

**O‘ZBEKISTON RESPUBLIKASI  
OLIV VA O‘RTA MAXSUS TA‘LIM VAZIRLIGI  
O‘RTA MAXSUS, KASB-HUNAR TA‘LIMI MARKAZI**

---

**Sh.A.Nazirov, M.M.Musayev,  
A.N. Ne‘matov, R.V.Qobulov**

# **DELPHI TILIDA DASTURLASH ASOSLARI**

**Kasb-hunar kollejlari uchun o‘quv qo‘llanma**

*Oliy va o'rta maxsus, kasb-hunar ta'limi ilmiy-metodik  
birlashmalari faoliyatini muvofiqlashtiruvchi kengash tomonidan  
nashrga tavsiya etilgan*

Tuzuvchilar:

**Sh.A. Nazirov** – professor, **M.M.Musayev** – professor, **A.N.Ne'matov** – dotsent, **R.V.Qobulov**–dotsent

Taqrizchilar:

**B. Qurmanbayev** – O'zMU professori, **F.N. Nuraliyev** – O'zRFA Informatika instituti katta ilmiy xodimi, dotsent

N-M-N-Q  $\frac{2210010000 - 8}{M 352(04) - 2007}$  qat'iy buyurtma, 2007

ISBN 978-9943-03-010-7

© **Sh.A. Nazirov, M.M. Musayev, A.N.Ne'matov, R.V.Qobulov, G'afur G'ulom** nomidagi nashriyot-matbaa ijodiy uyi, 2007-y.

# K I R I SH

Kompyuter dunyosida ko'plab dasturlash tillari mavjud. Bir xil turdagi ishni bajaradigan dasturlarni Basic, Pascal, Ci kabi tillarda yozish mumkin. Lekin, qaysi dasturlash tili yaxshi? Bu savolga javob berish oddiy emas. Biroq shuni ishonch bilan aytish mumkinki, Pascal tili boshqa tillarga qaraganda dastur tuzishni o'rganish uchun ancha qulay til bo'lib hisoblanadi.

Pascal tili Shvetsariyalik olim N.Virt tomonidan yaratilib, keyinchalik Borland korporatsiyasi tomonidan rivojlantirildi. Bu til rivojlantirilib Turbo Pascal, Borland Pascal va keyinchalik esa Object Pascal nomini oldi. Hozirgi kunda Object Pascal tili asosi bo'lgan Windows muhitida ishlovchi Delphi dasturlash vositasida murakkab professional dasturlari ishlab chiqilmoqda.

Kompyuterda dasturlash oxirgi yillarda juda tez rivojlanib, dastur tuzishga qiziquvchilar soni oshib bormoqda. 10–15 yil oldin o'z dasturlarini Windows muhitida yaratish ko'pgina dasturchilarning orzusi edi. Delphi dasturlash vositasining yaratilishi esa nafaqat professional dasturchilar, balki oddiy dastur tuzuvchilar uchun ham keng yo'l ochib berdi.

Ushbu qo'llanma Pascal tilining asosiy operatorlari va Delphi dasturlash vositasida dasturlar yaratish texnologiyalarini o'z ichiga olgan.

Qo'llanma to'qqizta bo'limdan iborat. Har bir bo'limda nazariy ma'lumotlar va mavzuga doir misollar keltirilgan. Birinchi bo'limda algoritmlar, dasturlash til elementlari va uning standart funksiyalari keltirilgan. Ikkinchi bo'limda operatorlar, protseduralar va funksiyalar haqida ma'lumotlar berilgan. Uchinchi bo'limda Delphi dasturlash vositasining oyna elementlari, sinf va obyektlar haqida tushunchalari berilgan. To'rtinchi bo'limda Delphi dasturlash vositasi komponentlarida ishlash texnologiyalari keltirilgan va har bir mavzuga doir misollar keltirilib, ularni bajarish tartibi berilgan.

Beshinchi bo'limda Delphi dasturlash vositasining grafik va multimedia imkoniyatlari yoritilgan. Oltinchi va yettinchi bo'limlarda

Delphi dasturlash vositasining qo‘shimcha imkoniyatlari va komponentalari haqida ma’lumotlar berilgan. Sakkiz va to‘qqizinchi bo‘limlarda MBni yaratish texnologiyalari keltirilgan.

Qo‘llanmada asosiy e‘tibor dasturlar tuzish usullariga qaratilgan bo‘lib, keltirilgan materiallar ketma-ket berilgan, uning yordamida o‘quvchi kompyuterda tez mustaqil holda dastur tuzish imkoniga ega bo‘ladi va zamonaviy vizual dasturlash texnologiyalari bilan tanishadi.

Qo‘llanmada ko‘rsatilgan dasturlash texnologiyalari bo‘yicha dastur tuzishga harakat qilib ko‘ring. Natijada, siz juda oson yo‘llar bilan dastur tuzish mumkin ekanligiga ishonch hosil qilasiz. O‘ylaymizki, qo‘llanma bilan tanishgan kasb-hunar kollejlari va Oliy o‘quv yurtlari talabalari, magistrilari va aspirantlari kompyuterda Delphi dasturlash vositasida o‘z dasturlarini yaratishga kirishadi.

# I. ASOSIY TUSHUNCHALAR

## 1.1. Algoritm va dastur tushunchasi

Algoritm soʻzi buyuk matematik Al-Xorazmiyning nomi bilan bogʻliq boʻlib, u birinchi boʻlib arab raqamlaridan foydalangan holda, arifmetik amallarni bajarish qoidasini bayon etdi.

Elektron hisoblash mashinalarining vujudga kelishiga qadar algoritmgga har xil taʼrif berilib kelindi. Lekin ularning bari maʼno jihatdan bir-biriga juda yaqin edi.

**Algoritm** – bu qoʻyilgan masalaning yechimiga olib keladigan, maʼlum qoidaga binoan bajariladigan amallarning chekli qadamlar ketma-ketligidir. Boshqacha qilib aytganda, algoritm boshlangʻich maʼlumotlardan natijagacha olib keluvchi jarayonning aniq yozilishidir.

Har qanday algoritm maʼlum koʻrsatmalarga binoan bajariladi va bu koʻrsatmalarga buyruq deyiladi.

Algoritm quyidagi xossalarga ega: aniqlik, tushunarlilik, ommaviylik, natijaviylik va diskretlik.

Aniqlik va tushunarlilik – deganda algoritmida ijrochiga berilayotgan koʻrsatmalar aniq mazmunda boʻlishi tushuniladi. Chunki koʻrsatmalardagi noaniqliklar moʻljallangan maqsadga erishishga olib kelmaydi. Ijrochiga tavsiya etiladigan koʻrsatmalar tushunarli mazmunda boʻlishi shart, aks holda ijrochi uni bajara olmaydi.

Ommaviylik – deganda har bir algoritm mazmuniga koʻra bir turdagi masalalarning barchasi uchun ham oʻrinli boʻlishi, yaʼni umumiy boʻlishi tushuniladi.

Natijaviylik – deganda algoritmida chekli qadamlardan soʻng albatta natija boʻlishi tushuniladi.

Diskretlik – deganda algoritmlarni chekli qadamlardan tashkil qilib boʻlamlash imkoniyati tushuniladi.

Algoritmning uchta turi mavjud: chiziqli, tarmoqlanuvchi va takrorlanuvchi (siklik).

Chiziqli algoritmlar – hech qanday shartsiz faqat ketma-ket bajariladigan jarayonlardir.

Tarmoqlanuvchi algoritmlar ma'lum shartlarga muvofiq bajariladigan jarayonlardir.

Takrorlanuvchi algoritmlar – biror-bir shart tekshirilishi yoki biron parametrning har xil qiymatlari asosida chekli ravishda takrorlanish yuz beradigan jarayonlardir.

Algoritmlarni turli usullarda tasvirlash mumkin:

- ◆ soʻz bilan ifodalash;
- ◆ formulalarda berish;
- ◆ blok-sxemalarda tasvirlash;
- ◆ dastur shaklida ifodalash va boshqalar.

Algoritmlarni blok-sxema koʻrinishida tasvirlash qulay va tushunarli boʻlgani uchun eng koʻp ishlatiladi. Bunda algoritmdagi har bir koʻrsatma oʻz shakliga ega. Masalan: parallelogramm koʻrinishidagi belgi maʼlumotlarini kiritish va chiqarish; toʻgʻri toʻrtburchak belgisi hisoblash jarayonini, romb belgisi shartlarning tekshirilishini bildiradi.

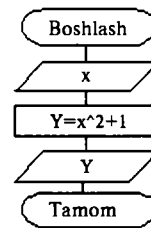
Misollar: Chiziqli algoritmga doir:

$y=x^2+1$  funksiyani  $x$  ning istalgan qiymatida hisoblash algoritmini tuzing.

Soʻzda berilishi:

1. Boshlash.
2.  $x$ -qiymatini kiritish.
3.  $y=x^2+1$  ni hisoblash.
4.  $y$ -qiymatini chiqarish.
5. Tamom.

Blok-sxemada:



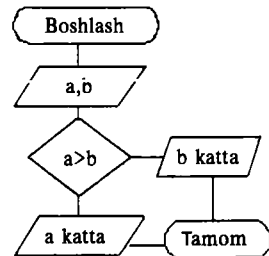
Tarmoqlanuvchi algoritmga doir:

Ikkita  $a$  va  $b$  sonlardan kattasini aniqlash algoritmini tuzing.

Soʻzda berilishi:

1. Boshlash.
2.  $a$  va  $b$ -qiymatini kiritish.
3. Agar  $a > b$  boʻlsa, natija  $a$  deb olinib  $S$ ga oʻtilsin.
4. Natija  $b$  deb olinsin.
5. Tamom.

Blok-sxemada:



Takrorlanuvchi algoritmga doir:

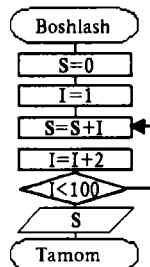
1 dan 100 gacha toq sonlar yigʻindisini hisoblash algoritmini tuzing.

Soʻzda berilishi:

1. Boshlash.
2.  $S$  ning qiymati nol deb olinsin.
3.  $i$  ning qiymati bir deb olinsin.

- 4.S ga i qo'shilib, natija S deb olinsin.
- 5.i ga 2 qo'shilib, uni i bilan belgilansin.
- 6.Agar  $i \leq 100$  bo'lsa, u holda 4 ga o'tilsin.
- 7.S qiymati chiqarilsin.
- 8.Tamom.

Blok-sxemada:



Masalani yechish algoritmi ishlab chiqilgandan so'ng dastur tuzishga o'tiladi.

**Dastur** – bu berilgan algoritmgaga asoslangan biror-bir algoritmik tilda yozilgan ko'rsatmalar (buyruqlar, operatorlar) to'plamidir.

**Dasturlash** – esa bu dastur tuzish jarayonidir. U quyidagi qadamlardan iborat:

- ◆ dasturga bo'lgan talablar;
- ◆ qo'yilgan masala algoritmini tanlash yoki ishlab chiqish;
- ◆ dastur kodlarini (matnlarni, buyruqlarni) yozish;
- ◆ dasturni to'g'rilash;
- ◆ test o'tkazish.

Hozirgi kunda juda ko'p algoritmik tillar mavjud bo'lib, ularni dasturlash tillari deb ataymiz. Algoritmik til – algoritmlarni bir xil va aniq yozish uchun ishlatiladigan belgilashlar va qoidalar tizimidir. Algoritmik til oddiy tilga yaqin bo'lib, u matematik belgilarni o'z ichiga oladi. Qo'yilgan masalalarni yechishga tuzilgan algoritmlarni to'g'ridan-to'g'ri mashinaga berib, yechib bo'lmaydi, shu sababli yozilgan algoritmnii biror-bir algoritmik tilga o'tkazish zarur. Har qanday algoritmik til o'z qo'llanilish sohasiga ega. Masalan, muhandislik hisob ishlarini bajarishda Pascal, Beysik va boshqalar. Ro'yxatlarni ishlash uchun PL/1 va boshqalar. Iqtisod masalalarini yechishda Pascal, Kobol va boshqalar. Mantiqiy dasturlash uchun Prolog va boshqalar. O'quv jarayonlari uchun Beysik, Pascal va boshqalar.

Pascal, Fortran va Kobol tillari universal tillardan hisoblanadi. Ci va Assembler tillari mashina tiliga ancha yaqin tillar bo'lib, o'rta darajadagi tillardir. Algoritmik til inson tillariga qancha yaqin bo'lsa, u tilga yuqori darajali til deyiladi. Mashina tili esa eng pastki darajali tildir.

## 1.2.Dasturlash tilining elementlari

Hozirgi kunda juda ko'p algoritmik tillar mavjud. Bu tillar ichida Pascal tili universal tillardan biri bo'lib, boshqa tillarga qaraganda imkoniyatlari kengroq tildir. So'nggi yillarda Pascal tili juda

takomillashib, tobora ommalashib bormoqda. Pascal tilida dastur tuzish uchun Turbo Pascal va Delphi dasturlash vositalari mavjud. Bu dasturlash vositalari zamonaviy kompyuter texnologiyasining hamma talablarini o'z ichiga olgan va unda dastur tuzuvchi uchun hamma qulayliklar yaratilgan.

Delphi dasturlash vositasi Turbo Pascal tilining rivoji bo'lgan Object Pascal tilini ishlatadi. Hozirgi kunda bu tilga juda ko'plab yangiliklar kiritilgan, uning imkoniyatlari yanada kengaytirilgan, shu sabab bu tilni Delphi tili deb ham atash mumkin.

Delphi tili ham boshqa dasturlash tillari kabi o'z alfavitiga va belgilariga ega. U 26 ta bosh lotin harflarini, 0 dan 9 gacha bo'lgan arab raqamlarini va quyidagi belgilarni ishlatadi: bo'shliq belgisi; 4 ta arifmetik amallar +, -, \*, /; mantiqiy amallarni bajarish uchun <, >, <=, >=, <>, = belgilarini ishlatadi. Bulardan tashqari vergul, nuqta, ikki nuqta, kichik qavs, katta va o'rta qavs. Dasturda izohlar istalgan joyda berilishi mumkin. Ular katta qavs ichida yoziladi.

Masalan. Program ad; { Bu dastur nomi }.

### 1.3. O'zgarmlar, o'zgaruvchilar va standart funksiyalar

Haqiqiy turdagi sonlar umumiy holda quyidagi ko'rinishda bo'ladi:

$$s a_1 a_2 \dots a_n . b_1 b_2 \dots b_k$$

Bu yerda s ishora (+ yoki -) yoki bo'sh joy;  $a_1 a_2 \dots a_n$  butun qism;  $b_1 b_2 \dots b_k$  kasr qism. Masalan: +3,147 soni +3.147 yoki 3.147.

$$-143,03 \text{ soni } -143.03$$

$$57,0 \text{ soni } 57.0$$

$$0,493 \text{ soni } 0.493 \text{ yoki } .493$$

Haqiqiy sonlarning o'zgarish diapazoni kompyuterning turiga qarab turlicha bo'ladi.  $10^{-38} < x < 10^{+38}$  x-ixtiyoriy son. Ular eksponensial (darajali) ko'rinishda ifodalanishi ham mumkin, ya'ni  $\pm m10^{\pm n}$ . Bunday sonlar quyidagicha yoziladi  $\pm mE \pm n$ . Masalan:

$$0,43 \cdot 10^{-6} \quad .43E-6$$

$$0,0003 \quad 3E-4$$

Butun sonlar umumiy holda quyidagicha yoziladi  $s a_1 a_2 \dots a_n$ .

$$\text{Masalan: } +345 \text{ soni } +345 \text{ yoki } 345$$

$$-106 \text{ soni } -106$$

Butun sonlar o'zgarish diapazoni -32768 dan +32767 gacha. Agar butun son qiymati bu diapazondan chiqsa, u haqiqiy son shaklida ifodalanadi yoki kompyuter turiga qarab, u o'n otililik sanoq tizimida ifodalanishi ham mumkin. Belgililar shtrix ichida yoziladi. Yozilish diapazoni 0 dan 255 tagachadir. Misol. 'Pascal', '405.5'.



Pascal tilida identifikator tushunchasi mavjud bo'lib, dasturda obyektlarni nomlashda ishlatiladi. O'zgarma(lar)ni, o'zgaruvchi(lar)ni, belgi(metka), protsedura va funksiyalarni belgilashda ishlatilgan nom **identifikatorlar** deyiladi. Identifikatorlar lotin alfaviti harflaridan boshlanib, qolgan harflari belgi yoki raqam ketma-ketligidan tashkil topgan bo'lishi mumkin. Masalan: xx, xx1, alfa&.

Delphi tilida dastur ishlashi mobaynida qiymati o'zgarmaydigan identifikatorlar **o'zgarma(lar)** deyiladi va ular dasturning bosh qismida **Const** so'zi bilan e'lon qilinib, unga aniq qiymat tenglashtiriladi.

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

Dastur ishlashi mobaynida qiymatlari o'zgarishi mumkin bo'lgan identifikatorga **o'zgaruvchi(lar)** deyiladi va ular dastur bosh qismida **Var** so'zi bilan e'lon qilinadi. O'zgaruvchi(lar) nomi keltirilib, ularning turlari beriladi. O'zgaruvchi(lar)ning eng ko'p ishlatiladigan turlari **butun, haqiqiy, belgili, qator** va **mantiqiy**dir. Ular mos ravishda butun – **Integer**, haqiqiy – **Real**, belgili – **Char**, qator (matn) – **String** va mantiqiy – **Boolean** deb yoziladi.

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

Mantiqiy o'zgaruvchi(lar) faqat ikkita qiymat qabul qiladi: «True» (chin) va «False» (yolg'on).

### Standart matematik funksiyalar

Funksiya nomi	Tilda yozilishi	Ma'nosi
Sinx	SIN(x)	x ning sinusi
Cosx	COS(x)	x ning kosinusi
Ln x	LN(X)	x ning natural logarifmi
e <sup>x</sup>	EXP(x)	Eksponenta
$\sqrt{x}$	SQRT(x)	Kvadrat ildiz
Arctgx	ARCTAN(x)	x ning arktangensi
x	ABS(x)	x ning moduli
x <sup>2</sup>	SQR(x)	x ning kvadrati
a <sup>b</sup>	EXP(b*LN(a))	a ning b chi darajasi

## Nostandart matematik funksiyalar.

$$1. \operatorname{Sec} x = \frac{1}{\operatorname{Cos} x} \quad 2. \operatorname{Cosec} x = \frac{1}{\operatorname{Sin} x} \quad 3. \operatorname{Tgx} = \frac{\operatorname{Sin} x}{\operatorname{Cos} x} \quad 4. \operatorname{Arcctg} x = \operatorname{Arctg} \frac{1}{x}$$

$$5. \operatorname{Arc} \sin x = \operatorname{Arctg} \frac{x}{\sqrt{1-x^2}} \quad 6. \operatorname{Arc} \cos x = \operatorname{Arctg} \frac{\sqrt{1-x^2}}{x}$$

$$7. \operatorname{Arc} \sec x = \operatorname{Arctg} \frac{1}{\sqrt{1-x^2}} \quad 8. \operatorname{Arc} \operatorname{cosec} x = \operatorname{Arctg} \sqrt{1-x^2}$$

$$9. \operatorname{Log}_a b = \frac{\operatorname{Ln} b}{\operatorname{Ln} a} \quad 10. \operatorname{Padian} = \frac{\operatorname{Gradus} \cdot \pi}{180}$$

## O'zgartirish funksiyalari

Funksiya	Qiymati
Chr(n)	Kodi n ga teng simbol
IntToStr(k)	Butun k ni tasvirlovchi satr
FloatToStr(n)	Haqiqiy n ni (2) tasvirlovchi satr
FloatToStrF(n,f,k,m)	Haqiqiy n ni (2) tasvirlovchi satr Bunda: f – format; k – aniqlik m-kasr qismidagi raqamlar soni
StrToInt(s)	Satrni butun songa o'tkazish
StrToFloat(s)	Satrni haqiqiy songa o'tkazish
Round(n)	Haqiqiy sonni yaxlitlash
Trunc(n)	Haqiqiy sonning kasr qismini olib tashlash
Frac(n)	Kasrli sonning kasr qismi
Int(n)	Kasrli sonning butun qismi

Dasturda arifmetik va mantiqiy ifodalar o'zgaruvchi, o'zgaruvmas, standart funksiyalar, qavslar va amal belgilari orqali tashkil qilinadi.

Ifodalarda hisoblashlar tartibi qavslar ichidagi ifodalar bajarilgandan keyin quyidagi tartibda bajariladi:

1. NOT amali;
2. \*, /, DIV, MOD, AND;
3. +, -, OR;
4. Taqqoslash belgilari: <, >, <=, >=, <>, =, IN.

Ifodadagi amal natijasi qanday turda bo'lishi amallarda qatnashayotgan o'zgaruvchilarning turlariga bog'liq. Agar ikkita o'zgaruvchining turi Integer yoki Real bo'lsa, amal natijasi ham

Integer yoki Real bo'ladi. Agar biri Integer, ikkinchisi Real bo'lsa natija Real bo'ladi. NOT, OR, AND va taqqoslash amallarining natijalari esa Boolean turida bo'ladi.

Kompyuter foydalanuvchi tomonidan qo'yilgan masalani aniq va tushunarli ko'rsatmalar berilgandagina bajara oladi. Bu ko'rsatmalar ma'lum bir ma'noni anglatuvchi so'zlardan iborat bo'lib, kompyuterga qanday amalni bajarish lozimligini bildiradi va bu ko'rsatmalarga **operatorlar** deyiladi. Operatorlar dastur ishlaganda ketma-ket ravishda bajariladi. Delphi tilida bir satrga bir necha operatorlarni yozish mumkin.

Delphi tilida dastur matni bosh va asosiy bo'limdan tashkil topadi. Bosh bo'lim dastur nomi va o'zgaruvchilar, o'zgarmaslar, massivlar, belgilar (metkalar), protseduralar va funksiyalarni tavsiflashdan iborat bo'ladi. Asosiy bo'lim dastur tanasi deyilib, unda dasturda bajariladigan hamma operatorlar ketma-ketligi beriladi va u Begin (boshlamoq) so'zi bilan boshlanib End (tugash) so'zi bilan tugaydi. Umumiy holda dastur tuzilmasi quyidagi ko'rinishga ega:

```
Program <dastur nomi>;  
Uses <Foydalanadigan bibliotekalar (modullar) ro'yxati>;  
Label <Ishlatiladigan belgilar (metkalar) ro'yxati>;  
Const <Ishlatiladigan o'zgarmaslarni aniqlash>;  
Type <Yangi turlarni aniqlash>;  
Var <O'zgaruvchilarni e'lon qilish>;  
      <Protsedura va funksiyalarni aniqlash>;  
Begin  
      <Bajariladigan operatorlar ketma-ketligi>;  
End.
```

## 1.4.Ma'lumotlar turlari

Ma'lumotlar turlarini Delphi tilida umumiy holda ikkiga ajratish mumkin:

- ◆ standart turlar. Bu turlar oldindan Delphi tili tomonidan aniqlangan bo'ladi;

- ◆ dasturchi tomonidan kiritiladigan (aniqlanadigan) turlar.

Standart turlar tarkibiga quyidagilar kiradi: butun, haqiqiy, belgili (simvol), qator, mantiqiy, ko'rsatgichli variant.

Dasturchi turlarni dasturning **Var** bo'limida o'zgaruvchilarni tavsiflashda aniqlaydi yoki maxsus turlarni aniqlash uchun bo'lim bo'lgan – **Type** turlarni tavsiflash bo'limida aniqlanadi.

Bu bo'lim umumiy holda quyidagicha bo'ladi.

## Type

<tur nomi>=<turning tavsifi>;

Misol:

Type

TColor=(Red, Blue, Black);

Var Color1, Color2, Color3: TColor;

Type bo'limida dasturchi tomonidan yangi Tcolor nomli tur kiritilmoqda va u Red,Blue,Black mumkin bo'lgan qiymatlarni qabul qilishi mumkin.

Var bo'limida dasturchi tomonidan turi aniqlangan uchta Color1, Color2, Color3 o'zgaruvchilar tavsiflanmoqda.

Bu o'zgaruvchilarni to'g'ridan-to'g'ri quyidagicha ham tavsiflash mumkin.

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

Standart turlarni Type bo'limida tavsiflash shart emas, ularni to'g'ridan-to'g'ri Var bo'limida tavsiflash mumkin.

Delphida standart turlarni quyidagicha klassifikatsiya qilish mumkin.

◆ Oddiy

♣ Tartibli

- Butun
- Belgi
- Mantiqiy
- Sanoqli (Перечисляемый)
- Chegaralangan

♣ Haqiqiy

◆ Qator

◆ Tuzilma

♣ To'plam

♣ Massiv

♣ Yozuv

♣ Fayl

♣ Klass

♣ Interfeys

◆ Ko'rsatgichli

◆ Protsedurali

◆ Variant

Oddiy turlarga tartiblashgan va haqiqiy turlar kiradi. Tartiblashgan turlar shu bilan xarakterlanadiki, uning har bir qiymati o'zining tartiblangan nomeriga ega. Haqiqiy tur qiymatlari kasr qismidan iborat bo'lgan sonlardan iboratdir.

Tartiblashgan turlarga butun, belgili, mantiqiy, sanoqli va chegaralangan turlar kiradi.

**Butun turlar.** Butun turlar butun sonlarni tasvirlash uchun ishlatiladi.

**Haqiqiy turlar.** Haqiqiy turlar haqiqiysonlarni tasvirlash uchun ishlatiladi.

**Belgili turlar.** Ma'lumotlarning belgili turlari faqat bitta belgini saqlash uchun xizmat qiladi.

**Mantiqiy turlar.** Mantiqiy turlar chin (True) yoki yolg'on (False) qiymatning birini qabul qiladi.

Tur	O'zgarish diapazoni	O'lcham (baytda)
Integer	-2147483648..2147483647	4
Cardinal	0..4294967295	4
Shjrtint	-128..127	1
Smallint	-32768..32767	2
Longint	-2147483648..2147483647	4
Int64	$-2^{63}..2^{63}-1$	8
Byte	0..255	1
Word	0..65535	2
LonoWord	0..4294967295	4

#### Dasturchi tomonidan kiritiluvchi turlar

Delphi tili dasturchiga o'zining turlarini kiritishga imkon beradi.

Tur	O'zgarish diapazoni	O'lcham (baytda)
Real	$5.0 \cdot 10^{-324}..1.7 \cdot 10^{308}$	8
Real48	$2.9 \cdot 10^{-39}..1.7 \cdot 10^{38}$	6
Single	$1.5 \cdot 10^{-45}..3.4 \cdot 10^{38}$	4
Double	$5.0 \cdot 10^{-324}..1.7 \cdot 10^{308}$	8
Extended	$3.6 \cdot 10^{-4951}..1.1 \cdot 10^{4932}$	10
Comp	$-2^{63}+1..2^{63}-1$	8

Bu turlar standart turlarga yoki avval kiritilgan turlarga asoslangan bo'lib quyidagi turlarga tegishli bo'lishi mumkin:

- sanovchi;
- interval;
- murakkab tur (yozuv).

Tur	O'lcham (baytda)
Char	1
ANSChar	1
WideChar	2

**Sanoqli turlar.** Sanoqli turlar tartiblangan qiymatlar to'plamini ishlatadi.

Tur	O'lcham (baytda)
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Tur = ( 1 Qiymat, 2 Qiymat, ... ,I Qiymat)

Masalan:

Type

Color=(black, green, yellow, blue, red, white);

Fam=( Petrov, Sidorov, Rahimov, Sobirov);

DayOfWeek=(mon, tue, wed, thu, fri, sat, sun);

Bu yerda

Color sanoq turi beshta ranglar ketma-ketligini aniqlaydi.

Fam sanoq turi to'rtta familiyani aniqlaydi.

DayOfWeek sanoq turi hafta nomlarini aniqlaydi.

Odatda Delphi tilida turlar nomlari T harfidan boshlanadi (Type — tip so'zidan).

Yangi tur ta'riflangandan so'ng shu turga tegishli o'zgaruvchini ta'riflash mumkin. Masalan:

Type

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

var

ThisDay, LastDay: TDayOfWeek;

Sanovchi tur ta'rifi qiymatlar o'zaro munosabatini ko'rsatadi. Eng chap element minimal, eng o'ng element maksimal hisoblanadi. Yuqorida kiritilgan DayOfWeek turi elementlari uchun quyidagi munosabat o'rinni:

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

Sanovchi tur elementlari orasidagi munosabat o'zgaruvchilarni boshqaruvchi instruksiyalarda qo'llashga imkon beradi. Masalan:

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

```
begin
```

```
{ agar kun shanba yoki yakshanba bo'lsa bajarilsin }
```

```
end;
```

Bu instruksiyani quyidagicha yozish mumkin:

```
if Day > FRI then begin
```

```
{ agar kun shanba yoki yakshanba bo'lsa bajarilsin }
```

```
end;
```

Sanovchi tur ta'rifini nomlangan konstantalarni kiritishning qisqartirilgan shakli deb qarash mumkin. Misol uchun TDayOfWeek turining ta'rifini quyidagi ta'riflarga tengdir:

Const

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

**Interval (diapazon) turi.** Interval (diapazon) turi beriladigan qiymatga chegara qo'yadi.

Type

<tur nomi>=<minimal>..**<maksimal>**;

Masalan:

Type

Color=red..green; // rangga chegara

Digit=0..9; //butun sonlarga chegara

Symb='A'..'Z'; // harflarga chegara

Haqiqiy turlarga chegara qo'yilmaydi.

Interval tur ta'rifida nomlangan konstantalardan foydalanish mumkin. Quyidagi misolda interval tur TIndex ta'rifida HBOUND nomlangan konstantadan foydalanilgan:

Const

HBOUND=100;

type

TIndex=1..HBOUND;

Interval turdan foydalanish massivlarni ta'riflashda quyidagidir:

Type

TIndex =1 .. 100;

var

tabl : array[TIndex] of Integer; i:TIndex;

Butun son turidan tashqari asos tur sifatida sanovchi turdan foydalanish mumkin. Quyidagi dastur qismida TMonth sanovchi tur asosida interval tur TSammer ta'riflangan:

Type

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

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

TSammer = Jun.. Aug;

## Yozuv

Dasturlash amaliyotida standart ma'lumotlardan tashkil topgan murakkab ma'lumotlar bilan ishlashga to'g'ri keladi. Misol uchun talaba to'g'risidagi ma'lumotda uning ismi sharifi, tug'ilgan yili, manzili, kursi, guruhi va hokazolardan iborat bo'lishi mumkin. Bunday ma'lumotlarni ta'riflash uchun Delphi da yozuv (record) lardan foydalaniladi.

**Yozuv** – bu alohida nomlangan har xil turli komponentalardan iborat murakkab turdir.

Har qanday tur kabi «yozuv» type bo‘limida ta’riflanishi lozim.  
Bu ta’rifning umumiy ko‘rinishi:

Nom = record

```
1_Maydon: 1_Tip; 2_Maydon: 2__Tip;...; K_Maydon: K__Tip; end;
```

Ta’riflarga misollar:

Type

TPerson = record

```
f_name: string[20];
```

```
l_name: string[20];
```

```
day: integer;
```

```
month: integer;
```

```
year: integer;
```

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

TDate = record

```
day: Integer; month: integer; year: integer;
```

```
end;
```

Yozuv turidagi o‘zgaruvchini quyidagicha ta’riflash mumkin:

Var

```
student : TPerson; birthday : TDate;
```

Yozuv elementiga (maydoniga) murojaat qilish uchun yozuv nomi va nuqtadan so‘ng maydon nomini ko‘rsatish kerak. Masalan:

```
Writeln («Imya», student.f_name + #13 + `Adres:`, student.address);
```

Instruksiya ekranga student o‘zgaruvchi–yozuvning f\_name (nom) va address (adres) maydonlarini chiqaradi.

Ba’zida o‘zgaruvchi – yozuv turi o‘zgaruvchilar e’lon qilish bo‘limida e’lon qilinadi. Bu holda, yozuv turi o‘zgaruvchi nomidan so‘ng ko‘rsatiladi. Misol uchun student yozuvi var bo‘limida quyidagicha ta’riflanishi mumkin:

```
student: record
```

```
f_name:string[20];
```

```
l_name:string[20];
```

```
day:integer;
```

```
month:integer;
```

```
year:integer;
```

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

```
end;
```

### **With instruksiyasi**

With instruksiyasi dasturda maydonlar nomlarini o‘zgaruvchi – yozuv nomini ko‘rsatmasdan ishlatishga imkon beradi. Umumiy holda with instruksiyasi quyidagi ko‘rinishga ega:

with nom do



begin

( dastur instruksiyasi } end;

Misol uchun dasturda quyidagi yozuv ta'riflangan bo'lsin:

student: record

f\_name: string[30];

l\_name: string[20];

address: string[50];

end;

va studentlar to'g'risidagi ma'lumotlar E1, E2 va E3 o'zgaruvchilarda joylashgan bo'lsin. U holda:

student.f\_name := E1;

student.l\_name := E2;

student.address := E3;

instruksiyalar o'rniga quyidagi instruksiyani yozish mumkin:

with student do begin

f\_name := E1; l\_name := E2; address := E3;

end;

## S a v o l l a r

1. Algoritm nima va u qanday xossalarga ega?
2. Algoritmning qanday turlari mavjud?
3. Qanday algoritmik tillar bor?
4. Dastur va dasturlash nima?
5. Masalani EHMda yechish qanday bosqichlardan iborat?
6. O'zgaruvchi va o'zgaruvchilar dasturda qanday tavsiflanadi?
7. O'zgaruvchilarning qanday turlari mavjud?
8. Mantiqiy o'zgaruvchilar qanday qiymat qabul qiladi?
9. Qanday standart matematik funksiyalar mavjud?
10. Pascal tilida dastur qanday tuzilmaga ega?
11. With instruksiyasi dastur tuzishda qanday imkoniyat yaratadi?
12. Yozuv qanday tur?
13. Ma'lumotlar turlarini Delphi tilida umumiy holda qanday turlarga ajratish mumkin?
14. Dasturchi tomonidan kiritiluvchi turlar qanday turlar?

## M a s h q l a r

1. Pascal tilidagi  $3E-4$  va  $0.2E5$  sonlarini oddiy yozuvda yozing.
2.  $1000000$  va  $0.0000001$  sonlarini eksponensial sonlar shaklida yozing.
3. Qiymatlarni tasvirlash uchun qaysi turdagi o'zgaruvchilar ishlatiladi?
  - a) narsalar sonini aniqlashda;
  - b) tenglama koeffitsiyentlari;
  - c) ikkita son katta-kichikligini aniqlashda.

- d) mahsulot nomlari;
  - e) o'rtacha temperatura;
  - f) bir yildagi dam olish kunlari.
4. Quyidagi tavsiflangan o'zgaruvchilar qanday qiymat qabul qilishi mumkinligini toping.  
Var a, b: Integer; c: Real; w: String; f: Boolean;

### **J a v o b l a r**

1. 0.0003 va 20000.0
2. 1E6 va 1E-7
3. a)Integer; b)Real; c)Boolean; d)String; e)Real; f)Integer.
4. a va b – butun; c – haqiqiy; w – matn qatori; f – mantiqiy.

## **II. OPERATORLAR, PROTSEDURA VA FUNKSIYALAR**

### **2.1. Ma'lumotlarni kiritish va chiqarish operatorlari**

Biror-bir masalani yechishning chiziqli bo'lgan algoritmgiga dastur tuzishda algoritmdagi keltirilgan ketma-ketliklar asosida operatorlar yoziladi. Bunday dasturlarni tuzishda asosan o'zgaruvchilar qiymatni kiritish, natijalarni chiqarish va shu bilan birga o'zlashtirish operatorlari ishlatiladi.

Dasturdagi o'zgaruvchilar qiymatlarini dastur ichida o'zlashtirish operatori yordamida ham berish mumkin. Lekin dasturga o'zgaruvchi qiymatni tashqaridan kiritish qulaylik tug'diradi va umumiylikni ta'minlaydi.

**Read** operatori o'zgaruvchilar qiymatlarini ekrandan kompyuter xotirasiga kiritish uchun ishlatiladi. U quyidagi ko'rinishlarga ega:

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

Bu yerda c1,c2,...,cn – o'zgaruvchilar nomi; ln qo'shimchasi qiymatni kiritib keyingi qatorga o'tishni bildiradi.

Misollar: Read(Sm1,Sm2);

```
Readln(x1,x2,x3);  
Readln.
```

Bu yerda birinchi operator Sm1 va Sm 2 o'zgaruvchilar qiymatini ekrandan kiritadi. Ikkinchi operator esa x1,x2,x3 o'zgaruvchilar qiymatini ekrandan kiritadi va kiritishni keyingi qatorga o'tkazadi. Oxirgi operator esa kiritishni kutadi va qatorga o'tkazadi.

**Write** operatori oddiy ma'lumotlarni va o'zgaruvchilar qiymatlarini kompyuter ekraniga chiqarish uchun ishlatiladi. U quyidagi ko'rinishlarga ega:

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

Bu yerda c1,c2,...,cn – oddiy matnlar yoki o'zgaruvchilar nomi; ln qo'shimchasi chiqarishni keyingi qatorga o'tishini bildiradi.

Misollar: Write(Summa);  
Write('Natija yo'q');  
Write('Tenglama yechimi x1=', x1, 'x2=', x2);

Oddiy ma'lumotlarni chiqarishda ularga matn deb qaraladi va u qo'shtirnoq ichida yoziladi. Chiqarish operatori yordamida o'zgaruvchilar qiymatini format ko'rinishda ham berish mumkin:

Write(c:m:n);

Bu yerda c – o'zgaruvchi; m—shu o'zgaruvchi qiymatning uzunligi; n—qiymatning kasr qismi va unda  $n-1 < m$  bo'lishi kerak.

Misol: Write(x:8:4);

Agar  $x=155.01021$  bo'lsa, quyidagi yozuv chiqadi 115.0102.

Write('Mahsulot soni:', kol:5);

Agar kol=15 bo'lsa, quyidagi yozuv ekranga chiqadi:

Mahsulot soni: 15.

Dastur matnini tushuntirish maqsadida ko'pincha dasturda izohlar keltiriladi. Dasturda izohlar istalgan joyda berilishi mumkin. Izoh katta qavs ichida yoziladi.

Masalan: { Bu matn dasturga izoh beradi }  
{ Bu joyda yechim aniqlanmoqda }

Dasturda ma'lum hisoblashlarning natijalarini biror-bir o'zgaruvchida saqlash uchun o'zlashtirish (yuborish) operatori ishlatilib, u «:=» belgisi yordamida qiymat yuborilishi kerak bo'lgan o'zgaruvchidan keyin qo'yiladi.

Masalan: i:=0; i – qiymati nolga tenglashadi, ya'ni i o'zgaruvchiga nol yuboriladi deb tushuniladi. Bunda mashina i o'zgaruvchi uchun ajratilgan xotirasiga nol yozib saqlaydi.

Misol: B:=5; C:=4; A:=(B+C)/2;

Bu yerda, agar A butun identifikator bo'lsa, uning qiymati aniqlanmaydi, aks holda esa 4.5 qiymatga ega bo'ladi.

Chiziqli tuzilmali algoritmlarni dastur shaklida yozish uchun oldin ishlatiladigan o'zgaruvchilar ro'yxati keltirilib, keyin algoritmdagi bajarilishlar ketma-ket ravishda amalga oshirilishi kerak.

Misol: Tekislikda ikki nuqta orasidagi masofani topish dasturi.

**Program XY;**

**Var**

**x1,y1,x2,y2,d: Real;**

**Begin**

**Write('Nuqta koordinatalarinl kiriting:');**

**Read(x1,y1,x2,y2);**

**d:=Sqrt(Sqr(x1-x2)+Sqr(y1-y2));**

**Writeln;**

**Writeln('Nuqta koordinatalari:',x1,y1,x2,y2);**

**Writeln('Masofa=',d);**

**Readln;**

**End.**

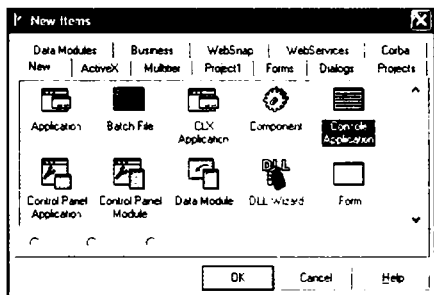
## 2.2.Delphining konsol ilovasini yaratish

Delphida konsol ilovalarini har xil usullarda yaratish mumkin. Ulardan eng oson usuli quyidagicha:

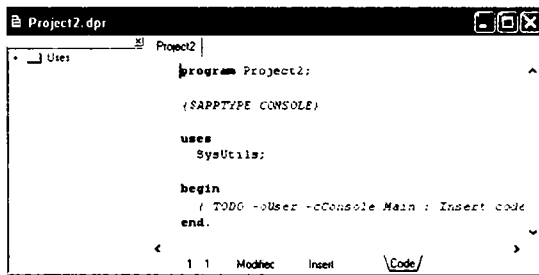
1.Delphi muhiti ishga tushiriladi.

**Пуск=>Программы=>Borland Delphi**

2.Bosh menyudan File punktini ochib, u yerdan New, keyin esa Other buyruqlari beriladi. **File=> New=> Other**



3.Forma va loyihalarni saqlash uchun ochilgan maxsus oynadan (bu oynaga Delphi arxiv oynasi deyiladi) «Console Application» piktogrammasi tanlanadi va Ok tugmasi bosiladi.



4.Natijada ekranda loyiha oynasi ochiladi (.dpr kengaytmali nom bilan).

Begin – end ichiga olingan.

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

izohi o'rniga loyiha faylining dastur matni kiritiladi.

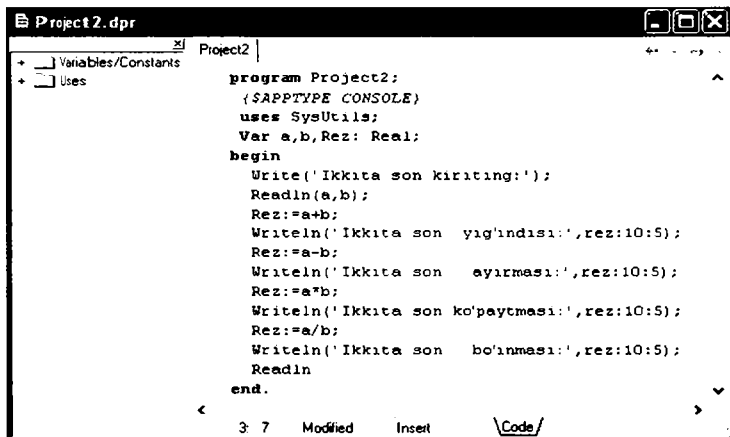
Tuzilgan dasturni ishga tushirishdan oldin uni saqlash kerak bo'ladi. Uni saqlash uchun file=>Save All buyruq'ini berish lozim. Har bir loyiha alohida yangi papkada saqlanishni tavsiya etadi. Loyiha faylini saqlashda alohida ko'rsatilmagan holatida ProjectN.dpr nomli fayl nomini tavsiya etadi. Bu yerda N har bir ketma-ket nomlanadigan loyiha nomeri (son, masalan 1,2,3,..). Lekin biz loyiha faylini istalgan nom bilan saqlashimiz mumkin. Masalan. MyProgram.dpr. Bu nom avtomatik ravishda chiqadi.

