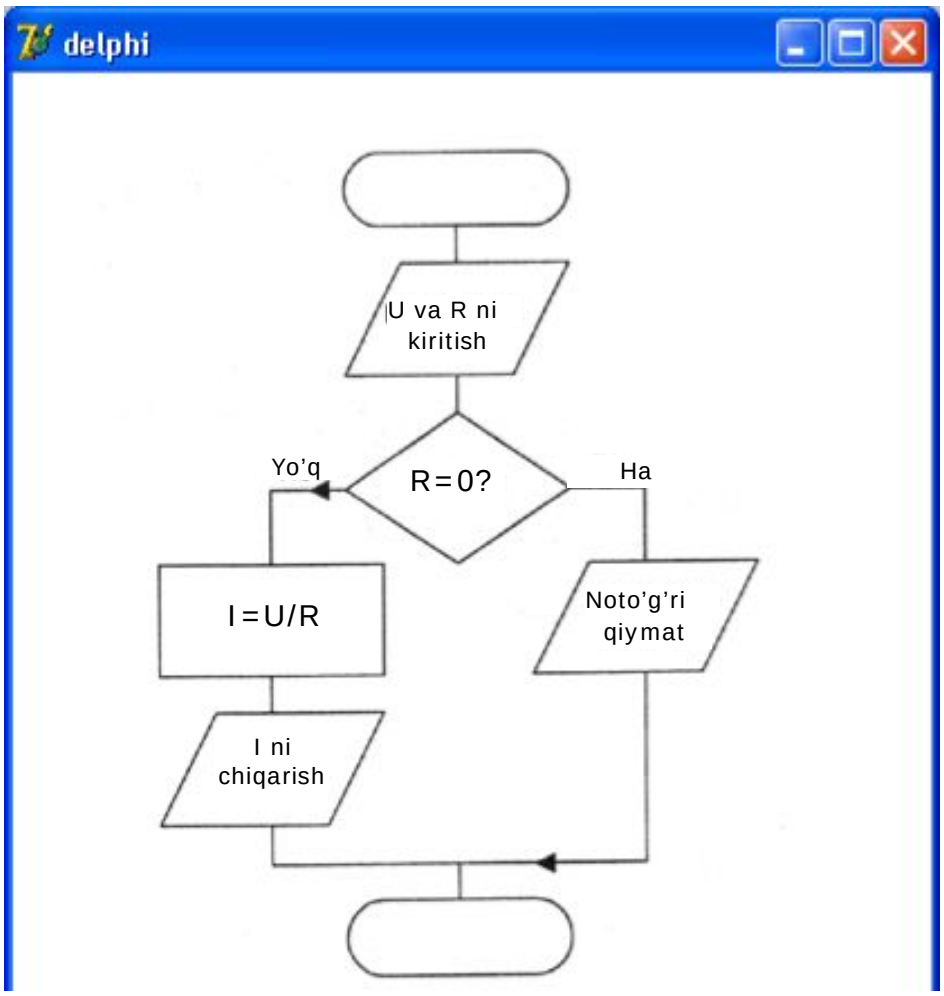
Oddiy shart ikkita operand va solishtirish operatoridan iborat. Shartning umumiy ko’rinishi quyidagicha bo’ladi:

On1 Operator On2

Bu yerda:

- On1 va On2 — o’zgaruvchi, konstanta, funksiya yoki ifoda bo’lishi mumkin bo’lgan shart operandlari;
- Operator — solishtirish operatori.





2.1-rasm. Bir masalani yechish algoritmining ikki xil varianti

Delphi tilida olti xil turdagi solishtirish operatorlari mavjud. Ular 2.1-jadval keltirilgan.

2.1-jadval

Operator	**Izox**	**Solishtirish natijasi**
>	Katta	Agar birinchi operand ikkinchisidan katta bo'lsa **True**, aks holda **False**
<	Kichik	Agar birinchi operand ikkinchisidan kichik bo'lsa **True**, aks holda **False**
=	Teng	Agar birinchi operand ikkinchisiga teng bo'lsa **True**, aks holda **False**
<>	Teng emas	Agar birinchi operand ikkinchisiga teng bo'lmasa bo'lsa **True**, aks holda **False**
>=	Katta yoki teng	Agar birinchi operand ikkinchisidan katta yoki teng bo'lsa **True**, aks holda **False**
<=	Kichik yoki teng	Agar birinchi operand ikkinchisidan kichik yoki teng bo'lsa **True**, aks holda **False**

Quyida shartga misol keltirilgan:

1) Summa < 1000
2) Score >= HBound
3) Sim = Chr(13)

Birinchi misolda shartning operandlari o'zgaruvchi va kostantadir. Bu shartning qiymati **Summa** o'zgaruvchisining qiymatiga bo'g'liq. Agar **Summa** 1000 dan kichik bo'lsa shart **rost** bo'ladi va **True** qiymatga ega bo'ladi. Agar **Summa** 1000dan katta yoki teng bo'lsa shart **yo'lg'on** bo'ladi va False qiymatga ega bo'ladi.

Ikkinchi misolda operandlar sifatida o'zgaruvchilar ishlatilgan. Agar Score o'zgaruvchisining qiymati Hbound o'zgaruvchisinikidan katta yoki teng bo'lsa bu shartning qiymati True bo'ladi.

Uchinchi misolda ikkinchi operand sifatida funksiyadan foydalanilgan. Agar **Sim** o'zgaruvchisining qiymati <Enter> tugmasi kodi (13) simvoliga teng bo'lsa shartning qiymati **True** bo'ladi.

Shart yozayotganda shartning operandlari bir xil tipda bo'lishiga asosiy e'tiborni berish kerak. Agar har xil tipli bo'lsa ularni bir xil tipga keltirish kerak. Masalan, agar **Key** o'zgaruvchisi **integer** sifatida e'lon qilingan bo'lsa, u holda quyidagi shart xato bo'ladi:

Key = Chr(13)

Chunki **Chr** funksiyasi **char** (belgili) tipidagi qiymatni qaytaradi.

Kompilyator noto'g'ri shartni topganida incompatible types xatoligini chiqaradi.

Oddiy shartlardan **and** (mantiqiy VA), **or** (nantiqiy YOKI) va **not** (inkor) mantiqiy operatorlaridan foydalanib murakkab shartlarni qurish mumkin.

Murakkab shartlarning umumiy ko'rinishi quyidagicha:

shart1 operator shart2

bu yerda:

- shart1 va shart2 — oddiy shartlar (mantiqiy tipdagi ifodalar);
- operator — and yoki or operatori.

Masalan:

(ch >= '0') **and** (ch <= '9')

(day = 7) or (day = 6)

(Forml.Editl.Text <> ' ') or (Forml.Edit2.Text <> ")

Forml.CheckBoxl.Checked **and** (Forml.Editl.Text <> ")

2.2.-jadvalda **and**, **or** va **not** mantiqiy operatorlarining bajarilish natijasi keltirilgan.

2.2-jadval.

Op1	Op2	Op1 and Op2	Op1 or Op2	not Op1
False	False	False	False	True
False	True	False	True	True

True	False	False	True	False
True	True	True	True	False

Murrakkab shartlarni yozganda oddiy shartlarni albatta qavs ichiga yozish kerak.

Masalan, chegirma olish sharti quyidagicha o'zgartirilgan bosin:"Agar xarid qiymati 100 so'mdan oshsa va yakshanba kuni xarid qilingan bo'lsa chegirma olinsin". Agar xafta kuni butun tipdagi Day o'zgaruvchisida aniqlansa va yakshanba qiymati ettiga teng bo'lsa, shart quyidagicha yoziladi:

(Summa > 100) **and** (Day = 7)

Agar shartga "xarid qiymati 500 so'mdan oshsa ixtiyoriy kunda chegirma olinsin" deb qo'shsak, u holda shartni quyodagicha yozish mumkin:

((Summa > 100) **and** (Day =7)) **or** (Summa > 500)

Tanlash

Delphida tanlash **if** va **case** operatorlari yordamida amalga oshiriladi.

If buyrug'i ikkita bo'lishi mumkin bo'lgan variantlardan birini tanlaydi, **case** buyrug'i esa bir nechtadan bittasini tanlaydi.

If buyrug'i

If buyrug'i dastur ishini davom ettirishi uchun ikkita bo'lishi mumkin bo'lgan variantlardan birini tanlaydi. Shart bajarilishiga qarab tanlash amalga oshiriladi.

If buyrug'ining umumiy ko'rinishi quyidagicha:

```
if shart then
begin
// agar shart rost bo'lsa,
// bajarilishi kerak bolgan buyruqlar
end
else
begin
// agar shart yolg'on bo'lsa,
// bajarilishi kerak bolgan buyruqlar
end;
```

Else buyrug'idan oldin (**end** dan keyin) nuqtali vergul qo'yilmasligiga e'tibor bering.

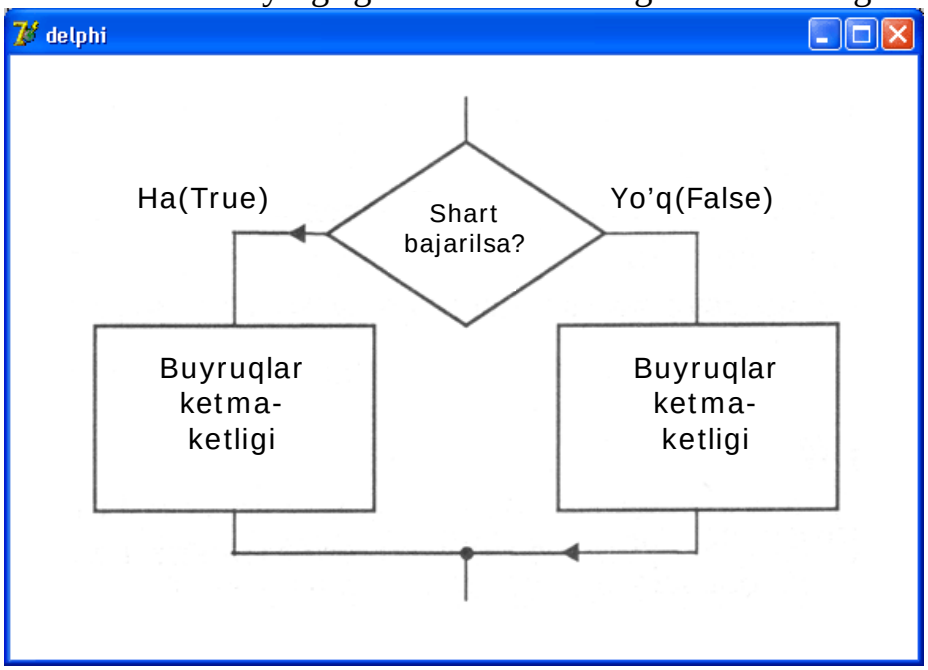
If buyrug'i quyidagicha bajariladi:

1. shatning qiymatini hisoblaydi;

2. Agar shart rost bo'lsa (True), u holda **then** so'zidan keyingi (**begin** va **end** oralig'i) buyruqlar bajariladi. Shu bilan if buyrug'ining bajarilishi tugaydi, ya'ni **else**

so’zidan keyingi buyruqlar bajarilmaydi. Agar shart yolg’on bo’lsa (False), u holda **else** so’zidan keyingi (**begin** va **end** oralig’i) buyruqlar bajariladi.

2.2-rasmda if-tnen-else buyrug’iga mos keluvchi algoritmi keltirilgan.



2.2-rasm. If-Then-Else

Masalan, **t** elektr zanjiriga qarshilikning ulanish tipini ko’rsatsa (t=1 ketma-ket, t=2 paralel ulanishga mos kelsa), **r1** va **r2** esa qarshiliklarning qiymati bo’lsa, u holda shartning ko’rinishi quyidagicha bo’ladi:

```
if t=l then
begin
z:=r1+r2;
end
else
begin
z:=(r1+r2)/(r1*r2);
end;
```

Agar **if** buyrug’ida **begin** va **end** orasida faqat bitta buyruq bo’lsa, u holda **begin** va **end** so’zlarini yozmasa ham bo’ladi.

Masalan:

```
if javob=3
then
begin
togri:=togri+1 ;
end
else
begin
ShowMessage('Xato!');
end;
```

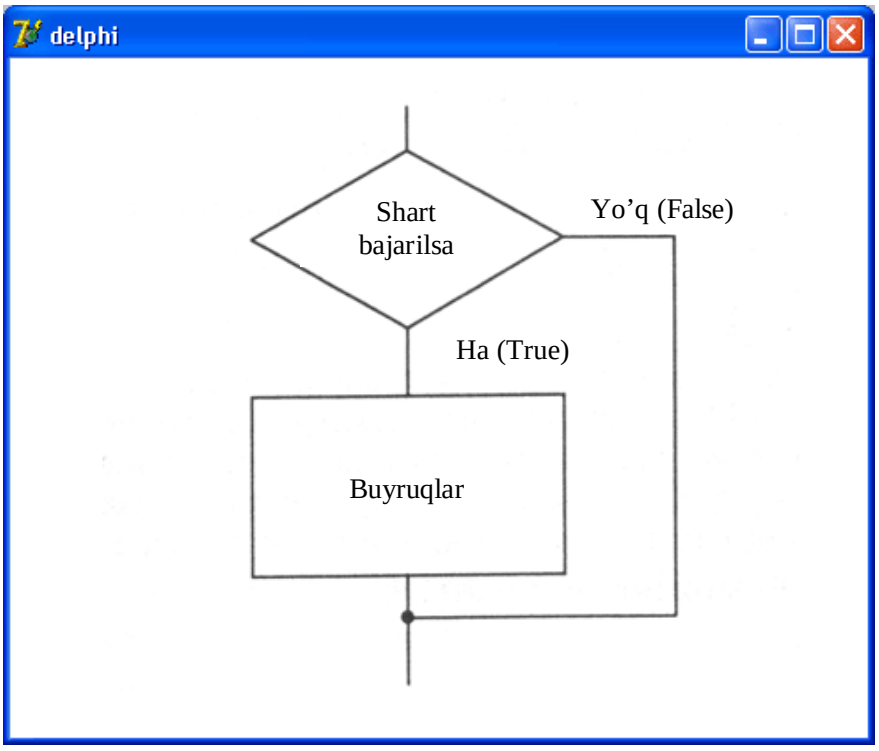
shartini quyidagicha yozsa ham bo’ladi:

```
if javob=3 then
togri:=togri+1 ;
else
ShowMessage('Xato!') ;
```

Agar biror bir buyruq faqat qo’yilgan shart rost bo’lganda bajarilishi kerak bo’lsa, u holda shart quyidagicha yoziladi:

```
if shart then
begin
{ Shart rost bo’lsa bajariladigan buyruqlar }
end
```

2.3-rasmda if-then buyrug’iga mos algoritm keltirilgan.



2.3-rasm . If-Then

```
if n=m
then c:=c+1;
```

Agar n va m teng bo’lsa c o’zgaruvchisinig qiymati ortadi.

If buyrug’ini ishlatishga misol sifatida xalqaro telefon so’zlashuvining qiymatini hisoblovchi dasturni ko’rib o’tamiz.

Ma’lumki, telefonda xalqaro so’zlashuv dam olish kunlari boshqa kunlarga nisbatan arzon bo’ladi. 2.1. listingda keltirilgan dastur so’zlashuv vaqtini va hafta kunini so’raydi, keyin esa so’zlashuv qiymatni hisoblaydi. Agar hafta kuni shanba yoki yakshanba bo’lsa, u holda so’zlashuv qiymati chegirma bilan hisoblanadi. 1 minut so’zlashuv narxi va chegirma qiymati dastur matnida kostanta sifatida beriladi. 2.4-rasmda dastur muloqot oynasining ko’rinishi keltirilgan.

Boshlang’ich ma’lumotlarni kiritish uchun (so’zlashuv vaqti, hafta kunining raqami) tahrirlash maydonidan, hamda natija va tushuntiruvchi matn chiqarish uchun chiqarish maydonidan foydalaniladi. 2.3-jadvalda komponentalar va ularning vazifasi ko’rsatilgan. 2.4 jadvalda esa ushbu komponentlarning hususiyat qiymatlari ko’rsatilgan.

2.4. So'zlashuv qiymati dasturining muloqot oynasi.

Eslatma

Bu yerda va bundan keyin formada ishlatiladigan komponentlarning faqat o’zgartiriladigan hususiyatlari beriladi. Qolgan hususiyatlar o’zgartirishsiz qoldirilishi mumkin yoki ixtiyoriy tarzda o’zgartirilishi mumkin.

2.3-jadval

Komponent	Vazifasi
Edit1	So’zlashuv vaqtini (minutda) kiritish maydoni
Edit2	Hafta kunining nomerini kiritish maydoni
Label1, Label2	Kiritish maydonlarining vazifasi to’g’risida ma’lumot chiqarish maydonlari.
Label3	So’zlashuv qiymatini hisoblash natijasini chiqarish maydoni.
Button1	So’zashuv qiymatini hisoblash protsedurasini ishga tushirish tugmasi.

Eslatma

Forma komponentlarinining hususiyatlarini o’zida mujassamlashtirgan jadvalda avval komponent nomi so’ngra nuqtadan keyin hususiyatnomi berilgan.

2.4.-jadval.

Hususiyat	Qiymati
Form1.Caption	So’zlashuv qiymat
Edit1.Text	
Edit2.Text	
Label1.Caption	So'zlashuv vaqti (min)
Label2.Caption	Hafta kunining raqami
Label3.Caption	
Button1. Caption	Hisoblash

Dastur **Hisoblash** buyruq tugmasini bosish natijasida hisoblashni amalga oshiradi. Bunda **OnClick** hodisasi yuz beradi va TForm1.Button1Click protsedurasida qayta ishlanadi.

2.1.-Listing Telefonda so’zlashuv qiymatini hisoblash.

```
unit Tel_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    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
PAY =150; // 1 minut so'zlashuv qiymati 150 so'm
```

```
DISCOUNT = 0.2; //  20 foiz chegirma
    var
    Time:Real; // So'zlashuv vaqti
    Day:integer; // hafta kuni
    Summa:real; // so'zlashuv qiymati
    begin
    // boshlang'ich ma'lumotlarni olish
    Time:=StrToFloat(Edit1.Text) ;
    Day:=StrToInt(Edit2.Text);
    // Hisoblash
    Summa:= PAY*Time;
    // Agar shanba yoki yakshanba bo'lsa chegirma olish
    if (Day = 6) OR (Day = 7)
    then Summa:=Summa*(1 - DISCOUNT);
    // Hisoblash natijasini chiqarish
    label3.caption:='To`lashga '
    + FloatToStr(Summa) + ' so`m.';
    end;
    end.
```

Ko’pincha dasturda ikkitadan ortiq variantdan birini tanlashni tashkil qilish kerak. Masalan, ma’lumki, har bir odamning optimal vazni mavjud va quyidagi formula bilan hisoblanadi:

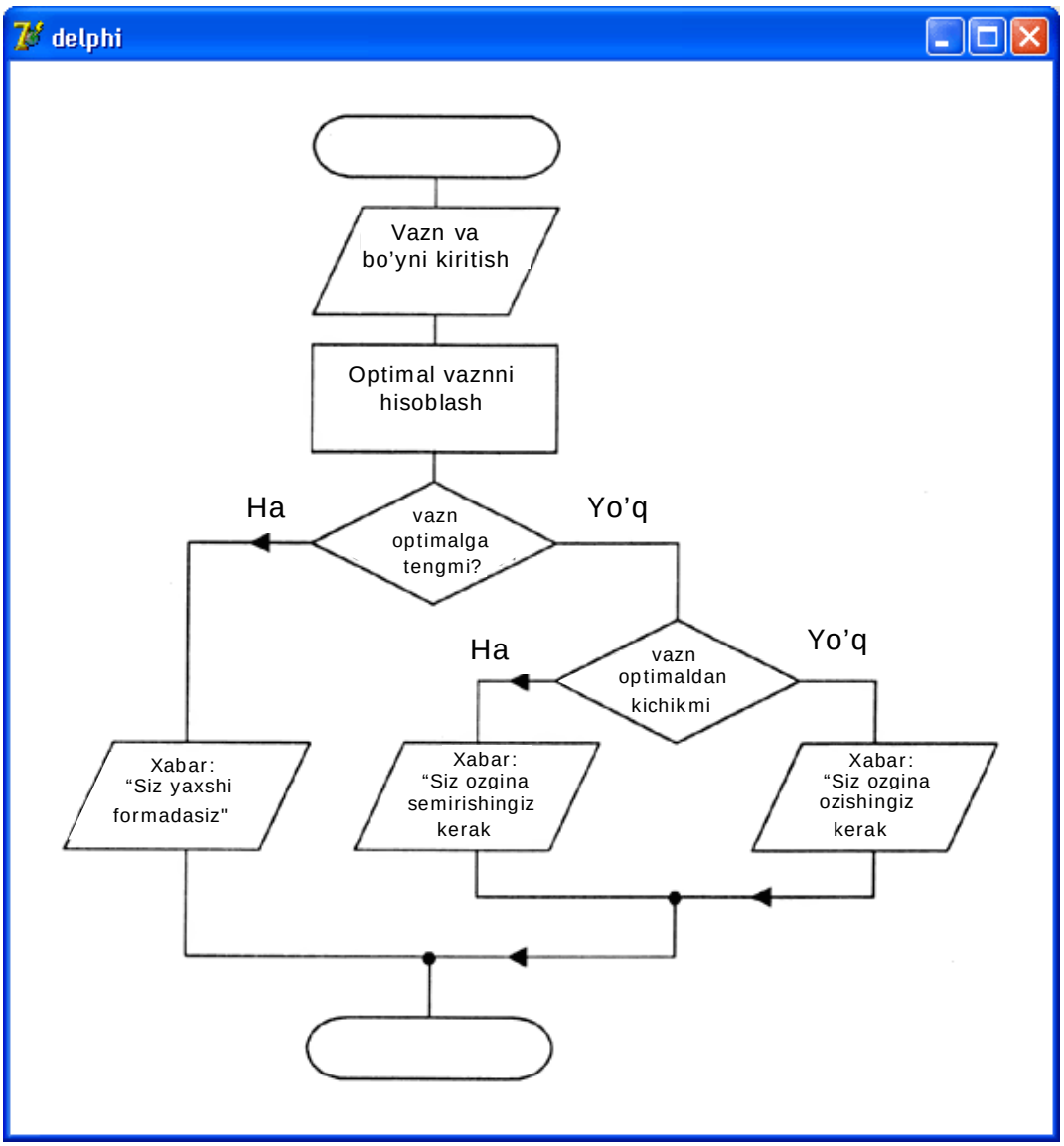
Bo’yi(sm)- 100.

Haqiqiy vazn optimal vazndan farq qilishi mumkin: kam, teng yoki ko’p bo’lishi mumkin.

2.5 rasmda muloqot oynasi keltirilgan navbatdagi dastur, vazn va bo’yni so’raydi, optimal qiymatni hisoblaydi, uni haqiqiy vazn bilan solishtiradi va mos xabarni chiqaradi.

2.5.-rasm. Vaznni nazorat qilish dasturining oynasi

2.6.-rasmda **Vaznni nazorat qilish** dasturining algoritmi keltirilgan.



2.6.-rasm.

2.2. Vaznni nazorat qilish

```
unit Vazn_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
```

```
{ Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
w:real; { vazn } h:real; { bo'y }
opt:real;{ optimal vazn }
d:real;
begin
w:=StrToFloat(Edit1.text);
h:=StrToInt(Edit2.Text);
opt:=h-100;
if w=opt then
        Label3.caption:='Siz yaxshi formadasiz!'
else
        if w < opt then
                begin
                        d:=opt-w;
                        Label3.caption:='Siz '
                        + FloatToStr(d)+ ' kg. ga semirishingiz kerak.';
                end
        else
                begin
                        d:=w-opt;
                        Label3.caption:='Siz'
                        + FloatTostr(d)+ ' kg. ga ozishingiz kerak.';
                end;
        end;
end.
```

Case buyrug’i

Avvalgi misolda ko’p variantli tanlash ichma-ich joylashgan ikkita **if** buyrug’idan foydalanib ammalga oshirilgan. Bunday yondoshish doimo qulay emas. Ayniqsa dasturda tanlash varianti ko’p bo’lsa.

Delphi tilida ko’p variantlardan tanlash uchun **Case** buyrug’idan foydalaniladi. Uning umumiy ko’rinishi quyidagicha:

```
case tanlovchi of
ro’yhat1:
        begin
                { buyruqlar 1 }
        end;
ro’yhat2:
        begin
                { buyruqlar 2 }
        end;
.....
 ro’yhatN:
        begin
                { buyruqlar N }
        end;
else
        begin
                { buyruq )
        end;
end;
```

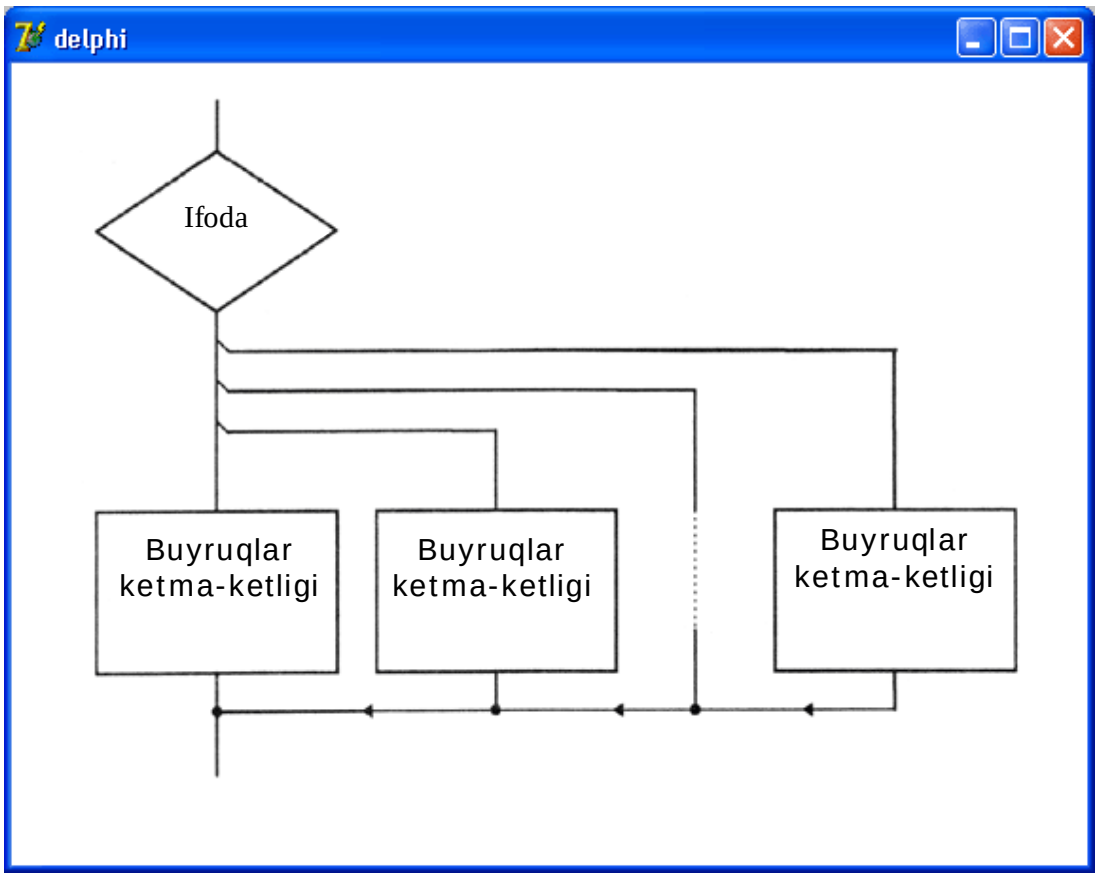
bu yerda:

- **tanlovchi** — dasturning keyingi bajariladigan qadamini aniqlovchi qiymatni beruvchi ifoda.
- **Ro’yhat N** — konstantalar ro’yhat. Agar konstantalar sonlar diapazonidan iborat bo’lsa, u holda ro’yhat o’rniga birinchi va ohirgi kostantalarni orasiga ikkita nuqta qo’yib berish mumkin. Masalan, 1,2,3,4,5,6 ro’yhat 1..6 ko’rinishida yozilishi mumkin.

Case buyrug’i quyidagicha ishlaydi:

1. Dastlab **tanlovchi** ifodaning qiymatini hisoblaydi.
2. **tanlovchi** ifodaning qiymati ketma-ket kostantalar ro’yhatidagi kostantalar bilan solishtiriladi.
3. Agar ifoda qiymati ro’yhadagi konstanta bilan mos tushsa, u holda shu ro’yhatga mos buyruqlar bajariladi. Shu bilan **case** buyrug’i ishi yakunlanadi.
4. Agar ifoda qiymati ro’yhatdagi kostantalarda birortasiga ham mos kelmasa, u holda **else** buyrug’idagan keyingi buyruqlar ketma-ketligi bajariladi.

Agar else yozilmasa u holda case bajarilmasa case dan keyingi buyruqlar bajariladi.
2.7-rasmda **case** buyrug’ining ishlash algoritmi keltirilgan.



2.7.-rasm.

Quyida **case** buyrug’iga misol keltirilgan.

```
case n_day of
1,2,3,4,5: day:='Ish kuni. ' ;
6: day:='Shanba!';
7: day:='Yakshanba!';
end;
case n_day of
1..5: day:='Ish kuni.';
6: day:='Shanba!';
7: day:='Yakshanba!';
end;
case n_day of
6: day:='Shanba!';
7: day:='Yakshanba!';
else day:='Ish kuni.';
end;
```

case operatorini qo’llashga misol sifatida og’irlikni funtdan kilogrammga o’tkazuvchi dasturni ko’rib o’tamiz. Dasturda turli mamlakatlarda funtning hisoblashni turlicha bo’lishi ham hisobga olinadi. Masalan, Rossiyada bir funt 409,5 gramm, Angliyada 453,592 gramm, Germaniya, Daniya va Islandiyada esa 500 grammga teng.

2.8-rasmda tasvirlangan dasturning muloqot oynasida mamlakatlarni tanlash uchun **Mamlakat** ro’yxatidan foydalaniladi.

2.8.-rasm

Mamlakat nomini tanlash uchun **ListBox** ro'yhat chiqarish komponenti ishlatiladi. **ListBox** komponentining belgisi **Standart** kimponentlar bo'limida joylashgan (2.9-rasm). 2.5-jadvalda **ListBox** komponentining hususiyatlari berilgan

2.9.-rasm.

2.5.-jadval

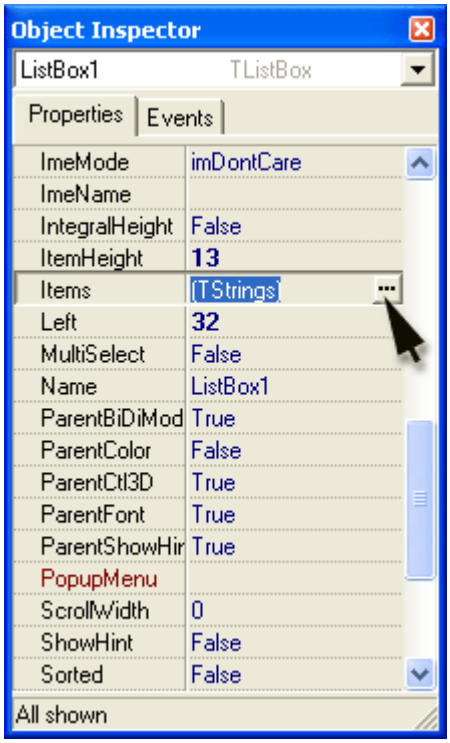
Hususiyat	**Aniqlaydi**
Name	Komponent nomi. Dasturda komponent hususiyatlariga murojaat etish uchun ishlatiladi.
Items	Ro'yhat element
Itemindex	Tanlangan ro'yhat elementining raqami.Ro'yhatdagi birinchi elementning raqami
Left	Komponentning chap chegarasidan formaning chap chegarasigacha bo'lgan masofa.

Top	Komponentning yuqori chegarasidan formaning yuqori chegarasigacha bo'lgan masofa.
Height	Ro'yhat maydonining balandligi
Width	Ro'yhat maydonining kengligi
Font	Ro'yhat elementlarini tasvirlash uchun ishlatiladigan shrift
Parent-Font	Formaning shriftini olish.

Bu erda asosiy e'tiborni Items va Itemindex hususiyatlariga berish kerak. Items hususiyati ro'yhat elementlaridan tashkil topgan bo'ladi. Itemindex hususiyati tanlangan ro'yhat elementining raqamini beradi. Agar bitta ham element tanlanmagan bo'lsa, u holda bu hususiyatning qiymati -1 ga teng bo'ladi.

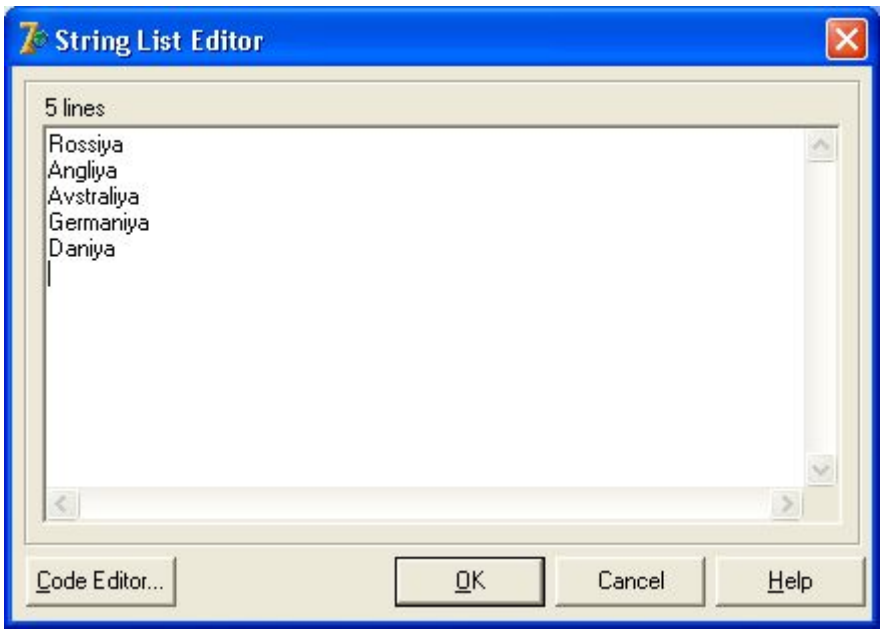
Ro'yhatni forma yaratilayotgan yoki dastur ishlayotgan vaqtda tuzish mumkin.

Forma yaratilayotgan vaqtda ro'yhat tuzish uchun Object Inspector oynasida Items hususiyatini tanlab, ro'yhat qatorlarini tahrirlash oynasini ishga tushiruvchi tugmani bosing (2.10-rasm)



2.10.-rasm

Ochilgan **String List Editor** muloqot oynasida (2.11-rasm) ro'yhatni kiritish kerak. Bunda ro'yhatning har bir elementi alohida qatorga yozilishi kerak. Yahgi qatorga o'tish uchun <Enter> tugmasidan foydalaniladi.



2.11.-rasm. Ro'yhat tahrirlovchi oyna.

2.6.-jadvalda ilova formasining komponentlari keltirilgan, 2.7-jadvalda esa ushbu komponentlarning hususiyat qiymatlari berilgan.

2.6.-jadval Formaning komponentlari.

Komponent	**Vazifasi**
ListBox1	Mamlakatni tanlash uchun
Edit1	Og'irlikni funtda kiritish uchun
Label1, Label2, Label3	Kiritish maydonlarining fazifasini ko'rsatuvchi tushuntirish matnlarni chiqarish uchun
Label4	Natijani chiqarish uchun
Button1	Funtda kilogrammga o'tkazuvchi procedurani ishlatish uchun

2.7.-jadval

Hususiyat	**Qiymat**
Form1 .Caption	**Case** dan foydalanishga misol
Edit1. Text	
Label1 . Caption	Mamlakatni tanlang
Label2 .Caption	Mamlakat
Label3 . Caption	Funt
Button1 . Caption	Hisoblash

Hisoblash buyruq tugmasi bosilganda hisoblash protsedurasi funtdagi og'irlikni kilogrammga o'tkazish koeffisientiga ko'paytiradi. Koeffisient qiymati ro'yhatdan tanlangan element raqami bo'yicha aniqlanadi.
2.3-listingda dastur matni keltirilgan.

2.3.-listing

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button1: TButton;
    Edit1: TEdit;
    ListBox1: TListBox;    //Ro'yhat
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
{
ListBox1.items.add('Rossiya');
ListBox1.items.add('Angliya');
ListBox1.iterns.add('Avstraliya');
ListBox1.items.add('Germaniya');
ListBox1.iterns.add ('Daniya');
ListBox1.iterns.add ('Italiya');
 ListBox1.iterns.add ('Gollandiya');
```

```
}
    ListBox1.itemindex:=0;
    end;

    procedure TForm1.Button1Click(Sender: TObject);
    var
    funt:real; // funtdagi og'irlik
    kg:real; // kg dagi og'irlik
    k:real; // qayta hisoblash koeffisienti
    begin
    case ListBox1.Itemindex of
    0: k:=0.4095; // Rossiya
    1: k:=0.453592; // Angliya
    2: k:=0.56001; // Avstraliya
    3..4,6:k:=0.5; // Germaniya, Daniya, Gollandiya
    5: k:=0.31762; // Italiya
    end;
    funt:=StrToFloat(Edit1.Text);
    kg:=k*funt;
    label4.caption:=Edit1.Text
    + ' f. - bu '
    + FloatToStrF(kg,ffFixed, 6,3) + 'kg.';
    end;
    end.
```

Forma yaratilayotgan vaqtda sodir bo'ladigan **FormCreate** hodisasini qayta ishlash protsedurasiga e'tibor bering. Bu protsedurani dasturdagi o'zgaruvchilarni e'lon qilish uchun ishlatish mumkin. Bizning misolda ro'yhatga element qo'shish uchun ishlatilgan, ammo izoxga aylantirib qoyilgan. Chunki biz komponentaning **Items** hususiyatidanfoydalanib ro'yhatni kiritib qo'yganmiz.

Navbatdagi misolda uchta o'zgaruvchi, ya'ni **day**(kun), **month** (oy) va **year** (yil) bilan ifodalangan bugungi kunning sanasi bo'yicha keyingi kunning sanasini aniqlovchi dasturni (2.5-listing) ko'rib chiqamiz. Dasturning oyna ko'rinishi 2.12-rasm ko'rsatilgan.

2.12-rasm.

Dastlab **case** buyrug'i yordamida joriy kun oyning ohirgi kuni yoki ohirgi kuni emasligi tekshirib chiqiladi. Agar joriy oy - fevral va joriy kunning sansi - 28 bo'lsa qushimcha ravishda yil kabisa yil yoki yil emasligi tekshiriladi. Buning uchun yilni 4

ga bo’lgandagi qoldiq hisoblanadi. Agar qoldiq nolga teng bo’lsa, u holda kabisa yili bo’ladi va oyning ohirgi kuni 29 bo’ladi.

Agar joriy kun ohirgi kun bo’lsa, u holda keyingi kuni birinchi bo’ladi. So’ngra joriy oy dekabrligi tekshiriladi. Agar **yo’q** bo’lsa, u holda oyning raqami bittaga oshiriladi, agarda **ha** bo’lsa, u holda yilning raqami bittaga oshiriladi va oyning raqami 1 bo’ladi.

2.5. -listing

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    Label4: TLabel;
    Label5: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Day,Month,Year:word;
implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var sana:TDateTime;
begin
sana:=Date;
//DecodeDate sanani yil,oy va kunga bo'ladi
DecodeDate(sana,Year,Month,Day);
```

```
Edit1.Text:=inttostr(Day);
        Edit2.Text:=inttostr(Month);
        Edit3.Text:=inttostr(Year);
        end;
        // Keyingi kunni hisoblash
        procedure TForm1.Button1Click(Sender: TObject);
        var
        last:boolean; // oyning ohirgi kuni bo'lsa,
        // u holda last = True
        r:integer; // agar yil kabisa bo`lsa,
        //year 4 ga boolganda r nolga teng bo`ladi.
        begin
        Day:=StrToInt(Edit1.Text);
        Month:=StrToInt(Edit2.Text);
        Year:=StrToInt(Edit3.Text);
        last := False; // kun oyning ohirgi kuni emas.
        case month of
        4,6,9,11: if day =31 then last:= True;
        2:if day = 28 then
                begin
                  r:= year mod 4;
                  if r <> 0 then last:= True;
                end
          else if day=29 then last:= True;
        else
        if day=31 then last:= True;
        end;//Case tugashi
        if last then // ohirgi kun bo'lsa
           begin
              day:=1;
              if month =12 then // ohirgi oy bo'lsa
                 begin
                   month:= 1;
                   year:= year + 1;
                 end
              else month:= month + 1;
           end
        else day:= day + 1;
        Edit1.Text:=inttostr(Day);
        Edit2.Text:=inttostr(Month);
        Edit3.Text:=inttostr(Year);
        end;
        end.
```

Takrorlanish

Juda ko’p masalalarning yechilish algoritmi takrorlanuvchan bo’ladi. Ya’ni natijaga erishish uchun bir xildagi buyruqlar ketma -ketligi bir necha marta bajarilishi kerak.

Masalan, bilimni nazorat qiluvchi dastur savolni beradi, javobni qabul qiladi, javobga qarab ballar yig’indisiga bal qo’shadi, so’ngra shu ishlarni qayta va qayta barcha savollar tugamagunicha bajaradi.

Boshqa misol. Ro’yhatdan kerakli odamning familiyasini topish uchun, ro’yhatdagi birinchi familiyani, so’ngra ikkinchisini, uchinchisini va hokozo toki kerakli familiya topilmaguncha yoki ro’yhat tugamaguncha tekshirish kerak.

Bir nacha marta bajarilishi kerak bo’lgan buyruqlar ketma-ketligi bor algoritm takrorlanuvchi deb ataladi, buyruqlar ketma-ketligi esa takrorlanish deb ataladi.

Takrorlanish dasturda For, While, va Repeat buyruqlaridan foydalanib amalga oshirish mumkin.

For buyrug’i

Quyidagi masalani ko’rib chiqamiz. -1,-0.5, 0, 0.5, va 1 nuqtalarda $y=5x^2-7$ funksiyaning qiymatini hisoblovchi dastur tuzish kerak bo’lsin.

Quyilgan masalaning yechilishini ta’minlovchi protsedura quyidagicha bo’lishi mumkin:

```
procedure TForm1.Button1Click(Sender: TObject);
var
y: real; // funksiyaning qiymati
x: real; // funksiyaning argumenti
dx: real; // ortish qiymati
st: string; // Javob jadval ko’rinishida
begin
st:='';
x := -1; dx := 0.5;
y := 5*x*x -7;
st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
x :=x + dx;
y := 5*x*x -7;
st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
x :=x + dx;
y := 5*x*x -7;
st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
x :=x + dx;
y := 5*x*x -7;
st := st+ FloatToStr(x)+' ' + FloatToStr(y)+chr(13);
x :=x + dx;
y := 5*x*x -7;
st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
x :=x + dx;
Label1.Caption := st;
```

end;

Dastur matnidan ko’rinib turibdiki, quyidagi buyruqlar ketma-ketligi 5 marta bajarilyapti.

```
y := 5*x*x -7;
st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
x :=x + dx;
```

For buyrug’idan foydalanib, yuqoridagi protsedurani quyidagicha yozish mumkin:

```
procedure TForm1.Button1Click(Sender: TObject);
var
y: real; // funksiyaning qiymati
x: real; // funksiyaning argumenti
dx: real; // ortish qiymati
st: string; // Javob jadval ko’rinishida
i : integer; // takrorlanishni hisoblagich
begin
st:=''; x := -1; dx := 0.5;
for i:=1 to 5 do
begin
    y := 5*x*x -7;
    st := st+ FloatToStr(x)+' '+ FloatToStr(y)+chr(13);
    x :=x + dx;
end;
Label1.Caption := st;
end;
```

Protseduraning ikkinchi variantida buyruqlar ketma-ketligi kam yozib, protsedura juda ixcham ko’rinishga keladi.

For operatoridan agar takrorlanishlar soni oldindan ma’lum bo’lsagina foydalaniladi.

For operatorining umumiy ko’rinishi quyidagicha bo’ladi:

```
for hisoblagich := boshl_qiymat to ohirgi_qiymat do
begin
    // bu yerda bir necha marta bajarilishi
    // kerak bo’lgan buyruqlar joylashadi
end
```

bu yerda:

- hisoblagich —takrorlanishlar sonini hisoblovchi o’zgaruvchi;
- boshl_qiymat - hisoblagichning boshlang’ich qiymatini aniqlovchi ifoda;
- ohirgi_qiymat — hisoblagichning ohirgi qiymatini aniqlovchi ifoda.

Boshl_qiymat va ohirgi_qiymat hisoblagich qiymatlari butun sonlardan iborat bo’lishi kerak.

Takrorlash operatorining takrorlashlar sonini quyidagi formula bilan aniqlanadi:

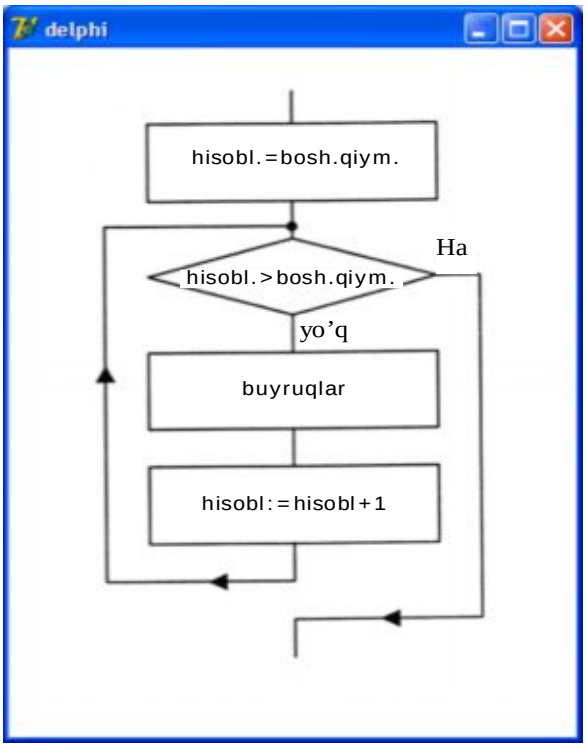
ohirgi_qiymat - boshl_qiymat + 1.

misollar:

```
for i:=l to 10 do begin
label1.caption:=label1.caption + '*'; end;
for i: =1 to n do s := s+i;
```

eslatma

Agar begin va end operatorpari orasida faqat bitta buyruq bo'lsa, u holda begin va end larni yozmaslik mumkin.



