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

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

M.S.Pozilov, M.Y.Xaydarova, F.SH.Qosimov,
U.I.Gapirova, U.A.Shodmonova., I.M.Boynazarov

Delphi 4 dasturlash tili asoslari

O'quv qo'llanma

Toshkent - 2009

Taqrizchilar

AT va DT kafedrasi dotsenti, G'oyibnazarov V. T.

M.S. Pozilov va boshqalar.

Delphi 4 dasturlash tili asoslari oquv qo'llanmasi.

M.S. Pozilov, M. Y. Xaydarova, F. SH. Qosimov, U.I. Gapirova, U.A. SHodmonova, I.M. Boynazorov:

t.f.d, professor B. Sh. Radjabov tahriri ostida-T .: 2009.

Kirish

Hozirgi vaqtga kelib kompyuter olamida ko'plab dasturlash tillari mavjuddir. Ular Beysik, Paskal, Si va boshqa dasturlash tillaridir. Xo'sh ulardan qay biri yaxshiroq? Albatta bu savolga Paskal dasturlash tili deyish mumkin. Chunki, uning boshqa dasturlash tillaridan farqi, u universal dasturlash tili bo'lib, turli xil dasturlarni tuzish va o'rganish uchun qulay hisoblanadi. Bu til 1969 yil N.Virt tomonidan yaratilgan bo'lib, keyinchalik Amerikaning Borland firmasi tomonidan qayta ishlangan va u Turbo-Pascal dasturlash tili deb nomlangan. Turbo-Pascalni qayta ishlash natijasida ob'yektli dasturlash yo'lga qo'yildi va u Object Pascal deb atala boshlandi. Hisoblash texnikasi va texnologiyasining rivojlanishi natijasida Borland firmasi tomonidan yangi Delphi dasturlash tili yaratildi.

Delphi dasturlash tili Windows uchun mo'ljallangan bo'lib, uning birinchi versiyasi Windows 3.1 operatsion sistema qobig'ida ishlangan. Windows 95 operatsion sistema yaratilganidan so'ng, 16-razryadli Delphi 2, keyinroq 32-razryadli Delphi 3 versiyasi yaratildi. Windows 98 operatsion sistemasi uchun Delphining to'rtinchi versiyasi va hozirgi kunda Delphi 7 paydo bo'ldi.

Ushbu o'quv qo'llanmada biz Delphi 4 ob'yektli dasturlashning imkoniyatlari bilan tanishamiz.

Delphi 4 dasturlash tili – dasturlarni qayta ishlash muhiti bo'lib, 32-razryadli Windows operatsion sistemasida ishlaydi. Unda ob'yektli dasturlash tili bo'lgan Object Pascal mujassamlashgan.

Delphi - vizual proyektlar, turli holat proseduralarini va dasturlarni qayta ishlashda vaqtdan yutish va boshqalarni o'z ichiga oladi.

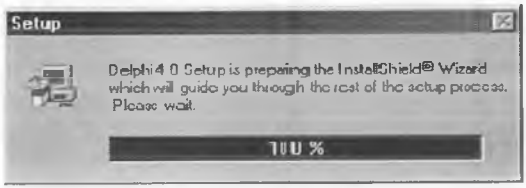
Delphini o`rnatish va ishlatish

Odatda Delphi paketini o'rnatish CD-ROM qurilmasi yordamida amalga oshiriladi. Kompakt 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 hosil 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 Delphi'ni o'rnatuvchi dastur ishga tushadi va oynada **Setup** (o'rnatish) oynasi xosil bo'ladi (1-rasm).

B



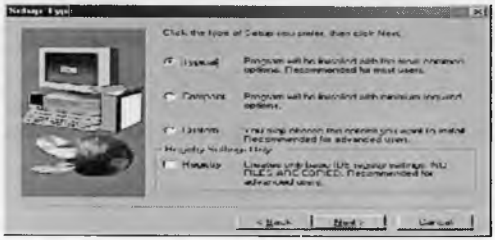
1-rasm. Delphini o'rnatish jarayonini boshlash.

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

2-rasm. Muloqat oynasi.

Undagi **Next**(keyingi) tugmasi bosilsa **Password Dialog** oynasi hosil bo'ladi. Bu oynada **Serial Number** (seriya raqami) va **Authorization key** (komplekt kodi)lar kiritiladi. **Next** tugmasi bosilsa **Software Lieense Agreement** (Litsyenziya kelishuvi)

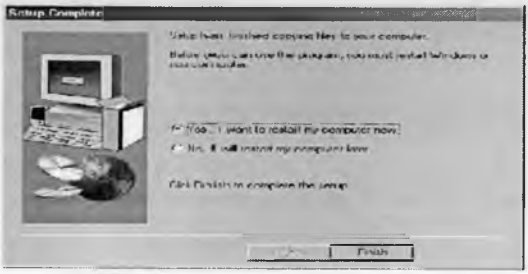
oynasi hosil bo'ladi (agar seriya raqami va komplekt kodi to'g'ri keltirilsa Delphini o'rnatish dastur tomonidan to'xtatiladi). Undagi matnni o'qib **Yes**(ha) tugmasi bosiladi. O'rnatish jarayonida quyidagi oyna hosil bo'ladi (3-rasm):



3-rasm. Setup Type muloqat oynasi.

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 **Setup Complete** oynasi hosil bo'ladi (4-rasm) va **Finish** (yakunlash) tanlanadi.



4-rasm. Setup Complete muloqat oynasi.

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 sichqonchaning chap tugmasini bosish bilan (5-rasm);

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

5-rasm. Windows'ning bosh menyusidan Delphini yuklash.

MS	1. Delphini yana qaysi yo`l bilan ishga tushirish mumkin? 2. Delphi muhitida oynalardan biri yo'q bo`lsa siz qanday qilib ana shu oynani aktivlashtirasiz?

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 kvadrat tenglamaning 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 bosqichida qo'yilgan masala kompyuterga tushiriladi va undagi muammolar ko'rib chiqiladi. Masalan, kvadrat tenglamaning ildizini hisoblash dasturi. Bunda kvadrat tenglama ildizini topish uchun kerakli ma'lumotlar kiritilishi kerak. Natijada «kvadrat tenglamaning 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*.

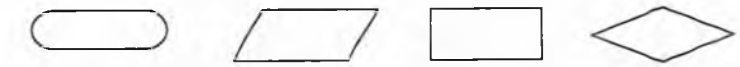
Topiladigan natija – *x1* va *x2* tenglama ildizlari.

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

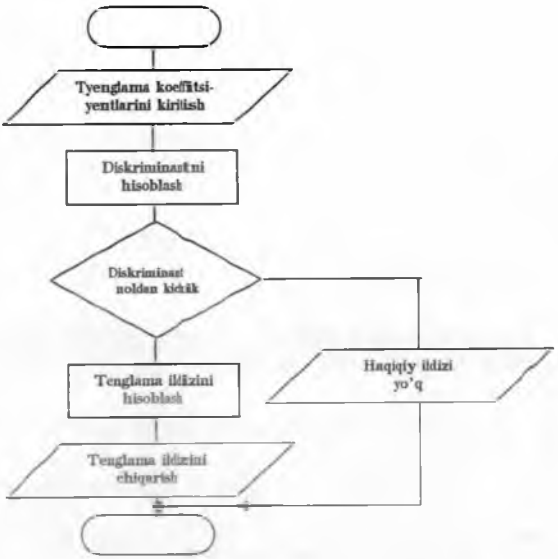
Agar diskriminant natijasi nolga teng yoki katta bo'lsa, u holda quyidagi formula bilan tenglama ildizlari topiladi: $x1 = \frac{-b-\sqrt{d}}{2a}$; $x2 = \frac{-b+\sqrt{d}}{2a}$

Agar diskriminant natijasi noldan kichik bo'lsa, bu tenglamaning xaqiqiy 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 tanlashdir(6-rasm).



6-rasm. Algoritmning blok-sxemasi uchun asosiy shakllar

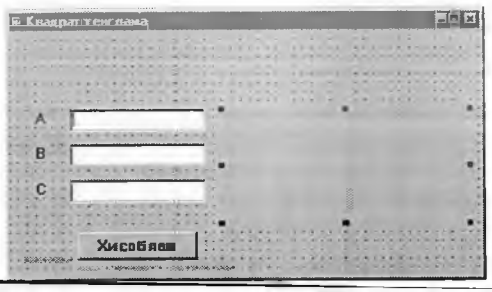


7-rasm. Kvadrat tenglamani hisoblash algoritmining blok-sxemasi

Misol uchun, kvadrat tenglamaning ildizini topish algoritmi 7-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}
```

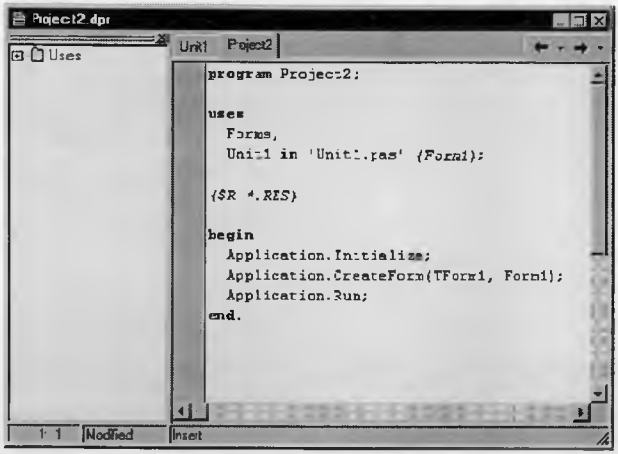
```
procedure TForm1.Button1Click(Sender: TObject);
Var
  a, b, c: Real;  { Tenglama koeffisiyentlari }
  d: Real;      { Diskriminant }
  x1, x2: Real;  { Tenglama ildizlari}
begin
  { Kerakli ma'lumotlarni kiritish }
  a := StrToFloat(Edit1.Text);
  b := StrToFloat(Edit2.Text);
  c := StrToFloat(Edit3.Text);
  { Diskriminantni hisoblash }
  d := b * b – 4 * a * c;
  If d < 0 Then
  Begin
    Label5.Caption := 'Diskriminant noldan kichik' + #13 +
    'Tenglamaning ildizi yo`q.'
  End
  Else
  Begin
    { Ildizlarni hisoblash }
    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 Hisoblash 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 hosil bo'lgandan so'ng, **Project** (Proyekt) menyusidan **View Source** (Ko'rish) tanlanadi. Natijada *Project2.dpr* deb nomlangan (9-rasm) oyna hosil 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`irligini kilogrammda kiriting');
   Writeln('va <Enter> tugmasini bosing');
   Write('→');
   Readln(k);
   F := k * 0.4095;
   Writeln(k: 10: 4,' kilogramm', f: 10: 4,' funt');
   Readln;
End.
```

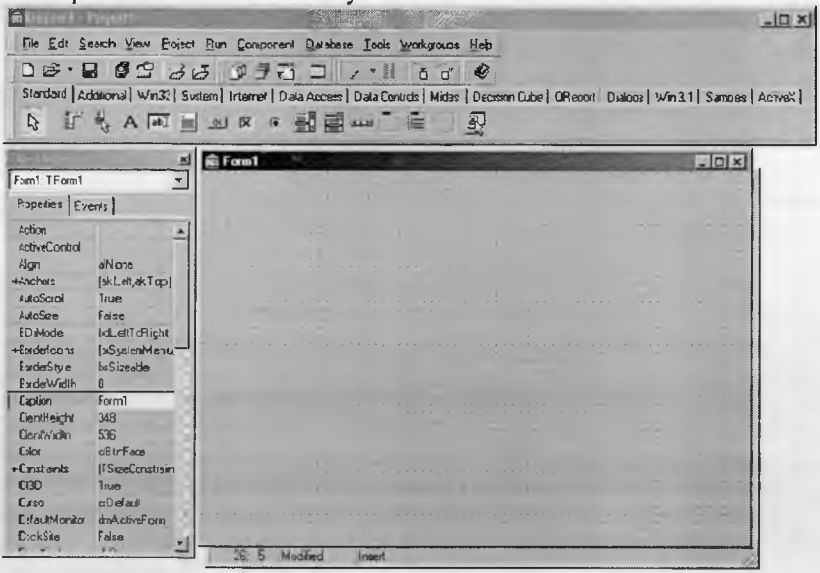
Yuqoridagi dasturda *{$APPTYPE CONSOLE}* qatori mavjud bo'lib, u izoh ko'rinishida yozilgan. Lekin u, dasturning konsolli ilova ekanligini bildiradi. Bunday dasturni tuzishda albatta *{$APPTYPE CONSOLE}* qatori yozilishi shart.

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

1. Algoritmik tillarga qiyosiy tavsifnoma bering.
2. Til alfaviti qanday tushuncha?

Delphi muhiti. Delphi da boshlang'ich amallar va proyektlar

Delphi dasturlash tilini ishga tushirganimizda uning ishchi oyna ko'rinishini ko'ramiz (qanday ishga tushirish avvalgi ma'ruzada ko'rib o'tilgan). U unchalik oddiy emas (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.



10-rasm.

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: Delfining 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 tushirishning 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 protsedura, funktsiya matnlari, tiplarning 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 harilar bilan belgilaydi.

Malumot 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

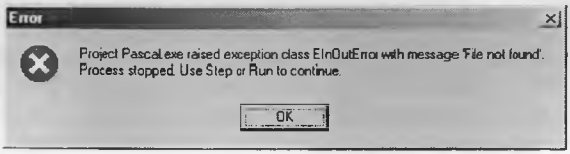
Odatda dastur tuzilayotganda ba'zi kamchilik yoki xatoliklarga yo'l qo'yamiz, dasturni ishga tushirgan vaqtimizda esa bu xatoliklar to'g'risida axborot beruvchi oyna hosil bo'ladi va bu xatoliklar quyidagilarga bo'linadi: Sintaktik; Dastur bajarilish vaqtidagi xatoliklar; Algoritmik.

Sintaktik xatoliklar operator, prosedura-funksiya, o'zgaruvchi nomi va boshqalarni noto'g'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 hosil 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 hosil 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 hosil 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`g'ri kiritilishi qaysi xatolik turiga kiradi?

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, SmallInt, 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. Asosiy 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.

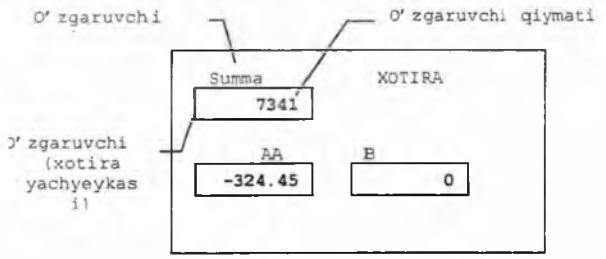
Bu amallarni faqat mantiqiy o'zgarmaslar sirtidagina ishlatish mumkin va natijada yana mantiqiy o'zgarmas hosil bo'ladi. Quyida mantiqiy o'zgarmaslar sirtida amallar jadvali ko'rsatilgan:

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 ma'lumot 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 koeffitsiyentlari, 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 u 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 opostrof (') 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, +, -.

Ifodaning 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 funksiyalarning 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. Tenglikning 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

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

Noto'g'ri tuzilgan dastur