Loyihani saqlab bo'lgandan so'ng, uni bajarishga beramiz. Buning uchun bosh menyudan quyidagi buyruqni berish lozim: Run=>Run yoki F9 funksional tugmachasini bosish kerak bo'ladi. Dastur normal ishga tushgandan so'ng ekranda DOSning standart dastur oynasi namoyon bo'ladi.

Misol: Ikkita sonning yig'indisi, ayirmasi, ko'paytmasi va bo'linmasini hisoblash dasturini yarating.

Bu misolni yechish uchun yuqorida keltirilgan to'rtta ketma-ketlikni bajaramiz va dastur kodini kiritamiz.

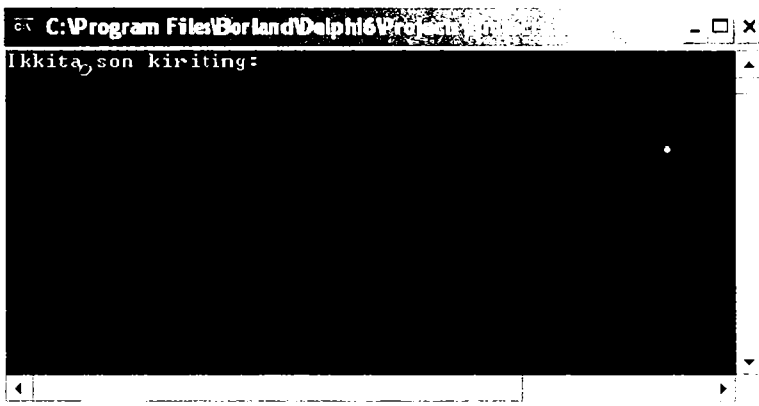
Dastur kodi kiritilgandan so'ng uni saqlab keyin ishga tushiramiz. Natijada ekranda Dos oynasi ochilib, unda «Ikkita son kiriting:» so'zi chiqadi. Keyin ikkita son kiritilib Enter tugmasini bosish kerak bo'ladi.



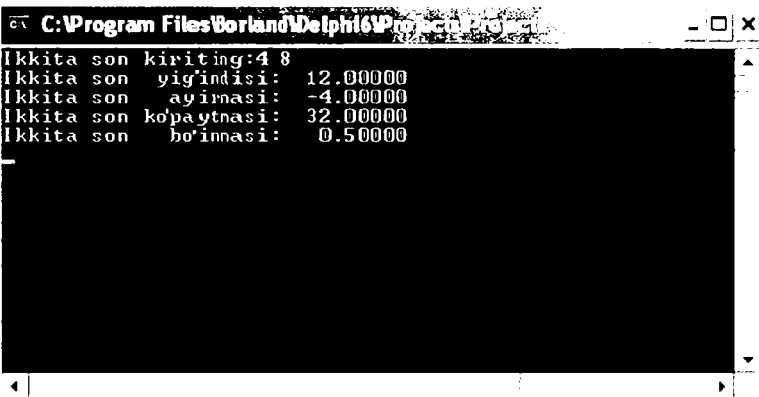
```
Project2
+ Variables/Constants
+ Uses
program Project2;
($APPTYPE CONSOLE);
uses SysUtils;
var a,b,Rez: Real;
begin
  Write('Ikkita son kiriting:');
  Readln(a,b);
  Rez:=a+b;
  Writeln('Ikkita son yig'indisi:',rez:10:5);
  Rez:=a-b;
  Writeln('Ikkita son ayirmasi:',rez:10:5);
  Rez:=a*b;
  Writeln('Ikkita son ko'paytmasi:',rez:10:5);
  Rez:=a/b;
  Writeln('Ikkita son bo'linmasi:',rez:10:5);
  Readln
end.
```

Natijada quyidagi javoblar chiqadi.

5.Keyin obyekt inspektori (Object Inspector) va obyekt daraxtlar (Object TreeView) oynalari yopiladi.



```
C:\Program Files\Borland\Delphi6\Pro...
Ikkita son kiriting:
```



```
C:\Program Files\Borland\Delphi6\Pro...
Ikkita son kiriting:4 8
Ikkita son yig'indisi: 12.00000
Ikkita son ayirasi: -4.00000
Ikkita son ko'paytmasi: 32.00000
Ikkita son bo'linasi: 0.50000
```

6. Bosh menyudan Project=>View Source buyrug'i beriladi.  
Endi Delphida konsol ilovasini yaratishning ikkinchi usulini ko'rib chiqamiz:

1. Delphi muhiti ishga tushiriladi.

**Пуск=>Программы=>Borland Delphi**

2. Bosh menyudan File punktini ochib, u yerdan New, keyin esa, Application buyruqlari beriladi. **File=>New=>Application**

3. Forma oynasi yopiladi.

4. Dastur kodini yozish (modul) oynasi yopiladi. Yopish vaqtida «Save changes to Unit1.pas?» («Unit1.pasdagi o'zgarishlar saqlansinmi?») so'rov oynasi chiqadi. U yerdan «NO» (Yo'q) buyrug'i beriladi.

Natijada quyidagi Project1.dpr loyiha fayli oynasi ekranga chiqadi.

7. Loyiha fayli agar lozim bo'lsa boshqa nom bilan saqlanadi.

Bu oynadan Program, Uses, Begin va End kalit so'zlari qoldirilib boshqalari o'chiriladi va keyin dasturga matn kodlari kiritiladi.

```
program Project2;
uses
  Forms;

{$R *.res}

begin
  Application.Initialize;
  Application.Run;
end.
```

1: 1 Modified Insert Code/

Agar loyiha fayli yozilgan papkaning ichi qaralsa, unda quyidagi fayllar ro'yxatini ko'ramiz.

- MyProgram.dpr – loyiha fayli (bosh loyiha moduli);
- MyProgram.exe – ilova fayli yoki bajariluvchi fayl. Bu fayl kompilyator yordamida, ya'ni kompilyatsiya jarayonida, agar dasturda sintaktik xatoliklar bo'lmasa tuziladi. Boshqacha so'z bilan aytganda, agar sizga o'z dasturingizni ishga tushirish mumkin bo'lsa, masalan, F9 tugmasini bosish bilan bajariluvchi fayl avtomatik ravishda tuziladi. Bajariluvchi fayl avtonom fayl bo'lib, uning uchun boshqa fayl yoki biror dasturiy sistema mavjud bo'lishi shart emas. Uni siz ishga tushirishingiz mumkin bo'lgan boshqa dasturlar kabi, masalan Paint, Bloknote yoki o'yin dasturlarini ishga tushirganday;
- MyProgram.cfg – loyiha konfiguratsiyasining fayli;
- MyProgram.dof – loyiha opsiyasining fayli. Unda dasturning to'g'ri ishlaganligi haqida axborotlar saqlanadi.

Loyiha opsiyasi va konfiguratsiyasining fayllari loyiha faylining tuzilishi bilan bir vaqtda Delphi tomonidan avtomatik ravishda tuziladi. Ko'p hollarda yuqorida keltirilgan fayllardan tashqari yana .dpr kengaytmali fayl ham tuziladi. Bu fayl loyiha faylining (rezerv fayli) nusxasi bo'lib hisoblanadi. Masalan, MyProgram.- dpr. Bu fayl loyiha fayli tuzilishi davrida bir vaqtning o'zida tuzib boriladi. Agar asosiy loyiha faylida buzilish yoki o'chirilish sodir bo'ladigan bo'lsa, u holda uni MyProgram.-dpr faylidan tiklash mumkin. Buning uchun kengaytma oldidagi «-» belgini olib tashlash kifoya.

### 2.3.Shartli o'tish operatori

Pascal tilida shart – bu mantiqiy turdagi ifoda bo'lib, u faqat «chin»(True) yoki «yolg'on»(False) qiymatni qabul qiladi.

Quyidagi mantiqiy belgilar ishlatiladi:  $>$ ,  $<$ ,  $<=$ ,  $>=$ ,  $<>$ ,  $=$ . Bularga munosabat amallari deyiladi.

Quyidagi mantiqiy amallar ishlatiladi:

- NOT – «inkor»;
- AND – «mantiqiy va»;
- OR – «mantiqiy yoki».

Bu mantiqiy amallarning bajarilish natijalari quyidagicha:

Op1	Op2	Op1 AND Op2	Op1 OR Op2	NOT Op1
False	False	False	False	True
True	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Masalan:  $(5 < 6)$  AND  $(6 < 50)$  – mantiqiy ifoda rost (True),

$(20 > 0)$  OR  $(20 < 0.5)$  – mantiqiy ifoda rost (True),

$(10 < 8)$  AND  $(10 < 15)$  – mantiqiy ifoda yolg'on (False),

NOT  $(100 > 3)$  – mantiqiy ifoda yolg'on (False).

Mantiqiy ifodalarni biror-bir mantiqiy o'zgaruvchiga yuborish ham mumkin.

Masalan:

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

Bu yerda, agar ikkala shart bajarilgandagina F mantiqiy o'zgaruvchi «chin» (True) qiymatni qabul qiladi. Aks holda «yolg'on» (False) qiymatni qabul qiladi.

Pascal tilida shartli o'tish operatorining ikki xil ko'rinishi mavjud: to'liq va qisqa.

To'liq ko'rinish:

```
If <shart> then Begin  
  <shart rost bo'lganda bajariladigan operatorlar>  
  End  
  Else  
    Begin  
    <shart yolg'on bo'lganda bajariladigan operatorlar>  
    End;
```

Qisqa ko'rinish:

```
If <shart> then Begin  
  <shart rost bo'lganda bajariladigan operatorlar>  
  End;
```

Bu yerda IF – agar; then – u holda; else – aks holda ma'nosini bildiruvchi xizmatchi (kalit) so'zlar.



Birinchi ko'rinishdagi shartli operatorlarda agar shart bajarilsa birinchi Begin va End ichidagi operatorlar ketma-ket bajariladi, aks holda ikkinchi Begin va End ichidagi operatorlar ketma-ket bajariladi.

Ikkinchi ko'rinishdagi shartli operator quyidagicha ishlaydi. Agar berilgan shart bajarilsa Begin va End ichidagi operatorlar ketma-ket bajariladi, aks holda ular bajarilmaydi.

Agar bajariluvchi operatorlar soni bitta bo'lsa Begin va End so'zlarini yozish shart emas.

Misollar:

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

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

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

Har bir shartli o'tish operatori ichida boshqa ichki shartli operatorlar joylashishi ham mumkin. Masalan:

If  $b_1$  then  $a_1$  else If  $b_2$  then  $a_2$  Else  $a_3$ ;

Misollar:

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

**Javob:**  $0.5 < -1.7$  yolg'on bo'lganligi sababli  $B := A$  operator bajariladi va bunda  $A = 0.5$  va  $B = 0.5$  ekanligi kelib chiqadi.

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

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

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

**Javob:**  $(0.1 < 0.1)$  yoki  $(0.1 > 0.5)$  bu mantiqiy ifoda yolg'on bo'lganligi sababli  $B = A$  shart tekshiriladi. Bu shart chin bo'lganligi sababli  $D = 0.5$  ga,  $C = 0.1$  qiymatlarga teng ekanligi kelib chiqadi.

## 2.4. Shartsiz o'tish va tanlash operatorlari

Dasturda shunday holatlar bo'ladiki, operatorlarning bajarilish shartiga qarab dasturning u yoki bu qismiga to'g'ridan-to'g'ri o'tishga to'g'ri keladi. Bunday holatlarda shartsiz o'tish operatoridan foydalanish mumkin.

Shartsiz o'tish operatorining ko'rinishi quyidagicha:

**Goto n;**

Bu yerda  $n$  – belgi(metka) bo'lib identifikator yoki butun son bo'lishi mumkin. Goto – o'tish ma'nosini bildiradi.

$n$  – belgi dasturning bosh qismida Label so'zi yordamida e'lon qilingan bo'lishi shart.  $n$  boshqarilish uzatiladigan joyga  $n$ : shaklida qo'yiladi.

Misol:

.....

Goto L2;

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

.....  
Ko'p hollarda biror-bir parametrning qiymatiga qarab kerakli operatorlarni bajarishga to'g'ri keladi. Bunday hollarda, tanlash operatorini ishlatgan qulay. Tanlash operatorining ko'rinishi quyidagicha bo'ladi:

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

Bu yerda Case – xizmatchi so'z bo'lib, tanlash ma'nosini beradi; of – «dan» ma'nosini beradi; s – operator selektori; 1,2,..n – operator belgilari; A1,A2,..An va B1,B2,..Bn-operatorlar.

Case operatori tarmoqlanish jarayonida berilgan bir necha operatoridan birini tanlash yo'li bilan amalga oshiradi. Operatorlar ketma-ketligini tanlash operator selektorining qiymatiga qarab aniqlanadi. Operator selektori haqiqiy bo'lmagan o'zgaruvchi yoki ifoda bo'lishi mumkin. Agar operator selektori qiymati operator belgilari o'zgarmas qiymatiga teng bo'lmasa B1,B2,..Bn-operatorlari ketma-ket bajariladi.

Shartli O'tish operatorining ko'rinishi quyidagicha:

```
If B Then A1 Else A2;  
tanlash operatorining quyidagi operatoriga ekvivalentdir.  
Case B of
```

```
  True: A1;  
  False: A2;  
End;
```

**Misol:**

$ax^2+bx+c=0$  kvadrat tenglamaning ildizlarini topish dasturi tuzilsin.

**PROGRAM Corni;**

**Label 20;**

**Var A, B,C, D, E, F, X, X1, X2, Z:real;**

**BEGIN**

**Writeln ('a, b, c koefitsiyentlar qiymatini kiriting:');**

```

Read (A,B,C);
if A=0 THEN
  BEGIN
    x:=-B/C; writeln (x);
  goto 20
end
else
  BEGIN
  D:=B*B-4.0*A*C; F:=soht(d)/Z;
  F:=sqrt(d)/Z;
  End;
  if D>0 THEN
  BEGIN
    x1:=E+F; x2:=E-F; writeln (x1,x2);
  End
  else if D=0 then
  Begin X:=E; writeln(x); end else writeln ('yechim yo'q');
20: end.

```

## 2.5.Sikl operatorlari

Ayrim masalalarda bir yoki bir necha parametrlarning o'zgarishiga qarab ma'lum hisoblashlar bir necha marta takrorlanib bajarilishi mumkin. Masalan,  $y=ax+b$  funksiyani  $x$  ning bir necha qiymatida uning mos qiymatlarini hisoblash kerak deylik. Bunday hisoblashlarni kompyuterda dastur tuzib bajarish uchun siklik tuzilmali dasturlar tuzish kerak bo'ladi. Bu kabi dasturlarni shartli o'tish operatori yordamida ham tuzish mumkin. Lekin Pascal tilida siklik strukturali dastur tuzish uchun bir necha maxsus operatorlar mavjud.

**For** operatori takrorlanishlar soni aniq bo'lgan siklik jarayonlar tashkil etishda ishlatiladi. Uning umumiy ko'rinishi quyidagicha:

**For i:=m1 to m2 Do S;**

Bu yerda  $i$  – sikl parametri;  $m_1, m_2$  – i parametrining boshlang'ich va oxirgi qiymati bo'lib, ular o'zgarmas son yoki ifoda bo'lishi mumkin;  $S$  – sikl tanasi bo'lib, bir necha operatorlardan tashkil topishi mumkin.

Agar sikl tanasi bir necha operatoridan iborat bo'lsa, ular **Begin** va **End** ichiga olinadi.

Misol: 1,2,...10 sonlar yig'indisini hisoblash dasturini tuzing.

Program S10;

Const kn=10;

Var i: Integer; S: Real;

```

Begin
  S:=0;
  For i:=1 to kn do S:=S+i;
  Write ('S=',S);  Readln;
  End.

```

Agar to so'zini **DoWnto** so'ziga almashtirilsa sikl parametri teskari bo'yicha o'zgaradi, ya'ni -1 qadam bilan. U holda sikl ko'rinishi quyidagicha bo'ladi.

**For i:=m1 DoWnto m2 Do S;**

Misol: 10 dan 1 gacha sonlarni ekranga chiqarish dasturini tuzing.

Program SP;

Var i: Integer;

Begin

For i:=10 DoWnto 1 do Write (i); Readln;

End.

**While** sikl operatori takrorlanishlar soni oldindan aniq bo'lmagan hollarda takrorlanishni biror-bir shart asosida bajaradi. Berilgan shart oldin tekshiriladi va keyin shartning bajarilishiga qarab kerakli operatorlar ketma-ketligi bajariladi. Bu operatorning umumiy ko'rinishi quyidagicha: **While B Do S;**

Bu yerda B – mantiqiy ifoda; S – sikl tanasi bo'lib, bir yoki bir necha operatorlar ketma-ketligidan iborat bo'lishi mumkin. Mantiqiy ifoda 'True' yoki 'False' qiymatini qabul qiladi.

Agar mantiqiy ifoda 'True' qiymatni qabul qilsa S operatorlari bajariladi, aks holda bajarilmaydi, ya'ni sikl ishlashdan to'xtaydi.

Misol: 1,2,...,10 sonlar yig'indisini hisoblash dasturini tuzing.

Program S10;

Const kn=10;

Var i: Integer; S: Real;

Begin

S:=0; i:=0;

While i<=kn do Begin i:=i+1; S:=S+i; End;

Write ('S=',S);

Readln;

End.

**Repeat** sikl operatori ham takrorlanishlar soni oldindan aniq bo'lmagan hollarda takrorlanishni biror-bir shart asosida bajaradi. Oldin sikl tanasidagi operatorlar ketma-ketligi bajariladi. Berilgan shart keyin tekshiriladi. Agar berilgan shart rost (True) bo'lsa, boshqaruv sikldan keyingi operatorni bajarishga o'tadi, aks holda, sikl takrorlanadi. Bu operatorning umumiy ko'rinishi quyidagicha:

## Repeat

### S

## Until B

Bu yerda B—mantiqiy ifoda, ‘True’ yoki ‘False’ qiymatni qabul qiladi; S—sikl tanasi bo‘lib, bir necha operatorlar ketma-ketligidan iborat bo‘lishi mumkin. Agar mantiqiy ifoda ‘False’ qiymatni qabul qilsa siklda takrorlanish davom etadi, aks holda, to‘xtaydi.

Misol: 1,2,...10 sonlar yig‘indisini hisoblash dasturini tuzing.

Program S10;

Const kn=10;

Var i: Integer; S: Real;

Begin

S:=0; i:=0;

Repeat

i:=i+1; S:=S+i;;

Until I>kn;

Write (‘S=’,S);

Readln;

End.

Odatda WHILE operatori REPEAT operatoriga nisbatan ko‘p ishlatiladi. Bunga sabab ko‘pchilik masalalarda sikl tugallanish shartini sikl boshlanmasdan avval tekshirish maqsadga muvofiqdir. Zarur bo‘lsa siklni umuman bajarmasdan o‘tish mumkin.

- Ko‘pchilik masalalarni yechishda tuzilgan dasturda ichma-ich joylashgan sikllarni tashkil etishga to‘g‘ri keladi. Bunday sikllarga murakkab sikllar deyiladi. Murakkab sikllar tashkil etilganda quyidagi talablar bajarilishi zarur:

- ichki sikl tashqi sikl ichida to‘liq yotishi kerak;
- sikllar bir-biri bilan kesishmasligi kerak;
- sikl ichiga tashqaridan to‘g‘ridan-to‘g‘ri kirish mumkin emas;
- sikl parametrlari boshqa-boshqa identifikatorlar bilan belgilanishi kerak.

Misol:  $S = \sum_{i=1}^{10} \prod_{j=1}^5 \frac{i+j}{\sqrt{i \cdot j}}$  ifodani hisoblash dasturini tuzing.

Bu formulada agar yig‘indini ochsak u quyidagi ko‘rinishga keladi.

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

**Program SP;**

**Var**

**i,j: Integer; S,P: Real;**

**Begin**

**S:=0;**

**For i:=1 to 10 do**

**Begin**

**P:=1;**

**For j:=1 to 5 do P:=P\*(i+j)/Sqrt(i\*j);**

**S:=S+P;**

**End; Write ('S=',S);**

**End.**

## 2.6. Massivlar

Ko'p hollarda jadval yoki matritsalar ko'rinishidagi ma'lumotlar bilan ish yuritish kerak bo'ladi. Jadvalda ma'lumotlar juda ko'p bo'lgani sabab, ularning har bir yacheykasidagi sonni mos ravishda bitta o'zgaruvchiga qiymat qilib berilsa, ular ustida ish bajarish ancha noqulayliklarga olib keladi. Shu sababli, dasturlashda bunday muammolar massivlarni ishlatish yordamida hal qilinadi.

Massiv – bu bir nom bilan belgilangan qiymatlar guruhi yoki jadvaldir. Massivning har bir elementi massiv nomidan, so'ng o'rtga qavs ichiga olingan raqam va arifmetik ifoda yozish bilan belgilanadi. Qavs ichidagi raqam massiv indeksini belgilaydi. Vektorni bir o'lchovli massiv, matritsani ikki o'lchovli massiv deb qarash mumkin.

Bir o'lchovli massivda uning har bir elementi o'zining joylashgan o'rin nomeri bilan aniqlanadi va nomeri qavs ichida indeks bilan yoziladi. Ikki o'lchovli massiv elementi o'zi joylashgan satr va ustun nomerlari yordamida aniqlanadi. Shu sababli, ikki o'lchamli massiv elementi ikkita indeks orqali yoziladi. Masalan:  $A[i,j]$  bu yerda  $i$ —sitr nomeri,  $j$ —ustun nomerini bildiradi.

Massivni e'lon qilish dasturning bosh qismida berilib, uning yozilishi umumiy holda quyidagicha bo'ladi:

**<Massiv nomi>:Array[o'lcham] of <element turi>;**

Masalan:

**A,B:Array[1..100] of real;**

**C,A1,D:Array[1..10,1..15] of real;**

Bu yerda **A** va **B** massivlari 100 tadan elementga ega. **C,A1,D1** massivlari esa  $10 \times 15 = 150$  tadan elementga ega.

Massivlarni e'lon qilishdan maqsad, massiv elementlari uchun kompyuter xotirasidan joy ajratishdir.

Massiv elementlari qiymatlarini kiritish uchun sikl operatorlaridan foydalaniladi.

Misol: For i:=1 to 10 do Read (A[i]);

Bu misolda A massivning 10 ta elementi qiymatini ekrandan ketma-ket kiritish kerak bo'ladi. Xuddi shunday massiv qiymatlarini ekranga chiqarish ham mumkin.

Misol: For i:=1 to 10 do Write(A[i]);

Dasturda massiv elementlarini ishlatganda ularning indeksi e'lon qilingan chegaradan chiqib ketmasligi kerak.

### **Massiv elementlarini tartiblash usullari**

Massivni tartiblashtirishning bir necha usullari (algoritm) mavjud. Ulardan quyidagi usullarni qarab chiqamiz:

- tanlash usuli;
- almashtirish usuli.

**Tanlash** usuli yordamida massivni o'sish bo'yicha tartiblashtirish algoritmi quyidagicha:

1. Massivning birinchi elementidan boshlab qarab chiqilib eng kichik element topiladi.

2. Birinchi element bilan eng kichik element joylari almashtiriladi.

3. Ikkinchi elementidan boshlab qarab chiqilib eng kichik element topiladi.

4. Ikkinchi element bilan eng kichik element joylari almashtiriladi.

5. Bu protsess bitta oxirgi elementgacha takrorlanadi.

Bu algoritm dasturi quyidagicha bo'ladi:

**Program Sort;**

**Const Size=5;**

**Var i,j,min,k,buf: Integer;**

**a: Array[1..Size] of Integer;**

**Begin**

**Writeln ('Massivni tartiblashtirish');**

**Write (Size:3, 'ta massiv elementini kiriting');**

**For k:=1 to Size Do Read(a[k]);**

**Writeln ('Tartiblashtirish');**

**For i:=1 to Size-1 Do**

**Begin**

**{ kichik elementni topish }**

```

min:=i;
For j:=i+1 to Size Do
  Begin
    If a[j]<a[min] then min:=j;
    buf:=a[i]; a[i]:=a[min]; a[min]:=buf;
    For k:=1 to Size Do Write (a[k], ' ');
    Writeln;
  End;
End;
Writeln('Massiv tartiblashtirildi.');
```

**End.**

Dastur natijasi:

Massivni tartiblashtirib,  
5 ta massiv elementini kiriting:

12 -3 56 47 10

Tartiblashtirish

-3 12 56 47 10

-3 10 56 47 12

-3 10 12 47 56

-3 10 12 47 56

Massiv tartiblashtirildi.

**Almashtirish** usuli yordamida massiv elementlarini o'sib borishida tartiblashtirish algoritmi quyidagicha:

1. Massivning birinchi elementidan boshlab ketma-ket hamma qo'shni elementlar bir-biri bilan solishtirilib, agar birinchisi ikkinchisidan kichik bo'lsa, ular joyi almashtirilib boriladi.

2. Bu protsess davomida kichik qiymatli elementlar massiv boshiga, katta elementlar esa oxiriga siljitib boriladi. Shu sabab bu usul «пузырёк» usuli ham deyiladi.

3. Bu protsess massiv elementlar sonidan bitta kam takrorlanadi.

Masalan:

3 2 4 5 1 bunda 3 bilan 2 va 5 bilan 1 almashtiriladi.

2 3 4 1 5 bunda 4 bilan 1 almashtiriladi.

2 3 1 4 5 bunda 3 bilan 1 almashtiriladi.

2 1 3 4 5 bunda 2 bilan 1 almashtiriladi.

1 2 3 4 5

Bu algoritm dasturi quyidagicha bo'ladi:

**Program Sort;**



```

Const Size=5;
  Var i, j, min, k, buf: Integer;
  a: Array[1..Size] of Integer;
  Begin
    Writeln ('Massivni puzyr k (ko'pikcha) usulida tartib-
lashtirish');
    Write (Size:3, 'ta massiv elementini kiriting');
    For k:=1 to Size Do Read(a[k]);
    Writeln ('Tartiblashtirish');
    For i:=1 to Size-1 Do
      Begin
        For k:=1 to Size-1 Do
          Begin
            If a[k]>a[k+1] then
              Begin
                buf:=a[k]; a[k]:=a[k+1]; a[k+1]:=buf;
              End;
            End;
          End;
        For k:=1 to Size Do Write (a[k], ' ');
        Writeln;
      End;
    Writeln('Massiv tartiblashtirildi.');
```

Dastur natijasi:

Massivni puzyr k usulida tartiblashtirib,  
5 ta massiv elementini kiriting:

3 2 4 1 5

Tartiblashtirish

2 3 4 1 5

2 3 1 4 5

2 1 3 4 5

1 2 3 4 5

Massiv tartiblashtirildi.

Massivda eng kichik yoki eng katta elementni izlash algoritmi, ma'lumki, birinchi element eng kichik (katta) deb olinib keyin boshqa elementlar bilan ketma-ket solishtirilib chiqiladi. Solishtirilish oxirgi elementgacha bajariladi.

Quyida bu algoritm dasturi keltirilgan:

**Program MinMax;**

```

Var i,min: Integer;
    a: Array[1..10] of Integer;
Begin
Writeln ('Massivdan eng kichik elementni izlash');
Write (' 10-ta massiv elementini kiriting');
For i:=1 to 10 Do Read(a[i]);
    min:=1;
    For i:=2 to 10 Do
        If a[i]<a[min] Then min:=i;
    Writeln('Izlanayotgan eng kichik element:',a[min]);
    Writeln('Element nomeri',min);
End.

```

### **Dinamik massiv**

Dinamik massiv ta'riflanganda uning uzunligini ko'rsatish shart emas.

Massiv uzunligini o'rnatish uchun `SetLength` funksiyasidan foydalanish mumkin. Uning ikki parametri mavjud:

1. Dinamik massiv tipidagi o'zgaruvchi:
2. Massiv uzunligi.

`High(r)` funksiyasi massiv elementlari sonini qaytaradi.

Misol:

```

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

```

`IntToStr` funksiyasi sonni satrga aylantiradi.

## **2.7. Qism dasturlari**

Dasturlash jarayonida shunday holatlar bo'ladiki, bir xil operatorlar ketma-ketligini dasturning bir necha joylarida takroran yozishga to'g'ri keladi. Bunday takrorlanishni yo'qotish maqsadida dasturlashning ko'pgina tillarda qism dasturining tushunchasi kiritilgan. Takrorlanadigan operatorlar ketma-ketligini mustaqil dastur bo'lagi – qism dastur ko'rinishida bir marotaba yoziladi va bu dastur bo'lagi, kerak bo'lgan

joylarda esa, unga murojaat qilinadi, xolos. Pascal tilida qism dastur protsedura yoki funksiya ko‘rinishida beriladi.

Ayrim masalalarni yechishda ma’lum parametrlarning har xil qiymatlarida bir xil hisoblashlarni bajarishga to‘g‘ri keladi. Bunday hollarda dastur hajmini kichraytirish maqsadida protsedura yoki funksiyalar tashkil qilish zarur. Protsedura yoki funksiyaga murojaat qilish dasturda uning nomini ko‘rsatish orqali amalga oshiriladi. Kerakli parametrlar shu nomdan keyin beriladi. Protsedura yoki funksiyalar tashkil qilinganda ular dasturning bosh qismida beriladi. Ularga murojaat qilish esa, dasturning asosiy qismining kerakli joyida beriladi. Asosiy dastur bilan protsedura orasida o‘zgaruvchilar qiymat almashuvi formal va faktik parametrlar yordamida amalga oshiriladi. Protsedura yoki funksiyaga murojaat qilinganda boshqarilish qayerdan uzatilsa yana shu joyga qaytib keladi. Protsedura ichida yana bir necha protsedura yoki funksiya ishlatilishi mumkin. Dasturda e‘lon qilingan o‘zgaruvchilar, shu dasturdagi protsedura va funksiyalarga nisbatan global deyiladi. Protsedura va funksiyalar ichida e‘lon qilingan o‘zgaruvchilar lokal deyiladi. Ularning ta’sir doirasi shu protsedura va funksiyalarning ichida bo‘ladi, xolos.

Protseduralarni e‘lon qilish dasturning bosh qismida keltiriladi va u quyidagicha boshlanadi.

**Procedure** <pros.nomi> (<formal parametrlar>);

M: **Procedure** AB (x,y);

Formal parametrlarni shu protsedura bosh qismida yoki sarlavhada e‘lon qilish mumkin.

M. **Procedure** AB (x,y: Real);

Har qanday protsedurani kichik bir dastur deb qarash mumkin. Protsedura ham dasturga o‘xshab bosh va asosiy qismlardan tashkil topadi. Bosh qismda protsedura nomi va uning parametrlari e‘lon qilinadi. Asosiy qism operatorlar ketma-ketligidan tashkil topgan bo‘lib, ular Begin – End ichiga olinadi. Protsedura nomi foydalanuvchi tomonidan beriladi.

**Misol:**

**Procedure** Dr(Var x,h1,h2,z1,z2 : Real);

Var h,z: Real;

Begin

h:=h1/z1+h2/z2;

z:=z1/z2;

x:=(h+z)/2;

End;

Bu protsedurada  $h1, z1, h2, z2$  parametrlar qiymati protseduraga murojaat qilinganda aniqlangan bo'lishi kerak. Natijani esa  $x$  – parametr uzatadi.  $h$  va  $z$  o'zgaruvchilar ichki o'zgaruvchilardir. Bu protseduraga dasturdan quyidagicha murojaat qilinadi.

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

Protseduraga murojaat qilinganda mos parametrlar qiymati bir-biriga uzatiladi. Beriladigan formal va faktik parametrlar soni teng va ular turlari bir xil bo'lishi shart. Lekin parametrlar nomlari har xil bo'lishi mumkin.

Funksiyalardan foydalanish va ularni tashkil qilish xuddi protsedura kabi bo'lib, ularni e'lon qilish dasturning bosh qismida keltiriladi va u quyidagicha boshlanadi:

**Function <f-ya nomi>(<formal parametrlar>):<f-ya turi>;**

M. Function Min ( $x, y$ :Real): Real;

Funksiya nomi foydalanuvchi tomonidan beriladi. Funksiyaga murojaat qilish uning nomi orqali beriladi.

Funksiya ham protseduraga o'xshab bosh va asosiy qismlardan tashkil topadi. Funksiyaning protseduradan farqi, unga murojaat qilinganda natija faqat bitta bo'lib, u shu funksiya nomiga uzatiladi.

**Misol 1.** Quyidagi hisoblashni funksiyani ishlatgan holda dasturni tuzing.

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

**Program Kol;**

**Var n,m,lnm: Integer; NCM: Real;**

**Function Fact (k: Integer): Real;**

**Var P,i: Integer;**

**Begin**

**P:=1;**

**For i:=1 to k do P:=P\*i;**

**Fact:=P;**

**End;**

**Begin**

**Read(n,m); nm:=n-m;**

**ncm:=Fact(n)/Fact(m)/Fact(nm);**

**Write('ncm=',ncm);**

**End.**

**Misol 2.** Quyidagi hisoblashning dasturini protsedurani ishlatgan holda tuzing.

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

```

Program Fun1;
  Var a,b,z,c,d,t1,t2,t3: Real;
  Procedure Th(Var x,r: Real);
    Var c: Real;
    Begin
      c:=exp(2.0*x);
      r:=(c-1)/(c+1);
    End;
  Begin
    Read(a,b); th(a,t1);
    c:=a-b; th(c,t2);
    d:=Sqr(a)-Sqr(b); th(d,t3);
    z:=(t1+Sqr(t2))/SQRT(t3);
    Write('z=',z:10:3);
  End.

```

### **Oldindan e'lon qilish**

Bizga ikki protsedura A va B berilgan bo'lib, A protsedura B protseduraga murojaat qilsin va A ta'rifi B ta'rifidan oldin kelsin. Masalan:

```

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

```

Bunday ta'rif xatolikka olib keladi. Chunki A protsedura hali ta'riflanmagan protseduraga murojaat qilmoqda. Bu holda B protsedurani quyidagicha oldindan e'lon qilish lozim:

```

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

```

Oldindan e'lon qilishda protsedura tanasi standart direktiva Forward bilan almashtiriladi.

### **Belgi va qatorlar bilan ishlashning maxsus funksiyalari**

Pascal tilida bir qancha maxsus protsedura va funksiyalar mavjud bo'lib, ular quyidagi guruhlariga bo'linadi:

- qatorni qayta ishlash;
- fayllar bilan ishlash;
- dinamik o'zgaruvchilar uchun xotirani boshqarish;
- arifmetik funksiyalar;
- ekran bilan ishlash.

Ularning ayrimlarini ko'rib chiqamiz:

Halt – dasturni bajarishdan to'xtatish;

Odd(i) – i-toq bo'lsa «True» aks holda «False» qiymat oladi;

Exit – bajarilayotgan blokdan chiqish;

Random – 0 dan 1 gacha bo'lgan sonni tasodifan olish;

Int(x) – sonning butun qismini olish;

Frac(x) – sonning kasr qismini olish;

Round(x) – berilgan sonni yaxlitlab butun olish;

GotoXY(x,y) – kursorni ko'rsatilgan joyga qo'yish;

ClrScr – ekranni tozalab, kursorni ekran boshiga qo'yish;

Trunc – argumentning butun qismi;

Str(I;Var S:String) – raqamni simvolga o'tkazish (I-ifoda yoki o'zgaruvchi);

Val(S:String; Var P;ko:Integer) – simvolni raqamga o'tkazish (P– o'zgaruvchi);

Length (S:String) – qator uzunligini aniqlash.

Pos(st l,st) – qatordagi qator qismi holatini aniqlash.

Misol: st:='Toshkent'; st l:='kent'; p:=pos(st l,st);

Javob: p=4. Agar izlanayotgan qator qism qatorda yo'q bo'lsa qiymat nolga teng bo'ladi.

Copy(st,m,n) – qatordan fragment kesib oladi.

Misol: st:='Toshkent'; p:=Copy(st,4,4);

Javob: p='kent'.

Delete(st,m,n) – qatordan fragment kesib olib tashlaydi.

Misol: st:='Toshkent'; p:=Delete(st,4,4);

Javob: p='Tosh'.

## **2.8. Modullar**

Turbo Pascal tizimida juda ko'p maxsus tayyor protsedura va funksiyalar mavjudki, ularning har qaysisi o'z vazifasiga ega bo'lib,

unga biblioteka modullari deyiladi. Har bir biblioteka funksiya va protseduralardan tashkil topgan bo'lib, ma'lum bir turdagi masalani yechishga mo'ljallangan.

Modul deb protsedura va funksiyaning alohida kompilyatsiya qilinib, maxsus .tpu kengaytmali fayl shaklida ifodalangan dasturiga aytiladi. Moduldan ixtiyoriy dastur ichida foydalanish mumkin. Moduldan foydalanish, ya'ni, uni aktivlashtirish uchun dasturning bosh qismiga quyidagilarni keltirish zarur:

**Uses <Modul 1 nomi, modul 2 nomi, . . . ,modul n nomi>;**

Misol:

Program SS;

Uses Crt,Graph;

Turbo Pascal tizimida har bir foydalanuvchi o'z modulini yaratishi uchun yaratiladigan modul tuzilmasini quyidagicha tashkil qilishi zarur.

**Unit <modul nomi>;**

**Interface**

.....

{Interfeys qism— ochiq (yozuvlar) qismi}

.....

**Implementation**

.....

{Yopiq (yozuvlar) qismi}

.....

**Begin**

.....

{Modulning asosiy qismi}

.....

**End.**

Bu yerda

**Unit** — modulning sarlavhasi;

**Interface** — modulning interfeysi, ya'ni, dastur va boshqa modullar uchun ochiq (ko'rinarli) qismining boshlanishini bildiradi. Bu qismda o'zgarmlar, kattaliklar tiplari, protsedura va funksiyalar aniqlanib ko'rsatilgan bo'ladi, lekin ularning butun ko'rinishi keyingi yopiq qismda beriladi.

**Implementation** — modulning dastur va modullar uchun yopiq, ya'ni ko'rinmaydigan qismining boshlanishini bildiradi. Bu yerda interfeys qismida aniqlangan protsedura va funksiyalar yana bir marta ko'rsatilishi shart (ularning sarlavhalari bir xil bo'lishi kerak).

Initsializatsiya qismi Begin yozuvidan keyin boshlanadi, agar bu qism mavjud bo'lmasa Begin ham bo'lmaydi. Bu qismda boshqaruvni asosiy programmaga o'tkazishga qadar bajarilishi kerak bo'lgan operatorlar ro'yxati joylashadi.

Misol tariqasida ikki sonning eng katta va eng kichigini topish modulini yaratish dasturini qaraymiz. Quyidagi dastur  $\text{Min}(x,y)$  va  $\text{Max}(x,y)$  funksiyalarini o'z ichiga olgan.

**Unit Study;**

**Interface {Interfeys qism}**

**Function  $\text{Min}(x,y:\text{Integer}):\text{Integer};$**

**Function  $\text{Max}(x,y:\text{Integer}):\text{Integer};$**

**Implementation {Yopiq qism}**

**Function  $\text{Min}(x,y:\text{Integer}):\text{Integer};$**

**Begin**

**If  $x \leq y$  Then  $\text{Min}:=x$  Else  $\text{Min}:=y$ ;**

**End;**

**Function  $\text{Max}(x,y:\text{Integer}):\text{Integer};$**

**Begin**

**If  $x \geq y$  Then  $\text{Max}:=x$  Else  $\text{Max}:=y$ ;**

**End;**

**{Initsializatsiya qismi mavjud emas}**

**End.**

Bu modul kompilyatsiya qilinib Study.tpu fayl nomiga ega bo'lishi kerak. Undan dasturda foydalanish uchun dastur bosh qismida Uses Study qatorini yozish kerak bo'ladi.

Turbo Pascal tizimida quyidagi biblioteka modullari mavjud:

System – standart protsedura va funksiyalarni o'z ichiga olgan bo'lib, bu modul avtomatik ravishda aktivlashtirilgan bo'ladi.

Dos – Ms Dos operatsion tizim imkoniyatlaridan foydalanuvchi protsedura va funksiyalarni o'z ichiga olgan.

Crt – monitor ekrani va klaviatura bilan ishlash imkoniyatini yaratuvchi protseduralar to'plamini o'z ichiga olgan.

Graph – har xil monitor videoadapterlarini qo'llagan holda kompyuter grafik imkoniyatlaridan foydalanuvchi ko'plab protseduralar to'plamini o'z ichiga oladi.

Printer – printer bilan ishlovchi kichik modul.

## **Dinamik bog'lanuvchi bibliotekalar (DLL)**

### ***Ta'rifi***

Dinamik bog'lanuvchi bibliotekalar dasturda boshqa tillarda yaratilgan protsedura va funksiyalardan foydalanishga imkon beradi.



Dinamik bibliotekalar bilan oddiy modullar orasida juda ko'p o'xshashliklar mavjud, lekin ikki jihatdan farq qiladi.

Birinchidan, dinamik bibliotekada e'lon qilingan o'zgaruvchilar va konstantalardan asosiy dasturda foydalanib bo'lmaydi.

Ikkinchidan, modullar statik usulda, ya'ni, kompilyatsiyaning komponovka bosqichida bog'lanadi. Dinamik bibliotekalar dinamik, ya'ni, dastur bajarilishi jarayonida bog'lanadi. Agarda ikki dastur oddiy modulga murojaat qilsa, shu modul ishlatilayotgan qismining ikki nusxasi xotirada yaratiladi. Dinamik bibliotekaning ikki dasturi ya'ni murojaat qilayotgan qismi faqat bir nusxada yaratiladi.

Dinamik bibliotekaning o'zgarishi dasturni qaytadan kompilyatsiya qilishga olib kelmaydi.

### ***Yaratilishi***

DLL yaratish uchun maxsus Library so'zi ishlatiladi.

