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

NAMANGAN MUHANDISLIK –
TEXNOLOGIYA INSTITUTI

«KIMIYO-TEXNOLOGIYA» FAKULTETI

«Oliy matematika»kafedrasi

«Informatika va AT» fanidan

KURS ISHI

7u-14 guruh talabasi: ***O'.Nishanov***

Ilmiy rahbar: ***F.Uzoqov***

Namangan - 2015

Mavzu: “Vizual dasturlash tillari”

REJA:

I. Kirish

II. Asosiy qism.

II. 1. Dasturlarni qayta ishlash bosqichlari.

II. 2. Delphi muhiti. Delphi da boshlang’ich amallar va proyektlar

II. 3. Delphida komponentlar. Xususiyat va xodisalar.

II. 4. Operator(buyruq)lar

III. **Хулоса**

IV. Foydalinilgan adabiyotlar

I. 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 uni Turbo Pascal dasturlash tili deb nomlangan. Turbo Pascalni qayta ishlash natijasida ob'yektli dasturlash yo'lga qo'yildi va uni 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 ishlagan. 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 kitobda 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.

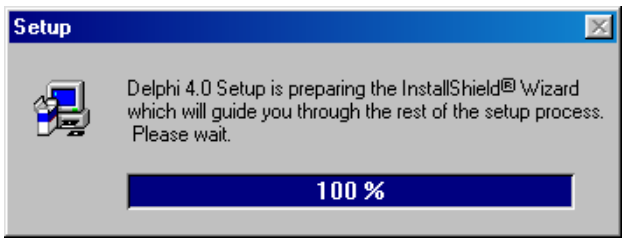
Delphi - vizual proyektlar, turli holat proseduralarini va dasturlarni qayta ishlashda vaqtdan yutish va boshqalarni o'z ichiga oladi.

Delphini o`rnatish va ishlatish

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

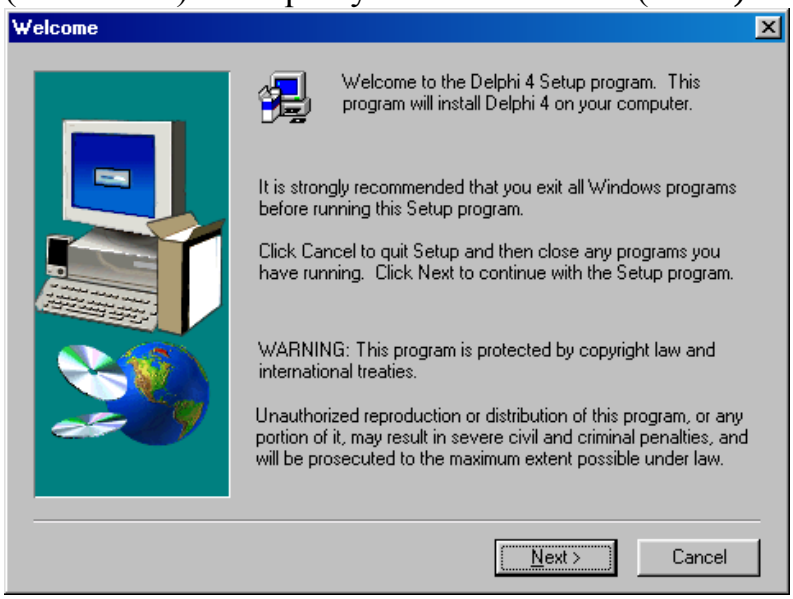
Eslatma: Delphini o'rnatish vaqtida barcha aktiv dasturlar ishini yakunlash kerak.

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



-rasm. Delphini o’rnatish jarayonini boshlash.