```
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. Xizmatchi so`zlarni yozing va ularning vazifalarini tushuntiring.
2. O`zgaruvchi va o`zgarmaslarning farqini aniqlang.

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.

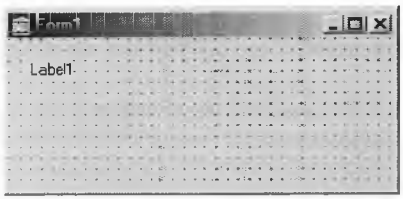
Komponentlar palitrasi guruhlar bo`yicha sahifalarga bo`lingan bo`lib, sahifalar soni 14 tadir. Komponentlar palitrasida **Standard, Additional, Dialogs** kabi sahifalar mavjud. Istalgan sahifa sirtida sichqonchaning chap tugmasi bosilsa, tanlangan sahifadagi komponentlar ro'yhati chiqarib beriladi. 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**

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

Ko`rsatkich	Vazifasi
taLeftJ sirti fy	**Matn ob`ektning *chap* qismida**
taRightJ sirtify	**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 formaning *yuqori* qismi bo`yicha**

	o`zgarmaydi
taLeft	**Ob`ektning turgan joyi formaning *chap* qismi bo`yicha o`zgarmaydi**
taRight	**Ob`ektning turgan joyi formaning *o`ng* qismi bo`yicha o`zgarmaydi**
taButton	**Ob`ektning turgan joyi formaning *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`zgarmas 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 formaning chap qismiga nisbatan joylashish o`rni (piksellarda).

Name – ob`ektning nomi. Ob`ekt nomining birinch harfi 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 turidir. Misol uchun, 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:

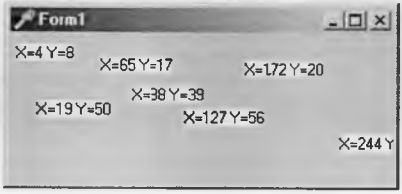
```
procedure TForm1.FormClick(Sender: TObject);
begin
  MessageDlg('Salomlar !!!', mtInformation, [mbOk], 0);
end;
```

Yuqoridagi amallar bajarilgach, dastur ishga yuklanidan so`ng formaning ishalgan 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 bilib 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:

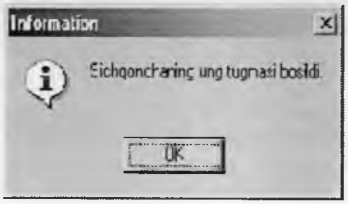
```
unit Unit1;

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:



MS	Xodisaning shunday turini tanlangki, uning yordamida formadagi tugmachalarning necha marta boslganligini bilish mumkin bo`lsin.

Operatorlar (Buyruqlar). O'tish operatori (Goto)

Odatda dastur o'z ishini, yozilgan operatorlar ketma-ketligi bo'yicha amalga oshiradi. Operatorlarning tabiiy bajarilish ketma-ketligini buzish uchun shartsiz o'tish operatoridan foydalaniladi. Dasturning bir operatoridan boshqarishni boshqa operatorga

uzatish uchun boshqarilish uzatiladigan operator oldiga belgi (metka) qo'yilishi kerak. Boshqarishni shartsiz uzatish operatori quyidagicha yoziladi: **goto** <metka>; bu holda boshqarish ko'rsatilgan metkali qatorga uzatiladi. Yuqorida aytganimizdek dasturda qatnashgan barcha metkalar dasturning metkalar bo'limida e'lon qilinishi kerak:

Uses <Modullar>;
Label <Metkalar>;
Var
Begin

End.

O'tish operatoriga doir misol:

```
A := 5.75;
B := spr(a);  goto L5;
C := 9.76;
L5:   d := a + b;
```

Dasturdagi C := 9.76 operatoridan boshqa barcha operatorlar bajariladi.

Umuman olganda, dasturchi iloji boricha o'tish operatoridan foydalanmaslikka harakat qilgani ma'quldir. Chunki o'tish operatoridan foydalanish dasturni o'qishni qiyinlashtirib yuboradi.

Shartlar

Algoritmlar nazariyasidan ma'lumki, hisoblash jarayonlarini shartli ravishda uch xil guruhga ajratish mumkin:

- **Chiziqli jarayonlar;**
- **Tarmoqlanuvchi jarayonlar;**
- **Takrorlanuvchi jarayonlar.**

Chiziqli jarayonni hisoblash algoritmi qat'iy ketma-ketlik asosida amalga oshiriladi. Bunday jarayonni hisoblash uchun o'zlashtirish operatorining o'zi yetarli bo'ladi.

Tarmoqlanuvchi jarayonni hisoblash yo'li ma'lum bir shartni bajarilishi yoki bajarilmasligiga qarab tanlanadi. Tarmoqlanuvchi jarayonlarni hisoblash uchun shartli operatordan foydalaniladi. Shartli operator ikki xil ko'rinishda bo'ladi:

- to'liq shartli operator;
- chala shartli operator.

To'liq shartli operator quyidagi ko'rinishda yoziladi:

<to'liq shartli operator> := **if** <mantiqiy ifoda> **then** <operator> **else** <operator>;

bu yerda **if** (agar), **then** (u holda), **else** (aks holda) xizmatchi so'zlar. Shunday qilib, to'liq shartli operatorni soddaroq quyidagicha yozish mumkin:

if S **then** S1 **else** S2;

bu yerda S – mantiqiy ifoda;

S1 – S mantiqiy ifoda rost qiymat qabul qilganda ishlaydigan operator;

S2 – S mantiqiy ifoda yolg'on qiymat qabul qilganda ishlaydigan operator.

Shartli operatorning bajarilishi unda yozilgan S1 yoki S2 operatorlaridan birini bajarilishiga olib keladi, ya'ni agar S mantiqiy ifoda bajarilishidan so'ng (*true*) rost qiymati hosil bo'lsa S1 operatori, aks holda (*false*) esa S2 operatori bajariladi.

To'liq shartli operatorga doir misollar:

1. **if** a = 2 **then** d := x + 2 **else** d := x - 2;
2. **if** (x < y) **and** z **then**
   begin
   y := x * sin(x);
   t := x * cos(x)
   end
   else
   begin
   y := 0;
   t := 1
   end;
3. **if** (x < 0) **or** (x = 3) **then** y := x * x + 1 **else if** x < 2 **then** y := sqr(abs(x - 1)) **else** y := x * x;

Chala (to'liqmas) shartli operatorning yozilishini quyidagicha ifodalasa bo'ladi:

if S **then** S1;

bu yerda S - mantiqiy ifoda, S1 - operator.

Agar S ifoda qiymati ***true*** (rost) bo'lsa S1 operatori bajariladi, aks holda esa boshqarish shartli operatordan keyin yozilgan operatorga uzatiladi.

Bu ikki xil shartli operatorlardan bir xil maqsadda bemalol foydalansa bo'laveradi.

Shartli operatordan foydalanib dastur tuzish uchun quyidagi misolni ko'rib chiqaylik:

$$y = \begin{cases} ax + b & \text{arap } x > 0 \\ cx + d & \text{arap } x \le 0 \end{cases}$$

bu yerda faraz qilaylikki a = 1,5 ; b = 4 ; c = 3,7; d = 4,2.

x - esa qiymati beriladigan noma'lum o'zgaruvchi.

"y" funksiyasini hisoblash dasturini tuzish talab etilsin.

1. To'liq shartli operatordan foydalanib tuzilgan dastur:

```
var
  x, y, a, b, c, d: real;
  s: string;
begin
  s := InputBox('','','0');
  x := StrToFloat(s);
  a := 1.5; b := 4; c := 3.7; d := -4.2;
  if x > 0 then y := a * x + b else y := c * x + d;
  Label1.Caption := IntToStr(y);
end;
```

2. Chala shartli operatordan foydalanib tuzilgan dastur:

```
label L1;
```

```
var
 x, y, a, b, c, d: real;
 s: string;
begin
 s := InputBox( '','','0');
 x := StrToFloat(s);
 a := 1.5;  b := 4;  c := 3.7; d := -4.2;
 if x > 0 then
  begin
   y := a * x + b;
   goto L1
  end;
 y := c * x + d;
L1:
 Label1.Caption := IntToStr(y);
end;
```

Takrorlanuvchi (sikl) operatorlar

Takrorlanuvchi jarayonlarni yuqorida sanab o`tilgan operatorlardan foydalanib ham tashkil etsa bo'ladi, lekin bunday jarayonlarni takrorlash operatorlari yordamida amalga oshirish osonroq kechadi. Takrorlash operatorlarining 3 xil turi mavjud bo`lib. ular quyidagilardir:

- **parametrli takrorlash operatori;**
- **repeat takrorlash operator;**
- **while takrorlash operatori.**

Yechilayotgan masalaning mohiyatiga qarab dasturchi o'zi uchun qulay bo'lgan takrorlash operatorini tanlab olishi mumkin.

Parametrli takrorlash operatori (For)

For operatorni quyidagicha ko'rinishi amalda ko'proq ishlatiladi:

for k := k1 **to** k2 **do** S;

bu yerda ***for*** (uchun), ***to*** (gacha), ***do*** (bajarmoq) - xizmatchi so'zlari;

k - sikl parametri (haqiqiy tipli bo'lishi mumkin emas);

k1 - sikl parametrining boshlang'ich qiymati;

k2 - sikl parametrining oxirgi qiymati;

S - sikl tanasi.

Operatorning ishlash prinsipi: sikl parametri (SP) boshlang'ich qiymat k1 ni qabul qilib agar bu qiymat k2 dan kichik bo'lsa shu qiymat uchun S operatori bajariladi;

SP ning qiymati yangisiga o`zgartirilib (agar k son bo'lsa o'zgarish qadami 1 ga teng, belgili o'zgaruvchi bo`lsa navbatdagi belgini qabul qiladi, va x.k.) yana S operatori bajariladi va bu jarayon k > k2 bo'lguncha davom ettiriladi. Shundan so'ng sikl operatori o'z ishini tugatib boshqarishni o'zidan keyingi operatorga uzatadi.

Parametrli takrorlash operatorining necha marta qaytadan takrorlanishini aniq bilsakgina undan foydalanish maqsadga muvofiq bo`ladi.

Misol: $S = \sum_{i=1}^{n} \frac{1}{i}$ yig'indining *n* ta hadi yig'indisini topish dasturini tuzish.

Masalaning formasi quyidagicha bo'ladi:

Masalaning dasturi esa quyidagicha:

```
 unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    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
   S: String;
   i, n: Integer;
   Summ: Real;
begin
   S := InputBox('Kiritish oynasi', 'N ni kiriting', '');
   N := StrToInt(S);
   Summ := 0;
   For I := 1 to n do Summ := Summ + (1 / i);
   Label2.Caption := 'Summa= ' + FloatToStr(Summ);
end;
end.
```

Ayrim paytlarda sikl parametrini o'sib borish emas, balki kamayish tartibida o'zgartirish mumkin, bu holda sikl operatori quyidagi formada yoziladi:

for k:=k2 **downto** k1 **do** S;

bu yerda ***downto*** (gacha kamayib) Paskal tilining xizmatchi so'zi. Bu operatorda k parametri k2 dan toki k1 gacha kamayish tartibida (agar k - butun qiymatli o'zgaruvchi bo'lsa sikl qadami - 1 ga teng) o'zgaradi. Operatorning ishlash prinsipi oldingi operatordagiday qolaveradi. Misol: yuqorida ko'rsatilgan misol dasturini qaytadan tuzaylik. Bu holda dasturdagi sikl operatorigina o'zgaradi holos:

for I:=n **downto** 1 **do**

qolgan operatorlar o'z o'rnida o'zgarmay qoladi.

Repeat takrorlash (sikl) operatori.

Yuqorida aytib o'tganimizdek sikldagi takrorlanishlar soni oldindan ma'lum bo'lsa parametrli (*for*) sikl operatori foydalanish uchun juda qulay. Lekin, ko'pgina hollarda siklik jarayonlardagi takrorlanishlar soni oldindan ma'lum bo'lmaydi, balki sikldan chiqish ma'lum bir shartning bajarilishi yoki bajarilmasligiga bog'liq holda bo'ladi. Bu hollarda ***repeat*** yoki ***while*** sikl operatorlaridan foydalanish zarur. Agar sikldan chiqish sharti siklik jarayonning oxirida joylashgan bo'lsa **repeat** operatoridan, bosh qismida joylashgan bo'lsa **while** operatoridan foydalanish maqsadga muvofiqdir. Repeat operatorining yozilish formasi quyidagicha bo'ladi:

repeat S1; S2; ... SN **until** B;

bu yerda ***repeat*** (takrorlamoq), ***until*** (gacha) - xizmatchi so'zlar;
S1, S2, ..., SN lar esa sikl tanasini tashkil etuvchi operatorlar;
B - sikldan chiqish sharti (mantiqiy ifoda).

Operatorning ishlash prinsipi juda sodda, ya'ni siklning tanasi B mantiqiy ifoda rost qiymatli natija bermaguncha takror - takror hisoblanaveradi. Misol sifatida yana yuqoridagi yig'indi hisoblash misolini olaylik. Bu yerda forma o'zgarmaydi lekin. *TForm1.Button1Click* proseduriga o'zgartirish kiritiladi:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  S: String;
  i, n: Integer;
  Summ: Real;
begin
  S := InputBox('Kiritish oynasi', 'N ni kiriting', '');
  N := StrToInt(S);
  Summ := 0;
  I := 1;
  Repeat
      Summ := Summ + (1 / I);
      I := I + 1;
  Until I > N;
  Label2.Caption := 'Summaq ' + FloatToStr(Summ);
end;
```

While takrorlash (sikl) operatori

Ahamiyat bergan bo'lsangiz, repeat operatorida siklning tana qismi kamida bir marta hisoblanadi. Lekin, ayrim paytlarda shu bir marta hisoblash ham yechilayotgan masalaning mohiyatini buzib yuborishi mumkin. Bunday hollarda quyidagi formada yoziluvchi while sikl operatoridan foydalanish maqsadga muvofiqdir:

while B **do** S;

bu yerda **while** (hozircha), **do** (bajarmoq) - xizmatchi so'zlari;
B - sikldan chiqishni ifodalovchi mantiqiy ifoda;
S - siklning tanasini tashkil etuvchi operator.

Bu operatorda avval B sharti tekshiriladi, agar u false (yolg'on) qiymatli natijaga erishsagina, sikl o'z ishini tugatadi. aks holda siklning tana qismi qayta - qayta hisoblanaveradi. While operatoriga misol sifatida yana yuqorida berilgan yig'indi hisoblash misolini ko'rib chiqaylik:

Bu yerda ham forma o'zgarmaydi lekin, TForm1.Button1Click prosedurasiga o'zgartirish kiritiladi.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  S: String;
  i, n: Integer;
  Summ: Real;
begin
 S := InputBox('Kiritish oynasi', 'N ni kiriting', '');
 N := StrToInt(S);
 Summ := 0;
 I := 1;
 While I <= N do
 Begin  Summ := Summ + (1 / N);
    I := I + 1;
 End;
 Label2.Caption := 'Summaq ' + FloatToStr(Summ);
end;
```

Variant tanlash operatori (Case)

Ayrim algoritmlarning hisoblash jarayonlari o'zlarining ko'p tarmoqliligi bilan ajralib turadi. Umuman olganda, tarmoqli jarayonlarni hisoblash uchun shartli operatordan foydalanish yetarlidir. Lekin, tarmoqlar soni ko'p bo'lsa shartli operatordan foydalanish algoritmning ko'rinishini qo'pollashtirib yuboradi. Bu hollarda shartli operatorning umumlashmasi bo'lgan variant tanlash operatoridan foydalanish maqsadga muvofiqdir.

Variant tanlash operatorini sintaktik aniqlanmasi quyidagicha:

<variant tanlash operatori> := **case** <operator selektori> **of**
    <variant ro'yxatining hadlari>
end;

Variant tanlash operatorining bajarilish paytida oldin selektorning qiymati hisoblanadi, shundan so'ng selektorning qiymatiga mos bo'lgan metkali operator bajariladi va shu bilan variant tanlash operatori o'z ishini yakunlaydi. Shuni esda tutish kerakki, <variant metka>si bilan <operator metka>si bir xil tushuncha emas va variant

metkasi metkalar bo'limida ko'rsatilmasligi kerak. Bundan tashqari ularni o'tish operatorida ishlatilishi mumkin emas. Misollar:

1. **Case i mod 3 of**
0: m := 0;
1: m := -1;
2: m := 1
end;

2. **Case** summa **of**
'q': k := 1;
'*', '+', '/', '-' : ;
'!' : k := 2;
':', ';': k := 3
end;

3. **Case** kun **of**
dush, sesh, chor, pay, jum: ShowMessage('ish kuni');
shan, yaksh: ShowMessage('dam olish kuni')
end;

Variant tanlash operatori ichiga kirish faqat *case* orqali amalga oshiriladi.
Endi shartli operatorni variant tanlash operatori orqali ifodasini ko'rib chiqaylik:

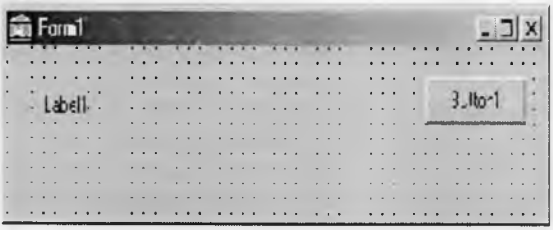
1. **if** B **then** S1 **else** S2	**Case** B **of** true: S1; false: S2; **end**;
2. **if** B **then** S	**Case** B **of** true: S; false: ; end.

Qanday hollarda **While** yoki **Repeat** buyruqlaridan foydalaniladi?

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;
```

Satrlar

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.

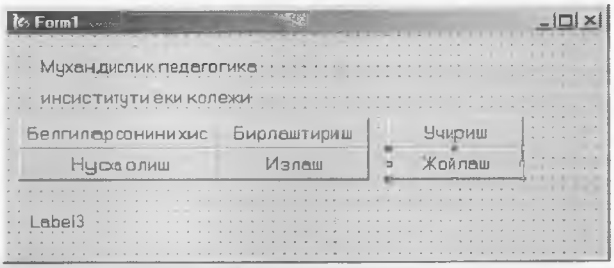
Satrlar va belgilar sirtida turli amallar bajarish mumkin. Ular quyidagilar:

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('xa ', S, 13);
    Label3.Caption := S;
end;
end.
```

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

Massivlar. Massivlar ustida 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 [1 15..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 ustida 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 o'rni' + 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 ustida 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 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 hadlarini kiritish}
    st:=InputBox('B massiv elementlari', '', '');
    B[i,j] := StrToFloat(st); { massiv hadlarini 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 dasturini tuzing.

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 ustida 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 murojaat qilinmoqda}
```

```
 u := S;
 a := 0.5;  b := u;
 max1;  {max1 proseduraSiga 2-marta murojaat qilinmoqda}
 v := S;
 ShowMessage(FloatToStr(u)+' '+FloatToStr(v));
end;
```

Asosiy programmaning operatorlar qismida ikki marta yozilgan max1 parametrsiz proseduraSiga murojaat, 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 murojaat 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 murojaat vaqtida shu tipli ifoda yozish mumkin bo`lsa, parametr – o`zgaruvchi uchun bu hol mutlaqo mumkin emas.

Prosedurani 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.

Vektoming 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 murojaat *max1(x, u); max1(y, v);* ko`rinishida amalga oshiriladi.

Proseduradagi A vektorini parametr - qiymat sifatida yozib qo`yganimiz uchun, proseduraga qilinayotgan har bir murojaatda A vektorga mos ravishda X va Y vektorlari ko`chirib yoziladi va so`ng prosedura o`z ishini bajaradi. Biz bilamizki, bir tarafdan, massivlarning ustida 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 sintaktik 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 sintaktik 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'yxati>)

Prosedura ismi dasturchi tomonidan tanlanadigan oddiy identifikator hisoblanadi.

Formal parametrlar ro'yxati quyidagicha aniqlanadi:

<formal parametrlar ro'yxati> := <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'yxati (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 mohiyatidan kelib chiqqan holda tanlaydi.

Misol:

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

bu yerda formal parametrlar ro'yxati 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'yxatini 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 murojaat yoki prosedura operatoridan qanday foydalanishni aniqlashni ko`rib chiqaylik (yaratilgan prosedurani «Aktivlashtirish», ya`ni ishlatish):

<prosedura operatori> := <prosedura ismi>| <prosedura ismi>(<faktik parametrlar ro'yxati>)

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

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

<faktik parametrlar ro'yxati> := <faktik parametr>{,<faktik parametr>}

Kvadrat tenglamaning ildizini aniqlab beruvch prosedura yarating va uni faollashtiring.

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 parallel 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 parallel, 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 parametrlari;
n – P proseduradagi lokal o`zgaruvchi.

Matematika kursidan funksiya tushunchasi bizga yaxshi tanish bo`lib, uning yordamida funksiya va argument 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'yxati>):<funksiya qiymatining tipi>;
label   <metkalar ro'yxati>;
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 deyarli 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 murojaat ham xuddi proseduradagi kabi amalga oshiriladi, lekin funksiyaning mos tipli ifodada qatnashish kabi qo`shimcha imkoniyati mavjud.

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

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

$$Y = \frac{20!+3!}{5!+31!} + \frac{(k+1)!}{m!}$$

```
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 aniqlanyapti, 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 murojaat 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 funksiyaning 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;

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

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.

Yangi tiplarni hosil qilish
Sanalma tiplar

Amalda turli xil tipdagi qiymatlar bilan ishlashga to'g'ri keladi. Masalan, rang tushunchasi qizil, qora, oq, sariq, kulrang va h. k.larni o'z ichiga oladi, yoki yil oylari tushunchasi yanvar, fevral, ..., dekabr kabi 12 ta oyni o'z ichiga oladi. Bunday qiymatli tiplarni sonlar orqali ifodalab olsa ham bo'ladi, lekin bu belgilab olish ularning mohiyatini yo'qotib, tushunishga qiyin holni hosil qiladi. Masalan:

if k = 7 **then**

dastur qatorini o'qib gapni nima haqida ketayotganligini dabdurustdan anglash qiyin. Ehtimol, gap bu yerda 7 - oy haqidadir, balki "k" o'zgaruvchini 7 butun soni bilan tekshirilayotgandir. Shunday qilib, "7" soni ostida nima yashiringanini bilish juda qiyin. Lekin, dasturning bu qatori

if k = iyul **then**

bo'lsa, gap yilning **iyul** oyi haqida ketayotganligini osongina anglash mumkin.

Yoqoridagi tushunmovchiliklarni bartaraf qilish, dasturni o'qishga qulayligini oshirish uchun qiymatlar tiplarining sanalma tipi kiritilgan.

Standart tiplar ichida bu tipga misol qilib *boolean* (mantiqiy) tipini ko'rsatish mumkin: boolean = (false, true). Sanalma qiymat tipini quyidagicha aniqlanadi:

<sanalma tipi> := (<ism>,<ism>,...)

yoki

<sanalma tipi> := (<ism> {,<ism>})

bu yerda kichik qavs ichidagi o'zaro vergul bilan ajratilgan <ism>lar aniqlangan tipning o'zgarmaslari hisoblanadi, ularning qavs ichiga olib yozilgan birikmasi esa shu tipning qiymatlar to'plami hisoblanadi. Sanalma tip qiymatlari qat'iy noldan boshlab nomerlangan.Masalan:

(dushanba, syeshanba, chorshanba, payshanba, juma, shanba, yakshanba)

sanalma tipi 7 ta haddan iborat bo'lib, bu yerda quyidagi hol o'rinlidir:

dushanba < seshanba < chorshanba < payshanba < juma < shanba < yakshanba,

ya'ni dushanba 0 - tartib raqamiga , seshanba 1-tartib raqamiga ega va h.k.

Bu tip dasturning yangi tiplar bo'limida aniqlanadi. Sanalma tipni aniqlashga doir misollar:

```
type
  Rang = (qizil, safsar, sariq, kuk, havorang, kulrang, qora, oq);
  Hafta = (dush, sesh, chor, pay, jum, shan, yaksh);
  Mevalar = (olma, nok, shaftoli, uzum);
  Gul = Rang;
```

bu yerda biz to'rtta sanalma tip kiritdik, oxirgi ***Gul*** tipi ***Rang*** tipi bilan bir xil qilib aniqlandi.

Suni esda tutish kerakki bir ismda har xil tip qiymatlari bo'lishi mumkin emas. Masalan yuqoridagi tiplarning safiga

Zirovor = (zira, qalampir, olma)

tipini qo'shish mumkin emas, chunki **olma** qiymati **Mevalar** tipida aniqlangan edi. Bunday tip e'lon qilish dasturni xatoligini anglatadi.

Dasturning *type* bo'limida aniqlab qo'yilgan tiplardan o'zgaruvchilarni tiplarini e'lon qilish bo'limida xuddi standart tiplar kabi foydalanilsa bo'ladi:

```
var
  Kun: Hafta;
  shar, kub: Rang;
```

Yangi sanalma tiplarni o'zgaruvchilarning tiplarini e'lon qilish bo'limida ham kiritish mumkin:

```
var
```

A, B: (stul, divan, stol, shkaf, parta);

Lekin, kiritilgan bu tip ismsiz bo'lganligi uchun bu tipga dasturning boshqa joylaridan murojaat qilish mumkin emas. Suning uchun, sanalma tipni aniqlashning birinchi usuli ma'qulroqdir.

Sanalma tipli "x" argumenti uchun *Succ*(x), *pred*(x) va ord(x) standart na'munalar funksiyalari mavjud. Yuqoridagi aniqlangan tiplarga doir misollardan ko'rib chiqaylik.

Faraz qilaylik, x = seysh qiymatli bo'lsin.

succ(x) = chor {x dan keyingi qiymat},
pred(x) = dush {x dan oldingi qiymat}.
org(x) = 1 {x qiymatning tartib raqami},

```
for x = seysh to yaksh do S;
```

Sanalma tipli qiymatlar sirtida hech qanday amalni bajarib bo'lmaydi.

Sanalma tipli qiymatlarni chop etish uchun odatda variant tanlash operatoridan foydalaniladi. Sanalma tipga misol sifatida quyidagi dasturni ko'rib chiqishimiz mumkin:

```
type
  TRang = (qizil, safsar, sariq, kuk, havo rang, kulrang, qora,oq);
Var
  Rang: TRang;
  S: string;
Begin
  Rang := sariq;
  Case Rang of
     qizil: S := 'QIZIL';
    safsar: S := 'SAFSAR';
     sariq: S := 'SARIQ';
       kuk: S := 'KUK';
  havorang: S := 'HAVORANG';
   kulrang: S := 'KULRANG';
      qora: S := 'QORA';
        oq: S := '0Q';
  end;
  ShowMessage(S + ' rang tanlandi');
End;
```

Cheklangan tiplar

Faraz qilaylik, "n" o'zgaruvchisi dasturda qaysidir oyning biror kunini ifodalovchi butun son bo'lsin. Bu o'zgaruvchini *integer* tipi bilan e'lon qilsak "n"ga ixtiyoriy butun sonni o'zlashtirish mumkin. Lekin, yechilayotgan masalaning mohiyatiga ko'ra "n"ning faqat [1; 31] oraliqdagi qiymatlarigina ma'noga ega holos. O'zgaruvchining boshqa qiymatga ega bo'lishi uning xato hisoblanayotganligini yoki dasturga berilgan ma'lumotlarning xatoligini anglatadi. Shunga o'xshash, dasturchi dasturni tuzish davomida ma'lum bir o'zgaruvchilar qiymatlarining o'zgarish oraliqlari haqida ma'lumotga ega bo'ladi. Bunday ma'lumotlarning dasturda ko'rsatib qo'yilishi dastur ishining to'g'ri bajarilayotganligi sirtidan nazorat qilib turish imkoniyatini yaratadi.

Shunday cheklashlarni amalga oshirish uchun Objekt Paskal tilida cheklangan tiplar kiritilgan. Har bir shunday tip oldindan ma'lum bo'lgan tiplarga cheklashlar kiritish orqali aniqlanadi.

Cheklangan tiplar quyidagicha aniqlanadi:

<cheklangan tip> := <o'zgarmas1>..<o'zgarmas2>

bu yerda <o'zgarmas1> va <o'zgarmas2>lar cheklangan tip kiritilayotgan, oldindan aniqlangan tipning o'zgarmaslaridir, hamda <o'zgarmas1> <<o'zgarmas2> sharti bajarilishi kerak. Cheklangan tiplarga doir misollar:

1..20 (integer tipidagi cheklanma);
dush..jum (sanalma tipli cheklanma);
'A'..'Z' (char tipidagi cheklanma).

Yuqorida aytganimizdek, cheklanma oldindan aniqlangan tipga nisbatan aniqlanadi. Masalan, (dush, sesh, chor, pay, jum, shan, yaksh) sanalma tipini aniqlamasdan turib dush..jum cheklangan tipini kiritish mumkin emas.

Cheklanma tip ham boshqa tiplar kabi yangi tip aniqlash bo'limida yoki o'zgaruvchilarning tiplarini e'lon qilish bo'limida aniqlanishi mumkin.

Cheklangan tiplarni e'lon qilishga doir bo'lgan misollardan yana ko'rib chiqaylik:

```
type
    Hafta = (dush, sesh, chor, pay, jum, shan, yaksh);
    Figura = (piyoda, ot, fil, ferz, shoh);
    Engil_Figura = piyoda..fil;
    Ish_Kunlari = dush..jum;
    Indeks = 10..20;
var
    x, y, z: real;
    i, j: integer;
  Kun: Hafta;
  L: Indeks;
    Ish_Kuni: Ish_Kunlari;
    Dam_Olish_Kuni: Shan..yaksh;
```

Misol sifatida quyidagi dasturni ko`rib chiqaylik:

```
type
```

```
  TRang = 0..7;
const
  qizil = TRang(0);
  safsar = TRang(1);
  sariq = TRang(2);
  kuk = TRang(3);
  havorang = TRang(4);
  kulrang = TRang(5);
  qora = TRang(6);
  oq =TRang(7);
Var
  Rang: TRang;
  S: string;
Begin
  Rang := Random(7);
  Case Rang of
     qizil: S := 'QIZIL';
    safsar: S := 'SAFSAR';
     sariq: S := 'SARIQ';
      kuk: S := 'KUK';
  havorang: S := 'HAVORANG';
   kulrang: S := 'KULRANG';
     qora: S := 'QORA';
      oq: S := 'OQ';
  end;
  ShowMessage('Dastur ' + S + ' rangni tanladi');
End;
```

Oddiy kombinatsiyali tiplar

Soddalik uchun, bir avlodli struktura orqali yozilgan kombinatsiyali tip ma'lumotlari bilan tanishib chiqaylik.

Yozuvlarni aniqlash (kiritish) quyidagi sintaksis qoida bo'yicha amalga oshiriladi:

<kombinatsiyali tipni aniqlash> := **record** < maydonlar ro'yhati> **end;**
<maydonlar ro'yhati> := <yozuv seksiyasi>{; <yozuv seksiyasi>};
<yozuv seksiyasi> := <maydon ismi>{, <maydon ismi>}: <tip>

Shu qoidaga asoslanib, kompleks sonli tip kiritaylik (a + bi -kompleks son, a, b - haqiqiy sonlar, i^2=-1) va tip nomini *complex* deb ataylik:

```
type
  CompLex = record
          re: real;
          im: real;
  end;
```

Bu tipni quyidagicha tushuntirish mumkin: *CompLex* tipiga tegishli ixtiyoriy qiymat ikkita hadli (maydonli) yozuvdan tashkil topgan strukturadir. Yozuv maydonlari **re** va **im** nomlari bilan ataladi va ular **real** tipiga tegishlidir.

re va **im** bir xil tipli bo'lgani uchun ularni bitta ro'yhatga birlashtirib yozsa ham bo'ladi:

```
type
 ComplEx = record
     re, im: real;
 end;
```

Endi barcha kompleks (mavhum) qiymatlar qabul qiluvchi o'zgaruvchilarning tiplarini *var* bo'limida aniqlash mumkin:

```
var
 x, y, z: CompLex;
```

Bu tipdagi o'zgaruvchilarga biror qiymat o'zlashtirish uchun ularning maydonlarini tashkil etuvchilarga qiymat berish kerak bo'ladi. Masalan. x o'zgaruvchiga *4,5 + i * 6,75* qiymatini o'zlashtirish uchun, **re** va **im** ismli maydonlarga qiymat berish kerak:

```
x.re := 4.5;  x.im := 6.75;
```

Bir xil kombinatsiyali tipga tegishli o'zgaruvchilar uchun faqat o'zlashtirish amali gina o'rinli holos:

```
y := x;
```

Yozuvlarga doir quyidagi misol sirtida ishlash malakamizni oshiraylik. Misol: **x** va **u** mavhum sonlari sirtida qo'shish, ayirish va ko'paytirish amallarini bajarish dasturini tuzing.

Masalani yechish algoritmi:

Agar x = Re x + i Im x, y = Re y + i Im y bo'lsa ular sirtida sanab o'tilgan amallarni bajarish algoritmi quyidagicha bo'ladi:

```
u = x + y      Re u = Re x + Re y,  Im u = Im x + Im y;
v = x - y      Re v = Re x - Re y,  Im u = Im x - Im y;
w = x * y      Re w = Re x * Re y - Im x * Im y,
               Im w = Re y * Im x + Re x * Im y;
```

Endi mazkur algoritmni dasturda ifoda etamiz:

```
type
 comp = record
  re, im: real
 end;
var
 x, y, u, v, w: comp;
begin
 {x va u mavxum sonlarning haqiqiy(Re x, Re y) va mavhum (Im x, Im y) qismlarini
    kiritish}
 x.re := StrToFloat(InPutBox('' , 'x.re qiymatini:'. '0'));
 x.im := StrToFloat(InPutBox('' , 'x.im qiymatini:', '0'));
```

```
  y.re := StrToFloat(InPutBox('' , 'y.re qiymatini:', '0'));
  y.im := StrToFloat(InPutBox('' , 'y.im qiymatini:', '0'));
  {u = x + y}
  u.re := x.re + y.re;  u.im := x.im + y.im;
  {v = x - y}
  v.re := x.re - y.re;   v.im := x.im - y.im;
  {w = x * y}
  w.re := x.re * y.re - x.im * y.im;
  w.im := x.re * y.im +  x.im * y.re;

  ShowMessage('x+y =' + FloatToStr(u.re) + ' + ' + FloatToStr(u.im));
  ShowMessage('x-y =' + FloatToStr(v.re) + ' + ' + FloatToStr(v.im));
  ShowMessage('x*y =' + FloatToStr(w.re) + ' + ' + FloatToStr(w.im));
end;
```

Ma`lum bir guruh talabalari haqidagi ma`lumotlarni jamlashtira oladigan yozuv va o`zgaruvchi yarating.

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.

Fayllar. Ma'lumotlarni faylga yozish va o'qish
Faylli tiplar

Faylli tipdagi o'zgaruvchilarni diskdan ma'lumot o'qib oluvchi yoki diskka ma'lumot yozib qo'yuvchi dasturlarda ishlatish mumkin. Faylli tipdagi o'zgaruvchilarni e'lon qilishda ***File*** va ***TextFile*** xizmatchi so'zlari ishlatiladi:

```
var  mfile1, mfile2: file;  afile: file;
 Prima: TextFile;
```

TextFile xizmatchi so'zi faylning matnli ekanligini anglatadi. Matnli fayllar maxsus belgilar bilan ajratilgan, uzunligi noma'lum bo'lgan qatorlardan tashkil topadi.

Ayrim paytlarda fayllarni bir xil tipli hadlar ketma-ketligi ko'rinishida qarash qo'layroq bo'ladi. Bu ketma-ketlik qatorlar, butun sonlar yoki yozuvlardan tashkil topishi ham mumkin:

```
var
 A1: file of byte; {A1 fayli baytlar ketma-ketligidan tashkil topgan}

 A2: file of integer;{A2 fayli butun sonlar ketma-ketligidan tashkil topgan}

 A3: file of string; {A3 fayli qatorlar ketma-ketligidan tashkil topgan}

 A4: file of string[20]; { A4 fayli 20ta belgili qatorlarning ketma-
                          ketligidan tashkil topgan}
 A5: textFile;        {A5 fayli matnli fayl hisoblanadi}
```

Agar faylning hadlari uchun tip aniqlangan bo'lsa, bunday fayllarni tiplashtirilgan, aks holda tiplashtirilmagan deb ataladi:

```
var  A: file ; { tiplashtirilmagan fayl}
     B: file of char; { tiplashtirilgan fayl}
```

Fayllar bilan ishlaydigan quyidagi dasturni ko'rib chiqaylik:

```
Var  mydata: file of integer;
     i, j, sum: integer;  s: String;
begin
 AssignFile(mudata, 'd:\tp\myfile.dat'); {mydata fayl uzgaruvchisi bilan faylning
                                          ismini myfile.dat va uning aniq
                                          yuli aniqlanmoqda}
 rewrite(mydata);     {fayl yozish uchun ochiq}
 ShowMessage('Salom noma'lum urtoq...');
 S := InputBox ('Kiritish oynasi', 'Birinchi sonni kiriting', '');
 I := StrToInt(S);
 ShowMessage('Kiritilgan sonni diskdagi myfile.dat fayliga
 yozilmokda');
 write(mydata, i);                {bu operator yordamida diskdagi myfile.dat  fayliga I
                                  sonining qiymati yoziladi}
 S := InputBox('Kiritish oynasi', 'Ikkinchi sonni kiriting', '');
 J := StrToInt(S);
 ShowMessage('Kiritilgan ikkinchi sonni diskdagi myfile.dat fayliga yozilmoqda');
```