DLL e'lonlar bo'limi Exports so'zidan boshlanib, eksport qilinayotgan qism qasturlar ro'yxatini o'z ichiga oladi:

**Library** MyLibrary;

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

**begin**

**end;**

**Procedure** MyProc;

**begin**

**end;**

**Exports**

**MyFunc, MyProc;**

**begin**

**end.**

Qism dasturdan avval nomidan tashqari DLLga uning tartib nomeri joylashtiriladi: birinchi qism dastur nomeri 0, keyingisi – 1 va hokazo. Dasturchi bu indeksatsiyani o'zgartirishi va 0 dan 32767 gacha nomer qo'yishi mumkin:

**Exports**

MyFunc index 1, MyProc index 2;

Dasturchi eksport qilinayotgan qism dasturi uchun tashqi nom berishi mumkin:

**Exports**

MyFunc index I name 'NEWFUNC';

Chaqirayotgan dastur eksport qilinayotgan qism dasturini tashqi nomi yoki indeksi bo'yicha chaqirilishi mumkin.

**Misol:**

Misol tariqasida cplx modulini ko'ramiz.

## Library Cmplx;

uses

SysUtils, Classes;

{*SR \*.RES*}

### Type

TComplex = **record** Re, Im: Real;

**end;**

**function** AddC(x, y: TComplex): TComplex; **stdcall;**

**begin**

Result.Im := x.Im + y.Im;

Result.Re := x.Re + y.Re **end;**

**function** SubC(x, y: TComplex): TComplex;

**stdcall;**

**begin**

Result.Im := x.Im - y.Im;

Result.Re := x.Re - y.Re

**end;**

**function** MulC(x, u: TComplex): TComplex;

**stdcall;**

**begin**

Result.Re := x.Re \* y.Re + x.Im \* y.Im;

Result.Im := x.Re \* y.Im - x.Im \* y.Re

**end;**

**function** DivC(x, y: TComplex): TComplex;

**stdcall;**

**var**

z: Real;

**begin**

z :=  $\sqrt{y.Re} + \sqrt{y.Im}$ ;

**try**

Result.Re :=  $(x.Re * y.Re + x.Im * y.Im)/z$ ;

Result.Im :=  $(x.Re * y.Im - x.Im * y.Re)/z$

**except**

Result.Re :=  $1e+309$ ;

Result.Im :=  $1e+309$

**end**

**end;**

### Exports

AddC index 1 name `ADDC` resident,

SubC index 2,

MulC index 3,

DivC index 4;

**begin**

**end.**

Bu yerda stdcall bu DLL ga Delphi dan boshqa tillardan murojaat qilishga imkon beradi. Agar bibliotekaga faqat Delphi tilidagi dasturdan murojaat qilinsa, bu soʻzning keragi yoʻq.

Qism dasturlariga murojaat qilish uchun ularni quyidagicha eʼlon qilish lozim:

**Procedure** MyProc; **External** `MyDLL`;

Agar indeks boʻyicha chaqirish lozim boʻlsa:

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

Bundan tashqari dasturchi tashqi nomni oʻzgartirishi ham mumkin:

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

**Foydalanish**

Statik yuklash

Dasturda Cmpix bibliotekasi quyidagicha eʼlon qilinishi lozim.

**Type**

TComplex = **record** Re, Im: Real;

**end;**

**function** ADDC(x, y: TComplex): TComplex; **stdcall**; **External** `Cmplx`;

**function** SubC(x, y: TComplex): TComplex; **stdcall**; **External** `Cmplx`;

**function** MulC(x, y: TComplex): TComplex; **stdcall**; **External** `Cmplx`;

**function** DivC(x, y: TComplex): TComplex; **stdcall**; **External** `Cmplx`;

Interfeysli modul

DLL – qism dasturlarini chaqirishda yozuv kabi murakkab turdagi maʼlumotlarni uzatishga toʻgʻri kelishi mumkin. Agar biror DLL ga koʻp murojaat qilinsa, murakkab tur eʼlon qilingan interfeysli moduldan foydalanish qulaydir. Masalan:

**Unit** Cmplx;

**Interface**

**type**

TComplex = **record** Re, Im: Real;

**end;**

**function** AddC(x, y: TComplex): TComplex; **stdcall**;  
**External** `Cmplx` index 1;

**function** SubC(x, y: TComplex): TComplex; **stdcall**;  
**External** `Cmplx` index 2;

**function** MulC(x, y: TComplex): TComplex; **stdcall**;

**External** 'Cmplx' index 3;

**function** DivC(x, y: TComplex): TComplex; **stdcall**;

**External** 'Cmplx' index 4;

**Implementation end.**

Boshqa tilda yozilgan DLL protseduralariga murojaat qilinganda ularning tashqi nomi Delphi qoidasiga to'g'ri kelmasligi mumkin. Masalan C++ tili identifikatorlarda «@» simvolidan foydalanishga ruxsat beradi. Bu holda Delphi qoidasiga mos nom berib «name» so'zidan so'ng asl nomini ko'rsatish kerak. Masalan:

**function MyFunction: WordBool; stdcall;**

**external** 'MyDLL' name '\_MyFunction@12'

## 2.9. Fayllar bilan ishlash funksiyasi va protseduralari

Kiritiladigan va chiqariladigan ma'lumotlar soni ko'p miqdorda bo'lsa, ularni faylda saqlash dasturda qulaylik tug'diradi. Bu ma'lumotlar oddiy matn (tekst) fayllarida saqlanadi. Fayl o'zgaruvchisi dastur bosh qismida e'lon qilinadi, ya'ni:

**Type f=text;**

**Var**

**fx:f;**

bu yerda f –fayl tipi, oddiy matn faylni bildiradi;

fx–fayl o'zgaruvchisi.

Kerakli fayldan ma'lumotlarni o'qishga tayyorlash uchun Assign va Reset funksiyalari ishlatiladi.

**Assign**–fayl o'zgaruvchisi bilan asosiy fayl orasida aloqa o'rnatadi.

Assign (fx, 'c:\a\fl.txt');

**Reset** – faylni topib uni ishga tayyorlaydi. **Reset (fx);**

**Bu yerda fx – fayl o'zgaruvchisi;**

'c:\a\fl.txt'-c: diskning A katalogidagi fx.txt

fayldan o'qishni bildiradi.

Fayldagi ma'lumotlarni o'qish uchun Read funksiyasi ishlatiladi.

Read (<fayl o'zgaruvchisi>, <o'zgaruvchilar, massivlar>);

Misol: Read (fx, x,y,z,A[i],B[I]);

Fayldan o'zgaruvchilar yoki massivlar qiymatlarini o'qib bo'lgandan keyin fayl yopiladi. Faylni yopish quyidagi funksiya yordamida bajariladi. Close (fayl o'zgaruvchisi);

Misol: Close (fx);

**Misol 1.** C: diskning AA katalogidagi AB fayli ma'lumotlarini o'qib A va B massivlarga joylashtiring. Fayl ma'lumotlarining tuzilmasi quyidagicha:

15.2	20.5
20.1	25.5
20.2	50.2
26.8	52.4
.....	.....

Ya'ni fayl tuzilmasi ikki ustundan iborat ma'lumotlar to'plamidan iborat.

```

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

```

Massiv qiymatlarini biron matn fayliga yozib qo'yish uchun Assign, Rewrite, Append, Write va Close funksiyalari ishlatiladi.

**Assign** – fayl o'zgaruvchisi bilan asosiy fayl o'rtasida aloqa o'ratadi va u quyidagicha yoziladi.

Assign (fayl o'zgaruvchisi, diskdagi fayl joyi va nomi);

**Append** – Fayldan yozish uchun joy tayyorlaydi.

Append (fayl o'zgaruvchisi);

**Write** – o'zgaruvchi qiymatini fayl o'zgaruvchisiga uzatadi va faylga yozadi. Write (fayl o'zgaruvchisi, o'zgaruvchilar);

**Close** – ochilgan faylni yopadi. Close (fayl o'zgaruvchisi);

**Misol 2.** C: diskning AA katalogidagi AB fayli ma'lumotlarini A va B massivlarda o'qib, shu massivlarga mos elementlarni qo'shib C massivini tashkil qiling va A,B,C massivlarini ABC nomli faylga joylashtiring.

Fayl ma'lumotlari tuzilmasi quyidagicha bo'lsin:

i	A	B	C
1	15.2	20.5	35.7
2	20.1	25.5	45.6

.....

```

Program FABS;
  Type

```

```

f=text;
Var
  A,B,C: Array[1..100] of Real; i,m: Integer;
  fax: f;
Begin
  Assign(fax,'c:\AA\AB.txt'); Reset(fax);
  m:=0;
  While not eof(fax) do
    Begin m:=m+1; Readln(fax,A[m],B[m]);
End;
    Close (fax);
  Assign(fax,'c:\AA\ABC.txt'); Rewrite(fax);
  Append(fax);
  For i:=to m do
    Begin Write(fax,i,A[i],b[i],c[i]); Writeln(fax);
End;
  Close(fax);
End.

```

### **S a v o l l a r**

- 1.O'zgaruvchilar qiymatini ekrandan kiritish operatorini tushuntiring.
- 2.Ma'lumotlar va o'zgaruvchilar qiymatini ekranga chiqarish operatorini tushuntiring.
- 3.Tanlash operatorini tushuntiring.
- 4.Takrorlanishlar soni jihatidan while va repeat – until operatorlari qanday farq qiladi?
- 5.Massivlar qanday tavsiflanadi?
- 6.Massivni tartiblashtirishning qanday usullari mavjud?
- 7.Protsedura va funksiyalar dasturda qanday hollarda ishlatiladi?
- 8.Protsedura va funksiya qanday tavsiflanadi?
- 9.Funksiya protseduradan nima bilan farqlanadi?
- 10.Qanday maxsus protseduralar va funksiyalarni bilasiz?
- 11.Ma'lumotlarni faylga yozish da qanday funksiya va protseduralar ishlatiladi?

### **M a s h q l a r**

- 1.Dastur bo'lagi bajarilishida A o'zgaruvchi qiymatini aniqlang:
  - a) B:=5; C:=4; A:=(B/5+C)\*3;
  - b) B:=5; A:=B; A:=A+B;
  - c) A:=0; A:=A+1; A:=A\*A;
  - d) B:=5;A:=10; C:=-B; A:=C;

2. Quyidagi ifodani Pascal tilida yozing:

$$a = \frac{x^{y+1} + e^{z-1}}{x + |y - Ln Z|}$$

3.  $(15 \geq 25)$  Or  $(6.3 < 12.4)$  va  $\text{Not}(5 > 12)$  And  $(2.3 < 3.4)$  mantiqiy ifodalar qiymatini aniqlang.

4. Quyidagi F va D mantiqiy o'zgaruvchilar qiymatini aniqlang:

A:=1; B:=A+3; C:=B+A; F:=(A<B) AND (A>C); D:=(A<C)OR(B>C);

5. Quyidagi ifodalar o'zgaruvchi A ning qanday qiymatlarida True qiymatini qabul qiladi.

a)  $(A \geq 100)$  AND  $(A \leq 150)$

b)  $(A \leq 100)$  AND  $(A < 25)$

c)  $(A = 5)$  OR  $((A > 10)$  AND  $((A < 1))$

6. Quyidagi dastur bo'lagi natijasini aniqlang:

X:=0; Y:=1; If X>Y then begin X:=X-10; Y:=Y+10 end;  
WriteLn('X=', X, 'Y=', Y);

7. Quyidagi dasturni while operatori o'rniga repeat – until operatorini ishlatib yozing: n:= 10; m:= 2;

while m <= n do m:= m+3;

8. Quyidagi dastur bo'lagida «Begin» va «End» orasidagi operatorlar necha marta takrorlanadi:

a) For i:=j to j+1 do

Begin

...

End;

b) k:=0;

For i:=2 Downto k do

Begin

...

End;

9. Ikki son kattasini topish dasturi tuzilsin:

a) protsedurani ishlatgan holda;

b) funksiyani ishlatgan holda.

10. Bir o'lchovli massivning eng kichik elementini topish dasturi tuzilsin:

a) protsedurani ishlatgan holda;

b) funksiyani ishlatgan holda.

11. Quyidagi dasturdan ADD funksiyasi qiymatini toping:

x:=7; y:=ADD(x+1); Write('y=', y:2);

12. Quyidagi dasturdan Frac funksiyasi qiymatini toping:

x:=14.62; y:=Frac(x+2); Write('y=', y:2);

13. Quyidagi dasturdan Int funksiyasi qiymatini toping:

x:=14.62; y:=Int(x+2); Write('y=', y:2);

## Javoblar

1. a) 15; b) 10; c) 1; d) -5
2.  $A:=(\text{EXP}((Y+1)*\text{LN}(X))+\text{EXP}(Y-1))/(x+\text{ABS}(Y-\text{LN}(Z)))$ ;
3. TRUE va TRUE
4. F:=FALSE va D:=TRUE
5. a) 100 dan 150 gacha b) 25dan kichik c) 5
6. X=0 Y=1
7. n:= 10; m:= 2;  
Repeat m:= m+3; Until m >n;
8. a) Ikki marta b) Uch marta
- 9.a)

```
Program Proc;  
  Var a,b,c: Real;  
  Procedure Maximum(x,y: Real; Var z: real);  
    Begin if x>y Then z:=x Else z:=y End;  
  Begin  
    Read (a,b);  
    Maximum(a,b,c); Write (a,b,c);  
  End.
```

9.b)

```
Program Func;  
  Var a,b,c: Real;  
  Function Maximum(x,y: Real): real;  
    Begin if x>y Then maximum:=x Else maximum:=y End;  
  Begin  
    Read (a,b);  
    c:=Maximum(a,b); Writeln (a,b,c);  
  End.
```

10.a)

```
Program ProcMasMin;  
  Type  
    Mas: Array[1..100] of real;  
  Var  
    a: mas; c: Real; i,n: Integer;  
    Procedure MasMin(n: Integer; x: mas; Var z: Real);  
      Var i: Integer;  
      Begin z:=x[1]; For i:=2 to n do if z>x[i] Then z:=x[i] End;  
  Begin  
    Writeln ('Massiv o'lchamini kiriting:'); Readln(n);  
    Writeln (n:2,'ta massiv elementi qiymatini kiriting:');  
    For i:=1 to n do Read (a[i]);  
    MasMin(n,a,c); Writeln (c);  
  End.
```



```

10.b)
Program FuncMasMin;
Type
  Mas=Array[1..100] of real;
Var
  a: mas; c: Real; i,n: Integer;
Function MasMin(n:Integer;x:mas): real;
  Var i: Integer; z: Real;
  Begin z:=x[1]; For i:=2 to n do if z>x[i] Then z:=x[i]; MasMin:=z
End;
Begin
  Writeln ('Massiv o'lchamini kiriting:'); Readln(n);
  Writeln (n:2, '-ta massiv elementi qiymatini kiriting:');
  For i:=1 to n do Read (a[i]);
  c:=MasMin(n,a); Writeln (c);
End.

11. False
12. 62
13. 16

```

### **III.DELPHI VIZUAL DASTURLASH MUHITI HAQIDA ASOSIY TUSHUNCHALAR**

#### **3.1.Delphini dasturlash muhiti**

Delphi –Windows operatsion tizimida dastur yaratishga yo‘naltirilgan dasturlash muhitidir. Delphida dastur tuzish zamonaviy vizual loyihalash texnologiyalariga asoslangan bo‘lib, unda dasturlashning obyektga mo‘ljallangan g‘oyasi mujassamlashgan. Delphida dastur Turbo Pascal dasturlash tilining rivoji bo‘lgan Object Pascal tilida yoziladi.

Delphi – bir necha muhim ahamiyatga ega bo‘lgan texnologiyalar kombinatsiyasini o‘zida mujassam etgan:

- yuqori darajadagi mashinali kodda tuzilgan kompilyator;
- obyektga mo‘ljallangan komponentalar modullari;
- dastur ilovalarini vizual tuzish;
- ma‘lumotlar bazasini tuzish uchun yuqori masshtabli vosita.

Delphi – Windows muhitida ishlaydigan dastur tuzish uchun qulay bo‘lgan vosita bo‘lib, kompyuterda dastur yaratish ishlarini avtomatlashtiradi, xatoliklarni kamaytiradi va dastur tuzuvchi mehnatini yengillashtiradi. Delphida dastur zamonaviy vizual loyihalash texnologiyasi asosida obyektga mo‘ljallangan dasturlash nazariyasini hisob-

ga olgan holda tuziladi. Delphi tizimi Turbo Pascal 7.0. tilining rivoji bo'lgan obyektga mo'ljallangan Object Pascal dasturlash tilini ishlatadi.

Ma'lumki, dastur tuzish sermashaqqat jarayon, lekin Delphi tizimi bu ishni sezilarli darajada soddalashtiradi va masala turiga qarab dastur tuzuvchi ishining 50–80% ni tizimga yuklaydi. Delphi tizimi dasturni loyihalash va yaratish vaqtini kamaytiradi, hamda Windows muhitida ishlovchi dastur ilovalarini tuzish jarayonini osonlashtiradi.

Delphi o'zida bir qancha zamonaviy ma'lumotlar bazasini boshqarish tizimlarini va dasturlash texnologiyalarini ma'lumotlar bazasini yaratishda ishlatadi.

### 3.2. Delphi tizimining oynasi va uning elementlari

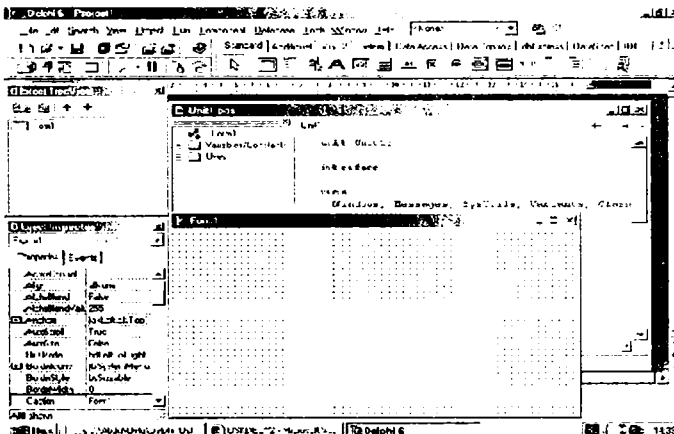
Delphi tizimida ishni boshlash uchun uni dasturlar menyusidan topib ishga tushiramiz.

Пуск=>Программы=>Borland Delphi=>Delphi

Delphi oynasi ko'rinishi odatdagidan ancha boshqacharoq bo'lib, u o'z ichiga beshta oynani oladi:

- bosh oyna – Delphi Project I;
- forma oynasi – Form I;
- obyekt xossalarini tahrirlash oynasi – Object Inspector;
- obyektlar ro'yxatini ko'rish oynasi – Object tree View;
- dastur kodlarini tahrirlash oynasi – Unit.pas.

Bosh oyna ekranning yuqori qismida joylashgan bo'lib, uning birinchi qatorida sarlavha, ya'ni, proyektning nomi joylashgan. Ikkinchi qatorda buyruqlar menyusi gorizontol ko'rinishda joylashgan. Keyingi qatorning chap tarafida uskunalar paneli va o'ng tarafida komponentalar politrasi joylashgan.



Buyruqlar menyusi quyidagilarni o'z ichiga olgan:

– File (fayl) bo'limi fayllar ustida ish bajarish uchun kerakli buyruqlarni o'z ichiga olgan;

– Edit (tahrir) bo'limi fayl ichidagi ma'lumotlarni tahrirlash uchun kerakli buyruqlarni o'z ichiga olgan:

– Seerch;

– View;

– Compile;

– Run formani ishga tushirish;

– Options;

– Tols servis xizmatidan foydalanish;

– Help yordam chaqirish.

Forma oynasida ilovalar yaratiladi. Object Inspector oynasi obyekt xossalarini tahrirlash uchun xizmat qiladi. Obyekt xossalari bu – obyektga berilgan xarakteristika bo'lib, uning ko'rinishi, joylashishi va holatidir. Masalan, Width va Height xossalari forma o'lchamini, top va Lift esa formaning ekrandagi holati, Caption – sarlavha matnini aniqlaydi.

Vizual dasturlash texnologiyasida obyekt deganda muloqot oynasi va boshqarish elementlari (kiritish va chiqarish maydoni, buyruq tugmalari, pereklyuchatellar va boshqa) tushuniladi.

Delphida dasturlash ikkita o'zaro ta'sir etuvchi bir-biri bilan bog'liq jarayon asosida tashkil qilinadi:

– dasturni vizual loyihalash jarayoni;

– dastur kodlarini kiritish (yozish) jarayoni.

Kodlarni yozish uchun maxsus kod oynasi mavjud bo'lib, u dastur matnini kiritish va tahrirlash uchun mo'ljallangandir. Bu kodlarni yozish oynasida dasturlash Pascal tilining rivoji bo'lgan va kengaytirilgan Object Pascal tilida tuziladi.

Kodlarni yozish oynasi boshlanishda o'z ichiga holi bo'sh formani akslantiruvchi dastur matnini yozib chiqaradi. Dastur loyihasini ishlash mobaynida dasturchi kerakli dastur operatorlarini kiritib, formani loyiha bo'yicha akslantiradi. Delphida dasturlash forma oynasini tashkil etishdan boshlanadi.

Oddiy dastur ilovasini yaratish ketma-ket File=> New=> Applisation buyrug'ini berish bilan boshlanadi. Bu buyruqni berishdan oldin ikkita asosiy ishni bajarish lozim:

– papka tashkil etish;

– tizimni to'g'rilash.

Papka tuzing, masalan, **My\_Delhp** nomli. **My\_Delhp** papkasi ichida yana o'z dasturingizni saqlash uchun papka ochish, masalan, Pom\_1.

Delphi muhitining standart nastroykasiga o'zgartirish kiritish uchun Tols=>Environment Options menyu buyrug'ini berish va muloqot darchasidan kerakli o'zgarishlarni bajarish lozim.

Delphi dasturlash muhitida ishlash jarayonida quyidagi kengaytmali fayllar ishlatiladi:

- loyiha fayli, kengaytmasi **.dpr**;
- paskal moduli fayli, kengaytmasi **.pas**;
- komponentalar joylashgan fayl, kengaytmasi **.dcu**;
- formalar joylashgan fayl, kengaytmasi **.dfm**;
- ma'lumotlar bazasi fayli, kengaytmasi **.dbf**.

Tayyorlanadigan Delphi dastur uchta asosiy etapdan o'tadi:

- kompilyatsiya;
- komponovka;
- bajarish.

**Kompilyatsiya** etapida tayyorlangan dastur matni Object Pascal tiliga o'tkaziladi. **Kompanovka** bosqichida esa kerakli qo'shimcha yordamchi dasturlar va ostdasturlar unga birlashtiriladi. F9 tugmasini bosish bilan Save UnitAs dialog oynasi paydo bo'ladi va sizdan Unit.pas moduli uchun fayl nomini va joylashadigan papkani ko'rsatishingizni so'raydi. Agar joyi ko'rsatilmasa Delphi avtomatik ravishda dasturingizni Bin papkasiga joylashtiradi. Yaxshisi siz bu papkani o'z ishchi papkangiz nomiga almashtiring, masalan, My\_Delph. Dastur kompilyatsiya qilinishi paytida Delphi sistemasi pas, dfm va dcu kengaytmali modullar tuzadi. .pas kengaytmali fayl kodlarni yozish oynasiga kiritilgan dastur matnini, .dfm forma oynasi tashkil etuvchilarini, .dcu kengaytmali fayl esa .pas va .dfm kengaytmali fayllarning birgalikdagi mashina kodiga o'tkazilgan variantini saqlaydi. Bu .dcu kengaytmali fayl kompilyator tomonidan tashkil qilinadi va yagona ishchi (bajariluvchi) .exe kengaytmali fayl tashkil qilishga baza yaratadi.

### 3.3.Delphi loyihasining tuzilmasi

Delphi dasturi – bu bir necha bir-biri bilan bog'liq fayllardir. Har qanday dastur .dpr kengaytmali loyiha fayli va bir yoki bir necha .pas kengaytmali modullardan tashkil topadi. Loyiha fayli dasturchi tomonidan kiritilmaydi, u foydalanuvchining ko'rsatmalari asosida avtomatik ravishda Delphi sistemali dasturi tomonidan tuziladi. Loyiha fayli matnini ko'rish uchun Project/View Source buyrug'ini berishi zarur. Loyiha matni umumiy holda quyidagicha bo'lishi mumkin:

**Program Project1;**  
**Uses**

**Forms,  
Unit1 in 'Unit1.pas' {Form1}  
{SR \*.res}**

**Begin**

**Application.Initialize;  
Application.CreateForm(TForm1, Form1);  
Application.Run;**

**End.**

Loyiha nomi dasturchi tomonidan loyiha faylini saqlash vaqtida beriladi va u Delphi muhitida bajariluvchi fayl, ya'ni, kengaytmasi .exe bo'lgan faylni tashkil qilishni aniqlaydi. Loyiha faylidan keyin ishlatiladigan modullar: standart modullar Forms va Unit1 joylashadi. {SR \*.res} direktivasi kompilyatorga ishlatilishi kerak bo'lgan resurs fayllari, masalan dasturlarni e'lon qilish kerakligini bildiradi. Yulduzcha belgisi resurs faylining kengaytmasi .res ekanligini bildiradi. Bosh modulning bajariluvchi qismi Begin .. End operatorlari orasiga joylashadi.

Modul – bu, biror-bir dastur. Modullar standart konstruksiyasiga ega. Object Pascalda modul tuzilmasi umumiy holda quyidagi ko'rinishda bo'ladi:

**Unit <Modul nomi>**

**Interface**

.....

**Implementation**

.....

**Initialization**

.....

**Finalization**

.....

**End.**

Delphi tizimini ishga tushirgandan keyin modul tuzilmasi quyidagi ko'rinishda bo'ladi:

**Unit unit1;**

**Interface**

**Uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms, Dialogs;**

**Type**

**TForm1 = class(TForm)**

**Private**

**{ Private declarations }**

**Public**

```
    { Public declarations }  
end;  
Var  
    Form1: TForm1;  
Implementation  
    {$R *.dfm}  
End.
```

### 3.4. Sinflar va obyektlar

Object Pascal obyektga mo'ljallangan dasturlash tilidir. Obyektga mo'ljallangan tilda yig'ilgan imkoniyatlarga dasturlash tilining obyekt modellari deyiladi. Object Pascalda obyekt modellarini ishlatilishining amaliy natijasi komponentalarni yaratish va ularni qo'llab-quvvatlashdir.

Ma'lumki, Delphi dasturlash vositasi Object Pascal obyektga yo'naltirilgan dasturlash tilini ishlatadi. Obyektga yo'naltirilgan dasturlash (OYD) – bu dastur ishlab chiqish usullari bo'lib, uning asosida real dunyo obyekt va uning holatini ifodalovchi ma'lum tuzilmaga ega obyekt tushunchasi yotadi. Object Pascalda obyekt modelining qo'llanilish natijasi bu komponentalarni qo'llash va yaratishdir. Object Pascal obyekt modelining asosi sinf va obyekt tushunchalaridir.

#### Sinf

**Sinf** – bu Object Pascalda maxsus turlar bo'lib, o'zida maydon, usullar va xossalarni mujassamlashtiradi.

Pascal sinfiy tili dasturchiga o'zining murakkab ma'lumotlar turlari – yozuvlar (records) kiritishiga imkon beradi. Obyektli dasturlash konsepsiyasiga asoslangan Delphi tili sinflar kiritishiga imkon beradi.

Sinf murakkab tuzilma bo'lib, ma'lumotlar ta'riflaridan tashqari, protsedura va funksiyalar ta'riflarini o'z ichiga oladi.

Sodda sinf ta'rifiga misol:

```
TPerson = class
```

```
private
```

```
fname: string[15]; faddress: string[35];
```

```
public
```

```
procedure Show;
```

```
end;
```

Sinf ma'lumotlari maydonlar, protsedura va funksiyalar usullari deb ataladi.

Keltirilgan misolda TPerson — sinf nomi, fname va faddress — maydonlar nomlari, show — usul nomi.

**Maydon** — bu sinfga birlashtirilgan ma'lumotlardir. Sinfga qarashli maydonlar oddiy yozuv maydoni kabi bo'lib, ularning farqi har xil turda bo'lishidir. Masalan:

```
Type  
TchildClass=Class  
Fone: Integer;  
Ftwo: String;  
Fthree: Tobject;  
End;
```

Maydonlarga murojaat qilish sinf xossalari va usullari yordamida amalga oshiriladi. Maydonga murojaat qilish uchun oldin sinf nomi yozilib, keyin ajratuvchi nuqta qo'yilib, maydon nomi yoziladi. Masalan:

```
Var  
MyObject: TchildClass;  
Begin  
MyObject.Fone:=16;  
MyObject.Ftwo:='qator qiymati';  
End;
```

Maydon nomi unga mos xossa nomining birinchi harfi «F» bo'lishi bilan farqlanadi.

Delphida qabul qilingan kelishuv bo'yicha maydon nomlari f (field — maydon so'zidan) harfidan boshlanishi lozim.

Sinf ta'rifi dasturda turlar ta'rifi bo'limiga joylashtiriladi (type).

**Usullar** — sinfga birlashtirilgan protsedura va funksiyalarga usullar deyiladi. Masalan:

```
Type  
TchildClass=Class  
Fore: Integer;  
Ftwo: String;  
Fthree: Tobject;  
Function FirstFunc(x:Real):Real;  
Procedure SecondProc;  
End;
```

Sinf usullari (sinf ta'rifiga kiritilgan protsedura va funksiyalar) sinf obyektlari ustida amal bajaradi. Usul bajarilishi uchun obyekt nomi va nuqtadan usul nomi ko'rsatilishi lozim. Masalan:  
professor. Show;

Sinf usuli ta'riflanganda sinf nomi va usul nomi ko'rsatiladi.

Masalan:

```
// TPerson sinfi Show usuli
```

```
procedure TPerson.Show;
```

```
begin
```

```
Write ('Nom:' + fname + #13+ 'Adres:' + faddress );
```

```
end;
```

Usul tanasida obyekt maydonlariga murojaat qilinganda obyekt nomi ko'rsatilmaydi.

Usulga murojaat qilish dasturda uning nomini ko'rsatish bilan bajariladi. Masalan:

```
Var
```

```
MyObject: TchildClass;
```

```
y: Real;
```

```
Begin
```

```
.....
```

```
MyObject. SecondProc;
```

```
y:=MyObject.FirstFunc(3.14);
```

```
End;
```

Sinfda aniqlangan usullarni statistik, virtual (Virtual), dinamik yoki abstrakt turlarga bo'lish mumkin. Agar usul turi ko'rsatilmasa, u avtomatik ravishda statistik turni oladi. Masalan:

```
Type
```

```
TBase=Class
```

```
Procedure MyJoy; Virtual;
```

```
End;
```

```
Tdescendant=Class(TBase)
```

```
Procedure MyJoy; Override;
```

```
End;
```

```
Var
```

```
FirstObject: TBase;
```

```
SecondObject: TDescendant;
```

```
Begin
```

```
.....
```

```
FirstObject.MyJoy;
```

```
SecondObject.MyJoy;
```

```
.....
```

```
End;
```

Sinf davomchisida ishlatiladigan usul uchun Override kalit so'zi ko'rsatilishi lozim.



Agar Tbase sinfida MyJoy usuli dinamik bo'lsa, Virtual so'zi Dynamic so'ziga almashtiriladi. Ularning asosiy farqi murojaat qilinganda Virtual usuli vaqt jihatdan ancha effektiv bo'lsa, Dynamic usuli esa operativ xotiradan ratsional foydalanish imkonini beradi.

## Obyekt

**Obyekt** — bu sinfning real nusxasi bo'lib, ma'lumotlar va funksiyalardan tashkil topadi. U dasturning Var bo'limida e'lon qilinadi.

Obyektlar sinflar vakillari sifatida dasturning Var bo'limida ta'riflanadi. Masalan:

### Var

student: TPerson; professor: TPerson;

Delphida obyekt — bu dinamik tuzilmadir. O'zgaruvchi—obyekt ma'lumotlarni emas, obyekt ma'lumotlaridagi ilovani o'z ichiga oladi. Shuning uchun dasturchi bu ma'lumotlarga xotiradan joy ajratishni ko'zda tutishi lozim.

Joy ajratishning sinfdagi maxsus usuli — konstruktor yordamida amalga oshiriladi. Bu usul odatda Create (yaratish) nomiga ega bo'ladi. Sinf ta'rifida konstruktor uchun «procedure» so'zi o'rniga «constructor» so'zi ishlatiladi.

Quyida tarkibida konstruktor qatnashgan TPerson sinfi ta'rifi keltirilgan:

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

Xotiradan joy ajratish konstruktor sinfga qo'llash natijasini qiymat sifatida berish orqali amalga oshiriladi. Misol uchun:

```
professor := TPerson.Create;
```

instruksiyasi bajarilishi natijasida professor obyektiga xotiradan joy ajratiladi. Xotiradan joy ajratishdan tashqari konstruktor, odatda, obyekt maydonlariga boshlang'ich qiymatlar berish, ya'ni, obyekt initsializatsiyasi vazifasini ham bajaradi. Quyida TPerson obyekt uchun konstruktor misoli keltirilgan:

```
Constructor TPerson.Create;  
begin  
fname := "";  
faddress := "";  
end;
```

Obyekt maydoniga murojaat qilish uchun obyekt nomi va nuqtadan so'ng maydon nomi ko'rsatiladi. Masalan:

professor.fname

Obyektga ajratilgan xotira qismini bo'shatish uchun maxsus usul – destruktor Free ishlatiladi. Masalan: professor.Free.

## **Inkapsulatsiya va obyekt xossalari**

**Inkapsulatsiya** – bu sinfga birlashtirilgan qayta ishlash uchun mo'ljallangan ma'lumotlar va qism dasturlaridir. Sinf maydonlari ma'lumotlarni o'z ichiga oladi. Bu ma'lumotlarni qayta ishlashda qo'llaniladigan protsedura va funksiyalarga usullar deyiladi. Obyektga mo'ljallangan dasturlashda sinf maydonlariga to'g'ridan-to'g'ri murojaat ruxsat etilmaydi. Shu sababli Object Pascalda «xossa» (свойства) deb ataluvchi maxsus konstruktsiya qaraladi. U mos usulni chaqirish yordamida maydondan o'qish yoki yozishni amalga oshiradi.

Misol tariqasida Delphida aniqlangan standart sinflarni keltirish mumkin:

TEdit – formaga qatorlarni kiritish ishlarini boshqarish va tashkil qilishni bajaradi.

TLabel – formaga belgilarni kiritish ishlarini boshqarish va tashkil qilishni bajaradi.

TButton – formaga joylashtirilgan tugmacha yordamida dasturchi tomonidan kiritilgan dastur kodlarining bajarilishini ta'minlaydi.

Sinf nomlarining bosh harfi «T» harfi bilan boshlanishi qabul qilingan.

Inkapsulatsiya deyilganda obyekt maydonlariga to'g'ridan-to'g'ri emas, faqat sinf usullari orqali murojaat qilishga aytiladi.

Delphi tilida obyekt maydonlariga murojaat obyekt xossalari orqali amalga oshiriladi. Obyekt xossasiga murojaat qilish uchun ikki usuldan foydalaniladi. Xossa tashqaridan sinf maydonini anglatsa, ayni holda, u maydondan foydalanish imkonini beruvchi boshqarish mexanizmidir. Xossa qandaydir sinf maydonlari bilan bog'liq bo'lgan o'qishda va yozishda ishlatilishi kerak bo'lgan sinf usullarini ko'rsatadi. O'qish uchun ishlatiladigan funksiya nomi Get bo'lib, unga mos xossa nomi qo'shib yoziladi. Yozish uchun ishlatiladigan usul bitta parametrli Set nomli qism dastur bo'lib, uning nomiga ham mos xossa nomi qo'shib yoziladi. O'qish va yozish usullari va uning parametri ham bir xil xossaga ega bo'lishi lozim. Xossani e'lon qilish uchun Property, Read va Write so'zlari ishlatiladi. Read va Write usul nomlari bo'lib, ular mos ravishda o'qish va yozish uchun mo'ljallangan. Masalan:

## Type

**TStudent=Class**

**Fage: Integer;**

**Function GetAge: Integer;**

**Procedure GetAge(Value:Integer);**

**Property Age: Integer Read GetAge Write SetAge;**

**End;**

Bu yerda Age – FAge maydoni hamda GetAge va SetAge usullari bilan bog‘liq xossa bo‘lib, Fage maydonida o‘qish yoki yozish uchun xizmat qiladi.

Dastur matnida xossaga murojaat qilish usuli va maydoni uchun qanday bo‘lsa xuddi shunday obyekt nomi, nuqta va xossa nomi yozilishi bilan ishlatiladi. Masalan:

## Var

**GoodStudent: TStudent;**

**HisAge: Integer;**

**Begin**

**GoodStudent:=Tstudent.Create;**

**GoodStudent.Age:=19;**

.....

**HisAge:=GoodStudent.Age;**

.....

**GoodStudent.Free;**

**End;**

Xossa qiymatini o‘rnatish – xossani yozish (write) usuli, xossa qiymatini olish, xossani o‘qish (read) usuli deb ataladi.

Sinf ta’rifida xossa nomidan oldin property (xossa) so‘zi yoziladi. Xossa nomidan so‘ng uning turi ko‘rsatiladi, read so‘zidan so‘ng xossani o‘qishni ta’minlovchi usul, write – so‘zidan so‘ng xossani yozishni ta’minlovchi usul nomi yoziladi.

Quyida ikki Name va Address xossalarini o‘z ichiga oluvchi TPerson sinfi ta’rifi keltirilgan:

## Type

TName = **string**[15]; TAddress = **string**[35];

TPerson = class

**private**

FName: TName;

FAddress: TAddress;

**Constructor** Create (Name:Tname);

**Procedure** Show;

**Function** GetName: TName;

```

Function GetAddress: TAddress;
Procedure SetAddress (NewAddress:TAddress);
public
Property Name: Tname
read GetName;
Property Address: TAddress
read GetAddress
write SetAddress; end;

```

Dasturda student obyektining Address xossasiga qiymat berishini quyidagicha yozish mumkin:

```
student.Address := `Toshkent, Yunusobod 21, kv.3`.
```

**Vorislik** – har qanday sinf boshqa sinf asosida yaratilishi mumkinligini bildiradi.

Bosh sinfdan yangi bir sinf yaratish quyidagi dastur kodi yordamida bajariladi.

```
TnewClass=Class(TotolClass);
```

Bu yerda TotolClass – bosh sinf, TnewClass – esa yangi sinf nomlari, Yangi sinf bosh sinfning barcha xossha va usullarini qabul qiladi.

Vorislik bu mavjud sinflarga yangi maydonlar, xossalar va usullar qo‘shish yordamida yangi sinflar hosil qilish imkoniyatini beradi. Yangi hosil qilingan avlod sinf asosi ya‘ni ajdod sinf xossalari va usullariga vorislik qiladi.

Avlod sinf ta‘rifida ajdod sinf nomi ko‘rsatiladi. Misol uchun TEmployee (xodim) sinfi TPerson sinfidan FDepartment (bo‘lim) maydonini qo‘shish yordamida hosil qilinishi mumkin. TEmployee sinfining ta‘rifi quyidagicha bo‘ladi:

```
TEmployee = class(TPerson)
FDepartment: integer;
constructor Create(Name:TName; Dep: integer);
end;
```

Bu misolda TEmployee sinfi TPerson sinfining vorisidir.

TEmployee o‘z konstruktoriga ega bo‘lishi lozim. TEmployee sinfi konstruktori quyidagicha berilishi mumkin:

```
constructor TEmployee.Create(Name:Tname;Dep: integer);
begin
inherited Create(Name);
FDepartment:=Dep;
end;
```

Bu misolda inherited direktivasi bilan ajdod sinf konstruktori chaqiriladi va avlod sinf maydoniga qiymat beriladi.

## Protected va private direktivalari

Sinf elementlariga murojaatni boshqarish uchun protected (himoyalangan) va private (xususiy) direktivalardan foydalaniladi.

Himoyalangan, ya'ni protected sinf elementlariga sinfdan tashqari faqat voris sinflarga murojaat qilish mumkin. Odatda, protected seksiyasiga sinf usullari ta'rifi joylashtiriladi.

Yopiq, ya'ni private sinf elementlari faqat modul ichida murojaat qilishi mumkin. Odatda, bu seksiyaga sinf maydonlari ta'riflari joylashtiriladi.

Quyida murojaatni boshqarish direktivalaridan foydalanilgan TPerson sinfi ta'rifi keltirilgan:

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

**Polimorfizm** — bu har xil sinfga kiruvchi usullar uchun bir xil nomlarni ishlatish imkoniyatini yaratishdir. Polimorfizm prinsipi shundan iboratki, sinf obyektiga mos bo'lgan biror ishning bajarilishida bir xil usulga murojaat qilish mumkinligini ta'minlab beradi. Misol uchun biz yangi sinf tashkil qilishga qaror qildik. Bu sinfning bosh sinfdan farqi uning usulida algoritm o'zgartirilgan deylik. Natijada biz, bir xil nomli usulga ega bo'lgan ikkita sinfni tashkil qilgan bo'lamiz. U holda, tashkil qilingan yangi sinf «polimorfizm» xossasiga ega bo'ladi. Sinflar dasturning Type bo'limida umumiy holda quyidagicha e'lon qilinadi:

**Type**

<yangi sinf nomi>=Class(<bosh sinf nomi>)

**Public**

<umumiy bo'lgan elementlar e'lon qilinadi>

**Published**

<inspektor obyektiga taalluqli bo'lgan elementlarni e'lon qilish>

## Projected

<yangi qaram sinfga taalluqli elementlarni e'lon qilish>

**Private**

<faqat modulga taalluqli elementlarni e'lon qilish>

**End;**

Har bir bo'limning ichida maydonlar, usullar, xossalar va hodisalarni e'lon qilish mumkin.

Polimorfizm – bu har xil sinflarga kiruvchi usullar uchun bir xil nomlardan foydalanish imkoniyatidir.

Ucha sinf ta'rifi berilgan bo'lib, bulardan biri qolgan ikki sinf uchun asos sinf bo'lsin:

**Ture**

fname: **string**;

**constructor** Create(name:string);

**function info:** **string**;

**virtual**;

**end**;

fgr:integer;

**constructor** Create(name:string;or:integer);

**function info:** **string**; **override**; **end**;

**fdep:**string;

**constructor** Create(name:string;dep:string);

**function info:** **string**;

**override**;

**end**;

Bu sinflarning har birida info usuli ta'riflangan. Asos sinfda virtual direktivasi yordamida info usuli virtual deb e'lon qilingan. Usulning virtual deb e'lon qilinishi avlod sinfda bu usulni shaxsiy usul bilan almashtirishga imkon beradi. Hosil qilingan sinfda virtual usulni almashtiruvchi usul override direktivasi bilan ta'riflanadi.