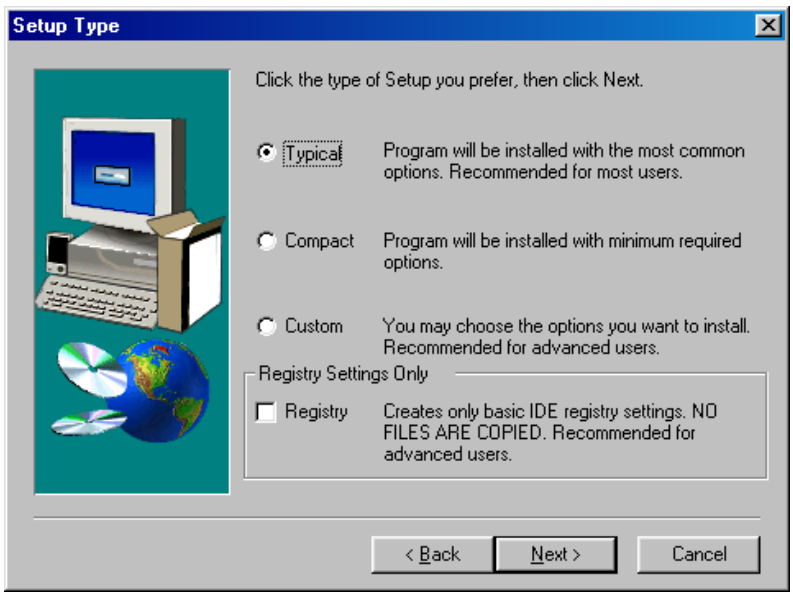
Bu oynada Delphini o’rnatish uchun tayyorgarlikni bajarilishini foiz hisobida ko’rinishi chiqadi. Tayyorgarlik oxiriga yetganidan so’ng oynada **Welcome** (boshlanish) muloqot oynasi xosil bo’ladi (-rasm).



-rasm. Welcome dialogli oyna.

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

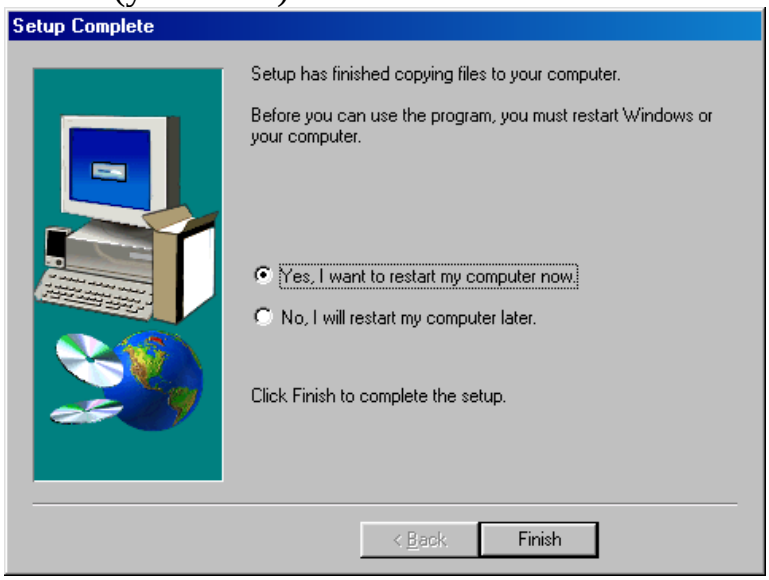
O’rnatish jarayonida quyidagi oyna hosil bo’ladi (-rasm):



-rasm. Setup Type dialogli oyna.

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

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



-rasm. Setup Complete dialogli oyna.

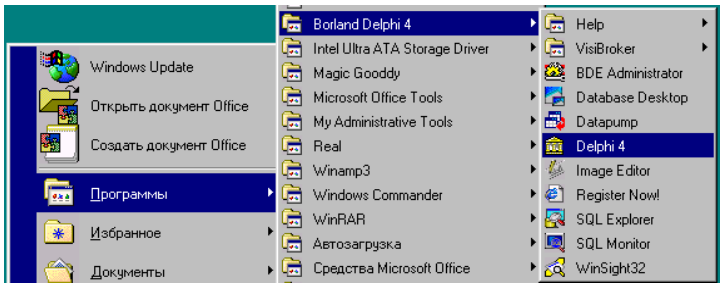
Shu bilan Delphini o'rnatish yakunlanadi va Delphi ishchi holatga tayyor holga keladi.