2.13-rasm. for operatori algoritmi

For operatoriga mos algoritm 2.13-rasm keltirilgan. E'tibor bering, agar hisoblagichning boshlang'ich qiymati ohirgi qiymatidan katta bo'lsa **begin** va **end** operatorlari oralig'ida yotgan buyruqlar biror marta ham bajarilmaydi.

Undan tashqari, takrorlashning tana qismi har bir bajarilishida hisoblagichning qiymati avtomatik tarzda oshib boradi.

Takrorlashning o'zgaruvchi hisoblagichidan uning tana qismida foydalanish mumkin (uning qiymatini o'zgartirmaslik kerak). Masalan, quyidagi dastur qismi bajarilishi natijasida tab1 o'zgaruvchining qiymati kvadratlar jadvalini hosil qiladi:

```
tab1: = '' ;
for i:=l to 5 do
begin
tab1:=tabl+IntToStr(i)+' '+IntToStr(i*i)+chr(13);
end;
```

1+1/2+1/3+... ketma-ketlikning dastlabki 10 ta hadining yig'indisini hisoblash dasturini ko'rib chiqaylik (ketma-ketlikning i-chi elementining qiymati 1/i formula

bilan aniqlanadi). Dasturning muloqot oynasida ikkita komponent bo'lishi kerak: belgilar maydoni (Label1) va buyruqlar tugmasi (Button1).

Ketma-ketlikning yig'indisini hisoblash va natijani ekranga chiqarishni OnClick hodisa protsedurasi amalga oshiradi, dastur matni quyida keltirilgan.

```
procedure TForm1.ButtonlClick(Sender: TObject);
var
i:integer; { qator elementining raqami }
elem:real;
{qator elementining qiymati }
summ:real;
{ qator elementining yig'indisi )
begin
summ:=0;
label1caption:= ' ' ;
for i:=1 to 10 do begin
elem:=1/i;
label1.caption:=label1.caption+
IntToStr(i)+' '+FloatToStr(elem)+#13;
summ: =summ+elem;
end;
label1.caption:=label1.caption+'Qator yigindisi:'+FloatToStr(summ);
end;
```

agar for operatoridagi to so'zining o'rniga downto yozilsa, u holda hisoblagichning navbatdagi qiymati oshmaydi, balki kamayib boradi.

while operatori

Agar takrorlanishlar soni oldindan belgilanmay, dastur ishi davomida aniqlansa **while** takrorlash operatoridan foydalaniladi.

While takrorlash operatoriga misol qilib hisoblashni berilgan aniqlikda bajarishni keltirish mumkin.

While operatori umumiy holda quyidagicha ko'rinishda yoziladi:

```
while shart do
begin
    // bir necha marta bajarilishi lozim bo'lgan
    // buyruqlar ketma-ketligi
end
```

bu yerda shart — maniqiy ifoda bo'lib, takrorlash davom etish yoki etmaslikni aniqlaydi.

While operatori quyidagica ishlaydi:

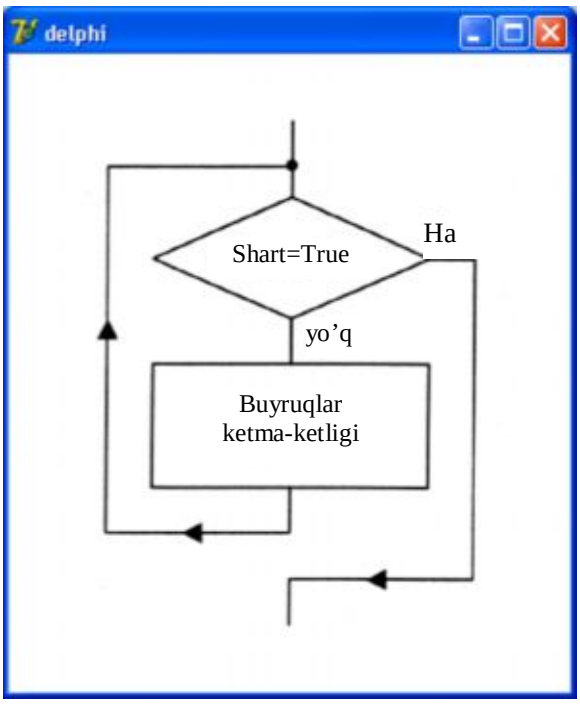
1. Avval shartni ifodalovchi qiymat tekshiriladi.
2. Agar tekshhirilgan qiymat **false** (shart bajarilmasa) ga teng bo'lsa, u holda while operatorining ishi yakunlanadi.
3. Agar tekshhirilgan qiymat **True** (shart bajarilsa) ga teng bo'lsa, u holda **begin** va **end** operatorlari orasida joylashgan operatorlar bajariladi. Bundan so'ng

shart yana tekshiriladi. Agar shart yana bajarilsa, u holda takrorlashning tanasi yana bajariladi va hokazo. Bu takrorlash toki shart bajarilmay qolgunigacha (**False**) davom etadi.

While operatoriga mos algoritm 2.14-rasmda keltirilgan.

Diqqat!

While takrorlanuvchi buyrug'ida **Begin** va **End** orasida joylashgan buyrug'lar xech bo'lmaganda bir marta bajarilishi uchun **While** buyrug'idagi shart rost bo'lishi kerak.



2.14-rasm. While operatorining algoritmi

Dastur ishlagan vaqtda foydalanuvchi tomonidan berilgan aniqlikda π soninng qiymatini ahiqlovchi dasturni ko'rib chiqaylik. Hisoblash algoritmi quyidagicha asoslanadi:

- 1 - 1/3 + 1/5 -1/7 + 1/9 + ... qatorning yig'indisi qator a'zolari soni etarlicha katta bo'lganda π/4 qiymatiga yaqinlashadi.
- Qatorning xar bir **n** - a'zosi **1/(2*n - 1)** formula bilan aniqlanadi va **n** juft bo'lganda **-1** ga ko'paytiriladi.
- Hisoblash qatorning navbatdagi a'zosining qiymati foydalanuvchi tomonidan berilgan aniqlik koeffisientidan kichik bo'lganda to'xtatiladi.

Dastur ishlagan vaqtdagi muloqot oyna ko'rinishi 2.15-rasmda ko'rsatilgan. Foydalanuvchi kiritish maydoniga (Edit1) aniqlik koeffisientini kiritadi. **Hisoblash** buyruq tugmasini (Button1) bosilganda π sonining qiymati hisoblanib natija chiqarish maydoniga (Label2) chiqariladi. 2.6.-listongda dasturning matni keltirilgan. Huddi avvalgidek asosiy ishni **Onclick** hodisasini qayta ishlash protsedurasi bajaradi.

2.15.-rasm.

2. 6. - listing

```
unit pi_;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);

var
   pi:real;      // PI sonining hisoblangan qiymati
   t:real;       // hisoblashdagi aniqlik
   n:integer;    // qator xadining raqami
   elem:real;    // qator xadining qiymati
begin
```

```
pi:=0;
      n:=1;

      t:=StrToFloat(edit1.text);
      elem:=1; // takrorlanishni boshlash uchun
      while elem >= t do
        begin
          elem:=1/(2*n-1);
     if n MOD 2 = 0
            then pi:=pi-elem
       else pi:=pi+elem;
     n:=n+1;
   end;
      pi:=pi*4;
      label1.caption:= 'PI '+ FloatToStr(pi) + ' ga teng'+ #13
          + 'Qator xadlarining soni '+IntTostr(n);
   end;
   end.
```

Repeat buyrug’i

Repeat buyrug’i huddi **While** buyrug’i kabi ishlaydi. Ya’ni takrorlanishlar soni oldindan belgilanmaydi, aksincha dastur ishi davomida aniqlanadi.

Repeat buyrug’ining umumiy ko’rinishi quyidagicha:

repeat
// Buyruqlar
until Shart

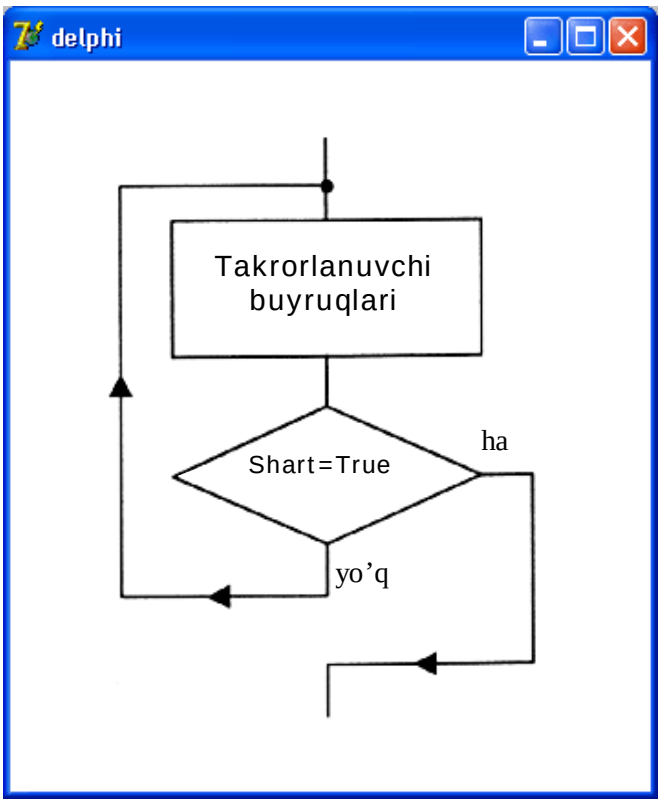
bu erda shart - mantiqiy tipdagi ifoda bo’lib, takrorlanishning tugash shartini aniqlaydi.

Repeat buyrug’i quyidagicha ishlaydi:

1. Dastlab repeat va until orasidagi buyruqlar bajariladi.
2. Keyin shartning qiymati aniqlanadi. Agar shart yolg’on bo’lsa (False), u holda takrorlanishning tanasi yana bir marta bajariladi.
3. Agar shart rost bo’lsa (True), u holda takrorlanish yakunlanadi.

Sunday qilib, repeat va until orasida joylashgan buyruqlar, shart rost qiymat qabul qilguncha bajarilaveradi.

Repeat buyrug’iga mos keluvchi algoritm 2.16-rasmda keltirilgan.



2.16-rasm.

Diqqat!

Repeat va until orasidagi buyruqlar hech bo'lmaganda bir marta bajariladi. Takrorlanish yakunlanishi uchun **repeat** va **until** orasida joylashgan buyruqlar shartli ifodadagi o'zgaruvchining qiymatini o'zgartirishi zarur.

Repeat buyrug'ini qo'llashga misol sifatida foydalanuvchi kiritgan son tub ekanligini aniqlovchi dasturni ko'rib o'tamiz.

Buning uchun n sonini ikkiga, uchga..., n gacha bo'lib, har bir bo'lishdan keyingi qoldiqni tekshirib boriladi. Agar navbatdagi bo'lishning qoldig'i nolga teng bo'lsa, u holda n ni qoldiqsiz bo'ladigan son topilganini bildiradi. N va n ni qoldiqsiz bo'lgan sonni solishtirib n soni tub yoki tub emasligi aniqlanadi.

Tub son ilovasining forma ko'rinishi 2.17-rasmda a uning dastur matni 2.7-listingda keltirilgan.

2.17-rasm.

2.7-listing.

```
unit simple_;
```

```
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
TForm1 = class(TForm)
Button1: TButton; // Tekshirish tugmasi
Label1: TLabel;
Edit1: TEdit; // son kiritish maydoni
Label2: TLabe1; // natijani chiqarish maydoni
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject) ;
var
n: integer; // tekshiriluvchi son
d: integer; // bo'linuvchi
r: integer; // n ni d ga bo'lgandagi qoldiq
begin
n:=StrToInt(Edit1.text);
d := 2; // avval 2 ga bo'lamiz
repeat
r := n mod d;
if r <> 0 // n d ga butun bo'linmasa
then d := d + 1;
until r = 0; // n qoldiqsiz bo'linuvchi son topildi
label2.caption:=Edit1.text;
if d = n then Iabel2.caption:=label2.caption + ' — tub son.'
else label2.caption:=label2.caption + ' — murakkab son.';
end;
end.
```

Goto buyrug'i

If va **case** buyruqlari biron-bir shart asosida dasturning navbatdagi buyruqlariga o'tish uchun ishlatiladi. Shuning uchun ularni ba'zan shartli o'tish operatorlari deb ham ataladi.

Bu buyruqlar qatorida dasturning bajarilish qadamini boshqarish uchun shartsiz o'tish operatori **goto** dan ham foydalanish mumkin.

Goto ning umumiy ko'rinishi quyidagicha:

goto belgi;

bu yerda belgi - goto buyrug’idan so’ng dasturning bajarilishi kerak bo’lgan buyruqdan oldin yoziladigan identifikator.

Goto buyrug’ida ishlatiluvchi belgi dasturning belgilar bo’limida e’lon qilinishi shart. Belgilarni e’lon qilish Label so’zi bilan boshlanib, o’zgaruvchilarni e’lon qilish bo’limidan oldin joylashadi.

Dasturda **goto** buyrug’i bajarilishi natijasida o’tish kerak bo’lgan buyruqdan oldinga belgi qo’yilib, buyruq va belgi orasiga ikki nuqta qo’yiladi.

2.8-listingda sonni tekshirish protsedurasining yana bir varianti keltirilgan. Bunda agar foydalanuvchi noto’g’ri ma’lumot kiritsa protseduraning ishini **goto** buyrug’idan foydalanib yakunlanadi.

2.8.-listing

```
procedure TForm1.Button1Click(Sender: TObject);
label // belgini e’lon qilish bo’limi
bye;
var
n: integer; // tekshiriluvchi son
d: integer; // bo’luvchi
r: integer; // n ni d ga bo’lgandankeyingi qoldiq
begin
n:=StrToInt(Edit1.text);
if n <= 0 then begin
MessageDlg('Sonni noldan katta bo’lishi kerak.', mtError, [mbOk] , 0) ;
Edit1.text:= "";
goto bye;
end;
// musbat son kiritilsa
d:= 2; // Dastlab 2 ga bo’lamiz
repeat
r:= n mod d;
if r <> 0 // n d ga qoldiqsiz bo’linmadi
then d:= d + 1;
until r = 0;
label2.caption:=Edit1.text;
if d = n
then label2.caption:=label2.caption
+ ' — tub son.'
else label2.caption:=label2.caption
+' — murakkab son.';
bye:
end;
```

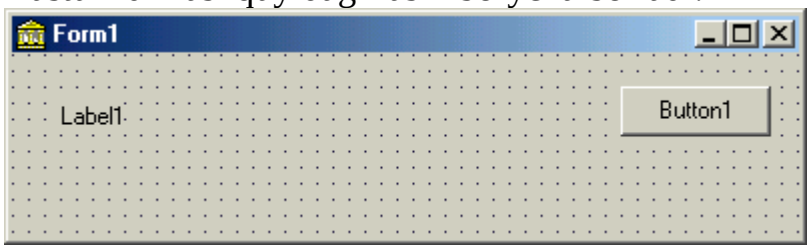
Dasturda chalkashliklar kelib chiqishini oldini olish maqsadida ba’zi bir adabiyotlarda **goto** buyrug’idan foydalanilmaslikni tavsiya etiladi. Biroq ba’zi

xolatlarda **goto** buyrug’idan foydalanish mumkin. Yuqoridagi misol huddi shunday xolatga kiradi.

Belgilar va satrlar

Belgili tip ***Char*** xizmatchi so’zi bilan e’lon qilinib, bu tipning qiymatlari kompyuter xotirasidan 1 bayt joy egallaydi. Tilning barcha belgilari bu tipning qiymatlar sohasiga tegishlidir. Belgili qiymatni qo’shtirnoq ichiga olib yoki # belgisidan keyin kerakli belgining ANSI kodini yozib aniqlash mumkin.
Misol: ‘A’ , yoki # 60.

Misol sifatida 32 dan 255 gacha bo’lgan ANSI kodlarda joylashgan belgilar ketma-ketligini quyidagi dastur yordamida ko’rishimiz mumkin:
Dastur formasi quyidagi rasm bo’ycha bo’ladi:



Dastur matni quyidagicha ko’rinishga ega bo’ladi:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Ch: Char;
  i: Integer;
begin
  Label1.Caption := '';
  for I := 32 to 255 do
  begin
     Ch := Chr(i);  {I-tartibdagi belgini Ch uzgaruvchiga uzatish}
     Label1.Caption := Label1.Caption + Ch + ' - ' + IntToStr(i) + #10#13;
  end;
end;
```

Satr*lar*

Satr - bu belgilarning oddiy ketma-ketligidir: 'Ab21#9!cd', 'Anvarjon Omonov'. Satr bo’sh yoki bitta belgili bo’lishi ham mumkin. Satrli o’zgaruvchi uzunligi 256 tagacha bo’lgan belgili qiymatlarni qabul qilishi mumkin. Umuman olganda, har bir qatorli o’zgaruvchiga xotiradan 256 bayt joy ajratiladi. Xotirani tejash uchun qatorning tipini quyidagicha ko’rsatish maqsadga muvofiqdir: ***String***[N], N - qatordagi belgilar soni. Bu holda belgili o’zgaruvchi uchun N bayt joy ajratiladi.

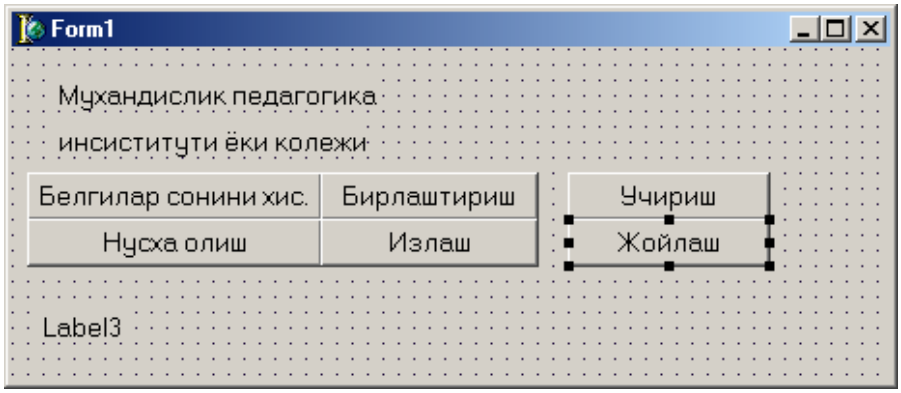
Belgilar va qatorlar sirtida bir qancha amallar bajarish mumkin, ya’ni qatordan kerakli bo’lakni kesib olish, qatorlarni bir-biriga qo’shish va natijada yangi qatorlar hosil qilish va boshqalar.

Satrlar va belgilar sirtida turli amallar bajarish mumkin. Ular quyidagilar:
Satrli belgilar sirtida amallar bajarish uchun

Yozilishi	Vazifasi
Function Length(S):Integer	**S** satrli o’zgaruvchidagi belgilar sonini aniqlaydi
Function Copy(S; Index, Count: Integer): **string**;	**S** satrli o’zgaruvchidagi **Index** - tadan **Count** ta belgidan nusxa olish
Function Concat(s1 [, s2,..., sn]: **string**): **string**;	**S1** dan **sn** tagacha bo’lgan satrli o’zgaruvchilarni bitta satrli o’zgaruvchiga birlashtirish
Function Pos(Substr: **string**; S: **string**): Integer;	Substr satri **S** satridan izlanadi. Agarda izlangan satr topilmasa natija nolga teng bo’ladi
Procedure Delete(var S: **string**; Index, Count:Integer);	**S** satrdagi **Index** – belgidan **Count** ta belgini o’chirib tashlaydi
Procedure Insert(Source: **string**; var S: **string**; Index: Integer);	**S** satriga **Index** – belgidan boshlab **Source** satrini joylashtiradi

Yuqoridagi amallarga misol qilib quyidagilarni keltirish mumkin:

Dastur formasining ko’rinishi



Satrlar sirtida turli amallar bajarish dasturi

unit Satr_p;
interface
uses

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Label3: TLabel;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    Label3.Caption := 'Birinchi yozuvdagi belgilar soniq ' +
    IntToStr(Length(Label1.Caption));
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
   Label3.Caption := Concat(Label1.Caption, Label2.Caption);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
```

```
   Label3.Caption := Copy(Label1.Caption, 13, 7);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
    Label3.Caption := 'Birinchi yozuvdagi "xan" suzi ' + IntToStr(Pos('xan',
    Label1.Caption)) + ' - belgidan boshlangan';
end;

procedure TForm1.Button5Click(Sender: TObject);
Var
  S: String;
begin
   S := Label2.Caption;
   Delete(S, 12, 11);
   Label3.Caption := S;
end;

procedure TForm1.Button6Click(Sender: TObject);
Var
  S: String;
begin
    S := Label1.Caption;
    Insert('va ', S, 13);
    Label3.Caption := S;
end;
end.
```

Shunday dastur tuzingki, kiritish ob`ektida faqat raqamlar yoki belgilar ketma-ketligi kiritilganligini aniqlasin.

5-Ma`ruza. Massivlar. Massivni e`lon qilish hamda massiv elementlarini kiritish va chiqarish. Massivlar ustida amallar bajarish. (2 soat)

O’quv modul birliklari:

1. Massivni e`lon qilish.
2. Massiv elementlarini kiritish va chiqarish.
3. Kiritilgan elementni massivdan izlash.
4. Massiv elementlarini tartiblash.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Matematik to’plam va mavssiv tushunchalariga ega bo’ladi.
2. Massiv yaratish, uning elementlarini kiritish va chiqarish dasturlarini yarata oladi.
3. Massivlar ustida turli amallarni bajaruvchi algoritmlar va dasturlarni yarata oladi.

Massivlar. Massivlar sirtida amallar

Dasturlashda eng ko’p qo’llaniladigan dastur ob’yektlarining biri bo’lgan massivlar bilan tanishib chiqamiz.

Massiv - bu bir xil tipli, chekli qiymatlarning tartiblangan to’plamidir. Massivlarga misol sifatida matematika kursidan ma’lum bo’lgan vektorlar, matritsalar va tenzorlarni ko’rsatish mumkin.

Dasturda ishlatiluvchi barcha massivlarga o’ziga xos ism berish kerak. Massivning har bir hadiga murojaat esa uning nomi va o’rta qavs ichiga olib yozilgan tartib hadi orqali amalga oshiriladi.

Massivning zarur hadiga murojaat quyidagicha amalga oshiriladi:

<massiv nomi>[<indeks>]

bu yerda <indeks> massiv hadining joylashgan joyini anglatuvchi tartib qiymati.

Umuman olganda, <indeks> o’rnida ifoda qatnashishi ham mumkin. Indeksni ifodalovchi ifodaning tipini indeks tipi deb ataladi. Indeks tipining qiymatlar to’plami albatta nomerlangan to’plam bo’lishi, shu bilan bir qatorda massiv hadlari sonini aniqlash va ularning tartibini belgilashi kerak.

Massivlarni e’lon qilishda indeks tipi bilan bir qatorda massiv hadlarining tipi ham ko’rsatilishi kerak. Bir o’lchamli massivni e’lon qilish quyidagicha amalga oshiriladi:

array [<indeks tipi>] **of** <had tipi>;

Ko’pincha <indeks tipi> sifatida cheklanma tiplardan foydalaniladi, chunki bu tipga tegishli to’plam tartiblangan va qat’iy nomerlangandir. Misol uchun 100 ta haqiqiy sonli hadlardan iborat massiv quyidagicha e’lon qilinadi:

array [1..100] **of** real;

Massivlarni e’lon qilish haqida to’liqroq ma’lumot berish uchun turli tipdagi indekslarga oid misollarni e’tiboringizga havola qilamiz:

1. **array** [1000..5000] **of** integer;
2. **array** [-754..-1] **of** byte;
3. **array** [0..100] **of** real;
4. **array** [0..10] **of** boolean;
5. **array** [10..25] **of** char;
6. **type**
   chegara = 1..100;
   vektor = **array** [chegara] **of** real;
   massiv1 = **array** [115..130] **of** integer;
   massiv2 = **array** [-754..-1] **of** integer;
   var
   A, B: vektor;
   c, d: massiv1;
   e: massiv2;
7. **var**
   r, t: **array** [chegara] **of** real;
   s, q: **array** [115..130] **of** integer;
   p: array [-754..-1] **of** integer;
   k, m: **array** [1..50] **of** (shar, kub, doira);
8. **type** kv1 = (yanvar, fevral, mart);
   var t, r: **array** [kv1] **of** real;
9. **type**
   belgi = **array** [boolean] **of** integer;
   belgi_kodi = **array** [char] **of** integer;
   var
   k: belgi;
   p: belgi_kodi;

Endi massivlar sirtida tipik amallar bajaruvchi bir nechta dastur bilan tanishib chiqaylik. Bir o’lchamli n ta hadli (n=30) massiv hadlarini yig’ish.

```
const n = 30;
var
   i: integer;
   x: array [1..n] of real;
   S: real;
   St: string;
begin
 for I := 1 to n do
 begin
   st:=InputBox(’’, ’’, ’’);
```

```
    x[i] := StrToFloat(st); { massiv xadlarini kiritish}
  end;
  S := 0;
  for I := 1  to  n do  S := S + x[i];
  ShowMessage('natijaq', S)
end;
```

2. Bir o'lchamli, n ta hadli (nq30) massiv hadlarining eng kattasini topish va uning joylashgan joyini aniqlash:

```
const  n = 30;
type
  gran = 1..30;
  vector = array [gran] of real;
var
  x: vector;
  S: real;
  l: byte;
  k: integer;
  st: string;
begin
  for I := 1 to n do
  begin
    st:=InputBox('', '', '');
    x[i] := StrToFloat(st); { massiv xadlarini kiritish}
  end;
  S := x[1]; k := 1;
  for I := 2  to n  do
        if x[i] > S  then
        begin
      S := x[i];  k := I
        end;
  ShowMessage('x  massivininig eng katta xadi' + FloatToStr(S));
  ShowMessage('max(x) ning urni' + FloatToStr(k))
end;
```

3. n ta hadli (n q 15) vektorlarni skalyar ko'paytmasini aniqlash:

```
const  n = 15;
type
gran = 1..n;
        mas = array [gran] of  real;
var
  i: byte;
  S: real;
  x, y: mas;
begin
```

```
  for I := 1 to n do
 begin
   st:=InputBox(' X massiv elementlari', '', '');
   x[i] := StrToFloat(st); { massiv xadlarini kiritish}
 end;
 for I := 1 to n do
 begin
   st:=InputBox('Y massiv elementlari', '', '');
   y[i] := StrToFloat(st); { massiv xadlarini kiritish}
 end;
 S := 0;
 for I := 1 to n do S := S + x[i] * y[i];
 ShowMessage('natija'+ FloatToStr(S))
end;
```

Ko'p o'lchamli massivlar

Bir o'lchamli massivlarning hadlari skalyar miqdorlar bo'lgan edi. Umumiy holda esa massiv hadlari o'z navbatida yana massivlar bo'lishi mumkin, agar bu massivlar skalyar miqdorlar bo'lsa natijada ikki o'lchamli massivlarni hosil qilamiz. Ikki o'lchamli massivlarga misol sifatida matematika kursidagi matritsalarni keltirish mumkin. Agar bir o'lchamli massivning hadlari o'z navbatida matritsalar bo'lsa natijada uch o'lchovli massivlar hosil qilinadi va h.k.

Ikki o'lchamli massiv tipini ko'rsatish quyidagicha bajariladi:

array[<indeks tipi>] **of array**[<indeks tipi>] **of** <skalyar tip>;

Ikki o'lchamli massivlar tiplarini aniqlashni bir necha xil usulda quyidagi misol sirtida ko'rib chiqaylik: (10 ta satr va 20 ta ustundan iborat matritsa tipini aniqlash, massiv hadlari real tipida bo'lsin)

1. **array** [1..10] **of array** [1..20] **of** real;
2. **var**
   A: **array**[1..10] **of array**[1..20] **of** real;
3. **type** matr = **array** [1..10] **of array** [1..20] **of** real;
   var
   A: matr;
4. **type**
   gran1 = 1..10;
   gran2 = 1..20;
   matr = **array**[gran1, gran2] **of** real;
   var
   A: matr;
5. **var**

A: **array**[1..10, 1..20] **of** real;

Yana shuni aytish mumkinki, ikki o'lchamli massiv indekslarining tiplari turli xil bo'lishi ham mumkin. Bu holni quyidagi misol yordamida ko'rib chiqaylik:

```
const  n = 24;
type
  hafson = (dush, sesh, chor, pay, jum, shan, yaksh);
  Ishkun = dush..jum;
  detson = array[1..n] of char;
var
  A: array[boolean] of array[1..n] of char;
  B: detson;
  S: array[1..365] of detson;
```

Ikki o'lchamli massivlar sirtida bir nechta tugallangan dasturlar bilan tanishib chiqaylik.

1. Matritsalarni qo'shish:

```
const  n = 3;  m = 4;
         { n -  matritsa satrlari soni,
           m -  ustunlar soni}
var
  i, j: integer;
  A, B, C: array[1..n, 1..m] of real;
  St: strin;
begin
  {A, V matritsa xadlarini kiritish}
  for i := 1 to n  do
    for j := 1 to m do
    begin
      st:=InputBox('A massiv elementlari', '', '');
      A[i,j] := StrToFloat(st); { massiv xadlarini kiritish}
      st:=InputBox('B massiv elementlari', '', '');
      B[i,j] := StrToFloat(st); { massiv xadlarini kiritish}
    end;
  Label1.Caption := '';
  for i := 1 to n do
  begin
    for j := 1 to m do
    begin
      C[i,j] := A[i,j] + B[i,j];
      Label1.Caption := Label1.Caption + FloatToStr(C[i,j])
    end;
    Label1.Caption := Label1.Caption + #10#13;
  end;
```

```
 end;
```

2. Matritsa hadlarining eng kattasi (kichigi)ni topish va uning joylashgan joyini aniqlash:

```
const n = 3;  m = 4;
var
  A: array[1..n, 1..m] of real;
  R: real;
  i, j: byte;  K, L: byte;
  st: string;
begin
  {A matritsa hadlarini kiritish}
  for i := 1 to n  do
    for j := 1 to m do
    begin
      st:=InputBox('A massiv elementlari', '', '');
      A[i,j] := StrToFloat(st); { massiv xadlarini kiritish}
    end;
  R := A[1,1];  L := 1;  K := 1;
  for I := 1 to n do
    for j := 1 to m do
    begin
      if R < A[i,j] then
          begin
        R := A[i,j];
            L := i;
        K := j;
          end;
    end;
  ShowMessage('max A = ' + FloatToStr(R) + #10#13 +
                       'satr  = ' + IntToStr(L) + #10#13 +
                       'ustun = ' + IntToStr(K));
end;
```

1. Bir o`lchovli massiv elementlarini o`sib borish ketma-ketligi bo`yicha tartiblash dasturini tuzing.
2. Matritsani vektorga ko`paytirish dasurini tuzing.

6-Ma`ruza. Parametrsiz va parametrli protseduralar. O'tkaziluvchi tiplar, oraliqli tiplar va yozuvlar. Parametr - qiymat va parametr-o'zgaruvchilar. (2 soat)

O'quv modul birliklari:

1. O’tkaziluvchi tiplar, oraliqli tiplar va yozuvlar.
2. Parametrsiz va parametrli protseduralar. Parametr - qiymat va parametr-o’zgaruvchilar.
3. Protsedura e`loni va unga murojaat qilish. Protsedura-funktsiyani e`lon qilish.
4. Funktsiyaga murojaat. Rekursiv funktsiyalar.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. O’tkaziluvchi tiplar, oraliqli tiplar va yozuvlar to’g’risida ma`lumotlarga ega bo’ladi.
2. Parametrsiz va parametrli protseduralar hamda ular asosida dastur yaratishni o’zlashtiradi.

Proseduralar.

Prosedura va funksiya haqida umumiy ma’lumotlar

Programma tuzish jarayonida, uning turli joylarida ma’nosiga ko`ra bir xil, mustaqil xarakterga ega bo`lgan va yechilayotgan asosiy masalaning biror qismini hal qilishni o`z bo`yniga olgan murakkab algoritmdan bir necha marotaba foydalanishga to`g`ri keladi. Masalan, matritsalarni ko`paytirish, matritsani vektorga ko`paytirish, chiziqsiz tenglamani yechish, chiziqli algebraik tenglamalar sistemasini yechish, faktorial hisoblash, yig`indi hisoblash va hokazo kabi masalalarni hal qilish algoritmlari juda ham ko`p masalalarni yechishning bosh algoritmlarida qayta-qayta, turli boshlang`ich ma’lumotlar bilan qatnashishi mumkin. Bunday hollarda, malakali dasturchi programma matnini ixchamlashtirish, programmaning ishonchlilik darajasini oshirish, programmani tahrirlash(otladka)ni tezlashtirish va programmaning umumiyligini (universalligini) ta’minlash uchun prosedura va funksiyalardan kengroq foydalanib, mukammal programma yaratishga harakat qiladi.

Prosedura va funksiyalar mustaqil programmali ob’yektlar hisoblanadi. Bu mustaqil programmali ob’yektni dasturchi o`z hoxishiga va undan olinadigan natijalariga ko`ra prosedura yoki funksiya ko`rinishida aniqlashi mumkin. Odatda olinadigan natija yagona qiymatli bo`lsa funksiyadan, olinadigan natijalar soni bir nechta bo`lsa proseduradan foydalanish maqsadga muvofiqdir.

Prosedurani yozish strukturasi xuddi asosiy programma strukturasi kabi bo`lib, faqat sarlavhalari bilangina farq qiladi holos:

procedure <prosedura ismi>(<formal parametrlar ro`yhati>);
label <metkalar ro`yhati>;
const <o`zgarmaslarni kiritish>;
type <yangi tiplarni aniqlash>;
var <o`zgaruvchilarning tiplarini e’lon qilish>;
<qism programmagagina tegishli bo`lgan ichki prosedura va funksiyalar e’loni>;

```
 begin
        <proseduraning tana qismi>;
end;
```

Proseduralar va funksiyalarni aniqlash asosiy programmaning ***var*** (o`zgaruvchilarning tiplarini e'lon qilish) bo`limida bajariladi. Proseduradan programmada foydalanish uchun uning ismi va faktik parametrlar ro`yhati yoziladi. Shunda prosedura o`ziga belgilangan ishni bajarib, o`zining faktik parametrlari orqali asosiy programmaga o`z natijasini beradi.

Proseduraning e'loni va unga murojat qilishni keyinchalik ko`riladigan misollar orqali o`zlashtirib olamiz.

Parametrsiz proseduralar

Yuqorida aytib o`tganimizdek, prosedura hisoblab bergan natijalar uning faktik parametrlari orqali asosiy programmaga uzatiladi. Lekin, ayrim paytlarda prosedura parametrsiz ham bo`lishi mumkin. Bu holda asosiy programmaning barcha parametrlari prosedura parametrlari rolini bajaradi. Parametrsiz prosedurada ham proseduraning barcha bo`limlari saqlanib qoladi, faqat parametrlar ro`yhatigina qatnashmaydi.

Proseduralarni aniqlash va ulardan foydalanishni quyidagi misol sirtida ko`rib chiqaylik:

Misol: u =max (x + y, x * y), v = max (0.5, u) – berilgan x va y haqiqiy sonlardan foydalanib u va v qiymatlarni aniqlash.

bu yerda x, u - qiymatlari kiritiladigan haqiqiy tipli o`zgaruvchilar.

1. Masalani yechish programmasining proseduradan foydalanmay tuzilgan holi:

```
var
  x, y, u, v: real;
  a, b, s: real;
begin
   {x, u - miqdorlarni kiritish};
  x := StrToFloat(Edit1.Text);
  y := StrToFloat(Edit2.Text);
  a := x + y;  b := x * y;
  if  a > b  then S := a else S := b;
  u := S;
  a := 0.5; b := u;
  if  a > b  then  S := a   else  S := b;
  v := S;
 {olingan natijalar};
  ShowMessage(FloatToStr(u)+' '+FloatToStr(v));
end;
```

Ahamiyat bersangiz, programmadagi shartli operator ikki marta takrorlanib, bir xil ish bajardi.

2. Masalani yechish programmasini parametrsiz proseduradan foydalanib tuzilgan holi (endi yuqoridagi programmada yo`l qo`yilgan kamchilikni proseduralar orqali tuzatishga harakat qilamiz):

```
var
  x, y, u, v: real;
  a, b, S: real;
procedure  max1;
begin
  if  a > b  then  S := a  else  S := b;
end;

begin
  x := StrToFloat(Edit1.Text);
  y := StrToFloat(Edit2.Text);
  a := x + y;  b:=x * y;
  max1;   {max1 proseduraSiga 1-marta murojat    qilinmoqda}
  u := S;
  a := 0.5;   b := u;
  max1;  {max1 proseduraSiga 2-marta murojat qilinmoqda}
  v := S;
  ShowMessage(FloatToStr(u)+’  ‘+FloatToStr(v));
end;
```

Asosiy programmaning operatorlar qismida ikki marta yozilgan max1 parametrsiz proseduraSiga murojat, e’lon qilingan proseduraNi ikki marta asosiy programmaga olib kelib ishlatishni tashkil qiladi. Ahamiyat berilsa, ikkinchi programma birinchi proseduraSiz tuzilgan programmaga ko`ra ixchamroq va soddaroqdir. Biz kiritgan prosedura hozircha faqat ikkita haqiqiy son ichidan kattasini aniqlab berdi holos, shuning uchun programma matnining hajmini kamaytirishdan erishgan yutuq salmoqli bo`lmadi. Lekin, proseduralar asosan ko`p hajmli matndagi amallarni, vazifalarni bajarishga mo`ljallanadi va bu holda erishilgan yutuq salmog`i ancha yuqori bo`ladi.

Parametrsiz proseduraning asosiy kamchiligi, uning asosiy programmaga va undagi ma’lum parametrlarga bog`lanib qolganligidir.

Parametrli proseduralar

Prosedura bilan asosiy programmani bog`laydigan asosiy faktor bu – prosedura parametrlaridir. Parametrlarni ikkita tipga ajratiladi: qiymatli parametrlar (parametr-qiymat), o`zgaruvchili parametrlar (parametr - o`zgaruvchi).

Parametr - qiymat bu prosedurani ishlash jarayonini ta’minlovchi parametrlar hisoblanadi, ya’ni asosiy programma qiymatlarini proseduraga uzatadigan parametrlardir.

Endi, yuqorida ko`rib chiqilgan sonlarni eng kattasini topish algoritmining programmacini qiymatli parametr bilan yozilgan proseduralar orqali amalga oshiraylik:

```
var
  x, y, u, v: real;
  S: real;
procedure max2( a, b: real );
begin
  if a > b then S := a  else  S := b;
end;
begin
  x := StrToFloat(Edit1.Text);
  y := StrToFloat(Edit2.Text);
  max2(x + y, x * y);
  u:=S;
  max2(0.5 , u);
  v:=S;
  ShowMessage(FloatToStr(u)+’ ‘+FloatToStr(v));
end;
```

bu yerda a, b - proseduraning qiymatli formal parametrlari.

Proseduraga murojat qilishda formal va faktik parametrlarning tiplari o`zaro mos kelishi kerak, aks holda programma xato tuzilgan hisoblanadi. Yuqoridagi programmadan ko`rinib turibdiki, a va b formal parametrlar o`rniga natijaviy qiymatlari ma’lum ifodalar qo`yildi. Demak, qiymatli faktik parametrlar o`rniga, shu tipli natijaga erishuvchi ifoda yozilishi mumkin. Bundan tashqari, prosedurada kiritilgan a va b parametrlari faqat proseduraning ichidagina ma’noga ega, tashqarida, misol uchun asosiy programmada ular tushunarsiz, qiymatlari aniqlanmagan miqdorlardir. Shuning uchun, qiymatli parametrlarga prosedura natijalarini o`zlashtirib, asosiy programmaga uzatib bo`lmaydi.

Yuqorida tuzilgan programmaning asosiy kamchiligi, topilgan katta son doim S o`zgaruvchisiga o`zlashtiriladi. Misolimiz shartiga ko`ra esa, natijalar u va v o`zgaruvchilariga o`zlashtirilishi kerak edi. Shuning uchun, programmada ikki marta qo`shimcha *u:=S* va *v:=S* o`zlashtirish operatorlari yozildi.

Bu kamchilikni tuzatish uchun proseduraga yana bir parametrni kiritamiz. Lekin, kiritilgan bu parametr proseduraga qiymat olib kirmaydi balki, prosedura natijasini asosiy programmaga olib chiqib ketadi. Bunday parametrni parametr - o`zgaruvchi deb ataladi.

Parametr-o`zgaruvchini parametr-qiymatdan farq qilish uchun prosedurani aniqlashdagi parametrlar ro`yhatida o`zgaruvchi oldidan ***var*** xizmatchi so`zi yoziladi. Parametr - o`zgaruvchidan so`ng albatta, uning tipi ko`rsatib qo`yiladi. Yuqorida

aytganimizdek, formal parametr - qiymat o`rniga proseduraga murojat vaqtida shu tipli ifoda yozish mumkin bo`lsa, parametr - o`zgaruvchi uchun bu hol mutlaqo mumkin emas.

Proseduraniy mukammallashtirib borish dinamikasini his etish uchun yana, yuqorida ko`rilgan maksimum topish misolining programmasini parametr - o`zgaruvchi ishlatgan holda ko`rib chiqamiz:

```
var
x, y, u, v: real;
procedure  max3(a, b: real; var S: real);
begin
  if  a > b then S := a else S := b;
end;
begin
  x := StrToFloat(Edit1.Text);
  y := StrToFloat(Edit2.Text);
  max3(x + y, x * y, u); {x+y va x*y ifodalarining kattasi u o`zgaruvchisiga
                                        o`zlashtirilmoqda}
  max3(0.5, u, v);       {0.5 va u ifodalarining kattasi V o`zgaruvchisiga
                                        o`zlashtirilmoqda}
  ShowMessage(FloatToStr(u)+’ ‘+FloatToStr(v));
end;
```

Shunday qilib, bitta programmani proseduraning uch xil varianti uchun tuzib chiqib, natijada ixcham va sodda programmaga ega bo`ldik.

Proseduralarni aniqlashda shu paytgacha oddiy tipli parametrlardan foydalanib keldik. Lekin, biz shuni yaxshi bilamizki, Delphi tilida hosilaviy tiplar ham mavjud. Parametr - o`zgaruvchiga hosilaviy, yangi tiplar berish xuddi oddiy skalyar tip berish kabi amalga oshirilaveradi. Ammo, parametr - qiymatlarda yangi tiplar masalasiga batafsilroq yondashish kerak.

Biz yuqorida eslatib o`tdikki, faktik parametr formal parametr - qiymatga mos tipli ixtiyoriy ifoda bo`lishi mumkin. Lekin, Delphi tilida ixtiyoriy tipli qiymatlar uchun shu tipdagi natija beruvchi hech qanday amal ko`zda tutilmagan. Shuning uchun, bu tiplar uchun faktik parametrlar faqat shu tipga mos o`zgaruvchilar bo`lishi mumkin holos. Bunday hol, xususiy holda massivlar uchun ham o`rinlidir.

Faraz qilaylik, programmada o`zgaruvchilar quyidagicha e’lon qilingan bo`lsin:

```
const  n = 20;
type vector = array [1..N]  of  real;
var
  u, v: real;
  x, y: vector;
```

Bu yerda u=max{x_i}, v=max {y_i} larni aniqlash talab qilinayotgan bo`lsin.

Vektorning eng katta hadini topishni albatta prosedura ko`rinishida tashkil qilamiz:

```
procedure max1(A:vector;  var S: real);
var
  i: integer;
begin
  S := A[1];
  for i:=2 to n do
    if  A[i] > S then S:= A[i]
end;
```

Bu proseduraga asosiy programmada murojat *max1(x, u); max1(y, v);* ko`rinishida amalga oshiriladi.

Proseduradagi A vektorini parametr - qiymat sifatida yozib qo`yganimiz uchun, proseduraga qilinayotgan har bir murojatda A vektorga mos ravishda X va Y vektorlari ko`chirib yoziladi va so`ng prosedura o`z ishini bajaradi. Biz bilamizki, bir tarafdan, massivlarning sirtida ko`chirish amalini bajarishga ancha vaqt ketadi, ikkinchi tarafdan, har safar yangidan proseduraga qilingan murojatda A vektor uchun xotiradan qo`shimcha joy ajratiladi. SHuning uchun, proseduraning sarlavhasida quyidagicha almashtirish qilsak, yuqoridagi ikki kamchilikni bartaraf qilgan bo`lamiz:

```
procedure max1(var A: vector; var S: real);
```

Endi prosedurani e’lon qilish, undan foydalanib programma yaratish malakasini hosil qilganimizdan so`ng, uni e’lon qilishning sintaksis qoidalarini ko`rib chiqaylik.

Prosedurani aniqlash (e’lon qilish) quyidagicha amalga oshiriladi:

<prosedurani aniqlash>:=<prosedura sarlavhasi>;<blok>

Bu yerda <blok> tushunchasi to`liqligicha <programma tanasi> tushunchasi bilan bir xil sintaksis qoida asosida aniqlangani uchun, bu tushunchaga ortiq qaytib o`tirmaymiz.

Endi esa <prosedura sarlavhasi>ga ta’rif beramiz:

<prosedura sarlavhasi>:=**Procedure** <prosedura ismi>│ Procedure <prosedura ismi>(<formal parametrlar ro`yhati>)

Prosedura ismi dasturchi tomonidan tanlanadigan oddiy identifikator hisoblanadi.

Formal parametrlar ro`yhati quyidagicha aniqlanadi:

<formal parametrlar ro`yhati> := <formal parametrlar seksiyasi> {; <formal parametrlar seksiyasi>}

Formal parametrlar seksiyasi deganda prosedura parametrlarining parametr-qiymat va parametr-o`zgaruvchi lardan iborat bo`lishligi tushuniladi:

<formal parametrlar seksiyasi>:={<ism> {, <ism>}: <ism tipi> | var <ism>{, <ism>}: <ism tipi>

bu yerda <ism> - formal parametrlar sifatida ishlatiladigan identifikator.