Quyida har bir avlod sinfda info usulining ta'rifi keltirilgan:

**function** TPerson.info:string;

**begin**

result := ``;

**end**;

**function** TStud.info:string;

**begin**

result := fname + ` op.` + IntToStr(fgr);

**end**;

**function** TProf.info:string;

**begin**

```
result := fname + ` kaf.` + fdep;
```

```
end;
```

Ikkala sinf bitta asos sinfdan hosil qilingani uchun talabalar va domlalar ro'yxatini quyidagicha ta'riflash mumkin:

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

Talabalar va domlalar ro'yxatini info usulini massiv elementlariga qo'llab chiqarish mumkin. Masalan:

```
st := '';
```

```
for i:=1 to SZL do
```

```
if list[i] o NIL
```

```
then st := st + list[i].Info+ #13;
```

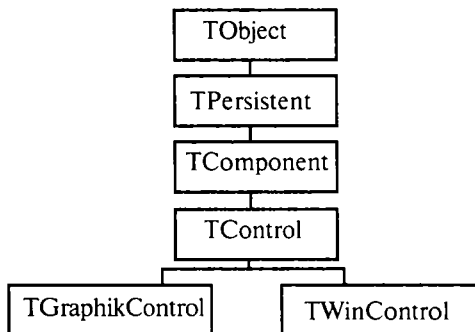
```
writeln (st);
```

### 3.5. Vizual komponentalar bibliotekasi

Delphi sinflari murakkab iyerarxik tuzilmaga ega bo'lgan vizual komponentalar bibliotekasini (Visual Component Library –VCL) tashkil qiladi. VCL tarkibiga kiruvchi yuzlab sinflar mavjud. Hamma boshqa sinfga ajdod sinf bo'luvchi asosiy sinflarga quyidagilar kiradi.

**Komponentalar** – sinflaming nusxalari bo'lib, TComponent sinfining davomchilaridir (avlodidir). Boshqa hamma sinf nusxalari **obyektlar** deyiladi. Komponentalar bilan obyektlar orasidagi farq shundaki, formada komponentalar bilan (manipulatsiya qilish) ish ko'rish mumkin, lekin obyekt bilan ish ko'rish mumkin emas. Masalan, komponenta bo'lmagan TFont sinf obyektini qaraydigan bo'lsak uni formaga joylashtirib bo'lmaydi. Komponentalar Label yoki Edit larni formaga joylashtirish mumkin va ularni joylashtirishda TFont sinf turiga kiruvchi Font xossasidan foydalaniladi.

VCLga kiruvchi sinf TObject boshqa hamma sinflarning eng yuqorisi bo'lib, obyektlarni tuzish va boshqarish imkonini beradi. Bu sinfga bir necha usullar birlashtirilgan.



VCLga kiruvchi TPersistent sinfi TObject sinfidan kelib chiqadi va u obyektlarni tashkil qilish uchun o'zida bir necha usullarni saqlaydi.

VCLga kiruvchi TComponent sinfi barcha komponentalar iyerarxiyasining eng yuqorisida joylashadi. TComponent sinfi davomchilari vizual bo'lmagan komponentalar bo'lib hisoblanadi. Vizual bo'lmagan komponentalar dasturning loyihalash bosqichidagi tashqi ko'rinishi dasturning bajarilishi bosqichidagi ko'rinishidan mutloq farq qiladi. Ayrimlari dasturning bajarilishi vaqtida umuman ko'rinmaydi. TComponent sinfi vizual komponentalar uchun asos sinf bo'lib hisoblanadi.

VCLga kiruvchi TControl sinfi katta qismdagi xossalar, usullar va vizual komponentalar hodisalarini ta'minlab beradi. Bular yordamida klaviaturadan va sichqonchadan foydalangan holda ma'lumotlarni ekranga chiqarish va dasturga kiritish mumkin.

TWinControl sinfi TControl sinfining davomchisi bo'lib, oyna elementlarini boshqarishni yaratish uchun ishlatiladi.

TGraphicControl sinfi TControl sinfining davomchisi bo'lib, grafik elementlarini boshqarish uchun ishlatiladi. TGraphicControl sinfining asosiy a'zolari quyidagilardir: Shape – geometrik figura; PaintBox – rasm chizish uchun panel; Image – tasvir; Bevel – uch o'lchovli ramka. TGraphicControl sinfi bitta usul va bitta xossaga ega.

Procedure Paint; virtual – grafik elementlarni boshqarish uchun tasvirlarni chizadi.

Property Canvas; TCanvas – grafik elementlarni boshqarishni ekranda tasvirlash uchun xizmat qiladi.

### 3.6.VCL tarkibiga kiruvchi sinflar usullari

**Sinf – TObject.** Bu sinf barcha VCL tarkibiga kiruvchi sinflar uchun bosh sinf bo'lib, obyektlarni tuzish, boshqarish va buzishni ta'minlab beradi. Buning uchun unda quyidagi usullar aniqlangan:

**Constructor Create:**

Bu usul obyektga kerakli dinamik xotirani ajratish uchun ish bajaradi.

**destructor Destroy; virtual;**

Bu usul o'chirilgan obyektga ajratilgan dinamik xotirani bo'shatadi.

**procedure Free;**

Bu usul obyektini o'chiradi va unga ajratilgan dinamik xotirani bo'shatadi.

**class function ClassName: ShortString;**

Bu usul funksiyasi sinf nomini o'z ichiga oluvchi qatorni qaytaradi. Masalan: 'Tedit', 'TButton', 'TLabel' va hokazo.



class function **ClassNameIs(const Name: string): Boolean;**

Bu usul true qiymatni qaytaradi, agar Name parametri sinf nomini o'z ichiga olgan bo'lsa.

**class function InberitsFrom(AClass: TClass): Boolean;**

Bu usul AClass parametri sinf yoki obyektning oldingi nomini o'z ichiga olgan yoki olmaganligini tekshiradi.

**class function InstanceSize: Longint;**

Sinf yoki obyekt o'lchamini baytda aniqlaydi.

**Sinf — TPersistent.** Bu sinf o'z-o'zidan TObject sinfidan kelib chiqadi va u potokli obyektlarni (потокovýй obyekt) tuzish uchun kerakli usullarni o'z ichiga oladi. **Потокли obyekt** —bu obyekt bo'lib, u potokda saqlanadi. O'z navbatida potok ham obyekt bo'lib ma'lumotlarni tashishga mo'ljallangan, masalan, xotira yoki disk fayllari. Boshqacha aytganda, sinf davomchisi Tpersistent operativ xotirada fayl formasida joylashgan bo'lib, u yerda yangilanib turiladi. Undagi usullardan to'g'ridan-to'g'ri foydalanish uchun quyidagilar ishlatiladi:

**procedure Assign(Source: TPersistent);**

Bu usul ishlatilayotgan obyektga Source parametrda nomi ko'rsatilgan ma'lumotlarni jo'natishda ishlatiladi.

**procedure AssignTo(Dest: TPersistent); virtual;**

Bu usul xuddi yuqoridagi usul kabi bo'lib, farqi uning virtualligi va himoyalanganligidir.

**procedure Define Properties(Filer: TFile); virtual;**

E'lon qilinmagan obyekt ma'lumotlarini fayl ni formaga joylashda ishlatiladi. TFile sinfi abstrakt asos sinf bo'lib, o'qish va yozish operatsiyalarini bajarishda hamda komponentalar va ular xossalarini saqlashda ishlatiladi.

**function GetNamePath: String; dynamic;**

Obyekt inspektoridagi obyekt nomini o'ziga oluvchi qatorni qaytaradi.

**function GetOwner: TPersistent; dynamic;**

Obyekt ko'rsatgichini qaytaruvchi himoyalangan usul.

**Sinf — TComponent.** TComponent sinfi komponentalar iyerarxiyasining eng yuqorisi bo'lib, undan barcha ilovalarda ishlatiladigan komponentalar tug'iladi. Uning davomchilari vizual bo'lmagan komponentalardir. Bu vizual bo'lmagan komponentalar dasturning ishlashi vaqtida ko'rinmaydi. TComponent sinfida quyidagi usullar va xossalar aniqlangan:

**Type TComponentName: String;**

**property Name: TComponentName;**

Komponentalar nomini aniqlaydi. Masalan, formaga Label1 yoki Edit2 larni joylashtirib, ular nomini o'zgartirish mumkin.

**Property Tag: Longint;**

Dasturchi uchun mo'ljallangan xossa. Bu xossada dastur tuzuvchi Longint turidagi biror sonni saqlashi mumkin.

**property ComponentCount: Integer;**

Komponentalar sonini aniqlaydi. Bu xossa faqat dastur ishlashi vaqtida va faqat o'qish uchun ruxsat etiladi.

**property ComponentIndex: Integer;**

Ro'yxatdan komponentalar holatini (o'rnini) aniqlaydi. Komponentalar nomeri noldan boshlanadi. Bu xossa ham faqat dastur ishlashi vaqtida va faqat o'qish uchun ruxsat etiladi.

**Property Components[Index: Integer]: TComponent;**

Xossalar massivi foydalanilgan komponentalar ro'yxatini aniqlaydi. Bu xossa ham faqat dastur ishlashi vaqtida va faqat o'qish uchun ruxsat etiladi.

**Procedure DestroyComponents;**

Dinamik xotiradan komponentani o'chiradi.

**procedure InsertComponent(AComponent: TComponent);**

Komponentalar ro'yxati oxiriga ASomponent parametrda ko'rsatilgan komponentni qo'yadi.

**procedure RemoveComponent(AComponent: TComponent);**

Komponentalar ro'yxatidan AComponent parametrda ko'rsatilgan komponentni o'chiradi.

**Sinf – TControl.** Bu sinf o'ziga vizual bo'lgan komponentalarning ko'pgina xossalari, usullari va hodisalarini (события) o'zida mujasamlashtirgan bo'lib, ular yordamida ma'lumotlarni ekranga chiqarish va klaviatura yordamida dasturga ma'lumotlarni kiritishi mumkin.

TControl sinfida **bosh boshqarish elementi** (parent controls) tushunchasi kiritilgan bo'lib, bu tushunchaning asl ma'nosi quyidagicha:

Har bir boshqarish elementi yoki qo'shimcha gruppalar komponentalari formaga joylashtirilgan bo'lishi mumkin, masalan, panelda (sinf TPanel). Birinchi holda, bosh boshqarish elementli forma, ikkinchi holda, esa, gruppalar elementi bo'ladi.

TControl quyidagi metodlarni ishlatadi:

**Function ClientToScreen(Const Point: TPoint): TPoint;**

Point parametrtdagi berilgan lokal koordinatani nuqtaning global koordinatasi qilib qaytaradi.

**Function ScreenToClient(Const Point: TPoint): TPoint;**

Point parametrtdagi berilgan global koordinatani nuqtaning lokal koordinatasi qilib qaytaradi.

TControl dastur bajarilishida boshqarish elementi tashqi ko'rinishini o'zgartirishi uchun yana bir necha usullar va xossalarni ham ishlatadi.

**Sinf – TWinControl.** TWinControl sinfi TControl sinfining davomchisi bo'lib, u oyna boshqarish elementlarini tashkil qilishda asos bo'lib ishlatiladi. TWinControl xossasini xarakterlovchilar namunasi sifatida Edit qatordan kiritish, Memo ko'pqatorli kiritish muharriri, ListBox ro'yxatini kiritish, Botton tugmasi va boshqalarni qarash mumkin. TwinControl ham bir qancha oyna boshqarish elementlarini tashkil qilishda ishlatiladigan xossa va usullarni o'zida mujassamlashtirgan.

Quyida TwinControl sinfining ayrim usullari berilgan:

**Type**

**Trect=record**

**Case Intelger of**

**0: (Left, Top, Reght, Buttom: Integer);**

**1: (TopLeft, ButtomRight: Tpoint);**

**End;**

Procedure AlignControls(AControl: TControl; Var Rect: TRect);  
Virtual;

Ko'rsatilgan Rect sohasida barcha «дочери» komponentalarni Align xossasiga mos holda to'g'rilaydi.

**Procedure DisableAlign;**

DisableAlign xossasi «дочери» komponentalari ichki oyna elementlarini to'g'rilashni vaqtinchalik so'raydi. Bu usul EnableAlign usuli bilan birga ishlatiladi,

**Function ConFocus: Boolean; Dynamic;**

Bu usul True qiymatini qaytaradi, agar oyna elementi kiritish fokusini olgan bo'lsa.

Procedure ChangeSkale(M D Integer); Override;

Bu usul komponentalar va uning «дочери» komponentalari mashtabini o'zgartiradi.

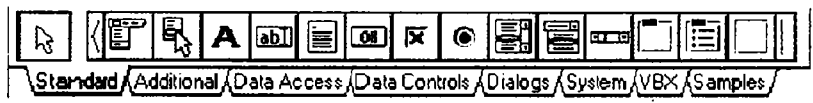
**Procedure EnableAlign;**

Oldindan chaqirilgan EnableAlign usulini (inkor qiladi) qaytaradi va «дочери» komponentalarini to'g'rilash uchun Realing chaqiradi.

### **3.7.Delphining forma komponentalari**

Forma komponentalari, bu dasturni boshqarish uchun maxsus tugmachalar bo'lib, uni formaga joylashtirishdan oldin bosh oynadan

kerakli komponentalar palitrasi tanlanadi. Masalan, Standart (**Standart**) komponentalar palitrasida quyidagi piktogrammalar (tugmachalar) majmuasi mavjud:



**MainMenu** – dastur bosh menyusi. Komponenta murakkab iyerarxik tuzilmali menyu yaratish uchun xizmat qiladi.

**PopupMenu** – yordamchi yoki lokal menyusi. Bu menyu oynada sichqonchanning o'ng tugmasini bosish bilan chiqadi.

**Label** – metka (belgi). Bu komponenta forma oynasiga uncha uzun bo'lmagan bir qatorli yozuvni chiqarishda ishlatiladi va uning piktogrammasi panelda «A» ko'rinishda berilgan.

**Edit** – kiritish qatori. Forma oynasida matnli qator kiritish va tahrirlashda ishlatiladi.

**Memo** – ko'p qatorli matn muharriri. Ko'p qatorli matnlarni kiritish yoki chiqarishda ishlatiladi.

**Button** – buyruq tugmasi (Обработчик события OnClick). Bu komponenta dasturchi tomonidan berilgan bir necha buyruqlarni bajarishda ishlatiladi.

**CheckBox** – bog'liq bo'lmagan tanlash tugmasi (переключатель). Dasturda bu komponentaning asosiy mantiqiy xossasi (Checked) o'zgartiriladi.

**RadioButton** – bog'liq bo'lgan tanlash tugmasi (переключатель). Yangi tanlash tugmasi bosilganda, oldin tanlangan tugmani avtomatik ravishda ozod etadi.

**ListBox** – ro'yxatdan tanlash. Ro'yxat variantlarini taqdim etadi va tanlash imkonini yaratadi.

**ComboBox** – kiritish qatoriga ega (комбинированный) ro'yxatdan tanlash. Ro'yxatdan kombinatsiya qilib tanlash.

**ScrollBar** – yo'lchali boshqarish. Windows oynasi chetlaridan gorizontal yoki vertikal yo'lcha tashkil etadi.

**GroupBox** – elementlar guruhi. Ma'no bo'yicha bir necha bog'liq komponentalarni guruhlashda ishlatiladi.

**RadioGroup** – bog'liq guruhlangan tanlash tugmalari (o'chirib yoquvchi tugmalar). Bir necha bog'liq tanlash tugmalari xossalarini saqlaydi.

**Panel** – panel. Bu komponenta xuddi GroupBoxga o'xshab bir necha komponentalarni birlashtirish uchun xizmat qiladi.

**Actionlist** – ta’sir qilish ro‘yxatlari. Foydalanuvchi dasturga markazlashgan holda ta’sir qilishi uchun ishlatiladi.

### 3.8. Asosiy xossalar va hodisalar

Loyiha formasi quyidagi asosiy xossalarga ega:

**ActiveControl** – Ko‘zda tutilgani bo‘yicha aktiv bo‘lishi lozim bo‘lgan komponentani ko‘rsatadi.

**Align** – Komponentani tekislash. Qiymatlari:

*alNone* – tekislanmaydi;

*alBottom* – Pastki chegarani tekislash;

*alLeft* – Chap chegarani tekislash;

*alRight* – O‘ng chegarani tekislash;

*alTop* – Yuqori chegarani tekislash.

**AlphaBlend** – Mantiqiy tip. Forma xossasi. Agar qiymati rost bo‘lsa forma shaffof bo‘ladi.

**AlphaBlendValue** – Butun turdagi xossa. Shaffoflik darajasi. Qiymati 0 dan 255 gacha.

**anchors** – Forma va komponenta xossasi. Ajdod obyektga mahkamlanish turini ko‘rsatadi:

*akLeft* – chap chegarani mahkamlash;

*akTop* – yuqori chegarani mahkamlash;

*akRight* – o‘ng chegarani mahkamlash;

*akBottom* – pastki chegarani mahkamlash.

**AutoScroll** – Mantiqiy tur. Agar qiymati true bo‘lsa forma avtomatik skrolling ya’ni siljitishni amalga oshiradi.

**AutoSize** – Mantiqiy tur. Komponentalar formada avtomatik o‘lchamlarini o‘zgartirishini ko‘rsatadi.

**BorderIcons** – Oynada qanday tugmalar bo‘lishi kerakligini ko‘rsatadi:

*biSystemMenu* – menyuni ko‘rsatish;

*biMinimize* – minimallashtirish tugmasi;

*biMaximize* – maksimalallashtirish tugmasi;

*biHelp* – yordam tugmasi.

**BorderStyle** – Forma xossasi. Forma chegarasi turini belgilaydi:

*bsSizeable* – Standart oyna. Kattaligini o‘zgartirishi mumkin.

*bsNone* – Chegaraviy hoshiyasiz oyna. Kattaligini sichqoncha bilan o‘zgartirish mumkin emas.

*bsSizeToolWin* – Ingichka chegaraviy hoshiyali oyna.

*bsToolWindow* – Ingichka chegaraviy hoshiyali oyna. Oyna kattaligini o‘zgartirish mumkin emas.

**BorderWidth** – Butun tur. Forma chegarasi kengligini belgilaydi.

**Caption** – Satrli tur. Oyna yoki komponenta sarlavhasi.

**ClientHeight** – Butun tur. Oyna kliyent, ya'ni, ishchi qismi balandligi.

**ClientWidth** – Butun tur. Oyna kliyent, ya'ni, ishchi qismi kengligi.

**Color** – oyna kliyent qismining rangi.

**Constraints** – Oyna o'Ichamlarining maksimal qiymatlari. Quyidagi parametrlari mavjud:





















**MaxHeight** – maksimal balandlik;

**MaxWidth** – maksimal kenglik;

**MinHeight** – minimal balandlik;

**MinWidth** – minimal kenglik.

**Cursor** – Sichqoncha tomonidan forma/komponent keltirilganda ko'rsatiladigan kursor shakli.

crNone	Her	CrArrow	
crCross		crIBeam	
crSizeNESW		crSizeNS	
crSizeNWSE		crSizeWE	
crUpArrow		crHourGlass	
crDrag		crNoDrop	
crHSplit		crVSplit	
crMultiDrag		crSQLWait	
crNo		crAppStart	
crHelp		crHandPoint	
crSize		crSizeAll	

**DockSite** – Mantiqiy tur. Forma/komponentaga boshqa komponentalarni Drag&Drop yordamida tashlash mumkinligini ko'rsatadi. Bu xossa MS Office da uskunalar panelini formadan ajratib yana mahkamlashga imkon beradi.

**DragKind** –s Drag&Drop da obyektни ko'chirish turi. Ikkita variant mavjud:

**dkDrag** – standart Drag&Drop. Obyekt joyida qoladi.

**dkDock** – Obyekt o'zi ko'chiriladi. Yuqorida ko'rsatilgan panel xossasiga ega bo'lish uchun shu variantni tanlash lozim.

**DragMode** – Drag&Drop rejimi. Ikkita variant mavjud:

**dmManual** – Obyekt ko'chirish rejimi foydalanuvchi tomonidan o'rnatiladi.

**dmAutomatic** – Drag&Drop rejimi avtomatik ishga tushadi.

**Enabled** – Mantiqiy tur. Agar xossa qiymati true bo'lsa, foydalanuvchi bu komponenta bilan ishlashi mumkin.

**Font** – Matni formaga chiqarishda ishlatiladigan shrift. Ikki marta shu qatorga chertilsa Windows shrift tanlash standart oynasi chiqadi.

**FormStyle** – Forma turi. Quyidagi variantlar mavjud:

*fsNormal* – normal oyna;

*fsMDIForm* – MDI oynalar uchun ajdod oyna;

*fsMDIChild* – avlod MDI oyna;

*fsStayOnTop* – oyna har doim qolganlari ustida bo'ladi.

**Height** – Butun tur. Oyna balandligi.

**Hint** – Forma/komponentaga sichqoncha keltirilganda ko'rinadigan yordamchi ma'lumot matni.

**HorzScrollBar** – Gorizontal siljitish yo'lchasi.

**Left** – Butun tur. Oynaning chap pozitsiyasi.

**Menu** – Asosiy oynadan foydalaniladigan menyular.

**Name** – Forma/komponenta nomi.

**ParentFont** – Mantiqiy tur. Agar qiymati true bo'lsa, matn uchun ajdod obyekt shrifti tanlanadi. Aks holda, foydalanuvchi ko'rsatgan shrift tanlanadi.

**Position** – Dastur ishga tushganda oynaning pozitsiyasi. Quyidagi variantlar mavjud:

*poDefault* – Oynaning o'rni va o'lchamlarini Windows tanlaydi.

*poDefaultPosOnly* – Oyna o'rnini Windows, o'lchamlarini foydalanuvchi tanlaydi.

*poDefaultSizeOnly* – Oynaning o'rnini foydalanuvchi, o'lchamlarini Windows tanlaydi.

*poDesioned* – Oynaning o'rni va o'lchovlarini foydalanuvchi tanlaydi.

*poDesktopCenter* – Oyna ishchi stoli markazida joylashadi.

*poMainFormCenter* – Oyna asosiy forma markazida joylashadi.

*poOwnerFormCenter* – Oyna o'zini chaqirgan oyna markazida joylashadi.

*poScreenCenter* – Oyna ekran markazida joylashadi.

**ShowHint** – Mantiqiy tur. Yordamchi axborot ko'rsatish kerakligini belgilaydi.

**Tao** – Butun tur. Hech narsaga ta'sir qilmaydi.

**Top** – Butun tur. Oynaning yuqori pozitsiyasi.

**TransparentColor** – Mantiqiy tur. Agar qiymati true bo'lsa, forma yoki komponenta har doim shaffof bo'ladi.

**TransparentColor Value** – Shaffof rang.

**VertScrollBar** – Vertikal siljitish yo'lchasi.

**Visible** – Mantiqiy tur. Agar qiymati true bo'lsa, yoki forma/komponent ko'rinadi, aksincha ko'rinmaydi.

**Width** – Butun tur. Oynaning kengligi.

**WindowState** – Oynaning holati. Quyidagi parametrlari mavjud:

*wsNormal* – oyna normal holatda;

*wsMaximized* – oyna maksimal holatda;

*wsMinimized* – oyna minimal holatda.

## Asosiy formaning hodisalari

Jadvalda asosiy formaning hodisalari qachon yuzaga kelishi ta'rifi berilgan. Bu hodisalarni obyektlar inspektorining Events bo'limida ko'rish mumkin.

Hodisa	Ta'rifi
OnActivate	Forma aktivlashganda
OnCanResize	Forma o'lchamini o'zgartirishdan oldin
OnClick	Formaga chertishda
OnClose	Forma yopilganda
OnCloseQuery	Formani yopishdan oldin
OnCreate	Forma yaratilganda
OnDbClick	Formaga ikki marta chertilganda
OnDeactivate	Forma deaktivlashganda
OnDestroy	Forma yo'q qilinganda
OnHide	Forma tasviri yo'qolganda
OnKeyDown	Tugma bosilganda
OnKeyPress	Tugma bosilib, qo'yib yuborilganda
OnKeyUp	Tugma qo'yib yuborilganda
OnMouseDown	Sichqoncha tugmasi bosilganda
OnMouseMove	Sichqoncha harakatlanganda
OnMouseUp	Sichqoncha tugmasi qo'yib yuborilganda
OnMouseWheel	Sichqoncha g'ildiragi tomonidan
OnMouseWheelDown	Sichqoncha g'ildiragi pastga aylantirilganda
OnMouseWheelUp	Sichqoncha g'ildiragi yuqoriga aylantirilganda
OnPaint	Forma qaytadan chizilganda
OnResize	Forma o'lchamlari o'zgariganda
OnShortCut	Issiq klavish bosilganda
OnShow	Forma hali chizilmasdan paydo bo'lganda

Bu forma tomonidan generatsiya qilinishi mumkin bo'lgan asosiy hodisalardir. Bu hodisalar komponentlarga ham tegishlidir.



## Savollar

1. Delphi dasturlash tilida qanday kengaytirilgan fayllar ishlatiladi?
2. Delphi oynasi qanday elementlardan tashkil topgan?
3. Tayyor dastur nechta asosiy etapdan o'tiladi?
4. Delphi tizimida ishga tushiriladigan modul tuzilmasi qanday ko'rinishda bo'ladi?
5. Sinfga ta'rif bering va unga misollar keltiring.
6. Maydon, usullar va xossalar tushunchasiga izoh bering.
7. Obyektga ta'rif bering va unga misollar keltiring.
8. Inkapsulatsiya tushunchasiga izoh bering.
9. Polimorfizm prinsipi qanday imkoniyatini yaratib beradi?
10. Delphi vizual komponentalar bibliotekasi (Visual Component Library – VCL) tarkibiga kiruvchi qanday sinflar mavjud. Va ularga izoh keltiring.
11. Qanday forma komponentalari bor?
12. Standart (**Standart**) komponentalar palitrasidagi piktogrammalar (tugmachalar) majmuiga izoh keltiring.
13. Loyiha formasining asosiy xossalariga tushuntirishlar bering.

## IV. DELPHI VIZUAL DASTURLASH MUHITIDA KOMPLEMENTALAR BILAN ISHLASH TEXNOLOGIYALARI

### 4.1. Label, Edit, Memo matn komponentalari va Button tugmachasi

**Label belgisi.** Belgi tushuntirishlar, nomlar, mavzular va boshqa har xil turdagi matnli ma'lumotlarni ekranga joylashtirish uchun ishlatiladi. Belgi uchun **Caption** asosiy xossalardan biri bo'lib, unda ekranga chiqariladigan matn joylashadi.

Matnni ekranga joylash uchun Delphining **Standart** palitrasidan (uskunalar panelidan) «**A**» piktogrammasi belgilanib forma ustiga kelinadi va sichqoncha tugmachasini bosgan holda matnga joylashtirilishi lozim bo'lgan joy ajratiladi. Natijada **Label1** matn maydoni hosil qilinadi va **Caption** xossasiga kirilib kerakli matn teriladi.

Matnga ishlov berish uchun (masalan, kattalashtirish yoki kichiklashtirish; kursiv yoki qalin qilish va boshqa) ya'ni unga o'zgartirish kiritish uchun kerakli xossa tanlanib ular o'zgartiriladi. Masalan, kiritilgan matnni kattalashtirish yoki kichiklashtirish uchun oldin matn maydoni ajratilib, keyin **Font** xossasiga kiriladi va muloqot darchasidan shrift, uning o'lchami va rangi tanlanib Ok tugmasi bosiladi.

**Label** komponentasi nafaqat ma'lumotlarni ekranga joylashtirish uchun xizmat qiladi, balki dastur natijalarini chiqarishda ham ishlatish mumkin. Buning uchun dasturda **Label5.caption: =`Dastur natijasi`**; buyrug'i berilishi kerak. Misol: **Label5.caption:=`yechim=`+s**; bu yerda **s:String** o'zgaruvchisi.

**Edit kiritish qatori.** Edit kiritish qatori matnni bir qatordan kiritish va uni tahrirlash uchun ishlatiladi.

Matn kiritish qatorini ekranga joylash uchun Delphining **Standart** palitrasi (uskunalar paneli) dan «**ab**» piktogrammasi belgilanib forma ustiga kelinadi va sichqoncha tugmachasini bosgan holda matn kiritilishi lozim bo'lgan joy ajratiladi. Natijada **Edit1** matn kiritish maydoni hosil qilinadi. Matnni kiritish dasturi ishchi holatiga o'tilganda bajariladi.

Matn qatoriga kiritilgan ma'lumot faqat matn, ya'ni String (qator) bo'lib hisoblanadi. Edit kiritish qatoriga kiritilgan ma'lumotni dasturda o'qib va uni raqamga o'tkazish uchun ko'p hollarda Val funksiyasidan foydalaniladi. Bu funksiya Turbo Pascalda quyidagicha yoziladi. **Val(Edit1.Text,a,cod)** – bu yerda a: Real; – o'zgaruvchisi bo'lib, Edit1.Text maydonidagi ma'lumotni raqam qilib o'zlashtiradi. **cod: Integer**; deb e'lon qilinadi.

**Memo matn chiqarish qatori.** Memo matnlarni bir necha qator qilib chiqarish uchun ishlatiladi.

**Memo matn chiqarish qatorini** ekranga joylash uchun Delphi-ning **Standart** palitrasi (uskunalar paneli) dan «**ab**» piktogrammasi yonidagi Memo tugmasi belgilanib forma ustiga kelinadi va sichqoncha tugmachasini bosgan holda matn chiqarilishi lozim bo'lgan joy ajratiladi. Natijada **Memo1** matn chiqarish maydoni hosil qilinadi. Bu matn chiqarish maydoni dasturda natijalarni chiqarishda qo'l keladi. Natijani chiqarishda u dastur ichida quyidagicha ishlatiladi. **Memo1.Lines.add(`Yechim=`+S)**;

Memo maydonini tozalash esa natijani chiqarishdan oldin modulda **Memo1.Clear**; buyrug'ini berish bilan amalga oshiriladi.

**Button tugmachasi.** Button tugmachasi bosilishi natijasida kutilishi lozim bo'lgan jarayonlar (masalan, hisoblashlar yoki bajarilishi lozim bo'lgan operatsiyalar) bajarilishga tushiriladi.

Button tugmachasini ekranga joylash uchun Delphining **Standart** palitrasi (uskunalar paneli) dan «**Ok**» piktogrammasi belgilanib forma ustiga kelinadi va sichqoncha tugmachasini bosgan holda tugmacha qo'yilishi lozim bo'lgan joy ajratiladi. Natijada **Bottom1** tugmachasi hosil qilinadi. Tugmacha nomini o'zgartirish **Caption** xossasiga kirilib o'zgartiriladi.

Dasturdagi hisoblash jarayonlari, kiritish va chiqarish operatsiyalari hosil qilingan tugmachani ikki marta tez-tez bosish bilan «события» ni qayta ishlash darchasiga o'tilib, u yerdan modul ichiga kerakli operatorlarni yozish bilan amalga oshiriladi.

## **4.2. Boshlang'ich forma ilovasini yaratish**

Delphida boshlang'ich formani tuzishda forma Form1 xossalarini o'zgartirish bilan boshlanadi. Forma xossalarini, uning tashqi ko'rinishini, ya'ni uning o'lchami, ekranda joylashishi, oynaning ko'rinishi va sarlavha matnini aniqlab beradi. Uning xossalari Object Inspector oynasida keltirilgan bo'lib, oynaning chap ustunida xossa nomlari va o'ng ustunida uning qiymatlari berilgan.

Formaga yangi komponentalarni joylashtirish uncha katta qiyinchilik tug'dirmaydi. Biror-bir komponentani joylashtirish uchun uni komponentalar politrasiidan belgilab olib, keyin uni bir marta chiqillatish kerak va keyin xossalarini o'zgartirish uchun uning formadagi ko'rinishini sichqonchada yana bir marta chiqillatish zarur.

Masalan, Label (metka) komponentasini formaga joylashtirish uchun uni Standart komponentalar politrasiidan topib u sichqonchada bir marta chiqillatiladi, natijada formada u Label1 nomi bilan joylashadi. Bu komponenta formaga har xil matnlarni joylashtirish uchun xizmat qiladi. Uning boshlang'ich holatini va xossalarini o'zgartirish uchun uni formadan belgilaymiz. Uning Label1 standart nomini o'zgartirish uchun obyekt inspektori oynasidan Caption xossasiga kiramiz va Label1 nomini o'chirib o'miga, masalan, «Mening birinchi dasturim» degan so'zni yozamiz. Yozilgan matn rangi va o'lchamini o'zgartirish esa Font xossasiga kirilib Size va Color parametrlarini o'zgartirish bilan amalga oshiriladi.

Yana bir komponentani – Botton (tugmacha) komponentasini formaga joylashtiraylik. Bu komponenta dasturchi tomonidan berilgan (kiritilgan) dastur kodlarini ishga tushirish uchun mo'ljallangan. Unga hodisalarni qayta ishlovchi (On Click)(обработчик события) deyiladi.

### **Misol 1.**

Delphining imkoniyatlarini va uning vizual loyihalash vositasi texnologiyasini namoyish etish uchun misol tariqasida kvadrat tenglama yechimlarini topish dasturini yaratishni qaraylik. Buning uchun loyihaning boshlang'ich elementlarini yaratishdan boshlaylik. Delphi foydalanuvchiga Form1 nomli standart formani taklif etadi. Foydalanuvchi forma xossalarini Object Inspector oynasidan o'zgartirish imkoniga ega. Forma xossalari uning ekrandagi ko'rinishini aniqlaydi. Xossalar ro'yxatini obyektlar inspektori (Object Inspector) oynasining

pastki qatoridagi Propertiesni (xossalarni) aktivlashtirish bilan ko'rish mumkin. Oynaning chap ustunida xossalarni nomlari, o'ng ustunida xossalarning joriy qiymatlari berilgan. Xossa qiymatini o'zgartirish uchun mos xossa yozilgan maydonni sichqonchada chiqillatib, natijada hosil bo'lgan o'ng tomondagi unga mos xossa qiymati aniqlanadi, ya'ni o'zgartiriladi. Masalan, caption (sarlavha) xossasi qiymatini o'zgartirish uchun, oldin caption sichqonchada chiqillatilib keyin «form1» xossa qiymati klaviaturadan Backspace tugmasini bosish bilan o'chirilib, o'rninga «Kvadrat tenglamani yechish dasturi» matni kiritiladi.

Masalan, kvadrat tenglamani yechish dasturchi uchun birinchi uchta piktogramma kerak bo'ladi. Bu piktogrammalarni formaga joylashtirish uchun komponentalar palitrasidagi kerakli piktogramma ikki marta sichqonchada chiqillatiladi va keyin forma maydonida hosil bo'lgan kiritish belgisi yoki tugmacha kerakli joyga joylashtiriladi.

Berilgan misol uchun formaga ishlanadigan ilovaga oltita metka qo'yish kerak bo'ladi. Birinchi Label1 tenglama yechimlarini chiqarish uchun, ikkinchi Label2 forma boshida ma'lumot berish uchun (masalan, tenglama koeffitsiyentlari:) va qolgan uchta Label 3, Label 4, Label 5 tahrirlash maydoniga tushuntirish berish uchun (masalan, koeff. a) formaga qo'yiladi.

Formaga yangi hisob va chiqish tugmachalarini joylashtirish uchun standart komponentalar palitrasidan «Ok» piktogrammasi uch marta ikki martadan chiqillatilib, formaning kerakli joylariga qo'yiladi va keyin ular nomlari, ya'ni, qiymatlari xossadan aniqlanadi. Natijada quyidagi formaga ega bo'linadi.

The image shows a screenshot of a Windows application window titled "Form1". The window contains a form with a grid background. The title of the form is "Kvadrat tenglamani yechish dasturi". Below the title, there is a section labeled "Koeffitsiyentlar" (Coefficients). Under this section, there are three input fields labeled "a=", "b=", and "c=", each with a small "Edit" button next to it. To the right of these input fields, there are three buttons stacked vertically: "Yangi" (New), "Hisob" (Calculate), and "Chiqish" (Exit). Below the input fields, there is a section labeled "Yechimlar" (Solutions). The form is set against a dotted grid background.

**Hodisa va uni qayta ishlash.** Yaratilgan forma ilovaning qay tarzda ishlashini ko'rsatib beradi. Formadagi buyruq tugmachalari biror ish bajarishi uchun ular sichqonchada ko'rsatilib chiqillatiladi. Sichqonchada tugmachani chiqillatish (bosish) hodisaga misol bo'lib, u ilovaning ishlash jarayonida hosil bo'ladi. Bu yerda hodisa so'zini yuz beradigan jarayon deb tushunish kerak.

Hodisalarga javob Delphida ularning qayta ishlovchi protseduralar ko'rinishida tashkil qilinadi. Turbo Pascal tilida yoziladigan bu protseduralar hodisa qayta ishlovchisi («обработчик») deb ataladi.

Delphi avtomatik ravishda qayta ishlovchiga ikkita qismdan iborat nom beradi. Birinchi qism nomi formani, obyektga kiruvchilarni o'z ichiga olib, ikkinchi qism nomi esa aynan obyekt o'zini va qayta ishlovchini aks ettiradi. Bizning misolimizda forma nomi – Form1, birinchi buyruq tugmasining nomi «hisob» – Button1, qayta ishlovchi nomi esa – Click. Endi Begin va End orasiga qayta ishlovchi bajaruvchi Pascal tilidagi operatorlarni quyidagi protseduraga kiritish mumkin. Bu protsedura «hisob» tugmasini ikki marta tez-tez chiqillatish bilan ekranga chaqiriladi.

**Procedure TForm1.Button1click(Sender:Tobject);**

**Var**

**A,B,C:Real;**

**D:Real;**

**X1,X2:Real;**

**S1,S2:String[7];**

**Code:Integer;**

**Begin**

**Val(Edit1.Text,a,Code);**

**Val(Edit2.Text,b,Code);**

**Val(Edit3.Text,c,Code);**

**If a=0 Then**

**Label6.Caption:='Xato! '+Chr(13)**

**+'Noma'lum ikkinchi darajasi koeffitsiyenti'**

**+Chr(13)+'noiga teng'**

**Else Begin**

**d:=b\*b-4\*a\*c;**

**If d<0 then label 6. Caption :='Yechim mavjud emas' Else Begin**

**x1:=(-b+Sqrt(d))/(2\*a);**

**x2:=(b+Sqrt(d))/(2\*a);**

**Str(x1:7:3,S1);**

```
Str(x2:7:3,S2);  
Label6.Caption:='Tenglama ildizlari:'  
+Chr(13)+'x1=' +S1+  
+Chr(13)+'x2=' +S2;
```

**End;**

**End;**

**End;**

Keltirilgan dastur matnida Turbo Pascalning oddiy Read va Write (Kiritish va chiqarish) operatorlari ishlatilmagan. O'zgaruvchilar qiymatini kiritish tahrirlash maydonidan Text xossasiga murojaat qilish bilan amalga oshiriladi. Kiritilgan o'zgaruvchilar qiymati matn bo'lgani uchun ular Val funksiyasi yordamida raqamga o'tkaziladi. Kvadrat tenglamaning ildizlari x1 va x2 qiymatlari Str funksiyasi orqali mos ravishda s1 va s2 o'zgaruvchilarga matnli qilib uzatiladi. Natijani ekranga matn ko'rinishida berish uchun Label6. Caption metkasiga qiymat qilib yuboriladi.

Xuddi shunday «yangi» va «chiqish» tugmachalari uchun ham qayta ishlovchi protseduralarini tashkil qilish kerak. Ularning matnlari quyidagi ko'rinishga ega.

```
Procedure Tform1.Button2Click(Sender:Tobject);
```

```
Begin
```

```
Edit1.Text:=' ';
```

```
Edit2.Text:=' ';
```

```
Edit3.Text:=' ';
```

```
Label2.Caption:=' ';
```

```
Edit1.SetFocus;
```

```
End;
```

```
Procedure Tform1.Button3click(Sender: Tobject);
```

```
Begin
```

```
Form1.Close;
```

```
End;
```

### **Loyihani saqlash. Ilovani kompilatsiya qilish va ishga tushirish.**

Loyihani saqlashda Delphi bir necha faylni tashkil qiladi. Ayrımlari butun loyihani tavsiflashni, boshqalari forma va dastur modulini tavsiflashni o'z ichiga oladi. Agar hali saqlanmagan loyiha bo'lsa Fayl (File) menyusidan Сохранит проект (Save Project) buyrug'i beriladi va keyin dastur moduli va proyektnomi beriladi.

Loyihani bog'lab bo'lgandan so'ng Compile menyusidan compile (Компилировать) buyrug'i beriladi. Agar dasturda sintaksis xato

bo'lmasa ekranda kompilyatsiya to'g'ri o'tganligi haqida xabar beriladi. Agar kompilyatsiya dasturda qandaydir xatoni topsa, xato haqida ekranga ma'lumot beradi. Kompilyatsiyadan to'g'ri o'tgan dastur uchun maxsus – .exe kengaytmali fayl tuzib beradi va u faylni Delphi tizimisiz ishlatish mumkin.

Delphi tizimidan chiqmasdan turib ilovani ishga tushirish mumkin, buning uchun Run menyusining Run buyrug'ini yoki F9 tugmachasini bosish kifoya bo'ladi. Yuqoridagi misol uchun ilova ishga tushirilib a, b va c qiymatlari kiritilib «hisob» tugmasi bosilsa dastur quyidagi natijani ekranga chiqaradi.



## Kvadrat tenglamani yechish dasturi

### Koeffitsiyentlar

a=

b=

c=

Yangi

Hisob

Chiqish

### Yechimlar

### Tenglama ildizlari

x1= 0.871

x2= 2.871

Protsedura TForm1.Button2Click «yangi» tugmachasini sichqonchada chiqillatish bilan ishlaydi va tahrirlash maydoniga kursorni koeffitsiyent qiymatlarini kiritish uchun olib kelib qo'yadi.

Protsedura TForm1.Button3Click «tamom» tugmachasini sichqonchada chiqillatish bilan ishlaydi va formani yopadi.

### Misol 2.

Delphi imkoniyatlarini va vizual loyihalash texnikasini ko'rsatish uchun sportsmen distansiyani chopib o'tgan tezlikni hisoblovchi loyihani ishlab chiqamiz. Dastur ishlash jarayonidagi oynasining ko'rinishi rasmda ko'rsatilgan.