Delphini ishga tushirish

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

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

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



-rasm. Windows ning bosh menyusidan Delphini yuklash.

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

1. Dasturlarni qayta ishlash bosqichlari.

Dasturni qayta ishlash bosqichlari.

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

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

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

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

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

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

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

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

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

Algoritmlar va dasturlar

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

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

Qo’yilgan masalani yechish uchun algoritmdan foydalaniladi.

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

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

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

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

Natijaviylik – algoritm bajarilishi jarayonida aniq natija olinishi kerak.

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

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

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

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

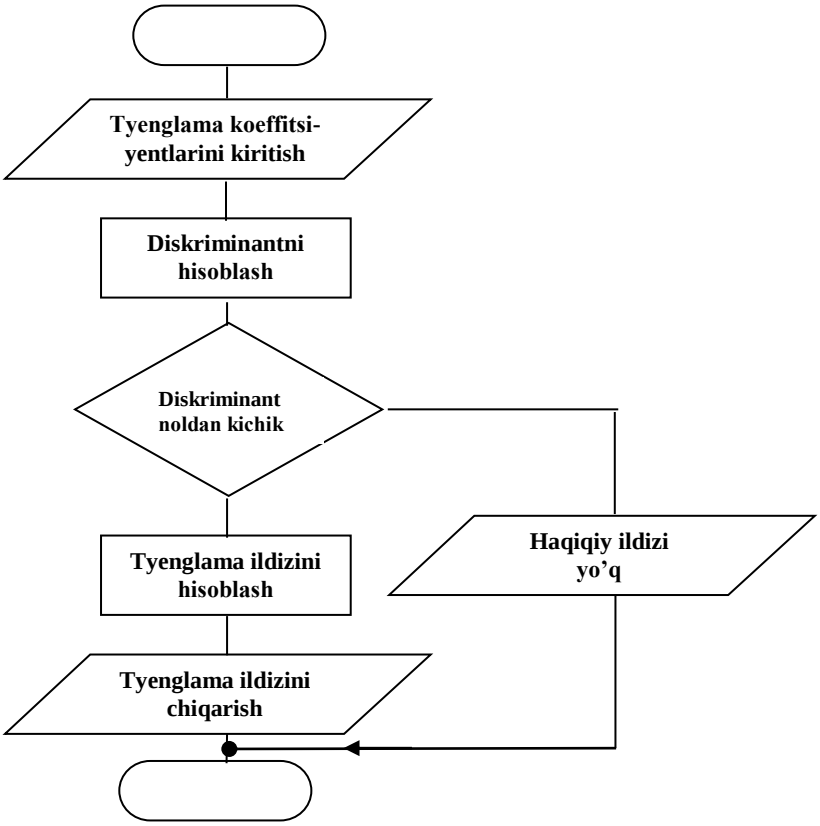
Agar diskriminant natijasi nolga teng yoki katta bo’lsa, u holda quyidagi formula bilan tenglama ildizlari topiladi: $x1 = \frac{-b-\sqrt{d}}{2a}$; $x2 = \frac{-b+\sqrt{d}}{2a}$