Endi yuqoridagi aniqlashlarga tushuntirishlar berib o`tsak.

Yuqoridagi Bekus-Naur formulalaridan ko`rinib turibdiki, formal parametrlar ro`yhati (agar u mavjud bo`lsa) bitta yoki bir nechta o`zaro nuqta- vergul (;) belgisi bilan ajratilgan seksiyalardan tashkil topgan. Har bir seksiyada esa o`z navbatida, bitta yoki bir nechta o`zaro nuqta-vergul bilan ajratilgan formal parametrlar qatnashishi mumkin. Proseduradagi formal parametrlar sonini, dasturchining o`zi prosedurani aniqlash moxiyatidan kelib chiqqan holda tanlaydi.

Misol:

Procedure P(A:Char; B:Char; Var C:Real; Var D:Real; E:Char);

bu yerda formal parametrlar ro`yhati beshta seksiyadan iborat: A,B,E – lar Char tipli qiymatlar, C, D – lar Real tipidagi o`zgaruvchilar. Shu bilan bir qatorda har bir seksiya faqat, bitta parametrni o`z ichiga olmoqda.

Bir xil tipli, hamda ketma-ket joylashgan qiymatlarni va o`zgaruvchilarni bitta seksiyaga birlashtirib prosedura sarlavhasini quyidagicha yozish ham mumkin:

Procedure P(A, B: Char; Var C, D: Real; E: Char);

Shunday qilib, seksiya deganda bir xil tipli parametr - qiymatlar yoki parametr - o`zgaruvchilarning ro`yhatini tushunish mumkin.

Ko`pchilik boshlovchi dasturchilar yo`l qo`yadigan quyidagi xatoliklardan ehtiyot bo`lmoq zarur:

Procedure P(Var X: Real; Y: Real);

bu sarlavha

Procedure P(Var X, Y: Real);

sarlavhasi bilan bir xil emas.

Aniqlangan proseduraga murojat yoki prosedura operatoridan qanday foydalanishni aniqlashni ko`rib chiqaylik (yaratilgan prosedurani «Aktivlashtirish», ya’ni ishlatish):

<prosedura operatori> := <prosedura ismi> | <prosedura ismi>(<faktik parametrlar ro`yhati>)

Agar prosedura aniqlanishida parametrsiz bo`lsa, unga murojat qilish ham faqat, prosedura ismini yozish bilangina amalga oshiriladi.

Agar prosedura aniqlanishida parametrli bo`lsa, albatta prosedura-operator ham unga mos faktik parametrlar ro`yhatiga ega bo`ladi. Shu parametrlar orqali proseduraga murojat qilinayotganida, formal parametrlar faollashtiriladi:

<faktik parametrlar ro`yhati> := <faktik parametr>{,<faktik parametr>}

	Kvadrat tenglamaning ildizini aniqlab beruvch prosedura yarating va uni faollashtiring.

7-Ma`ruza. Funktsiyaga murojaat. Rekursiv funktsiyalar. Modullar. Modulli dasturlar va ularni tashkil etish usullari. (2 soat)

O’quv modul birliklari:

1. Funktsiyaga murojaat. Rekursiv funktsiyalar.
2. Parametrlarni lokallashtirish printsipi. Funktsiya parametrlar va protsedura parametrlar.
3. Modullar. Standart modullar. Modulli dasturlar va ularni tashkil etish usullari.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Funktsiyalar va ular asosida dastur yaratishni biladi.
2. Modullar yaratishni o’zlashtiradi.

Prosedura-funksiyalar

Prosedura-funksiyaning vazifasi va uning strukturasi

Hajmi katta va murakkab programmalarni ishlab chiqishda, tabiiyki katta qiyinchiliklarga duch kelinadi. Katta, kompleks programmalarni zarur muddatda yaratishga bitta dasturchining esa vaqti yetmaydi. Bunday hollarda, ya’ni muhim ahamiyatga ega bo`lgan va qisqa muddatlarda yaratilish kerak bo`lgan programmalarni ishlab chiqish uchun dasturchilarning katta guruhini jalb etishga to`g`ri keladi. Bunday, yagona programmani yaratishdagi paralel ish olib borishda prosedura va funksiyalarning roli juda katta bo`ladi. Bajarilishi kerak bo`lgan ishni mustaqil bo`limlarga ajratilib, har bir mustaqil ish alohida programmalanib, keyinchalik ular yagona - asosiy programmaga birlashtiriladi.

Asosiy programmada ishlatiluvchi o`zgaruvchilar va prosedura parametrlarini qanday tanlab olish kerak degan muammo, bajariladigan ishning eng og`ir qismlaridan biri bo`lib qoladi. Agar ularni bir-birlariga bog`lab yuborilsa u holda asosiy programmadagi biror o`zgaruvchiga kiritilgan o`zgartirish, prosedurada ishlatilgan va shu o`zgaruvchiga bog`liq barcha ishlarni qaytadan tahlil qilib, tekshirib chiqishga olib keladi. Bunday chalkash va og`ir ishni bajarishning qiyinligi programma tuzishda parallyel, bir nechta dasturchining ish olib borishiga halaqit beradi.

Shuning uchun, prosedura va funksiyalarni yozishda har bir programmaga o`zi yechayotgan masalaga muvofiq holda, turli xil ichki o`zgaruvchilar, programmali ob'yektlar o`zgaruvchilarining turli qiymatlarini tanlab olish huquqi beriladi. Xattoki, bitta o`zgaruvchini turli xil vazifalarda ishlatsa ham bo`ladi. Delphi tilida bunday masalani hal qilish uchun lokallashtirish prinsipi ishlab chiqilgan, ya'ni prosedura yoki funksiyada ishlatilgan o`zgaruvchi shu prosedura yoki funksiyaning ta'sir doirasida (ichida) gina o`z qiymatini saqlab qoladi. Prosedura va funksiyalarning ichida aniqlanib, qiymatlangan o`zgaruvchilarni lokal (ichki) o`zgaruvchilar deb ataladi. Tashqarida, ya'ni asosiy programmada kiritilgan o`zgaruvchilar esa umuman olganda programmaning ixtiyoriy joyida o`z qiymatini saqlab qola oladi. Bu o`zgaruvchilarni global (tashqi) o`zgaruvchilar deb ataladi.

Quyidagi misolda lokallashtirish prinsipi yaqqol ko`zga tashlanadi:

```
const
  n = 1;
var
  t: real;
  x: char;
procedure P(x, y: real);
var
  n: real;
begin
  n := x + t;
  t := y;
  ShowMessage(FloatToStr(n)+' '+FloatToStr(t) +' '+x);
end;
begin
  t:= n/2; x:='+';
  P(n,0.8);
  ShowMessage(FloatToStr(n)+' '+FloatToStr(t) +' '+x);
end;
```

bu yerda t – asosiy programmaning global o`zgaruvchisi;
x, y – R proseduraning formal parametrlar;
n – P proseduradagi lokal o`zgaruvchi.

Matyematika kursidan funksiya tushunchasi bizga yaxshi tanish bo`lib, uning yordamida funksiya va argumyent o`rtasidagi bog`liqlik aniqlanadi. Delphi tilida ham funksiya tushunchasi kiritilgan bo`lib, uni shartli ravishda ikki turga ajratsak bo`ladi: standart funksiyalar, dasturchi tomonidan aniqlangan prosedura - funksiyalar. Standart funksiyalar har bir algoritmik til uchun aniqlanib, amalda ko`p uchrab turuvchi funksiyalarning qiymatlarini hisoblab berishga mo`ljallangan. Masalan, *sin(x), cos(x), exp(x), abs(x), sqrt(x)* va h.k.

Xuddi standart funksiyalar kabi dasturchi ham o`zi uchun zarur, mustaqil programma ob'yektlarini funksiyalar ko`rinishida aniqlab, undan kerakli paytda foydalanishi mumkin.

Funksiya Delphi tilida quyidagi struktura bo`yicha aniqlanadi:

function <funksiya ismi>(<formal parametrlar ro`yhati>):<funksiya qiymatining tipi>;
label <metkalar ro`yhati>;
const <o`zgarmaslarning qiymatlarini aniqlash>;
type <yangi tiplarni kiritish>;
var
<o`zgaruvchilarning tiplarini e'lon qilish>;
<funksiya uchun ichki proseduralar va funksiyalarni aniqlash>;
begin
<funksiyaning tana qismi>
end;

Yuqorida eslatib o`tganimizdek, funksiyalar ham proseduralar kabi mustaqil programmalar hisoblanib, asosiy programma orqali boshqariladi va xuddi asosiy programma va proseduraga o`xshash strukturada yoziladi.

Funksiyani ham prosedura kabi programmaning ***var*** (o`zgaruvchilar tiplarini e'lon qilish) bo`limida aniqlab qo`yiladi. Prosedura uchun aytilgan gaplarning dyeyarli barchasi funksiya uchun ham o`rinlidir. Funksiyaning proseduradan asosiy farqi quyidagilardir:

funksiya sarlavhasi boshqacha aniqlanadi;

funksiyaning ishi davomida olinadigan natija funksiyaning ismiga o`zlashtiriladi, ya'ni funksiyaning tana qismida albatta, funksiya ismiga mos tipli qiymat o`zlashtirilgan bo`lishi kerak;

funksiyadan asosiy programmaga uning ismi orqali bittagina qiymat beriladi.

Funksiyaga murojat ham xuddi proseduradagi kabi amalga oshiriladi, lekin funksiyaning mos tipli ifodada qatnashish kabi qo`shimcha imkoniyati mavjud.

Endi funksiyani aniqlash va unga murojat qilishni to`liqroq o`rganish uchun quyidagi misolni e'tiboringizga havola qilamiz:

Misol: f(n) = n! (n! = 1 * 2 * 3 * ...* n - faktorial) funksiyadan foydalanib,

$Y = \frac{20!+3!}{5!+31!} * \frac{(k+1)!}{m!}$ - ifodani hisoblashni tashkil qiling:

var

```
   k, m, i: integer;
  y: real;
function fact (n: integer): integer;
var
  j: integer;
  P: byte;
begin
  j := 1;
  for p := 1 to  n do j := j * p;
  Result := j;
end;
begin
  k := StrToFloat(Edit1.Text);
  m := StrToFloat(Edit2.Text);
  y:= (fact(20) + fact(3))/(fact(5) + fact (31)) * fact(k+1)/fact(m);
  ShowMessage(FloatToStr(y));
end;
```

Funksiyalarni aniqlashda doim shunday harakat qilish lozimki, uning tana qismida formal parametrlar va funksiyani aniqlash uchun zarur bo`lgan lokal o`zgaruvchilargina qatnashsin. Programmaning global o`zgaruvchisiga iloji boricha prosedura yoki funksiya ichidan turib qiymat bermaslik kerak, aks holda programma xato natija berishi mumkin.

Misol:

```
var
  x,y: integer;
Funktion  f(t: integer): integer;
begin
  f := t * t;
  x := 7;
end;
begin
  x := 5;
  ShowMessage(FloatToStr(x));
  y := f(2) + x;
  ShowMessage(FloatToStr (x)+’  ‘+FloatToStr (y));
end;
```

Bu programmaning ishlashi natijasida x=5, y=11 va x=7 qiymatlar ekranga chiqariladi, ya’ni funksiyaning ichki qismidagi x=7 qiymati asosiy dasturdagi natijaviy qiymatlarga o`z ta’sirini o`tkazmoqda.

Rekursiv funksiyalar

Delphi tilida prosedura – funksiyalar bilan ishlashda, funksiyalarning rekursivlik hossasidan foydalanish imkoniyati yaratilgan.

Rekursiya tushunchasiga misol qilib oddiy faktorial hisoblashni keltirish mumkin:

$$n! = \begin{cases} 1 & agar \quad n = 0 \\ n \cdot (n-1)! & agar \quad n > 0 \end{cases}$$

bu yerda ko`rinib turibdiki n! qiymati (n-1)! orqali aniqlanayapti, ya'ni rekursiya degani o`zi orqali o`zini aniqlash ma'nosini anglatadi.

Delphi tili ham funksiyalarni rekursiv aniqlash imkoniyatini beradi. Funksiyani rekursiv aniqlash uning tana qismida o`ziga - o`zi murojat qilish orqali amalga oshiriladi.

Yuqoridagi faktorial hisoblashni rekursiv funksiyalar orqali amalga oshiraylik:

```
var
  n: integer;
  y: integer;
function fact(m: integer): integer;
var
  k: integer;
begin
  if m = 0 then fact := 1 else fact := fact(m-1) * m;
end;
begin
  n := StrToFloat(Edit1.Text);
  y := fact(n);
  ShowMessage(FloatToStr(y));
end;
```

Funksiyalarni rekursiv aniqlash qisqa va tushunarli tilda bo`ladi, rekursiv emas aniqlash esa uzoq va funksiyani ko`rinish effyektini buzadi, lekin birinchi holda sarflangan EHM vaqti va xotira nisbatan ancha yuqoridir.

Parametrlarni lokallashtirish prinsipi

Yuqorida ko`rib chiqilgan va aniqlangan barcha prosedura va funksiyalarning parametrlari yoki qandaydir tipli qiymat, yoki o`zgaruvchilar bo`lgan edi. Ammo, shunday hollar ham uchrab turadiki ayrim parametrlarni funksiyalar yoki proseduralar orqali aniqlash lozim bo`ladi. Bu holga misol sifatida

$$y = \int_a^b f(x)dx \quad \text{va} \quad z = \int_c^d g(x)dx$$

aniq integrallarni trapetsiya usulida taqribiy hisoblash programmasini ko`rib chiqamiz.

Bu yerda a, b, c, d - qiymatlari beriladigan o`zgaruvchilar;

$f(x) = e^{2x} + \sin 6x$, $g(x) = x^2 - 3x^3 \cos x$

Aniq integrallarni trapetsiya usuli yordamida hisoblash algoritmi quyidagi formula asosida bajariladi:

$$y = \int_a^b f(x)dx \approx h\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a+kh)\right]$$

bu yerda $h = \frac{b-a}{n}$, n - [a, b] oraliqni bo`lishlar soni.

```
var
  a, b, c, d, y, z: real;
function f(x: real): real;
begin
  f := exp(2 * x) + sin(6 * x)
end;
function g(x: real): real;
begin
  g := sqr(x) – 3 * x * sqr(x) * cos(x)
end;
procedure  integ(A, B: real; function F(x: real): real; var R: real);
const n = 20;
var
  x, h: real;
  k: integer;
begin
  h := (B - A) / n;
  R := (f(A) + f(B)) / 2;
  for k := 1 to n-1  do R := R + f(a + k * h);
  R := R * h;
end;
{asosiy programmaning operatorlar bo`limi}
begin
     {1 - integral chegaralarini kiritish}
  a := StrToFloat(Edit1.Text);
  b := StrToFloat(Edit2.Text);
  integ(a, b, f, y);
  ShowMessage('y= ' + FloatToStr(y));
   {2-integral chegaralarini kiriting}
  c := StrToFloat(Edit1.Text);
  d := StrToFloat(Edit2.Text);
  integ( c, d, g, z);
  ShowMessage('z= ' + FloatToStr(z));
end;
```

	**Ctg, Tg** funksiyalarini yarating va ulardan foydalanuvchi dastur ta`minotini tuzing.

Modullar.

Modullarning umumiy tavsifi

Shaxsiy kompyuterlarning eng katta kamchiligi ularda amaliy dasturlar kutubxonasining to`liq emasligidir. Katta EHMlarda dasturchilar uchun juda katta dasturlar kutubxonasi xizmaat qilar va ulardan foydalanib tuzilgan dasturlar o`zlarining ishonchlilik darajasi bilan yuqori turar edi. Shaxsiy kompyuterlarning bu kamchiligini yo`qotish uchun Delphida modullar tushunchasi kiritilgan. Umuman olganda har bir malakali dasturchi o`z dasturini prosedura va funksiyalardan foydalanib tuzadi. Lekin, bu prosedura va funksiyalardan boshqa programmalarda foydalanish uchun ularning matinlarini qayta ko`chirib yozish lozim bo`ladi.

Delphida bu masalani echish uchun modullar yaratilib, ularni kompilyasiya qilinadi va bu moduldan boshqa dasturlarda bemalol foydalansa bo`ladi.

Modul – bu aloxida fayl ko`rinishdagi, vazifalari bo`yicha tartiblangan prosedura va funksiyalarning kutubxonasidir. Delphida modul faylining kengaytmasi *.DCU.

Har bir modul o`zining vazifasi bo`yicha alohida bo`lingan bo`lib, undan foydalanish uchun **uses** buyrug`idan foydalaniladi. Oddiy dasturlarni tuzish jarayonida Delphi dasturchi yozgan kodga qarab modullarni e`lon qilish bo`limi(**uses**)ga avtomatik tarzda kerakli modulni qo`shib qo`yadi. Lekin, shuni ta`kidlab o`tish lozimki Delphining standart modullaridan tashqari modullarni ishlatish uchun ko`pchilik hollarda dasturchining o`zi modul nomini kiritishi zarur bo`ladi.

Oddiy dastur tuzish jarayonida Delphining kodlar oynasida quyidagi modul nomlarini ko`rishimiz mumkin:

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)

. . . . . .
```

Deyarli barcha dasturlarni tuzish jarayonida yuqoridagi modullardan foydalaniladi. Shuning uchun bu modullarga ta`rif berib o`tamiz:

Windows – Windows operatsion sistemasi bilan ishlashni tashkil qiluvchi funksiya va proseduralardan tashkil topgan. Undan tashqari oyna, klaviatura, tovushli dinamika bilan ishlashni funksiya va proseduralarini ham o`z ichga oladi;

Message – habarlarning tip va constantalarini o`zida saqlaydi;

SysUtils - qo`shimcha utilitlar paketi bilan ishlash funksiya va proseduralarini o`zida saqlaydi;

Classes – Delphining barcha asosiy (bazaviy) sinflarini o`z ichiga olgan;

Controls – Qurilmalarni boshqarish funksiya va proseduralarini o`zida saqlaydi;

Graphics – tasvirlarni hosil qilish funksiya va proseduralarini o`zida saqlaydi;

Forms – formalarni tashkil qilish va boshqarish funksiya, proseduralarini o`zida saqlaydi;

Dialogs – xabar oynalarining ko`rinish uslublarini o`zida saqlaydi.

Misol sifatida **Dialogs** modulidan foydalanib xabar beruvchi dastur tuzamiz Dastlab buyruqlar menyusidan **Project\View Source** tanlanadi. So`ngra oynadagi dastur butunlay o`chirib, o`rniga quyidagi dastur kiritiladi:

```
program Project1;
uses Dialogs;
var
  s: String;
begin
  s := InputBox('Kiritish oynasi', 'Buyruqlardan birini kiriting','');
  ShowMessage('Siz ' + s + ' buyrugini kiritdingiz.');
end.
```

Faqat **Forms** modulidan foydalanuvchi dastur tuzing.

8-Ma`ruza. Faylli tiplar bilan ishlash. Fayllarga ma’lumotlarni yozish va o’qish. Fayllar bilan ishlashdagi xatoliklar. (2 soat)

O’quv modul birliklari:

1. Fayl yaratish. Kiritish uchun fayl ochish.
2. Kiritish va o’qish uchun fayl ochish.
3. Fayl ochishdagi xatoliklar. Faylni yopish.
4. Sinf, ob`ekt, uslub, inkopsulyatsiya va avlod qoldirish.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Fayllarni dasturiy yaratish va qayta ishlashni o’rganadi.
2. Faylga ma’lumotlarni yozish va o’qish dasturlarini o’zlashtiradilar
3. Xatoliklarni tahlil qila oladilar

Fayllar

Ayni vaqtgacha ko’rib o’tilgan dasturlarning natijalari ekranga chiqarilgan. Shuningdek, Delphi dastur natijalarini kompyuter xotirasiga saqlab qo’yish imkonini ham beradi. Bu natijalardan keyinchalik foydalanish mumkin bo’ladi.

Fayllarni e’lon qilish

Fayl - axborotlarning nomlangan strukturasi bo’lib, axborotning barcha elementlari bir tipga tegishli bo’ladi. Undagi elementlar miqdori amaliy jixatdan chegaralanmagan. Odatda fayl ham o’zgaruvchilarni e’lon bo’limida e’lon qilinishi kerak. Faylni e’lon qilishda fayl elementlarining tipi ko’rsatiladi.

Umumiy holda faylni e’lon qilish quyidagicha:

Nom:file of ElementTipi;
Misollar:
res: file of char; // belgi tipli fayl
koef: **file of** real; // haqiqiy tipli fayl
f: file of integer; // butun tipli fayl

Ma’lumotlari belgili tipga tegishli bo’lgan komponentli fayl belgili yoki matnli deb ataladi. Matnli faylni e’lon qilish quyidagicha bo’ladi:

Nom:TextFile;
Bu yerda:

- nom — faylli o’zgaruvchining nomi;
- TextFile — tipni ifodalash, ya’ni Nom - matnli fayl o’zgaruvchisi.

Faylning vazifasi

Faylli o’zgaruvchini e’lon qilish faqat fayl komponentining tipini beradi xolos. Dastur faylga ma’lumotlarni yozishi yoki fayldan o’qishi uchun aniq bir fayl ko’rsatilishi kerak, ya’ni, faylli o’zgaruvchini aniq bir fayl bilan bog’lash kerak (fayl nomini berish).

Fayl nomi AssignFile protsedurasiga murojaat qilish bilan beriladi. AssignFile protsedurasining ko’rinishi quyidagicha bo’ladi:

AssignFiie(var f, FaylNomi: string)

Faylning nomi Windows qoidalariga bo’ysingan holda beriladi.
Quyida AssignFile protsedurasini chaqirishga misollar keltirilgan:

```
AssignFile(f, 'a:\result.txt');
AssignFile(f, '\students\ivanov\korni.txt');
fname:=('otchet.txt'); AssignFiie(f,fname);
```

Faylga yozish

Ma’lumotlarni matnli faylga bevosita yozish write yoki writeln buyruqlari yordamida amalga oshiriladi. Bu buyruqning umumiy holda yozilishi quyidagicha bo’ladi:

```
write (FailliO’zgaruvchi, kiritilayotganRo’yxat) ;
writeln (FailliO’zgaruvchi, kiritilayotganRo’yxat);
```

bu yerda:

- FailliO’zgaruvchi — kiritish bajarilayotgan faylni ifodalovchi o’zgaruvchi;
- kiritilayotganRo’yxat - faylga o’zaro vergul bilan ajratib yozilgan kiritiluvchi qiymat. O’zgaruvchining nomi o’rnida satrli konstantalardan foydalansa ham bo’ladi.

Masalan, agar f TextFiie tipidagi o’zgaruvchi bo’lsa, u holda x1 va x2 o’zgaruvchilarning qiymatlarini faylga kiritish buyrug’i quyidagicha bo’ladi:

```
write(f, 'Tenglama ildizlari', xl, x2);
```

write va writeln buyruqlarining o’zaro farqi shundaki, writeln buyrug’ida ro’yxatda ko’rsatilgan barcha qiymatlarni kiritib bo’lgandan so’ng faylga “yangi satr” belgisini yozadi, ya’ni, kursorni yangi satrga o’tkazadi.

Faylni yozish uchun ochish

Faylga yozish uchun uni avval ochish kerak. Agarda yaratiluvchi faylni hosil qiluvchi dastur avval ham ishlatilgan bo’lsa, u holda diskda ishning natijaviy fayli mavjud bo’lishi mumkin. Shuning uchun dasturchi eski dasturga qanday kirish ni tashkil qilib olishi kerak: eski ma’lumotlarni yangisiga almashtirish kerakmi yoki eskisiga yangi ma’lumotlarni qo’shish kerakmi? Eski fayldan foydalanish uslubi faylni ochish vaqtida aniqlanadi.

Quyidagi uslub bilan ochilgan faylga ma’lumotlarni yozish mumkin:

- ustidan yozish (yangi fayl eskisini ustidan yoziladi va natijada eski fayldagi axborotlar yo’qotiladi)
- joriy faylga yangi ma’lumotni qo’shish

Yangi faylni yaratish yoki mavjud faylni yangisi bilan almashtirish uchun Rewrite(f) protsedurasiga murojaat qilinadi. Bu yerda f - TextFile tipidagi faylli o’zgaruvchi.

Mavjud faylga yangi ma’lumotlarni qo’shish uchun esa Append(f) protsedurasiga murijaat qilishga to’g’ri keladi. Bu yerda f - TextFile tipidagi faylli o’zgaruvchi.

7.1-rasmda matnli faylga ma’lumot yozish yoki qo’shish dasturining muloqot oynasi keltirilgan.

7.1-rasm.

7.1-lisitngda esa **Yozish** tugmasini bosish bilan ishga tushuvchi protsedura keltirilgan. U yangi faylni yaratadi yoki joriy faylni yangisi bilan almastirib faylga Memo1 maydonidagi matnni yozadi.

Yaratiluvchi faylning nomini Edit1 maydoniga kiritish kerak.

7.1-listing.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: TextFile;       // fayl
  fName: String[80]; // fayl nomi
  i: integer;
begin
  fName := Edit1.Text;
  AssignFile(f, fName);
  Rewrite(f);  // faylni yangitdan yaratish
  // faylga yozish
  for i:=0 to Memo1.Lines.Count do // satrlar noldan raqamlanadi
       writeln(f, Memo1.Lines[i]);
  CloseFile(f); // fayni yopish
  MessageDlg('Ma`lumot faylga YOZILDI !!!',mtInformation,[mbOk],0);
end;
```

7.2-listingda esa Qo’shish tugmasi bosilganda ishga tushuvchi protsedura keltirilgan. Bunda mavjud faylga Memo1 maydonidagi ma’lumotlar qo’shiladi.

7.2-listing.

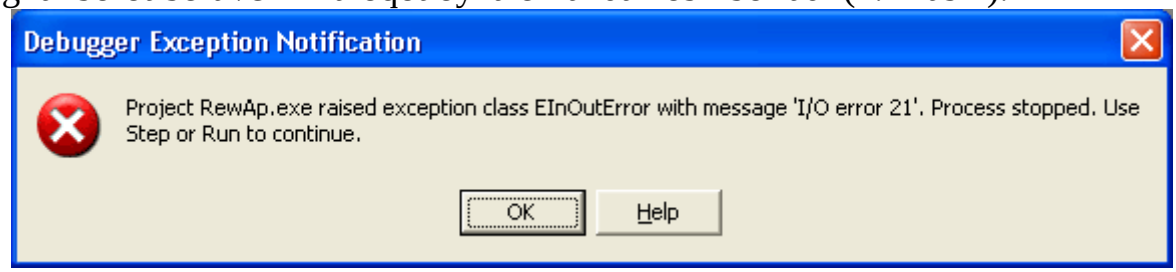
```
procedure TForm1.Button2Click(Sender: TObject);
var
  f: TextFile;       // fayl
```

```
    fName: String[80]; // fayl nomi
   i: integer;
begin
   fName := Edit1.Text;
   AssignFile(f, fName);
   Append(f);  // qo'shish uchun ochish
   // faylga yozish
   for i:=0 to Memo1.Lines.Count do // satrlar noldan raqamlanadi
        writeln(f, Memo1.Lines[i]);
   CloseFile(f); // faylni yopish
   MessageDlg('Ma`lumotlar faylga QO`SHILDI',mtInformation,[mbOk],0);
end;
```

Faylni tashkil qilishdagi xatoliklar

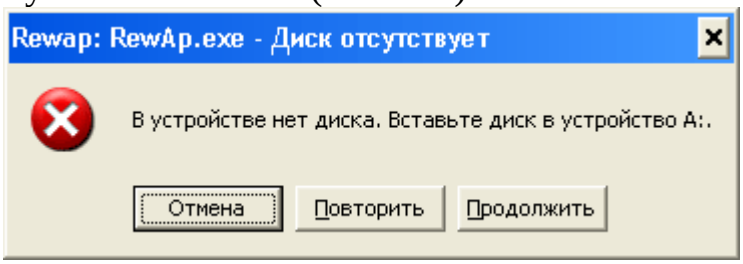
Faylni tashkil qilishga urinish muvaffaqqiyatsiz tugashi mumkin va dastur ishi davomida xatolik hosil bo'ladi. Bunga sabablar bir nechta bo'lishi mumkin. Masalan, fayl ishga tayyor bo'lmagan yumshoq diskda yaratilayotganda (disk yozishdan homiyalangan yoki disk yurituvchi qo'yilmagan bo'lsa). Boshqa bir sabab - avval yaratilmagan faylga ma'lumot qo'shishda (fayl yo'q - hech qayerga qo'shib bo'lmaydi)

Dastur Delphidan ishga tushirilganda va xatolik yuz vergan vaqtda xatolik haqidagi axborot beruvchi muloqot oyna ekranda hosil bo'ladi (7.2-rasm).



7.2-rasm.

Agar dastur Windowsda ishga tushirilsa va xatolik yuz bersa u holda quyidagicha muloqot oynasi hosil bo'ladi (7.3-rasm)

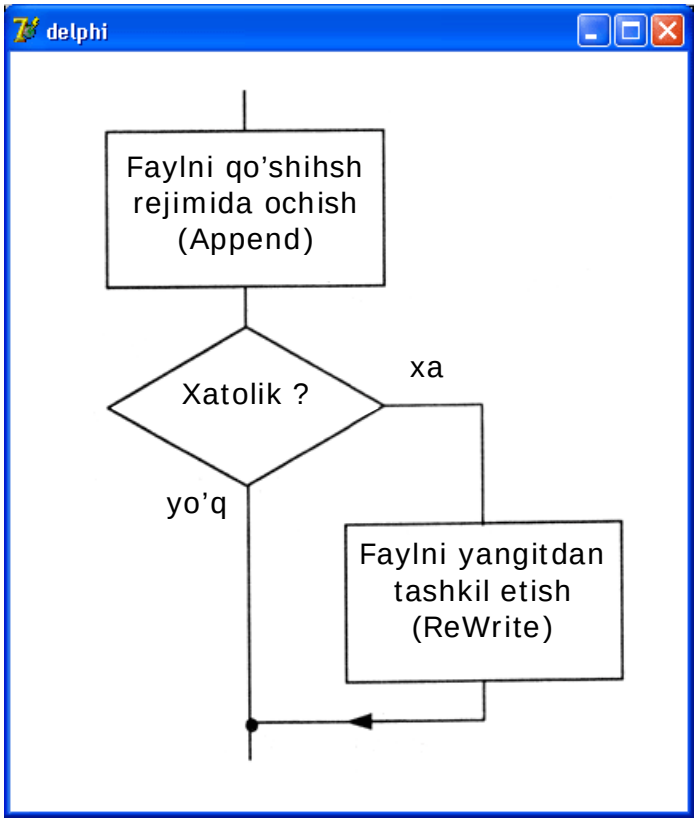


7.3-rasm.

Dastur yordamida fayl yaratishdagi natijani nazorat qilish mumkin. Buning uchun IOResult (Input-Output Result - kirish/chiqish dagi natija) funksiyasining qiymatini tekshirish kerak. Agar kirish/chiqish amali muvofaqiyatli yakunlansa IOResult funksiyasi 0 qaytaradi, aks holda xatolik kodini qaytaradi (nol emas).

Dastur kirish/chiqish amalining bajarilish natijasini tekshirisha olishi uchun unga ruxsat berish kerak. Buning uchun faylni ochish protsedurasini chaqirishdan avval kompilyatorning {$I-} direktivasini joylashtirish kerak. Bu deriktiva kirish/chiqishdagi xatoliklarni avtomatik qayta ishlashni taqiqlaydi va kompilyatorga dastur xatoliklarni nazorat qilishni o'z bo'yniga olishi haqida xabar beradi. Faylni ochish buyrug'idan so'ng {$I+} direktivasini joylashtirish kerak. Bu deriktiva kirish/chiqishdagi xatoliklarni avtomatik qayta ishlash rejimini tiklaydi.

7.4 -rasmda faylni ma'lumot qo'shish uchun ochishning algoritm blok-sxemasi keltirilgan. Bu dasturda agar qo'shish uchun diskda fayl mavjud bo'lmasa, fayl yangitdan yaratilishi ta'minlangan (mavjud bo'l,agan faylga murojaat qilishdagi xatolikni tao'g'rilash).



7.4-rasm.

Yuqorida ko'rsatilgan algoritmning dasturi quyida berilgan.

```
AssignFile(f,filename);
{$I-}
Append(f) // qo'shish uchun ochish
{$I+}
if IOResult<> 0 // ochishdagi xatolik
then Rewrite(f); // yozish uchun ochish
```

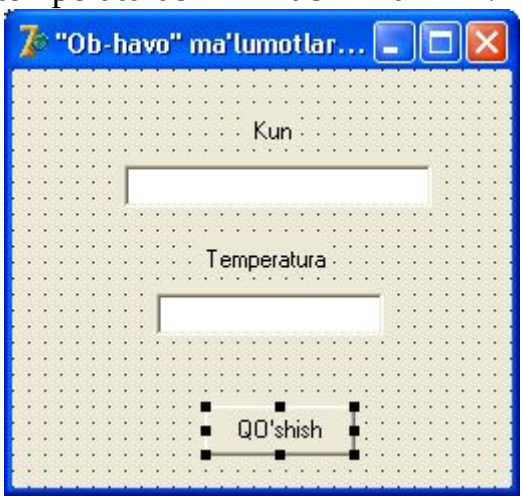
Faylni yopish

Ishni yakunlanishidan oldin barcha ochiq fayllar yopilishi kerak. Buni Close protsedurasi amalga oshiradi. Close protsedurasi bitta parametrga ega, ya'ni faylli o'zgaruvchi nomiga. Protsedurani qo'llashga misol:

Close(f)

Dasturga misollar

Navbatdagi dastur oddiy ma’lumotlar bazasini ifodalaydi. Uning har ishga yuklanishida ekranda muloqot oyna hosil bo’ladi (7.5-rasm). Bu oynada foydalanuvchi sana va havo temperaturasini kiritishi mumkin.



7.5-rasm.

Sana Edit1 maydoniga, temperatura esa Edit maydoniga kiritiladi. Dastur matni 7.3-listingda keltirilgan.

7.3-listing.

```
unit pogoda_;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;      // kun
    Edit2: TEdit;      // temperatura
    Button1: TButton;  // Qo'shish tugmasi
    Label1: TLabel;
    Label2: TLabel;

    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
 var
  Form1: TForm1;
implementation
{$R *.dfm}
const
   DBNAME = 'ob_havo.db';
var
   db: TextFile; // ma'lumotlar bazasi fayli
procedure TForm1.FormActivate(Sender: TObject);
begin
  AssignFile(db, DBNAME);
  {$I-}
  Append(db);
  if IOResult = 0
    then begin
        Edit1.Text:= DateToStr(Date); // Joriy kunni olish
        Edit2.SetFocus; // kursorni Edit2 ga o'rnatish
         end
     else
       begin
         Rewrite(db);
        if IOResult <> 0 then
         begin
         // kiritish oynasi va tugmasini ruxsat etilmagan
          // holatga qo'yish
               Edit1.Enabled := False;
               Edit2.Enabled := False;
               Button1.Enabled := False;
          ShowMessage('Yaratishdagi xatolik '+DBNAME);
           end;
       end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
   if (Length(edit1.text)=0) or (Length(edit2.text)=0)
     then ShowMessage('Ma`lumotlarni kiritishda xatolik.'
            +#13+'Barcha maydonlar to`ldirilishi shart.')
     else writeln(db, edit1.text,' ',edit2.text);
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
       CloseFile(db); // faylni yopish
end;
end.
```

Ma’lumotlar bazasi faylini onActivate hodisasini qayta ishlovchi FormActivate protsedurasi tashkil etadi. OnActivate hodilsasi forma aktivlashishi bilan sodir bo’ladi, shuninh uchun ilova formasi aktivlashganda protsedura avtomatik tarzda ishga tushadi. Agar faylni ochish amali muvoffaqiyatli yakunlansa, u holda Edit1 maydonida joriy sana hosil bo’ladi. Joriy sana haqidagi ma’lumotni Date funksiyasi qaytaradi. Date funksiyasi qaytaruvchi qiymatni (qiymat tipi Double) Edit1 maydoniga chiqarish uchun DateToStr sanani satrli tipga aylantiruvchi funksiyadan foydalaniladi. Edit1 maydoniga sana yozilganidan keyin SetFocus uslubi bilan temperaturani kiritish maydonida kursor o’rnatiladi.

Agar faylni ochish yoki yangi fayl yaratish jarayonida hatolik yuz bersa, u holda Qo’shish tugmasi ruxsat etilmagan holatga qo’yiladi va bu haqda ma’lumot chiqariladi.

TForm1.Button1Click protsedurasi (OnClick hodisasini qayta ishlovchi protsedura) Qo’shish tugmasi (Button1) bosilishi bilan ishga tushadi. Natijada kiritilgan ma’lumotlar ma’lumotlar bazasi - ob_havo.db fayliga yoziladi. Ma’lumot faylga yozilishidan oldin dastur formaning barcha maydonlari to’ldirilganligini tekshiradi, agar to’ldirilmagan bo’lsa ekranda ma’lumot hosil bo’ladi.

Protsedura ishi natijasida joriy sana (kun, oy, yil) va temperatura ob_havo.db faylining oxirida yoziladi.

Ushbu dasturda Write emas Writeln buyrug’idan foydalanildi. Chunki kiritilayotgan har kunlik ma’lumot yangi satrdan joy oladi.

E’tibor bering, writeln buyrug’ining chiquvchi qiymatlari uchta elementdan tashkil topgan. Faylga sana kiritilgandan so’ng bo’sh joy belgisi qo’yiladi. Undan keyin temperatura yoziladi.

Ma’lumotlar bazasini onClose hodisasini qayta ishlovchi Tform1.Formclose protsedurasi ilova formasi yopilayotganda berkitadi.

Dasturni bir necha marta ishga tushirilganidan keyin ob_havo.db fayli quyidagicha ko’rinishga ega bo’lishi mumkin:

9.05.2001 10 10.05.2001 12 11.05.2001 10 12.05.2001 7

Ma’lumotlarni fayldan kiritish

Dasturda kerakli ma’lumotlarni faqatgina klaviatura orqali emas, balki matnli fayllardan ham kiritsa bo’ladi. Buning uchun TextFile tipidagi faylli o’zgaruvchini e’lon qilib olish kerak. Undagi ma’lumotlarni read yoki readln buyruqlari bilan o’qish va olish uchun AssignFile buyrug’i yordamida fayl nomi tayinlanadi.

Faylni ochish

Fayllarni o’qish uchun ochish Reset protsedurasini chaqirish bilan amalga oshiriladi. U fayl tipidagi bitta parametrga ega. Faylni Reset protsedurasi bilan chaqirishdan avval AssignFile funksiyasi yordamida aniq bir fayl ko’rsatilishi kerak.

Masalan, quyidagi buyruq faylni o’qish uchun ochadi:

```
AssignFile(f, 'c:\data.txt');
```

```
Reset(f);
```

Agar fayl noto’g’ri ko’rsatilgan bo’lsa, masalan, ko’rsatilgan nomdagi fayl diskda mavjud bo’lmasa, u holda dastur bajarilish vaqtida xatolik yuz beradi.

Faylni yozish uchun ochishdagi kabi, dastur IoResult funksiyasining qiymatini tekshirish bilan faylni ochishda yuz berishi mumkin bo’lgan xatoliklarni qayta ishlash vazifasini o’z zimmasiga olishi mumkin. Buni 7.4-listingda berilgan dastur matnida ko’rishimiz mumkin.

7.4 - listing.

```
var
fname : string[80]; // fayl nomi
f : TextFile; // fayl
res : integer; // faylni ochishdagi xatolik kodi
                        //(lOResult qiymati)
answ : word; // foydalanuvchiga javob
begin
fname := 'test.txt';
 AssignFile (f, fname);
repeat
{$I-}
Reset(f); // faylni o’qish uchun ochish
{$!+}
res:=IOResult;
if res <> 0
then answ:=MessageDlg('Ochishdagi xatolik '
        + fname+#13 +'Urinishni takrorlaysimi?',mtWarning,
                                              [mbYes, mbNo],0);
until (res= 0) or (answ = mrNo);
if res <> 0
then exit; // Protsedurani yakunlash
end;
```

Malumotlarni fayldan o’qish

Fayldan o’qish **read** va **readln** buyruqlari yordamida amalga oshiriladi. Uning umumiy yozilishi quyidagicha:

```
read( faylli_o’zgaruvchi, o’zgaruvchilar_ro’yxati);
readln(faylli_o’zgaruvchi, o’zgaruvchilar_ro’yxati);
```

bu yerda:

- faylli_o’zgaruvchi —TextFile tipidagi o’zgaruvchi;
- o’zgaruvchilar_ro’yxati — vergullar bilan o’zaro ajratib yozilgan o’zgaruvchilar ro’yxati

Sonlarni o’qish

Shunga e’tibor berish kerakki, matnli faylda son emas, balki uning tasviri joylashadi. Read va readln buyruqlari yordamida matnli fayldan sonlarni o’qish uchun dastlab fayldan ajratish belgisi (bo’sh joy yoki satr ohiri belgilari) uchragunga qadar belgilar o’qib olinadi. So’ngra o’qilgan sonni ifodalovchi belgi songa aylantiriladi va olingan qiymatlar read yoki readln buyrug’ining parametrlari sifatida berilgan o’zgaruvchiga o’zlashtiriladi.

Masalan, agar matnli fayl data.txt quyidagi satrdan iborat bo’lsa:

23 15
45 28
56 71

u holda quyidagi buyruqlar bajarilishi natijasida:

```
AssignFile(f, 'data.txt');
Reset(f);
read (f, a);
read(f, b, c);
read(f, d);
```

o’zgaruvchilarning qiymatlari quyidagicha bo’ladi:

a = 23, b = 15, c = 45, d = 28.

Readln buyrug’ining **read**dan farqi shundaki, fayldan navbatdagi son o’qib olinib, olingan qiymat r**eadln** buyrug’idagi mos o’zgaruvchiga yuklanadi. Agar ushbu o’zgaruvchi **readln** buyrug’idagi so’nggi o’zgaruvchi bo’lsa, u holda qiymatni o’qish navbatdagi satr boshiga uzatiladi (kursor yangi satrga qo’yiladi), hattoki o’qilgan satrda yana son qolgan bo’lsa ham.

Shuning uchuin quyidagi buyruqlar bajarilishi natijasida

```
AssignFile(f,'a:\data.txt');
 Reset(f);
readln(f, a);
readln(f, b, c);
readln(f, d);
```

o’zgaruvchilar quyidagi qiymatlarga ega bo’ladi:

a = 23, b = 45, c = 28, d = 56.

Agar fayldan sonli o’zgaruvchiga qiymat o’qilayotgan vaqtda faylda son o’rnida boshga belgi bo’lsa, u holda hatolik yuz beradi.

Satrni o’qish

Dasturda satrli o’zgaruvchilarni satr uzunligi ko’rsatilgan yoki ko’rsatilmagan holda e’lon qilish mumkin.

masalan:

satr1:**string**[10];
satr2**:string;**

Uzunligi aniq ko’rsatilgan satrli o’zgaruvchiga fayldan qiymat o’qishda joriy satrdan faqat ko’rsatilgan uzunlikdagi belgilar ketma-ketligi o’qib olinadi.

Uzunligi aniq ko’rsatilmagan satrli o’zgaruvchiga fayldan qiymat o’qishda joriy satrning o’qib olinganidan qolgan qismi o’qib olinadi. Boshqacha qilib aytganda fayldan butun satrni o’qib olish kerak bo’lsa, u holda ushbu o’zgaruvchining uzunligi o’qib olinayotgan satr uzunligidan katta bo’lishi kerak.

Readln buyrug’ida ikkita o’zgaruvchiga qiymat o’qib olish kerak bo’lsa u holda u quyidagi holatlardan biri bo’lishi mumkin:

1. Agar oz’garuvchilarning uzunligi aniq ko’rsatilgan bo’lsa, u holda birinchi o’zgaruvchi ko’rsatilgan uzunlikdagi satr qismini o’zlashtiradi, ikkinchi o’zgaruvchi ham satrning qolgan qismidan ko’rsatilgan uzunlikdagi satr qismini o’zlashtiradi.
2. Agar o’zgaruvchilarning uzunliklari aniq ko’rsatilmagan bo’lsa, u holda birinchi o’zgaruvchi joriy satrni to’liq o’zlashtiradi va ikkinchi o’zgaruvchi qiymatga ega bo’lmaydi, ya’ni, satr uzunligi nolga teng bo’ladi.

Masalan, friends.txt matnli fayl quyidagi satrlarga ega bo’lsin:

Narzullayev G’ayrat
Jumabayev Ibrohim
Qodirov Behzod

7.1-jadvalda oz’garuvchilarni e‘lon qilishning bir nechta variantlari keltirilgan.

7.1-jadval.

E’lon qilish	**Fayldan o’zgaruvchilarga o’qish buyrug’i**	**O’zgaruvchilarning fayldan o’qilganidan keyingi qiymati**
fam: string[15] name: string[10]	Readln (f, fam, name)	fam= ' Narzullayev' name= ' G`ayrat'
fam, name: string;	Readln (f, fam, name)	fam= ' Narzullayev G`ayrat' name= ' '
dust: string[80]	Readln (f, dust)	dust =' Narzullayev G`ayrat'

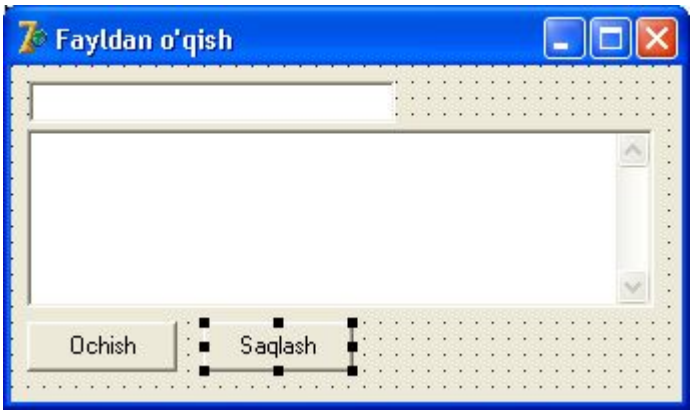
Fayl oxiri

Diskda bir nechta satrdan iborat matnli fayl mavjud bo’lsin. Muloqot oynasiga bu faylda mavjud barcha satrlarni chiqarish kerak. Masala echimi juda oddiy: dastlab faylni ochish kerak, so’ngra birinchi satrni o’qish kerak, keyin ikkinchisini, uchinchisini va hokazo fayl oxiriga etguncha o’qish kerak. Lekin fayl oxiriga etganini qanday bilish mumkin?