### Sportsmen tezligini hisoblash dasturi

Distansiya(m)

Vaqt (min.cek)

**Hisoblash**

**Distansiya: 1000 m**  
**Vaqt: 3 min 20 cek**  
**Tezlik: 18.00 km/soat**

**Chiqish**

Chopish tezligi dasturining matni.

```

unit Unit1;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms, Dialogs, StdCtrls;
type
    TForm1 = class(TForm)
        Edit1: TEdit;
        Edit2: TEdit;
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        Label4: TLabel;
        Button1: TButton;
        Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    private
    { Private declarations } public
    
```



```

{ Public declarations } end;
var
Form1: TForm1;
implementation
{ $R *.dfm }
procedure TForm1.Button1Click(Sender: TObject);
var
dist : integer;
t: real;
min :integer;
sek : integer;
v: real;
begin

dist := StrToInt(Edit1.Text);
t := StrToFloat(Edit2.Text);
min := Trunc(t);
sek := Trunc(t*100) mod 100;
v := (dist/1000) / ((min*60 + sek)/3600);
label4.Caption := 'Distansiya: ' + Edit1.Text + ' m' + #13
+ 'Vaqt: ' + IntToStr(min) + ' min '
+ IntToStr(sek) + ' sek ' + #13 +
'Tezlik: ' + FloatToStrF(v,ffFixed,4,2) + 'km/sgat';
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close;
end;
end.

```

Button1click funksiyasi tezlikni hisoblab, natijani Label4 maydoniga chiqaradi. Boshlang'ich qiymatlar Edit1 va Edit2 tahrirlash maydonlarining Text xossasiga murojaat qilish yordamida kiritiladi. Text xossasi foydalanuvchi kiritgan simvollardan iborat satrni o'z ichiga oladi. Satrni sonlarga aylantirish uchun StrToInt va StrToFloat funksiyalaridan foydalaniladi. StrToInt funksiyasi Edit1.Text ga kiritilgan satrni tekshirib, agar simvollar raqamlardan iborat bo'lsa, butun songa aylantirib dist. o'zgaruvchisiga qiymat sifatida beradi. Shu kabi StrToFloat funksiyasi Edit2. Text kiritilgan satrni haqiqiy songa aylantirib t o'zgaruvchisiga qiymat sifatida beradi.

So'ngra Trunc funksiyasi t o'zgaruvchisining butun qismini ajratadi — bu minutlarga mos keladi. So'ngra Trunc(t\*100) mod 100 ifodasi sekundlarni ajratadi.

Tezlik km/soat birlikda aniqlangani uchun, metr va sekundlar kilometr va soatlarga aylantiriladi.

Hisoblangan tezlik qiymati Label4 maydonida Caption xossasi yordamida akslantiriladi. Sonni satrga aylantirish uchun IntToStr va FloatToStr funksiyalaridan foydalaniladi.

Chiqish tugmasini bosilganda dastur ishini to'xtatadi. Buning uchun close usuli yordamida dastur oynasi berkitiladi.

### 4.3. Tanlash tugmalarini o'rnatish

**RadioGroup** guruhli tanlash tugmalari ilovalar yaratishda bir necha variantlardan birini tanlash uchun ishlatiladi. Bu komponenta Standart komponentalar palitrasida joylashgan bo'lib, u  $\Xi$  ko'rinishdagi piktogrammaga ega. Uning asosiy xossasi Items bo'lib, u tugmalar nomlari ro'yxatini o'zida saqlaydi. Tugmalar nomlari ro'yxatini kiritishdan oldin RadioGroup tugmasi uchun formadan joy ajratiladi va keyin Items xossasi ko'rsatilib, undan uch nuqtali tugmacha bosiladi, natijada StringList Editor oynasi ochiladi. Bu oynadan tanlash tugmalari nomlarining har qaysisi yangi qatordan kiritiladi va keyin «Ok» tugmasi bosiladi. Formaga RadioGroup guruhli tanlash tugmasi joylashtirilganda u RadioGroup1 nom bilan yoziladi. Bu nomni boshqa mos nomga almashtirish Caption xossasiga kirib amalga oshiriladi.

**CheckBox** komponentasi ro'yxatdan bir nechtasini tanlash imkonini beradi. **CheckBox** komponentasi **Additional** palitrasida joylashgan. RadioGroup bog'liq pereklyuchatelami, **CheckBox** esa bog'liq bo'lmagan pereklyuchatellarni birlashtiradi. Bunda yoquvchi uch xil holatda bo'lishi mumkin:

- yoqilgan (включен) – to'g'ri belgisi;
- o'chirilgan (выключен) – bo'sh belgisi;
- neytral holat – ko'kish rangda to'g'ri belgisi.

**CheckBox** ning asosiy xossalari:

**AllowOryer** – uchinchi neytral holat variantini ishlatishni taqiqlaydi;

**Items** –tanlash tugmalari nomlari ro'yxatini saqlaydi.

### M i s o l 3.

Edit kiritish qatorida terilgan matn holatini o'zgartiruvchi ilova yaratish.

## Ye ch i sh

1. Yangi ilova yaratamiz.

2. Formaga Standart komponentalar palitrasidan RadioGroup komponentasini uch marta RadioGroup1, RadioGroup2, RadioGroup3 nomlari bilan o'rnatamiz va uning caption xossasi qiymatiga mos ravishda «yozuv», «o'lcham» va «rang» qiymatlarini beramiz.

3. Items xossasiga o'tamiz. Undan uch nuqtali tugmachani bosib, StrinoList Editor oynasiga kiramiz va bu oynadan включателлар nomlarini har qaysisini yangi qatordan kiritamiz:

RadioGroup1 — komponentasi uchun

обычный

курсив

полужирный

полужирный курсив

RadioGroup2 — komponentasi uchun

8

10

12

14

RadioGroup3 — komponentasi uchun

черный

зеленый

красный

синий

Har qaysisi uchun kiritishni tugatgandan so'ng «Ok» tugmasi bosiladi.

4. **CheckBox** komponentasini **Additional** palitrasidan olib formaga o'rnatamiz va uning Items xossasiga kirib включателлар nomlarini kiritamiz.

Зачеркнутый

Подчеркнутый

5. **Edit** komponentasini **Standart** palitrasidan olib formaga o'rnatamiz va uning Text xossasiga «Kompyuter» qiymatini kiritamiz.

6. Mos ravishda **CheckBox** va **Edit1** komponentasi ramkalari yuqorisiga Label1 va label2 metkalarini o'rnatamiz va ularning Caption xossasiga «Атрибутлар» va «Образец» qiymatini beramiz.

7. RadioGroup1 komponentasi maydonini ikki marta tez bosamiz va paydo bo'lgan kodlarni quyidagi kiritish maydoniga kiritamiz:

**Case RadioGroup1.ItemIndex of**

**0: Edit1.Font.Style:= [ ];**

**1: Edit1.Font.Style:= [FsItalic];**

**2: Edit1.Font.Style:= [FsBold];**

**3: Edit1.Font.Style:= [FsItalic,FsBold];**

**End.**

8. RadioGroup2 komponentasi maydonini ikki marta tez bosamiz va paydo bo'lgan kodlarni quyidagi kiritish maydoniga kiritamiz:

**Case RadioGroup2.ItemIndex of**

**0: Edit1.Font.Size:=8;**

**1: Edit1.Font.Size:=10;**

**2: Edit1.Font.Size:=12;**

**3: Edit1.Font.Size:=14;**

**End.**

9. RadioGroup3 komponentasi maydonini ikki marta tez bosamiz va paydo bo'lgan kodlarni quyidagi kiritish maydoniga kiritamiz:

**Case RadioGroup3.ItemIndex of**

**0: Edit1.Font.Color:=ClBlack;**

**1: Edit1.Font.Color:=ClGreen;**

**2: Edit1.Font.Color:=ClRed;**

**3: Edit1.Font.Color:=ClBlue;**

**End.**

10. CheckBox komponentasi maydonini ikki marta tez bosamiz va paydo bo'lgan kodlarni quyidagi kiritish maydoniga kiritamiz:

**If CheckBox1.Checked[0]**

**Then Edit1.Font.Style:=Edit1.Font.Style+[FsStrikeOut]**

**Else Edit1.Font.Style:=Edit1.Font.Style-[FsStrikeOut];**

**If CheckBox1.Checked[1]**

**Then Edit1.Font.Style:=Edit1.Font.Style+[FsUnderLine]**

**Else Edit1.Font.Style:=Edit1.Font.Style-[FsUnderline];**

11. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlaymiz.

12. Yangi nom bilan saqlangan loyiha (проект), ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi. Endi ilova bilan ishlash mumkin.

Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

**Unit pxx2;**

**interface**

**uses**

<b>Yozuv</b>	<b>O'lcham</b>	<b>Rang</b>
<input type="radio"/> Обычный	<input type="radio"/> 8	<input checked="" type="radio"/> Черный
<input checked="" type="radio"/> Курсив	<input type="radio"/> 10	<input type="radio"/> Зеленый
<input type="radio"/> Полужирный	<input checked="" type="radio"/> 12	<input type="radio"/> Красный
<input type="radio"/> Полужирный курс	<input type="radio"/> 14	<input type="radio"/> Синий

Atribut

Abrazets

<input type="checkbox"/> Зачеркнутый
<input checked="" type="checkbox"/> Подчеркнутый

КОМПЬЮТЕР

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

Type

```

TForm1 = class(TForm)
  Edit1: TEdit;
  CheckListBox1: TCheckListBox;
  Label1: TLabel;
  Label2: TLabel;
  RadioGroup1: TRadioGroup;
  RadioGroup2: TRadioGroup;
  RadioGroup3: TRadioGroup;
  procedure RadioGroup1Click(Sender: TObject);
  procedure RadioGroup2Click(Sender: TObject);
  procedure RadioGroup3Click(Sender: TObject);
  procedure CheckListBox1Click(Sender: TObject);
  private
  { Private declarations }
  public
  { Public declarations }
  end;
Var
  Form1: TForm1;

```

**implementation**

**{SR \*.dfm}**

**procedure TForm1.RadioGroup1Click(Sender: TObject);**

**begin**

**Case RadioGroup1.ItemIndex of**

**0: Edit1.Font.Style:= [ ];**

**1: Edit1.Font.Style:= [FsItalic];**

**2: Edit1.Font.Style:= [FsBold];**

**3: Edit1.Font.Style:= [FsItalic, FsBold];**

**End;**

**{CheckBox1ClickCheck(Self);}**

**end;**

**Procedure TForm1.RadioGroup2Click(Sender: TObject);**

**begin**

**Case RadioGroup2.ItemIndex of**

**0: Edit1.Font.Size:=8;**

**1: Edit1.Font.Size:=10;**

**2: Edit1.Font.Size:=12;**

**3: Edit1.Font.Size:=14;**

**End;**

**end;**

**Procedure TForm1.RadioGroup3Click(Sender: TObject);**

**begin**

**Case RadioGroup3.ItemIndex of**

**0: Edit1.Font.Color:=ClBlack;**

**1: Edit1.Font.Color:=ClOreen;**

**2: Edit1.Font.Color:=ClRed;**

**3: Edit1.Font.Color:=ClBlue;**

**End;**

**end;**

**Procedure TForm1.CheckListBox1Click(Sender: TObject);**

**begin**

**If CheckListBox1.Checked[0]**

**Then Edit1.Font.Style:=Edit1.Font.Style+[FsStrikeOut]**

**Else Edit1.Font.Style:=Edit1.Font.Style-[FsStrikeOut];**


**If CheckListBox1.Checked[1]**

**Then Edit1.Font.Style:=Edit1.Font.Style+[FsUnderLine]**

**Else Edit1.Font.Style:=Edit1.Font.Style-[FsUnderLine];**

end;  
end.

#### 4.4. ListBox va ComboBox komponentalari


**ListBox** komponenti ro'yxat va massiv ko'rinishidagi ma'lumotlarni ekranda aks ettirishda ishlatiladi. Ma'lumotlarni kiritishda esa Edit komponentasidan foydalaniladi. ListBox komponentasi Standart komponentalar palitrasida joylashgan bo'lib, u  ko'rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan ro'yxat uchun joy ajratiladi va keyin esa xossalari aniqlanadi.

Uning ayrim asosiy xossalari:

Items – ro'yxat elementlarini beradi;

Sorter – ro'yxat elementlarini alfavit bo'yicha avtomatik ravishda tartiblaydi;

Clear – barcha ro'yxat elementlarini o'chiradi.

**ComboBox** komponenti ro'yxat va massiv ko'rinishdagi ma'lumotlarni ekranga kiritish uchun ishlatiladi. U ListBox va Edit komponentalarining birgalikdagi ishini bir o'zi bajaradi. Tashqi ko'rinishdan bu komponent oddiy Edit kiritish qatorini eslatadi. Uning o'ng qismida pastga belgisi bo'lib, kiritilayotgan ma'lumotlarni ko'rib borish mumkin. Bu komponenta Standart komponentalar palitrasida joylashgan bo'lib, u  ko'rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan ro'yxat uchun joy ajratiladi va keyin esa xossalari aniqlanadi. Uning ayrim asosiy xossalari:

DropDownCount – ro'yxatdagi ekranga chiqadigan ma'lumotlar sonini aniqlaydi. Bu xossaning boshlang'ich qiymati 8 ga teng bo'ladi. Agar ekranga chiqadigan ma'lumotlar sonini 10 ta bo'lsin desak, unda uning qiymatini 10 ga o'zgartirish kerak bo'ladi. Agar kiritilgan ma'lumotlar undan ortiq bo'lsa, u holda pastga va yuqoriga siljitish tugmachasi avtomatik ravishda paydo bo'ladi;

Style – ro'yxatdagi ma'lumotning ko'rinishini tasvirlaydi;

Text – ro'yxatdagi kiritilgan ma'lumot matn ekanini bildiradi.

#### M i s o l 4.

Butun qiymatli A(10) massiv elementlari ichidan eng katta va eng kichiklari topilsin. Ilovada Listbox komponentasini ishlatning.

#### Ye ch i sh

1. Yangi ilova yaratamiz.
2. Formaga Standart komponentalar palitrasidan Listbox komponentasini ListBox1 nom bilan, Edit komponentasini Edit1

nom bilan va ikkita Botton1 va Botton2 tugmalarini o'rnatamiz.

3.Edit komponentasining text xossasiga kirib, Edit1 qiymatini bo'sh qator qilib beramiz.

4.Botton1 va Botton2 tugmachalarining Caption xossasiga kirib, ularni «Kiritish» va «Yechish» qiymatiga tenglashtiramiz.

5.»Yechish» tugmasi pastiga «Memo» komponentasini «Memo1» nom bilan o'rnatamiz.

6.Forma ustiga sichqonchada ikki marta bosib, kodlarni yozish oynasiga o'tamiz va quyidagilarni kiritamiz:

**i:=0;**

**ListBox1.Clear;**

Inteface bo'limiga massiv va ishlatiladigan o'zgaruvchilarni Var so'zidan keyin tavsiflaymiz:

**a:Array[1..10] of Integer;**

k,i,maxx,minn: Integer;

s1,s2: String;

7. «Kiritish» tugmasini aktivlashtirish uchun uni ikki marta tez-tez bosib, kodlarni yozish oynasiga o'tarniz va quyidagilarni kiritamiz:

**ListBox1.Items.Add(Edit1.text);**

**i:=i+1;**

**a[i]:=StrToInt(Edit1.text);**

**Edit1.SetFocus;**

8. «Yechish» tugmasini aktivlashtirish uchun uni ikki marta tez-tez bosib, kodlarni yozish oynasiga o'tamiz va quyidagilarni kiritamiz:

**minn:=a[1];**

**maxx:=a[1];**

**For k:=1 to 10 do**

**Begin**

**If minn>a[k] Then Minn:=a[k];**

**If maxx<a[k] Then maxx:=a[k];**

**End;**

**Str(maxx:5,S1);**

**Str(minn:5,S2);**

**Memo1.Clear;**

**Memo1.Lines.Add('Eng kattasi=' +s1);**

**Memo1.Lines.Add('Eng kichigi =' +s2);**

9.Kiritish fokusi Edit1 kiritish qatorida turishi uchun uni ikki marta bosib kodlarni yozish oynasiga o'tamiz va quyidagini kiritamiz:

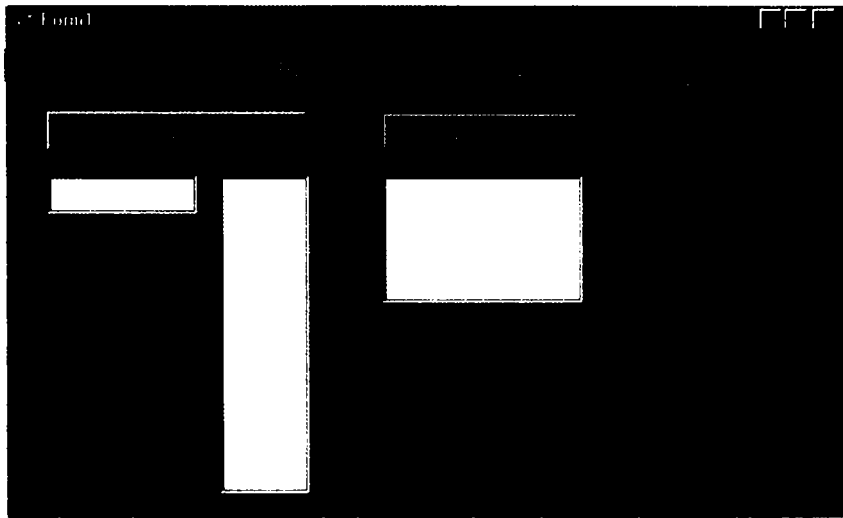
**If key=13 Then Button1.SetFocus;**



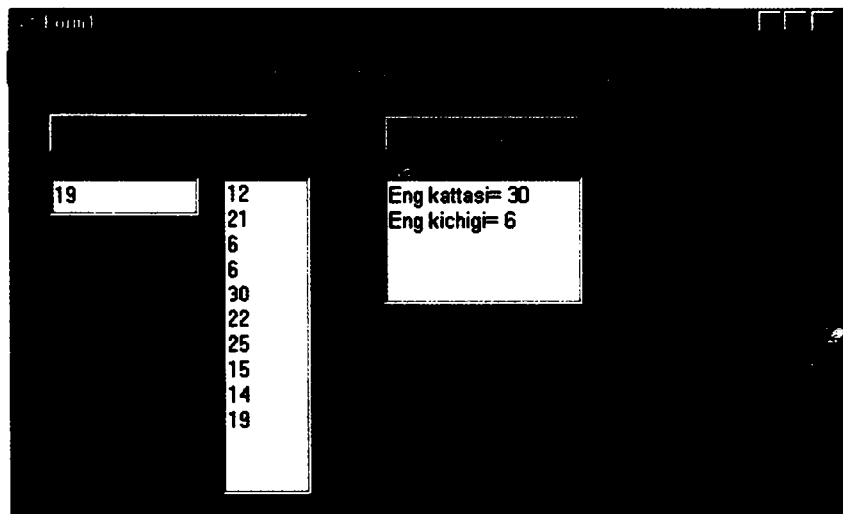
10. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlaymiz.

11. Yangi nom bilan saqlangan proyekti, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi.



Massiv elementining qiymatlarini kiritib, «Yechish» tugmasini bosamiz va quyidagi natijaga ega bo'lamiz.



```

Tashkil qilingan modulning to'liq ko' rinishini keltiramiz:
unit pxx3;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms, Dialogs, StdCtrls;
type
    TForm1 = class(TForm)
        Listbox1: TListBox;
        Button1: TButton;
        Button2: TButton;
        Edit1: TEdit;
        Memo1: TMemo;
        Label1: TLabel;
        procedure FormCreate(Sender: TObject);
        procedure Button1Click(Sender: TObject);
        procedure Button2Click(Sender: TObject);
        procedure Edit1KeyDown(Sender: TObject;var key: Word;
            Shift: TshiftState);
    private
        { Private declarations }
    public
        { Public declarations }
    end;
var
    Form1: TForm1;
    a:Array[1..10] of Integer;
    k,i,maxx,minn:Integer;
    s1,s2:String;
implementation
    {SR *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    Listbox1.Items.Add(Edit1.text);
    i:=i+1;
    a[i]:=StrToInt(Edit1.text);
    Edit1.SetFocus;
end;

procedure TForm1.FormCreate(Sender: TObject);

```

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

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

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

### **M i s o l 5.**

Butun qiymatli A(10) massiv elementlari ichidan eng katta va eng kichiklari topilsin. Ilovada ComboBox komponentasini ishlatning.

### **Ye ch i sh**

- 1.Yangi ilova yaratamiz.
- 2.Formaga Standart komponentalar palitrasidan **ComboBox** komponentasini ComboBox1 nom bilan, Memo komponentasini Memo1 nom bilan va ikkita Botton1 va Botton 2 tugmalarini o‘rnatamiz.
- 3.Oldingi misol kabi bu komponentalarning xossalarini ham o‘rnatamiz va dastur kodlarini ham kiritamiz. Hamma dastur kodlari «kiritish» tugmasiga bog‘liq, ya’ni Botton1 moduli kodlari quyidagicha bo‘ladi.

```

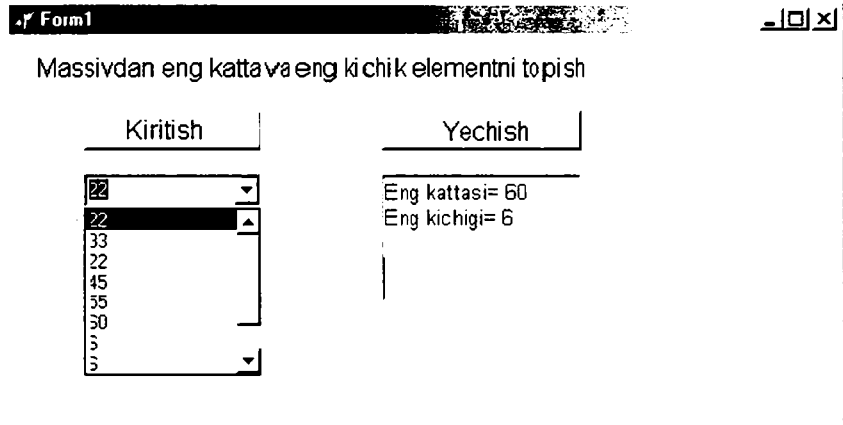
ComboBox1.Items.Add(ComboBox1.text);
i:=i+1;
a[i]:=StrToInt(ComboBox1.text);
ComboBox1.SetFocus;

```

4. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlaymiz.

5. Yangi nom bilan saqlangan proyekt, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi.



Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

```

unit s1p;
interface
uses

```

```

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

```

**Type**

```

TForm1 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Memo1: TMemo;
  Label1: TLabel;
  ComboBox1: TComboBox;
  procedure FormCreate(Sender: TObject);

```

```

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

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

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

procedure TForm1.Button2Click(Sender: TObject);
begin
    minn:=a[1];
    maxx:=a[1];
    For k:=1 to 10 do
        Begin
            If minn>a[k] Then Minn:=a[k];
            If maxx<a[k] Then maxx:=a[k];
        End;
    Str(maxx:5,S1);
    Str(minn:5,S2);

```

```
Memo1.Clear;  
Memo1.Lines.Add('Eng kattasi=' +s1);  
Memo1.Lines.Add('Eng kichigi =' +s2);  
end;  
End.
```

## 4.5.StringGrid jadval komponentasi

**StringGrid** jadval komponentasi ikki o'lovli ma'lumotlarni, masalan, matritsa elementlarini qiymatini ekranda jadval ko'rinishida tasvirlash, ular qiymatini kiritish va tahrirlash uchun ishlatiladi. Jadvalning qator va ustun nomerlari noldan boshlanadi. Jadvalning ustun va qatorlar sonini keraklicha o'zgartirish mumkin. Bu uning xossasi yordamida aniqlanadi. Jadvalning har bir kesishgan ustun va satri yacheyka deyilib, unga kiritilgan ma'lumot simvol qatori bo'lib aniqlanadi. Masalan, (3,5) yacheyka to'rtinchi ustun va oltinchi qatorda joylashgan.

**StringGrid** jadval komponentasining asosiy xossalari:

**ColCount** – jadvaldagi ustunlar sonini aniqlaydi;

**RowCount** – jadvaldagi satrlar sonini aniqlaydi;

**FixedCols** – fiksirlangan ustunlar sonini aniqlaydi;

**FixedRows** – fiksirlangan satrlar sonini aniqlaydi;

**Options** –jadval holatini aniqlaydi (aniqlash uning parametrlariga asosan bajariladi, masalan **GoEditing** parametr true qiymatiga ega bo'lsa yacheykani tahrirlash mumkin, aks holda mumkin emas. Bu parametrlarni aniqlash uchun **Options** xossasiga o'tib, u ikki marta tez-tez bosiladi);

**ColWidths** – jadvaldagi har bir ustun kengligini aniqlaydi;

**DefaultColWidth** – jadvalning boshlang'ich ustunlar kengligini aniqlaydi;

**DefaultRowHeight** – jadval satrining boshlang'ich balandligini aniqlaydi;

**FixedColor** – fiksirlangan yacheyka rangini aniqlaydi;

**RowHeights** – jadval satri balandligini aniqlaydi;

**Cells** – simvol qatorli ikki o'lchamli massivni aniqlaydi.

### **M i s o l 6.**

Butun qiymatli A(4,4) massiv elementlari yig'indisi va o'rta arifmetik qiymati topilsin.

## Ye'chish

1. Yangi ilova yaratamiz.

2. Formaga Additional komponentalar palitrasidan **StrinGrid** komponentasini StrinGrid1 nom bilan, Standart komponentalar palitrasidan Memo komponentasini Memo1 nom bilan va Botton1 tugmalarini o'ratamiz.

3. StrinGrid komponentasining xossalarini o'ratamiz:

FixedCols - 0;

FixedRows - 0;

ColCount - 4;

RowCount - 4;

Demak, hosil qilinadigan jadval 4 ta ustun va 4 ta satrga ega.

Option xossasiga kiramiz va uni ikki marta tez-tez chiqillatamiz.

U yerdan GoEditing parametrini True qiymatiga tenglashtiramiz.

4. Botton1 tugmasining Coption xossasiga kirib, uning nomini «Yechish» nomiga o'zgartiramiz.

5. «Yechish» tugmasini aktivlashtiramiz, ya'ni, uni ikki marta tez-tez bosib dastur kodlarini yozish oynasiga o'ratamiz va quyidagi kodlarni kiritamiz:

```
Var i,j,cod:integer;
```

```
A:array[1..4,1..4] of Real;
```

```
S:real; s1:String;
```

```
begin
```

```
For i:=1 to 4 do
```

```
For j:=1 to 4 do
```

```
Val(StringGrid1.cells[i-1,j-1],a[i,j],cod);
```

```
S:=0;
```

```
For i:=1 to 4 do
```

```
For j:=1 to 4 do
```

```
s:=s+a[i,j];
```

```
Str(s:7:2,s1);
```

```
Memo1.Clear;
```

```
Memo1.Lines.add('Summa =' +s1);
```

```
s:=s/4/4;
```

```
Str(s:7:2,s1);
```

```
Memo1.Lines.add('O'rtacha=' +s1);
```

```
end;
```

6. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modulning nomlarini mos nomlar bilan almashtirib saqlaymiz.

7. Yangi nom bilan saqlangan proyekt, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi.

Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

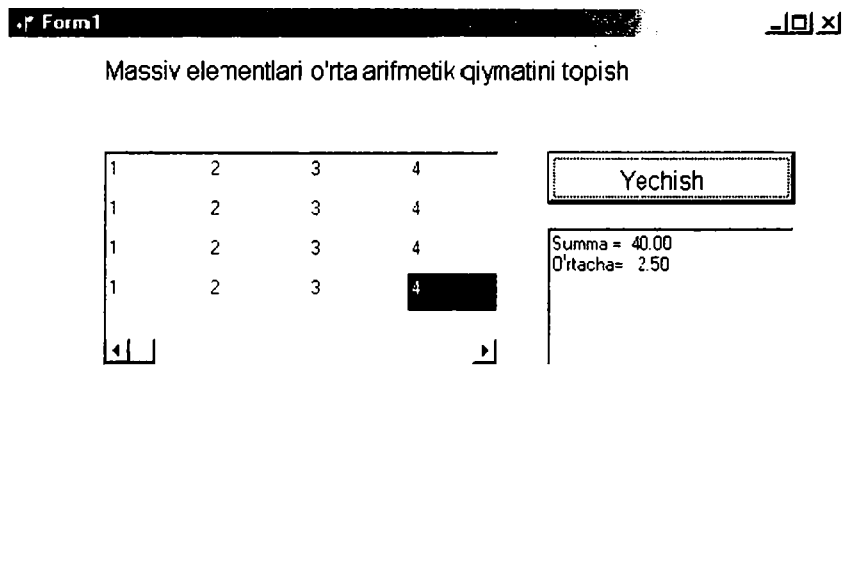
**Unit j1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,**

**Dialogs, Grids, StdCtrls;**



Type

**TForm1 = class(TForm)**

**StringGrid1: TStringGrid;**

**Button1: TButton;**

**Label1: TLabel;**

**Memo1: TMemo;**

**procedure Button1Click(Sender: TObject);**

**private**

**{ Private declarations }**

**public**



```

    { Public declarations }
end;

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

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

```

## 4.6.Muloqot oynalarini yaratish

Windows operatsion tizimi bir qancha standart muloqot oynalariga ega. Bu oynalar misoliga fayllarni ochish va saqlash, shriftlarni tanlash va to‘g‘rilash, rang berish, printerni boshqarishni keltirish mumkin. Delphi sistemasi ham bu muloqot oynalarini qo‘llaydi.

### Kiritish oynasi

Kiritish oynasi – standart dialog oynasi bo‘lib, inputBox funksiyasini chaqirish natijasida ekranga chiqariladi. InputBox funksiyasi qiymati – foydalanuvchi kiritgan qatordir.

Umumiy holda InputBox funksiyasini chaqirish quyidagi ko‘rinishga ega:

```

O‘zgaruvchi := InputBox(Sarlavha, Izoh, Qiymat);
Quyidagi rasmda dialog oynasining ko‘rinishi keltirilgan.

```

## Og'irlikni kilogrammda kiriting



OK

Cancel

Bu kiritish oynasini dasturga quyidagi instruksiya orqali chiqarilishi mumkin:

```
s:=InputBox('Kilogramm-gramm', og'irlikni kilogrammda kiriting', '0');
```

Agar dastur bajarilishi davomida foydalanuvchi qator kiritib OK tugmasini bossa InputBox funksiyasi qiymati kiritilgan qatorga teng bo'ladi. Agar Cancel tugmasi bosilsa funksiya qiymati funksiyaga parametr sifatida berilgan satrga teng bo'ladi.

Shuni ta'kidlash lozimki inputBox funksiyasi qiymati satr (string) turiga tegishli. Shuning uchun dasturga son qaytarish lozim bo'lsa, mos o'zgartirish funksiyasidan foydalanish lozim. Masalan:

```
s:=InputBox('Kilogramm-gramm', 'Og'irlikni kilogrammda kiriting', '0');
```

```
g := StrToFloat(s);
```

### Ma'lumot oynasini chiqarish

Ekranga ma'lumot oynasini chiqarish uchun ShowMessage protsedurasidan yoki MessageDlg funksiyasidan foydalanish lozimdir.

ShowMessage protsedurasi ekranga matnli hamda OK buyruq tugmasiga ega bo'lgan ma'lumot oynasini chiqaradi.

ShowMessage protsedurasini chaqirish instruksiyasi quyidagi ko'rinishga ega:

```
ShowMessage(Ma'lumot);
```

Quyidagi rasmda keltirilgan instruksiyani bajarish natijasida ekranda aks etuvchi ma'lumot oynasi ko'rsatilgan:

```
ShowMessage(«Og'irlikni kilogrammda kiriting»);
```

Ma'lumot oynasining sarlavhasida Project Options oynasining Application bo'limida ko'rsatilgan ilova nomi aks etadi. Agar ilova nomi berilmagan bo'lsa sarlavhada bajarilayotgan fayl nomi aks etadi.

## Og'irlikni kilogrammda kiriting



MessageDlg funksiyasi universal xarakterga egadir. Bu funksiya ma'lumotli oynaga standart belgilardan birini, masalan «Внимание», buyruq tugmalarining sonini va turini berishga hamda foydalanuvchi qaysi tugmani bosganligini aniqlashga imkon beradi. Rasmda quyidagi instruksiyaning bajarilish natijasi keltirilgan.

```
r:=MessageDlg( ' Fayl o'chiriladi.', mtWarning,
[mbOk,mbCancel],0);
```

**Warning****X****Fayl o'chiriladi****Cancel**

MessageDlo funksiyasining qiymati qaysi buyruq tugmasi bosilganligini aniqlashga imkon beruvchi sonidir.





MessageDlo funksiyasiga murojaatning umumiy ko'rinishi quyidagichadir:

Tanlov: = MessageDlg (Ma'lumot, Tur, Tugmalar, Kontekst Spravkalar)

Bu yerda:

- Ma'lumot – ma'lumot matni;
- Tur – ma'lumot turi. Ma'lumot informatsion, ogohlantiruvchi yoki kritik xato haqidagi ma'lumot bo'lishi mumkin. Har bir ma'lumot turiga ma'lum belgi mos keladi. Ma'lumot turi nomlangan konstanta bilan beriladi.

### MessageDlg funksiyasi konstantalari:

Konstanta	Ma'lumot turi	Belgi
mtWarning	Diqqat	
mtError	Xato	
mtInformation	Ma'lumot	
mtConfirmation	Tasdiqlash	
mtCustom	Oddiy	Belgisiz

• Tugmalar — ma'lumot oynasida aks etuvchi tugmalar ro'yxati. Ro'yxat nomlangan konstantalardan iborat bo'ladi.

### MessageDlg funksiyasi konstantalari:

Konstanta	Tugma	Konstanta	Tugma
mbYes	Yes	MbAbort	Abort
mbNo	No	mbRetry	Retry
mbOK	OK	MbIgnore	Ignore
mbCancel	Cancel	mbAll	All
mbHelp	Help		

Masalan, ma'lumot oynasida OK va Cancel tugmalari paydo bo'lishi uchun tugmalar ro'yxati quyidagicha berilishi lozim:

[mbOK,mbCancel]

Keltirilgan tugmalardan, konstantalardan tashqari eng ko'p qo'llanadigan konstantalar: mbOkCancel, mbYesNoCancel va mbAbortRetryIgnore.


• kontekstSpravkalar — foydalanuvchi <F1> tugmasini bosganda ekranda paydo bo'luvchi spravka tizimining bo'limidir. Agar bu parametr qiymati nolga teng bo'lsa spravka ekranga chiqarilmaydi.

Quyidagi jadvalda MessageDlg qaytarishi mumkin bo'lgan qiymatlar va ularga mos buyruq tugmalari berilgan.

### MessageDlg funksiyasi konstantalari:

MessageDlg funksiyasi konstantalari	Bosilgan tugma
mrAbort	Abort
mrYes	Yes
mrOk	Ok
mrRetry	Retry
mrNo	No
mrCancel	Cancel
mrIgnore	Ignore
mrAll	All

Delphi tizimida muloqot oynalarini qo'llash uchun maxsus **Dialogs** nomli komponentalar palitrasi mavjud bo'lib, u o'z ichiga bir necha vizual bo'lmagan komponentalarni oladi. Ulardan **OpenDialog**, **SaveDialog** va **FontDialog** komponentalarini ko'rib chiqamiz.

**OpenDialog** komponentasi kompyuter fayl tizimini ko'rish va undan kerakli fayl nomini tanlash imkonini beradi. Bu komponenta piktogrammasi  ko'rinishga ega. U vizual bo'lmagan komponenta bo'lib, uni formaga sichqonchada bir marta bosib qo'yiladi va keyin uning xossalari o'rnatiladi.

Uning asosiy xossalari ko'rib chiqamiz:

DefaultExt – faylning kengaytma nomini saqlaydi.

FileName – tanlangan fayl nomini saqlaydi.

Filter – fayl nomlarini muloqot darchasiga ko'rsatilgan kengaytma nom bo'yicha filtrlab chiqaradi. Masalan, agar .pas ko'rsatilgan bo'lsa muloqot oynasida faqat .pas kengaytmali fayllar chiqadi.

Filter xossasiga o'tilib uch nuqtali tugmacha bosilsa, Filter Editor muloqot oynasi chiqadi. U ikki qismdan iborat bo'lib, birinchi qismida filtr matni, ikkinchi qismida esa filtrning o'zi beriladi.

Masalan:

filtr matni nomlari:

Файлы модулей Delphi (\*.pas)

Текстовые документы (\*.txt,\*.doc)

Все файлы (\*.\*)

va boshqa

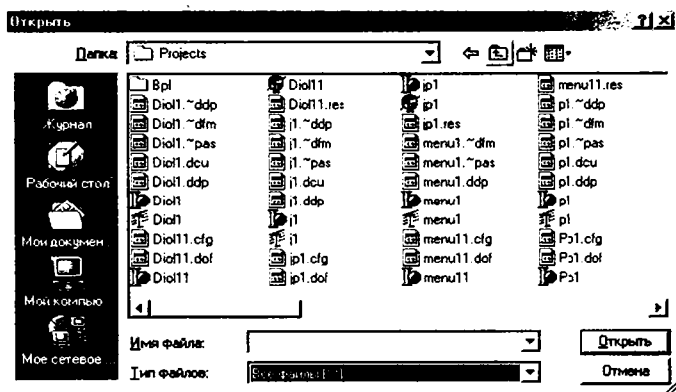
mos filtrlar:


\*.pas

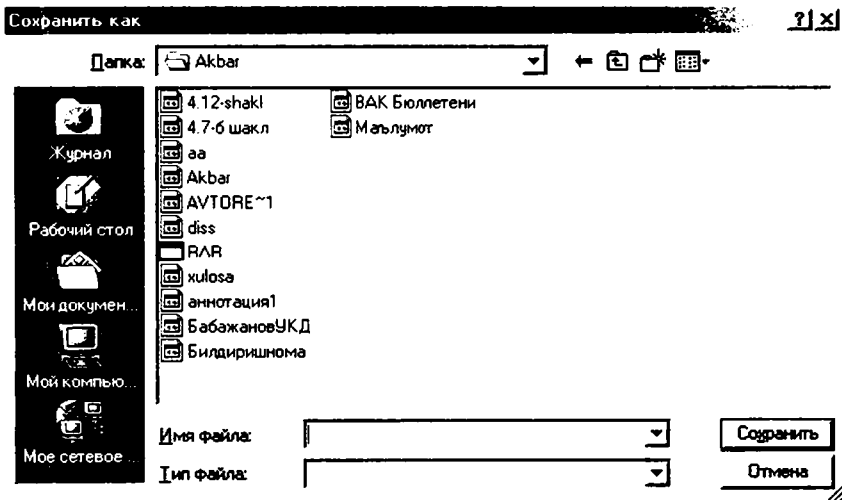
\*.txt; \*.doc


\*\*

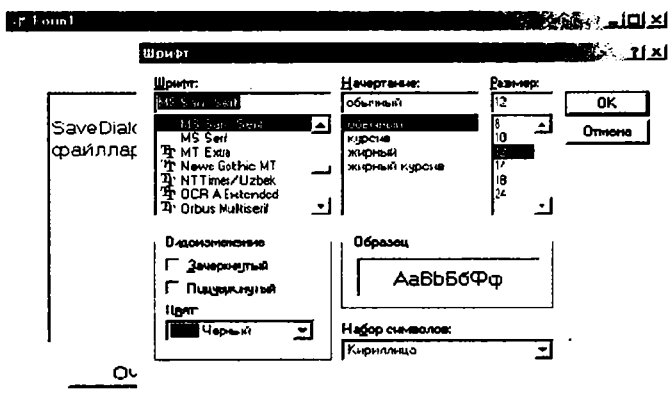
Faylni ochishda muloqot oynasining ko'rinishi.



**SaveDialog** komponentasi kompyuter xotrasiga fayllarni saqlash imkonini beradi. Bu komponenta piktogrammasi  ko'rinishga ega. U vizual bo'lmagan komponenta bo'lib, uni formaga sichqonchada bir marta bosib qo'yiladi va keyin uning xossalari o'rnatiladi. Agar uning DefaultExt xossasi qiymati .txt qilib tenglashtirilsa, faylni saqlashda avtomatik ravishda uning kengaytmasi .txt qilinib saqlanadi. Faylni saqlashda muloqot oynasining ko'rinishi.



**FontDialog** komponentasi foydalanuvchiga shriftlarni tanlaydi va uning xarakteristikasini belgilaydi. Bu komponenta  piktogrammasi ko'rinishga ega. U vizual bo'lmagan komponenta bo'lib, uni formaga sichqonchada bir marta bosib qo'yiladi va keyin uning xossalari o'rnatiladi. Uning Font xossasi shrift xarakteristikasini beradi. Shriftni tanlashda muloqot oynasining ko'rinishi.

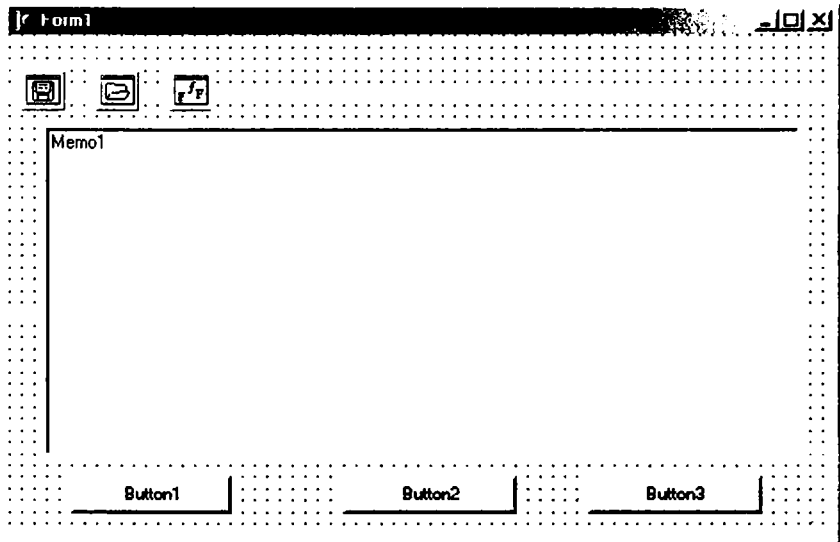


## Misol 7.

**OpenDialog, SaveDialog** va **FontDialog** komponentalarini ishlatgan holda oddiy matn muharriri yaratilsin.

### Ye ch i sh

1. Yangi ilova yaratamiz.
2. Formaga matnlarni chiqarish uchun Standart komponentalar palitrasidan Memo komponentasini Memo1 nom bilan o‘rnatamiz.
3. Forma yuqorisiga OpenDialog, SaveDialog va FontDialog komponentalarini o‘rnatamiz. Bu komponentalarni vizual bo‘lmaganligi sabab, istalgan joyga o‘rnatasa bo‘ladi. Chunki dastur ishlashi vaqtida bu komponentalar ko‘rinmaydi.
4. Formaning pastki qismiga Standart komponentalar palitrasidan Botton komponentasini uch marta Botton 1, Botton 2 va Botton 3 nomlar bilan o‘rnatamiz.