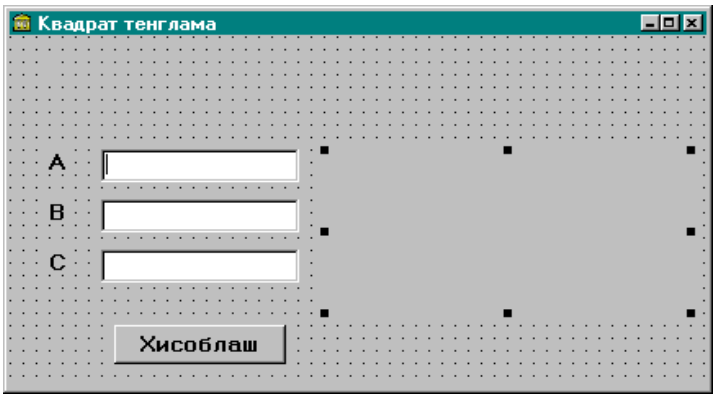
Agar diskriminant natijasi noldan kichik bo’lsa, bu tenglamaning xaqiqiiy ildizi yo’qligini bildiradi.

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

boshlash/tamom kiritish/chiqarish qayta ishlash tanlash

Rasm. Algoritmning blok-sxemasi uchun asosiy shakllar





-Rasm. Kvadrat tenglamani hisoblash algoritmining blok-sxemasi

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

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

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

Rasm. Kvadrat tenglamaning dialogli oynasi

Dastur matni

```
unit Kvadrat;

interface

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

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

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

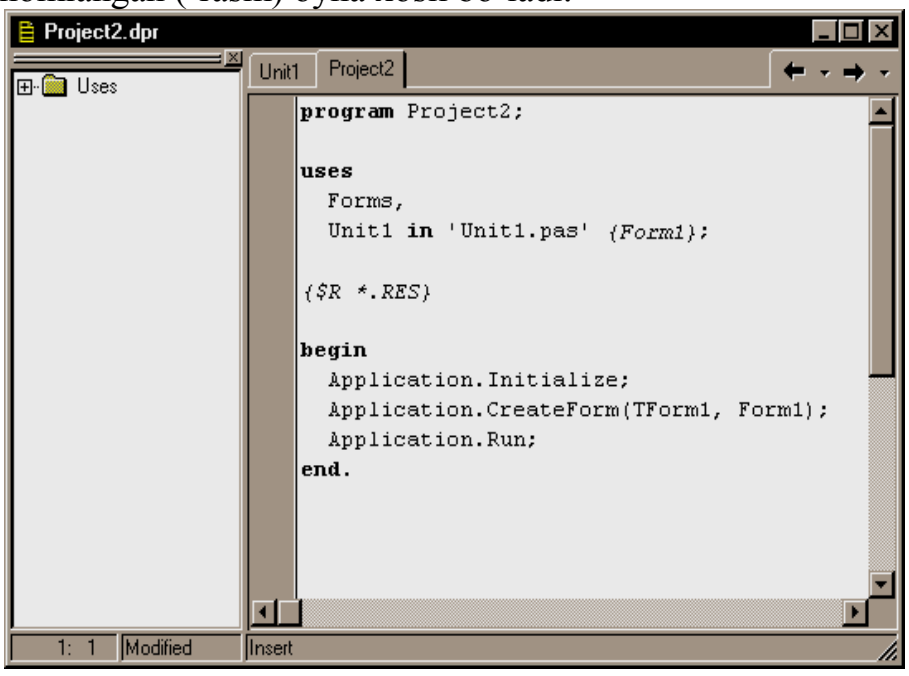
```
        + #13 + 'x1= ' + FloatToStr(x1)
        + #13 + 'x2= ' + FloatToStr(x2);
  End;
End;

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

end.
```

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

Konsolli ilovalar

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

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



-rasm

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

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

Dastur matni