Fayl oxirini aniqlash uchun EOF (End of File) funksiyasidan foydalanish mumkin. EOF funksiyasining bitta fayl tipidagi parametri bor. Agar o’qilgan element faylda so’ngisi bo’lmasa EOF funksiyasining qiymati False bo’ladi, agar o’qilgan element faylda so’ngisi bo’lsa, u holda EOF ning qiymati True bo’ladi.

EOF funksiyasining qiymatini fayl ochilishi bilan tekshirish mumkin. Agar uning qiymati True bo’lsa, bu fayl bitta ham elementga ega emasligini bildiradi (bunday fayl hajmi nolga teng).

7.5-listingda qo’yilgan masalani bajaruvchi protsedura keltirilgan. U foydalanuvchi tomonidan berilgan fayldan satrlarni o’qiydi va Memo maydoniga o’qilgan satrlarni chiqaradi. 7.6-rasmda dasturning muloqot oynasi berilgan.



7.6-rasm.

7.5-listing.

```
unit rd_;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    Button2: TButton;
    Edit1: TEdit;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
var
   f: TextFile;       // fayl
   fName: String[80]; // fayl nomi
   buf: String[80];   // fayldan o'qish uchun bufer
```

```
 begin
   fName := Edit1.Text;
   AssignFile(f, fName);
   {$I-}
   Reset(f);  // o'qish uchun ochish
   {$I+}
   if IOResult <> 0 then
     begin
MessageDlg('Faylga murojaatda xatolik ' + fName, mtError,[mbOk],0);
       exit;
     end;
Memo1.Clear;
   // fayldan o'qish
   while not EOF(f) do
     begin
     readln(f, buf);       // satrlarni fayldan oq'ish
Memo1.Lines.Add(buf); // Memo1 maydoniga satrlarni qo'shish
     end;
   CloseFile(f); // faylni yopish
end;
// faylga yozish
procedure TForm1.Button2Click(Sender: TObject);
var
   f: TextFile;       // fayl
   fName: String[80]; // fayl nomi
   i: integer;
begin
   fName := Edit1.Text;
   AssignFile(f, fName);
   Rewrite(f);  // qayta yozish uchun ochish
   // faylga yozish
   for i:=0 to Memo1.Lines.Count do // satrlar noldan
                                                  //raqamlanadi
         writeln(f, Memo1.Lines[i]);
   CloseFile(f); // faylni yopish
   MessageDlg('Ma`lumotlarni faylga yozish',
                              mtInformation,[mbOk],0);
end;
end.
```

Faylni qayta ishlashni tashkil qilish uchun While takrorlash buyrug'idan foydalanilgan. Bunda EOF funksiyasining qiymati har bir o'qishdan oldin tekshiriladi.

Saqlash tugmasi Memo maydonidagi ma'lumotlarni faylga saqlash imkonini beradi. Navbatdagi satrni Memo maydoniga qo'shish Lines xususiyatining Add uslubi yordamida amalga oshiriladi.

	Yozuvlardan foydalanib fayl tashkil qiling va uni tahrirlovchi dastur tuzing.

9-Ma`ruza. Sinf, ob`ekt, uslub, inkopsulyatsiya va avlod qoldirish. Polimorfizm va virtual usullar. (2 soat)

O'quv modul birliklari:

1. Delphida Sinf va ob`ekt tushunchasi
2. Uslub, inkopsulyatsiya va avlod qoldirish.
3. Polimorfizm va virtual usullar.

Aniqlashtirilgan o'quv maqsadlari:

Talaba ushbu mavzuni to'la o'zlashtirgandan so'ng:

1. Delphida sinf va ob`ekt tushunchasining mohiyatini o'rganadi
2. Uslub, inkopsulyatsiya va avlod qoldirish strukturasini biladi
3. Polimorfizm va virtual usullarni o'zlashtiradi.

Ob'yektga mo'ljallangan dasturlash (OMD) - asosida ob'yekt tushunchasi yotgan dasturlarni qayta ishlash uslubidir. Dasturlashda qo'llaniluvchi ob'yekt tushunchasi real hayotdagi ob'yekt tushunchasiga qaysidir ma'noda mos keladi (Ob'yekt real hayotdagi ob'yekt tushunchasiga mos keluvchi qandaydir strukturadir).

Ob'yekt – real hayotga mos keluvchi ob'yekt strukturasi.

OMD uslubidan foydalanib yechiluvchi masala ob'yekt terminlari va ular ustidagi bajariluvchi amallar bilan ifodalanadi, bunday yondoshishdagi dastur esa ob'yektlar to'plami va ular o'rtasidagi aloqalardan tashkil topadi.

Eslatma

Delphida dasturlash muhiti taklif qilgan komponentlar to'plamidan foydalanib dastur yaratish uchun OMD mohiyatini bilish zaruriy emas.

Sinf (Class)

Pascal dasturlash tili dasturchiga murakkab ma'lumotlar tipi - yozuvlarni (records) aniqlash imkonini berdi. Delphi tili esa ob'yektga mo'ljallangan dasturlash mohiyatidan kelib chiqib sinflarni aniqlash imkonini beradi. Sinf - bu ma'lumotlar,

protsedura va funksiyalar berilishidan tashkil topgan hamda ularni sinf vakili tomonidan ishlatish mumkin bo'lgan murakkab strukturadir.

Oddiy sinfni e'lon qilishga misol:

```
TPerson = class
private
fname: string[15];
faddress: string[35];
public
procedure Show;
end;
```

Sinf o'zgaruvchilari **maydon** deb ataladi, prosedura va funksiyalar esa **uslub** deb ataladi. Yuqoridagi misolda **TPerson** – sinf nomi, **FName** va **FAddress** – maydon nomi, **Show** – uslub nomi.

Eslatma

Delphi maydon nomlarini f harfi bilan boshlash kelishib olingan (Field - maydon so'zidan kelib chiqadi).

Sinf tiplarni (type) e'lon qilish bo'limida e'lon qilinadi.

Ob'yekt

Ob'yektlar sinf vakili sifatida dasturning **var** bo'limida e'lon qilinadi, masalan:

```
var
student: TPerson;
professor: TPerson;
```

Eslatma

Delphida ob'yekt - bu dinamik struktura. Ob'yekt o'zgaruvchisi o'zida ma'lumotni emas balki ob'yekt ma'lumotiga yo'lni saqlaydi. Shuning uchun dasturchi ushbu ma'lumotlar uchun ajratilgan xotira haqida o'ylashi kerak bo'ladi.

Xotirani ajratish sinfning maxsus uslubi **create** (yarat) nomiga ega bo'lgan konstruktor yordamida amalga oshiriladi. Konstruktorning asosiy rolini ta'kidlash uchun sinfning berilishida **procedura** so'zi o'rniga **constructor** so'zidan foydalaniladi.

Quyida tarkibiga konstruktor kiritilgan **TPerson** sinfini e'lon qilish keltirligan:

```
TPerson = class
private
fname: string [ 15 ];
faddress: string[35];
constructor Create; // konstruktor
public
```

```
 procedure show; // uslub
end;
```

Ob’yekt (professor) ma’lumotlari uchun xotira ajratish, unga uslub-konstruktorni (Create) ob’yekt tipiga (TPerson) qo’llash natijasining qiymatini o’zlashtirish yo’li bilan amalga oshiriladi. Masalan,

professor := TPerson.Create;

buyrug’i bajarilgandan so’ng, **professor** ob’yektining ma’lumotlari uchun kerakli xotira ajratiladi.

Xotiradan joy ajratish bilan birga konstruktor qoidaga asosan ob’yekt maydonlariga boshlang’ich qiymatni yuklaydi. Quyida Tperson ob’yekti uchun konstruktorni qo’llashga misol keltirilgan:

```
constructor TPerson.Create;
begin
fname := '';
faddress := '';
end;
```

Konstruktorni qo’llash bir oz g’ayrioddiydir. Birinchidan, konstruktor tanasida dinamik xotira ajtatishni ta’minlovchi **New** buyrug’i yo’q. Xotira ajratish bo’yicha barcha zaruriy ishlarni konstruktorning o‘zi bajaradi. Ikkinchidan, dasturda konstruktorga murojaat funksiya uslibida bo’lsa ham konstruktor natija qaytarmaydi.

Ob’yekt maydonariga murojaat etish uchun ob’yekt va maydon nomlarini bir-biridan nuqta bilan ajratilgan holatda ko’rsatish kerak. Ob’yekt oz’i ko’rsatkich bo’lgani bilan maydonlarga murojaat etish vaqtida “^” belgisidan foydalanilmaydi. Masalan, **professor** ob’yektining **fname** maydoniga murojaat etish uchun **professor^.fname** o’rniga **professor.fname** yozish kerak.

Agar dasturda qandaydir ob’yekt foydalanilmayotgan bo’lsa, u holda ushbu ob’yektning maydonlari egallab turgan joyni bo’shatish mumkin. Bu ishni amalga oshirish uchun **Free** destruktor uslubidan foydalaniladi. Masalan, **professor** ob’yektining maydonlari egallab turgan xotirani bo’shatish uchun:

professor.Free;

buyrug’ini yozish kifoya.

Uslub

Sinf uslublari (sinf ichida e’lon qilingan funksiya va protseduralar) sinf ob’yektlari ustida amallar bajaradi. Uslub bajarilishi uchun ob’yekt va uslub nomlarini bir-biridan nuqta bilan ajratib ko’rsatish kerak. Masalan,

professor. Show;

buyrug’i **show** uslubini **professor** ob’yektiga qo’llanishini chaqiradi.

Sinf uslublari dasturda xuddi oddiy protsedura va funksiya kabi aniqlanadi, ammo uslub hisoblangan protsedura va funksiyalar ikki qismdan iborat bo’ladi: uslub tegishli bo’lgan sinf nomidan va uslub nomidan. Sinf nomi uslub nomidan nuqta bilan ajratiladi.

Quyida **Tperson** sinfining **show** uslubini aniqlashga misol keltirilgan.

```
procedure TPerson.Show;
begin
ShowMessage( 'Ismi:'+fname+#13+'Adresi:'+faddress );
end;
```

Eslatma

Uslub buyruqlarida ob’yekt maydonlariga murojaat etish uchun ob’yekt nomi ko’rsatilmaydi.

Ob’yekt xususiyati va inkapsulyatsiya

Inkapsulyatsiya - bu yashirin maydonlarni tushunish va ularga sinf uslubi vositasi yordamida ruhsat (dostup) berishni ta’minlashdir.

Delphi tilida ob’yekt maydonlariga murojaat etishni cheklash ob’yekt xususiyatlari yordamida amalga oshiriladi. Ob’yekt xususiyatlari xususiyat qiymatini saqlovchi maydonlar va xususiyat maydonlariga murojaat etishni ta’minlovchi ikkita uslub bilan xarakterlanadi. Xususiyat qiymatlarini o’rnatish uslubi xususiyatni yozish (write) uslubi deyiladi, xususiyat qiymatlarini olish uslubi esa xususiyatni o’qish (read) uslubi deyiladi.

Sinfning berilishida xususiyat nomidan oldin property (xususiyat) so’zi yoziladi. Xususiyat nomidan keyin uning tipi, so’ngra esa xususiyat qiymatiga murojaat etishni ta’minlovchi uslub nomi ko’rsatiladi. Read so’zidan keyin xususiyatni o’qishni ta’minlovchi uslub nomi, write so’zidan keyin esa xususiyatga yozishni ta’minlovchi uslub nomi ko’rsatiladi.

Quyida ikkita xususiyatdan (Name va Address) iborat bo’lgan Tperson sinfining berilishiga misol keltirilgan:

```
type
TName = string[15];
TAddress = string[35];
TPerson = class // sinf
private
FName: TName; // Name xususiyatining qiymati
FAddress: TAddress; //  Address xususiyatining qiymati
Constructor Create(Name:Tname);
Procedure Show;
Function GetName: TName;
Function GetAddress: TAddress;
Procedure SetAddress(NewAddress:TAddress);
public
Property Name: Tname // Name xususiyati
read GetName; // faqat o’qish uchun
Property Address: TAddress // Address xususiyati
```

```
 read GetAddress // o’qish va
write SetAddress; // yozish uchun
end;
```

Dasturda xususiyat qiymatini o’rnatish uchun ob’yekt xususiyat qiymatini o’rnatish uslubini qo’llovchi buyruqlarni yozilmaydi, balki xususiyat qiymatini oddiy o’zlashtirish buyrug’ini yozish kifoya. Masalan, student ob’yektining Address xususiyatiga qiymat berish uchun quyidagi buyruqni yozish yetarli:

```
student.Address := 'Namangan sh, Do`stlik ko`cha, 11-uy';
```

Kompilyator yuqoridagi xususiyat qiymatini o’zlashtirish buyrug’ini uslubni chaqirish buyrug’iga o’zgartiradi:

```
student.SetAddress('Namangan sh, Do`stlik ko`cha, 11-uy');
```

Dasturda xususiyatni tashqi qo’llash ob’yekt maydonlaridan foydalanishdan hech qanday farq qilmaydi. Biroq, xususiyat va ob’yekt maydoni o’rtasida jiddiy farq mavjud: xususiyat qiymatlarini yuklash va o’qish vaqtida bir qator ishlarni bajaruvchi protsedura avtomatik tarzda chaqiriadi.

Dasturda xususiyat uslublariga bir qator qo’shimcha vazifalarni yuklash mumkin. Masalan, uslub yordamida yuklangan xususiyat qiymatining to’g’riligini tekshirish, xususiyat bilan mantiqiy bog’langan boshqa maydonlarga qiymat o’rnatish, yordamchi protseduralarni chaqilish mumkin.

Ob’yekt ma’lumotlarini xususiyat sifatida rasmiylashtirish ob’yekt xususiyatini saqlovchi maydonlarga murojaat etishni cheklash imkonini yaratadi: masalan, faqat o’qish uchun ruhsat berish mumkin. Dastur buyruqlari hususiyat qiymatlarini o’zgartira olmasligi uchun hususiyat berilishida faqat o’qish uslubining nominigina berish kerak. Faqat o’qish uchun mo’ljallangan hususiyat qiymatini o’zgartirishga urinish kompilyatsiya vaqtida xatolik sodir bo’ladi. Yuqorida keltirilgan **Tperson** sinfining **Name** hususiyati faqat o’qish uchun, **Address** hususiyati esa o’qish va yozish uchun mo’ljallangan.

Yozishdan himoyalangan hususiyat qiymatini ob’yektni yaratish vaqtida o’rnatish mumkin. Quyida TPerson sinfining ob’yektini yaratish va uning hususiyatlariga murojaat etishni ta’minlovchi TPerson sinfining uslublari keltirilgan.

```
// TPerson ob’yektining kostruktori
Constructor TPerson.Create(Name:TName);
begin
FName:=Name;
end;
// Name hususiyatining qiymatini olish uslubi
Function TPerson.GetName;
begin
Result:=FName;
end;
// Address hususiyatining qiymatini olish uslubi
```

```
 function TPerson.GetAddress;
begin
Result:=FAddress;
end;
// Address hususiyatining qiymatini o’zgartirish
Procedure TPerson.SetAddress(NewAddress:TAddress);
begin
if FAddress =' '
then FAddress := NewAddress;
end;
```

Keltirilgan **TPerson** ob’yektining kostruktori ob’yektni yaratadi va **Name** hususiyatining qiymatini aniqlovchi **FName** maydonining qiymatini o’rnatadi.

TPerson sinfning ob’yektini yaratuvchi va uning qiymatlarini o’rnatishni ta’minlovchi dastur buyruqlari quyidagicha bo’lishi mumkin:

```
student := TPerson.Create('Qodirov');
student.Address := 'Namangan sh, Do`stlik ko`cha, 11-uy';
```

Vorislik (me’ros olish)

Ob’yektga mo’ljallangan dasturlashning mohiyati mavjud sinfga maydon, hususiyat va uslub qo’shish yo’li bilan yangi sinf yaratish imkonini beradi. Yangi sinflarni yaratishning bunday mexanizmi tug’ilish deb ataladi. Bunda yangi tug’ilgan sinf (avlod) o’zining ajdod sinfidan hususiyat va uslublarni me’ros qilib oladi.

Avlod sinfni e’lon qilishda ajdod sinf ko’rsatiladi. Masalan, TEmployee (Xodim) sinfi yuqorida ko’rib o’tilgan TPerson sinfiga FDepartment (Bo’lim) maydonini qo’shish yo’li bilan tug’ilgan bo’lishi mumkin. U holda TEmployee sinfini e’lon qilish quyidagicha bo’lishi mumkin:

```
TEmployee = class(TPerson)
FDepartment: integer; // bo’lim raqami
constructor Create(Name:TName; Dep:integer);
end;
```

Qavs ichidagi TPerson sinfining nomi TEmployee sinfi TPerson sinfidan hosil qilib olinganini bildiradi. O’z navbatida TPerson sinfi TEmployee sinfiga manbaa hisoblanadi.

TEmployee sinfi ajdod sinfga va o’zining maydonlariga boshlang’ich qiymatlarni yuklash uchun oz’ining shaxsiy kostruktoriga ega bo’lishi kerak. Quyida TEmployee sinfining konstruktorini qo’llashga misol keltirilgan:

```
constructor TEmpioyee.Create(Name:Tname; Dep:integer);
begin
inherited Create(Name);
```

```
 FDepartment:=Dep;
end;
```

Keltirilgan misolda **inherited** direktivasi yordamida ajdod sinfning kostruktori chaqirilgan. So’ngra avlod sinfning maydoniga boshlang’ich qiymat yuklanyapti.

Avlod sinfning ob’yektini yaratib olgandan keyin dasturda ajdod sinfning maydon va uslublaridan foydalanish mumkin. Quyida ushbu imkoniyatni namoyish etuvchi dastur qismi keltirilgan:

```
engineer := TEmployee.Create('Qodirov',413);
engineer.address := 'Namangan sh, Do`stlik ko`cha, 11-uy';
```

Bu yerda birinchi buyruq TEmployee tipidagi ob’yekt yaratadi, ikkinchisi esa ajdod sinfga tegishli bo’lgan **Address** hususiyatining qiymatini o’rnatadi.

Protected va Private direktivalari

Sinf berilishi sinf elementlarini (maydon, uslub va hususiyatlarni) e’lon qilishdan tashqari qoidaga asosan dasturda sinf elementlarining ko’rinish darajasini o’rnatuvchi **Protected** (himoyalangan) va **Private** (berkitilgan) direktivalariga ega bo’ladi.

Protected bo’limida e’lon qilingan sinf elementlariga murojaat etish faqat uning avlodlarigagina ruhsat etiladi. Ammo ushbu bo’limdagi sinf elementlarining ko’rinish sohasi sinf berilishi joylashgan modulda cheklanmaydi. Odatda **protected** bo’limida sinf uslublari joylashadi.

Private bo’limida e’lon qilingan sinf elementlari faqat sinf joylashgan modul ichidagina ko’rinadi. Bu elementlarga moduldan tashqarida, xatto avlod sinfda ham ruhsat berilmaydi. Odatda **Private** bo’limida sinf maydonlari joylashadi, bu maydonlarga murojaat etishni ta’minovchi uslublar esa **protected** bo’limida joylashadi.

Quyida ruhsat etishni boshqaruvchi direktivalar qo’shilgan TPerson sinfining berilishi keltirilgan

```
TPerson = class
private
FName: TName; // Name hususiyatining qiymati
FAddress: TAddress; // Address hususiyatining qiymati
protected
Constructor Create(Name:TName);
Function GetName: TName;
Function GetAddress: TAddress;
Procedure SetAddress(NewAddress:TAddress);
Property Name: TName
read GetName;
Property Address: TAddress
```

read GetAddress
write SetAddress;
end;

Eslatma

Ba’zan sinfning barcha elementlarini yashirishga to’g’ri keladi. Bu holda sinfning berilishini alohida modulga joylashtirishga to’g’ri keladi, ushbu sinfning ob’yektlarini ishlatuvchi dasturda esa modulga ko’rsatkich qo’yiladi.

Polimorfizm va virtual uslub

Polimorfizm - bu har xil sinflarga kiruvchi uslublar uchun bir hil nomlardan foydalnish imkoniyatidir. Polimorfizmning mohiyati uslubni ob’yektga qo’llanganda, aynan ob’yekt sinfiga tegishli bo’lgan uslubdan foydalinish imkonini beradi.

Quyidagi misolda uchta sinf aniqlangan bo’lib, ulardan biri qolgan ikkitasiga manbaa bo’lsin:

```
type
// Manbaa sinf
TPerson = class
fname: string; // Ism
constructor Create(name:string);
function info: string;  virtual;
end;
// TPerson sinfidan hosil qilingan
TStud = class(TPerson)
fgr:integer; // guruh raqami
constructor Create(name:string; gr:integer);
function info: string; override;
end;
// TPerson sinfidan hosil qilingan
TProf = class(TPerson)
fdep:string; // kafedra nomi
constructor Create(name:string; dep:string);
function info: string; override;
end;
```

Ushbu sinflarning har birida **info** uslubi aniqlangan. Manbaa sinfda **virtual** direktivasi yordamida **info** uslubi virtual deb e’lon qilingan. Uslubni virtual holda e’lon qilish avlod sinfga virtual uslubni o’zining uslubiga almashtish imkonini beradi. Har bir avlod sinfda ajdod sinfdagi mos uslubning o’rnini oluvchi o’zining **info** uslubi aniqlangan. Ajdod sinfdagi virtual uslubning o’rnini oluvchi avlod sinf uslubi **override** direktivasi bilan belgilanadi.

Quyida har bir sinfning **info** uslubi aniqlanishi keltirilgan.

```
function TPerson.info:string;
```

```
 begin
result := '';
end;
function TStud.info:string;
begin
result := fname + '  ' + IntTostr(fgr)+' gr.';
end;
function TProf.info:string;
begin
result := fname + '  ' + fdep+' kaf.';
end;
```

Har ikki sinf bitta manbaa sinfdan tug'ilgani sababli, talabalar va o'qituvchilar ro'yhatini quyidagicha e'lon qilish mumkin (bu yerda ob'yekt ko'rsatkich ekanligini yodga olish kerak):

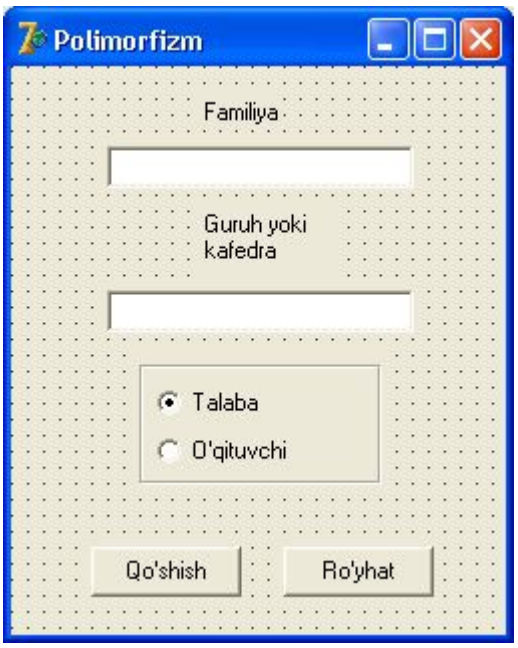
list: **array**[1..SZL] of TPerson;

Ro'yhatni bunday yo'l bilan e'lon qilish mumkin chunki Delphi tili ajdod sinfga qo'yilgan ko'rsatkich orqali avlod sinfga qo'yilgan ko'rsatkichga qiymat yuklash imkonini beradi. Shuning uchun TStud sinfining ob'yekti ham, TProf sinfining ob'yekti ham **list** massivining elmenti bo'lishi mumkin bo'lishi mumkin.

Talabalar va O'qituvchilar ro'yhatini **info** uslubini massiv elementiga qo'llash oraqli chiqarish mumkin. Masalan:

```
st := '';
for i:=1 to SZL do // SZL - massiv ro'yhatining soni
if list[i] <> NIL
then st := st + list[i].Info + #13;
ShowMessage (st);
```

Navbatdagi dastur yuqorida e'lon qilingan TPerson, TStud va TProf sinflaridan foydalanib talabalar va o'qituvchilarning ro'yhatini tashkil qiladi va chiqaradi. Dastur matni 9.1-listingda, muloqot oynasi esa 9.1- rasmda keltirilgan.



9.1-rasm.

9.1-listing

```
unit polimor_;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type
   // Manbaa sinf
```

```
    TPerson = class
        fName: string; // Ism
    constructor Create(name:string);
        function info:string; virtual;
  end;

  // Talaba sinfi1
  TStud = class(TPerson)
        fGr:integer; // guruh raqami
        constructor Create(name:string;gr:integer);
        function info:string; override;
  end;

  // O'qituvchi sinfi
  TProf = class(TPerson)
        fdep:string; // kafedra nomi
        constructor Create(name:string;dep:string);
        function info:string; override;
  end;

const
  SZL = 10; // ro'yhat o'lchami
var
  Form1: TForm1;
  List: array[1..SZL] of TPerson; // ro'yhat
  n:integer = 0;          // ro'yhatdagi odamlar soni

implementation

{$R *.DFM}
constructor TPerson.Create(name:string);
   begin
      fName := name;
   end;

constructor TStud.Create(name:string;gr:integer);
   begin
     inherited create(name); // manbaa sinf konstruktorini chaqirish
     fGr := gr;
   end;

constructor TProf.create(name:string; dep:string);
   begin
inherited create(name); // manbaa sinf konstruktorini
                                    // chaqirish
```

```
        fDep := dep;
    end;

function TPerson.Info:string;
  begin
    result := fname;
  end;

function TStud.Info:string;
  begin
    result := fname + ' ' + IntToStr(fGr)+' gr.';
  end;

function TProf.Info:string;
  begin
    result := fname + ' ' + fDep+' kaf.';
  end;

// Qo'shish tugmasini bosilishi
procedure TForm1.Button1Click(Sender: TObject);
  begin
    if n < SZL then
      begin
        // ob'yektni ro'yhatga qo'shish
        n:=n+1;
        if Radiobutton1.Checked
           then // TStud ob'yektini yaratamiz
             List[n]:=TStud.Create(Edit1.Text,StrToInt(Edit2.Text))
           else // TProf ob'yektini yaratamiz
             List[n]:=TProf.Create(Edit1.Text,Edit2.Text);

 // kiritish maydonlarini bo'shatish
 Edit1.Text := '';
 Edit2.Text := '';
Edit1.SetFocus;  // Familiya maydoniga kursorni o'tkazish
      end
    else ShowMessage('Ro`yhat to`lgan!');
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  i:integer; // indeks
  st:string; // ro'yhat
begin
  for i:=1 to SZL do
```

```
    if list[i] <> NIL then st:=st + list[i].info + #13;
  ShowMessage('Ro`yhat'+#13+st);
end;
end.
```

Qo’shish (Button1) tugmasi bosilishi bilan ishga tushuvchi TForm1.Button1Click protsedurasi TStud yoki TProf sinfiga tegishli bo’lgan **List[n]** ob’yektini yaratadi. Yaratilayotgan ob’yekt sinfi RadioButton tanlash maydonchasining holati bilan aniqlanadi. Tanlash maydonchasini **Talaba** holatiga o’rnatilsa **TStud** sinfi, **O’qituvchi** holatiga o’rnatilsa **TProf** sinfi aniqlanadi.

Ro’yhat (Button1) tugmasi bosilishi bilan ishga tushuvchi TForm1.Button2Click protsedurasi ro’yhatning har bir obyektiga **info** uslubini qo’llab, o’zida to’liq ro’yhatni mujassamlashtirgan satrni tashkil qiladi.

Delphining sinflari va ob’yektlari

Delphi o’z ishchi muhitini yaratish uchun sinflar kutubxonasidan foyadlanadi. Bu kutubxona forma va formaning turli komponentlarini (buyruq tugmalari, taxrirlash maydonlari va boshqalar) qo’llab-quvvatlovchi katta miqdordagi turli-tuman sinflardan tashkil topgan.

Ilova formasini loyihalash vaqtida Delphi dastur matniga zaruriy ob’yektlarni avtomatik tarzda qo’shadi. Delphida yangi forma ochilgan vaqtda dastur matnini tahrirlash oynasi ko’rilsa, u yerda quyidagi satrlarni ko’rish mumkin:

```
type
TForm1 = class(TForm)
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1
```

Agar dasturchi formaga kerakli kompnentlarni qo’shsa Delphi forma sinfining berilishiga ushbu qo’shilgan komponent e’lon qilinishini avtomatik tarzda qo’shadi. Agar dasturchi formaning yoki uning komponentining hodisasini qayta ishlash protsedurasini yozsa, Delphi forma sinfining berilishiga protsedura-uslub e’lon qilinishini avtomatik tarzda qo’shadi.

Sinflar kutubxonasiga vizual (ko’rinadigan) komponentalarning sinflaridan tashqari, vizual bo’lmagan (ko’rinmaydigan) deb ataluvchi komponentalarning sinflari ham kiradi. Ular mos ravishda ob’yektlar yaratishni va ularning uslub va hususiyatlariga murojaat etishni ta’minlaydi. Vizual bo’lmagan komponentaga misol tariqasida taymer (TTimer), fayllarni ochish (TOpenDialog) yoki yozish (TSaveDialog) kabi komponentlarni keltirish mumkin.

10-Ma`ruza. Delphi ning grafik imkoniyatlari. Dum, barmoq va qalam. To’g’ri chiziq. Aylana va ellips. (2 soat)

O’quv modul birliklari:

1. Delphi ning grafik imkoniyatlari.
2. Dum, barmoq va qalam.
3. To’g’ri chiziq.
4. Aylana va ellips.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Delphi ning grafik imkoniyatini o’rganadi.
2. Delphida qalam va mo’yqalam xususiyatlarini biladi
3. Delphi ning grafik imkoniyati yordamida turli figuralar chizish dasturlarini yarata oladi.

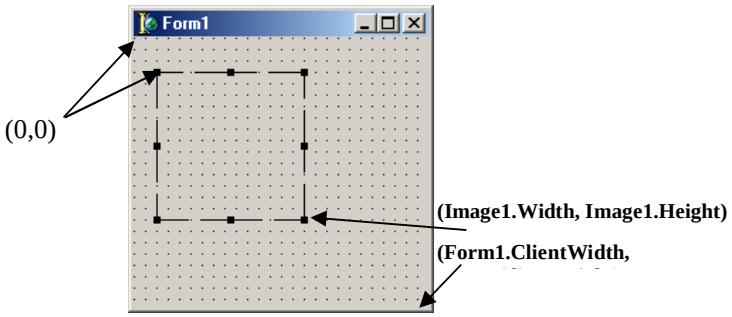
Delphi ning grafik imkoniyatlari

Delphi dasturchiga turli xildagi sxemalar, chizmalar va illyustratsiyalar bilan ishlash imkoniyatlarini beradi. Dastur grafikani ob’yekt (forma yoki komponent Image) sirtida hosil qiladi. Ob’yekt sirti **Canvas** xususiyatiga mos keladi. Grafik element (to’g’ri chiziq, aylana, to’g’ri to’rtburchak va x.k.)larni ob’yekt yuzasida hosil qilish uchun **Canvas** dan foydalaniladi.

Masalan, ***Form1.Canvas.Rectangle(10,10,50,50)*** instruktsiyasi dastur oynasida to’g’ri to’rtburchak hosil qiladi.

Chizma hosil bo’luvchi sirt.

Yuqorida aytib o’tilganidek, grafikani hosil qiluvchi sirt (yuza) **Canvas** xususiyatiga to’g’ri keladi. O’z navbatida **Canvas** xususiyati **TCanvas** tipidagi ob’yektdir. Bu tip uslublari grafik primitivlarni (nuqta, chiziq, aylana va x.k.) hosil bo’lishini ta’minlaydi, xususiyati esa hosil bo’luvchi grafikani harakteristikalarini: rangi, chiziq qalinligi va turi; bo’yaluvchi hududni rangi va ko’rinishini; harfni harakteristikalarini beradi. Canvas «sirt», «chizish uchun yuza» sifatida tarjima qilinadi. Chizish yuzasi alohida nuqta – piksyellardan tashkil topadi. Piksyelni joylashuvi gorizontal (X) va vertikal (Y) koordinatalar bilan harakterlanadi. Chap yuqoridagi nuqta koordinatasi (0,0). Koordinatalar yuqoridan pastga va chapdan o’ngga qarab o’sib boradi (12-rasm).



12-rasm. Chizish yuzasi nuqta koordinatalari.

Chizish yuzasi o'lchamlarini illyustratsiya (Image) hududi uchun **Height** va **Width**, forma uchun esa **ClientHeight** va **ClientWidth** lar aniqlash mumkin.

Qalam va mo'yqalam.

Odatda rassom surat chizish uchun qalam va mo'yqalamdan foydalanadi. Delphi niDelphining grafik imkoniyatlari ham qalam va mo'yqalamdan foydalanish imkoniyatlarini yaratadi. Qalamdan chiziq va kontur chizishda, mo'yqalamdan esa kontur bilan chegaralangan yuzani bo'yash uchun foydalaniladi.

Turli grafik tasvirlarni hosil qilish **Pen** (qalam) va **Brush** (mo'yqalam) xususiyatlariga xosdir. Shu bilan birga ular **TPen** va **TBrush** tiplariga tegishlidir.

Qalam.

Qalamdan nuqta, chiziq, geometrik shakllar: to'g'ri to'rtburchak, aylana, ellips va h.k. larni chizish uchun qurol sifatida foydalaniladi. **TPen** ob'yekt xususiyati quyidagi jadvalda keltirilgan:

Xususiyat	Vazifasi
Color	Chiziq (kontur) rangi
Width	Chiziq qalinligi
Style	Chiziq ko'rinishi
Mode	Tasvirlash rejimi

Color xususiyati chizuvchi qalam rangini belgilaydi. Quyidagi jadvalda **PenColor** xususiyatlari keltirilgan:

Konstanta	Rang	Konstanta	Rang
clBlack	qora	clSilver	kumushrang
clMaroon	kashtanrang	clRed	qizil
clGreen	yashil	clLime	salatrang
clOlive	olivkoviy	clBlue	ko'k

clNavy	to’q ko’k	clFuchsia	Fuchsia
clPurple	atirgulrang	clAqua	yorug’ ko’k
clTeal	Teal	clWhite	oq
clGray	kulrang		

Width xususiyati chizuvchi qalam qalinligini (piksyelda) belgilaydi.

Masalan, Canvas.Pen.Width := 2 chiziq qalinligi 2 piksyelga teng bo’ladi.

Style xususiyati chiziluvchi chiziqning turini belgilaydi. ***Style*** komponentlari quydagi jadvalda keltirilgan.

Konstanta	Chiziq ko’rinishi
psSolid	To’g’ri chiziq
psDash	Uzun shtrixli punktir chiziq
psDot	Qisqa shtrixli punktir chiziq
psDashDot	Uzun-qisqa shtrixli punktir chiziq
PsDashDotDot	Bir uzun va ikki qisqa shtrixli punktir chiziq
PsClear	Ko’rinmas chiziq

Mo’yqalam

Mo’yqalam(Canvas.Brush)dan yopiq sohalarni to’ldirish uchun foydalaniladi, masalan, geometrik shakllarni bo’yash va h.k. Mo’yqalam ob’yekt sifatida quyidagi ikki xususiyatni o’z ichiga oladi:

Color – bo’yaluvchi soha rangi

Style – to’ldiruvchi soha tipi

Masalan, konturning ichki sohasi bo’yalishi yoki shtrixlanishi mumkin.

Color xususiyati sifatida Tcolor ning barcha o’zgarmaslaridan foydalanish mumkin. Style xususiyatlari quyidagi jadvalda keltirilgan:

Konstanta	Bo’yaluvchi soha tipi
bsSolid	to’liq
bsClear	bo’yalmaydi
bsHorizontal	gorizontal shtrixlash
bsVertical	vertikal shtrixlash
bsFDiagonal	oldinga egilgan diagonal shtrixlash
bsBDiagonal	orqaga egilgan diagonal shtrixlash
bsCross	gorizontal-vertikal setkali shtrixlash
bsDiagCross	diagonal setkali shtrixlash

Quyida maydonlarni to’ldirish (bo’yash) usulining dasturi berilgan. Natijada -rasmdagi chizmani xosil qiladi.

```
unit Graf12_1P;

interface

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

type
  TForm1 = class(TForm)
    Image1: TImage;
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject);
const
  BsName: array[1..8] of string =
        ('BsSolid', 'bsClear', 'bsHorizontal',
        'bsVertical', 'bsFDiagonal', 'bsBDiagonal',
        'bsCross', 'bsDiagCross');
var
   x, y: integer;   {To`g`ri to`rtburchakning yuqori chap burchak  kordinatalari}
   w, h: Integer;   { To`g`ri to`rtburchakning uzunligi va bo`yi}
   bs: TBrushStyle; {Maydonlarni to`ldirish usuli}
   k: Integer;      {Tuldirish usulining rakami}
   i, j: integer;
begin
   w := 40; h := 40; {Tugri turtburchak xajmi}
   y := 20;
//   Image1.Canvas.Brush.Color := ClRed;
//   image1.Canvas.Pen.Color := ClRed; //ClBlack;
```

```
   for i := 1 to 2 do
  Begin
    X := 10;
    For j := 1 to 4 do
    Begin
      K := J + (i – 1) * 4;  { Tuldirish usulining rakami }
      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;

      {Maydonlarni chop etish}
      Image1.Canvas.Brush.Color := ClBlack;
      Image1.Canvas.Brush.Style := bs;
      Image1.Canvas.Rectangle(x, y, x+w, y+h);

      {Maydon nomini chop etish}
      Image1.Canvas.Brush.Style := bsClear;
      Image1.Canvas.TextOut(x, y-15, bsName[k]);
      X := x + w + 30;
    End;
    Y := y + h + 30;
  End;
end;

end.
```

13-rasm. «Maydonlarni to’ldirish

usuli» dasturining dialogli oynasi.

Matn hosil qilish

Grafik ob’yekt sirtida matnni hosil qilish uchun **TextOut** uslubidan foydalaniladi. **TextOut** uslubining yozilish formati quyidagicha:

Ob’yekt.Canvas.TextOut(x, y, Text);

Bu yerda
Ob’yekt – matn hosil bo’luvchi ob’yekt nomi;
x, y – matn boshlanuvchi koordinata (14-rasm);
Text – hosil bo’luvchi belgi kattalikdagi matn yoki satrli o’zgaruvchi.

14-rasm. Matn hosil bo’luvchi soha koordinatasi

Hosil bo’luvchi matn belgilari Canvas ob’yektiga muvofiq keluvchi Font xususiyati orqali ifodalanadi. Font xususiyati TFont ob’yektiga tegishli bo’lib, quyidagi jadvalda belgi harakteristikalari va qo’llaniluvchi uslublari keltirilgan:

Xususiyat	Aniqlanishi
Name	Foydalaniluvchi shrift. Qiymat sifatida shrift nomi yoziladi, masalan, Arial Cyr
Size	punktlarda ifodalaniluvchi shrift o’lchami. Punkt-poligrafiyada qo’llaniluvchi o’lchov birligi bo’lib, u taxminan 1/72 dyuym[1]ga teng
Style	belgini yozish usuli, quyidagicha bo’lishi mumkin: oddiy, qalin, kursiv, ostiga chizilgan, sirtiga chizilgan. Bular quyidagi konstantalar yordamida amalga oshiriladi: *fsBold*(qalin), *fsItalic*(kursiv), *fsUnderline*(ostiga chizilgan), *fsStrikeOut*(sirtiga chizilgan). *style* bir nechta usullarni kombinatsiya qilishi mumkin. Masalan, qalin kursiv holatini ifodalash: Ob’yekt.Canvas.Font := [fsBold, fsItalic]
Color	Belgi rangi. Qiymat sifatida *TColor* konstantalaridan foydalanish mumkin.

Quyidagi dastur qismi **TextOut** uslubini qo’llash uchun misol bo’la oladi:

with Form1.Canvas **do**

[1] **Dyuym taxminan 2,5 sm ga tyeng.**

```
 begin
  Brush.Color := Form1.Color;
  Font.Size := 14;
  Font.Style := [fsItalic, fsBold];
  TextOut(10, 10, 'Salom, Delphi!');
end;
```

Matn oynada hosil bo'lgandan so'ng ko'rsatkich uning o'ng yuqori burchagiga siljiydi.

Ba'zida matndan so'ng biror ma'lumotni chiqarish kerak bo'lib qoladi. Agar matn uzunligi noma'lum bo'lsa ko'rsatkich turgan koordinatani aniqlash mushkul. Masalan «so'm» so'zini raqamdan keyin hosil qilish kerak bo'lsin. Bunday holatlarda ko'rsatkich turgan koordinatadan boshlab davom etish uchun PenPos dan foydalanishga to'g'ri keladi:

```
with Form1.Canvas do
begin
  TextOut(10, 10, SumPr);   // SumPr – String tipli kattalik
  TextOut(PenPos.X, PenPos.Y, ' sum');
end;
```

To'g'ri chiziq.

Delphi da to'g'ri chiziq hosil qilish uchun **LineTo** uslubidan foydalaniladi. Uning yozilish formati quyidagicha:

```
Komponent.Canvas.LineTo(x, y);
```

LineTo to'g'ri chiziqni qalam (ko'rsatkich) turgan koordinatadan boshlab x, y – nuqtagacha chizadi. Shuning uchun chiziqning boshlang'ich nuqtasini kerakli joyga o'rnatib olish lozim bo'ladi. Bunda biz **MoveTo** uslubiga murojaat qilamiz:

```
Komponent.Canvas.MoveTo(X0, Y0)
```

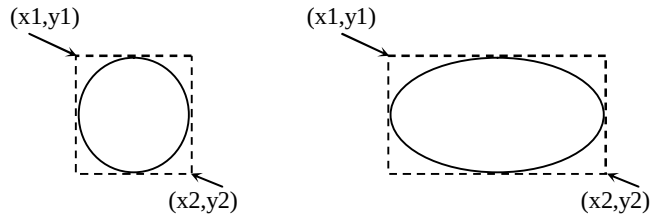
Chiziqning ko'rinishi (rangi, qalinligi va turi) **Pen** ob'yekti bilan ifodalanadi.

Aylana va ellips.

Ellipse uslubi ellips va aylana chizish uchun qo'laniladi. **Ellipse** ning yozilish formati quyidagicha:

Ob'yekt.Canvas.Ellipse(x1,y1,x2,y2)

bu yerda,
ob'yekt – chizma hosil bo'luvchi ob'yekt nomi;
x1,y1,x2,y2 – hosil bo'luvchi aylana yoki ellipsga tashqi chizilgan to'g'ri to'rtburchakning mos ravishda yuqori chap va quyi o'ng nuqtalarini koordinatalari (15-rasm).
Chiziqning ko'rinishi (rangi, qalinligi va turi) **Pen** ob'yekti bilan ifodalanadi.



15-rasm.

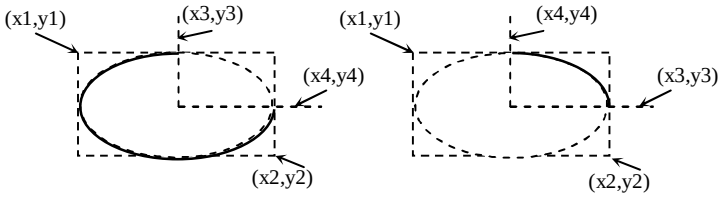
Yoy.

Yoy hosil qilish uchun **Arc** uslubidan foydalaniladi. Uning yozilish formati quyidagicha:

Ob'yekt.Canvas.Arc(x1, y1, x2, y2, x3, y3, x4, y4);

bu yerda
ob'yekt – yoy chiziluvchi ob'yekt nomi;
x1, y1, x2, y2 – hosil bo'luvchi yoyni davom ettirib hosil qilinuvchi ellips (aylana)ga tashqi chizilgan to'g'ri to'rtburchakning mos koordinatalari;
x3, y3 – yoyning boshlang'ich nuqtasi;
x4, y4 – yoyning tugash nuqtasi.

Shuni aytib o'tish lozimki, yoy soat stryelkasi yo'nalishiga qarama-qarshi yo'nalishda chiziladi (16-rasm).



16-rasm.

Chiziqning ko’rinishi (rangi, qalinligi va turi) Pen ob’yekti bilan ifodalanadi.

11-Ma`ruza. To’g’ri to’rtburchak. Ko’pburchak. Multiplikatsiya. (2 soat)

O’quv modul birliklari:

1. Delphi grafikasining qo’shimcha imkoniyatlari.
2. Tayyor tasvirlardan foydalanish
3. Animatsion harakatlarni yaratish.
4. Multiplikatsiya

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Delphi grafikasining qo’shimcha imkoniyatlari bilan tanishadi.
2. Tayyor tasvirlardan foydalanishni biladi
3. Animatsion harakatlarni yarata oladi
4. Sodda multiklar yarata oladi

To’g’ri to’rtburchak.

To’g’ri to’rtburchak hosil qilishda **Rectangle** uslubidan foydalaniladi. Uning yozilish formati quyidagicha:

Ob’yekt.Canvas.Rectangle(x1, y1, x2, y2)

Bu yerda

ob’yekt – tasvir hosil bo’luvchi ob’yekt nomi;

x1, y1, x2, y2 – to’g’ri to’rtburchakning mos ravishda yuqori chap va quyi o’ng burchak koordinatalari.

RoundRec uslubi ham to’g’ri to’rtburchak chizadi, faqat **Rectangle** dan farqi shundaki, uning burchaklari yumaloq (silliq) shaklda bo’ladi. Yozilish formati:

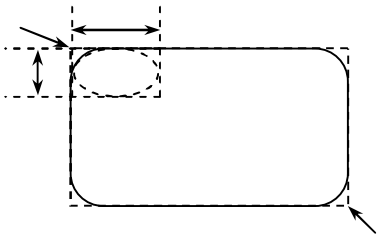
Ob’yekt.Canvas.RoundRec(x1, y1, x2, y2)

Bu yerda

ob’yekt – tasvir hosil bo’luvchi ob’yekt nomi;

x1, y1, x2, y2 – to’g’ri to’rtburchakning mos ravishda yuqori chap va quyi o’ng burchak koordinatalari;

x3,y3 – yumaloq hosil qilishda qo’llaniluvchi ellips o’lchamlari (17-rasm).