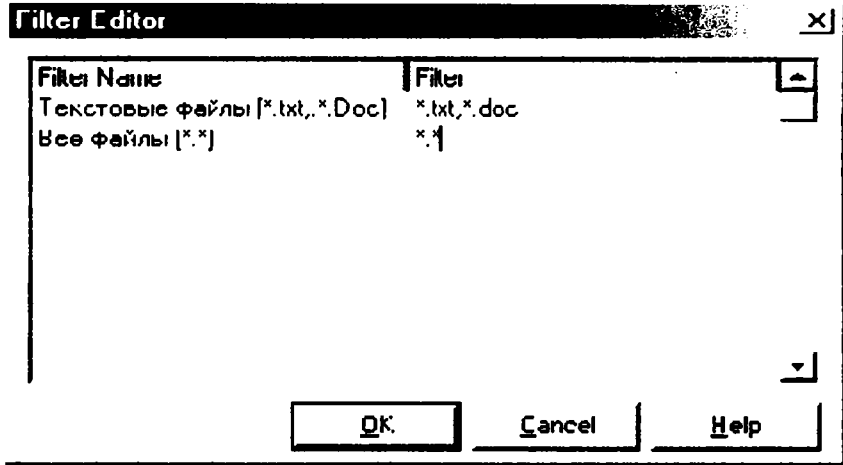


5. Memo1 komponentasining Lines xossasiga kelib, uch nuqtali tugmachani bosamiz va muloqot oynasidan Memo1 so‘zini o‘chiramiz va Ok tugmasini bosamiz. (Bu degani ilovani ishga tushirganda muharrir oynasi bo‘sh chiqadi.)

6. OpenDialog komponentasi xossalarini o‘rnatamiz. Buning uchun Filter xossasiga kirib, uning uch nuqtali tugmasini bosamiz. Hosil bo‘lgan Filter Editor muloqot darchasiga quyidagilarni kiritamiz va Ok tugmasini bosamiz.

Filter Name qismiga

Текстовые документы (\*.txt, \*.doc)  
Все файлы (\*.\*)  
Filter qismiga  
\*.txt; \*.doc  
\*.\*



7. SaveDialog komponentasi xossalarini o'rnatamiz. DefaultExt xossasi qiymatini .txt qilib tenglashtiramiz.

8. Botton1, Botton 2 va Botton 3 tugmachalar nomlari ularning Caption xossasiga kirib, mos ravishda «Ochish», «Saqlash» va «Shrift» nomlariga o'zgartiramiz.

9. Botton1 tugmasini ikki marta tez-tez bosib, dastur kodlarini kiritish darchasiga o'tib, quyidagi operatorlarni kiritamiz:

```
With Opendialog1 do
  Begin
    If not Execute then Exit;
    Memo1.Lines.LgadFromFile(Filename)
  End;
```

10. Botton 2 tugmasini ikki marta tez-tez bosib, dastur kodlarini kiritish darchasiga o'tib quyidagi operatorlarni kiritamiz:

```
With Savedialog1 do
  Begin
    If not Execute then Exit;
    Memo1.Lines.SaveToFile(Filename);
  End;
```



11. Botton 3 tugmasini ikki marta tez-tez bosib, dastur kodlarini kiritish darchasiga o'tib quyidagi operatorlarni kiritamiz.

**With Fontdialog1 do**

**Begin**

**If not Execute then Exit;**

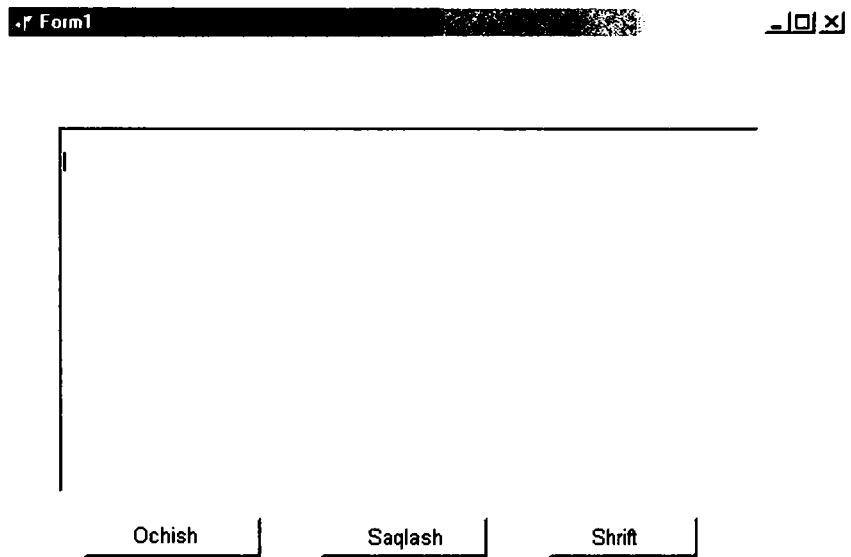
**Memo1.Font:=Font;**

**End;**

12. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modulning nomlarini mos nomlar bilan almashtirib saqlaymiz.

13. Yangi nom bilan saqlangan loyiha, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi.



Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

**Unit Diol1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,**

**Dialogs, StdCtrls, Menus;**

**Type**

```

TForm1 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Memo1: TMemo;
  OpenDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  FontDialog1: TFontDialog;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

Var
  Form1: TForm1;

implementation
{SR *.dfm}
Procedure TForm1.Button1Click(Sender: TObject);
begin
  With Opendialog1 do
    Begin
      If not Execute then Exit;
      Memo1.Lines.LoadFromFile(Filename)
    End;
end;

Procedure TForm1.Button2Click(Sender: TObject);
begin
  With Savedialog1 do
    Begin
      If not Execute then Exit;
      Memo1.Lines.SaveToFile(Filename);
    End;
end;


procedure TForm1.Button3Click(Sender: TObject);

```


```
begin
  With Fontdialog1 do
    Begin
      If not Execute then Exit;
      Memo1.Font:=Font;
    End;
  end;
end.
```

#### 4.7. Ilovalar uchun menyu yaratish

Ko'pchilik ilovalar bosh menyuga ega bo'lib, bajariladigan operatsiyalar ro'yxatini o'z ichiga oladi. Bosh menyu punktlari nolinch darajadagi menyu elementlari deyiladi. Ularning har biriga bog'liq bo'lgan birinchi darajali menyu elementlarini o'z ichiga olishi mumkin.

Delphida bosh menyu tashkil qilish uchun maxsus vizual bo'lmagan MainMenu komponentasi mavjud. Bu komponenta Standart komponentalar palitrasida joylashgan bo'lib, u  belgili piktogrammaga ega.

MainMenu komponentasining asosiy xossasi Items xossasidir. U o'zida ilova bosh menyusining nolinch darajali elementlarini saqlaydi.

Ma'lumki Windows sistemasida sichqonchani o'ng tugmasi bosilganda yordamchi menyu chiqadi. Bu menyuga «контекстное меню» deyiladi. Delphida bunday kontekstli menyuni tashkil qilish uchun maxsus vizual bo'lmagan PopupMenu komponentasi mavjud. Bu komponenta Standart komponentalar palitrasida joylashgan bo'lib, u  belgili piktogrammaga ega.

#### M i s o l 8.

File, Edit va Run tuzilmaga ega bosh menyu va ular tanlanganda unga mos punktlarga o'tish dastur ilovasi yaratilsin.

#### Ye ch i sh

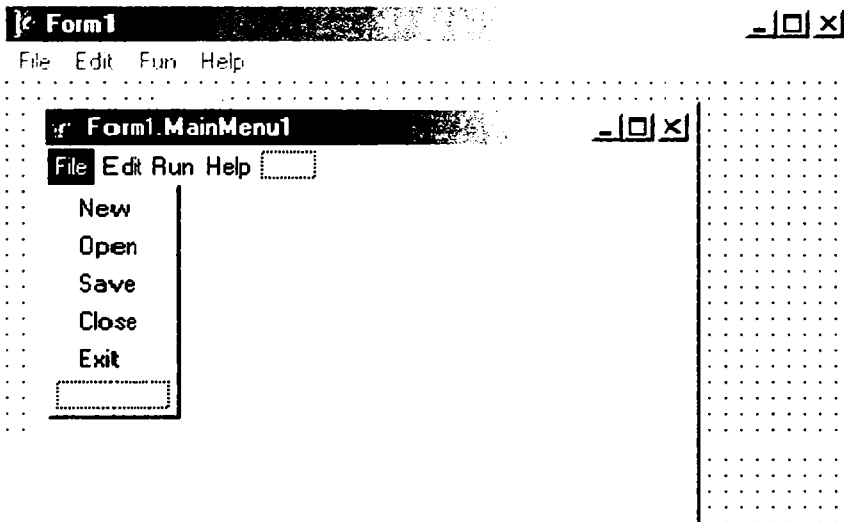
1. Yangi ilova yaratamiz.

2. Formaga Standart komponentalar palitrasidan **MainMenu** komponentasini MainMenu1 nom bilan o'ratamiz. Buning uchun bu komponenta ikki marta tez-tez bosiladi.

3. MainMenu1 komponentasiga Items xossasini o'ratamiz. Buning uchun Items xossasiga kirib, uning uch nuqtali tugmasini bosamiz. Natijada ekranda menyu konstruktorining muloqot oynasi chiqadi.

4. Menyua konstruktori yordamida File, Edit, Run bosh menyua nomlarini va unga mos ularning buyruqlarini (New, Open, Save va

boshqa elementlarini) kiritamiz. Buning uchun Caption xossasiga o'tilib va kerakli buyruq yoziladi.



5. Menyuning konstruktor oynasidan chiqamiz. Menyu punktlarini tanlaganda ularning reaksiyasini (ta'sirini) aniqlash uchun ularning har birini sichqonchada bir marta chiqillatamiz. Natijada, kodlarni kiritish muharriri oynasi chiqadi va u yerdan jarayonlarni qayta ishlash (On Click) uchun kerakli dastur kodlarini kiritamiz. Masalan, har bir punktga kirganligi haqida ma'lumot chiqaramiz. Ma'lumotni chiqarish uchun maxsus standart funksiya mavjud bo'lib, uning nomi ShowMessage deyiladi. New punkti uchun bu dastur kodi quyidagicha bo'ladi.

```
Procedure TForm1.New1Click(Sender: TObject);  
begin  
    ShowMessage('New punkti');  
end;
```

6. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modulning nomlarini mos nomlar bilan almashtirib saqlaymiz.

7. Yangi nom bilan saqlangan proyekt, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda uning quyidagi ko'rinishi ekranda namoyon bo'ladi.

Tashkil qilingan modulning to'liq ko'rinishini keltiramiz.

```
Unit menu1;  
interface  
uses  
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
    Dialogs, Menus;  
type  
TForm1 = class(TForm)  
    MainMenu1: TMainMenu;  
    File1: TMenuItem;  
    Edit1: TMenuItem;  
    Run1: TMenuItem;  
    New1: TMenuItem;  
    Open1: TMenuItem;  
    Save1: TMenuItem;  
    Close1: TMenuItem;  
    Cut1: TMenuItem;  
    Copy1: TMenuItem;  
    Past1: TMenuItem;  
    Delete1: TMenuItem;  
    RunF91: TMenuItem;  
    StepOver1: TMenuItem;  
    Exit1: TMenuItem;  
    Help1: TMenuItem;
```

```

procedure New1Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure Save1Click(Sender: TObject);
procedure Close1Click(Sender: TObject);
procedure Cut1Click(Sender: TObject);
procedure Copy1Click(Sender: TObject);
procedure Past1Click(Sender: TObject);
procedure Delete1Click(Sender: TObject);
procedure RunF91Click(Sender: TObject);
procedure StepOver1Click(Sender: TObject);
procedure Help1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;

implementation
  {$R *.dfm}

Procedure TForm1.New1Click(Sender: TObject);
Begin
  ShowMessaoe('New punkti');
end;

Procedure TForm1.Open1 Click(Sender: TObject);
Begin
  ShowMessage('Open punkti');
end;

Procedure TForm1.Save1Click(Sender: TObject);
Begin
  ShowMessage('Save punkti');
end;

Procedure TForm1.Close1 Click(Sender: TObject);
Begin
  ShowMessage('Close punkti');
end;

```

```
procedure TForm1.Cut1Click(Sender: TObject);  
begin  
    ShowMessage('New punkti');  
end;
```

```
procedure TForm1.Copy1Click(Sender: TObject);  
begin  
    ShowMessage('New punkti');  
end;
```

```
Procedure TForm1.Past1Click(Sender: TObject);  
Begin  
    ShowMessage('New punkti');  
end;
```

```
Procedure TForm1.Delete1Click(Sender: TObject);  
Begin  
    ShowMessage('New punkti');  
end;
```

```
Procedure TForm1.RunF91Click(Sender: TObject);  
Begin  
    ShowMessage('New punkti');  
end;
```

```
procedure TForm1.StepOver1Click(Sender: TObject);  
begin  
    ShowMessage('New punkti');  
end;
```

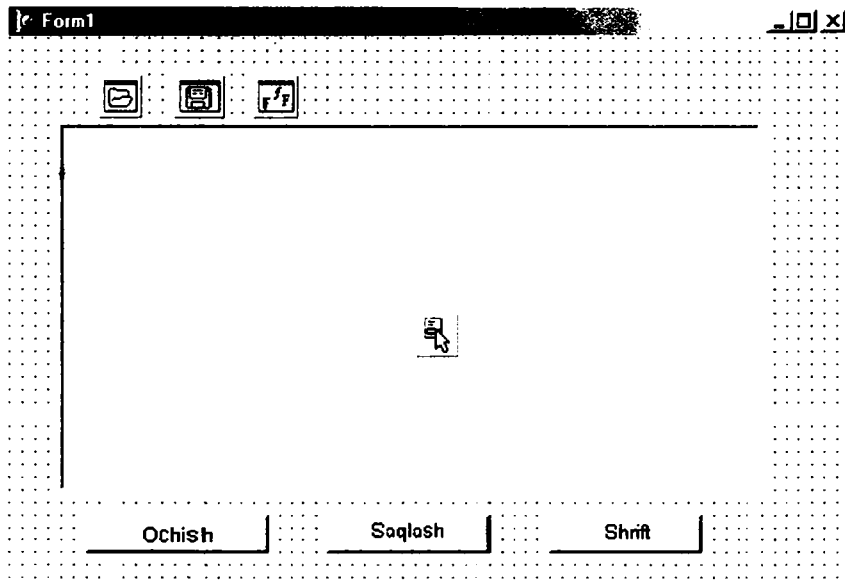
```
procedure TForm1.Help1Click(Sender: TObject);  
begin  
    ShowMessage('New punkti');  
end;  
end.
```

### **M i s o l 9.**

Open, Save va Font tuzilmasiga ega kontekstli menyu (PopupMenu) yaratilsin. Yuqoridagi 8-misoldagi loyiha uchun qo‘shimcha kontekstli menyu punktlarining dasturi tuzilsin.

## Ye ch i sh

1. Yangi ilova yaratamiz.
2. 8-misoldagi formaga Standart komponentalar palitrasidan **PopupMenu** komponentasini ham **PopupMenu1** nom bilan o'ratamiz. Buning uchun bu komponenta ikki marta tez-tez bosiladi.



3. **PopupMenu1** komponentasining **Items** xossasiga o'tamiz va uning uch nuqtali tugmasini bosamiz. Kontekst menyu elementlarini kiritamiz: **Open**, **Save** va **Font**.

4. Menyu konstruktoridan chiqmasdan **Open**, **Save** va **Font** buyruqlari uchun dastur kodi muharririni chaqiramiz. Ularning har biri uchun quyidagi dastur kodlarini kiritamiz:

```
Procedure TForm1.Open1Click(Sender: TObject);  
Begin  
    Button1Click(Button1);  
end;
```

```
Procedure TForm1.Save1Click(Sender: TObject);  
Begin  
    Button2Click(Button2);  
end;
```



**Procedure TForm1.Font1Click(Sender: TObject);**

**Begin**

**Button3Click(Button3);**

**end;**

**Izoh!** Bu yerga qora yozilgan dastur kodlari kiritiladi. Qolganlari o'zi mavjud. Bu qora yozilgan kodlar mos modul protseduralariga murojaatni anglatadi. Bu modul protseduralari oldin «Ochish», «Saqlash» va «Shrift» tugmalari uchun yozilgan. Bu murojaatlarni yozmasdan ular o'rninga «Ochish», «Saqlash» va «Shrift» tugmalari uchun yozilgan dastur kodlarini yozsa ham bo'ladi.

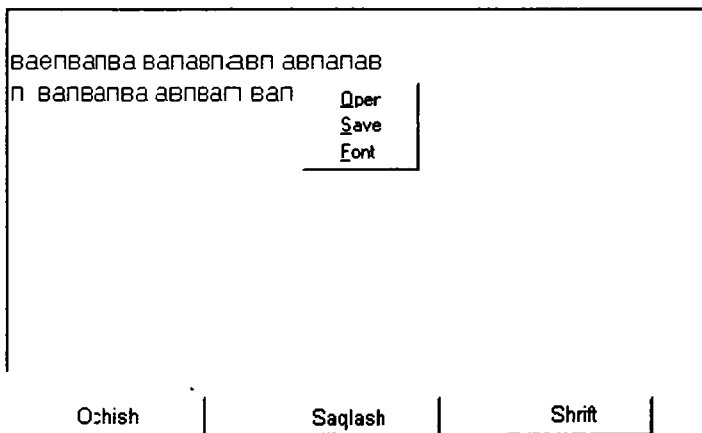
5. Tuzilgan loyiha (proyekt) ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlaymiz.

6. Yangi nom bilan saqlangan loyiha, F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilgandan keyin sichqonchanning o'ng tugmasi bosilsa u quyidagi ko'rinishda ekranda namoyon bo'ladi.

Form1

[-] [ ] [X]



#### **4.8. Bir necha formalar bilan ishlash**

Loyihada bir necha formalar bilan ishlashni misolda ko'rib chiqamiz. Yangi loyiha yaratib, formaga boshqarish tugmasi (Button1) va ilova komponentasini (Label1) joylashtiramiz. Boshqarish tugmasining Click hodisasiga quyidagi kodni kiritamiz:

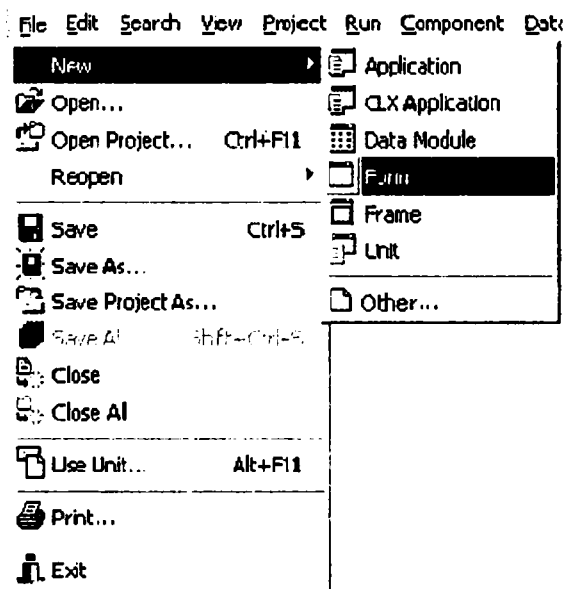
```

Procedure TForm1.Button1Click(Sender: TObject);
Begin
Form2.Show;
Label1.Caption:='Ko'p formali loyiha';
end;

```

Endi loyihaga yangi forma qo'shamiz. Buning uchun Menyu File bo'limida n New punktini, so'ngra Form punktini.

Loyihalar menejerini (View ->Project Manager) ochib Project1.exe loyihamizda ikki forma *Unit1* i *Unit2* borligini ko'rishimiz mumkin. Formalarning biriga ikki marta chertilsa, tizim oynasida shu formani o'zgartirish mumkin bo'ladi.

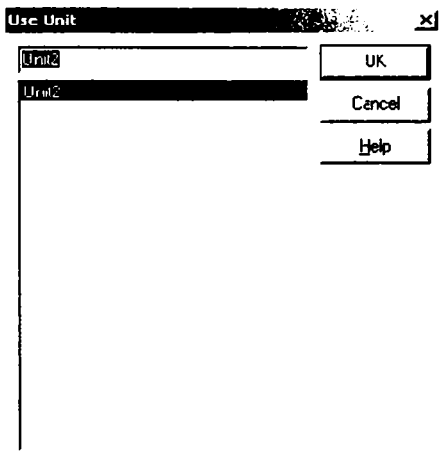


Ikkinchi formaga boshqarish tugmasi (Button1) va tahrirlash qatorini (Edit1) joylashtiramiz. Boshqarish tugmasining Click hodisasiga quyidagi kodni kiritamiz:

```

Procedure TForm2.Button1Click(Sender: TObject);
Begin
Form1.Label1.Caption:=Edit1.Text;
Form2.Close;
end;

```

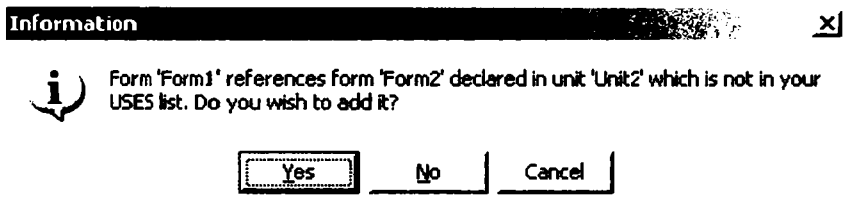


Agar loyihani kompilyatsiya qilsak, xato haqida ma'lumot chiqadi. Chunki ikkinchi forma Unit 2 da ta'riflangan, biz uni birinchi formada ishga tushir-yapmiz. Birinchi formaga o'tib, *File* menyusidan *Use Unit* punktini tanlaymiz. Quyidagi oyna ochiladi:

Bu oynadan kerakli Unit tanlab, OK tugmasini bosamiz. Quyidagi kod qo'shiladi:

```
var  
  Form1: TForm1;  
implementation  
uses Unit2;
```

Bu kodni qo'lga kiritish ham mumkin. Bundan tashqari, agar modul qo'shilmagan bo'lsa, loyihani ishga tushirishda quyidagi ma'lumot oynasi chiqadi:



Agar «Yes» tugmasi bosilsa, modul avtomatik ravishda qo'shiladi.

Agar biz dasturni ishga tushirsak, ekranda birinchi formani aktiv holda ko'ramiz. Agar boshqarish tugmasi bosilsa, ekranda ikkinchi forma aktiv holda paydo bo'ladi. Shu bilan birga, Label1 ustidagi yozuv ham o'zgaradi, ya'ni Caption xossasi dasturda ko'rsatilgan qiymatni oladi. Ikkinchi formada tahrirlash qatoriga biror satr kiritib, boshqarish tugmasini bossak, bu forma bekiladi va Label1 komponentasi Caption xossasiga biz kiritgan satr qiymat sifatida beriladi.

Endi birinchi formadagi boshqarish tugmasining Click hodisasiga quyidagi kodni kiritamiz:

```

Procedure TForm1.Button1Click(Sender: TObject);
Begin
Form2.ShowModal;
Label1.Caption:='Ko'p formalı loyiha';
end;

```

Ikkinchi formadagi boshqarish tugmasining Click hodisasiga quyidagi kodni kiritamiz:

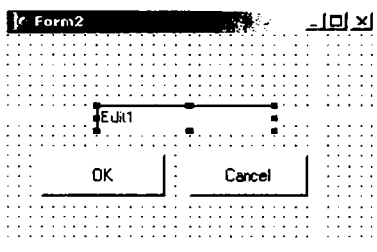
```

Procedure TForm2.Button1Click(Sender: TObject);
Begin
Form1.Label1.Caption:=Edit1.Text;
end;

```

Agar dasturni ishga tushirsak, birinchi forma aktiv shaklda ekranda paydo bo'ladi. Agar boshqarish tugmasini bossak, ikkinchi forma aktiv shaklda paydo bo'ladi, lekin Label1 ustidagi yozuv o'zgar olmaydi. Chunki biz ikkinchi formani modul rejimda ochdik. Boshqarish bu rejimda ochilgan formaga beriladi va faqat bu oyna berkitilganda boshqarish asosiy formaga qaytadi. Shu bilan birga tahrirlash matniga biror satr kiritib, boshqarish tugmasini bossak, bu satr asosiy formada aks etadi.

Boshqarish tugmalarining ModalResult xossalari mavjud bo'lib, ma'lumot almashishda foydalidir. Ikkinchi formaga yangi boshqarish tugmasini qo'shib, tugmalar nomlarini o'zgartiramiz.



Birinchi tugmaning ModalResult xossasiga *mrOk* qiymatini, ikkinchi tugmaning ModalResult xossasiga *mrCancel* qiymatini beramiz. Bu tugmalarining Click hodisasiga kiritilgan kodni tozalaymiz. Chunki ModalResult xossasiga qiymat berilishi, bu tugmalarni bosganda forma bekiilishiga olib keladi.

Endi birinchi formadagi boshqarish tugmasining Click hodisasiga quyidagi kodni kiritamiz:

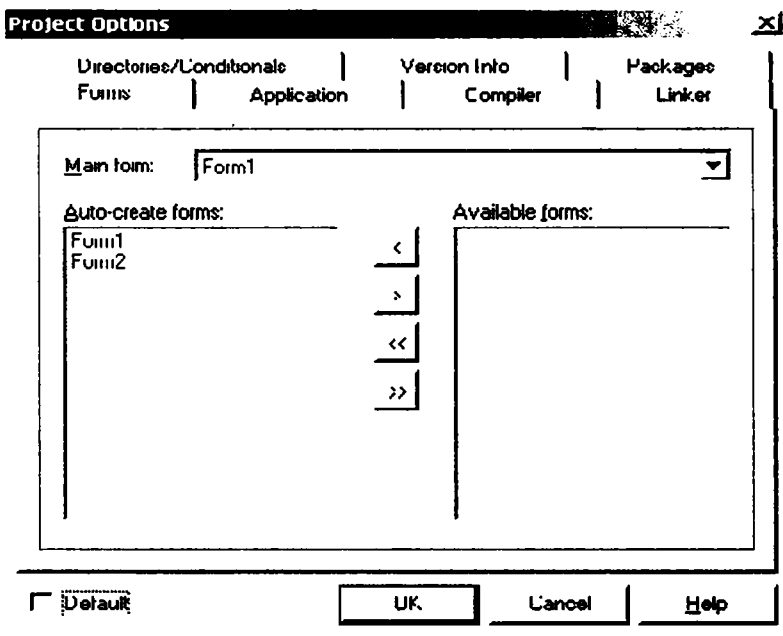
```

Procedure TForm1.Button1Click(Sender: TObject);
Begin
Form2.ShowModal=mrOK then
Label1.Caption:= Form2.Edit1.Text;
end;

```

Endi ikkinchi formada OK tugmasi bosilsa, birinchi formada satr aks etadi, aks holda aks etmaydi.

Loyiha ishga tushganda har doim birinchi forma ishga tushadi. Formalarning ishga tushirish ketma-ketligini o'zgartirish uchun menyuning Project bo'limidagi Options punktini tanlash lozim. Bunda quyidagi dialog oynasi paydo bo'ladi:



Bu oynada Main form qatorida ixtiyoriy formani asosiy forma sifatida tanlash mumkin. Mana shu tanlangan forma birinchi bo'lib ishga tushadi.

## Savollar

1. Label, Edit, Memo matn komponentlari va Button tugmachasi vazifalarini aytib bering.
2. Caption xossasi nima vazifani bajaradi?
3. RadioGroup guruhli tanlash tugmalariga tushuntirish bering.
4. CheckListBox komponentasi qanday vazifani bajaradi?
5. CheckListBox ning asosiy xossalarini aytib bering.
6. ListBox va ComboBox komponentalarining vazifasini tushuntiring.
7. StringGrid komponentasining vazifasi va asosiy xossalarini tushuntiring.
8. Muloqot oynalarini yaratish qanday amalga oshiriladi?

9. OpenFileDialog, SaveDialog va FontDialog komponentalari qanday funksiyalarni bajaradi va ularning qanday xossalari bilan bilasiz?
10. Delphida bosh menyu tashkil qilish qanday amalga oshiriladi?
11. Delphida bir necha formada ish yuritish qanday tashkil qilinadi?

## V. DELPHI MUHITIDA GRAFIKA VA MULTIMEDIA

### 5.1. Delphining grafik imkoniyatlari

Delphi dasturchiga grafik dasturlar sxemasi, chizma va illyustratsiyalar yaratishga imkon beradi. Dastur grafikani obyekt (forma yoki Image komponentasi) yuzasiga chiqaradi. Obyekt yuzasiga canvas xossasi mos keladi. Obyekt yuzasiga grafik element (to'g'ri chiziq, aylana, to'rtburchak va hokazo) chiqarish uchun bu obyektning canvas xossasiga mos usulni qo'llash lozim. Misol uchun Form1.canvas.Rectangle (10,10,100,100) instruksiyasi dastur oynasida to'rtburchak paydo bo'ladi.

#### Chizish sohasi

Yuqorida ko'rilgan canvas xossasi – TCanvas tipidagi obyektidir. Grafik primitivlarini chiqarish usullari Canvas xossasini abstrakt chizish sohasi deb qaraydi. Chizish sohasi alohida nuqtalar – piksellardan iborat. Pikel holati uning gorizont (X) va vertikal (Y) koordinatalari bilan aniqlanadi. Chap yuqori pikel koordinatalari (0,0). Koordinatalar yuqoridan pastga va chapdan o'ngga qarab o'sib boradi.

**Soha o'lchovlarini** image komponentasining Height va width xossalari va **formaning** ClientHeight va Clientwidth xossalari orqali aniqlash mumkin.

#### Qalam

Qalam geometrik figuralarni chizish uchun ishlatiladi. Chiziq ko'rinishi Tren obyektining quyidagi jadvalda ko'rsatilgan xossalari orqali aniqlanadi.

Tren (qalam) xossalari:

Xossa	Ta'rifi
Color	Chiziqning rangi
Width	Chiziqning qalinligi
Style	Chiziqning ko'rinishi
Mode	Akslantirish rejimi

Quyidagi jadvalda color xossasining qiymati sifatida beriluvchi nomlangan konstantalar sanab o'tilgan.

Color xossasining qiymatlari:

Konstanta	Rang	Konstanta	Rang
clBlack	Qora	clSilver	Kumush
clMaroon	Och jigarrang	clRed	Qizil
clGreen	Yashil	clLime	Och yashil
Olive	Och sariq	clBlue	Ko'k (zangori)
clNavy	Timko'k	clFuchsia	Och pushti
clPurple	Pushti	clAqua	Firuza
clTeal	Ko'kish havorang	clWhite	Oq
clGray	Kulrang		

Chiziq qalinligi width xossasi orqali piksellarda beriladi.

Chiziq turini style xossasi belgilaydi. Quyidagi jadvalda chiziq turini belgilovchi nomlangan konstantalar sanab o'tilgan.

Style xossasining qiymatlari:

Konstanta	Chiziq ko'rinishi
psSolid	Uzluksiz chiziq
psDash	Punktir chiziq, uzun shtrixlar
psDot	Punktir chiziq, qisqa shtrixlar
psDashDot	Punktir chiziq, uzun va qisqa shtrixlar ketma-ketligi
psDashDotDot	Punktir chiziq, bitta uzun va ikkita qisqa shtrixlar ketma-ketligi
psClear	Chiziq aks etmaydi

Mode xossasi chiziq rangining fon rangiga munosabatini ko'rsatadi. Odatda chiziq rangi Pen.Color xossasining qiymati bilan belgilanadi.

Dasturchi chiziq uchun fon rangiga nisbatan invers rang berishi mumkin. Bu holda hatto chiziq va fon rangi bir xil berilgan bo'lsa ham chiziq ajralib turadi.

Quyidagi jadvalda Mode xossasining qiymati sifatida ishlatish mumkin bo'lgan konstantalar berilgan.

Mode xossasining qiymatlari:

Konstanta	Chiziq rangi
pmBlack	Qora, Pen. Color xossasining qiymatiga bog'liq emas
pmWhite	Ok, Pen. Color xossasi qiymatiga bog'liq emas
pmCopy	Chiziq rangi Pen. Color xossasi qiymatiga bog'liq
pmNotCopy	Chiziq rangi Pen. Color xossasi qiymatiga invers
pmNot	Chiziq rangi sohaning mos nuqtasi rangiga invers

## Mo'yqalam

Mo'yqalam (Canvas.Brush) yopiq sohalarni chizish va soha ichini bo'yash uchun mo'ljallangan usullardan foydalaniladi. Mo'yqalam obykti jadvalda ko'rsatilgan ikki xossaga ega.

TBrush (mo'yqalam) xossalari:

Xossa

Ta'rifi

Color

Yopiq sohani bo'yash rangi

Style

Sohani to'ldirish uslubi

Kontur ichidagi soha bo'yalishi yoki shtrixlanishi mumkin.

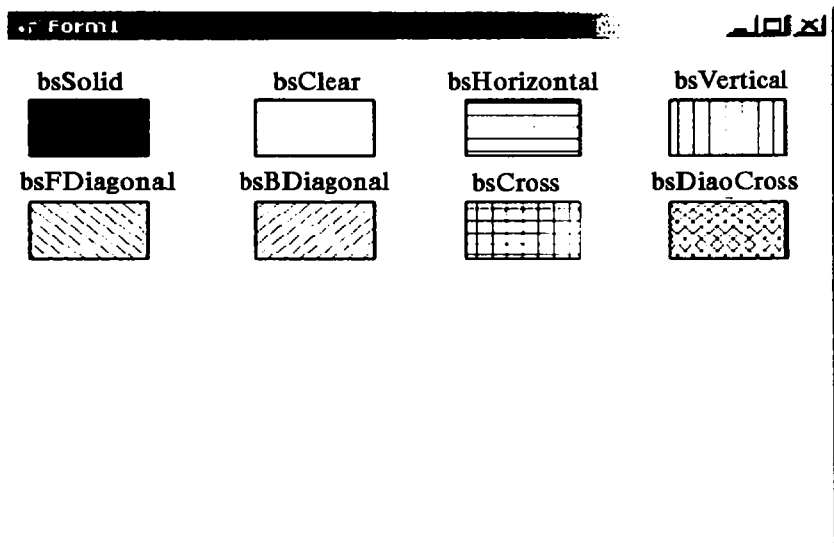
Sohani to'ldirish usulini belgilovchi konstantalar quyidagi jadvalda berilgan.

Brush.style xossasining qiymatlari:

Konstanta	Sohani bo'yash uslubi
bsSolid	Uzluksiz bo'yash
bsClear	Soha bo'yalmaydi
bsHorizontal	Gorizontal shtrixlash
bsVertical	Vertikal shtrixlash
bsFDiagonal	Diagonal shtrixlash, oldinga og'ish
bsBDiagonal	Diagonal shtrixlash, orqaga og'ish
bsCross	Katakli gorizontal-vertikal shtrixlash
bsDiaoCross	Katakli diagonalini shtrixlash

Misol tariqasida sohalarni bo'yash usullari dasturining keltiramiz:

Sohani bo'yash usullarining dastur oynasi.





Sohani bo'yash usullarining dastur matni:

**Unit Unit 1;**

**interface**

**uses**

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

**Type**

**TForm1 = class(TForm)**

**Button1: TButton;**

**procedure Button1Click(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation**

**{ \$R \*.dfm }**

**procedure TForm1.Button1Click(Sender: TObject);**

**Const**

**bsName: array[1..8] of string =**

**('bsSolid', 'bsClear', 'bsHorizontal',**

**'bsVertical', 'bsFDiagonal', 'bsBDiagonal',**

**'bsCross', 'bsDiaoCross');**

**Var**

**x,y: Integer;**

**w,h: Integer;**

**bs: TBrushStyle;**

**k: Integer;**

**i,j: Integer;**

**Begin**

**Button1.visible:=false;**

**w:=60; h:=40;**

**y:=80;**

**for i:=1 to 2 do**

**begin**

```

x:=10;
for j:=1 to 4 do
begin
k:=j+(i-1)*4;
case k of
1: bs:= bsSolid;
2: bs:= bsClear;
3: bs:= bsHorizontal;
4: bs:= bsVertical;
5: bs:= bsFDiagonal;
6: bs:= bsBDiagonal;
7: bs:= bsCross;
8: bs:= bsDiagCross; end;
Canvas.Brush.Color := clOreen;
Canvas.Brush.Style := bs;
Canvas . Rectangle (x, y, x+w, y-h) ;
Canvas.Brush.Style := bsClear;
Canvas.TextOut(x, y-55, bsName[k]);
x := x+w+30;
end;
y:= y+h+30;
end;
end;
end.

```

## Matni chiqarish

Grafik obyekt yuzasiga matn chiqarish uchun TextOut usuli qo'llaniladi. Bu usulni chaqirish instruksiyasi quyidagi ko'rinishga ega:

**Obyekt.Canvas.TextOut(x, u, Tekst)**

Matn shrifti Font xossasining qiymati bilan aniqlanadi. Font xossasi TFont tipidagi obyektidir. Quyidagi jadvalda TFont obyektining xossalari keltirilgan.

TFont obyektining xossalari:

Xossa	Ta'rifi
Name	Shrift nomi, masalan Arial
Size	Shriftning punktlardan kattaligi
Style	Simvollar chiqarish uslubi. Quyidagi konstantalar orqali beriladi: fsBold (полужирный), fsItalic (курсив), fsUnderline (подчеркнутый), fsStrikeOut (перечеркнутый).

Bu xossa bir necha uslublarni kombinatsiyasini olishga imkon beradi.  
Masalan: Obyekt. Canvas . Font := [fsBold, fs Italic]  
Color Simvollar rangi.

Matn chiqarish sohasi mo'yqalam joriy rangiga bo'yaladi. Shuning uchun matn chiqarishdan oldin Brush. Color xossasiga bsClear qiymatini yoki soha rangiga mos qiymatni berish lozim.

### M i s o l :

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

Textout uslubi orqali matn ekranga chiqarilgandan so'ng qalam matn chiqarish sohasining yuqori o'ng burchagiga keltiriladi.

Agar matn uzunligi ma'lum bo'lmasa, chiqarilgan matn o'ng chegarasi koordinatalarini PenPos xossasiga murojaat qilib aniqlash mumkin.

### M i s o l :

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

## Grafik primitivlarni chizish usullari

### Chiziq

To'g'ri chiziq LineTo usuli orqali amalga oshiriladi.

#### **Komponent.Canvas.LineTo(x,y)**

LineTo usuli qalamning joriy pozitsiyasidan berilgan koordinatali nuqtagacha to'g'ri chiziq chizadi. Boshlang'ich nuqtani kerakli nuqtaga ko'chirish uchun MoveTo usulidan foydalanish mumkin.

## Tutashgan chiziq

O'zaro tutashgan kesmalardan iborat shaklni chizish uchun polyline usulidan foydalaniladi. Bu usul parametri TPoint tipli massivdan iborat.

Polyline usuliga misol tariqasida ma'lum qiymat o'zgarishi grafigini chizuvchi protsedurasini keltiramiz:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
gr: array[1..50] of TPoint;  
x0,y0: Integer;  
dx,dy: Integer;  
i: Integer; begin  
x0 := 10; u0 := 200; dx :=5; dy := 5;  
for i:=1 to 50 do begin  
gr[i].x := x0 + (i-1)*dx;  
gr[i].y := y0 - Data[i]*dy;  
end;  
with form1.Canvas do begin  
MoveTo(x0,y0); LineTo(x0,10);  
MoveTo(x0,y0); LineTo(200,y0);  
Polyline(gr);  
end;  
end;
```

Polyline usuli yordamida yopiq ko'pburchak chizish uchun massivning birinchi va oxirgi elementi bir nuqtaning koordinatalaridan iborat bo'lishi kerak.

## Aylana va ellips

Aylana yoki ellips chizish uchun Ellipse usuli chaqiriladi. Usulni chaqirish instruksiyasining umumiy ko'rinishi:

**Object.Canvas.Ellipse(x1,y1, x2,u2)**

Bu yerda x1, y1, x2, u2 – ellipsni o'z ichiga olgan minimal to'rtburchak koordinatalari. Agar to'rtburchak kvadrat bo'lsa aylana chiziladi.

## Yoy

Yoyni chizish uchun Arc usuli qo'llaniladi va u quyidagi umumiy ko'rinishga ega:

### **Object.Canvas.Arc(x1,y1,x2,y2,y3,y3,x4,y4)**

Bu yerda:

- $x_1, y_1, x_2, y_2$  – yoyga tegishli bo‘lgan ellips yoki aylana parametrlari;
  - $x_3, y_3$  – yoyning boshlang‘ich nuqta parametrlari;
  - $x_4, y_4$  – so‘ngi nuqtali parametrlari.
- Yoy soat miliga teskari tartibda chiziladi.

### **To‘rtburchak**

To‘rtburchak Rectangle usuli bilan chizilib, bu usulni chaqirish instruksiyasining umumiy ko‘rinishi quyidagicha:

#### **Object.Canvas.Rectangle(x1, y1,x2, y2)**

Bu yerda  $x_1, y_1$  va  $x_2, y_2$  – chapgi yuqori va o‘nggi pastgi burchaklar koordinatalari.

RoundRec usuli burchaklari yumaloq to‘rtburchak chizishga imkon beradi. RoundRec usulini chaqirish instruksiyasi quyidagi ko‘rinishga ega:

#### **Object.Canvas.RoundRec(x1,y1,x2, y2, x3, y3)**

Bu yerda:

- $x_1, y_1, x_2, y_2$  – to‘rtburchak parametrlari;
- $x_3, y_3$  – chorak qismi yumaloq burchak chizish uchun ishlatiladigan ellips kattaligi.

Yana ikki usul mo‘yqalamdan foydalanib to‘rtburchak chizishga imkon beradi. FillRect usuli ichi bo‘yalgan to‘rtburchak chizadi, FrameRect – faqat kontur. Bu usullarda faqat bitta parametrga ega – TRect tipidagi tuzilma. Quyidagi misolda FillRect va FrameRect usullari orqali forma yuzasiga qizil to‘rtburchak soha va yashil konturli to‘rtburchak chizuvchi protsedura keltirilgan.

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
Var
```

```
r1, r2: TRect;
```

```
begin
```

```
r1 := Rect(20,20,60,40);
```

```
r2 := Rect(10,10,40,50);
```

```
with form1.Canvas do begin
```

```
Brush.Color := clRed;
```

```
FillRect(r1);
```

```
Brush.Color := clGreen;
```

```
FrameRect(r2);  
end;  
end;
```

## Ko'pburchak

Polygon usuli ko'pburchak chizishga mo'ljallangan bo'lib, parametri TPoint tipidagi massivdir. Massivning har bir elementi (x,y) maydonlari ko'pburchak uchi koordinatalaridan iborat bo'lgan yozuvdir.