```
 write (mydata, j);        {Diskka yozish ajarilmoqda}
 sum := i + j;
 ShowMessage ('Yigindi ='+ IntToStr(sum));
 ShowMessage ('Yigindi diskdagi myfile.dat fayliga yozilmoqda');
 write(mydata, sum);              {Diskka yozish bajarilmoqda}
 closeFile(mydata);       {mydata fayli yopildi}
 ShowMessage ('Xayr noma'lum urtoq...');
End;
```

E'tiboringizga havola etilgan dasturda *Assign*File, *Rewrite*, *Write* va *Close*File proseduralaridan foydalanildi. Endi shu proseduralarning va keyingi dasturda ishlatiluvchi *Reset* va *Read* proseduralarning vazifalari va qanday aniqlanganligi haqida qisqacha ma'lumot berib o'taylik:

AssignFile prosedurasi.

Vazifasi: Faylli o'zgaruvchiga tashqi fayl ismini o'zlashtiradi.

Aniqlanishi: AssignFile(f; name: **string**);
bu yerda
f - ixtiyoriy tipli faylli o'zgaruvchi;
name - qatorli tipdagi ifoda yoki qator, fayl ismi (agar faylning to'liq yo'li ko'rsatilmagan bo'lsa fayl ishlanayotgan katalogda joylashgan bo'ladi).

CloseFile prosedurasi.

Vazifasi: ochiq faylni yopadi.

Aniqlanishi: CloseFile(f);
bu yerda f - oldindan ochilgan faylga mos keluvchi faylli o'zgaruvchi.

Read prosedurasi.

Vazifasi: Fayl hadini o'zgaruvchiga o'qiydi.

Aniqlanishi: Read(f, v);
bu yerda
f - faylning ixtiyoriy tipiga mos faylli o'zgaruvchi (faqat matnli, tipli emas);
v - fayl hadi tipi bilan bir xil tipli o'zgaruvchi.

Reset prosedurasi.

Vazifasi: mavjud faylni ochadi.

Aniqlanishi: Reset(f: file);
bu yerda f – faylning ixtiyoriy tipiga mos faylli o'zgaruvchi va u fayl bilan *Assign* prosedurasi orqali bog'langan bo'lishi kerak. *Reset* prosedurasi shu faylni ochadi.

Rewrite prosedurasi.

Vazifasi: yangi faylni yaratadi va ochadi.

Aniqlanishi: Rewrite(f: file);
bu yerda f – ixtiyoriy faylli tipdagi faylli o'zgaruvchi.

Rewrite prosedurasini ishlatishdan oldin f o'zgaruvchi *Assign* prosedurasi yordamida diskdagi fayl bilan bog'lanishi kerak. *Rewrite* prosedurasi yangi fayl tashkil qiladi.

Write prosedurasi.

Vazifasi: fayl hadiga o'zgaruvchini yozib qo'yadi.

Aniqlanishi: Write(f, v);

bu yerda f – faylli o'zgaruvchi;

v - f fayli hadi bilan bir xil tipli o'zgaruvchi.

Oldingi tuzgan dasturimiz «d:» diskdagi tp katalogida myfile.dat faylini tashkil qildi. Endi shu fayldan qanday qilib ma'lumotlarni o'qishni ko'rib chiqaylik:

```
Var  mydata: file of integer;
     i, j, sum: integer;
begin AssignFile(mydata, 'd:\tp\myfile.dat');
     reset(mydata);   {fayl uqish uchun ochilmoqda}
     ShowMessage('Salom noma'lum urtoq...');
     read(mydata, i);
     ShowMessage('myfile.dat faylidan birinchi son o'qildi');
     read(mydata, j);
     ShowMessage('diskdagi myfile.dat faylidan ikkinchi son o'qildi');
     read(mydata, sum);
     ShowMessage('myfile.dat faylidan uchinchi son o'qildi');
     closeFile(mydata);          {mydata.dat fayli yopiladi}
     ShowMessage('Xayr noma'lum urtoq...');
End;
```

TextFile standart faylli tip matnli fayllarni aniqlaydi. Matnli fayllar o'zaro yangi qatorga o'tish belgilari bilan ajratilgan qatorlardan tashkil topadi.

Matnli fayllar bilan ishlash uchun maxsus kiritish (*Readln*) chop etish (*Writeln*) proseduralari kiritilgan. Bu proseduralar uzunligi noma'lum qatorlarni fayllardan o'qish va fayllarga yozish uchun ishlatiladi.

Endi matnli fayllar bilan ishlashga doir quyidagi dastur bilan tanishib chiqaylik:

```
var mytext: TextFile; s: string;
begin
 AssignFile(mytext, 'd:g'tpg'mytext.txt'); {mytext faylli uzgaruvchi orqali fayl ismi va
                                            yuli aniqlanmoqda}
 rewrite(mytext);  {fayl yozish uchun ochiq}
 s := InpurBox ('Kiritish', 'Sizning ismingiz?', s);
 ShowMessage('Ismingizni diskdagi mytext.txt fayliga yozilmoqda');
 writeln(mytext, s);       {s -qatori mytext.txt fayliga yozilmoqda}
 closeFile( mytext);       {mytext fayli yopildi}
end;
```

Yozuvlardan foydalanib fayl tashkil qiling va uni tahrirlovchi dastur tuzing.

Ob'yektli dasturlash tiliga kirish

Ob'yektli dasturlash – bu tarkibida ob'yekt tushunchasi bo'lgan dasturlarni qayta ishlash uslubidir. Qo'yilgan masalalar ob'yektlar yordamida yechilsa, ular ob'yektli dasturlar deb ataladi. Undagi asosiy dastur ob'yektlar ketma-ketligini o'zida saqlaydi va ularni bir-biri bilan bog'laydi. Ob'yekt o'z navbatida o'ta keng tushunchaga ega bo'lib, u quyidagi uchta o'zak tushunchalarga asoslangan holda tashkil etiladi: inkapsulyatsiya, naslyedovaniya (meros qoldirish) va polimorfizm.

Sinf

Object Pascal dasturlash tili dasturchilar uchun murakkab tipli yozuv(record)larni e'lon qilish imkoniyatini tug'dirgan bo'lsa, Object Pascal dasturlash tili, sinflardan foydalanish imkoniyatini yaratadi. **Sinf** – bu, murakkab ko'rinishga ega bo'lib, bir joyga jamlangan ma'lumot yozuvlarini, prosedura va funksiyalarni o'zida mujassamlashtiradi.

Quyidagi misolda oddiy sinfni e'lon qilish yo'lini ko'rib chiqaylik:

```
TTPerson = 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 **TTPerson** – sinf nomi, **FName** va **FAddress** – maydon nomi, **Show** — uslub nomi.

Eslatma: Maydon nomlarining birinchi belgisini F harfi bilan boshlash Delphi dasturlash tilida kelishib olingan (Field – maydon deyiladi).

Uslubni e'lon qilish dasturning **Type** bo'limi ichiga yoziladi.

Ob'yekt

Ob'yektlar, sinflar singari Var bo'limida e'lon qilinadi. Masalan:

```
Var
    Student: TTPerson;
    Professor: TTPerson;
```

Object Pascal dasturlash tilida **ob'yekt** – bu dinamik struktura. Ob'yekt o'zgaruvchisi ma'lumotni o'zida saqlamaydi, balki ob'yekt ma'lumotiga yo'lni o'zida saqlaydi. Shuning uchun dasturchi belgilangan xotira (tezkor xotira hajmi) haqida o'ylashi kerak bo'ladi.

Belgilangan xotira ***konstruktor*** deb ataluvchi sinfning alohida uslubi yordamida yaratiladi. Bu ***Create*** (yaratish) nomi bilan bajariladi.

Konstruktor e'loni uchun *Procedure* so'zining o'rniga ***Constructor*** so'zi ishlatiladi.

Quyidagi misolda **TTPerson** sinfi ichiga konstruktor ham kiritilgan:

```
TTPerson = class
      Private
        Fname: String[15];
        Faddress: String[35];
    Constructor Create;      // konstruktor
      Public
        Procedure Show;      // uslub
End;
```

Agarda dastur ishlashi davomida qandaydir, boshqa ishlatilmaydigan va ob'yektda joy egallab turgan maydon bo'lsa uni xotiradan tozalab tashlash mumkin. Bu amalni bajarish uchun ***Free*** uslubidan foydalaniladi. Misol uchun, Professor nomli ob'yekt maydonini xotiradan tozalash uchun quyidagini yozish kifoya: Professor.free;

Uslub

Sinf ***uslub***lari (sinf ichida e'lon qilingan prosedura va funksiyalar) sinf ob'yektlari sirtida turli amallar bajaradi. Uslubni ishlatish uchun, metod nomini va uslub nomini ko'rsatish kerak bo'ladi. Birinchi nom ikkinchisidan nuqta bilan ajratiladi. Masalan:

```
Professor.show;
```

Professor ob'yektidan **Show** metod chaqirilmoqda. Yana bir misol sifatida. TTPerson sinfiga qarashli Show uslubining yozilishini ko'rishimiz mumkin.

```
// TTPerson sinfidagi  Show uslubi
Procedure TTPerson.Show;
Begin
   ShowMessage('Nomi:' + fname + #13 + 'Adryes:' + faddress);
End;
```

Inkapsulyatsiya va ob'yekt xususiyati

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

Object Pascal dasturlash tilida maydonlarga ruhsatni cheklash uchun ob'yekt xususiyatidan foydalaniladi. Xususiyat maydoniga ruhsat berishni ta'minlovchi ikki uslub mavjuddir. Xususiyat qiymatni o'rnatuvchi uslub ***xususiyatni yozish*** (**write**) uslubi deb nomlanadi. Xususiyat qiymatini o'quvchi uslub esa ***xususiyatni o'qish*** (**read**) uslubi deyiladi.

Sinf ichida xususiyatni yozishdan avval ***Property*** (xususiyat) so'zi yoziladi. Xususiyat nomidan so'ng u(xususiyat)ning tipi ko'satiladi va xususiyat qymatiga ruhsat berishni ta'minlovchi uslub yoki uslublarni yozish mumkin. Read so'zidan so'ng xususiyatni o'qishni ta'minlovchi uslub nomi ko'rsatiladi. Write so'zidan keyin esa xususiyatni yozishni ta'minlovich uslub nomi ko'rsatiladi.

Quyidagi misolda ikki **Name** va **Addres** xususiyatlarini o'z ichiga olgan **TTPerson** sinfi keltirilgan:

```
Type
 TName = String[15];
TAddress = String[35];
TTPerson = class
Private
FName:TName;    //Name xususiyatining uzgaruvchisi FAddress:TAddress; //Address xususiyatining uzgaruvchisi Constructor Creat(Name:TName);
    Procedure Show;
    Function GetName:TName;
    Function GetAddress:TAddress;
    Procedure SetAddress(NewAddress:TAddress);
Public
    Property Name:TName
Read GetName;
    Property Address:TAddres
Read GetAddress
    Write SetAddress;
End;
```

Dastur ichida xususiyatga qiymat berish oddiy yo'l bilan amalga oshiriladi. Masalan, Student ob'yektidagi Address xususiyatining qiymatni tahrirlash quyidagicha amalga oshirilishi mumkin:

```
Student.Address := 'Norin sh., Yangi tol k., 19-uy';
```

Kompilyator esa uni quyidagi ko'rinishda translyatsiya qiladi:

```
Student.SetAddress('Norin sh., Yangi tol k., 19-uy');
```

Xususiyat qiymatini faqat o'qish uchun ruhsat berilgan bo'lsa, lekin unga qiymat berilsa dasturni kompilyatsiya qilish vaqtida xatolik chiqadi. Masalan, yuqorida keltirilgan misolda Name xususiyati qiymatini o'qish mumkin holos, Address xususiyati qiymatini o'qish va yozish mumkin.

Quyida TTPerson sinfini yaratish va undan foydalanishga ruhsat berishni ta'minlovchi misol keltirilgan:

```
// TTPerson ob'yektining konstruktori
Constructor TTPerson.Create(Name:TName);
Begin
    Fname := Name;
End;
// Name xususiyatidan qiymat olish uslubi
Function TTPerson.GetName;
Begin
    Result := FName;
End;
```

```
// Address xususiyatidan qiymat olish uslubi
Function TTPerson.GetAddress;
Begin
    Result := FAddress;
End;

// Address xususiyatining qiymatini uzgartirish uslubi
Procedure TTPerson.SetAddress(NewAddress:TAddress);
Begin
    If Faddress = '' Then FAddress := NewAddress;
End;
```

Dastur tuzishda TTPerson sinfidan foydalanish quyidagi ko'rinishga ega bo'lishi mumkin (Student:TTPerson):

```
Student := TTPerson.Create('Kamoldin');
Student := TTPerson.Address('Norin sh., YAngi tol k., 19-uy');
```

Avlod qoldirish (Meros qoldirish)

Ob'yektga mo'ljallangan dasturlashda, ob'yekt o'zidan avlod qoldirishi mumkin. Qoldirigan avlodga ajdod ob'yektdagi barcha xususiyat, uslub va maydonlar mye'ros bo'lib o'tadi. Avlod qoldirishning biz uchun foydali tarafi shuki, ajdodning imkoniyatiga qo'shimcha ravishda unga yangi xususiyat va uslublarni qo'shishimiz mumkin.