```
{$APPTYPE CONSOLE}
Program Project2;
Var
  k, f: Real;
 Begin
   Writeln('Buyum og'irlugini kilogrammda kiriting');
   Writeln('va  <Enter> tugmasini bosing');
   Write('→');
   Readln(k);
   F := k * 0.4095;
   Writeln(k: 10: 4,'  kilogrammq', f: 10: 4,'  funt');
   Readln;
End.
```

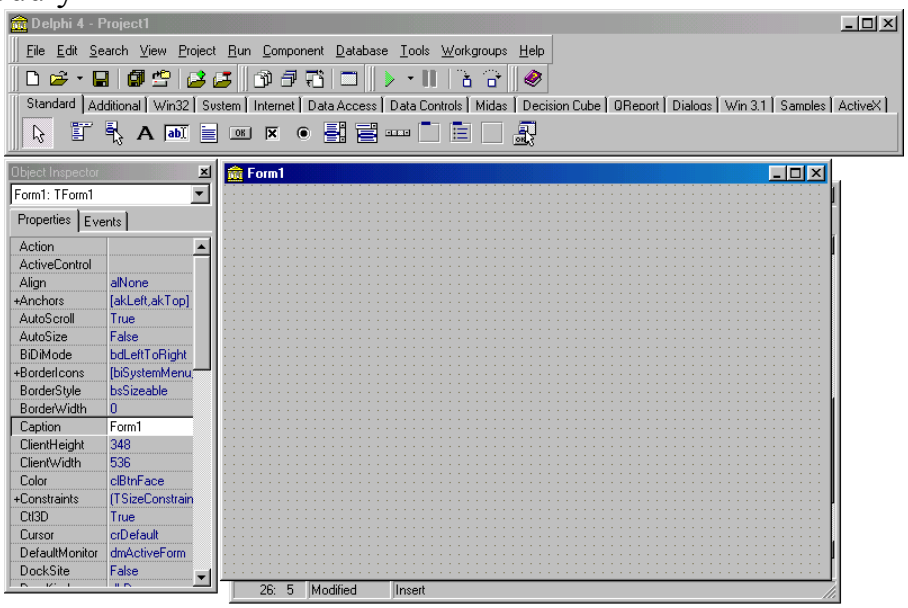
Yuqoridagi dasturda *{$APPTYPE CONSOLE}* qatori mavjud bo'lib, u izoh ko'rinishida yozilgan. Lekin u, dasturning konsolli ilova ekanligini bildiradi. Bunday dasturni tuzishda albatta *{$APPTYPE CONSOLE}* qatori yozilishi shart.

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

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

2. Delphi muhiti. Delphi da boshlang'ich amallar va proyektlar

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



-rasm.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Kodlar oynasida tahrirlash oddiy matn muharrirlari kabi amalga oshiriladi. Ya'ni belgilangan (blokka olingan) kod nusxasini olish, qirqib olish va kerakli joyga qo'yish mumkin. Undan tashqari kodlar ichidan kerakli belgini izlab topish va almashtirish, **Delete** tugmasi yordamida kursordan keyin turgan belgini, **Backspace** yordamida esa kursordan oldin turgan belgi yoki belgilarni o'chirish mumkin. **Ctrl+→, Ctrl+←** tugmachalari yordamida bir so'z keyinga va oldinga, **PgDn, PgUp** tugmachalari yordamida esa bir oyna pastga va yuqoriga o'tiladi.

Dastur bajarilayotganda yuz beradigan xatoliklar.

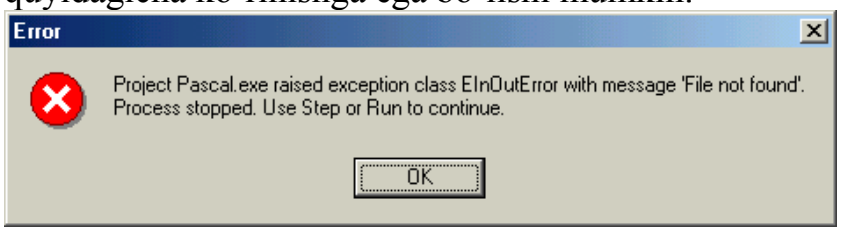
Odatda dastur tuzilayotganda ba'zi kamchilik yoki xatoliklarga yo'l qo'yamiz, dasturni ishga tushirgan vaqtimizda esa bu xatoliklar to'g'risida axborot beruvchi oyna hosil bo'ladi va bu xatoliklar quyidagilarga bo'linadi: Sintaktik; Dastur bajarilish vaqtidagi xatoliklar; Algoritmik.

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

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

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

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



Dastur bajarilish vaqtidagi xatolik.

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

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