Quyida polygon usuli yordamida uchburchak chizish protsedurasi keltirilgan:

```
procedure TForm1.Button2Click(Sender: TObject);  
var  
pol: array[1..3] of TPoint;  
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 sektori pie usuli bilan chizilib, chaqirish instruksiyasi quyidagi umumiy ko'rinishga ega:

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

Bu yerda:

- x 1, y 1, x 2, y 2 – ellips yoki aylana parametrlari;
- x 3, y 3, x 4, y 4 – sektor chegarasini tashkil qiluvchi to'g'ri chiziqlar, oxirgi nuqtalar koordinatalari.

## Nuqta

Canvas obyektining pixels xossasi tipidagi ikki o'lchovli massiv bo'lib, har bir soha nuqtasining rangi haqidagi ma'lumotni o'z ichiga oladi. Pixels xossasidan foydalanib ixtiyoriy nuqta rangini o'zgartirish, ya'ni nuqta chizish mumkin. Misol uchun:

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

Instruksiyasi soha nuqtasini qizil rangga bo'yaydi.

Quyida keltirilgan dastur pixels xossasidan foydalanib,  
 $y = 2 \cdot \sin x \cdot e^{\frac{x}{5}}$  funksiyasi grafigini chiqaradi.

**Unit Unit1;**

**interface**

**uses**

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

**Type**

**TForm1 = class(TForm)**

**Button1: TButton;**

**procedure Button1Click(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation**

**{ \$R \*.dfm }**

**Function f(x:real):real;**

**Begin**

**f:=2\*Sin(x)\*exp(x/5) ;**

**end;**

**Procedure GrOfFunc;**

**Var**

**x1,x2:real;**

**y1,y2:real;**

**x:real;**

**y:real;**

**dx:real;**

**l,b:Integer;**

**w,h:Integer;**

**mx,my:real;**

**x0,y0:Integer;**

**Begin**

**l:=10;**

**b:=Form1.ClientHeight-20;**

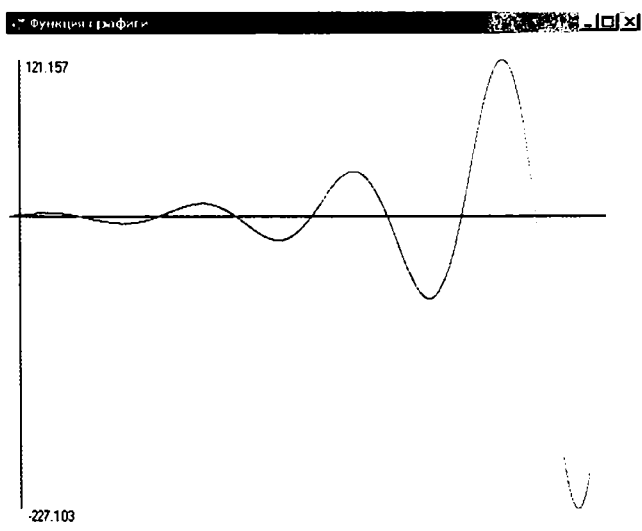
```

h:=Form1.ClientHeight-40;
w:=Form1.Width-40;
x1:=0;
x2:=25;
dx:=0.01;
y1:=f(x1);
y2:=f(x1);
x:=x1;
Repeat
  y := f (x);
  if y < y1 then y1:=y;
  if y > y2 then y2:=y;
  x:=x+dx; until (x >= x2);
  my:=h/abs(y2-y1);
  mx:=w/abs(x2-x1);
  x0:=1;
  y0:=b-Abs(Round(y1*my)) ;
  with form1.Canvas do
  Begin
  // osi
  MoveTo(l,b);LineTo(l,b-h);
  MoveTo(x0,y0);LineTo(x0+w,y0);
  TextOut(l+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
  TextOut(l+5,b,FloatToStrF(y1,ffGeneral,6,3));
  x:=x1; repeat
  y:=f(x);
  Pixels[x0+Round(x*mx),y0-Round(y*my)]:=clRed;
  x:=x+dx;
  Until (x >= x2);
  end;
  end;
  procedure TForm1.Button1Click(Sender: TObject);
  begin
  Button1.Visible:=false;
  GrOfFunc;
  end;
  end.

```

Asosiy vazifani GrOfFunc protsedurasi bajaradi. Avval [x1,x2] oraliqda funksiyaning maksimal (y2) va minimal (y1) qiymatlari hisoblanadi. So'ngra koordinatalar y ki bo'yicha masshtab hisoblanadi. Shundan so'ng protsedura grafikni quradi.





GrOfFunc protsedurasi tomonidan qurilgan grafik.

Keltirilgan dastur universal xarakterga ega. O'zga funksiya grafagini chizish uchun  $f(x)$  tanasini o'zgartirish yetarli.

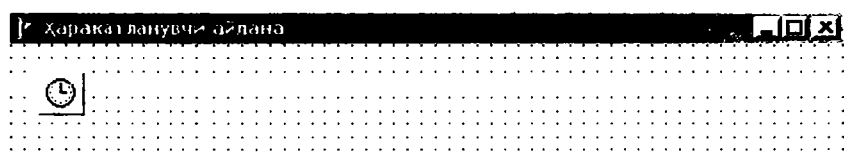
Dastur to'g'ri ishlaydi, agar funksiya ham musbat, ham manfiy qiymatlarni qabul qilsa.

## Multiplikatsiya

Multiplikatsiya deyilganda harakatlanuvchi rasm tushuniladi. Rasmni harakatlantirish uchun avval u ekranga chiziladi, ma'lum vaqtdan so'ng rasmni o'chirib yangi joyga chiziladi.

Quyidagi dastur, aylananing chapdan o'ngga harakatini ko'rsatadi.

**Harakatlanuvchi aylana** dasturining formasi.



**Harakatlanuvchi aylana** dasturining matni:

**Unit Unit1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,**

**Dialogs, ExtCtrls, StdCtrls;**

**Type**

**TForm1 = class(TForm)**

**Timer1: TTimer;**

**procedure FormActivate(Sender: TObject);**

**procedure Timer1Timer(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Procedure Ris;**

**Var**

**Form1: TForm1;**

**x,y: byte;**

**implementation**

**{ \$R \*.dfm }**

**Procedure Ris;**

**Begin**

**form1.Canvas.Pen.Color:=form1.Color;**

**form1.Canvas.Ellipse(x,y,x+10,y+10);**

**x:=x+5;**

**form1.Canvas.Pen.Color:=clBlack;**

**form1.Canvas.Ellipse(x,y, x+10, y+10) ;**

**end;**

**Procedure TForm1.FormActivate(Sender: TObject);**

**Begin**

**x:=0;**

**y:=10;**

**timer1.Interval:=50;**

**end;**

**Procedure TForm1.Timer1Timer(Sender: TObject);**

**Begin**


**Ris;**

**end;**

**end.**

## 5.2. Grafik komponentalar

**Image** komponentasi formaga rasmlarni joylashtirish uchun ishlatiladi. Joylashtirilishi lozim bo'lgan rasmlar bitli fayllar (kengaytmalari .Bmp), piktogrammali (kengaytmalari .Ico), metafayllar (kengaytmalari .wmf) bo'lishi kerak.

Image komponentasi Additional palitrasida joylashgan bo'lib, u  ko'rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan rasm uchun joy ajratiladi va keyin esa xossalar bo'limidan Picture xossasi tanlanib, u yerdan uch nuqtali tugmacha bosiladi. Natijada ekranda rasmni aniqlash va joylash uchun muloqot darchasi ochiladi. Muloqot darchasi quyidagi tugmachalarga ega:

Load – fayldan rasmni chaqirish;

Save – rasmni faylda saqlash;

Clear – tanlangan rasmni olib tashlash;

Ok – tanlangan rasmni ajratilgan joyga yozish;

Cancel – qilingan o'zgartirishlarni bekor qilish.

**Shape** komponentasi formaga aylana, to'rtburchak, ellips va boshqa shakllarni joylashtirish uchun ishlatiladi. Uning quyidagi xossalari mavjud:

Brush – shaklni bo'yash uchun cho'tkacha;

Pen – shakl chetini chizish uchun qalam;

Shape – ekranga chiqadigan shaklni aniqlaydi:

StRectangle – to'rtburchak;


StSquare – kvadrat;

StRoundRect – chetlari aylana to'rtburchak;


StRoundSquare – chetlari aylana kvadrat;

StEllipse – ellips;


StCircle – aylana.

Shape komponentasi ham Additional palitrasida joylashgan bo'lib, u  ko'rinishdagi piktogrammaga ega. Bu tugmachani bosib, formadan shakl uchun joy ajratiladi va keyin esa xossalar bo'limidan Shape xossasiga kirilib, kerakli shakl tanlanadi.

**PaintBox** komponentasi formaga chegaralangan maydonga shakllarni chizish imkonini beradi.

PaintBox komponentasi System palitrasida joylashgan bo'lib, u  ko'rinishdagi piktogrammaga ega. Bu tugmachani bosib, formadan shakl uchun joy ajratiladi va keyin esa xossalar bo'limidan Shape xossasiga kirilib, kerakli shakl tanlanadi.

Timer vizual bo‘lmagan komponenta bo‘lib, formada bajariladigan ma’lum bir operatsiyalarni vaqt bo‘yicha boshqaradi.

Timer komponentasi System palitrasida joylashgan bo‘lib, u  ko‘rinishdagi piktogrammaga ega. Bu tugmachani bosib, formaga olib kelib qo‘yiladi.

U quyidagi xossalarga ega:

Enabled – true qiymati o‘rnatilgan bo‘lsa u bo‘ladigan jarayonga ta’sir qiladi;

Interval – millisekundlarda vaqt intervalini aniqlaydi va jarayonning ekranga chiqishiga ta’sir ko‘rsatadi. Tegmagan holda 1000 (1 sekund)ni ko‘rsatadi.

### **M i s o l :**

Ilova uchun «zastavka» yaratish.

### **Ye ch i sh**

Zastavka grafik tasvirlar ko‘rinishida bo‘lib, programmalar ishga tushirilganda bir necha sekunddan so‘ng ekranda paydo bo‘ladi. Unda programma nomi va uning mualliflari haqida ma’lumot bo‘lishi mumkin.

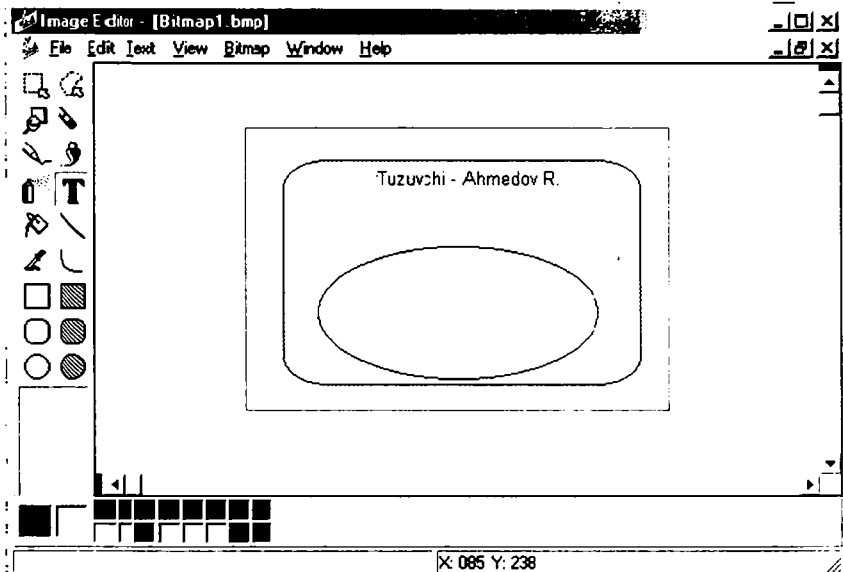
Grafik tasvirni, ya’ni .bmp kengaytmaga ega bo‘lgan faylni grafik muharriri yordamida tayyorlaymiz. Delphi sistemasini ishga tushirishdan avval tuziladigan ilovani saqlash uchun o‘zimizga papka tashkil qilamiz.

1.Oldin tuzilgan biror-bir ilovani ochamiz yoki yangi ilova tashkil etamiz.

2.Bosh menyudan grafik muharririni ishga tushiramiz: Tols=>Image Editor. (Bu Delphi grafik muharriri oddiy Paint grafik muharriridan uncha katta farq qilmaydi.)

3.Delphi grafik muharriri Image Editor menyusidan File=>New=>Bitmap File(.bmp) buyrug‘i beriladi. Natijada, ekranda rasm parametrlarini berish uchun muloqot darchasi paydo bo‘ladi. Muloqot darchasidan kerakli parametrlar tanlanib, Ok tugmasi bosiladi. Tayyor mavjud rasm fayllaridan ham foydalanish mumkin.

4.Grafik muharriri oynasidan ajratilgan joyga ixtiyoriy rasm chizililadi va, u saqlanadi. Masalan, aylana va unga tashqi chizilgan rasm chizib, ichiga «Tuzuvchi – R.Ahmedov » so‘zi yozib qo‘yilsin. Matnni yozish uchun uskunalar panelining «T» (Text) tugmachasidan foydalaniladi.



5. Grafik fayli saqlanadi va undan chiqiladi.

6. System palitrasidan Timer komponentasining tugmachasini bosib, formaga olib kelib qo'yiladi va u Timer1 nomini oladi. Interval xossasini 3000 ga tenglashirib olamiz.

7. Additional palitrasidan Image komponentasi tugmachasini bosib formadan rasm uchun joy ajratiladi va keyin esa xossalar bo'limidan Picture xossasi tanlanib, u yerdan uch nuqtali tugmachasi bosiladi. Natijada, ekranda rasmni aniqlash va joylash uchun muloqot darchasi ochiladi. Muloqot darchasidan Load buyrug'i berilib, saqlangan rasm faylimiz tanlanadi va Ok tugmasi bosiladi. Rasm to'liq formaga joylashishi uchun Autosize xossasiga True qiymatini o'rnatamiz.

8. Timer1 komponentini aktivlashtiramiz, ya'ni uni ikki marta tez-tez bosamiz va kodlarni yozish oynasiga quyidagi qora yozilgan kodlarni kiritamiz:

```
Procedure TForm1.Timer1Timer(Sender: TObject);  
Begin  
    Image1.Free;  
    Timer1.Free;  
end;
```

Bu shuni bildiradiki, dasturlash ishga tushgandan so'ng 3000 millisekunddan o'tishi bilan Image1 va Timer1 komponentalari kompyuter xotirasidan va mos ravishda ekrandan o'chiriladi.

9. Tuzilgan loyiha (проект) ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlanadi.

10. Yangi nom bilan saqlangan loyiha, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda ekranda yuqoridagi 4-punkttdagi rasm «zastavka» ko'rinishida namoyon bo'ladi.

Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

**Unit px1;**

**interface**

**uses**

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

**Type**

**TForm1 = class(TForm)**

**procedure Timer1Timer(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation**

**{ \$R \*.dfm }**

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

**begin**

**Image1.Free;**

**Timer1.Free;**

**End;**

**end.**

Yaratilgan «zastavka»miz qism dasturining ishlashi davomida o'chib-yonib turishi uchun quyidagi OnTimer hodisasini qayta ishlash kodini yozishimiz kerak bo'ladi.

**Procedure TForm1.Timer1Timer(Sender: TObject);**

**Begin**

**If Image1.visible=true then Image1.hide**

## Else Image1.Show;

**End;**

### **M i s o l :**

Oyning Yer atrofida aylanishini namoyish etuvchi ilova yaratish.

### **Ye ch i sh**

1.Yangi ilova yaratamiz.

2.Formaga Timer komponentasini Timer1 nomi bilan joylashtiramiz.Uning Interval xossasini 55 qilib o'rnatamiz. Jarayon ya'ni hodisa 55 millisekundda paydo bo'ladi (uyg'onadi).

3.Additional palitrasidan Shape komponentasini Shape1 nomi bilan formaga joylashtiramiz va uning quyidagi xossalarini o'rnatamiz.

```
Shape – StCircle;  
Height – 121;  
Width – 121;  
Left – 240;  
Top – 104.
```

Brush xossasini tanlab ikki marta sichqonchani bosamiz, natijada ikkita yana qo'shimcha xossalar paydo bo'ladi: Color va Style. Color xossasini tanlab, unga siBlue qiymatini o'rnatamiz.

4.Formaga ikkinchi Shape komponentasini Shape2 nomi bilan joylashtiramiz va uning quyidagi xossalarini o'rnatamiz.

```
Shape – StCircle;  
Height – 41;  
Width – 41;  
Left – 400;  
Top – 152.
```

Brush xossasiga ciYellow rangini o'rnatamiz.

5. Formaning yuqori qismiga Label komponentasini Label1 nom bilan joylashtiramiz va uning Caption xossasini «Oyning Yer atrofida aylanishi» qiymatiga o'zgartiramiz. Font xossasiga kirib kerakli shriftni va uning o'lchamini aniqlaymiz (agar kerak bo'lsa). Masalan:

```
Шрифт – Courier New;  
Начертание – polujirniy;  
Размер – 16;  
Набор символов – krilitsa.
```

Transparent xossasi qiymatini True qilib o'rnatamiz.

6.Timer1 komponentini aktivlashtiramiz, ya'ni uni ikki marta tez-tez bosamiz va kodlarni yozish oynasiga quyidagi qora yozilgan kodlarni kiritamiz:

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

```
Begin
```

```
x:=x+0.1;
```

```
Shape2.Left:=265+Trunc(50*Cos(x));
```

```
Shape2.top:=150-Trunc(50*Sin(x));
```

```
end;
```

7.Tuzilgan loyiha ya'ni Project1 va Unit1 standart modul nomlarini mos nomlar bilan almashtirib saqlanadi.

8.Yangi nom bilan saqlangan loyiha, ya'ni ilova F9 tugmachasini bosish bilan ishga tushiriladi.

Ilova ishga tushirilganda ekranda Oyning Yer atrofida aylanishi namoyish qilinadi.

Tashkil qilingan modulning to'liq ko'rinishini keltiramiz:

```
Unit px2;
```

```
interface
```

```
Uses
```

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

```
Type
```

```
TForm1 = class(TForm)
```

```
Timer1: TTimer;
```

```
Shape1: TShape;
```

```
Shape2: TShape;
```

```
Label1:TLabel;
```

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

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
Var
```

```
Form1: TForm1;
```

```
X: Real;
```

```
implementation
```

```
{ $R *.dfm }
```

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



```

begin
  x:=x+0.1;
Shape2.Left:=265+Trunc(50*Cos(x));
Shape2.top:=150-Trunc(50*Sin(x));
End;

```

### Initialization

```

x:=0
end.

```

Qism dasturi o'zgaruvchi x global holda e'lon qilingan. Shu tufayli uning boshlang'ich qiymati Initialization seksiyasida berilgan.

## 5.3.Delphining multimedia imkoniyatlari

Windows muhitidan foydalanuvchi dasturlarning ko'pchiligi multimedia dasturlaridir. Bunday dasturlar videorolik va multiplikatsiya ko'rish, musiqa va nutqni eshitishga imkon beradi. Multimediyali dasturlarga o'yinlar o'rgatuvchi dasturlar misol bo'la oladi.

Delphida multimediyali dasturlar yaratish uchun ikki komponentadan foydalanish mumkin:

- Animate – oddiy animatsiya yaratish uchun (masalan, fayllardan nusxa olishda foydalanuvchi ko'radigan animatsiya);
- MediaPlayer –murakkab vazifalarni bajarish uchun, videoroliklarni ko'rish, tovushli animatsiya.

### Animate komponentasi

Animate komponentasining belgisi **Win32** qatorda joylashgan bo'lib, kadrlari AVI-faylda joylashgan sodda animatsiyani ko'rishga imkon beradi. AVI-fayldagi animatsiya tovushli bo'lishi mumkin bo'lsa ham Animate komponentasi faqat tasvirni aks ettirishga imkon beradi.

Animate komponentasi formaga oddiy usulda qo'shiladi. Komponenta formaga qo'shildandan so'ng uning xossalarini o'rnatish lozim. Animate xossalari jadvalda keltirilgan:

Animate komponentasining xossalari:

Xossa	Ta'rifi
Name	Komponenta nomi
FileName	Animatsiyada joylashgan AVI-fayl nomi
StartFrame	Animatsiyani birinchi kadrlarning nomeri

stopFrame	Animatsiyadagi oxirgi kadning nomeri
Activate	Animatsiyani aks ettirish jarayonini aktivlashtirish belgisi
Color	Komponenta fonining rangi
Transparent	Animatsiyani aks ettirishda shaffof rangdan foydalanish rejimi
Repetitions	Animatsiyani qaytarish soni

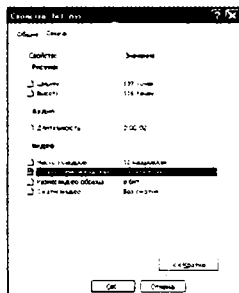
Agar FileName xossasiga tovushli fayl nomi yozilsa, Delphi faylni ochish mumkin emasligi haqida ma'lumot chiqaradi (**Cannot open AVI**). AVI-faylda animatsiya va tovush yoki faqat animatsiya yozilganligini aniqlash uchun Windowsda kerakli papkani ochib, AVI-faylni belgilash va kontekstli menyudan **Свойства** komandasini tanlash lozim. Natijada **Свойства** oynasi ochilib, **Сводка** qatorida fayl haqida to'liq ma'lumot beriladi.

Animate komponentasi dasturchiga Windows standart animatsiyasidan foydalanishga imkon beradi. Animatsiya turi Sommon AVI xossasi qiymati bilan belgilanadi. Xossa qiymati nomlangan konstantalar orqali beriladi. Quyidagi jadvalda konstantalar qiymatlari, animatsiya turi va jarayon ta'rifi berilgan.

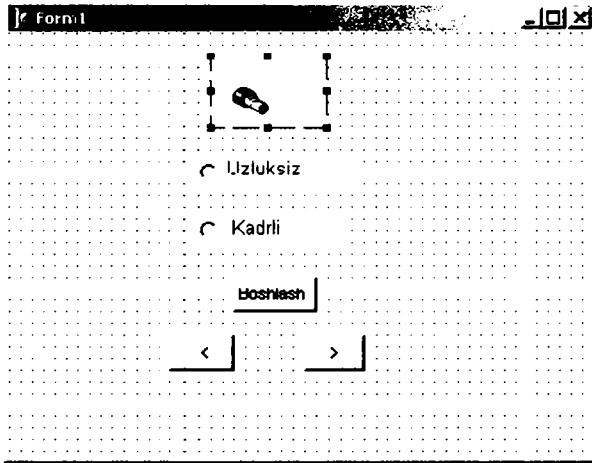
Common AVI xossasining qiymatlari:

Qiymat	Animatsiya	Jarayon
aviCopyFiles	↵ , ↵	Fayldan nusxa olish
AviDeleteFile	↵	Faylni o'chirish
aviRecycleFile	↵ ↵	Faylni korzinağa o'chirish

Quyidagi dastur Animate komponentasidan foydalanishga misol bo'ladi. Dastur formasining ko'rinishi Animate komponentasi xossalari qiymatlari jadvalida berilgan.



**Svodka bo'limida AVI-fayli haqida ma'lumot aks etadi.  
Animatsiya qurish dasturining formasi.**



Animatel xossalari ning qiymatlari:

<b>Xossa</b>	<b>Qiymat</b>
FileName	D:\music\ms\COMMON\GRAPHICS\AVIS\SEARCH.AVI
Active	False
Transparent	True

Dastur ishga tushirilgandan so'ng formaga birinchi animatsiya kadri chiqariladi. Dastur animatsiyani ko'rishning ikki rejimini ta'minlaydi:

- uzluksiz;
- kadrlri.

Button1 tugmasi animatsiyani ko'rish jarayonini initsializatsiya qilish yoki to'xtatib turish uchun ishlatiladi. Animatsiyaning uzluksiz aks etishi Pusk tugmasining Onclick hodisasini qayta ishlash protsedurasida Active xossasiga True qiymatini berish orqali initsializatsiya qilinadi. Bu protsedura Button1 tugmasidagi Pusk so'zini Stop so'ziga almashtiradi. Animatsiyani ko'rish rejimi RadioButton1 va RadioButton 2 tugmalari orqali tanlanadi.

**Animatsiyani ko'rish** dasturining matni:

**Unit Unit1;**

**Interface**

**Uses**

**Windows, Messages, SysUtils,  
Classes, Graphics, Controls,  
Forms, Dialogs, StdCtrls, ComCtrls, ExtCtrls;  
Type**

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

```

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

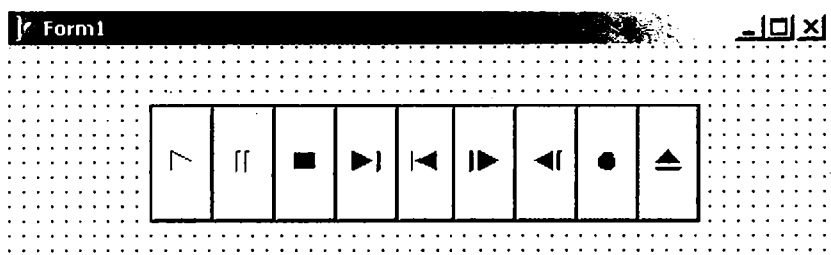
```

## MediaPlayer komponentasi

MediaPlayer komponentasining belgisi **System** qatorida joylashgan bo'lib, videoroliklar, tovush va tovushli animatsiyani qurishga imkon beradi. Formaga MediaPlayer komponentasi joylashtirilganda tugmalar guruhi paydo bo'ladi.

Bu tugmalar vazifalari jadvalda berilgan. MediaPlayer xossalari keyingi jadvalda keltirilgan.

MediaPlayer komponentasi.



MediaPlayer komponentasining tugmalari:

<b>Tugma</b>	<b>Belgisi</b>
Eshittirish	btPlay
Pauza	btPause
Stop	btStop
Keyingi	btNext
Oldingi	btPrev
Qadam	btStep
Orqaga	btBack
Yozuv	btRecord
Ochish/Yopish	btEject

## Tovushlarni eshittirish

Tovush fragmentlari WAV kengaytmali fayllarda saqlanadi. Misol uchun C:\Winnt\Media katalogida Windows standart tovushlari yozilgan fayllar joylashgan.

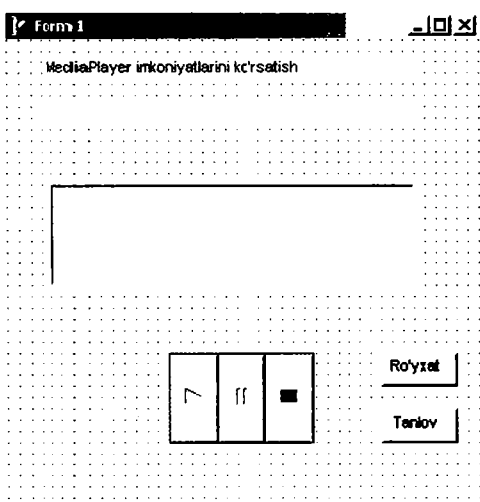
Quyidagi dasturda MediaPiayer komponentasidan WAV-fayllarida joylashgan tovushlarni eshitish uchun foydalanish ko'rsatilgan.

MediaPiayer komponentasidan tashqari formada ListBox va ikki Label komponentalari mavjud bo'lib, birinchisi ma'lumotning aks etishi, ikkinchisi ro'yxatdan tanlangan WAV-fayl nomini aks ettirish uchun ishlatiladi. Bundan tashqari, ikki Button komponentalari mavjud bo'lib,

birinchisi tovushli fayllar ro'yxatini ekranga chiqarish, ikkinchisi ro'yxatdan tovushli faylni tanlash uchun mo'ljallangan.

Dastur quyidagicha ishlaydi. Dialog oynasi paydo bo'lgandan so'ng «Microsoft tovushi» eshitiлади, so'ngra foydalanuvchi ro'yxatdan C:\Windows\Media katalogidagi ixtiyoriy tovushli faylni tanlaydi va **Воспроизведение** tugmasini bosgandan so'ng shu faylni eshittiriladi.

**Microsoft Windows tovushlarining** dastur formasi.



MediaPlayer1 komponentasining o'zgartirilgan xossalar qiymatlari jadvalda berilgan.

MediaPlayer1 komponentasining xossalarining qiymatlari:

Xossa	Qiymat
DeviceType	DtAutoSelect
FileName	C:\Winnt\Media\3vuk Microsoft.wav
AutoOpen	True
VisibleButtons .btNext	False
VisibleButtons .btPrev	False
VisibleButtons .btStep	False
VisibleButtons .btBack	False
VisibleButtons .btRecord	False
VisibleButtons .btEject	False

**Microsoft Windows tovushlarining** dastur matni:

```
unit Unit1;
interface
uses
```

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

Type

```
TForm1 = class(TForm)
  MediaPlayer1: TMediaPlayer;
  Label1: TLabel;
  Label2: TLabel;
  ListBox1: TListBox;
  Button1: TButton;
  Button2: TButton;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject); private
    { Private declarations } public
  { Public declarations } end;
```

Const

```
SOUNDPATCH='C:\WINDOWS\Media\';
```

var

```
Form1: TForm1;
```

implementation

```
{ $R *.dfm }
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

var

```
SearchRec: TSearchRec;
```

begin

```
if FindFirst(«C:\WINDOWS\Media\»+'*.wav', faAnyFile,
SearchRec) =0 then
```

Begin

```
Form1.ListBox1.Items.Add(SearchRec.Name) ;
```

```
while (FindNext(SearchRec) = 0) do
```

```
Form1.ListBox1.Items.Add(SearchRec.Name);
```

```
end;
```

```
end;
```

```
Procedure TForm1.Button2Click(Sender: TObject);
```

Begin

```
Label2.Caption:=ListBox1.Items[ListBox1.itemindex];
```

```
if (Label2.Caption <> '') then
```

begin

```
with MediaPlayer1 do begin
```

```
FileName:='C:\WINDOWS\Media\' + Label2.Caption;
```

```
Open; end;
```



end;  
end;  
end.

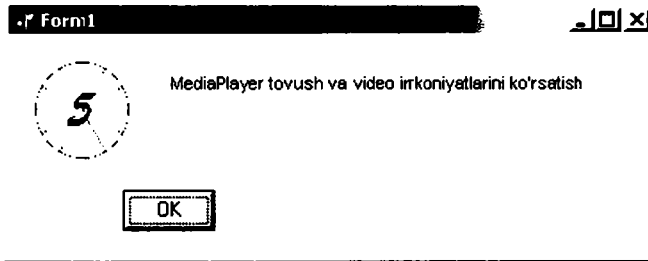
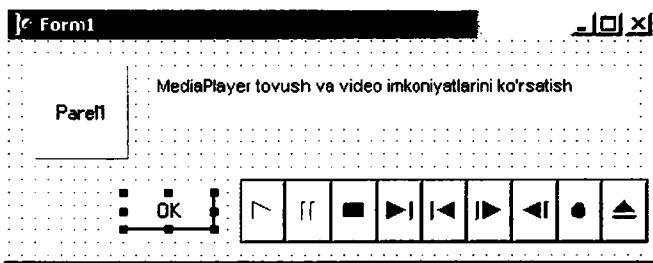
### Videorolik va animatsiyalarni ko'rish

Tovush eshittirishdan tashqari MediaPlayer komponentasi AVI-fayl (AVI -Audio Video Interleave so'zining qisqartmasi bo'lib, tovush va video almashuvi deb o'qiladi) sifatida berilgan videorolik va multiplikatsiyani ko'rishga imkon beradi.

Quyidagi dasturda buyruq tugmasini bosish natijasida tovushli multiplikatsiyada soat mili bo'yicha aylanuvchi Delphi so'zini ko'rish mumkin (delphi.avi fayli bu multiplikatsiyani o'z ichiga olgan).

Dasturning dialog oynasi rasmda, MediaPlayer1 komponentasi xossalari qiymati jadvalda berilgan.

### MediaPlayerdan foydalanish dasturining formasi va dialog oynasi.



MediaPlayer1 komponentasining xossalari:

Xossa	Qiymat
Name	MediaPlayer1
FileName	C:\WINDOWS\clock.avi
DeviceType	dtAVIVideo
AutoOpen	True
Display	Panel1
Visible	False

Formada Panell komponentasida animatsiya aks etadi, uning nomi MediaPlayer1 komponenta Display xossasining qiymati sifatida beriladi.

Animatsiya sohasi o'Ichovlari MediaPlayer komponentasi DisplayRect xossasi qiymati sifatida beriladi. Bu qiymat dastur ishlash jarayonida MediaPlayer1.DisplayReet:=Rect(0,0,60,60) instruksiyasini bajarish natijasida o'rnatiladi.

Tovushli animatsiyani aks ettirish dasturining matni:

```
Unit Unit1;  
interface  
uses  
Windows, Messages, SysUtils,  
Classes, Graphics, Controls,  
Forms, Dialogs, MPlayer, StdCtrls, ExtCtrls;  
Type  
TForm1 = class(TForm)  
Label1: TLabel;  
Panel1: TPanel;  
Button1: TButton;  
MediaPlayer1: TMediaPlayer;  
procedure Button1Click(Sender: TObject);  
procedure FormCreate(Sender: TObject);  
private  
{ Private declarations } public  
{ Public declarations } end;  
Var  
Form1: TForm1 ;  
implementation  
{SR *.DFM}  
Procedure TForm1.Button1Click(Sender: TObject);  
Begin  
MediaPlayer1.Play;  
end;  
procedure TForm1.FormCreate(Sender: TObject);  
begin  
MediaPlayer1.DisplayRect:=Rect(0,0,60,60);  
end;  
end.
```

## 5.4. Bosmaga chiqarish

### Matnli rejim

Agar printerda biror matn bosmaga chiqarilishi lozim bo'lsa, **AssignFile** o'rniga **AssignPrn** protsedurasini chaqirish lozim:

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
```

```
To_Prn : TextFile;
```

```
begin
```

```
AssignPrn(To_Prn);
```

```
Rewrite(To_Prn);
```

```
Writeln(To_Prn, `Printer in Text Mode`);
```

```
CloseFile(To_Prn);
```

```
end;
```

Delphi tilida System modulidagi ba'zi funktsiya va o'zgaruvchilar nomlari o'zgartirilgan:

- **Assign o'rniga AssignFile**
- **Close o'rniga CloseFile** вмечро
- **Text o'rniga TextFile**

### Grafik rejim (TPrinter obykti)

Grafik rejimda bosmaga chiqarish uchun formaning Print usulini qo'llash mumkin. Lekin maxsus Printer obyektidan (TPrinter sinfiga tegishli) foydalanish qulayroqdir. Bu obyektidan foydalanish uchun Printers modulini dasturga qo'shish ya'ni uses bo'limida e'lon qilish lozim. Printer obykti Canvas xossasidan foydalanishga imkon beradi.

#### **Printer xossalari:**

Aborted – mantiqiy tur; foydalanuvchi Abort usuli bilan bosmani to'xtatganini ko'rsatadi.

Canvas – grafika chiqarish sohasi.

Fonts – shriftlar ro'yxati.

Handle - Windows API chaqirilganda foydalaniladi.

Orientation – sahifa yo'nalishi, vertikal yoki gorizontaal.

PageWidth, PageHeight, PageNumber – sahifa kengligi, balandligi va nomeri.

Printers – printerlar.

PrinterIndex joriy printer nomeri. O'ratilgan printer nomeri – 1.

Printing – mantiqiy tur, bosmaga chiqarish boshlanganligini ko'rsatadi (BeginDoc usuli bilan).

Title – Print Manager uchun sarlavha.

### Printer usullari:

Abort – bosmani to‘xtatish.

BeginDoc – sohaga chizishdan oldin chaqiriladi.

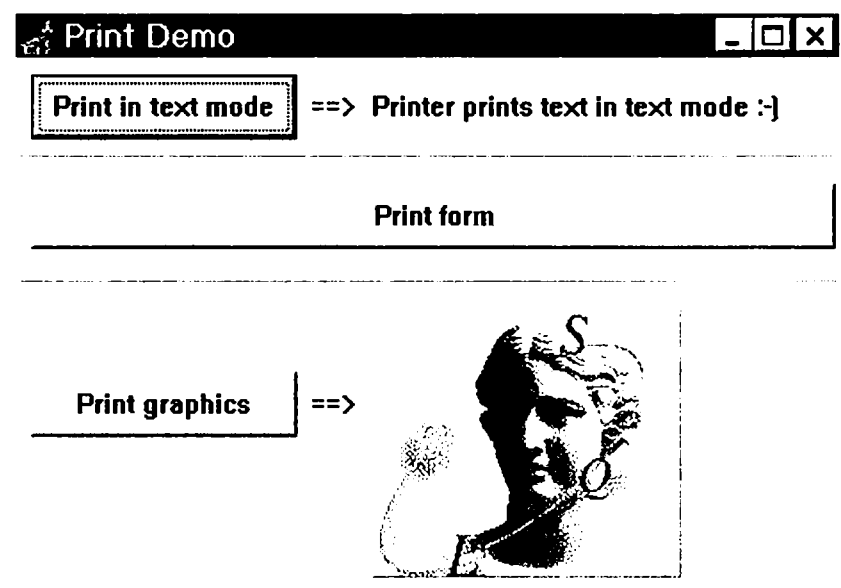
EndDoc – sohaga chizib bo‘lingandan so‘ng chaqiriladi. Shundan so‘ng printer ishga tushadi.

NewPage – yangi satrga o‘tish.

Shunday qilib, grafik ma‘lumotni bosmaga chiqarish quyidagi tartibda amalga oshiriladi:

- **BeginDoc usuli chaqiriladi.**
- **Sohaga (Canvas) chiziladi.**
- **Agar bir necha sahifaga ajratish zarur bo‘lsa NewPage usuli chaqiriladi.**
- **EndDoc usuli bilan grafik ma‘lumot printeriga yuboriladi.**

Quyidagi misolda bosmaga chiqarishning uchta usuli ko‘rsatilgan. Dastur formasining ko‘rinishi.



**Dastur matni:**

```
Unit Pri_form;
```

```
interface
```

```
uses
```

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

**Type**

```
TForm1 = class(TForm)  
  Button1: TButton;  
  Label1: TLabel;  
  Label2: TLabel;  
  Bevel1: TBevel;  
  Button2: TButton;  
  Bevel2: TBevel;  
  Button3: TButton;  
  Label3: TLabel;  
  Panel1: TPanel;  
  Imaoe1: TImage;  
  procedure Button1Click(Sender: TObject);  
  procedure Button2Click(Sender: TObject);  
  procedure Button3Click(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }  
end;
```

**Var**

```
Form1: TForm1;
```

**implementation**

```
{$R *.DFM}
```

```
Procedure TForm1.Button1Click(Sender: TObject);
```

```
Var
```

```
  To_Prn : TextFile;
```

```
Begin
```

```
  Button1.Enabled:=False;
```

```
  AssignPrn(To_Prn);
```

```
  Rewrite(To_Prn);
```

```
  Writeln(To_Prn, Label2.Caption);
```

```
  CloseFile(To_Prn);
```

```
end;
```

```
Procedure TForm1.Button2Click(Sender: TObject);
```

```
Begin
```

```
Button2.Enabled:=False;  
Form1.Print;  
end;
```

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

## Savollar

- 1.Soha o'lchovlari qaysi xossalar yordamida aniqlanadi?
- 2.Matn shrifti qaysi xossalar orqali aniqlanadi?
- 3.Grafik obyekt yuzasiga matn chiqarish uchun qaysi usul qo'llaniladi va bu usulni chaqirish instruksiyasi ko'rinishini keltiring.
- 4.To'g'ri chiziq qaysi usul orqali amalga oshiriladi va bu usulni chaqirish instruksiyasi ko'rinishini keltiring.
- 5.Aylana yoki ellips chizish uchun qaysi usul chaqiriladi va bu usulni chaqirish instruksiyasi ko'rinishini keltiring.
- 6.Yoyni chizish uchun qaysi usul qo'llaniladi va u qanday umumiy ko'rinishga ega?
- 7.To'rtburchak chizish uchun qaysi usul qo'llaniladi va u qanday umumiy ko'rinishga ega?
- 8.Ko'pburchak chizish uchun qaysi usul qo'llaniladi va u qanday umumiy ko'rinishga ega?
- 9.Ellips yoki aylana sektorini chizish uchun qaysi usul qo'llaniladi va u qanday umumiy ko'rinishga ega?
- 10.Image komponentasi formaga nimani joylashtirishda qo'llaniladi?
- 11.Shape komponentasi formaga nimalarni joylashtirishda qo'llaniladi?
- 12.Multiplikatsiya deyilganda nima tushuniladi?

## VI.DELPHI TILINING IMKONIYATLARI

### 6.1.Yozuvlarni faylga yozish va fayldan o'qish

Yozuvlarni faylda saqlash mumkin. Dasturning o'zgaruvchi yozuv qiymatini faylda saqlash uchun yozuv turidagi faylni e'lon qilish kerak. Misol uchun:

#### Type

TRerson = **record**

f\_riame: **string** [20] ;

l\_name: **string**[20];

address: **string**[50]; **end; var**

f: **file of** TPerson;

Instruksiya komponentalari TPerson turidagi yozuvlar bo'lgan faylni e'lon qiladi. Yozuvlar fayli bilan ishlash oddiy fayllar bilan ishlashdan farq qilmaydi.

### Yozuvlarni faylga yozish

Файлга ёзув қўшиш

Sportsmen

Mamlakat

Sport turi

Medal

Oltin

Kumush

Bronza

Kumush

Foydalanuvchi tomondan kiritiluvchi musobaqa natijalarini faylga yozuvchi dasturni ko'rib chiqamiz. Ma'lumotlar dialog oynasiga kiritilib, komponentalari TMedal turidagi yozuvlar bo'lgan faylda saqlanadi.

Sportsmen familiyasini kiritish uchun Edit komponentasi ishlatiladi. Sport turi va mamlakat nomini kiritish uchun ComboBox komponentasi ishlatiladi. Ro'yxatni hosil qilish uchun Object Inspector oynasida Items xossasini tanlab, uch nuqtali tugmasini bosib, ro'yxat muharririni aktivlashtirish lozim.