Misol uchun yuqorida ko'rib o'tilgan TTPerson sinfidan **TEmpl** nomli yangi sinf yaratib ungan **Department** (bo'lim) nomli maydonni qo'shish quyidagicha amalga oshiriladi:

```
TEmpl = class(TTPerson)
    FDepatment: Integer;    // Bulim rakami
    Constructor Create(Name: TName; Dep: Integer);
End;
```

Yuqoridagi misoldan kelib chiqib, TEmpl sinfi o'zining shaxsiy konstruktorini yaratishi mumkin. U quyidagi ko'rinishga ega bo'ladi:

```
Constructor TEmpl.Create(Name: TName; Dep: Integer);
Begin
    Inherited Create(Name);
    FDepartment: qDep;
End;
```

Yuqoridagi misolda ***Inherited*** direktivasi yordamida ajdod sinfining konstruktori chaqirilib, so'ngra avlodning yangi maydoni qo'shib qo'yilmoqda.

Dastur tuzishda TTPerson sinfidan qoldirilgan avlod TEmpl sinfidan foydalanish quyidagi ko'rinishga ega bo'lishi mumkin (Modif: TTPerson):

```
Modif := TEmpl.Create('Akramjon',112);
Modif.Address := 'Norin sh., YAngi tol k., 19-uy';
```

Protected va Private direktivalari

Sinf elementlarini (maydon, usul, xususiyat) e'lon qilishda, dastur ichidagi ko'rinishini ta'minlovchi Protected (himoyalangan) va Private (berkitilgan) direktivalaridan foydalaniladi.

Protected seksiyasidagi e'lon qilingan sinf elementlaridan foydalanish faqat uning avlodiga ruhsat etiladi. Dastur tuzish jarayonida bu seksiyadagi sinf elementlari ro'yhatini ko'rish cheklanmagan. Oddiy qilib aytiladigan bo'lsa bu seksiyaga sinfning usullari yoziladi.

Private seksiyasida e'lon qilingan sinf elementlari faqatgina yaratilgan modul ichida ko'rinishi mumkin. Oddiy qilib aytiladigan bo'lsa bu seksiyaga sinfning maydonlari yoziladi.

```
TTPerson = class
    Private
    FName: TName;  // Name xususiyatining uzgaruvchisi
    FAddress: TAddress; //Address xususiyatining uzgaruvchisi
    Protected
    Constructor Creat(Name: TName);
    Procedure Show;
    Function GetName: TName;
    Function GetAddress: TAddress;
    Procedure SetAddress(NewAddress: TAddress);
    Property Name: TName
    Read GetName;
    Property Address: TAddres
    Read GetAddress
    Write SetAddress;
End;
```

Polimorfizm va virtual uslub

Polimorfizm – bu har xil sinflardan chiquvchi bir xil nomli uslublar imkoniyatidan foydalanishdir. Ya'ni polimorfizm yordamida ajdoddagi uslub nomini va uning imkoniyatini saqlab qolgan holda, avlodda ham ana shu nomli lekin boshqacha imkoniyatli uslub yaratish mumkin.

Quyidagi misolda uchta sinf e'lon qilingan bo'lib, ulardan biri asosiy hisoblanadi:

```
type
    // asosiy sinf
    TPerson = class
        Fname: string; // nom
    Constructor Create(Name: String);
    Function info: string; virtual;
    End;
// TPerson dan koldirilgan avlod
TStud = class(TPerson)
```

```
    FGr: Integer; {gurux rakami}
  Constructor Create(Name: String; gr: Integer);
    Function info: string; override;
End;

// TPerson dan koldirilgan avlod
TProf = class(TPerson)
    FDep: String;  // Kafedra nomi
  Constructor Create(Name: String; Dep: String);
    Function info: string; override;
End;
```

Keltirilgan barcha sinflarda **Info** uslubi aniqlanmoqda. Asosiy sinfda **virtual** direktivasi yordamida **Info** uslubi **virtual** deb e'lon qilindi. Usullarni **virtual** holda e'lon qilish. qoldirilgan avlodda uni o'zining shaxsiy uslubini saqlab qolishdir. Ikkala avlod tarkibiga ham ajdod uslubi nomi bilan bir xil **Info** uslubi kiritilgan (qoldirilgan avlodda **virtual** sinfni o'zlashtirish **override** direktivasi yordamida amalga oshriladi).

Quyidagi misolda barcha sinflar uchun Info uslubini kiritish keltirilgan:

```
Function TPerson.info: string;
Begin
    Result := '';
End;
Function TStud.info: string;
Begin
    Result := fname + '  gr.' + IntToStr(fgr);
End;
Function TProf.info: string;
Begin
    Result := fname + '  kaf.' + fdep;
End;
```

Dastur tuzishda talabalar ro'yhatini chiqarish uchun massiv e'lon qilib TPerson sinfidan quyidagicha foydalanish mumkin:

```
List: array [1..Sizel] of TPerson;
```

Massivdagi studentlar ro'yhatini chiqarish qo'yidagicha bulishi mumkin:

```
St := '';
For I := 1 to Sizel do      //Sizel – massiv chegarasi
  If List[I] <> Nil
    Then st := st + list[I].info +  #13;
ShowMessage(St);
```

Navbatdagi dasturda yuqorida e'lon qilingan TPerson, TStud va TProf sinflaridan qanday foydalanish keltirilgan.
Uning oyna ko'rinishi quyidagicha:

Dasturi quyidagicha kiritiladi:

```
unit Stud_P;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  StdCtrls,
ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    RadioGroup1: TRadioGroup;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type
  // asosiy sinf
  TPerson = class
      Fname: string; // nom
      Constructor Create(Name: String);
      Function info: string; virtual;
  End;

  // TPerson dan koldirilgan avlod
  TStud = class(TPerson)
      FGr: Integer; {gurux rakami}
      Constructor Create(Name: String; gr: Integer);
```

```
    Function info:string; override;
  End;

  // TPerson dan qoldirilgan avlod
  TProf = class(TPerson)
    FDep: String; // Kafedra nomi
    Constructor Create(Name: String; Dep: String);
    Function info: string; override;
  End;
const
  Size1 = 10;          // ruy xat xajmi
var
  Form1: TForm1;
  List: array [1..Size1] of TPerson; // Ruyxat
  n: integer;
implementation
{$R *.DFM}
constructor TPerson.Create(Name: String);
begin
    fName := Name;
end;
constructor TStud.Create(Name: String; gr: Integer);
begin  inherited Create(name);
    fgr := gr;
end;
constructor TProf.Create(Name: String; dep: String);
begin  inherited Create(name);
    fdep := dep;
end;

Function TPerson.info: string;
Begin
  Result := '';
End;
Function TStud.info: string;
Begin  Result := fname + ' gr.' + IntToStr( fgr);
End;
Function TProf.info: string;
Begin  Result := fname + ' kaf.' + fdep;
End;
// ruyxat ichiga yangi element kushish
procedure TForm1.Button1Click(Sender: TObject);
begin  if n <= Size1 then
  begin   if RadioGroup1 .ItemIndex = 0 then  // Stud ob'yektini yaratamiz
     List[n] := TStud.Create(Edit1.Text, StrToInt(Edit2.Text))
```

```
  else
    List[n] := TProf.Create(Edit1.Text, Edit2.Text);
     N := n + 1;
 end else ShowMessage('Ruyxat tulgan !');
end;
procedure TForm1.Button2Click(Sender: TObject);
var i: Integer; st: String;
begin
  for i := 1 to Sizel do
    if list[i] <> Nil Then st := st + list[i].info + #13;
  ShowMessage(St);
end;
end.
```

Kiritish (Button1) tugmasi bosilganida ya'ni **TForm1.Button1Click** prosedurasi ishga tushganida TStud yoki TProf sinfidan List[n] ob'yekti yaratiladi. Kiritilgan yozuv yaratilayotgan ob'yekt sinfining qaysinisiga yozilishini **RidioGroup1** holatiga qarab aniqlanadi.

Ro'yhat (Button2) tugmasi bosilganida ya'ni **TForm1.Button2Click** prosedurasi ishga tushganida talaba va o'qituvchilar ro'yhati oynaga chiqariladi.

Sinf va uslubning farqini tushuntiring.

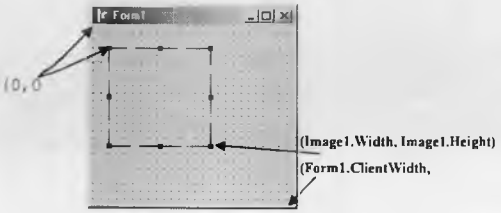
Delphining grafik imkoniyatlari

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. Delphining 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.

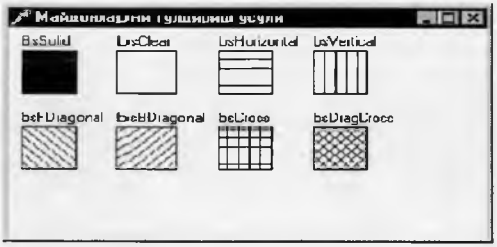
Color xususiyati sifatida Tcolor ning barcha o'zgarmaslaridan foydalanish mumkin. Style xususiyatlari quyidagi jadvalda keltirilgan:

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 (-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.

[1] Dyuym taxminan 2,5 sm ga teng.