17-rasm.

Ko’pburchak.

Polygon uslubidan foydalanib ko’pburchak chizish mumkin. **Polygon TPoint** tipli massivni parametr sifatida qabul qiladi. Har bir massiv elementi o’zida ko’pburchakning bitta burchagi koordinatasi(x,y)ni saqlaydi. **Polygon** esa shu nuqtalarni ketma-ket to’g’ri chiziqlar bilan tutashtirib chiqadi.

Chiziqning ko’rinishi (rangi, qalinligi va turi) **Pen** ob’yekti bilan ifodalanadi.

Quyida uchburchak chizish uchun dastur qismi keltirilgan:

```
procedure TForm1.Button1Click(Sender:TObject);
var
  pol: array[1..3] of TPoint; //uchburchak nuktalari koordinatasi
Begin
  Pol[1].x := 10; Pol[1].y := 50;
  Pol[2].x := 40; Pol[2].y := 10;
  Pol[3].x := 70; Pol[3].y := 50;
  Form1.Canvas.Polygon(pol);
End;
```

Sektor.

Ellips yoki aylana sektorini hosil qilishda **Pie** uslubidan foydalaniladi. **Pie** ning umumiy yozilish formati:

Ob’yekt.Canvas.Pie(x1, y1, x2, y2, x3, y3, x4, y4);

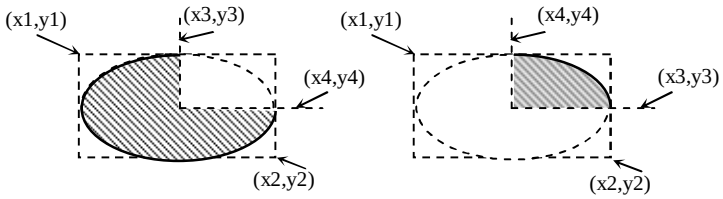
bu yerda

ob’yekt – yoy chiziluvchi ob’yekt nomi;

x1, y1, x2, y2 – hosil bo’luvchi sektorni davom ettirib hosil qilinuvchi ellips (aylana)ga tashqi chizilgan to’g’ri to’rtburchakning mos koordinatalari;

x3, y3 – sektorning boshlang’ich nuqtasi;

x4, y4 – sektorning tugash nuqtasi.

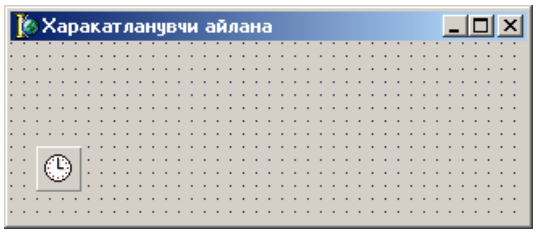


18 -rasm.

Multiplikatsiya.

Multiplikatsiya deyilganda odatda harakatlanuvchi yoki o’zgaruvchi rasmni tushuniladi. Oddiy holatlarda rasm harakatlanishi yoki o’zgarishi mumkin. Hosil qilingan rasm (chiziq, aylana, yoy va x.k.)larni siljitish juda oddiy: avval rasm hosil qilinadi, bir ozdan so’ng uni tozalanadi va yana yangitdan avvalgi joyidan boshqa yerda hosil qilinadi. Bunday almashtirish bir maromda davom ettirilsa, natijada tasvir oyna bo’ylab harakatlanayotganga o’xshaydi.

Quyidagi kichik dastur yordamida aylanani dastur oynasining chap chegarasidan o’ng chegarasiga qarab harakatlantirishimiz mumkin. -rasmda forma ko’rinishi keltirilgan.



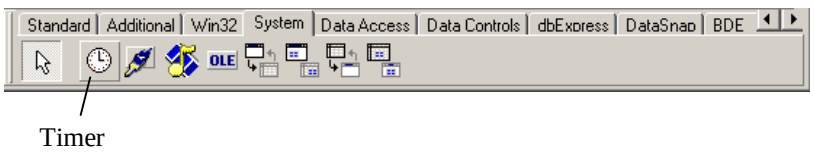
19 -rasm.

Dasturi

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
```

```
    procedure Timer1Timer(Sender: TObject);
   procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  x, y, dx: byte;
implementation
{$R *.dfm}
procedure Ris;
begin
  {aylanani ko`rinmas qilish}
  Form1.Canvas.Pen.Color := form1.Color;
  Form1.Canvas.Ellipse(x, y, x + 10, y + 10);
  X := x + dx;
  {aylanani yangi joyda xosil kilish}
  Form1.Canvas.Pen.Color := clBlack;
  Form1.Canvas.Ellipse(x, y, x + 10, y + 10);
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  ris;
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
  x := 0;
  y := 10;
  dx := 5;
  Timer1.Interval := 50;
  Form1.Canvas.Brush.Color := Form1.Color;
end;
end.
```

Asosiy ishni aylanani o’chirib yangi joyda hosil qiluvchi Ris prosedurasi bajaradi. Aylanani o’chirishni uning rangini forma rangiga o’zgartirish yo’li bilan amalga oshiriladi.



20-rasm.

Forma yoki dasturda e’tibor bergan bo’lsangiz vizual bo’lmagan komponent **Timer** (taymer)dan foydalandik. Uning yordamida harakatni vaqt bo’yicha amalga oshirilishi ta’minlangan. **Timer** komponenti komponentlar palitrasining **System** bo’limida joylashgan (20-rasm). **Timer** xususiyatlari quyidagi jadvalda keltirilgan:

Konstanta	Bo’yaluvchi soha tipi
Name	Komponent nomi
Interval	Millisekundlarda beriluvchi OnTimer gyenyeratsiyasi
Enabled	Ishga ruhsat berish. Qiymat true bo’lsa ruhsat beriladi, false bo’lsa berilmaydi.

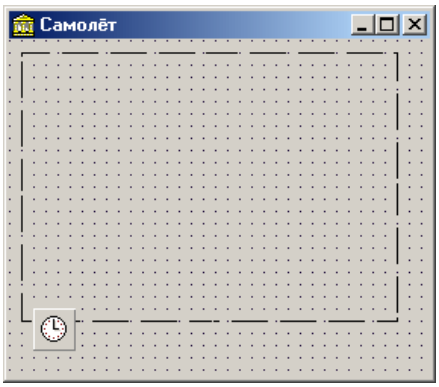
OnTimer xodisasi **Timer** komponentini ishga tushiradi. **OnTimer** vaqtli xodisasi millisekundlarda o’zgaradi va **Interval** xususiyatlariga mos keladi. **Enabled** xususiyati esa dasturda taymerni «ishga tushirish» yoki «to’xtatish» imkoniyatini yaratadi. Agar **Enabled *True*** (rost) bo’lsa **OnTimer** xodisasi ishlamaydi.

Bitli tasvirlardan foydalanish.

Yuqoridagi misolda tasvirni o’zimiz hosil qilib oldik. Endi esa qanday qilib bir murakkab tasvirni boshqa fonida harakatlanishini ko’rib o’tamiz. Masalan, shahar tasviri fonida samolyotni yurgizishni olaylik.

Suratni siljitish effyekti suratni bir nechta joyda vaqti-vaqti bilan qaytadan chizish usuli bilan tashkil qilinishi mumkin. Tasvirni yangi nuqtada chiqarishdan avval uni avvalgisi o’chiriladi. Suratni o’chirish to’liq fonni boshqatdan yoki faqat o’sha qismini chizish yo’li bilan amalga oshirilishi mumkin. Biz ko’rib o’tadigan dasturda ikkinchi yo’ldan foydalanamiz. Tasvir Image komponentining Canvas xususiyatida Draw uslubi bilan chiqariladi, tozalash esa fonning kerakli qismini nusxasini olish yo’li (**CopyRect** uslubi) bilan amalga oshiriladi.

Dastur forma ko’rinishi 21-rasmda keltirilgan. **Image** komponenti fonni chikarish uchun, **Timer** komponenti esa harakatni xosil qilish uchun foydalaniladi.



21-rasm. «Samolyot» dasturining forma ko’rinishi.

Dasturi

```
unit anim_;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;
type
  TForm1 = class(TForm)
    Image1: TImage;
    Timer1: TTimer;
    Procedure FormActivate(Sender:TObject);
    Procedure Timer1Timer(Sender:TObject);
    Procedure FormClose(Sender:Tobject; var Action:TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  Back, bitmap, Buf: Tbitmap; //fon, tasvir, bufyer
  BackRCT, bitmapRct, BufRct: Trect; //fon, tasvir, bufyer soxasi
  x, y: integer;   // tasvirning yukori chap burchak koordinatasi
  w, h: integer;  //  tasvir ulchami
implementation
{$R *.DFM}
Procedure TformActivate(Sender:TObject);
Begin
  Back := Tbitmap.Create;  //fon
  Bitmap := Tbitmap.Create;  //tasvir
  Buf := Tbitmap.Create;    //bufyer

  // yuklash va fonni xosil kilish
  Back.LoadFromFile('factory.bmp');
  Form1.Image1.Canvas.Draw(0, 0, Back);

  // harakatlanuvchi tasvirni yuklash
  bitmap.LoadFromFile('aplane.bmp');
  bitmap.TransParent := true;
  bitmap.TransParentColor := bitmap.Canvas.pixels[1, 1];

  // fon soxasi nusxasini saklash uchun bufyer tashkil etish
  w := bitmap.Width;
```

```
   h := bitmap.Height;
  Buf.Width := w;
  Buf.Height := h;
  Buf.Palette := Back.Palette;
  Buf.Canvas.CopyMode := cmSrcCopy;
  BufRct := Bounds(0, 0, W, H);
  X := -40;
  Y := 20;

  //saklanuvchi fon soxasini aniklaymiz
  BackRct := Bounds(x, y, w, h);

  // va uni saklaymiz
  Buf.Canvas.CopyRect(BufRct, Back.Canvas, backRct);
End;

Procedure TForm1.Timer1Timer(Sender: TObject);
Begin
  // fonni tiklaymiz (bufyerdan) rasmni yuk kilamiz
  Form1.Image1.canvas.Draw(x, y, Buf);
  X := x + 2;
  If x > form1.Image1.width then x := -40;

  //saklanuvchi fon soxasini aniklaymiz
  BackRct := Bounds(x, y, w, h);

  // uning nusxasini saklaymiz
  Buf.Canvas.CopyRect(BufRct, Back.Canvas, BackRct);

  //rasmni chikaramiz
  Form1.Image1.canvas.Draw(x, y, bitmap);
End;

Procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
Begin
   Back.Free;
   Bitmap.Free;
   Buf.Free;
End;
end.
```

1. **Pen** va **Brus** xusiyatlarining asosiy farqini tusuntirin.
2. Bitli tasvir deganda nimani tushunasiz.

12-Ma`ruza. Dasturchi komponentini yaratish. Komponent xolatini o’rganish va bazadan sinf tanlash. Komponent modulini yaratish. (2 soat)

O’quv modul birliklari:

1. Komponent xolatini o’rganish
2. Bazadan sinf tanlash.
3. Komponent modulini yaratish.

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Dasturchi komponenti to’g’risida kengroq tushunchaga ega bo’ladi.
2. Komponent modulini yarata oladi

Dasturni komponentini yaratish

Delphi dasturchiga o’zi uchun komponent yaratish imkonini beradi. Uni komponentlar palitrasining tarkibiga joylash va undan boshqa komponentlar singari foydalanish mumkin bo’ladi.

Komponentni yaratish quyidagi bosqichlarda amalga oshiriladi:

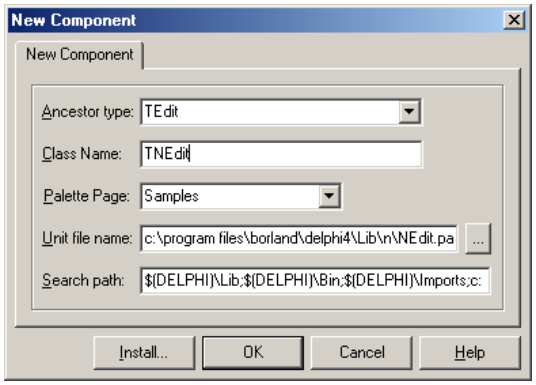
- komponentni aniqlash va baza sinfini tanlash;
- komponent modulini yaratish;
- komponentni tekshirish;
- komponentni komponentlar paketiga qo’shish.

Quyida komponent yaratishni **NEdit** misolida ko’rib chiqamiz. **NEdit** komponentidan kasr sonlarni kiritish va tahrirlash uchun foydalaniladi.

Komponentni aniqlash va baza sinfini tanlash

Yangi komponent yaratishdan avval uning bajarishi lozim bo`lgan vazifalari o`ragnib chiqiladi va uning imkoniyati qaysi komponent imkoniyatiga yaqinroq ekanligi ko`riladi. Izliangan komponent topilganidan so`ng, u yangi komponent yaratish uchun bazaviy qilib olinadi.

Komponent modulini yaratish.



22-rasm. New Component mulokot oynasi

Komponent modulini yaratish uchun **Component** menyusidan **New Component** buyrug’ini tanlash lozim. Xosil bo’lgan **New Component** muloqot darchasida (22-rasm) yaratiluvchi komponent haqidagi ma’lumotlarni kiritish kerak bo’ladi.

Ancestor type maydonida komponent uchun tegishli bo’lgan baza tipi joylashadi. Biz ko’rib chiqayotgan yangi komponent **Edit** (tahrir maydoni) bazasiga tegishli bo’lgani uchun uning tipini **TEdit** deb belgilaymiz.

Class Name maydoniga komponent sinfi nomi kiritiladi, bizda TNEdit.

Palette Page maydonida yaratilayotgan komponentni komponentlar palitrasining qaysi bo`limiga joylashtirish kerakligi ko’rsatiladi. Bu yerga yangi nom kiritilsa, komponent qo’shilgandan so’ng yangi bo`lim ochiladi.

“Ok” tugmasi bosilgandan keyin shu proyekt uchun Delphi-modul kod oynasi xosil bo’ladi. Modul matni quyida keltirilgan:

```
unit NEdit;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
  TNEdit = class(TEdit)
private
    { Private declarations }
protected
    { Protected declarations }
public
    { Public declarations }
published
    { Published declarations }
end;
procedure Register;
implementation
procedure Register;
begin
  RegisterComponents('Samples', [TNEdit]);
end;
end.
```

Yuqoridagi dasturga bir nechta o’zgartirishlar kiritamiz:

```
unit NEdit;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
```

```
   TNEdit = class(TCustomEdit)
  private
     { Private declarations }
     FNumb: single;
     function GetNumb: single;
     Procedure SetNumb(value: single);
  protected
    { Protected declarations }
    procedure kyepress(var Key: Char); override;
  public
    { Public declarations }
  published
    { Published declarations }
    constructor Create(AOwner: TComponent); override;
    property Numb: single
           read GetNumb
           write SetNumb;
  end;
procedure Register;
implementation
procedure Register;
begin
  RegisterComponents('Samples', [TNEditt1]);
end;
constructor TNEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
end;
function TNEdit.GetNumb: single;
Begin
  try
    Result := StrToFloat(Text);
  except
      on EConvertError do
      Begin
         Result := 0;
         Text := '';
      end;
  end;
end;
Procedure TNEdit.SetNumb(Value: Single);
begin
 FNumb := Value;
 Text := FloatToStr(value);
end;
```

```
 Procedure TNEdit.Keypress(var key: char);
begin
 case key of
   '0'..'9', #8, #13: ;
   '-': if Length(Text) <> 0 then key := #0;
   else
      if not((key  = DecimalSeparator) and
      (Pos(DecimalSeparator, text) = 0)) then key := #0;
 end;
 inherited KeyPress(key);
end;
end.
```

TNEdit sinfiga **Numb** xususiyati qo'shilgan bo'lib, u o'zida sonlarni saqlaydi va tahrirlash maydonida joylashadi. **Numb** xususiyati qiymatini saqlash uchun **Fnumb** maydonidan foydalaniladi. **GetNumb** funksiyasi **Fnumb** maydonidan foydalanish imkonini yaratadi, **SetNumb** prosedurasi esa xususiyat qiymatini o'rnatish uchun ishlatiladi.

TNEdit sinf konstruktori avval **Text** xususiyati qiymatini o'zida mujassamlashtirgan **TEdit** sinf konstruktorini chaqiradi, so'ngra unga **Numb** xususiyat qiymatini o'rnatadi.

TNEdit.KeyPress prosedurasi **NEdit** komponentining tarkibida bo'lib, klaviaturadan kiritilgan belgilarni tekshiradi. Agarda ruhsat etilgan belgilar kiritilgan bo'lsa **OnKeyPress** xususiyatiga qayta ishlash uchun uzatiladi.

13-Ma`ruza. Komponentni o'rnatish. Komponentni tekshirish. Komponent palitralarini sozlash. (2 soat)

O'quv modul birliklari:

1. Komponentni o'rnatish.
2. Komponentni tekshirish.
3. Komponent palitralarini sozlash.

Aniqlashtirilgan o'quv maqsadlari:

Talaba ushbu mavzuni to'la o'zlashtirgandan so'ng:

1. Komponentni o'rnatishni biladi
2. Komponentni tekshira oladi
3. Komponent palitralarini sozlashni o'zlashtiradi.

Komponentni o'rnatish.

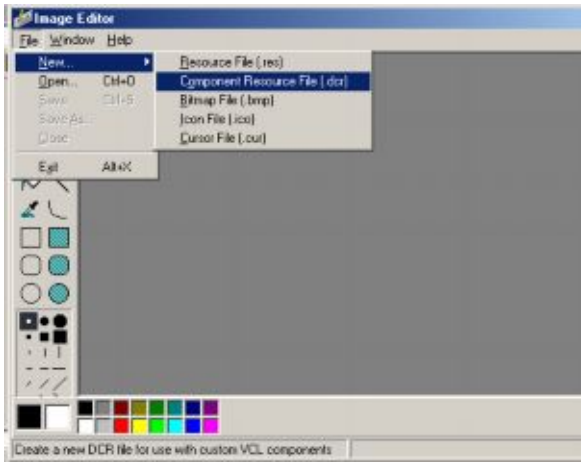
Komponent belgisi komponentlar palitrasida hosil bo'lishi uchun **Delphi**ning komponentlar paketlaridan (packages) biriga qo'shilgan bo'lishi lozim.

Komponentlar paketi – bu ***.DPK** kengaytmali fayl (Delphi Package File). Masalan, dasturchi tomonidan yaratiluvchi komponentlar **dclusr40.dpk** paketida joylashadi.

Komponentni paketga qo'shish vaqtida Delphi komponent modulidan va komponent **resurs** faylidan foydalanadi. Ularning nomi bir xilda bo'lib, bir papkada joylashgan bo'lishi kerak. **Resurs** fayl kengaytmasi ***.DCR** (Dynamic Component Resource) bo'ladi.

Resurslar faylini yaratish.

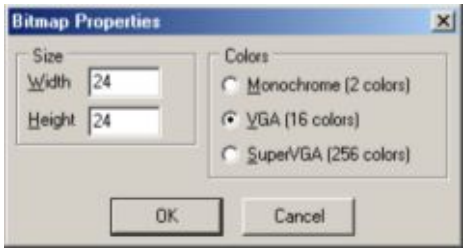
Komponentning resurs faylini yaratish uchun **Image Editor** utilitidan foydalaniladi. U **Tools** menyusidagi **Image Editor** buyrug'i yordamida ishga yuklanadi. Yangi resurs faylini yaratish uchun **File** menyusidan **New** buyrug'i tanlanadi, hosil bo'lgan ro'yhatdan yaratiluvchi fayl tipi tanlanadi. U **Component Resource File** (23-rasm).



23– Rasm. Yaratiluvchi fayl tipini tanlash.

Natijada **Untitled.dcr** komponenti resurslar fayli oynasi ochiladi. **Image Editor** muloqot oynasining menyusida yangi **Resource** bo'limi xosil bo'ladi. Endi **Resource** menyusidan **New** buyrug'i tanlanadi va xosil bo'lgan ro'yhatdan yaratiluvchi resurs **Bitmap** tipi tanlanadi.

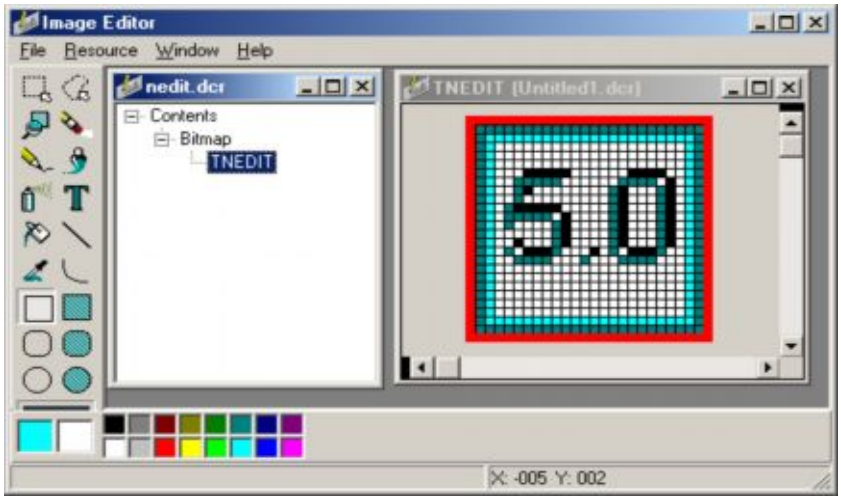
Shundan so'ng **Bitmap Properties** (24-rasm) muloqot oynasi ochiladi. U yerda komponent belgisi bitli tasvirining harakteristikasi o'rnatilishi kerak: **Size** (O'lchami) – 24x24 piksyelda, **Colors** (Ranglar soni) – 16 va **Ok** tugmasi bosiladi.



24-rasm. Bitmap Properties muloqot oynasi

Yuqoridagi amallar bajarilishi natijasida yangi resurs Bitmap1 nomli bitli tasvir xosil bo'ladi. Sichqonchaning chap tugmachasini resurs (Bitmap1) nomi sirtida ikki marta bosish bilan kerakli rasmni chizish mumkin bo'lgan bitli tasvir tahrirchi oynasini xosil qilamiz. Tasvir xosil qilinganidan so'ng, uni komponent moduli nomi bilan bir xil nomda saqlanadi.

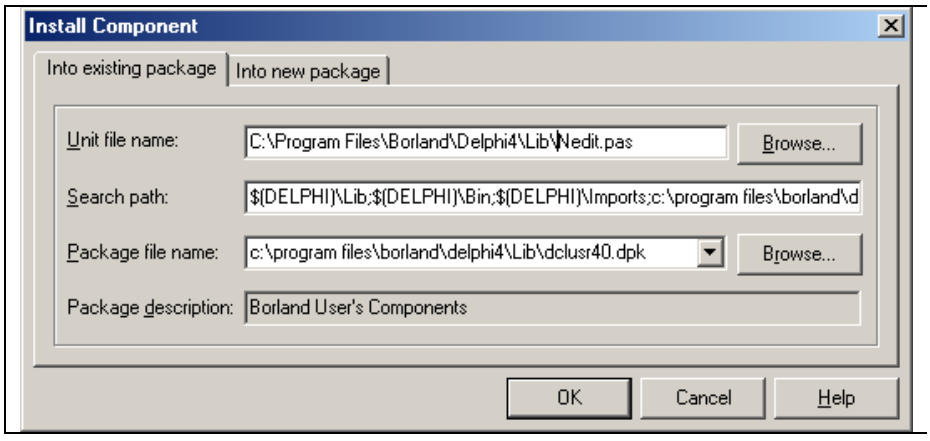
25-rasmda **Image Editor** oynasining ko'rinishi ko'rsatilgan. Oynaning chap tarafida **TNEdit** (NEdit.dcr) komponenti resurs fayli, o'ng tarafida esa bitli tasvirni xosil qiluvchi tahrirchi oynasi joylashgan.



25-rasm. Image Editor muloqot oynasi.

O'rnatish

Komponent resurs faylini yaratilgandan so'ng, yangi komponentni o'rnatish mumkin. Buning uchun **Component** menyusidan **Install Component** buyrug'i tanlanadi. Xosil bo'lgan **Install Component** muloqot oynasining maydonlari to'ldiriladi (26-rasm).



26-rasm. Install Component muloqot oynasi

Unit file name (Modul fayli nomi) maydonida komponentning modul fayli nomini kiritiladi yoki **Browse** tugmasi yordamida izlanadi.

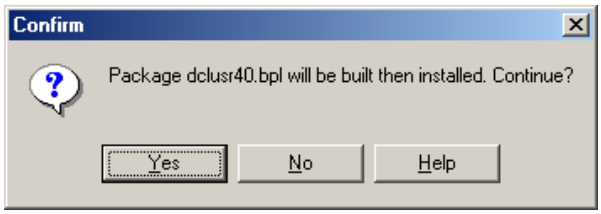
Search path (Izlanuvchi yo'l) maydoni komponentni o'rnatish vaqtida kerakli fayllarni izlashi mumkin bo'lgan kataloglar ro'yhatini o'zida joylashtiradi va ular nuqtali vergul bilan ajratib yoziladi.

Eslatma: ***Komponentning resurs fayli*** *Search path* ***maydonida joylashgan bo'lishi shart, aks holda bazaviy komponyet belgisi qo'yib yuboriladi.***

Package file name maydoni o'rnatilgan paket nomini o'zida saqlashi lozim. Agarda o'zgartirilmasa **dclusr40.dpk** paketiga joylaydi.

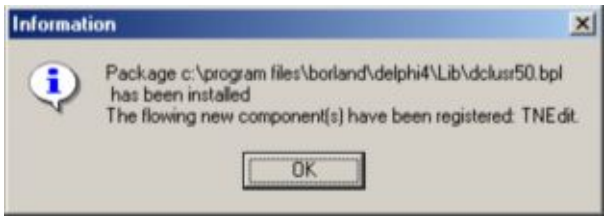
Package Description maydoni o'zida paket nomini saqlaydi. **Dclusr40.dpk** paketi uchun quyidagi matn kiritilgan: **borland user's Components**.

Barcha maydonlar to'ldirilib **OK** tugmachasi bosilganidan so'ng, o'rnatish jarayoni boshlanadi. Avval yangi maydon xosil bo'layotganligi xaqida ma'lumot beruvchi **Confirm** (27-rasm) oynasi xosil bo'ladi.



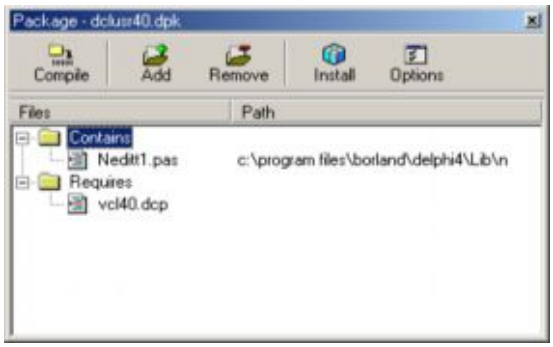
27-rasm. Yangi paket xosil bo'layotganligini ogohlantiruvchi muloqot oynasi.

Yes tugmachasi bosilganidan keyin paketni o'rnatish jarayoni davom etadi va 28-rasmda ko'rsatilgan yangi komponent muvaffaqiyatli o'rnatilganligi haqidagi ma'lumot oynasi xosil bo'ladi.



28-rasm. O'rnatish muvaffaqiyatli yakunlanganligi xaqidagi ma'lumot oynasi.

Paketdagi komponent o'rnatilganidan so'ng, **Package** (Komponentlar paketining tahrirchisi) paketda joylashgan komponentlar ro'yhatini ko'rsatuvchi dialogli oynasi xosil bo'ladi (29-rasm).



29-rasm. Koponyentlar paketini tahrirlash oynasi.

Yuqoridagi amallar bajarilganidan so’ng, komponentlarni o’rnatish jarayoni yakunlanadi.

O’rnatish vaqtidagi xatoliklar.

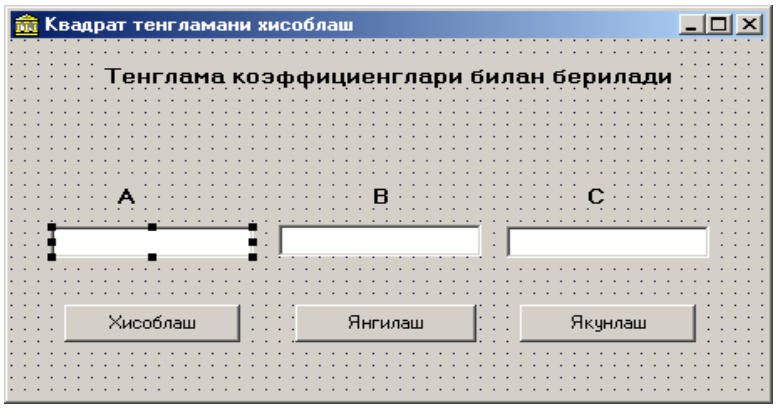
Yangi komponentni o’rnatish (qayta o’rnatish) vaqtida turli xatoliklar uchrashi mumkin.

Bunday holda Delphi quyidagi xatolikni chiqaradi: **The package alredy contins unit named…** (… nomli modul avval o’rnatilgan) va o’rnatish jarayoni to’xtatiladi. Bu xatolikni yo’qotish uchun paketda joylashgan komponentni o’chirish lozim va o’rnatish jarayoni qaytadan amalga oshiriladi.

Komponentni tekshirish

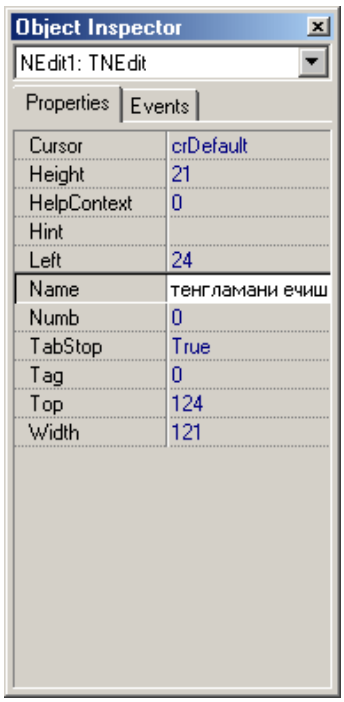
Paketlar to’plamiga komponent qo’shilganidan so’ng, uning belgisi komponentlar palitrasidan tekshiriladi. Agarda u yerda komponent belgisi bo’lsa demak uni yangi komponent sifatida ishlatish mumkin bo’ladi.

NEdit komponentining ishchi holatini "Kvadrat tenglamani yechish" misolida ko’rish mumkin. Forma ko’rinishi 30-rasmda keltirilgan.



30-rasm. Kvadrat tenglamani yechish formasi.

NEdit komponentini **Edit** komponentidan farqi unga **Numb** xususiyati qo’shilgan bo’ladi. Uni **Object Inspector** oynasida ko’rishimiz mumkin 31-rasm.



31-rasm. Object Inspector oynasidagi TNEdit komponentining xususiyati.

Quyidagi dastur matnida "Kvadrat tenglamani yechish" moduli ifodalangan (TestofComp_u.pas):

```
unit TesfComp;

interface

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

type
  TForm1 = class(TForm)
    NEdit1: TNEdit;
    NEdit2: TNEdit;
    NEdit3: TNEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
    procedure Button3Click(Sender: TObject);
   procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

// kvadrat tenglamani hisoblash
procedure SqRoot(a, b, c: TNEdit1; Roots: TLabel);
        {a, b, c - tenglama koeffisiyentlari}
Var
  d: Real;    {Diskriminant}
  x1, x2: Real; {Tenglama ildizlari}
Begin
    If a.Numb = 0 Then
    Begin
        Roots.Font.Color := ClRed;
        Roots.Caption :='Ikkinchi daragali noma`lum'+#13
        +'Nolga teng.';
    End
    Else
    Begin
        {tenglamani yechish}
        {Diskriminantni hisoblash}
        d := Sqr(b.Numb) – 4 * a.Numb * c.Numb;
        If d < 0 Then
        Begin
           Roots.Font.Color := ClRed;
           Roots.Caption := 'Diskriminant noldan kichik'+#13
           +'Tenglamaning ildizi yuq.';
        End
        Else
        Begin
            x1 := (b.Numb + Sqrt(d)) / (2 * a.Numb);
            x2 := (-b.Numb + Sqrt(d)) / (2 * a.Numb);

            {Natijani oynaga chiqarish}
```

```
            Roots.Font.Color := ClBlack;
            Roots.Caption := 'Tenglamaning ildizlari'
            + #13 + 'x1= '+FloatToStr(x1)
            + #13 + 'x2= '+FloatToStr(x2);
        End;
    End;
End;

//"Hisoblash" tugmasi bosilganida
procedure TForm1.Button1Click(Sender: TObject);
begin
    If  (NEdit1.Text <> '') or
        (NEdit2.Text <> '') or
        (NEdit3.Text <> '')
    Then SqRoot(NEdit1, NEdit2, NEdit3, Label1)
    Else ShowMessage('Barcha koeffisiyentlarni kiriting');
end;

// "Yangilash" tugmachasi bosilganida
procedure TForm1.Button2Click(Sender: TObject);
begin
    NEdit1.Text := '';
    NEdit2.Text := '';
    NEdit3.Text := '';
    Label1.Font.Color := ClBlack;
    Label1.Caption := '';
    NEdit1.SetFocus;
end;