Dasturni qadamlab bajartirish uchun birinchi navbatda **Run** menyusidagi **Add watch** (**Ctrl+F5**) buyrug'i tanlanadi va o'zgaruvchi qiymatlarini o'zida saqlaydigan (**Watch List**) oyna xosil bo'ladi. Unga xatolikni tekshirish jarayonida zarur bo'lgan o'zgaruvchilar kiritiladi. So'ngra kursor noto'g'ri bajarilayotgan algoritm sirtiga olib boriladi va **Run** menyusidan **Run to Cursor** (**F4**) buyrug'i tanlanadi. Yuqoridagi amalni bajarganimizdan so'ng, dastur ishga yuklanadi va dasturdagi ketma-ketlik kursor turgan joyga kelganida dastur o'z ishini vaqtinchalik to'xtatadi. Dasturning qolgan qismini **F7** tugmachasi yordamida qadamlab bajartirish mumkin va har bir qadamdan so'ng, **Watch List** oynasidagi o'zgaruvchi qiymatlari tekshirib boriladi.

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

3. Delphida komponentlar. Xususiyat va xodisalar.

Delphida komponentlar

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

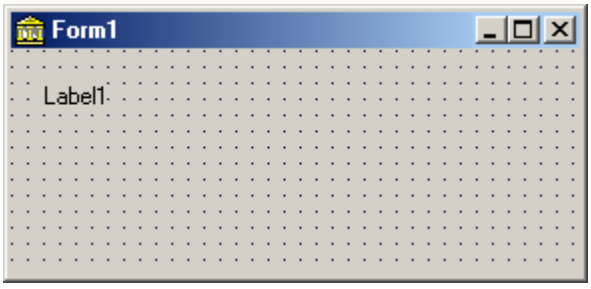
Komponentlar palitrasi guruhlar bo`yicha sahifalarga bo`lingan bo`lib, sahifalar soni 14 tadir. Komponentlar palitrasida **Standard, Additional, Dialogs** kabi sahifalar mavjud. Istalgan sahifa sirtida sichqonchaning chap tugmasi bosilsa, tanlangan sahifadagi komponentlar ro`yhati chqarib beriladi. Quyidagi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

Tanlangan komponent “Objekt Inspector” oynasi yordamida tahrirlanadi.

Xususiyat

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

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



Label komponentining xususiyatlari quydagilardir:

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

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

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

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

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

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

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

BiDiMode – Ikki tomonlama rejimdan foydalanish imkoniyatini yaratadi.

Caption – foydalanuvchi uchun izox vazifasini bajaradi. Undagi yozilgan matn ob`ekt sirtida o`z aksini topadi. Ampersand (&)qaysi belgining oldingi qismiga

qo`yilsa ana shu belgi ob`ektni aktivlashtiruvchi hisoblanadi. Alt+<belgi> bosilsa kerakli ob`ekt aktivlashadi. Misol uchun:

```
Label1.Caption := 'F&ile';
```

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

Color – Ob`ekt sirtining rangini tanlash imkonini beradi.

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

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

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

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

Font – ob`ektning sriftini o`zgartirish.

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

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

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

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

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

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

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

Tag – qo`shimcha xususiyat.

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

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

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

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

Xodisalar

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

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

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

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

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

end;
```

Biz unga quyidagi o`zgartirishni kiritamiz:

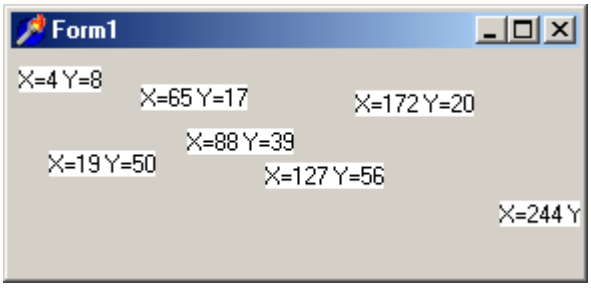
```
procedure TForm1.FormClick(Sender: TObject);
begin
  MessageDlg('Salomlar !!!', mtInformation, [mbOk], 0);
end;
```

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

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

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

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



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

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  MessageDlg(Chr(Key), mtInformation, [mbOk], 0);
end;
```

Dasturni ishga tushirib qanday natija chiqishini bilib olarsiz.

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

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

bu erda

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

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

```
unit Unit1;

interface

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

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

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

Uni quyidagicha o`zgartiramiz:

```
unit Unit1;

interface

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

type
```

```
  TForm1 = class(TForm)
  private
    { Private declarations }
   procedure WMRButtonDown(var Msg: TWMMouse); message WM_RBUTTONDOWN;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMRButtonDown(var Msg : TWMMouse);
begin
  MessageDlg('Sichqonchaning ung tugmasi bosildi.', mtInformation, [mbOK], 0);
end;

end.
```

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

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

4. Operator(buyruq)lar

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 matiqiy ifoda yolgon 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 moxiyatiga 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 parametrinig boshlang’ich qiymati;

k2 - sikl parametrining oxirgi qiymati;

S - sikl tanasi.

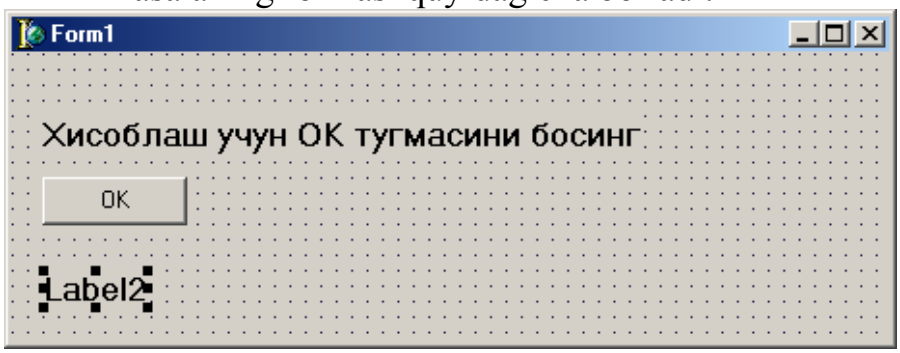
Operatorning ishlash prinsipi: sikl parametri (SP) boshlang’ich qiymat k1 ni qabul qilib agar bu qiymat k2 dan kichik bo’lsa shu qiymat uchun S operatori bajariladi;

SP ning qiymati yangisiga o’zgartirilib (agar k son bo’lsa o’zgarish kadami 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 bilsakkina 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'lik 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 qicmida 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* proseduraga 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 masalani mohiyatini buzib yuborishi mumkin. Bunday hollarda quyidagi formada yoziluvchi while sikl operatoridan foydalanish maqsadga muvofiqdir:

```
  while B do S;
```

bu yerda **while** (hozir), **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 siklni 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 sintaksis aniqlanmasi quyidagicha:

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

Variant tanlash operatorini 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?

5.Hulosa

Hulosa shiki Delphi dasturlash tili eng yuqori darajadagi tillardan biri hisoblanadi. Ushbu tilda Windows tizimida ishlash kulay. Delphi dasturida ishlash kulay, keng ko’lamda masalalarni echish uchun foydalanuvchiga yahshi imkoniyat beradi. Delphi-da dasturlashni boshlovchilar uchun ham, amaliy dasturlovchilar uchun ham, sistema dasturlari uchun ham etarlicha imkoniyatga ega. Delphi dasturini barcha imkoniyatlarini bilish kerak ekan degan hilosaga kelish mumkin.