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

```
with Form1.Canvas do
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.

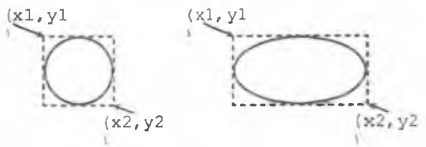
Aylana va ellips.

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).



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

Yoy

Yoy hosil qilish uchun **Arc** uslubidan foydalaniladi. Uning yozilish formati quyidagicha: Ob'yekt.Canvas.Arc(x1, y1, x2, y2, x3, y3, x4, y4);

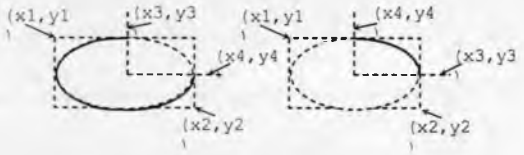
bu yerda ob'yekt – yoy chiziluvchi ob'yekt nomi;

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.

To'g'ri to'rtburchak

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

Ob'yekt.Canvas.Rectangle(x 1, y1, x2, y2)

Bu yerda ob'yekt – tasvir hosil bo'luvchi ob'yekt nomi;

x1, y 1, x2, y2 – to'g'ri to'rtburchakning mos ravishda yuqori chap va quyi o'ng burchak koordinatalari.

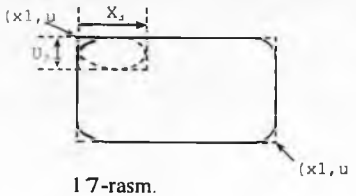
RoundRec uslubi ham to'g'ri to'rtburchak chizadi, faqat **Rectangle** dan farqi shundaki, uning burchaklari yumaloq (silliq) shaklda bo'ladi. Yozilish formati: Ob'yekt.Canvas.RoundRec(x1, y1, x2, y2)

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);

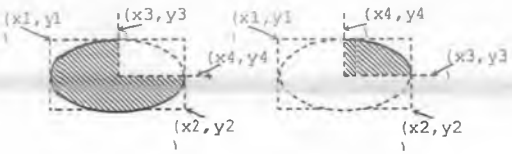
bu yerda

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

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 (-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 -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 yuklkilamiz
 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.

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.

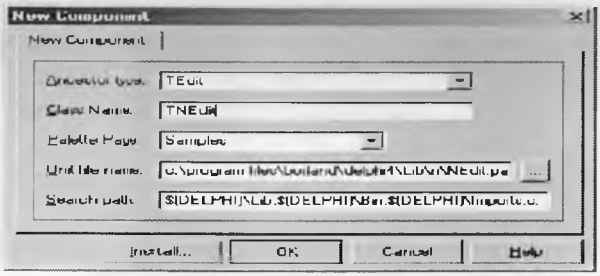
Quyida komponent yaratishni **NEdit** misolida ko'rib chiqamiz. **NEdit** komponentidan kasr sonlarni kiritish va tahrirlash uchun foydalaniladi.

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

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.



22-rasm.New Component mulokot oynasi

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** proseduras 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 proseduras **NEdit** komponentining tarkibida bo'lib, klaviaturadan kiritilgan belgilarni tekshiradi. Agarda ruhsat etilgan belgilar kiritilgan bo'lsa **OnKeyPress** xususiyatiga qayta ishlash uchun uzatiladi.

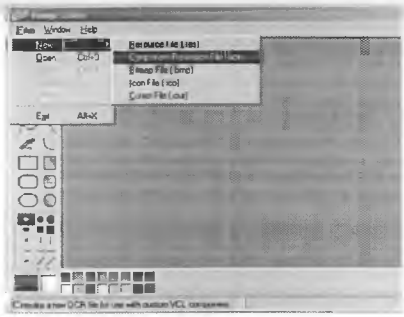
Komponentni o'rnatish

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

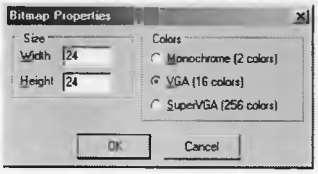
Komponentning resurs faylini yaratish uchun **Image Editor** utilitidan foydalaniladi. U **Tools** menyusidagi **Image Editor** buyrug'i yordamida ishga yuklanadi. Yangi resurs faylini yaratish uchun **File** menyusidan **New** buyrug'i tanlanadi, hosil bo'lgan ro'yhatdan yaratiluvchi fayl tipi tanlanadi. 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.

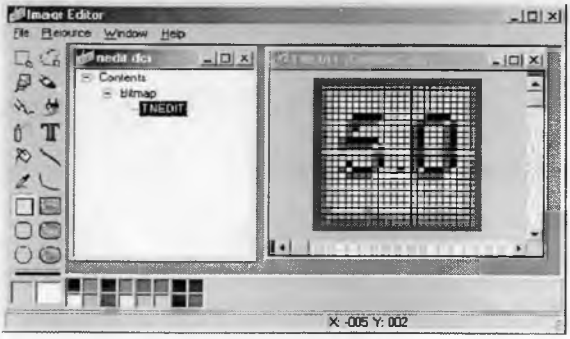
Shundan so'ng **Bitmap Properties** (24-rasm) muloqot oynasi ochiladi. U yerda komponent belgisi bitli tasvirining harakteristikasi o'rnatilishi kerak: **Size** (O'lchami) – 24x24 piksyelda, **Colors** (Ranglar soni) – 16 va **Ok** tugmasi bosiladi.



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.

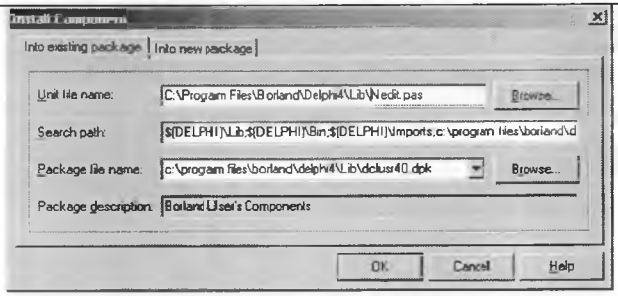


25-rasm. Image Editor muloqot oynasi

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.

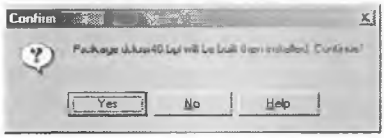
Search path (Izlanuvchi yo'l) maydoni komponentni o'rnatish vaqtida kerakli fayllarni izlashi mumkin bo'lgan kataloglar ro'yhatini o'zida joylashtiradi va ular nuqtali vergul bilan ajratib yoziladi.

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.

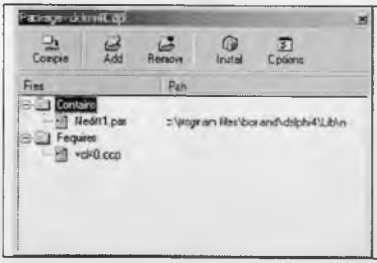


27-rasm. Yangi paket xosil bo'layotganligini ogohlantiruvchi muloqot oynasi.

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

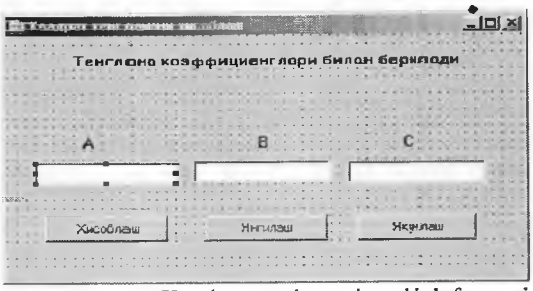
Yangi komponentni o'rnatish (qayta o'rnatish) vaqtida turli xatoliklar uchrashi mumkin.

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

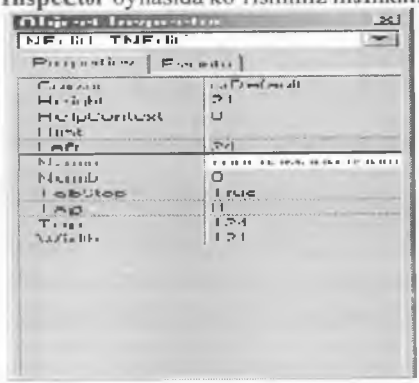
Paketlar to'plamiga komponent qo'shilganidan so'ng, uning belgisi komponentlar palitrasidan tekshiriladi. Agarda u yerda komponent belgisi bo'lsa demak uni yangi komponent sifatida ishlatish mumkin bo'ladi.

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 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, Label 1)
   Else ShowMessage('Barcha kotffisiyentlarni 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.

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 ketmaketlikdagi 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.

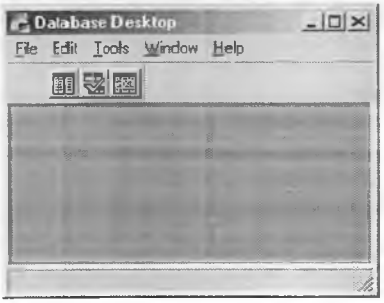
Ma'lumotlar ombori(ma'lumotlar fayli)ning nomi va undagi maydon nomlarini lotin harflarida yozish maqsadga muvofiqdir.