// "Yakunlash" tugmachasi bosilganida
procedure TForm1.Button3Click(Sender: TObject);
begin
    Close;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
    NEdit1.SetFocus;
end;
end.
```

Dasturchi yaratgan komponentlar qaysi papkada joylashadi.

14-Ma`ruza. Delphi da ma`lumotlar bazasi bilan ishlash. Ma`lumotlar ombori. Ma`lumotlar omborining tuzilishi. (2 soat)

O’quv modul birliklari:

1. Ma`lumotlar ombori.
2. Ma`lumotlar omborining tuzilishi.
3. Ma’lumotlar ombori bilan ishlash komponentlari

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Delphi da ma`lumotlar bazasining umumiy tuzilishini o’rganadi.
2. Delphi da ma`lumotlar bazasi bilan ishlash komponentlari bilan tanishadil

Ma’lumotlar bazasi

Foydalanuvchi nuqtau nazari bo’yicha ma’lumotlar bazasi bu ma’lumotlar bilan ishlashni ta’minlovchi dastur. Bunday dasurlar ishga yuklanganda, doimgidek jadval hosil bo’ladi, foydalanuvchi uni ko’rib chiqishi va qiziqtirgan ma’lumotlarni ko’rishi mumkin. Agar sistema ruxsat bersa, u holda ma’lumotlar bazasiga o’zgartirish kiritish mumkin: yangi ma’lumot kiritish yoki keraksizini o’chirib tashlash.

Dasturchi nuqtai nazari bo’yicha ma’lumotlar bazasi bu o’zida turli ma’lumotlarni saqlovchi fayllar to’plamidir. Foydalanuvchi uchun ma’lumotlar bazasini qayta ishlash bilan dasturchi ma’lumotlar fyli bilan ishlashni ta’minlovchi dastur yaratadi.

Hozirgi vaqtda lokal (dBASE, FoxPro, Access, Paradox) va masofadagi (Interbase, Oracle, Sysbase, Infomix, Microsoft SQL Server) ma’lumotlar bazalarini yaratish va foydalanish imkonini berivchi yetarlicha dasturiy sistemalar mavjud.

Delphi tarkibiga esa turli sistemalarda yaratilgan (dBase dan toki Infomix va Oracle) ma’lumotlar fayllari bilan ishlay oluvchi dasturlarni yaratish imkonini beradigan komponentlar kiritilgan. Shuningdek, Delphi foydalaniuvchiga Borland Database Desktop utiliti yordamida turli formatlardagi ma’lumotlar bazasi fayllarini yaratish imkonini ham beradi.

Ma’lumotlar bazasining klassifikatsiyasi

Foydalaniluvchi dasturlar ma’lumotlarni joylashishiga va ularni bir nechta foydalanuvchilar o’rtasida bo’linish usuliga bog’liq ravishda ma’lumotlar bazasi (MB) lokal va global MB ga bo’linadi.

Lokal MB

Lokal MB ning ma’lumotlari bitta (lokal) qurilmada, ya’ni kompyuter diski yoki tarmoq diskida (tarmoqda ishlayotgan boshqa bir kompyuter diski) joylashadi.

Ma’lumotlarni bir yoki bir nechta kompyuterlar o’rtasida taqsimlashni (ma’lumotlardan foydalanishga ruhsat berish) ta’minlash uchun lokal MB larda

fayllarni blokirovkalash usuli qo'llaniladi. Bu usulda, agar ma'lumotlar bir foydalanuvchi tomonidan foydalanilayotgan bo'lsa, boshqa boshqa bir foydalanuvchi bu ma'lumotlar bilan ishlay olmaydi, ya'ni ma'lumotlar uning uchun yopiq, blokirovkalangan.

Paradox, dBase, FoxPro va Access – lokal MB lardir.

Masofadagi (uzoqdagi) MB

Masofadagi MB larning ma'lumotlari (fayllari) uzoqdagi kompyuterda joylashadi. Masofadagi MB bilan ishlash dasturi ikki qismdan tashkil topadi: (kliyent) mijoz va server. Dasturning mijoz qismi foydalanuvchi kompyuterida ishlaydi va server dastrulari bilan muvofiqlikni ta'minlaydi.

Dasturlarning server qismi masofadagi kompyuterda ishlaydi va mijoz dasturlaridan so'rovlarni qabul qiladi, ularni bajaradi va ma'lumotlarni jo'natadi. So'rovlar SQL (strukturalashtirilgan so'rovlar) tili buyruqlari yordamida amalga oshiriladi. Masofadagi serverda ishlayotgan dastur shunday loyihalashtirilganki, ma'lumotlarga bir vaqtnin o'zida bir nechta foydalanuvchi murojaat qilishi mumkin. Shuning uchun ma'lumotlarga ruhsat (доступ)ni ta'minlash uchun fayllarni blokirovkalash mexanizmining o'rniga tranzaksiya mexanizmi qo'llaniladi.

Tranzaksiya – bu ma'lumotlar uzatilishi bilan ular ustida bajarilishi shart bo'lgan amallar ketma-ketligidir. Agar tranzaksiyani tashkil etuvchilari amallardan birida xatolik yuz bersa, u holda barchasi yangidan qayta bajariladi. Shunday qilib, tranzaksiya mexanizmi apparatli uzilishlardan himoyalanishni ta'minlaydi. Shuningdek, ma'lumotlar'lumotlarga ko'pfoydalanuvchilik ruhsatini berish imkoniyatini yaratadi.

Masofadagi Mblar bilan ishlovchi dasturlarni yaratish – murakkab masala. Uni yechish dasturchidan chuqur bilim va dastur tuzish bo'yicha yuqori malaka talab qiladi. Shuning uchun ushbu kitobda masofadagi MB larni yaratish to'g'risida to'xtalmaymiz.

MB strukturasi

MB – bir qator me'zonlari bo'yicha tartiblangan, bir jinsli bo'lgan axborotlar to'plamidir. MB "qog'oz" yoki kompyuter ko'rinishida taqdim qilinishi mumkin.

"Qog'oz"li MB ga yaqqol misol qilib kutubxona katalogini, ya'ni kitoblar to'g'risida axborotlar berilgan qog'oz kartochkalarni keltirish mumkin. Bu bazadagi axborotlar bir jinsli (faqat kitob to'g'risida ma'lumotlar'lumot berilgan) va tartiblangan (masalan, kartochkalar avtorlarning familiyalari bo'yicha alfavit shaklida).

Telefon qo'llanmasi yoki poyezdlar harakatining jadvali ham bunga yaqqol misol bo'lishi mumkin.

Kompyuterli MB esa o'zida axborotlarni saqlovchi fayl (yoki o'zaro bog'langan fayllar to'plami) ko'rinishida bo'ladi.

MB yozuvlardan tashkil topadi. Xar bir yozuv bir nusxada axborotni o'zida saqlaydi. Masalan, "O'zbekistonning obidalari" MB ning har bir yozuvi faqat bittadan obida to'g'risida to'liq ma'lumotni saqlaydi.

Yozuv maydonlardan tashkil topadi. Xar bir maydon bitta nusxaning faqat bitta tavsifi (xarakteristika)ni o'zida saqlaydi. Masalan, "O'zbekistonning obidalari" MB si quyidagi maydonlardan tashkil topgan: "Obida", "Me'mor" va "Tarixiy ma'lumot", bu yerda "Obida", "Me'mor" va "Tarixiy ma'lumot" – maydon nomlari. Bu maydonlardagi ma'lumotlar'lumotlar aniq bir obidani tasvirlaydi. E'tibor qilish kerakki, xar bir yozuv bir xil mayonlardan tashkil topgan. Ba'zi bir maydonlar to'ldirilmasligi mumkin, lekin yozivga tegishliligicha qoladi.

MB ni qog'ozda jadval ko'rinishida taqdim qilish qulay (17.1-rasm). Jadvalning har bir satri yozuv bo'lib, jadval kataklari esa maydondir. Shuning uchun jadval ustunining sarlavhasi – maydon nomi, satr raqami esa – yozuv nomeri bo'ladi.

Kompyuterdagi MB ham odatda ekranga jadval ko'rinishida chiqariladi.

№	Obida	Me'mor	Tarixiy ma'lumot
1	Tuman Oqo majmuasi	usta Shayx Muhammad ibn Xoji Bandgir	XVX asrda Amir Temurning xotini Tuman Oqo qurdirgan. U mulozimxona, masjid va maqbaradan iborat.
2	Usto Ali Maqbarasi	Usta Ali	O'rta asrning 60-70 yillarida qurilgan.

17.1-rasm. MB ni jadval ko'rinishida berilishi

Delphida MB modeli

Xar bir jadval alohida faylda saqlanadi. Lekin, MB bilan jadvalni bir narsa deb qarash mumkin emas, chunki, ko'p hollarda bitta yozuvning maydoni bir nechta jadvallarga taqsimlangan bo'lib, turli fayllarda joylashadi.

Oddiy holatlarda MB bilan ishlovchi dasturlar uchun axborot manbai bo'lib barcha jadvallar xizmat qiladi. Lekin, foydalanuvchini MB dagi barcha ma'lumotlar qiziqtirmaydi, balki, ularning qaysidir bir qismigina kerak bo'ladi xolos. U faqat so'roviga mos tushgan yozuvlarnigina tanlaydi va ko'rib chiqadi. Shuning uchun barcha ma'lumotlarni o'zida mujassamlashtirgan MB modelida "so'rov" tushunchasi kiritilgan bo'lib, u MB yozuvlarini tanlash imkonini yaratadi.

MB taxallusi (Псевдоним)

Dasturchi MB bilan ishlash dasturini yaratish vaqtida MB fayllari qaysi disk yoki katalogda joylashganligini bilishi shart emas. Masalan, foydalanuvchi MB ni C: yoki D: disklarning birorta katalogiga joylashtirishi mumkin. Shuning uchun MB ning joylashgan o'rni haqidagi ma'lumotlarni dasturga uzatishda muammo tug'iladi.

Delphida yuqoridagi muammo MB taxallusidan foydalanish yo'li bilan hal qilinadi.

Taxallus (**Alias**) – MB joylashgan haqiqiy to'liq katalogning qisqacha nomi. Masalan, C:\data\UzbObida katalogining taxallusi Obida bo'lishi mumkin. MB bilan ishlovchi dastur Mblardan foydalanishi uchun ularning to'liq nomi emas, balki taxallusidan foydalanadi. Ma'lumotlardan foydalanishga ruhsat berishni

ta’minlovchi dastur Borland Database Engine (BDE) kutubxonasiga ulanadi va u yerda tizimda ro’yxatga olingan, barcha taxalluslar haqida ma’lumotga ega bo’lgan konfiguratsiya faylidan foydalanadi. MB taxallusi BDE Administrator utiliti yordamidayaratilgan (ro’yxatga olingan) bo’lishi mumkin. Bu utilit taxallusga tegishli kataloglarni o’zgartirish imkonini beradi.

MB ni yaratish

MB – ma’lumotlar joylashgan fayllar (jadvallar) to’plamidir. Qoidaga ko’ra, MB bitta katalogda joylashgan bir nechta jadvallardan tashkil topadi. MB uchun katalog oddiy usul bilan, masalan, provodnik yordamida yaratiladi. Jadvalni esa Delphi tarkibiga kiruvchi Borland Database Desktop utiliti yoki SQL-so’rovlar yordamida yaratish mumkin.

Shunday qilib, MB ni yaratish jarayonini quyidagi ketma-ketlikda amalga oshiriladi:

1. Katalog yaratish
2. Taxallus yaratish.
3. Jadval yaratish.

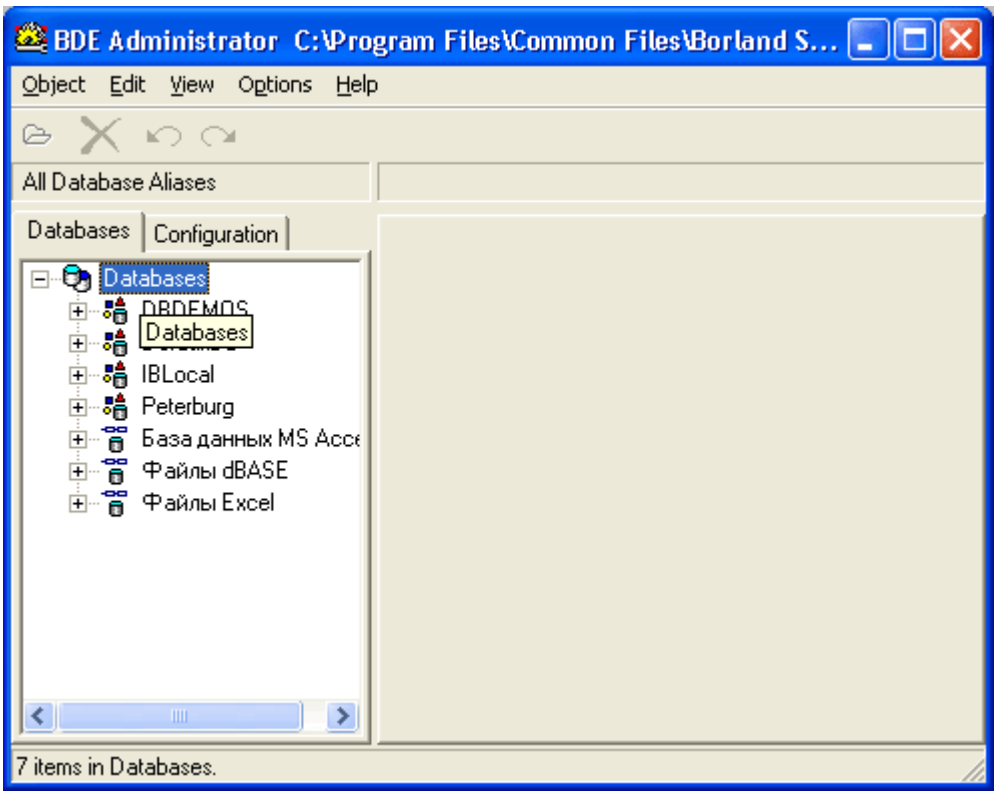
Katalog yaratish

MB fayllari uchun katalog (papka) oddiy usul bilan, masalan Provodnik yordamida yaratiladi. Odatda lokal MB ning fayllari MB bilan ishlovchi dastur katalogi ichida, alohida katalogda joylashadi.

Taxallus (псевдоним) yaratish

MB taxallusi Delphining BDE Administrator utiliti yordamida yaratiladi. Utilitni ishga tushirish uchun **Пуск | Программы | Borland Delphi** 7 **| BDE Administrator** buyrug’ini tanlash kerak.

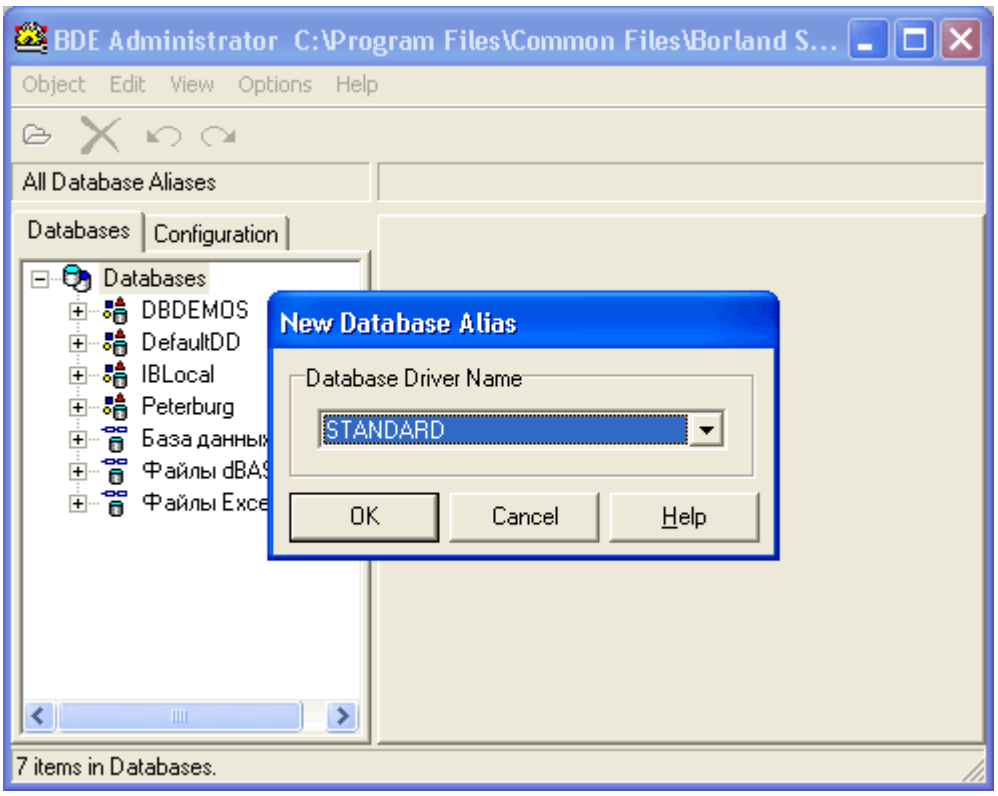
BDE Administrator ishga yuklangandan so’nggi oyna ko’rinishi 17.2-rasmda berilgan.



17.2-rasm. BDE Administrator oynasi

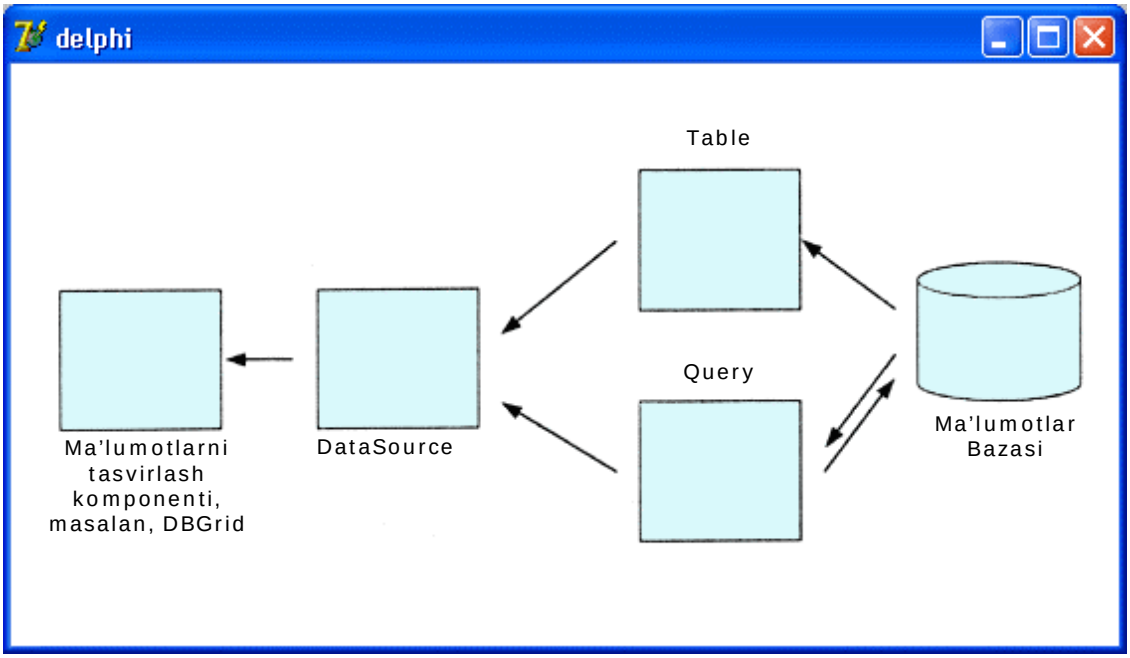
Oynaning chap qismining **Database** bo'limida ushbu kompyuterda ro'yxatga olingan barcha taxalluslar keltirilgan. Yangi taxallusni yaratish uchun **Object** menyusidan New buyrug'i tanlanadi. Ochilgan New **Database Alias** (MB ning yangi taxallusi) oynasidagi **Database Driver Name** ro'yxatidan yaratiluvchi MB uchun drayverni, ya'ni MB ning tipini tanlash kerak (17.3-rasm).

Taxallusni yaratish vaqtida, odatda **STANDARD** (default driver) drayveri taklif qilinadi, bu esa Paradox formatidagi jadvallardan foydalanish imkonini beradi.



17.3-rasm. New Database Alias muloqot oynasi

Drayverni tanlab, Ok tugmasini bosgandan keyin taxalluslar ro’yxatiga yangi element qo’shiladi (17.4-rasm).



17.4-rasm. Yangi taxallusni ro’yxatga olish

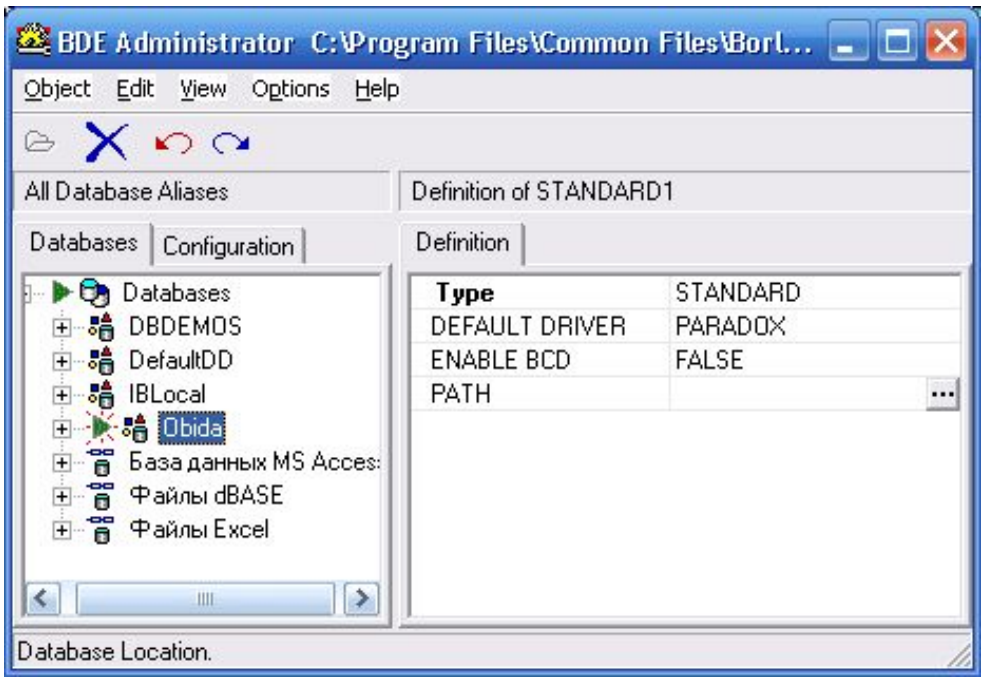
Shundan so’ng avtomatik tarzda yaratilgan taxallus nomini o’zgartirib taxallusi yaratilayotgan MB fayllariga yo’lni ko’tsatish kerak.

Taxallusning nomini Windows dagi oddiy usul bilan o’zgartirish mumkin: taxallus nomi usida sichqonchaning o’ng tugmasi bosiladi, hosil bo’lgan menyudan

Rename (qayta nomlash) buyrug'i tanlanadi va oshilgan oynaga yangi nom kiritiladi. MB fayllariga yo'l ko'rsatish uchun **Definition** bo'limining **Path** maydoniga klaviatura yoki **Select** oynasidan foydalanib to'liq yo'l kiritiladi.

17.5-rasmda **BDE Administrator** oynasining "O'zbekistonning tarixiy obidalari" MB ci uchun Obida taxallusi yaratilgandan keyingi holati tasvirlangan.

Yaratilgan taxallus konfiguratsiya fayli (Idapi.cfg) da ro'yxatga olinishi uchun **Object** menyusidan **Apply** (tasdiqlash) buyrug'ini tanlash kerak. Hosil bo'lgan **Confirm** oynasida konfiguratsiya fayliga o'zgarishlarni saqlash kerakligini tasdiqlash lozim.



17.5-rasm. Taxallusni yaratish natijasi

Jadvallarni yaratish

MB ni yaratish vaqtida asosiy jarayon – bu ma'lumotlarni yozuvlar maydoni bo'yicha taqsimlashdir. Ma'lumki, ma'lumotlar maydonlar bo'yicha turlicha taqsimlanishi mumkin.

Masalan, O'zbekisonning tarixiy obidalari haqida ma'lumot quyidagi maydonlardan tashkil topgan yozuvlar ko'rinishida bo'lishi mumkin: "Obida" va "Tarixiy ma'lumot" yoki "Obida", "Me'mor", "Yili" va "Tarixiy ma'lumot".

Birinchi variantda "Obida" maydoni obidaning nomini saql;aydi, masalan GO'ri Amir, "Tarixiy ma'lumot" maydoni esa qolgan barcha ma'lumotlarni saqlaydi. Shuning uchun foydalanuvchi o'zini qiziqtirgan tarixiy obida haqidagi ma'lumotni faqat nomi bo'yichagina topish mumkin xolos. Ikkinchi variantda esa obida, me'mor nomlari va yana bir qator xususiyatlari bo'yicha ham opish mumkin. Quyidagi qoidani kiritish mumkin: agar MB dan foydalanish vaqtida ma'lumotlarni tanlash bir qator me'zonlari bo'yicha amalga oshirilishi lozim bo'lsa, u holda bunday imkoniyatni ta'minlay oladigan ma'luomtlarni alohida maydonlarga ajratish kerak.

Yozuvlar maydoni aniqlangandan so’ng maydonlarni jadvallar yoyish kerak. Oddiy MB larda barcha maydonlar bitta jadvalda joylashadi. Murakkab MB larda esa maydonlar bir nechta jadvallarga taqsimlanadi, jadvallararo bog’liqlikni ta’minlovchi bir qator qo’shimcha ma’lumotlar kiritiladi.

Eslatma.

O’zaro bog’langan bir nechta jadvallardan tashkil topgan MB relyatsion MB deyiladi. Relyatsion MB da jadvallardagi ma’lumotlar bir xilligidan qochish maqsadida asosiy ma’lumotlarga bir qator xizmatchi ma’lumotlar qo’shiladi. Relyatsion MB larni tashkil qilish bo’yicha ma’lumotlar ushbu kitobda berilmagan.

MB yozuvlarining strukturasi aniqlanganidan so’ng bevosita jadvallarni yaratishga kirishish mumkin. Delphining **Database Desktop** utiliti yordamida jadval yaratiladi.

Database Desktop utiliti MB bilan ishlashda barcha zaruriy amallarni bajarish imkonini beradi. U turli formatdagi (Paradox, Dbase, MSAccess) MB jadvallarini yaratish, ko’rish va modifikatsiyalashni ta’minlaydi.

Bundan tashqari utilit ma’lumotlarni so’rovlar tuzish yo’li bilan tanlash imkonini ham beradi.

Yangi jadvalni yaratish uchun Tools menyusidan Database Desktop buyru’ini tanlash bilan Database Desktop ishga tushiriladi. Hosil bo’lgan Database Desktop utility oynasidagi File menyusidan New buyrug’i tanlanadi. Paydo bo’lgan ro’yxatdan esa yaratiladigan fayl tipi – Table ni tanlanadi. Keyin hosil bo’lgan Create Table muloqot oynasidan esa yaratiladigan jadval tipi tanlanadi (odatda Paradox7 tipi tavsiya etiladi). Natijada Create Paradox 7 Table muloqot oynasi ochiladi, bu yerda jadval yozuvlarining strukturasini belgilash mumkin. Jadvalning har bir maydoni uchun nom, tip, agar zarur bo’lsa maydon o’lchami berilishi lozim. Maydon nomi ma’lumotga murojaat qilish uchun qo’llaniladi. Maydon nomi Field name ustuniga kiritiladi, nom kiritishda lotin alifbosidagi 25 tadan ko’p bo’lmagam belgilardan bo’sh joy tashlamay foydalanish mumkin.

Maydon tipi unga joylashtiriluvchi ma’lumot tipi bo’yicha aniqlanadi. Maydon tipi Type ustuniga kiritiladi.

Maydon tiplari va ularga mos keluvchi konstantalar 17.1-jadvalda keltirilgan.

17.1 jadval. Maydon tiplari.

Tip	Konstanta	izoh
Alpha	A	Belgili satr. Satrning maksimal uzunligi qiymat oralig’i 1-255 bo’lgan Size xarakteristikasi bilan aniqlanadi
Number	N	10-30 – 10308 oraliqdagi sonli qiymat
Money	$	Pul farmatidagi son. Son raqamlari guruh razryadlarini ajratgichi yordamida ajratiladi. Shuningdek pul birligi belgisi ham chiqadi.
Short	S	-32767 - 32767 oraliqdagi butun son.
Long Integer	I	-2147483648 – 2147483647 oraliqdagi butun son.
Date	D	Sana

Time	T	Vaqt
Time stamp	@	Vaqt va sana
Memo	M	Ihtiyoriy uzunlikdagi belgili satr. Memo tipidagi maydon matnli axborotlarni saqlash uchun foydalaniladi. Maydon o'lchami (1-240) jadvalda nechta belgi saqlanishini aniqlaydi. Qo;gan belilar jadval faylining nomiga mos keluvchi faylda saqlanib, fayl kengaytmasi *.mb bo'ladi.
Formatted Memo	F	Ihtiyoriy uzunlikdagi belgili satr (Memo tipiga o'xshash). Shriftning tipini va o'lchamini, rasmiylashtirish tipi va beli ranglarini belgilash imkoniyatlariga ega.
Graphic	G	Grafika
Logical	L	Mantiqiy qiymat "Rost" (True) yoki "Yolg'on" (False)
Auto-increment	+	+ Butun son. Jadvalga navbatdagi yozuv qo'shilganda maydonga avvalgi yozuvdagi maydon qiymatidan 1 songa ko'p qiymat yoziladi.
Bytes	Y	Ikkilik ma'lumotlar.
Binary	B	Ikkilik ma'lumotlar. Ushbu tipdagi maydonda DataBase Desktop interpretatsiya qila olmayigan ma'lumotlar saqlanadi. Memo maydoni ma'lumotidek bu ma'lumotlar ham jadval faylida joylashmaydi. Binary tipli maydon o'zida audio ma'lumotlarni saqlaydi.

Maydon tipini aniqlovchi konstantalar klaviaturadan kiritilishi mumkin yoki Type ustunida sichqonchaning o'n tugmasini yoki "probel"ni bosish bilan hosil qilinuvchi ro'yhatdan tanlash bilan ham belgilanishi mumkin.

Bir yoki bir nechta maydonni kalit sifatida belgilash mumkin. Kalitli maydonlar javaldagi yozuvlarni mantiqiy ketma-ketlik tartibini aniqlaydi. Masalan, A ga belgili (Alpha tipli) Fam (familiya) maydonini kalitli qilib belgilansa, u holda jadvalagi familiyalar yozuvlar ekranga alfavit bo'yicha tartiblangan holda chiqariladi. Agar maydon kalitli qilib belgilanmasa, u holda yozuvlar qanday tartibda kiritilgan bo'lsa, shunday tartibda chiqariladi. E'tibor berish kerakki, bir kalitli maydonda bir hil ma'lumot bo'lmaslii kerak. Shuning uchun qaralayotgan misolimizda Fam (familiya) va Name (ismi) maydonlari kalitli maydon bo'lishi kerak. U holda jadvalga bir hil familiyalarni ham kiritish mumkin bo'ladi. Lekin, ismi bir xil bo'lgan familiyadoshlarni ham kiritib bo'lmaydi. Shuning uchun, odamlar ro'yhatini kiritish uchun mo'ljallangan jadvallarda Pasp (pasport) maydonini kalitli qilib belgilash maqsadga muvofiq bo'ladi.

Maydonni kalitli qilib belgilash uchun Key kolonkasida sichqoncha ugmasi ikki marta tez bosiladi.

Agar, belgilangan maydon kiritilishi shart bo’lgan ma’lumot uchun ajratilgan bo’lsa, u holda **Required Field** ga bayroqcha o’rnatish kerak. Masalan, Fam (familiya) maydonini to’ldirish shart bo’lsa, u vaqtda Tel (telefon) maydoniga o’xshashlari bo’sh qoldirilishi mumkin.

Agar maydonga yoziluvchi qiymat aniq bir oraliqda joylashishi kerak bo’lsa, u holda maydonning **Minimum value** (minimal qiymat) va **Maximum value** (maksimal qiymat) lari uchun oraliq qiymatlarini berish mumkin. **Devault value** maydoni esa qiymatni o’z holicha berilishini ta’minlaydi.

Picture maydoni maydonga kiritiluvchi ma’lumotlarni to’g’riligini nazorat qiluvchi shablonni belgilash imkonini beradi. Ushbu shablon oddiy va maxsus belgilar ketma-ketligidan iboratdir. Maxsus belgilar ro’yxtau 17.2-jadvalda sanab o’tilgan.

Ma’lumotlarni maydonning maxsus belgiga mos keluvchi pozitsiyasiga kiritish vaqtida faqat shablonning ushbu belgiga to’g’ri keluvchi belgilarigina hosil bo’ladi.

Masalan, shablonning pozitsiyasida # belgisi turgan bo’lsa, u holda ushbu belgiga to’g’ri keluvchi belgi - faqat raqamlarni kiritish mumkin. Agar pozitsiyada oddiy belgi turgan bo’lsa, u holda ushbu pozitsiyaga ma’lumot kiritish vaqtida avtomatik tarzda ko’rsatilgan belgi hosil bo’ladi.

Masalan, A tipli (belgili satr) Tel maydoni telefon raqamlarini saqlash uchun mo’ljallangan bo’lsin, va MB bilan ishlovchi dastur telefon raqamlari odatiy ko’rinishda berilishini talab qiladi. Bunday holatda Picture maydoniga quyidagi shablonni kiritish kerak bo’ladi: ###-##-##. Tel maydoniga ma’lumot kiritishda faqat raqamlardan foydalaniladi (boshqa tugmalarni bosish qabul qilinmaydi) va uchinchi hamda beshinchi raqamlardan keyin avtomatik tarzda chiziqcha (defis) qo’yiladi.

15-Ma`ruza. Ma`lumotlar omborini yaratish. Ma`lumotlar bazasini ko’rish va tahrirlash. (2 soat)

O’quv modul birliklari:

1. Ma`lumotlar omborini yaratish.
2. Ma`lumotlar bazasini ko’rish va tahrirlash.
3. Sodda bazalar ushun dastur yaratish

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Ma`lumotlar omborini yarata oladi.
2. Ma`lumotlar bazasini ko’rish va tahrirlash dasturini yarata oladi
3. Sodda bazalar ushun dastur yarata oladi

Ma’lumotlar omborining tuzilishi

Dasturchilar uchun ***ma’lumotlar ombori*** – bu fayl ko’rinishidagi turli xildagi axborotlar jamlanmasidir. Ma’lumotlar ombori uchun dastur tuzish esa axborotlarni qayta ishlashni ta’minlaydi. Bunday dasturlar ishga yuklanganida foydalanuvchi

undan o’zini qiziqtirgan ma’lumotlarni olishi, yangi yozuvlarni unga qo’shib keraksizini o’chirib tashlashi mumkin.

Bugungi kunda quyidagi: lokallashtirilgan (dBASE, FoxPro, Access, Pradox) va masofaviy ma’lumotlar ombori (Interbase, Oracle, Sysbase, Infomix, Microsoft SQL Server) bilan ishlashga mo’ljallangan dasturiy tizimlar mavjud.

Delphi dasturlash tilida barcha turdagi ma’lumotlar ombori (dBASE dan totib Interbase va Oracle gacha) bilan ishlash uchun komponentlar yaratilgan.

Lokallashtirilgan ma’lumotlar ombori

Lokallashtirilgan ma’lumotlar ombori (ma’lumotlar fayli) bu – yakka (lokalli) qurilma, ya’ni kompyuter diski yoki setli disk (setda ishlovchi kompyuter diski) axborotlar majmuasidir.

Lokallashtirilgan ma’lumotlar omborida ma’lumotlar bilan aniq va kafolatli ishlashni ta’minlash maqsadida uslublar (ma’lumotlar ombori bilan ishlovchi) yaratilgan bo’lib, ularning yordamida ma’lumotlar ombori himoyalab qo’yiladi. Bunda ma’lumotlar omborini qayta ishlash faqatgina bir foydalanuvchiga ruhsat etiladi. Qolgan foydalanuvchilar esa uni ko’rishi mumkin, lekin o’zgartira olmaydi.

Masofaviy ma’lumotlar ombori

Masofaviy ma’lumotlar ombori(*Удаленная база данных*)dagi ma’lumotlar (fayllar) yuqori imkoniyatli kompyuter(server)larda joylashgan bo’ladi. *Shuni ham ta’kidlab o’tish lozimki, yuqori imkoniyatli kompyuterlarning kataloglari setli kompyuter disklari kabi tasavvur qilinmaydi*.

Yuqori imkoniyatli ma’lumotlar ombori bilan ishlovchi dasturlar ikkiga, ya’ni kliyent va server dasturlarga bo’linadi. Kliyentlarga mo’ljallangan dastur qismida kliyent kompyuterdan yuqori imkoniyatli kompyuterdagi server dasturga bog’lanib ma’lumotlar omborini tahrirlashga ruhsat olish va kerakli ma’lumotlarni so’rab olish (zapros) yo’lga qo’yiladi. Dasturning server qismi esa yuqori imkoniyatli kompyuterda joylashgan bo’lib, so’ralgan ma’lumotlarni kliyent dasturga jo’natishni ta’minlaydi. So’rovlarni o’zida mujassamlashtirgan SQL (Structured Query Language) tilining buyruqlari yordamida ma’lumotlarni so’rash amalga oshiriladi.

Serverdagi yuqori imkoniyatli ma’lumotlar omborini barcha foydalanuvchilar tahrirlashi mumkin (albatta server ruhsati bilan).

Ma’lumotlarni himoya qilish uchun tranzaktsiya myexanizmi ham ishlaydi. ***Tranzaktsiya*** – bu qo’shimcha imkoniyat bo’lib, apparat uzilishlari sodir bo’lganda havfsizlikni yo’lga qo’yishdir. Masalan, kliyent dastur server dasturdan so’rov olayotgan vaqtda setda xatolik paydo bo’lsa, tranzaktsiya qaytadan so’rovni amalga oshiradi.

Shuni ham ta’kidlash joizki, yuqori imkoniyatli ma’lumotlar ombori uchun dastur tuzish juda ham qiyin jarayon bo’lib, dasturchidan yuqori saviyadagi bilim va mahorat talab qiladi.

Ma'lumotlar omborining umumiy ko'rinishi – bu turli xil tipdagi va bir xil ketma-ketlikdagi yozuvlar jamlanmasidir. Uning tarkibini o'rganish uchun guruh talabalari haqidagi ma'lumotlarni ko'rib chiqamiz.

Faraz qilaylik ma'lum bir guruh talabalarining familiyasi, ismi, tug'ilgan yili va yashash manzili haqida ma'lumot beruvchi ombor yaratish lozim bo'lsin. Biz omborni yaratish uchun dastlab quyidagi jadvalni tuzib olamiz.

№	Familiyasi va ismi	Tug'ilgan yili	Yashash manzili
0.	Kamalov Zokor	1980	Norin t., Xorazim qish., Do'stlik k., 19-uy
1.	Raxmanov Nodir	1981	Pop t., Nayman q., Bog' k., 2-uy
2.	Mamatov Komil	1980	Namangan sh., Navoiy k., 145-uy
3.	Murodov Akram	1979	CHust t., YOrqo'rg'on q. Oybyek k., 12-uy
N	……	……	……

Jadvaldagi «familiyasi, ismi, tug'ilgan yili va yashash manzili» deb yozilgan kataklar ***maydon*lar**, ularning ostidagi ma'lumotlar ***yozuv*lar**, tartib ***raqam*lar** esa yozuvlarning ma'lumotlar omboridagi joylashgan joyini bildiradi.

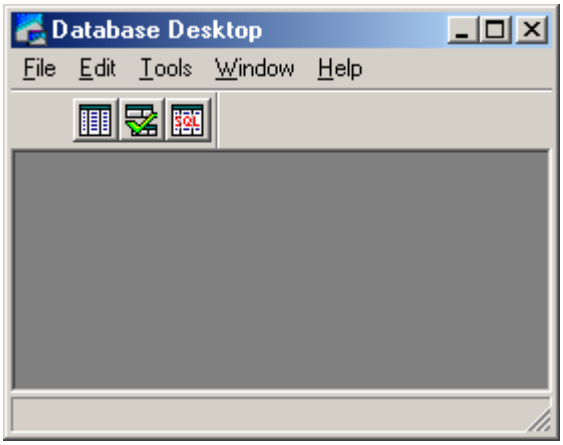
Ma'lumotlar ombori(ma'lumotlar fayli)ning nomi va undagi maydon nomlarini lotin harflarida yozish maqsadga muvofiqdir.

Ma`lumotlar omborini tashkil qilish

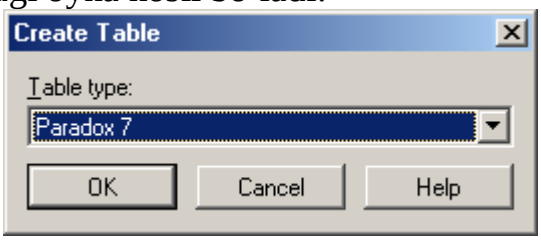
Ma`lumotlar omborini tashkil qilish uchun **Database Desktop** dasturidan foydalanamiz. Uni ishga tushirishning ikki yo`li mavjud bo`lib ular quyidagilar:

- Delphining yuqori menyusidan **Tools/ Database Desktop** buyrug`i tanlanadi;
- "Программы / Borland Delphi 4 / Database Desktop" buyrug`i tanlanadi.

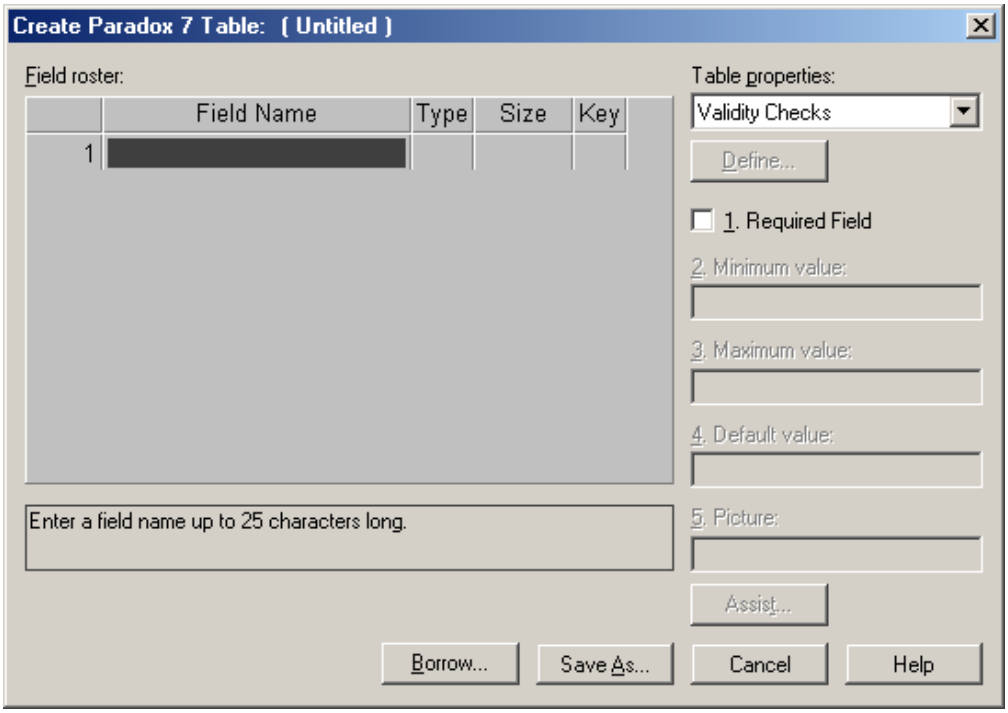
Database Desktop dasturi formasining ko`rinishi quyidagicha:



Bu dasturning yuqori menyusidan “File / New / Table” tanlanadi. So`ng quyidagi oyna hosil bo`ladi:



Yuqoridagi oyna ma`lumotlar omborining turini tanlash oynasidir. Delphida ishlovchilar uchun “Paradox 7” ma`lumotlar omborini tashkil qilish maqsadga muvofiqdir. Bu oynada “OK” tugmasi tanlanadi, shundan so`ng quyidagi oyna hosil bo`ladi:



Yuqorida tasvirlangan oynaning vazifasi ma`lumotlar omborini tashkil qilish bo`lib, u yerda “Field roster: ” (Maydonlar ro`xati) bo`limi bo`lib, unda jadval keltirilgan. Quyida ustunlarga ta`rif beramiz:

- Field Name (Maydon nomi) – bu ustunga maydon nomlari kiritiladi. Maydon nomlari lotin xarflaria yozilishi shart;
- Type – yozilgan maydonning tiplari kiritiladi;
- Size – maydon yozuvi uchun ajratiladigan joy;
- Key – kalit. Uning vazifasi ma`lumotlar omborini bir-biriga bog`lashdir.

Quyidagi jadvalda maydonlarning tiplari keltirilgan:

Tip	Konstanta	Izox	Hajmi
Alppha	A	Belgilar satri	**1..255**

Number	N	No`mer	**$10^{-307}..10^{308}$**
Money	$	Valyuta	
Short	S	Butun son	**-32767..32767**
Long Integer	I	Butun son	**-2147483648.. 2147483647**
Date	D	Sana	
Time	T	Vaqt	
Timestamp	@	Sana va vaqt	
Memo	M	Satrlar to`plami	**1..240**
Formated Memo	F	Satrlar to`plami. Shrift va ranglarni ham o`zida saqlaydi	**1..240**
Graphic	G	Grafika	
Logical	L	Mantiqiy qiymat	
Autoincrement	+	Yozuvlarni avtomatik nomerlash	
Bytes	Y	Ikkilik ma`lumotlar	
Binary	**B**	**Ikkilik ma`lumotlar. audio – ma`lumotlari**	

Misol sifatida guruhdagi talabalar xaqida ma`lumot saqlovchi omborini tashkil qilishni ko`rib chiqamiz:

Aytaylik ma`lumotlar ombori quyidagi maydonlardan tashkil topgan bo`lsin:

FIO – Talabaning familiyasi, ismi va sharfi;

T_Yil – Tugilgan yili;

Jinsi – Jinsi;

Tugil_Joy – Tug`ilgan joy;

Yash_Joy – Xozirda yashayotgan manzili;

Urt_Ball – Fanlardan to`plagan o`rtacha bali.

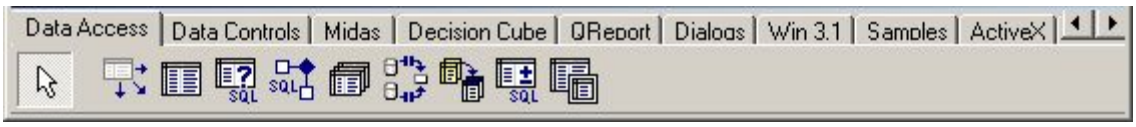
Maydonlar ro`yhatini quyidagi jadvalga kiritamiz:

Field Name	Type	Size	Key
FIO	A	35	*****
T_Yil	D		
Jinsi	L		
Tugil_Joy	A	45	
Yash_Joy	A	45	
Urt_Ball	**N**		

Maydonlar to`ldirib bo`linganidan so`ng, ma`lumotlar omborini saqlash uchun “Save As…” tugmachasidan foydalanamiz. Saqlangan ma`lumotlar ombori faylining kengaytmasi *.DB ko`rinishida bo`ladi (Biz **talaba.db** deb nomlab saqlashimiz mumkin).

Ma`lumotlar ombori bilan ishlash

Delphida ma`lumotlar ombori bilan ishlash uchun komponentlar palitrasida **Data Access** va **Data Controls** sahifalari mavjud. Ularning yordamida ma`lumotlar omborini tashkil qilish, tahrirlash va bir-biriga bog`lash mumkin. Ushbu sahifalarning ko`rinish quydagicha bo`ladi:

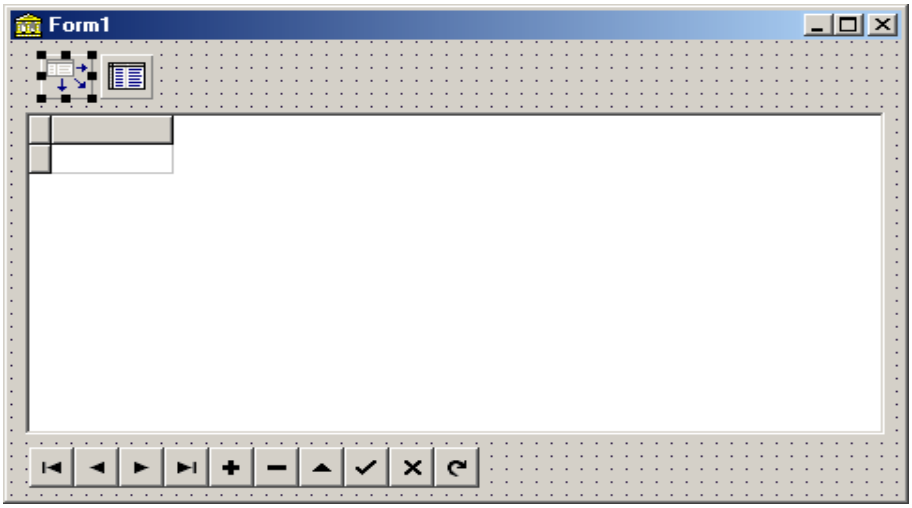


Ma`lumotlar omborini tashlik qilish uchun asosan 4 xil komponentdan foydalanish mumkin. Ular **Table, DataSource, DBGrid** va **DBNavigator** lardir. Endi quyida ularga qisqacha ta`rif beramiz:

- **Table** – ma`limotlar omborini aktivlashtirish;
- **DataSource** – aktivlashtirilgan ma`lumotlar omborini boshqa ob`ektlar bilan bog`lash (Masalan, DBEdit);
- **DBGrid** - ma`lumotlar omborini jadval ko`rinishida formaga chiqarish;
- **DBNavigator** - ma`lumotlar omborini tahrirlash uchun ishlatiladi.

Yuqorida yaratilgan (talaba.db) ma`lumotlar ombori sirtida turli amallar bajarishni ko`rib chiqaylik.

1. Komponentlar politrasidan yuqoridagi 4 ta (Table, DataSource, DBGrid va DBNavigator) komponentni tanlab formaga joylaymiz. Komonentlar joylangan forma ko`rinishi quyidagicha bo`ladi:



2. **Table** ob`ekti tanlanadi, undagi **DatabaseName** xususiyati(Objekt Inspector dagi)ga ma`lumotlar omborining yo`li ko`rsatiladi (Masalan, C:\BASE\). So`ngra, **TableName** xususiyatidan kerakli ma`lumotlar ombori tanlanadi. Undan so`ng **Active** xususiyati **True** holiga keltiriladi.

3. **DataSource** ob`ektini tanlab, undagi **DataSet** xususiyatidan **Table1** tanlanadi.

4. DBGrid ob`ektining **DataSource** xususiyatidan **DataSource1** tanlanadi.

5. **DBNavigator** ob`ektining ham **DataSource** xususiyatidan **DataSource1** tanlab qo`yiladi.

Yuqoridagi amallarni bajarganimizdan so`ng dasturni ishlatib ko`rishimoz mumkin.

	**DBEdit** komponentini **Teble** komponentining maydonlaridan biriga bog`lang.

16-Ma`ruza. Delphida yordamchi (Help) tizimini yaratish. (4 soat)

O’quv modul birliklari:

1. Yordamchi ma’lumotlar faylini yaratish.
2. Microsoft Help Workshop dasturi
3. Yordam tizimi oynasining xarakteristikasi
4. Loyihani kompilyatsiya qilish

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. Yordamchi ma’lumotlar faylini yarata oladi
2. Microsoft Help Workshop dasturida ishlashni o’rganadi
3. Yordam tizimi oynasining xarakteristikasi bilan tanishadi
4. Loyihani kompilyatsiya qila oladi

Yordam tizimi

Har bir dastur foydalanuvchiga dasturda qanday qilib ishlash haqida ma’lumot beruvchi o’zining yordam tizimiga ega bo’lishi kerak.

Windows muhitida ishlovchi yordam tizimi dasturi **WinHelp** dasturidan foydalanib foydalanuvchi so’rovi bo’yicha yordamchi ma’lumotlarni chiqaruvchi fayllar to’plamiga ega. Yordamchi ma’lumotlar joylashgan HLP-fayllar yordam tizimining asosiy elementlari hisoblanadi. Sodda holatda yordam tizimi dasturi yagona HLP faylga ega bo’ladi.

Yordam tizimini (HLP-faylni) Delphi dasturi bilan birga o’rnatiluvchi **Microsoft Help Workshop** dasturi yordamida yaratish mumkin. HLP-faylni yaratish uchun RTF-fayl ko’rinishida keltirilgan boshlang’ich yordamchi ma’lumotlar kerak bo’ladi.

Yordam tizimini yaratish jarayonini quyidagi ikki qadam ketma-ketligida keltirish mumkin:

1. Yordamchi ma’lumotlarni tayyorlash (yordamchi ma’lumotlar faylini yaratish).
2. Yordamchi ma’lumotlar faylini yordam tizimi fayliga aylantirish.

Yordamchi ma’lumotlar fayli

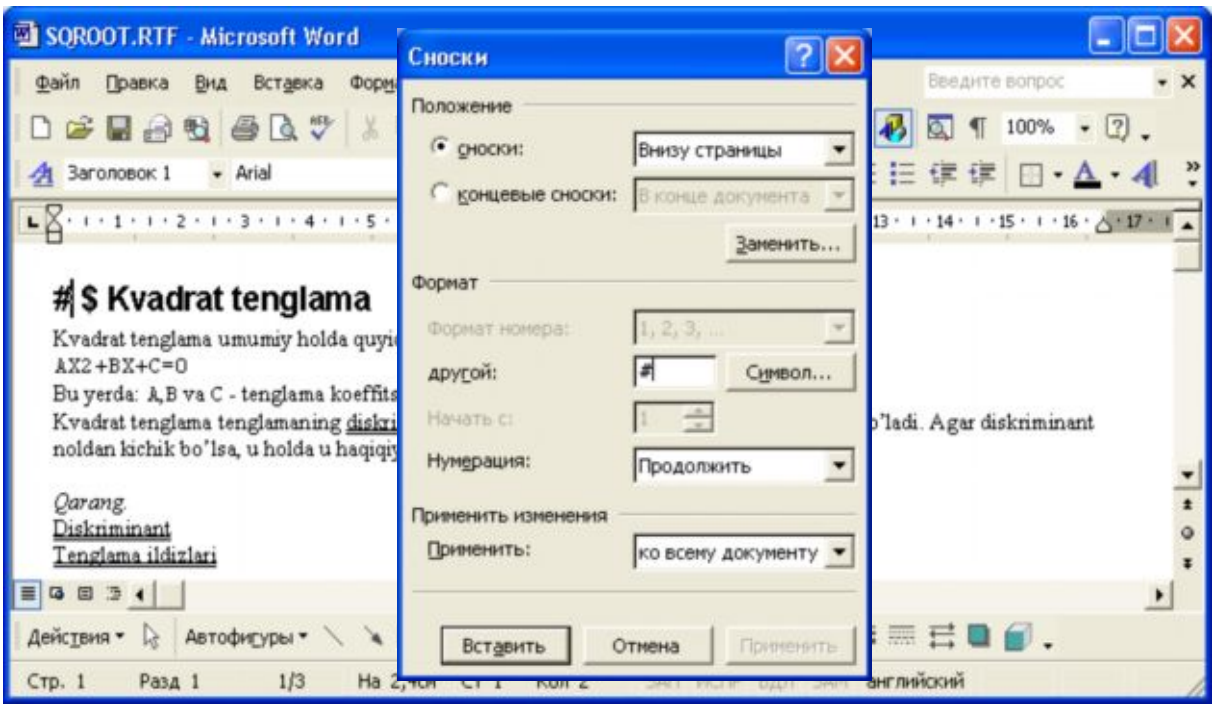
Yordamchi ma’lumotlar fayli RTF-fayl ko’rinishida keltiriladi. Yordamchi ma’lumotlarning RTF-faylini Microsoft Word dasturi yordamida yaratish mumkin. Dastlab yordamchi ma’lumotlar matnini terib olish kerak. Bunda har bir bo’lim matni alohidadan sahifada joylashgan bo’lishi kerak (“Sahifani ajratish” belgisi bilan yakunlangan bo’lishi kerak).

Ma’lumotlar terib bo’lingandan keyin, yordamchi ma’lumotlar bo’limlarining sarlavhalarini snoskalardan (14.1-jadval) foydalanib belgilash kerak (snoskalar yordamchi tizim kompilyatori tomonidan RTF-faylni HLP-faylga o’tkazish jarayonida ishlatiladi).

14.1-jadval.

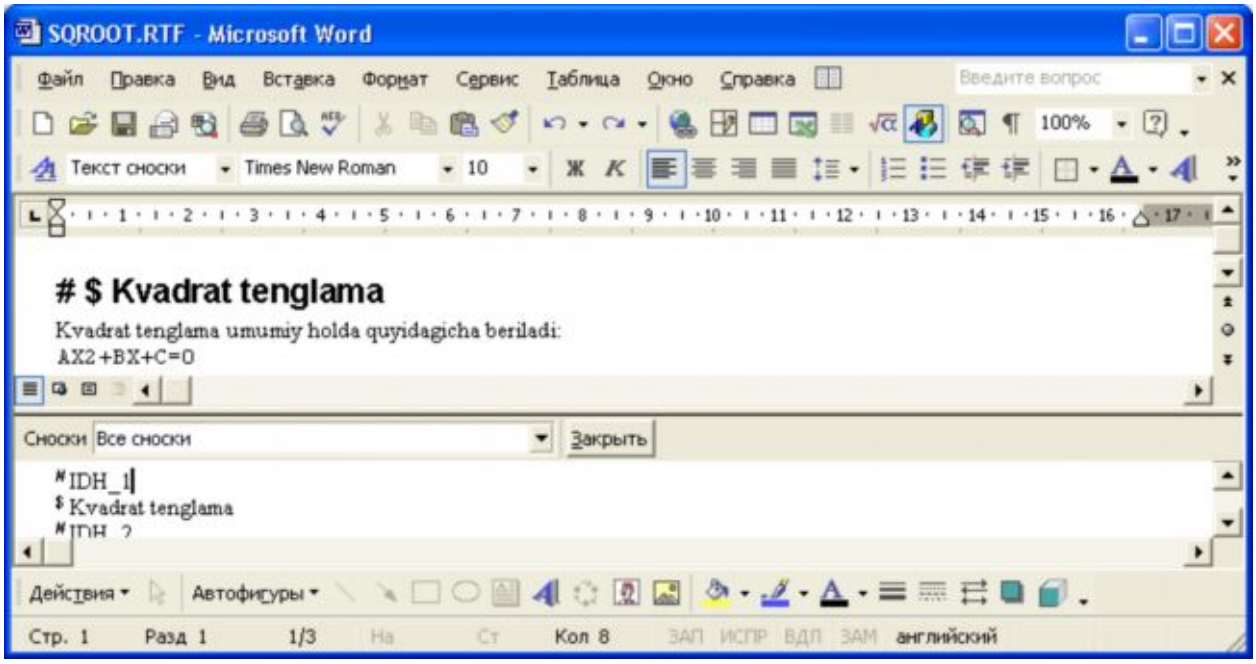
Snoska	Vazifasi
#	Ushbu snoska bilan belgilangan bo’limga boshqa bo’limlardan o’tish uchun ishlatiladi.
$	Ushbu snoska bo’lim nomini yordam tizimida izlash natijasining ro’yhatiga va bo’limlar ro’yhatiga chiqarish uchun ishlatiladi.
к	Izlash vaqtida kalit so’z sifatida ishlatiladi.

Bo’lim sarlavhasini snoska bilan belgilab qo’yish uchun sarlavhaning birinchi harfi oldiga kursorni qo’yib, **Вставка** menyusidan **Сноска** buyrug’ini tanlash kerak. Natijada **Сноски** muloqot oynasi ochiladi (14.1-rasm). Bu oynada **Другой** maydoniga "#" belgisini qo’yib **OK** tugmasini bosish kerak.



14.1-rasm.

Natijada xujjatga “#” snoskasi qo’yiladi va xujjatning quyi qismida snoska matnini kiritish oynasi ochiladi. Bunda snoska belgisidan keyin bo’limni belgilash kalit so’zini kiritish kerak (14.2-rasm).



14.2-rasm.

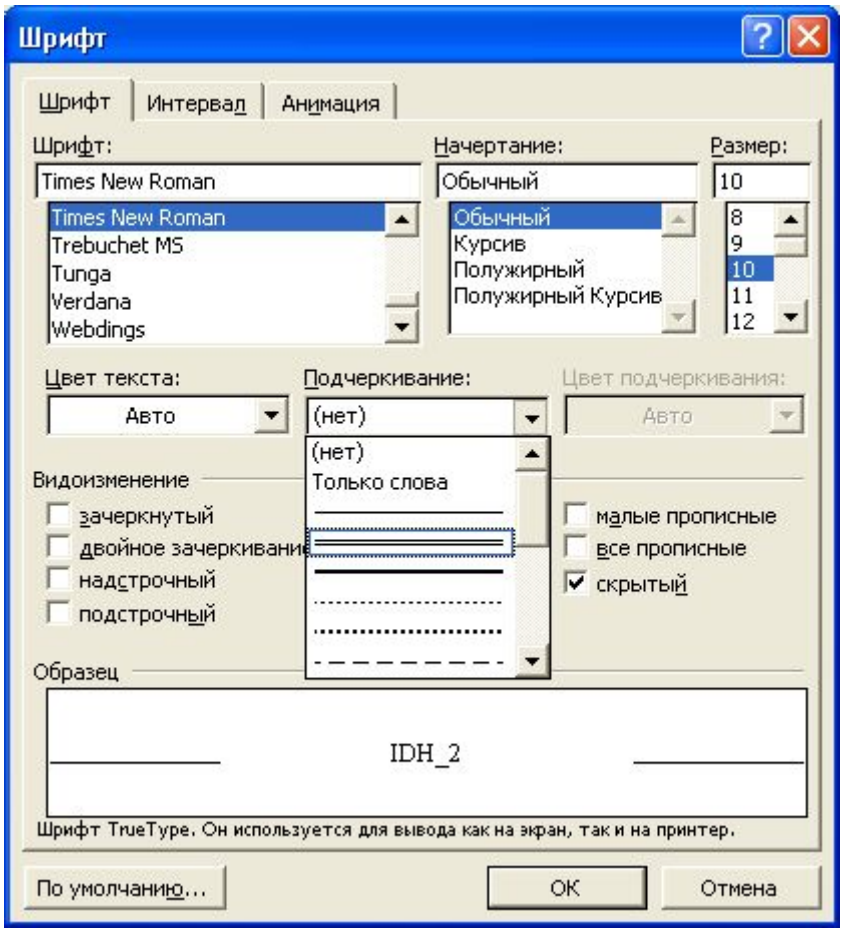
Kalit so’z sifatida sarlavha nomining qisqartirmasini yoki ketma-ket raqamlangan so’zlarni berish mumkin, masalan Ti (Topic Identifier). Ammo IDH_ kalit so’zi bilan belgilansa yaxshi bo’ladi.

Qoidaga asosan yordam tizimi bo’limlarida boshqa bo’limlarga ishoratlar bo’ladi. Yordam tizimi oynasida boshqa bo’limlarga qo’yilgan ishoratlar (ishorat qo’yilgan so’zlar) asosiy matnda rang va tagiga chiziligan chiziq bilan farq qiladi.

Yordamchi ma’lumotlar matnini tayyorlayotgan vaqtda, boshqa bo’limga o’tishni ta’minlash kerak bo’lgan so’zning tagiga qo’sh chiziq chizish kerak. So’ngra shu so’zdan keyin, bo’sh joy qoldirmasdan, o’tish kerak bo’lgan bo’lim sarlavhasi oldiga qo’yilgan snoskaning kalit so’zini yozish kerak (masalan, bizning misolda IDH_1) va yo’zilgan kalit so’zni yashirin (скрытый) qilib qo’yish.

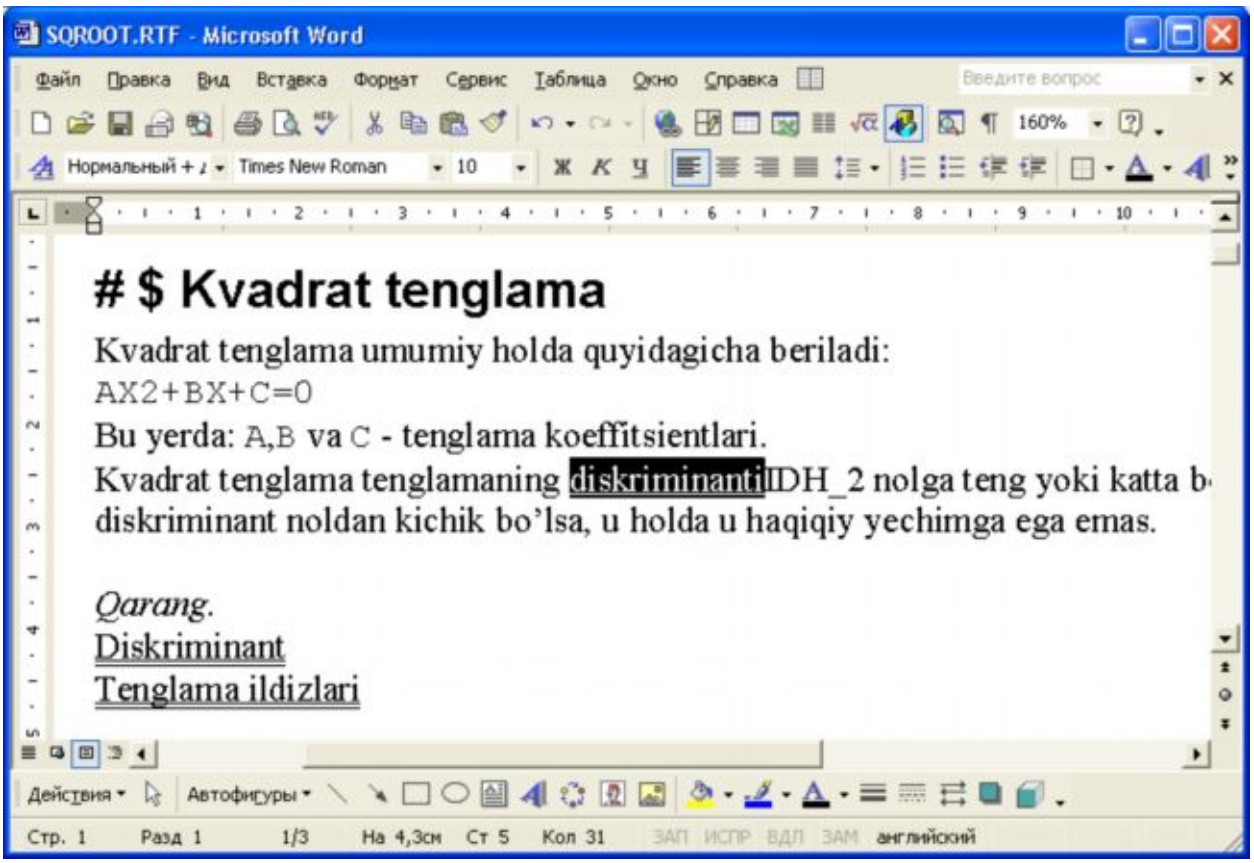
Eslatma.

Wordda so’zning tagiga chizish yoki yashirin qilib qo’yish uchun shu so’zni belgilab **Формат** menyusidan **Шрифт** buyrug’ni tanlash kerak. Natijada 14.3-rasmdagi muloqot oynasi ochiladi. Bu oynada so’zning tagiga chizish uchun **Подчеркивание** maydonidan qo’sh chiziqni tanlash kerak. So’zni yashirin qilib qo’yish uchun esa **Скрытый** maydonini belgilash kerak.



14.3-rasm.

14.4-rasmda kvadrat tenglamani yechish dasturi uchun yordam ma’lumotlarni tayyorlash vaqtidagi tahrirlash oynasining ko’rinishi keltirilgan. Bu rasmda "дискриминант" so’zi bo’shqa bo’limga ishorat sifatida belgilangan. Bunda diskriminant haqida ma’lumot joylashgan bo’lim IDH_2 kalit so’ziga ega bo’lgan "#" snoskasi bilan belgilangan bo’lishi kerak.



14.4-rasm.

Yordam tizimida boshqa bo'limlarga o'tishni ta'minlovchi ishoratdan tashqari shu bo'limning o'ziga izox matnini ham chiqarish mumkin. Buning uchun yodamchi ma'lumotlar matnini tayyorlash vaqtida izox matnini alohida sahifaga yozish kerak, oddatda izox matni sarlavhasiz bo'ladi. So'ngra izox matnining oldiga # snoskasini qo'yish kerak va snoskaning kalit so'zini yozish kerak. Izoxga ishorat qo'yish quyidagicha amalga oshiriladi: dastlab ishorat qo'yilishi kerak bo'lgan so'z tagiga yakka chiziq chizish kerak, so'ngra shu so'zdan keyin, bo'sh joy belgisini qo'ymasdan, izox matni oldiga qo'yilgan snoskaning kalit so'zini yozib, shu kalit so'zni yashirin qilib qo'yish kerak. Yordam tizimi ishlagan vaqtda izoxga ishorat qo'yilgan so'z uzuq chiziq bilan belgilanadi (14.5-rasm).

14.5-rasm.

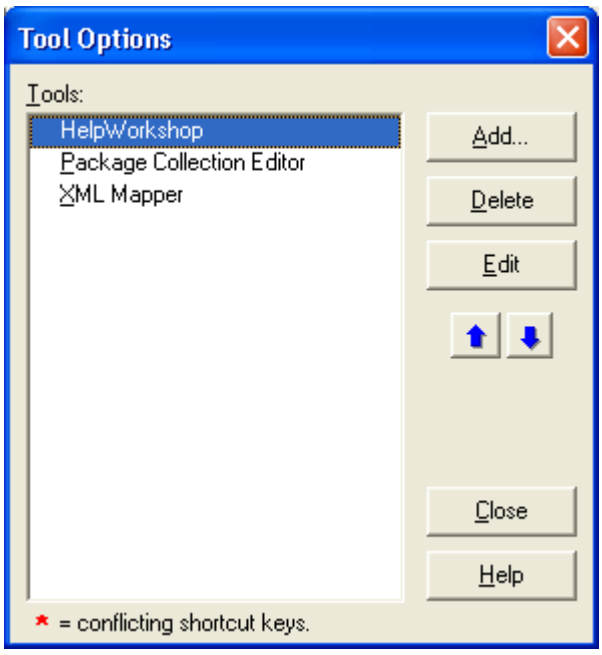
Yordam tizimini yaratish

Yordam tizimining loyihasini yaratish

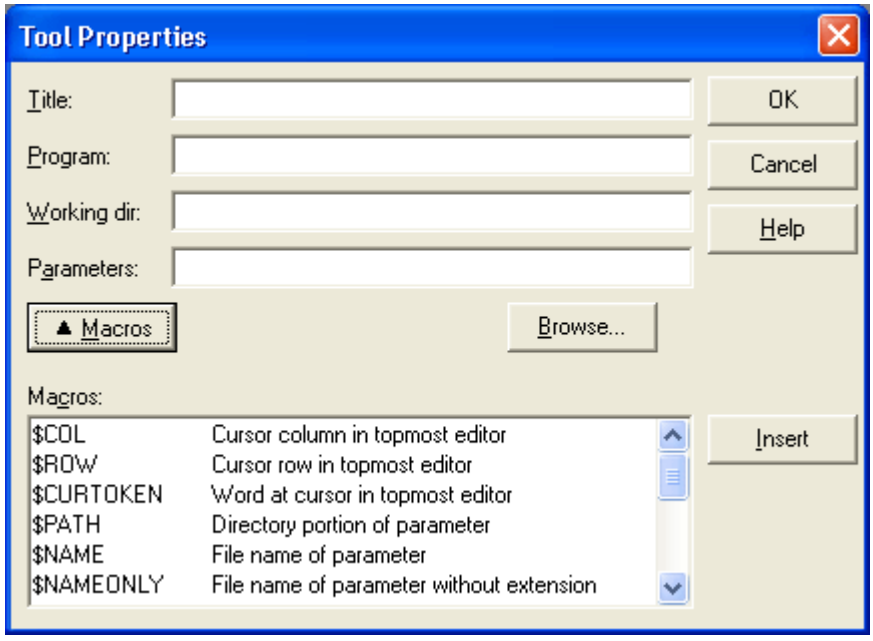
Yordamchi ma’lumotlar fayli (RTF-fayl) tayyor bo’lgandan keyin, yordam tizimini (HLP-faylni) yaratishga kirishish mumkin. Buning uchun Delphi bilan birga o’rnatiluvchi HCW.exe faylida joylashgan Microsoft Help Workshop dasturidan foydalanish juda qulay.

Microsoft Help Workshop dasturini Windows muhitidan yoki Delphining **Tools** menyusidagi **Help Workshop** buyrug’ini tanlash bilan ishga tushirish mumkin.

Agar **Tools** menyusida **Help Workshop** buyrug’i yo’q bo’lsa, u holda bu buyruqni o’rnatish mumkin. Buning uchun shu menyuning o’zidan **Configure Tools** b uyrug’ini tanlsh kerak, natijada **Tool Options** muloqot oynasi ochiladi (14.6-rasm). Bu oynada **Add** tugmasini bosish kerak, natijada **Tool Properties** muloqot oynasi ochiladi (14.7-rasm). Bu oynada **Title** maydoniga dastur nomini (Help Workshop) kiritish kerak, **Program** maydoniga esa bajariluvchi dastur faylining to’liq nomini (yo’lni ko’rsatish bilan) yozish kerak. Bizning misolda **Microsoft Help Workshop—** C:\Program Files\Borland\Delphi7\Help\Tools\HCW.exe. Fayl nomini kiritish uchun **Browse** tugmasidan foydalanish mumkin.



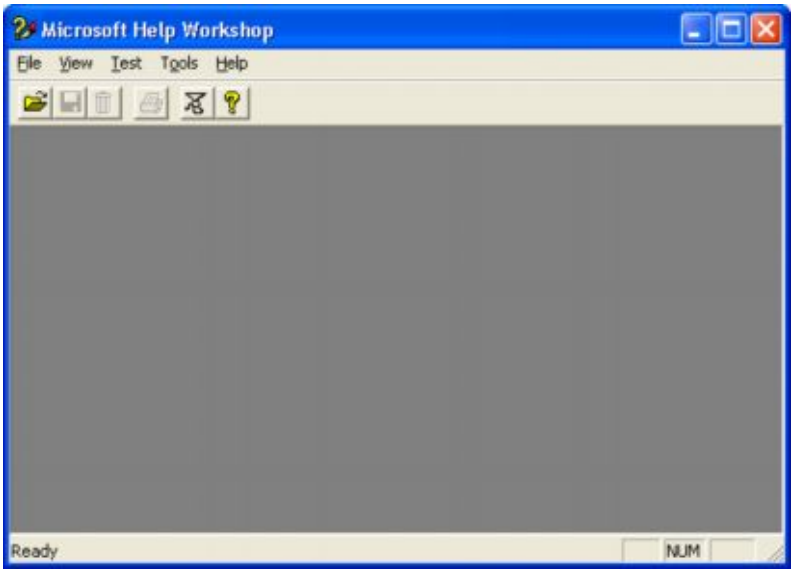
14.6-rasm.



14.7-rasm.

Microsoft Help Workshop dasturini ishga tushirgandan keyin ekranda dasturning asosiy oynasi paydo bo'ladi (14.8-rasm).

Yordam tizimini yaratishni boshlash uchun **File** menyusidan **New** buyrug'ini tanlash kerak, so'ngra ochilgan muloqot oynasida yaratilayotgan fayl tipini tanlash kerak— **Help Project.** Natijada **Project File Name** oynasi ochiladi. Bu oynada dastlab yaratilayotgan yordam tizimi fayli joylashadigan katalogni tanlash kerak (yordamchi ma'lumotlar fayli va yordam tizimi fayli bitta katalogda joylashsa maqsadga muvofiq bo'ladi). So'ngra **Имя файла** maydoniga yordam tizimi loyihasining nomini kiritish kerak. **Сохранить** tugmasi bosilgandan keyin yordam tizimi loyihasining oynasi ochiladi.



14.8-rasm.

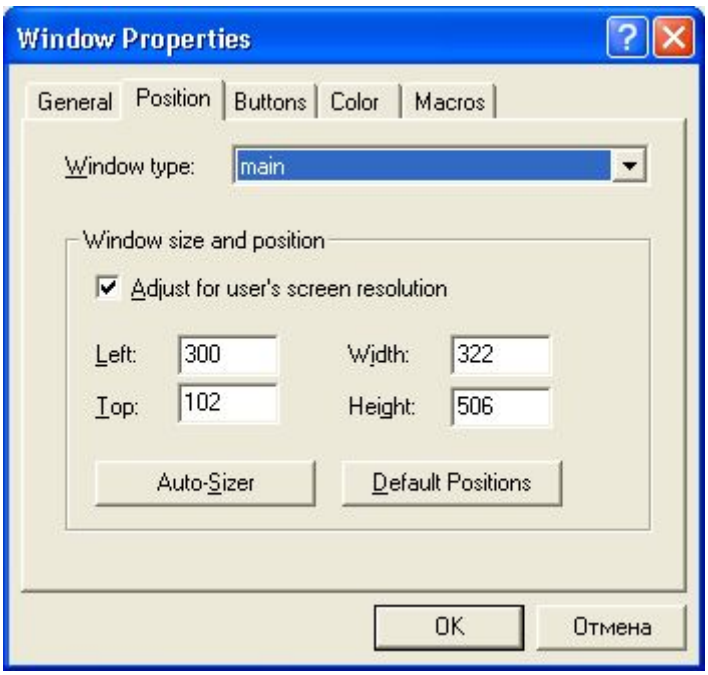
Loyihaga yordamchi ma’lumotlar faylini (RTF-faylni) qo’shish

Loyihaga yordamchi ma’lumotlar faylini (RTF-faylni) qo’shish uchun **Files** tugmasini bosish kerak va ochilgan **Topic Files** muloqot oynadsida **Add** tugmasini bosish kerak. Natijada standart **Открытие файла** oynasi ochiladi. Bu oynada kerakli RTF-fayli tanlanadi. Natijada loyiha oynasida [FILES] bo’limi paydo bo’ladi. Agar yordamchi ma’lumotlar bir nechta faylda joylashgan bo’lsa, u holda faylni qo’shish jarayonini takrorlash kerak.

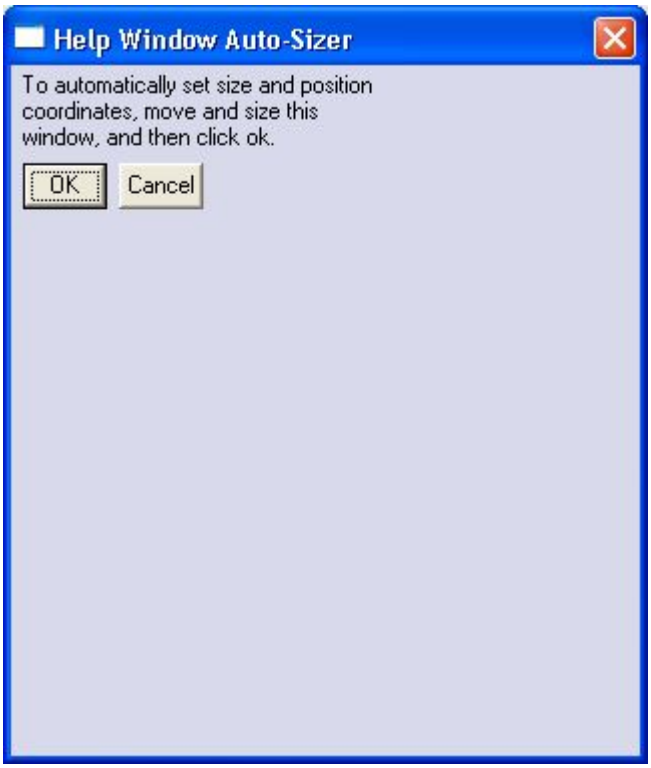
Yordam tizimi oynasining xarakteristikasi.

Yordam tizimi asosiy oynasining xarakteristikasini berish uchun, loyiha oynasida **Windows** tugmasini bosish kerak. Natijada **Create a window** oynasi ochiladi. Bu oynada **Create a window named** maydoniga **main** so’zini kiritish kerak. **OK** tigmasiga bosish natijasida **Window Properties** oynasi ochiladi. Bu oynaning **General** bo’limidagi **Title bar text** maydoniga yaratilayotgan yordam tizimining sarlavhasini kiritish kerak.

Window Properties muloqot oynasining **Position** bo’limidan foydalanib yordam tizimining joylashishini va o’lchamini berish mumkin (14.9-rasm). **Position** bo’limida **Auto-Sizer** tugmasi joylashgan bo’lib, bosish tanijasida **Help Window Auto-Sizer** oynasi ochiladi (14.10-rasm). Bu oynada sichqon yordamida oynaning o’lchamini va holatini o’zgartirish mumkin. **OK** tugmasi bosilgandan keyin **Help Window Auto-Sizer** oynasining o’chami va holati **Position** bo’limining maydonlariga yoziladi.



14.9-rasm.



14.10-rasm.

Color bo’limidan foydalanib, yordam tizimining sarlavha maydoni **(Nonscrolling area color)** va matn maydonlari **(Topic area color)** fonining rangini o’zgartirish mumkin. Buning uchun mos ravishda **Change** tugmasini bosib, standart **Цвет** oynasidan kerakli rangni tanlash kerak.

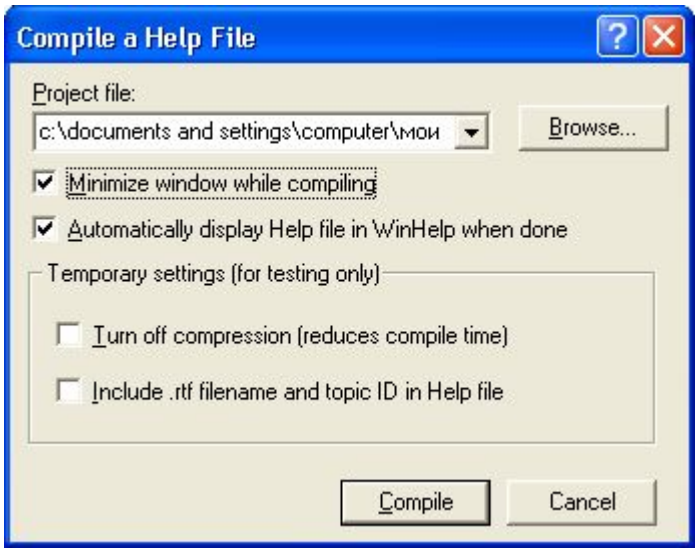
Yordam tizimi bo’limlarining kalit so’zlariga sonli qiymatlarni berish

Yordam tizimini ishlatuvchi Delphida yaratilgan dastur, yordam tizimining aniq bir bo’limiga murojaat eta olishi uchun bo’limlarning kalit so’zlari uchun sonli qiymatlarni berish kerak. Buning uchun yordam tizimining loyiha oynasidagi **Map**

tugamasini bosish kerak, natijada **Map** muloqot oynasi ochiladi. Bu oynada **Add** tugmasini bosib, ochilgan **Add Map Entry** muloqot oynasidagi **Topic ID** maydoniga bo'limning kalit so'zini, **Mapped numeric value** maydoniga esa kalit so'zga mos sonli qiymatni kiritish kerak. **Comment** maydoniga kalit so'zga mos bo'lim nomi uchun izox yozish mumkin.

Loyihani kompilyatsiya qilish

Loyiha fayli tayyor bolgandan keyin loyiha oynasida joylashgan **Save and Compile** tugmasini bosib loyihani kompilyatsiya qilish mumkin. Biroq loyihani birinchi marta kompilyatsi qilishda **File** menyusidagi **Compile** buyrug'ini tanlash yo'li bilan amalga oshirish maqsadga muvofiq bo'ladi. Natijada **Compile a Help File** muloqot oynasi ochiladi (14.11-rasm).

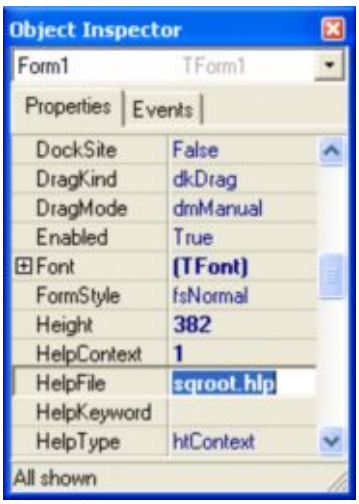


14.11-rasm.

Bu oynada **Automatically display Help file in WinHelp when done** (kompilyatsiya yakunlangandan keyin yordam tizimini avtomatik tarzda ko'rsatish) maydonini belgilab qo'yish, so'ngra **Compile** tugmasini bosish kerak. Kompilyatsiya yakunlangandan keyin ekranda kompilyatsiya natijasi haqida xabar beruvchi oyna paydo bo'ladi. Agar kompilyatsiya muvaffaqiyatli yakunlangan bo'lsa, yaratilgan yordam tizimi paydo bo'ladi. Loyiha fayli qaysi katalogda joylashgan bo'lsa yaratilgan yordam tizimining fayli (HLP-fayl) ham shu katalogda joylashadi.

Yordam tizimiga murojaat etish

Dastur ishlagan vaqtda foydalanuvchi <F1> tugmasini bosib, yordam tizimini yoki uning ixtiyoriy bo'limini chaqirib olishi uchun, ilova bosh oynasining **HelpFile** hususiyati yordam tizimining nomiga va **HelpContext** hususiyati esa kerakli bo'limning sonli qiymatiga ega bo'lishi kerak (14.12-rasm). Bu sonli qiymatlar yordam tizimi loyihasidagi [MAP] bo'limidagi sonli qiymatlar hisobanadi.



14.12-rasm

Ilovaning HLP-faylini ilovaning EXE-fayli bilan bitta katalogga joylashtirish maqsadga muvofiq bo'ladi.

Formaning har bir komponentiga, masalan kiritish maydoniga (Edit), o'zining yordam tizimi bo'limini berish mumkin. Buning uchun komponentning HelpContext hususiyatiga yordam tizimi bo'limining mos sonli qiymatini berish kerak. Natijada ushbu komponentada kursor turgan vaqtda <F1> tugmasi bosilsa yordam tizimining mos bo'limi ochiladi. Agar HelpContext hususiyatining qiymati nolga teng bo'lsa, u holda <F1> tugmasi bosilishi natijasida ilova formasiga belgilangan yordam tizimi bo'limi ochjiladi.

Agar muloqot oynasida **Ma'lumot** tugmasi mavjud bo'lsa, u holda yordam tizimi boshqacha usulda chiqariladi. Ya'ni tugma bosilganda WinHelp funksiyasidan foydalanib, Windows Help (Winhlp32.exe-fayli) dasturi ishga tushiriladi. WinHelp funksiyasidan foydalanganda quyidagi parametrlar beriladi: yordam tizimini chaqirayotgan oyna Handle i; HLP-fayl nomi; Windows Help dasturi bajarishi kerak bo'lgan ishni aniqlovchi konstanta va oydinlashtiruvchi parametr.

Agar yordam tizimining aniq bir bo'limini chiqarish kerak bo'lsa, u holda ishni aniqlovchi parametr sifatida HELP_CONTEXT konstantasi ishlatiladi va oydinlashtiruvchi parametr sifatida esa kerak bo'limning sonli qiymati beriladi.

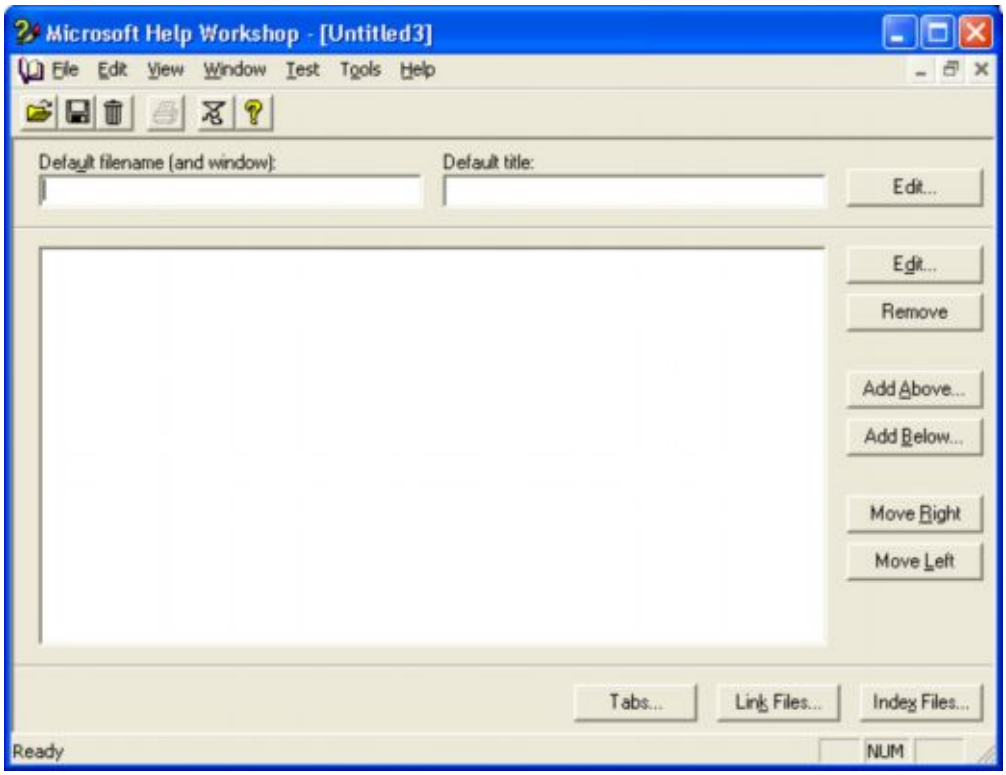
Quyida misol sifatida kvadrat tenglamani echish dasturining muloqot oynasidagi **Ma'lumot** (Button2) tugmasi bosilganda yordam tizimini chaqiruvchi protsedura keltirilgan.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  winhelp(Form1.Handle,'sqroot.hlp',HELP_CONTEXT,1);
end;
```

Yordam tizimining mundarijasini yaratish

Yordam tizimini yaratish uchun **Microsoft Help Workshop** dasturining **File** menyusidan **New** buyrug'ini tanlash kerak va ochilgan oynada **Help Contents**

bo’limini tanlab **OK** tugmasini bosish kerak. Natijada 14.13-rasmda ko’rsatilgan muloqot oynasi ochiladi.



14.13-rasm.

Bu oynada **Default filename** maydoniga avval yaratilgan HLP-faylning nomini va **Default title** mydoniga esa yordam tizimi sarlavhasini kiritish kerak. So’ngra **Add Above** yoki **Add Below** tugmalaridan foydalanib mundarijani yaratish mumkin. Bu tugmalar mos ravish joriy qatordan yuqoriga yoki pastga qator qo’shadi. **Move Right** va **Move Left** tugmalari mos ravishda joriy qatorni o’ngga va chapga suradi. **Edit** va **Remove** tugmalari mos ravishda joriy qatorni taxrirlaydi yoki o’chiradi. Qo’shish tugmalaridan biri bosilganda **Edit Contents Tab Entry** muloqot oynasi ochiladi. Bu oynada quyidagi ishlarni bajarish mumkin:

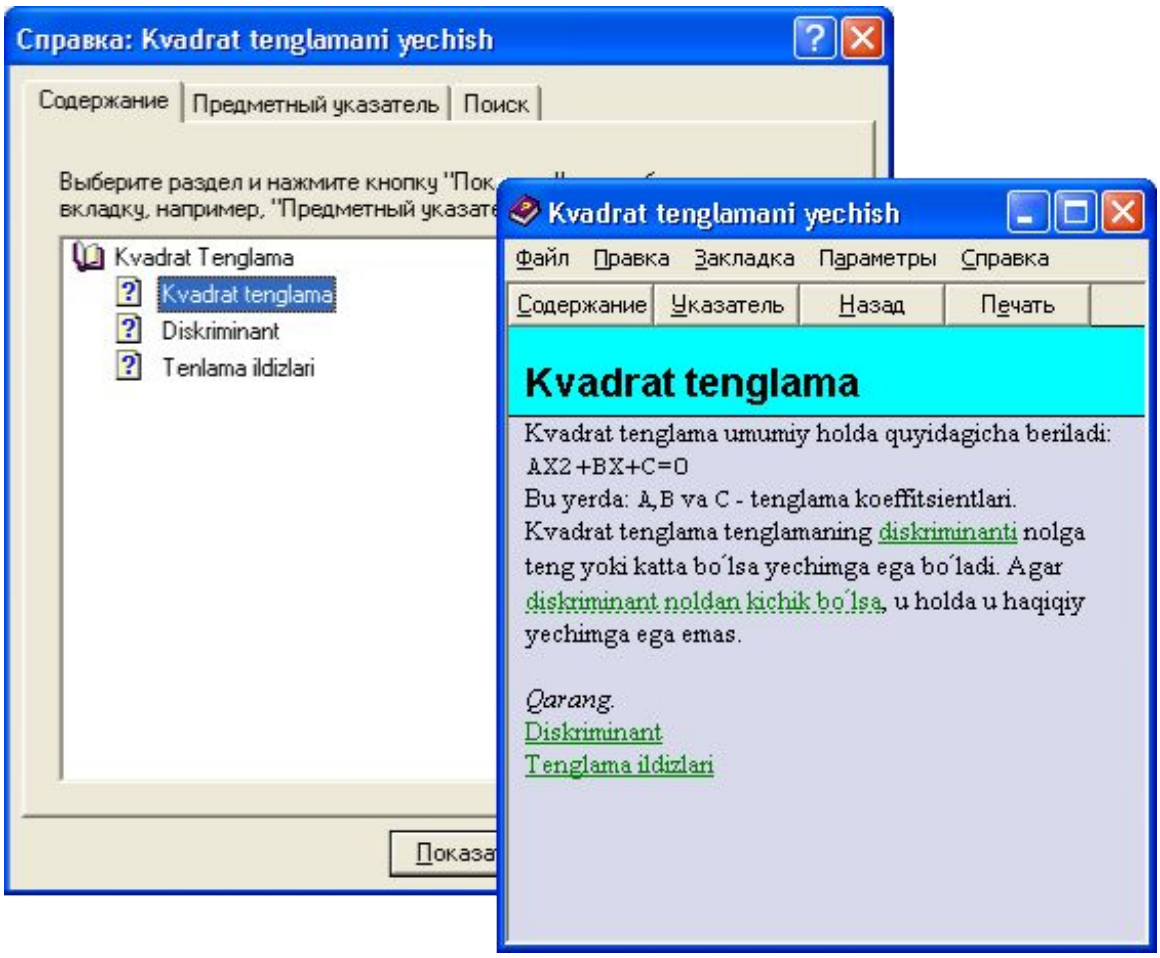
1. Mundarijaga yangi bo’lim qo’shish uchun **Heading** maydonini tanlab, **Title maydoniga** bo’lim nomi kiritiladi (14.14-rasm).

14.14-rasm.

2. Mundarijaga yangi mavzu qo'shish uchun **Topic** maydonini tanlab, **Title** maydoniga mavzu nomini va **Topic ID** maydoniga HLP-fayldagi sarlavhalar oldiga qo'yilgan snoskaning kalit so'zini kiritish kerak (14.15-rasm). Agar mavzu matni boshqa HLP-faylda bo'lsa **Help file** maydoniga o'sha fayl nomini ko'rsatish kerak.

14.15-rasm.

Mundarijani kiritib bo'lgandan so'ng uni **File** menyusining **Save** buyrug'ini tanlab saqlash kerak. Mundarijani yordam tizimi CNT-fayl ko'rinishida saqlaydi. Saqlash vaqtida CNT-faylni HPJ-fayl (loyiha-fayl) joylashgan katalogga saqlash kerak. So'ngra **Test** menyusida **Contents File** buyrug'ini tanlab, ochilgan oynada **Browse** tugmasini bosib, CNT faylni tanlash kerak. Natijada dastur avtomatik tarzda tekshirishni takliv qiladi. Agar **Browse** tugmasidan foydalanilmasa, ya'ni **Contents File** maydoniga CNT-fayl kiritilsa **Test** tugmasi ishlatiladi. Test (tekshirish) vaqtida xatolik bo'lmasa, test muvaffaqiyatli yakunlangani to'g'risida xabar chiqariladi. CNT fayl bog'langandan keyin loyihani qayta kompilyatsiya qilinsa yordam tizimi mundarija bilan chiqadi (14.16-rasm).


Справка: Kvadrat tenglamani yechish
Содержание
Предметный указатель
Поиск
Kvadrat Tenglama
Kvadrat tenglama
Diskriminant
Tenlama ildizlari
Kvadrat tenglamani yechish
Файл
Правка
Закладка
Параметры
Справка
Содержание
Указатель
Назад
Печать
Kvadrat tenglama
Kvadrat tenglama umumiy holda quyidagicha beriladi:
AX2+BX+C=0
Bu yerda: A,B va C - tenglama koeffitsientlari.
Kvadrat tenglama tenglamaning diskriminanti nolga teng yoki katta bo´lsa yechimga ega bo´ladi. Agar diskriminant noldan kichik bo´lsa, u holda u haqiqiy yechimga ega emas.
Qarang.
Diskriminant
Tenglama ildizlari

14.16-rasm.

17-Ma`ruza. O’rnatuvchi disklarni (Installyatsion) yaratish. (2 soat)

O’quv modul birliklari:

1. O’rnatuvchi disklarning zaruriyati
2. InstallShield Express dasturi
3. O’rnatiluvchi komponentlarni tanlash
4. Foydalanuvchi sistemasini konfiguratsiyalash
5. Muloqotlarni sozlash
6. O’rnatuvchi disk obrazini yaratish

Aniqlashtirilgan o’quv maqsadlari:

Talaba ushbu mavzuni to’la o’zlashtirgandan so’ng:

1. O’rnatuvchi disklarga bo’lgan zaruriyat haqida ma’lumotga ega bo’ladi
2. InstallShield Express dasturi bilan tanishadi
3. O’rnatiluvchi komponentlarni tanlashni biladi
4. Foydalanuvchi sistemasini konfiguratsiyasini o'rganadi
5. Muloqotlarni sozlay oladi
6. O’rnatuvchi disk obrazini yarata oladi

Tayanch so’z va iboralar

1. Kompyuter qurilmalari bilan ishlaydigan modullar.
2. Kompyuter qurilmalariga Delphi orqali murojaat qilish usullari.
3. Kompyuter qurilmalari xaqidagi ma’lumotlarni olish usullari.

O`rnatuvch disklarni yaratish

O’rnatuvchi disklarni yaratish.

Zamonaviy dasturlar disketlarda yoki CD-disklarda joylashadi. Dasturni o’rnatish jarayoni, yani, kerakli kataloglarni yaratish va unga mos ishchi fayllarni hamda ma’lumotlarni o’zida saqlovchi fayllarni joylashtirish ko’pgina foydalanuvchilar uchun qiyinchilik tug’diradi.

Shuning uchun, amaliy dasturlarni kompyuterga o’rnatishda maxsus dasturdan foydalaniladi. Bu dastur joriy diskda joylashadi.

Installyatsion dasturlar boshqa dasturlar singari yaratilishi mumkin. Lekin bu juda ham murakkab jarayon hisoblanadi. Yuqoridagilardan kelib chiqqan holda installyatsion dasturlarni yaratuvchi maxsus dasturlar tuzilgan bo’lib, uni yordamida dasturchi xech qanday kod yozmasdan installyatsion dastur yaratishi mumkin.

InstallShield Express dasturi

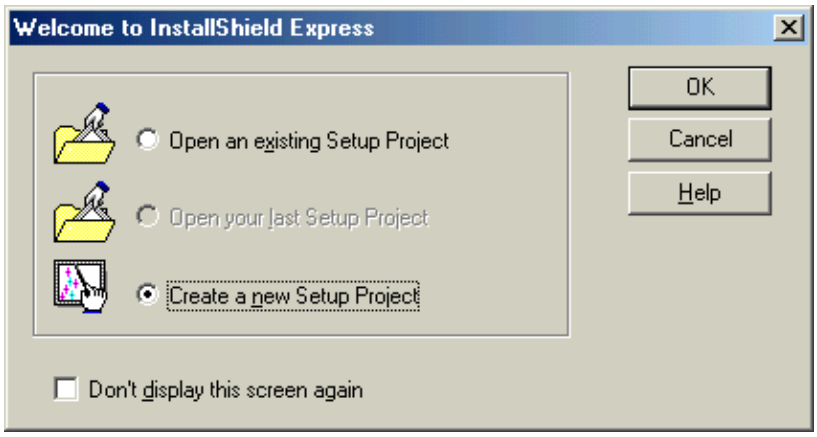
Dasturlar yaratilganidan so`ng, ularni boshqa kompyuterlarda ishlatish uchun, o`rnatuvchi (Установочный) disklarni yaratish lozim bo`ladi. Bunday disklarni

yaratish uchun maxsus tuzilgan dasturlar bo`lib, ulardan keng tarqalgani **InstallShield Expres** dasturidir. Borland shu dasturdan foydalanishni tavsiya etgani bois InstallShield Express dasturi Delphining o'rnatuvchi diskidan joy olgan.

InstallShield Express yordamida installyatsion disketlarni yaratish jarayonini quyidagi misol yordamida ko'rib o'tamiz. Universal test dasturi uchun installyatsion disk yaratish kerak bo'lsin.

Eslatma: *InstallShield Express utiliti Delphini o'rnatish paytida avtomatik tarzda o'rnatilmaydi. Uni o'rnatish uchun Delphini o'rnatuvchi dasturni ishga tushirish va hosil bo'lgan Delphi Setup Launcher muloqot oynasidan InstallShield Express Custom Edition for Delphi buyrug'I tanlanadi.*

Installyatsion disketni yaratish uchun InstallShield Express ni ishga tushirish kerak. Hosil bo'lgan Welcome muloqot oynasidan Create a New Setup Project (yangi proyekt yaratish) ni tanlab, Ok tugmasi bosiladi (32-rasm).

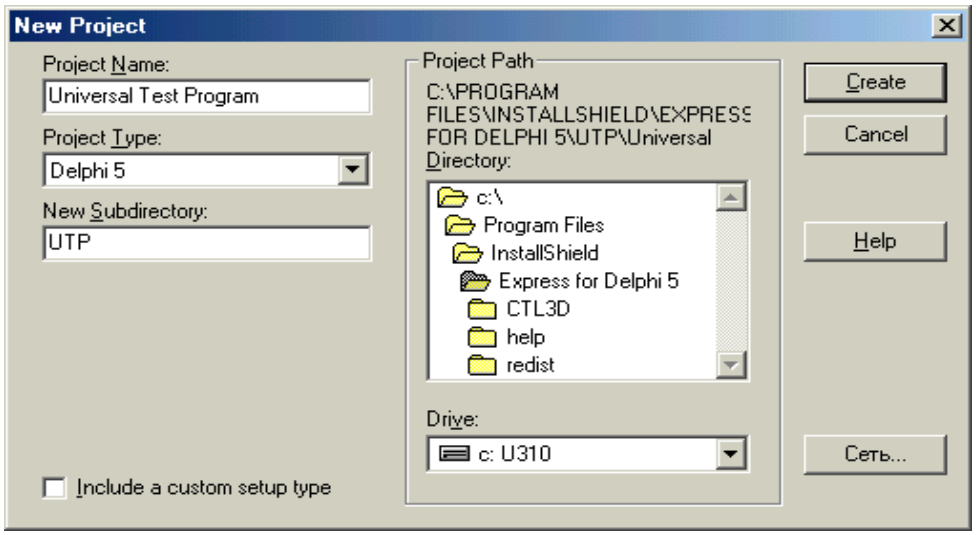


32-rasm. Dastur ishga tushirilganidan keyingi InstallShield Express muloqot oynasi.