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:

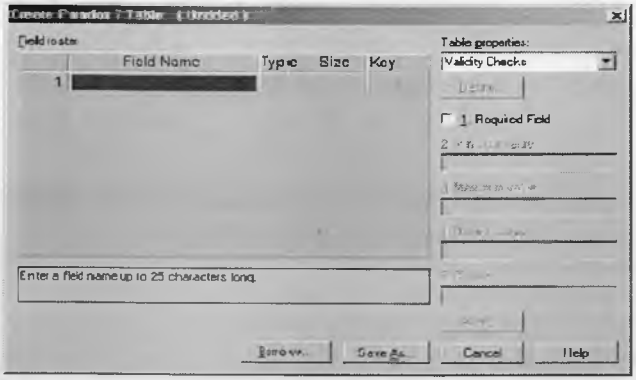
- Delphining yuqori menyusidan **Tools/ Database Desktop** buyrug`i tanlanadi;
- "Программы / Borland Delphi 4 / Database Desktop" buyrug`i tanlanadi.

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 sharifi;

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.

MS	**DBEdit** komponentini **Teble** komponentining maydonlaridan biriga bog`lang.

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.

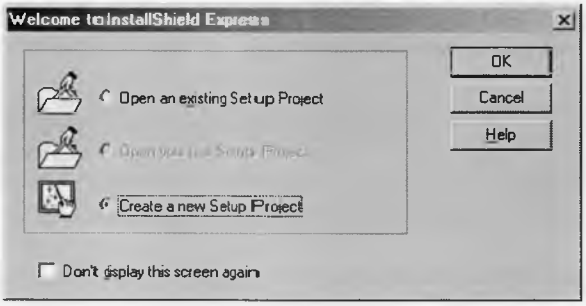
InstallShield Express dasturi

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).

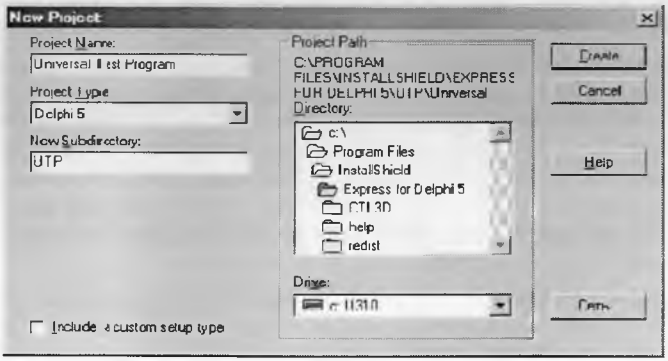


32-rasm. Dastur ishga tushirilganidan keyingi InstallShield Express muloqot oynasi.

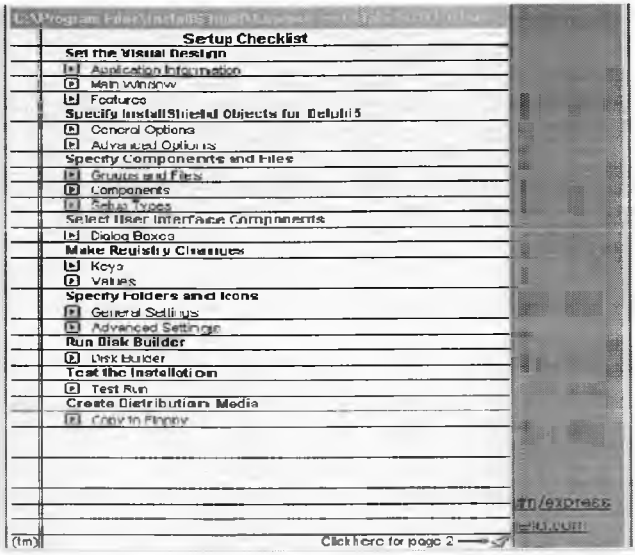
Hosil bo'lgan New Project (33-rasm) muloqot oynasiga proyekt haqidagi ma'lumotlar kiritiladi:

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



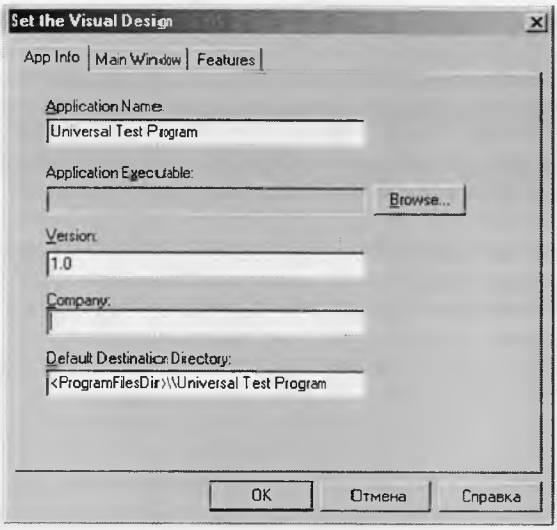
34-rasm. Installyatsion dasturni yaratuvchi proyekt oynasi.

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 sirtida 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

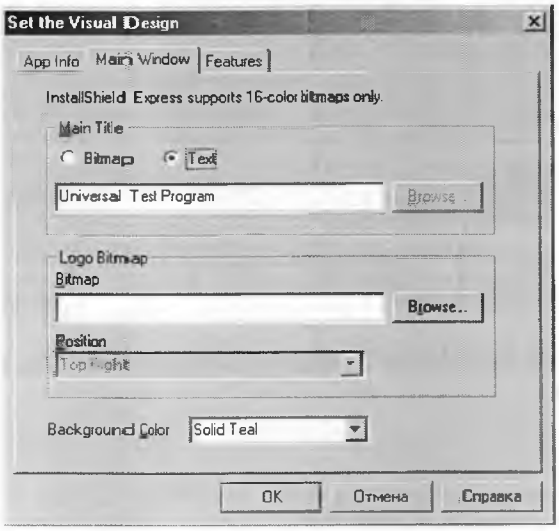
Sahifaning ba'zi maydonlari avtomatik to'ldirilganligiga e'tibor qilish lozim. InstallShield Express dasturi ilova nomi (Application Name) va ilova katalogi nomi (Default Destination Directory) sifatida proyekt nomidan foydalanishni taklif qiladi (dasturchi boshqa nom berishi ham mumkin).

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.



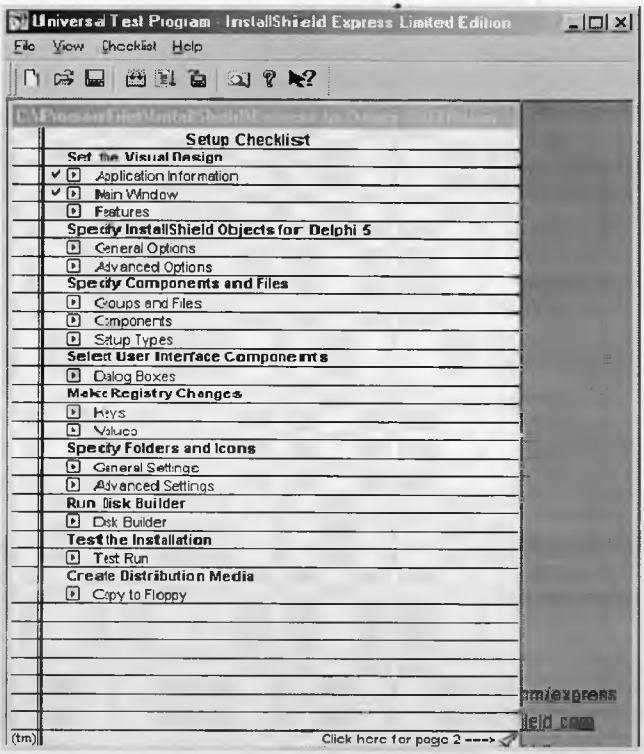
36-rasm. Main Window sahifasi.

Main Title guruhi o'rnatuvchi dastur bosh oynasining sarlavhasi tipini berish imkonini beradi. Agar Text tanlangan bo'lsa sarlavha 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).

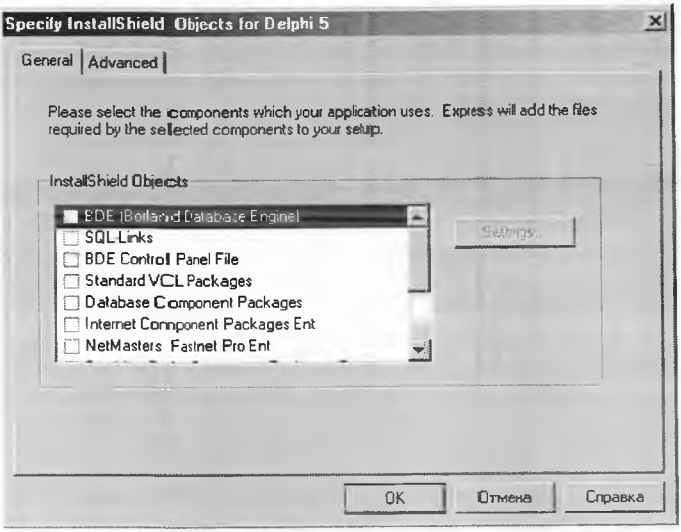


37-rasm.

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).



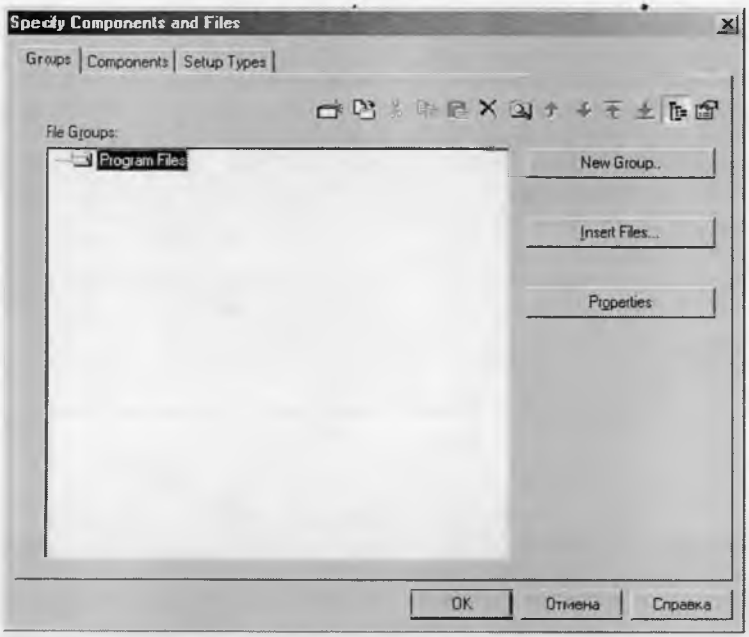
38-rasm. General sahifasi.

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.



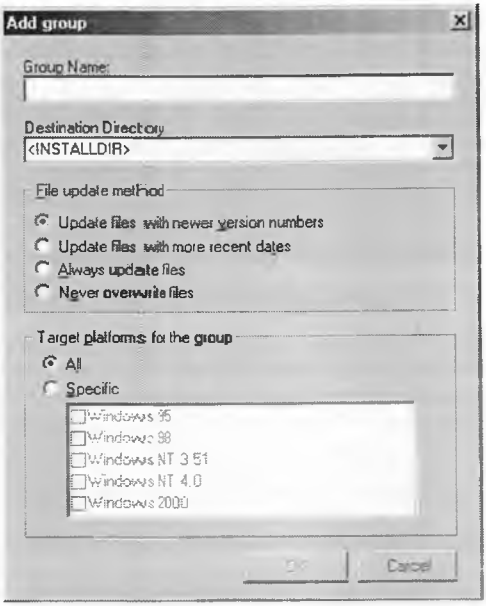
39-rasm. Specify Components and Files muloqot oynasi

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.

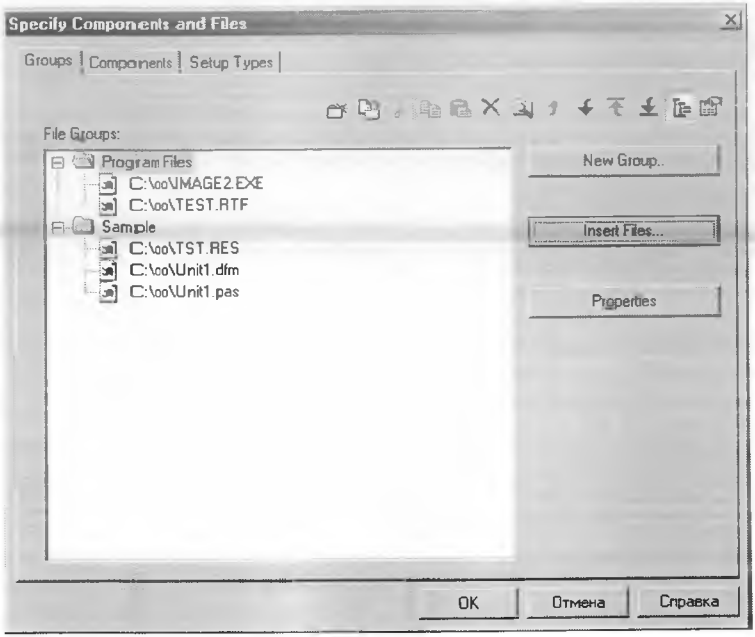
Dasturchi hohishiga ko'ra ba'zi fayllarni alohida katalokka ajratish mumkin. U holda bu fayllar uchun yangi guruh tashkil qilishga to'g'ri keladi. Yangi guruhni tashkil qilish uchun New Group tugmasidan foydalaniladi. Natijada Add group (40-rasm) muloqot oynasi hosil bo'ladi. Group Name maydoniga guruh 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.

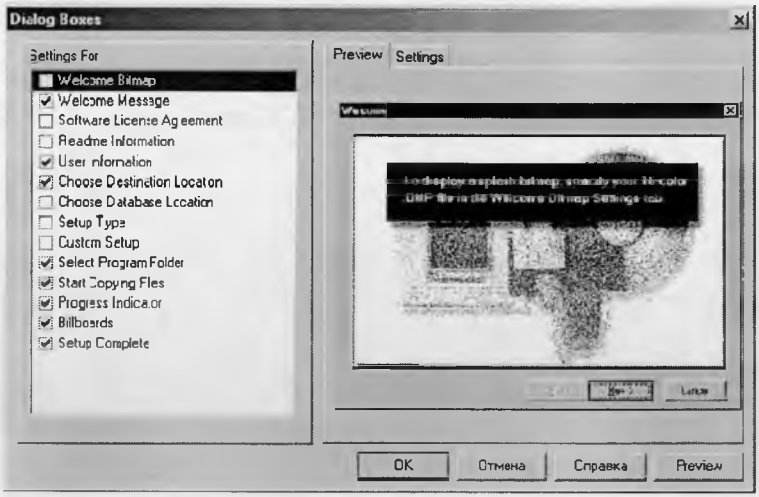


41-rasm. Foydalanuvchi kompyuteriga ko'chiriluvchi fayllar.

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.). Hosil bo'lgan muloqot oynadagi Settings For ro'yxatda installyatsion dastur ishi davomida hosil bo'luvchi muloqot oyna nomlari keltirilgan (2-jadval).



42-rasm. Dialog Boxes muloqot oynasi.

2-jadval

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

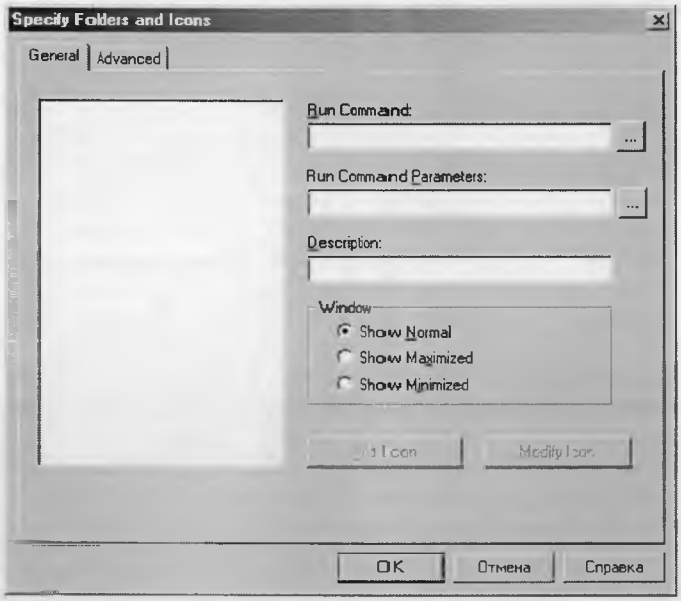
Specify Folders and Icons guruh buyruqlari o'rnatiluvchi dasturning ishga tushiruvchi buyrug'I joylashuvchi menyuni berish, shuningdek isjga tushirish buyrug'I va belgisini berish imkoniyatini yaratadi.

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 sahifasi

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:

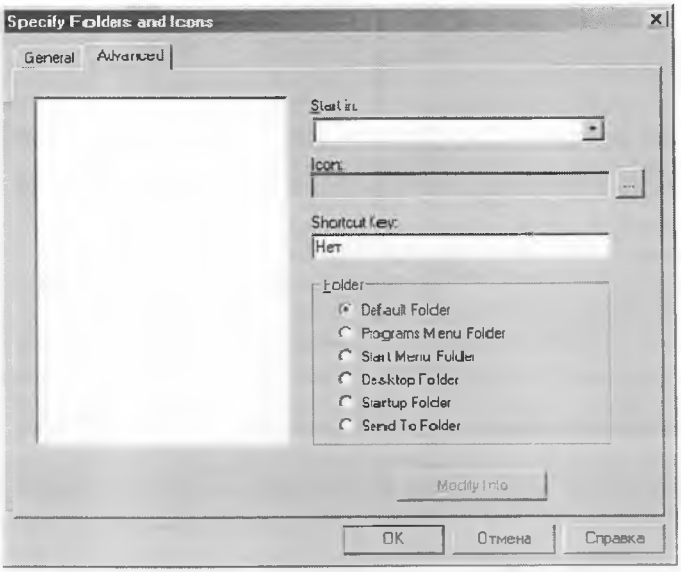
o'rnatuvchi disket nushasini yaratish;

o'rnatish jarayonini tekshirish;

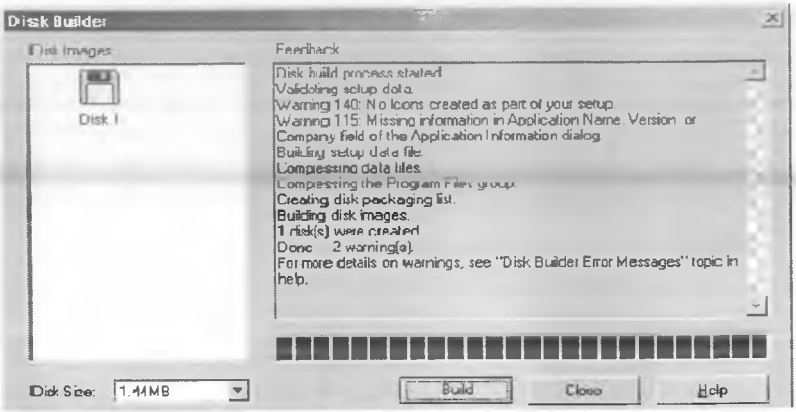
o'rnatuvchi disket nushasini disketga yozish.

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.

Foydalanilgan adabiyotlar

1. Н.Культин // Программирование на Object Pascal в Delphi, Киев. 1999г.
2. S.Irisqulov, M.Ataxanov // Algoritmik tillar va dasturlash, Namangan. 2000y.
3. А.Хоменко // Самоучитель Delphi 5, Киев. 1999г.
4. А.Епонешников, В.Епонешников, В.Епонешников // Программирование в среде Turbo Pascal 7.0, Москва. 1993г.
5. А.Дарахвалидзе, К.Марков // Delphi 4 МАСТЕР, Москва. 2000г.
6. А.Епонешников, В.Епонешников // Delphi 5. Язык Object Pascal.- М.:Диалог-МИФИ, 2000г.
7. Н.Культин // Программирование в Turbo Pascal 7.0 и Delphi, 2-е издание, С.Петербург, БХВ-С.Петербург, 1999г.

Web saytlar

1. www.bhv.ru
2. www.info@citmgu.ru
3. www.rusdos.ru
4. www.mda.hotmail.ru

Mundarija

Ma'sul muharrir: t.f.d., prof. B.Sh. Radjabov

Bosishga ruhsat etildi_____
Bichim 60x84 1/16. Adadi _____
Buyurtma. Bosma tabog'i _____

Toshkent axborot texnologiyalari universitetining tasarrufidagi
"ALOQAChI" nashriyot-matbaa markazida chop etildi.
Toshkent sh., Amir Temur ko'chasi, 108-uy.