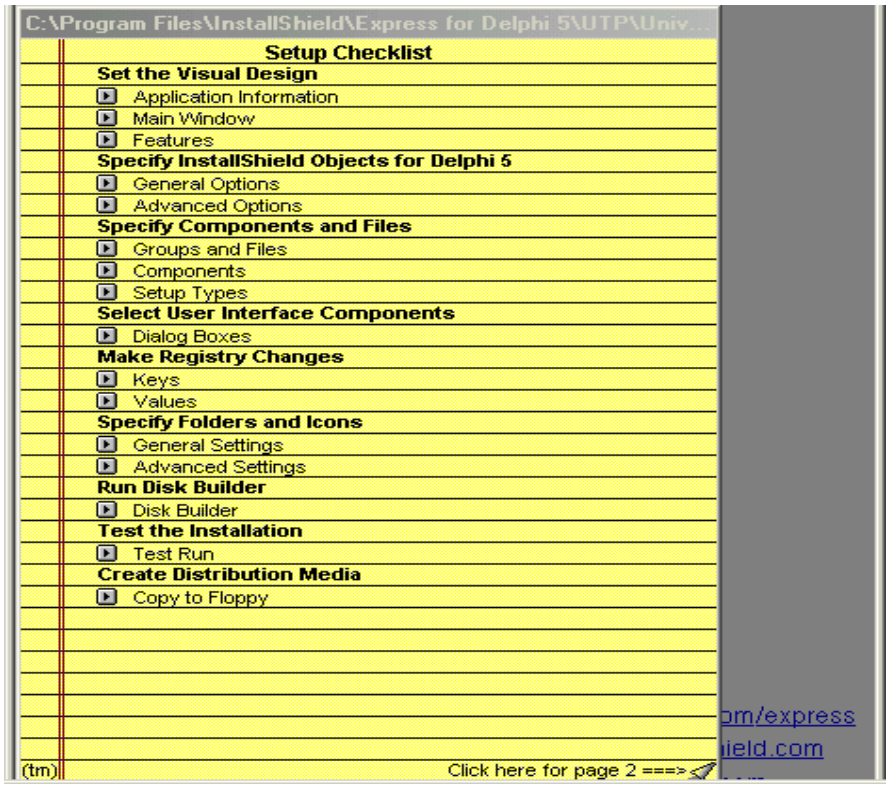
Hosil bo'lgan New Project (33-rasm) muloqot oynasiga proyekt haqidagi ma'lumotlar kiritiladi:

Project Name maydoniga – proyekt nomi; New Subdirectory maydoniga katalog nomi, InstallShield Express dasturi ushbu katalokka barcha yaratiluvchi fayllarni joylashtiradi. Directory va Drive ro'yxatdan foydalanib proyekt katalogini qayerga joylashishini ko'rsatish mumkin.

Create tugmasi bosilgandan so'ng installyatsion dastur yaratuvchi proyekt oynasi ochiladi. Bu yerda yaratilayotgan o'rnatuvchi dastur parametrlarini aniqlovchi buyruqlar ketma-ketligi joylashgan (34-rasm).



33-rasm. New Project muloqot oynasi



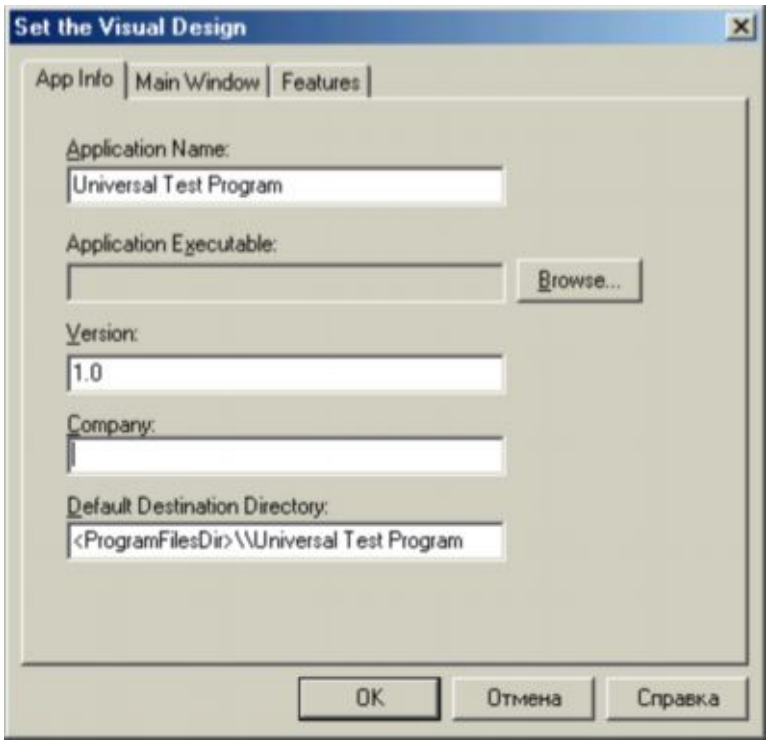
34-rasm. Installyatsion dasturni yaratuvchi proyekt oynasi.

O’rnatuvchi dasturni sozlash.

O’rnatuvchi dasturni sozlash buyruqlari guruhlarga ajratilgan (guruh sarlavhasi qalin xarflar bilan ajratilgan). Buyruqni tanlash uchun sichqoncha tugmasini buyruq satrida yoki strelkali tugma sirtilda bosish yetarli. Natijada guruhning muloqot oynasi ochiladi.

Umumiy ma’lumotlar.

Set the Visual Design guruh buyruqlari o’rnatiluvchi ilova to’g’risiadgi umumiy ma’lumotlarni berish va o’rnatuvchi dasturning bosh oyna ko’rinishini belgilash imkonini beradi. Application Information buyrug’ini tanlash bilan App Info sahifasi ochiladi (35-rasm). Bu yerda ilova nomi, bajariluvchi fayl va katalog nomi ko’rsatiladi.



35-rasm. App Info sahifasi

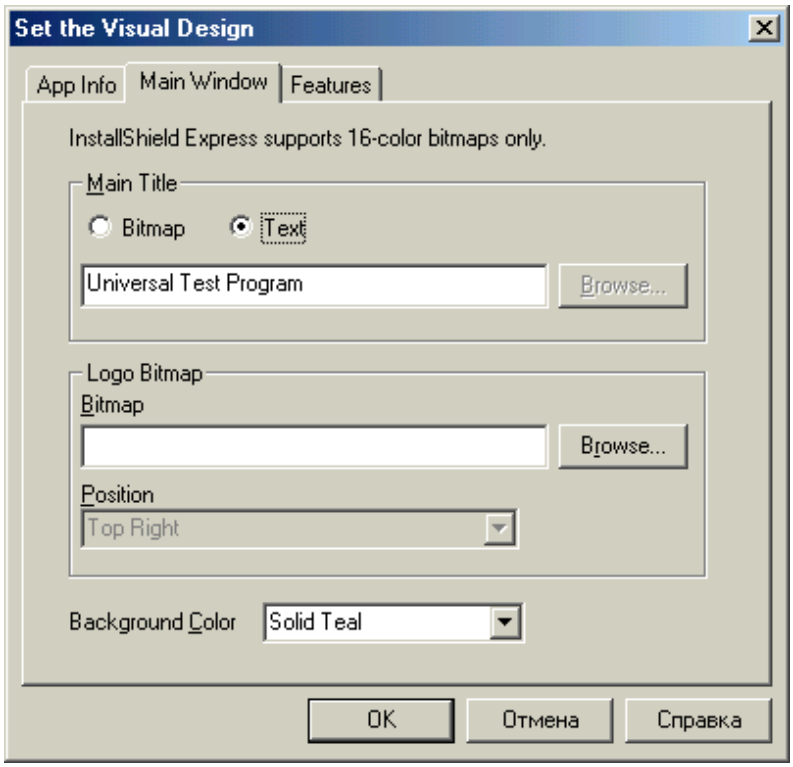
Sahifaning ba’zi maydonlari avtomatik to’ldirilganligiga e’tibor qilish lozim. InstallShield Express dasturi ilova nomi (Application Name) va ilova katalogi nomi (Default Destination Directory) sifatida proyekt nomidan foydalanishni taklif qiladi (dasturchi boshqa nom berishi ham mumkin).

Default Destination Directory maydoniga e’tibor beramiz. Bu maydonda o’rnatiluvchi ilova katalogining nomi va uning holati ko’rsatilgan.

InstallShield Express dasturida foydalanuvchi Windows katalog ko’rsatkichlari quyidagi jadvalda keltirilgan:

Ko’rsatkich	**Katalog**
<WINDIR>	Windows katalogi, masalan, c:\Windows
<WINSYSDIR>	Windowsning sistema katalogi, masalan, c:\Windows\System
<ProgramFilesDir>	Dastur katalogi, masalan, c:\Program Files
<INSTALLDIR>	Installyatsion dastur ilovani o’rnatuvchi katalog

Main Window (36-rasm) sahifasi o’rnatuvchi dasturning bosh oynasi xarakteristikasini berish imkonini beradi.



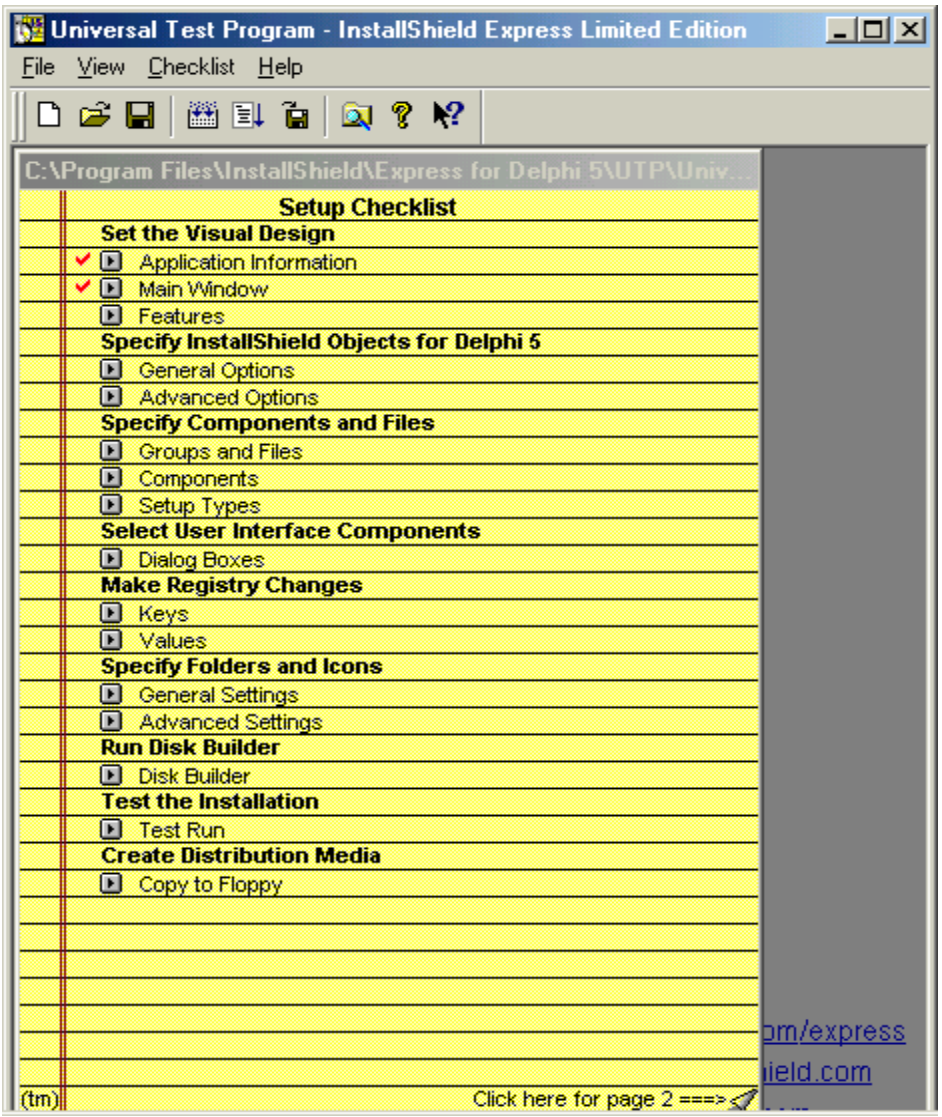
36-rasm. Main Window saxifasahifasi.

Main Title guruhi o'rnatuvchi dastur bosh oynasining sarlavhasi tipini berish imkonini beradi. Agar Text tanlangan bo'lsa sarlavxasarlavha o'zida matnni mujassamlashtiradi. Agar Bitmap tanlangan bo'lsa, ilova sifatida 16-rangli rasm ishlatiladi, faylni Browse tugmasi yordamida aniqlanadi.

Logo Bitmap guruhida dasturchi o'z emblemasini o'rnatishi mumkin. Buning uchun Bitmap maydoniga emblema joylashgan fayl nomi kiritiladi. Position ochiluvchi ro'yxat yordamida emblema joylashuvini belgilash mumkin. Oynaning markazida, yuqori chap yoki o'ng qismida.

BackGround Color ochiluvchi ro'yxatdan bosh oynaning rangini tanlash mumkin.

Ok tugmasi bosilgandan so'ng Set the Visual Design muloqot oynasi yopiladi. InstallShield Express esa foydalanilgan guruh buyruqlari oldiga belgi qo'yib qo'yadi (37-rasm).

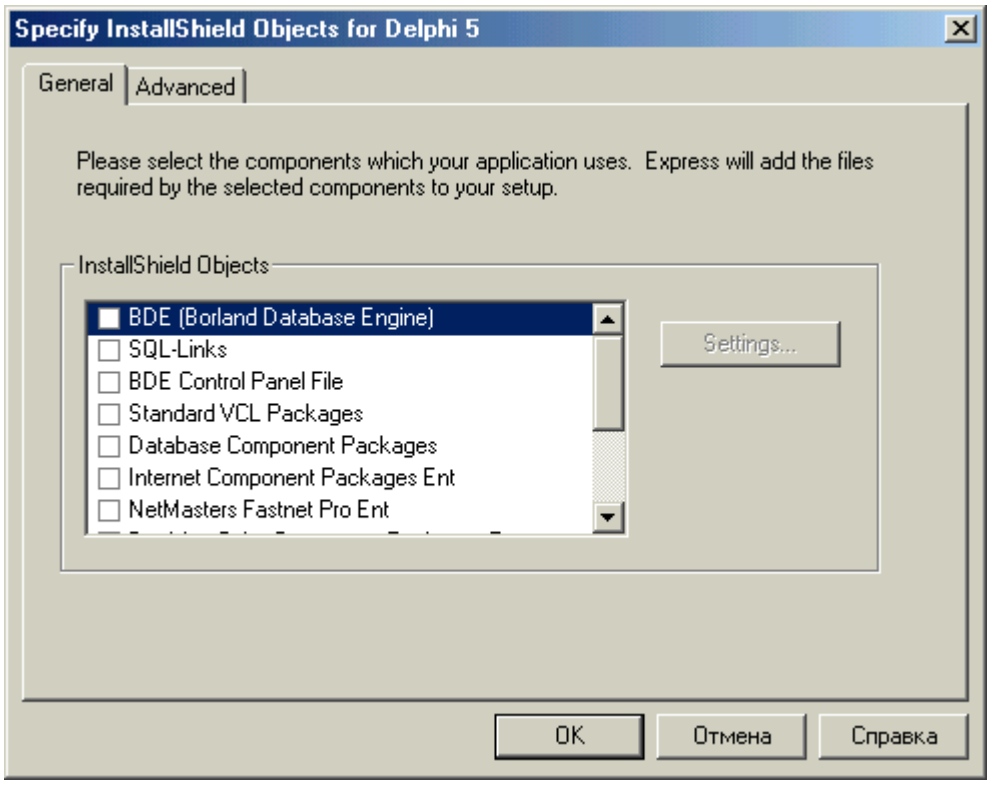


37-rasm.

O’rnatuvchi disklar uchun Delphi obyektini tanlash.

Specify InstallShield Objects for Delphi buyruqlar guruhi Delphining qaysi obyektlarini foydalanuvchi kompyuteriga ko’chirish lozimligini ko’rsatish imkonini hosil qiladi. Masalan, dinamik kutubxonalar yoki komponentlar paketi bo’lishi mumkin.

O’rnatuvchi disklarga joylashtirish kerak bo’lgan obyektlar InstallShield Objects ro’yxatdan tanlanadi (38-rasm).



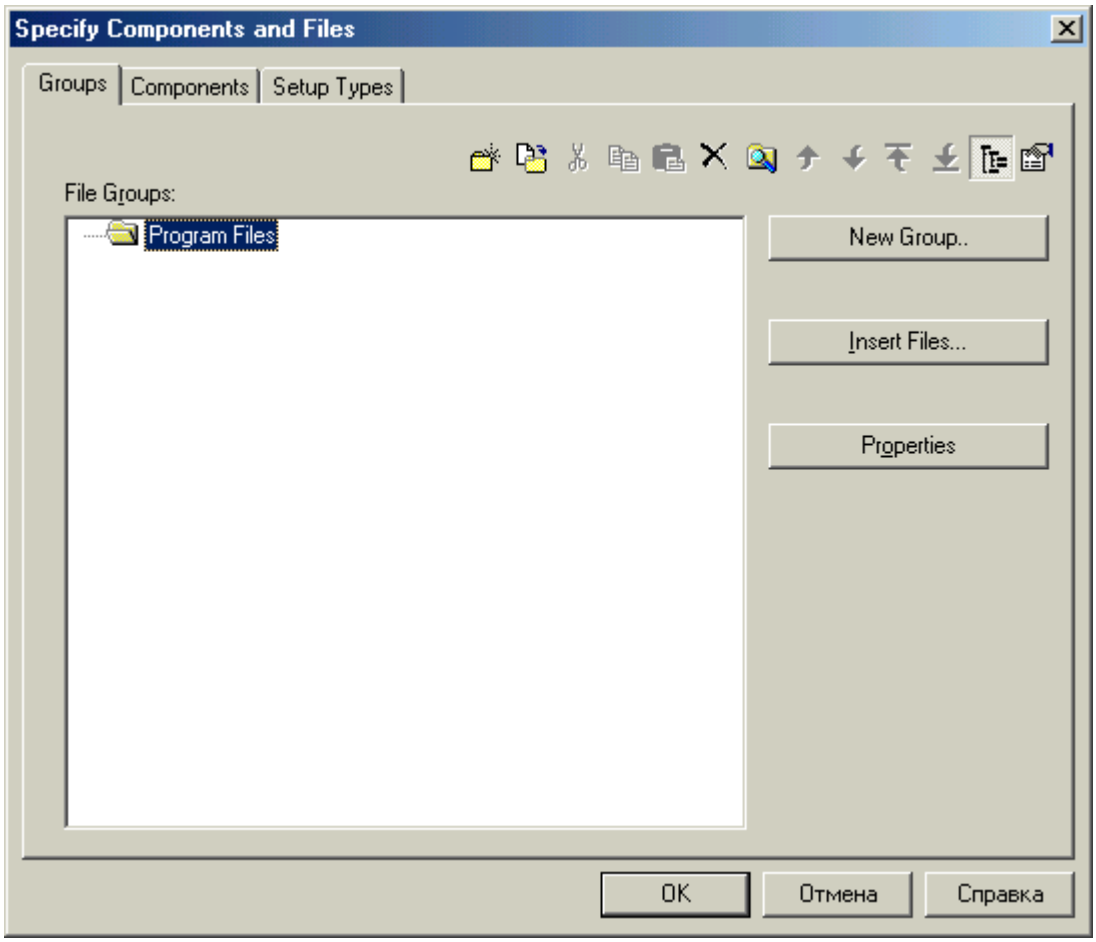
38-rasm. General sahifasi.

Qo’shimcha komponentlar va fayllar.

Specify Components and Files guruh buyruqlari foydalanuvchi kompyuteri hamda o’rnatuvchi diskka ko’chiriluvchi fayllarni yuklash imkonini beradi.

Eslatma: *Katta dasturlarni o’rnatishda installyatsion dastur foydalanuvchidan o’rnatish tipini tanlashni so’raydi (to’la, eng kichik yoki tanlash). Tanlash jarayonida foydalanuvchi kerakli komponentlarni ko’rsatib o’tadi.*

Graph and Files buyrug’I bajarilishidan avval qanday fayllarni foydalanuvchi kompyuteriga ko’chirib o’tish kerakligi ko’rsatiladi. Qaralayotgan misolimizda bu quyidagicha: dastur fayli (test1.exe), “Pamyatniki I arxitektunie soorujeniya Sankt-Peterburga” test fayli, readme.doc fayli (faylda dastur haqida qisqacha ma’lumot saqlanadi). Bundan, dastur fayli va readme.doc fayli ilovaning bosh katalogiga, test fayli esa bosh katalogning Sample ostkatalogiga joylashtiriladi. Groups and Files buyrug’ini tanlash natijasida Specify Components and Files (39-rasm) muloqot oynasi ochiladi.



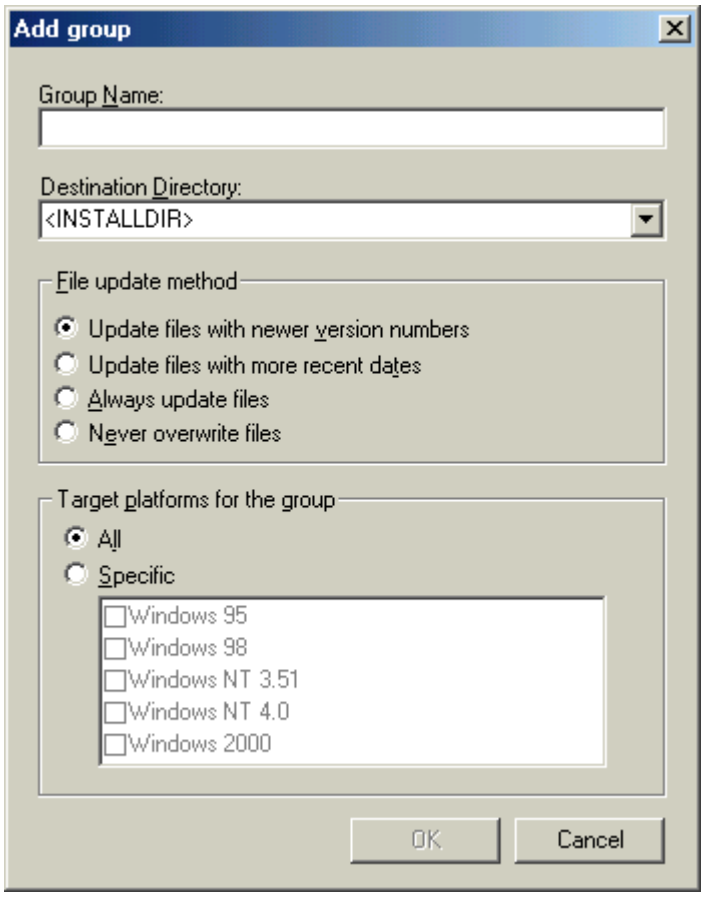
39-rasm. Specify Components and Files muloqot oynasi

Avtomatik hosil qiliingan Program Files guruhi o'rnatiluvchi dasturning bosh katalogi hisoblahadi. Odatda bu guruhga dasturning bajariluvchi fayli, ma'lumot (spravka)lar fayli va read.me fayli joylashtiriladi. Bu guruhga yangi faylni qo'shish uchun Insert Files tugmasi bosiladi va hosil bo'lgan Faylni ochish standart oynadan kerakli fayl tanlanadi. "Plus" belgili kvadratni bosish bilan guruxni ochish va u yerda qanday fayllar mavjudligini ko'rish mumkin.

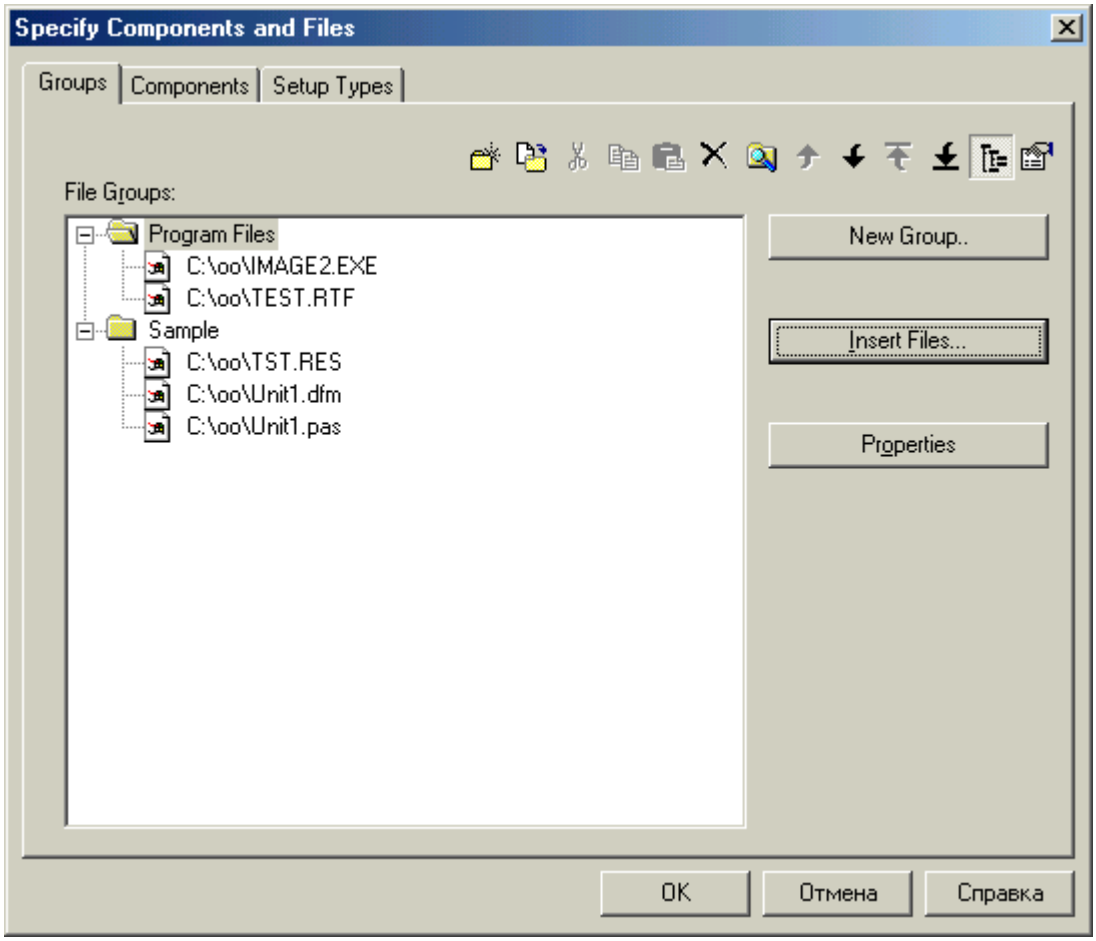
Dasturchi hohishiga ko'ra ba'zi fayllarni alohida katalokka ajratish mumkin. U holda bu fayllar uchun yangi guruh tashkil qilishga to'g'ri keladi. Yangi guruhni tashkil qilish uchun New Group tugmasidan foydalaniladi. Natijada Add group (40-rasm) muloqot oynasi hosil bo'ladi. Group Name maydoniga guruh nomi, Destination Directory maydoniga esa guruh fayllari nusxasi joylashishi lozim bo'lgan katalog nomi kiritiladi. Standart holatda yangi guruh <INSTALLDIR> katalogiga ega, yani ilovaning bosh katalogi.

Agar guruh fayllari ko'chiriluvchi yangi katalog ilovaning bosh katalogi ichida yaratilsa, u holda yangi katalogning nomi <INSTALLDIR> dan so'ng "\" belgisi bilan ajratib yoziladi. Katalog nomini ochiluvchi ro'yxatdan tanlab ko'rsatilishi ham mumkin.

41-rasmda misol sifatida Specify Components and Fiels muloqot oynasining Sample guruhi tashkil etilgan va fayllar guruhi tanlangan holda ko'rsatilgan.



40-rasm. Add Group muloqot oynasi.

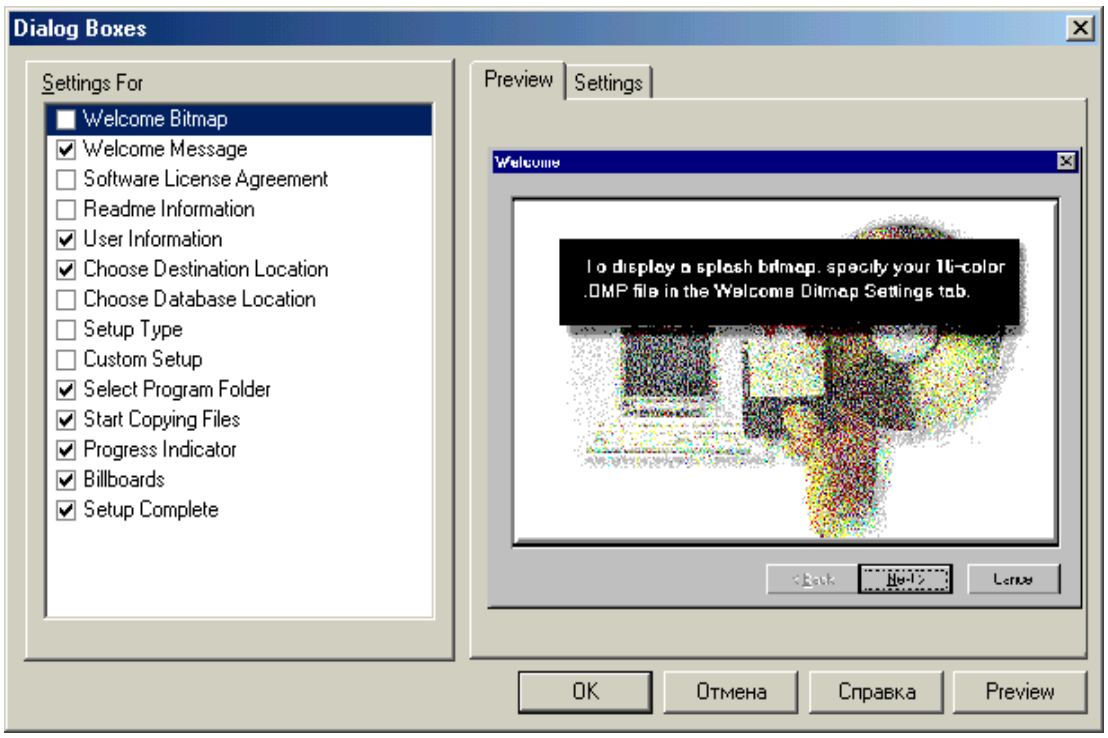


41-rasm. Foydalanuvchi kompyuteriga ko'chiriluvchi fayllar.

Muloqotni sozlash.

Installyatsion dastur ishi davomida ketma-ket o’zgarib boruvchi standart muloqot oynalarni ko’rishimiz mumkin. Installyatsiya dasturini tashkil etish bilan dasturchi qanday oynalar hosil bo’lishi va ularning ko’rinishlarini aniqlashi mumkin.

Installyatsion dastur ishga tushirilganda ekranda hosil bo’luvchi muloqot oynasini tashkil etish uchun Select User Interface Components guruhidan Dialog Boxes buyrug’ini tanlanadi (42-rasm.).



42-rasm. Dialog Boxes muloqot oynasi.

Hosil bo’lgan muloqot oynadagi Settings For ro’yxatda installyatsion dastur ishi davomida hosil bo’luvchi muloqot oyna nomlari quyidagi jadvalda keltirilgan.

Muloqot oynasi	Aniqlanishi
Welcome Bitmap	O’rnatiluvchi dastur haqidagi ma’lumot sifatida chiquvchi illyustratsiya
Welcome Message	Axborotli ma’lumotni chiqarish
Software License Agreement	Faylda joylashgan litsenzion ma’lumotni chiqarish
Readme Information	O’rnatiluvchi dastur haqidagi ma’lumotlarni chiqarish
User Information	Foydalanuvchi to’g’risidagi ma’lumotlarni kiritish (ismi, firma nomi) va o’rnatiluvchi nusxa raqami ham bo’lishi mumkin
Choose Destination Location	Dastur o’rnatish uchun aniqlangan katalogni foydalanuvchi uchun o’zgartirish imkonini berish
Setup Type	O’rnatilayotgan dastur tipini tanlash imkonini berish

	(Typical-oddiy, Compact-minimal, Custom-tanlash bilan)
Custom Setup	Tanlash (Custom) bilan o’rnatilayotganda o’rnatiluvchi komponentlarni belgilash
Select Program Folder	O’rnatiluvchi dasturning ishga tushiruvchi buyrug’ini topshiriqlar panelining qaysi menyusiga qo’yishni tanlash imkonini berish
Progress Indicator	Dasturni o’rnatish vaqtida bajarilgan ishning foizini ko’rsatish
Setup Complete	O’rnatish tugallanganligi to’g’risida foydalanuvchini ogoxlantiradi

Ushbu muloqot oynalari intallyatsion dastur ishi davomida hosil bo’lishi uchun muloqot oyna nomining chap tarafiga bayroqcha o’rnatish lozim. Muloqot oynalarining asosiy qismida parametr sifatida matn yoki .bmp (16 rangli) fayl nomi so’raladi.

Oddiy hollarda installyatsiya dasturi quyidagi muloqot oynalari bilan chegaralanishi ham mumkin:

Redame Information;

Choose Destination Location;

Select Program Folder;

Progress Indicator

Setup Complete.

Reestrga o’zgartishlar kiritish.

Make Registry changes (reestrga o’zgartish kiritish kiritish) guruh buyruqlari foydalanuvchi kompyuteridagi Windows reestrida o’zgartirishlarni bajarish imkonini beradi. Windows reestri ilovalar, qurilmalar va Windowsning sozlangan parametrlari joylashgan va iyerarxik tashkil etilgan ma’lumotlar bazasini o’zida mujassamlashtiradi.

Ishga tushiruvchi ilova buyrug’i va joylashuvi

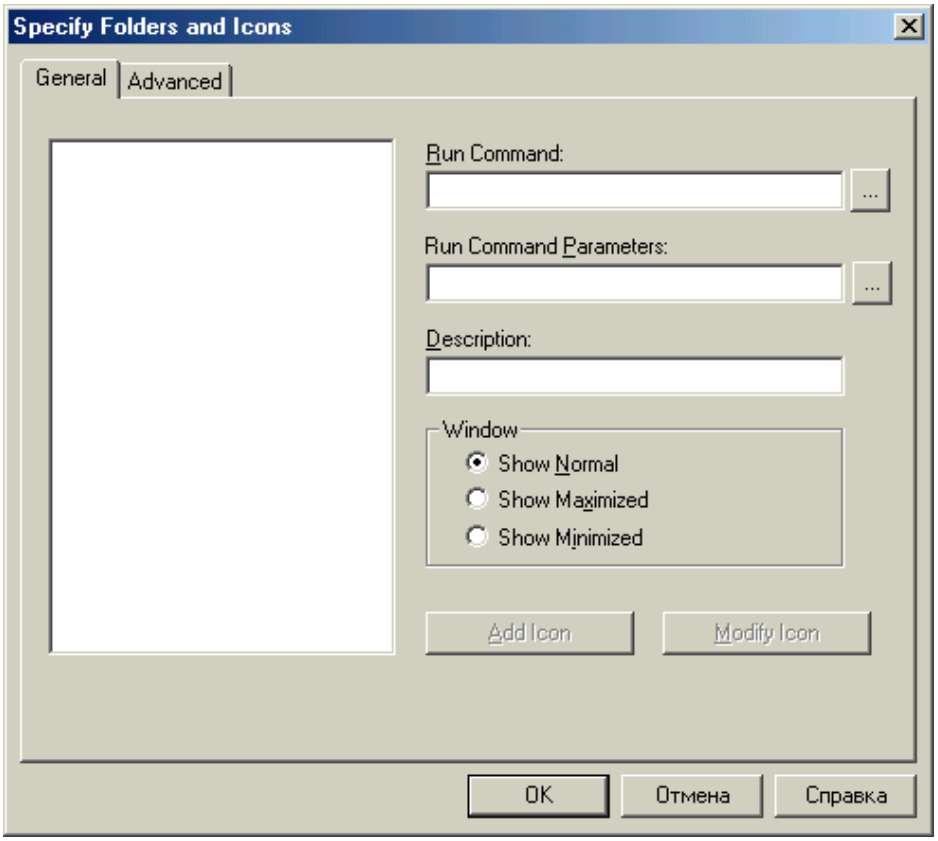
Specify Folders and Icons guruh buyruqlari o’rnatiluvchi dasturning ishga tushiruvchi buyrug’I joylashuvchi menyuni berish, shuningdek isjga tushirish buyrug’I va belgisini berish imkoniyatini yaratadi.

General Settings buyrug’ini tanlash bilan General sahifasi ochiladi (43-rasm). Bu yerda o’rnatiluvchi ilovaning ishga tushiruvchi buyrug’ini berish mumkin. Run Command (Ishga tushirish buyrug’i) maydonida o’rnatiluvchi dasturning bajariluvchi fayl nomi kiritiladi, Description (izoh) maydonida matn kiritiladi. Bu matn buyruqlar menyusida hosil bo’ladi.

Ishga tushiruvchi buyruq va yozuvlar kiritilgandan so’ng AddIcon tugmasi bosiladi, natijada belgilar oynasida izoh va dasturning belgisi hosil bo’ladi. Windows guruhi yordamoda dastur ishga tushirilganda hosil bo’luvchi oynaning o’lchamlarini berish mumkin. Agar Show Maximized tanlansa oyna ekranni to’ldirib turadi, Show Minimized tanlansa oyna eng kichik holatda hosil bo’ladi.

Advanced sahifasining Start in maydonida o’rnatiluvchi ilova dasturining ishchi katalogi va Icon maydonida belgi faylining nomi kiritiladi.

Eslatma: *Ilova belgilari faylining kengaytmasi .ico ko’rinishda bo’ladi. Belgi faylini dasturchini o’zi ham yaratishi mumkin. Masalan, Delphining ImageEditor utility yordamida.*



43-rasm. General saxifasahifasi

Folder guruhi o’rnatiluvchi oliva belgisini qayerga joylashishini ko’rsatish imkonini beradi: Программы (Programs Menu Folder) menyusida, Пуск (Start Menu Folder) menyusida, ishchi stolda (Desktop Folder), Автозагрузка (Startup Folder) yoki Послать (Send to Folder) menyusida.

O’rnatuvchi diskni yaratish

O’rnatiluvchi dasturning barcha tavsiflari aniqlanib bo’linganidan keyin o’rnatuvchi disketlarni yaratishni amalga ishirish mumkin. Bu jarayon quyidagi qadamlarda bajariladi:

o’rnatuvchi disket nushasini yaratish;

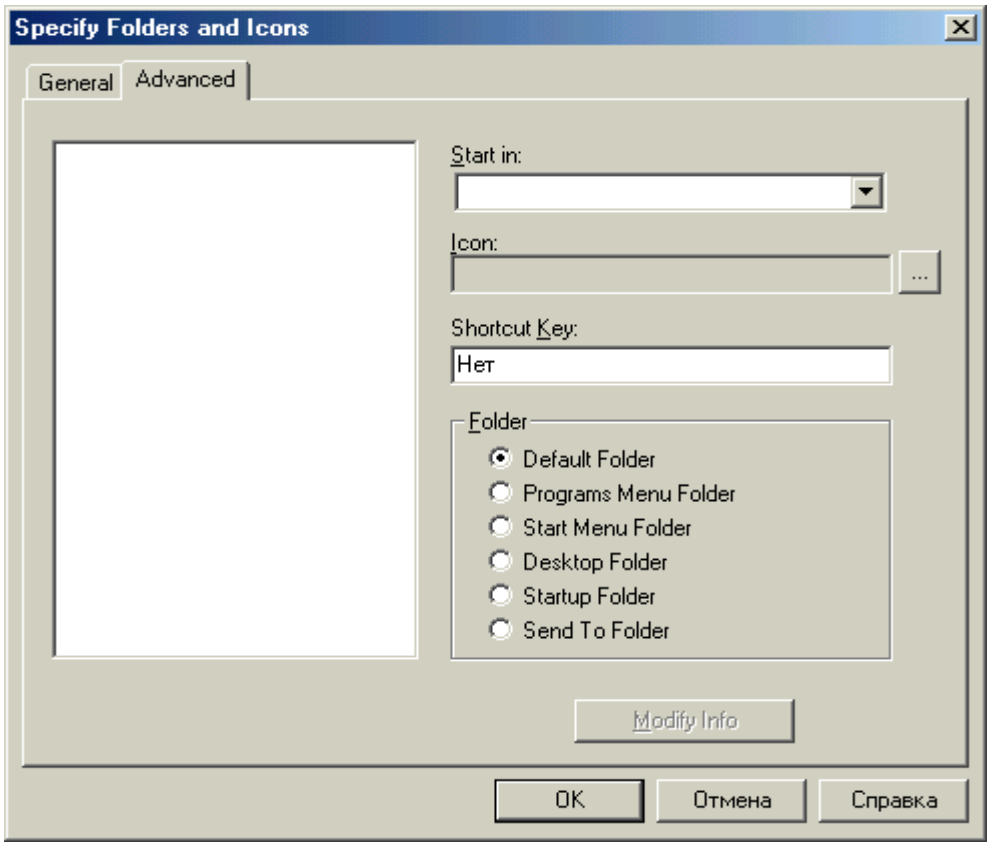
o’rnatish jarayonini tekshirish;

o’rnatuvchi disket nushasini disketga yozish.

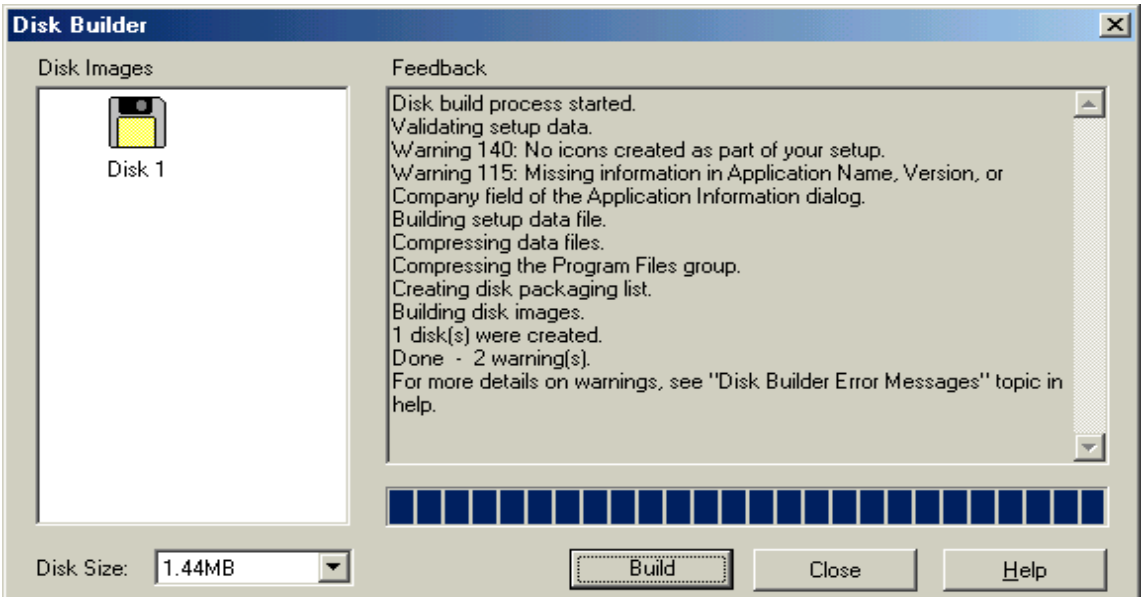
O’rnatuvchi disklarning nushasini yaratish uchun Checklist menyusidan RunDisk Builder buyrug’I tanlanadi. Hosil bo’lgan DiskBuilder muloqot oynasining

Disk Size ochiluvchi ro’yxatidan disk hajmi ko’rsatiladi (44-rasm). Yani, tashkil etiluvchi installyatsion dastur ko’rsatilgan hajmli disklarga joylashtiriladigan qilib yaratiladi (uncha katta bo’lmagan dasturlar uchun 1,44 Mbayt hajmli disklar qo’llaniladi).

Installyatsion dastur tashkil qilingandan keyin uni nechta diskdan joy olishini DiskImages oynasidan bilish mumkin (45-rasm).



44-rasm. Disk Builder muloqot oynasi



45-rasm. Disk Builder muloqot oynasining o’rnatuvchi disk yaratish jarayoni tugatilganidan keyingi ko’rinishi

o’rnatuvchi disklarni yaratish tugatilganidan so’ng uni qanday ishlashini Checklist menyusining Test The Installation buyrug’I yordasmida sinab ko’rish mumkin.

Agar o’rnatuvchi dastur xatosiz ishlasa, uni disklarga ko’chirishni boshlash mumkin.

	O`rnatishga tayyorlanayotgan proyektni boshqa formatdagi disklarga bo`lish uchun qanday yo`l tutasiz.

Foydalaniladigan asosiy darsliklar va o’quv qo’llanmalar ro’yhati

Asosiy darslik va o`quv qo’llanmalar

1. Баженов И. Delphi 7.Самоучитель про., КУДИЧ, 2003 г. (Дарслик)
2. Фаронов В. Delphi: Програмирование на языке высокого уровня, Петербург, 2003 г. (Дарслик)
3. Джулиан Бакнел. Фундаментальные алгоритмы и структуры данных в Delphi. Пер. с англ. – СПб: ООО «ДиаСофтЮП», 2003 г
4. Очков и др.128 советов начинающему программисту, Москва, 1991 г. (Дарслик)
5. Марков Б. Визуалная программирования, Москва, 2003 г. (Дарслик)
6. А. Ануньев, А. Федоров. «Самоучитель Delphi 6.0». «BHV-Санкт-Петербург» 2001 г. (Дарслик)
7. М.МакКелви «Delphi-5» Санкт-Бетербург, 1998 г. (Дарслик)
8. DELPHI NET Complete Publication, NewDehli, 2002 г. (Дарслик)
9. Гринзоу Лу Философия програмирования для WINDOWS 95-NT Символ-Плюс, 2002 г. (Дарслик)

Qo’shimcha adabiyotlar

10. М.Арипов. Информатика, Университет нашриёти, 2001. (Дарслик)
11. Ўзбекистон Давлат таълим стандарти: Ўрта махсус, касб-ҳунар таълими умумтаълим фанлари. — «Маърифат», №86,2000 й. 4 ноябрь. (Ўқув қуланма)
12. Ғуломов С. С, Шермуҳамедов А. Т., Бегалов Б. А. Иқтисодий информатика: «Ўзбекистон», 1999. (Дарслик)
13. Ғуломов С. С. ва бошқалар. Ахборот тизимлари ва технологиялари: Олий ўқув юрти талабалари учун Тошкент «Шарқ», 2000 йил.(Дарслик)

14. www.referat.ru
15. www.intuit.ru
16. www.intuit.ru
17. www.ziyonet.uz