5 lines

O'zbekiston  
 Rossiya  
 Germaniya  
 Fransiya  
 Portugaliya

Code Editor ..

OK

Cancel

Help

**Dastur matni:**

**Unit Unit1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Classes,**

**Graphics, Controls, Forms,**

**Dialogs, StdCtrls, Grids, ExtCtrls;**

**Type**

**TForm1 = class(TForm)**

**Edit1: TEdit;**

**Label2: TLabel;**

**Label3: TLabel;**

**Label4: TLabel;**

**Label5: TLabel;**

**Button1: TButton;**

**ComboBox1: TComboBox;**

**ComboBox2: TComboBox;**

**RadioGroup1: TRadioGroup;**

**procedure FormActivate(Sender: TObject);**

**procedure FormClose(Sender: TObject;**

**var Action: TCloseAction);**

**procedure Button1Click(Sender: TObject);**

**private**



```

{ Private declarations } public
{ Public declarations } end;
TKind = (GOLD, SILVER, BRONZE);
TMedal=record
country: string[20];
sport: string[20];
person: string[40];
kind: TKind;
end;
Var
Form1: TForm1;
f: file of TMedal;
implementation
{$R *.DFM}
procedure TForm1.FormActivate(Sender: TObject);
var
resp : word;
begin
AssignFile(f, 'a:\medals.db');
{$I-}
Reset (f);
Seek(f, FileSize(f));
{$I+}
if IOResult = 0
then button1.enabled:=TRUE;
else begin
resp:=MessageDlg('Файл базы данных не найден.'MB Fayli
topilmagan.
+ 'Yangi MB yaratasizmi?', mtinformation,[mbYes,mbNo],0);
if resp = mrYes then begin {$I-}
rewrite(f); {$I+}
if IOResult = 0
then button1.enabled:=TRUE
else ShowMessage('Ошибка создания файла БД. '); MB faylini
yaratishda xatolik.
end;
end;
end;
Procedure TForm1.Button1Click(Sender: TObject);
var
medal: TMedal;

```

```

begin
with medal do begin
country := ComboBox1.Text;
sport := ComboBox2.Text;
person := Edit1.Text;
case RadioGroup1.ItemIndex of
0: kind := GOLD;
1: kind := SILVER;
2: kind := BRONZE;
end;
end;
write(f,medal);
end;
procedure TForm1.FormClose(Sender: TObject;
var
Action: TCloseAction);
begin
CloseFile(f);
end.

```

Bu dasturda TForm1.FormActivate protsedurasi faylni ochadi. Fayl matnli bo'lmagani uchun Rewrite protsedurasi bilan ochilib Seek protsedurasi ko'rsatgichni fayl oxiriga o'rnatadi.

TForm1.Button1Click protsedurasi faylga yozuv qo'shishni amalga oshiradi.

TForm1.FormClose protsedurasi faylni berkitadi.

Ko'rilgan dasturda ro'yxat loyihalashtirish jarayonida o'rnatiladi. Dastur bajarilish jarayonida ro'yxatni o'rnatish uchun items xossasiga Add usulini qo'llash kerak.

Masalan:

```

Form1.ComboBox1.Item.Add(' Rossiya ');
Form1.ComboBox1.Item.Add(' Avstriya ');
Form1.ComboBox1.Item.Add(' Germaniya ');
Form1.ComboBox1.Item.Add(' Fransiya ');

```

### Fayldan yozuvni o'qish

Quyidagi dasturda fayl ochilib, hamma mamlakatlar yoki tanlangan mamlakat sportchilari tomonidan yutib olingan medallar ro'yxatini ekranga chiqaradi.

Quyidagi jadvalda forma komponentalarining xossalari berilgan:

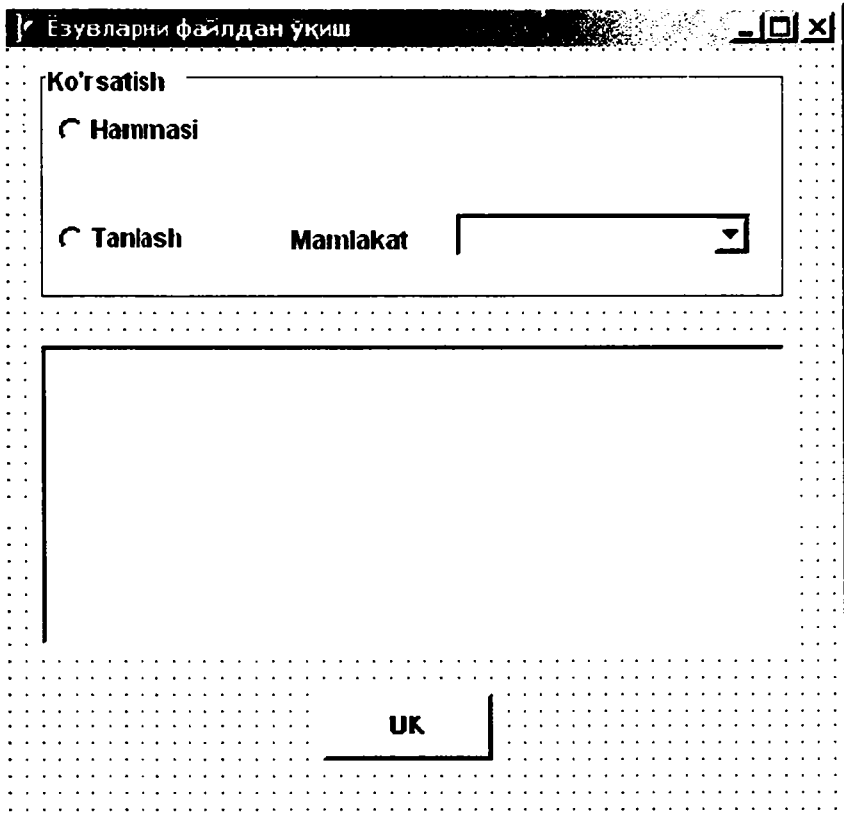
Xossa	Qiymat
RadioButton1.Checked	True
Label1.Enabled	False
ComboBox1.Enabled	False
Memo1.ReadOnly	True
Memo1.ScrollBars	Vertical

Mamlakat nomini tanlash uchun ComboBox1 komponentasidan foydalanilgan. Ro'yxat loyihalashtirish jarayonida kiritilishi lozim.

**Dastur matni:**

**Unit Unit1;**

**interface**



```

uses
Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls;
Type
TForm1 = class(TForm)
  RadioButton1: TRadioButton;
  RadioButton2: TRadioButton;
  Button1: TButton;
  GroupBox1: TGroupBox;
    Label1: TLabel;
    ComboBox1: TComboBox;
    Memo1: TMemo;
  Procedure Button1Click(Sender: TObject);
  procedure RadioButton2Click(Sender: TObject);
  procedure RadioButton1Click(Sender: TObject);
private
  { Private declarations } public
  { Public declarations } end;

```

```

Var
Form1: TForm1;
implementation
{ $R *.DFM }
procedure TForm1.Button1Click(Sender: TObject) ;
Type
  // tip medali
  TKind = (GOLD,SILVER,BRONZE);
  // zapis fayla
  TMedal = record
  country:string[20]; sport:string[20];
  person:string[40]; kind:TKind;
  end;
  Var
  f: file of TMedal;
  rec: TMedal;
  n: Integer;
  st: string[80];
  Begin
  AssionFile(f,'a:\medals.db');
  {$I-}

```

```

Reset (f);
{$I-}
if IOResult <> 0 then begin
ShowMessaoe('Fayl ochishda xatolik.');
```

Exit;

end;

// обработка BD

MB qayta ishlash

if RadioButton2.Checked then

Memo1.Lines.Add('\*\*\* ` + ComboBox1.Text + `\*\*\*'); n := 0;

Memo1.Clear; // очистить список поля Memo

while not EOF(f) do begin

read(f, rec); // прочитать запись

if RadioButton1.Checked or

(rec.country = ComboBox1.Text) then begin

n := n + 1;

st := rec.person+ `, ` + rec.sport;

if RadioButton1.Checked then

st := st + `, ` + rec.country; case rec.kind of

OOLD: st := st+ `, oltin`;

SILVER:st := st+ `, kumush`;

BRONZE:st := st+ `, bronza`;

end;

Memo1.Lines.Add(st); end;

end;

CloseFile(f); if n = 0 then

ShowMessage('Kerakli ma'lumot yo'q.');

end;

Procedure TForm1.RadioButton2Click(Sender: TObject);

Begin

Label1.Enabled := True;

ComboBox1.Enabled := True;

ComboBox1.SetFocus;

end;

Procedure TForm1.RadioButton1Click(Sender: TObject);

Begin

Label1.Enabled := False;

ComboBox1.Enabled := False;

end;

end.

TForm1.Button1Click protsedurasi faylni ochib yozuvlarni ketma-ket o'qiydi. Yozuv Memol maydoniga, mamlakat nomi country maydoni qiymati bilan mos kelsa yoki RadioButton1 tanlangan bo'lsa qo'shiladi.

## 6.2. Dinamik tuzilmalar

### Ko'rsatkich

Ko'rsatkich deb qiymati boshqa o'zgaruvchi yoki ma'lumotlar tuzilmasi adresiga teng bo'lgan o'zgaruvchiga aytiladi.

Ko'rsatkichlar quyidagicha ta'riflanadi:

Nom: ^Tur;

Masalan:

p1: ^Integer; r2: ^real;

Agar ko'rsatkich hech narsaga tengligi ko'rsatilmasa uning qiymati NIL ga teng deyiladi.

Ko'rsatkichga biror o'zgaruvchi adresi qiymat sifatida berilsa @ adres olish operatoridan foydalaniladi. Masalan:

r := @n;

Ko'rsatkichga boshqa ko'rsatkich qiymatini berish mumkin, agar ular bir turli bo'lsa. Masalan:

p2 := p1;

Agar ko'rsatkich i o'zgaruvchiga ko'rsatayotgan bo'lsa,

r^ := 5;

instruksiya bajarilgandan so'ng i qiymati 5 ga teng bo'ladi.

### Dinamik o'zgaruvchilar

Dinamik o'zgaruvchi deb dastur bajarilish jarayonida xotiraga ajratiladigan o'zgaruvchiga aytiladi. Xotira ajratish new protsedurasini chaqirish orqali amalga oshiriladi. Dinamik xotiraga faqat ko'rsatkich yordamida murojaat qilish mumkin. Dinamik o'zgaruvchini yo'qotish, ya'ni bu o'zgaruvchi egallagan xotirani ozod qilish uchun Dispose protsedurasi ishlatiladi.

Quyidagi protsedurada dinamik o'zgaruvchilarni yaratish va yo'qotish ko'rsatilgan:

**Procedure TForm1.Button1Click(Sender: TObject); var**

**p1,p2,p3: Integer; begin**

**New(p1);**

**New(p2);**

**New(p3);**

```

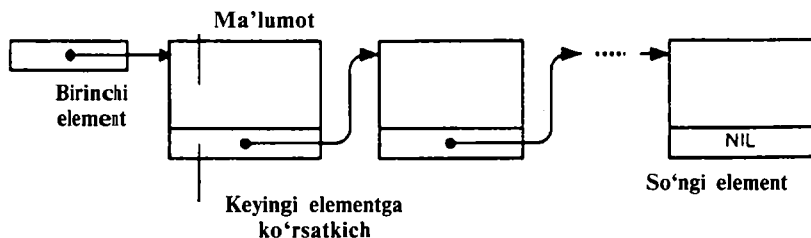
p1^ := 5;
p2^ := 3;
p3^ := p1^ + p2^;
ShowMessage
('Summa barobar' + IntToStr(p3^);
Dispose(p1);
Dispose(r2);
Dispose(r3);
end;

```

## Ro'yxat

Ko'rsatkichlar va dinamik o'zgaruvchilar ro'yxati daraxtlar kabi murakkab dinamik ma'lumotlar tuzilmalarini yaratishga imkon beradi.

Ro'yxatni quyidagicha tasvirlash mumkin:



Ro'yxatning har bir elementi ikki qismdan iborat yozuvdir. Birinchi qism informatsion qism. Ikkinchi qism oldingi elementlar bilan bog'lanishni ta'minlaydi.

Dasturda ro'yxatdan foydalanish uchun ro'yxat komponentalari turi va birinchi elementdagi ko'rsatkich aniqlanishi lozim. Quyidagi ro'yxatda talabalar familiyalarining ta'rifi keltirilgan:

```

Type
TPStudent = ^TStudent;
TStudent = record
surname: string[20];
name: string[20];
group: integer;
address: string[60];
next: TPStudent;
end;
var
head: TPStudent;

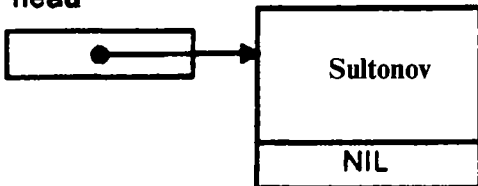
```

Rasmda ro'yxatga yangi element qo'shish jarayoni ko'rsatilgan. Ikkinchi element qo'shilgandan so'ng head shu elementni ko'rsatadi.

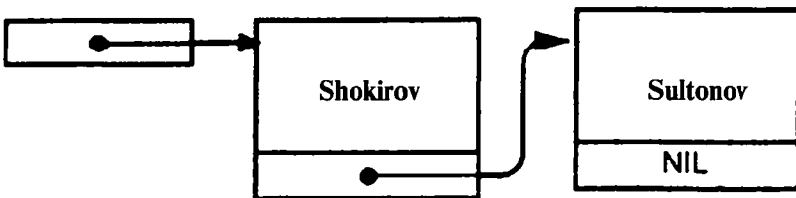
head



head

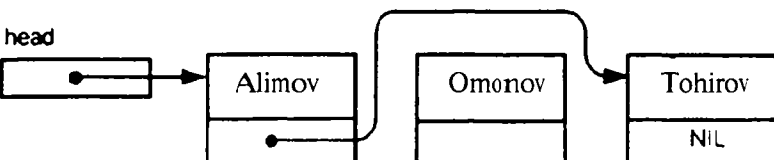
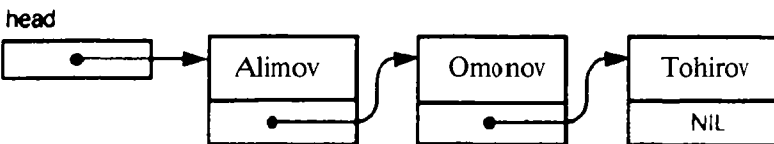


head



### Ro'yxatdan elementni o'chirish

Ro'yxatdan elementni o'chirish uchun oldingi element ko'rsatgichi qiymatini o'zgartirish lozim:





Element dinamik o'zgaruvchi bo'lgani uchun ro'yxatdan o'chirilgandan so'ng unga ajratilgan xotira ozod qilinishi lozim.

Quyida dinamik o'zgaruvchi yaratilib, yo'qotilishi ko'rsatilgan:

**Var**

**r: ^integer;**

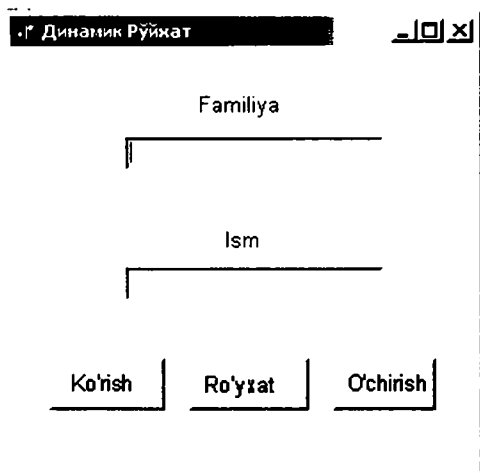
**begin**

**new(p);**

{ инструкции программы } dispose(p);

**end.**

## Dinamik ro'yxat



Quyidagi dastur talaba familiyasini ro'yxat boshiga qo'shib, talabalar ro'yxatini hosil qiladi. Ma'lumotlar tahrirlash komponentasiga kiritilib, **Qo'shish** (button1) tugmasini bosib, ro'yxatga kiritiladi. Kiritilgan elementlar ro'yxati **Ro'yxat** (button 2) tugmasini bosib, alohida oynaga chiqariladi.

Elementni ro'yxatdan o'chirish uchun, **Ro'yxat** (button 3) o'chirish tugmasi bosiladi.

Dastur matni:

**Unit Unit3;**

**interface**

**uses**

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

**Type**

**TForm1 = class(TForm)**

**Label2: TLabel;**

**Edit1: TEdit;**

**Label3: TLabel;**

**Edit2: TEdit;**

**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}

```

**Type**

```

TPStudent=^TStudent;
TStudent = record
f_name:string[20];
l_name: string[20];
next: TPStudent;
end;

```

**Var**

```

head: TPStudent; // начало (голова) списка
procedure TForm1.Button1Click(Sender: TObject);
var
curr: TPStudent;
begin
new(curr);
curr^.f_name := Edit1.Text;
curr^.l_name := Edit2.Text;
curr^.next := head; head := curr;
Edit1.text:=''; Edit2.text:= '';
end;
Procedure TForm1.Button2Click(Sender: TObject);
var
curr: TPStudent;
n:Integer;
st:string;
begin n := 0; st := '';

```

```

curr := head;
while curr <> NIL do begin
n := n + 1;
st := st + curr^.f_name + ` + curr^.l_name+#13;
curr := curr^.next;
end;
if n <> 0
then ShowMessage(«Список:» – Ro‘yxat + #13 + st)
else ShowMessage(«В списке нет элементов.»);– Ro‘yxatda
elementlar yo‘q.

```

```

end;
Procedure TForm1.Button3Click(Sender: TObject);
var
curr: TPStudent;
begin
if head<>NIL then

if head^.next=NIL then
begin
Dispose(head);
head:=NIL;
end
else
begin
new(curr);
curr:=head;
head:=curr^.next;
Dispose(curr);
end;
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
head:=NIL;
end;
end.

```

Elementlarni ro‘yxatga qo‘shishni TForm1.Button1Click protsedurasi bajaradi. Bu protsedura dinamik o‘zgaruvchi ro‘yxat yaratib, maydonlariga qiymat beradi va head ko‘rsatgichi qiymatini to‘g‘rilyadi.

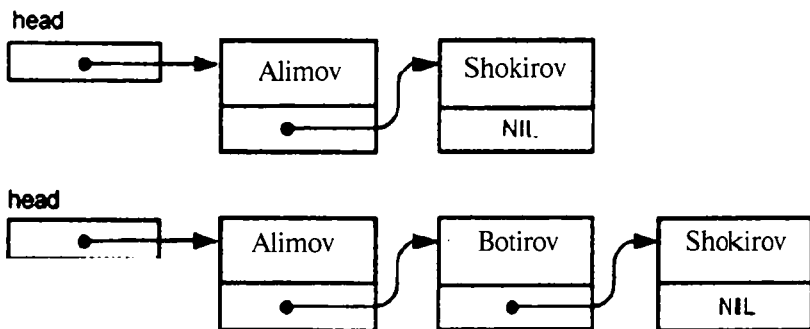
Ro'yxatni TForm1.Button2Click protsedurasi chiqaradi. Ro'yxat elementlariga murojaat qilish uchun curr ko'rsatkichidan foydalaniladi. Oldin uning qiymati birinchi element adresiga teng bo'ladi. Keyin unga next maydoni qiymati beriladi. Jarayon to next maydonining qiymati NIL bo'lmaguncha davom etadi.

Elementni ro'yxatdan o'chirish TForm1.Button3Click protsedurasi tomonidan amalga oshiriladi.

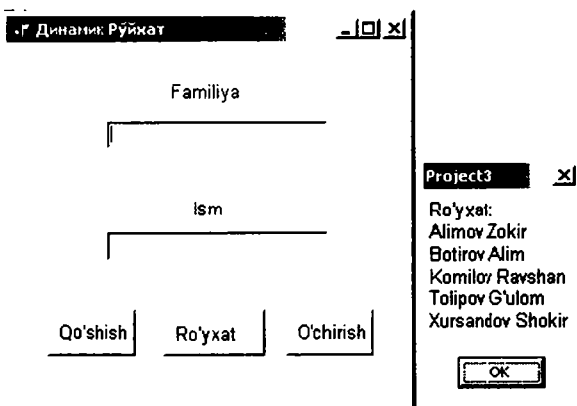
### Tartiblangan ro'yxat

Odatda ro'yxatlar tartiblangan bo'ladi. Misol uchun talabalar ro'yxati familiya bo'yicha tartiblangan bo'ladi.

Elementni tartiblangan ro'yxatga qo'shish uchun avval shu elementdan oldin turishi kerak bo'lgan element topiladi. Shundan so'ng ko'rsatkichlar qiymatlari o'zgartiriladi.



Quyidagi dastur **Familiya** maydoni bo'yicha tartiblangan ro'yxat hosil qiladi. Dastur formasining ishlash jarayoni rasmda ko'rsatilgan.



```

Dastur matni:
Unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Edit2: TEdit;
    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}
Type
TPStudent=^TStudent;
TStudent = record
f_name:strino[20];
l_name: strino[20];
next: TPStudent;
end;
Var
head: TPStudent;
procedure TForm1.Button1Click(Sender: TObject);
var
node: TPStudent;

```

```

curr: TPStudent;
pre: TPStudent;
Begin
new(node);
node^.f_name:=Edit1.Text;
node^.l_name:=Edit2.Text;
curr:=head;
pre:=NIL;
while (curr<>NIL) and (node.f_name > curr^.f_name) do
begin
pre:= curr;
curr:=curr^.next;
end;
if pre = NIL then
begin
node^.next:=head; head:=node;
end
else
begin
node^.next:=pre^.next;
pre^.next:=node;
end;
Edit1.text:='';
Edit2.text:='';
Edit1.SetFocus;
end;
Procedure TForm1.Button2Click(Sender: TObject);
var
curr: TPStudent;
n:Integer;
st:string;
begin n := 0; st := '';
curr := head;
while curr <> NIL do begin
n := n + 1;
st := st + curr^.f_name + ` + curr^.l_name+#13;
curr := curr^.next;
end;
if n <> 0
then ShowMessage(`Nienie:` + #13 + st)
else ShowMessage(`A nienea iao yeaiaioia.`);

```

```

end;
Procedure TForm1.Button3Click(Sender: TObject);
Var
curr: TPStudent;
begin
if head<>NIL then

    if head^.next=NIL then
    begin
    Dispose(head);
    head:=NIL;
    end
    else
    begin
    new(curr);
    curr:=head;
    head:=curr^.next;
    Dispose(curr);
    end;
    end;
    Procedure TForm1.FormActivate(Sender: TObject);
    Begin
    head:=NIL;
    end;
    end.

```

### 6.3.Rekursiya

#### Rekursiya tushunchasi

Rekursiv funksiya deb, o'z-o'ziga murojaat qiluvchi funksiyaga aytiladi. Rekursiv funksiyaga misol quyidagi faktorial hisoblash funksiyasini misol qilish mumkun.

```

function factorial(n: Integer): integer;
begin
if n <> 1
then factorial n * factorial(n-1)
else factorial := 1;
end;

```

Shunga e'tibor berish kerakki, agar parametr qiymati 1 ga teng bo'lsa, funksiya o'ziga murojaat qilmaydi, balki qiymat qaytaradi va rekursiv jarayon tugaydi.

Quyida rekursiya asosida faktorial hisoblash loyihasi formasi keltirilgan:

Факториал



Rekursiv funksiya yordamida faktorial hisoblash



Loyiha dasturining matni:

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

Type

```
TForm1 = class(TForm)  
  Label1: TLabel;  
  Edit1: TEdit;  
  Label2: TLabel;  
  Button1: TButton;  
  procedure Button1Click(Sender: TObject);
```

Private

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

Var



**Form1: TForm1;**

**implementation**

**{ \$R \*.dfm }**

**function factorial(n: integer): integer;**

**begin**

**if n > 1**

**then factorial := n \* factorial(n-1)**

**else factorial:= 1;**

**end;**

**Procedure TForm1.Button1Click(Sender: TObject);**

**var**

**k:integer;**

**f:integer;**

**begin**

**k := StrToInt(Edit1.Text);**

**f := factorial(k);**

**label2.caption:=Edit1.Text + `soni faktoriali ` +  
IntToStr(f)+` ga teng`;**

**end;**

**end.**

Quyidagi ikki rasmda hisoblash natijalari keltirilgan. Ikkinchi natija kutilgandan farq qiladi.



**Рекурсив функция yordamida faktorial hisoblash**

**10**

**10 soni faktoriali 3628800 ga teng**

**Hisoblash**

Rekursiv funksiya yordamida faktorial hisoblash

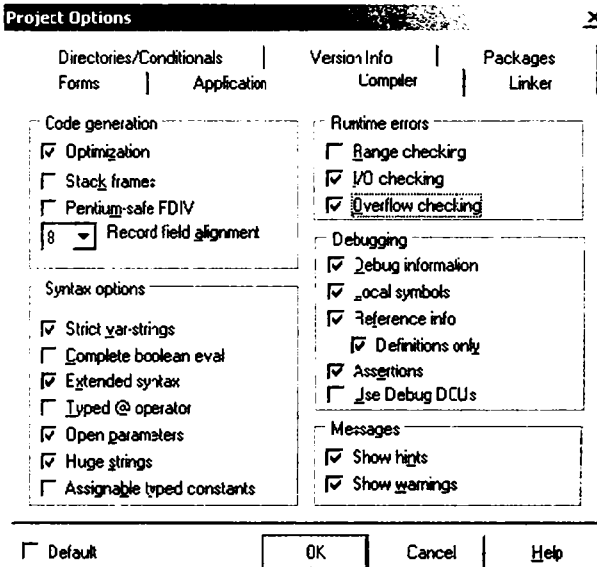
44

44 soni faktoriali 0 ga teng

Hisoblash

Natija kutilganga mos kelmaydi. 44 soni faktoriali nolga teng! Bu natijaning sababi 44 faktoriali katta son bo'lib, integer turi uchun maksimal qiymatdan kattadir.

Delphi bajarilayotgan dasturga o'zgaruvchi diapazonini kontrol qilish instruksiyasini qo'shishga imkon beradi. Buning uchun **Project Options** dialog oynasi **Compiler** bo'limida **Runtime errors** (Ошибки времени выполнения) guruhidagi **Overflow checking** (Контроль переполнения) bayroqchasini o'tatish lozim.



## Rekursiyadan foydalanishga misollar

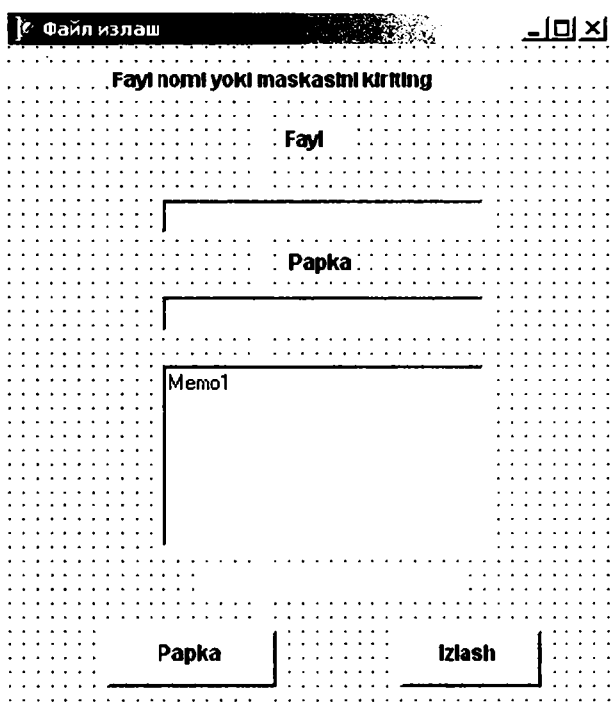
### Fayllarni izlash

Misol tariqasida fayllarni izlash masalasiga rekursiyani qo'llashni ko'rib chiqamiz. Foydalanuvchi ko'rsatgan katalog va uning ostki kataloglarida, masalan, bmp kengaytmali hamma fayllar topilsin.

Bu masalani hal qilish algoritmini quyidagicha ta'riflash mumkin:

1. Berilgan shartga mos hamma fayllar ro'yxati chiqarilsin.
2. Agar katalogda ostki katalog mavjud bo'lsa, u ham shunday ko'rib chiqilsin.

**Fayl** (Edit1) maydoni fayl nomini kiritish uchun mo'ljallangan. Katalog nomi **Papka** maydoniga kiritiladi yoki **Обзор папок**, dialog oynasi yordamida tanlanadi. Bu dialog oynasi SelectDirectory standart funksiyasi yordamida chiqariladi. Oddiy satrni WideChar turiga aylantirish uchun StringToWideChar funksiyasidan foydalanilgan.



Asosiy vazifani Find rekursiv funksiyasi bajaradi. Find funksiyasida bitta parametr searchRec tuzilmasi ishlatiladi. Birinchi va keyingi fayllarni izlash uchun FindFirst va FindNext funksiyalaridan foydalaniladi.

```

Dastur matni:
Unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;FileCtr;
Type
TForm1 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Button1: TButton;
Button2: TButton;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
    Memo1: TMemo;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
Var
Form1: TForm1;
implementation
{ $R *.dfm }
Var
FileName: string;
cDir: string;
n: integer;
procedure Find;
Var
SearchRec: TSearchRec;
begin
GetDir(0,cDir);
if cDir [length (cDir) ] <> `V` then cDir := cDir+'\';
if FindFirst(FileName, faArchive,SearchRec) = 0
then repeat

```

```

if (SearchRec.Attr and faAnyFile) = SearchRec.Attr
then begin
Form1.Memo1.Lines.Add(cDir + SearchRec.Name);
n := n + 1; end; until FindNext(SearchRec) <> 0;
if FindFirst('*', faDirectory, SearchRec) = 0 then repeat
if (SearchRec.Attr and faDirectory) = SearchRec.Attr then
begin
if SearchRec.Name[1] <> '.' then begin
ChDir(SearchRec.Name);
Find;
ChDir(..);
end;
end;
until FindNext(SearchRec) <> 0;
end;
function GetPath(mes: string):string;
var
Root: string;
pwRoot : PWideChar; Dir: string;
begin
Root := '';
GetMem(pwRoot, (Length(Root)+1) * 2);
pwRoot := StringToWideChar(Root, pwRoot, MAX_PATH*2);
if SelectDirectory(mes, pwRoot, Dir) then
if length(Dir) = 2
then GetPath := Dir+'\' else GetPath := Dir else
GetPath:= '';
end;
Procedure TForm1.Button1Click(Sender: TObject);
begin
Memo1.Clear;
Label4.Caption := '';
FileName := Edit1.Text;
cDir := Edit2.Text;
n:=0;
ChDir(cDir);
Find;
if n = 0 then
ShowMessage('Shartga mos keluvchi fayllar yo'q.')
else Label4.Caption := 'Fayllar soni:' + IntToStr(n);
end;

```

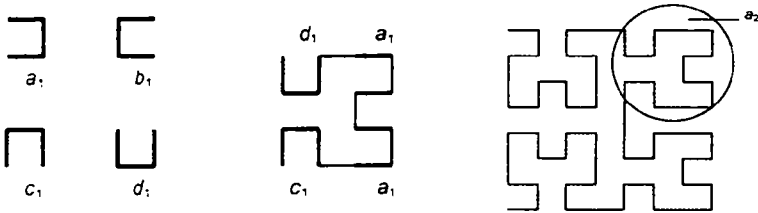
```

Procedure TForm1.Button2Click (Sender: TObject);
var
Path: string; begin
Path := GetPath('Papkani tanlang');
if Path <> ""
then Edit2.Text := Path;
end;
end.

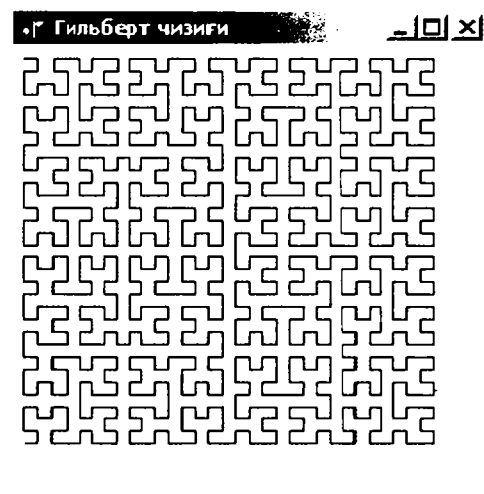
```

### Gilbert chizig'i

Quyidagi rasmda Gilbertning birinchi, ikkinchi va uchinchi darajali egri chiziqlari keltirilgan:



Gilbert egri chizig'ini chizish algoritmi rekursivdir. Quyida shu algoritmgaga asoslangan dastur matni berilgan. Rasm chizish **Chizish** (button1) tugmasini bosib amalga oshiriladi. Tugma bosilgandan so'ng visible xossasiga false qiymati berilgani uchun tugma oynada ko'rinmay qoladi. Quyidagi rasmda dastur ishlash jarayonida chizilgan Gilbertning beshinchi darajali chizig'i aks ettirilgan:



Gilbert chizig'i dasturining matni:

**Unit Unit1;**

**interface**

**uses**

**Windows, Messages, SysUtils,**

**Variants, Classes, Graphics,**

**Controls, Forms, Dialogs, StdCtrls, ComCtrls;**

**Type**

**TForm1 = class(TForm)**

**Button1: TButton;**

**procedure Button1Click(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation {\$R \*.dfm}**

**Var**

**p: integer =5;**

**u: integer =7;**

**procedure a(i:integer; canvas: TCanvas); forward;**

**procedure b(i:integer; canvas: TCanvas); forward;**

**procedure c(i:integer; canvas: TCanvas); forward;**

**procedure d(i:integer; canvas: TCanvas); forward;**

**procedure a(i: integer; canvas: TCanvas);**

**begin**

**if i > 0 then begin**

**d(i-1, canvas);**

**canvas.LineTo(canvas.PenPos.X+u,canvas.PenPos.Y);**

**a(i-1, canvas);**

**canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);**

**a(i-1, canvas);**

**canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);**

**c(i-1, canvas);**

**end;**

**end;**

**Procedure b(i: integer; canvas: TCanvas);**

**begin**

**if i > 0 then begin**

```

c(i-1, canvas);
canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);
b(i-1, canvas);
canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
b(i-1, canvas);
canvas.LineTo(canvas.PenPos.X+u, canvas.PenPos.Y);
d(i-1, canvas);
end;
end;

```

```

Procedure c(i: integer; canvas: TCanvas);

```

```

begin
if i > 0 then begin
b(i-1, canvas);
canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
c(i-1, canvas);
canvas.LineTo(canvas.PenPos.X-u,canvas.PenPos.Y);
c(i-1, canvas);
canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);
a(i-1, canvas);
end;
end;

```

```

Procedure d(i: integer; canvas: TCanvas);

```

```

begin
if i > 0 then begin
a(i-1, canvas);
canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y+u);
d(i-1, canvas);
canvas.LineTo(canvas.PenPos.X+u,canvas.PenPos.Y) ;
d(i-1, canvas);
canvas.LineTo(canvas.PenPos.X,canvas.PenPos.Y-u);
b(i-1, canvas);
end;
end;

```

```

Procedure TForm1.Button1Click(Sender: TObject);

```

```

begin
button1.Visible:=false;
Form1.Canvas.MoveTo(p, u) ;
a(5,Form1.Canvas);
end;
end.

```

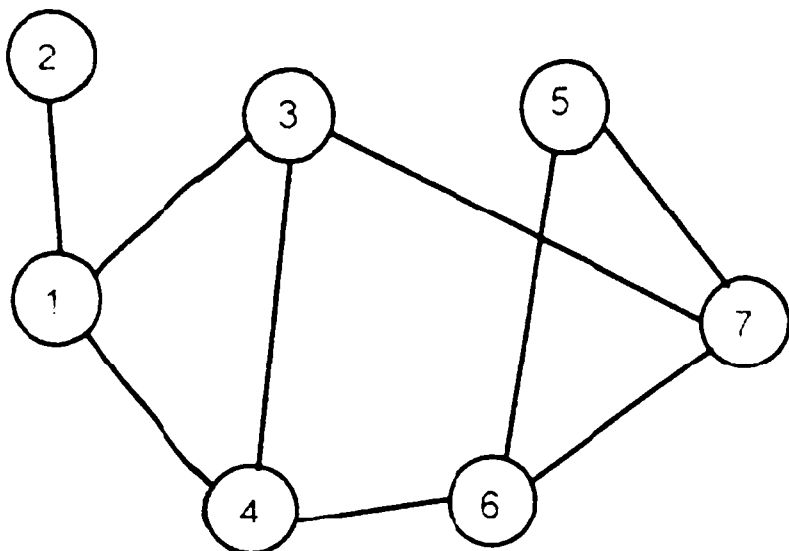


## 6.4. Graflarga rekursiyani qo'llash

### Yo'l izlash

Rekursiya qo'llashga yana bir misol ikki shahar orasiga yo'l topish masalasidir.

Yo'llar xaritasi graf sifatida tasvirlanishi mumkin.



Yo'l izlash jarayoni qadamlar ketma-ketligidan iborat bo'ladi. Har bir shahardan borish mumkin bo'lgan shahar tanlanadi. Agar bu shahar berilgan so'nggi shahar bo'lsa jarayon tugaydi. Bu shahardan yana yangi shaharga o'tiladi. Agar o'tish mumkin bo'lgan shahar mavjud bo'lmasa, bir qadam orqaga qaytiladi.

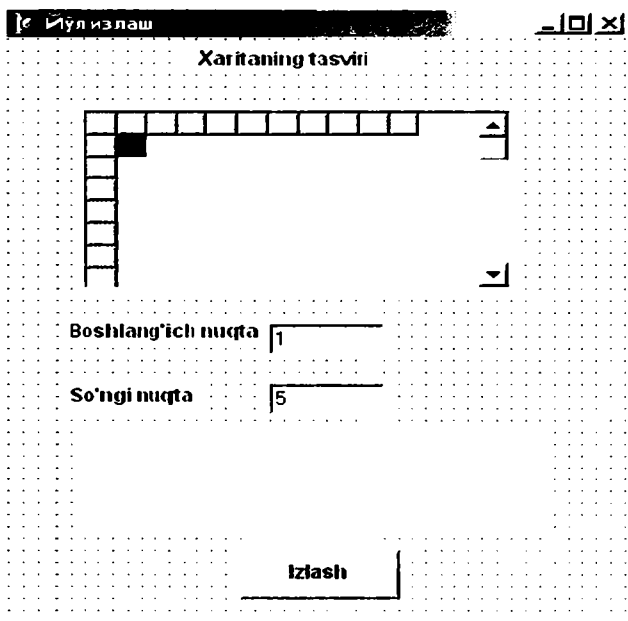
Izlash algoritmi rekursiv xarakterga ega.

Graf ikki o'lchovli massiv ko'rinishida beriladi. Bu massivni map (xarita) deb ataymiz. Massiv  $map[i, j]$  elementi qiymati – bu  $i$  va  $j$  shaharlari orasidagi masofaga teng. Agar bu shaharlarni bog'lovchi to'g'ri yo'l mavjud bo'lmasa nolga tengdir.

Keltirilgan graf uchun map massivini quyidagi jadval ko'rinishida tasvirlash mumkin:

	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	1	0	0	1
4	1	0	1	0	0	1	0
5	0	0	0	0	0	1	1
6	0	0	0	1	1	0	1
7	0	0	1	0	1	1	0

Bu massivdan tashqari o'tilgan shaharlar nomerlarini o'z ichiga oluvchi road (yo'llar) massivi va incl (include – qo'shish) massividan foydalanamiz. Agar i nomerli shahar marshrutiga kiritilgan bo'lsa inci[i] ga true qiymatini yozamiz.



Xaritani tasvirlovchi massivni kiritish uchun stringGrid1 komponentasidan foydalanamiz. Natijalarni chiqarish uchun Label1 maydoni marshrut boshi va oxirini ko'rsatishi uchun Edit1 va Edit2 maydonlari ishlatiladi. Izlash protsedurasi Izlash (Button1) tugmasini bosish bilan chaqiriladi. Label2, Label3 i Label4 maydonlari izoh uchun ishlatiladi.

StringGrid1 komponentasi xossalarning qiymatlari:

Xossa	Qiymat
Name	StringGrid1
ColCount	11
RowCount	11
FixedCols	1
FixedRows	1
Options . goEditing	TRUE
DefaultColWidth	16
DefaultRowHeight	14

**Unit Unit1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Classes,**

**Graphics, Controls, Forms,**

**Dialogs, StdCtrls, Grids;**

**Type**

**TForm1 = class(TForm)**

**StringGrid1: TStringGrid;**

**Edit1: TEdit;**

**Edit2: TEdit;**

**Label1: TLabel;**

**Label2: TLabel;**

**Label3: TLabel;**

**Button1: TButton;**

**Label4: TLabel;**

**procedure FormActivate(Sender: TObject);**

**procedure Button1Click(Sender: TObject);**

**private**

**{ Private declarations } public**

**{ Public declarations } end;**

**Var**

**Form1: TForm1;**

**implementation**

```

{$R *.DFM}
procedure TForm1.FormActivate(Sender: TObject);
Var
i:integer; begin
for i:=1 to 10 do
StringGrid1.Cells[0,i]:=IntToStr(i);
for i:=1 to 10 do
StringGrid1.Cells[i,0]:=IntToStr(i);
StringGrid1.Cells[1,2]:='1';
StringGrid1.Cells[2,1]:='1';
StringGrid1.Cells[1,3]:='1';
StringGrid1.Cells[3,1]:='1';
StringGrid1.Cells[1,4]:='1';
StringGrid1.Cells[4,1]:='1';
StringGrid1.Cells[3,7]:='1';
StringGrid1.Cells[7,3]:='1';
StringGrid1.Cells[4,6]:='1';
StringGrid1.Cells[6,4]:='1';
StringGrid1.Cells[5,6]:='1';
StringGrid1.Cells[6,5]:='1';
StringGrid1.Cells[5,7]:='1';
StringGrid1.Cells[7,5]:='1';
StringGrid1.Cells[6,7]:='1';
StringGrid1.Cells[7,6]:='1';
end;
procedure TForm1.Button1Click(Sender: TObject);
const
N=10;
Var
map:array[1..N,1..N] of integer;
rgad:array[1..N]of integer;
incl:array[1..N]of boolean;
start,finish:Integer;
found:boolean; i,j:integer;
procedure step(s,f,p:integer);
Var
c:integer;
i:integer;
begin
if s=f then begin
found:=TRUE;

```

```

Label4.caption:=Label4.caption+#13+'Yo'l';
for i:=1 to p-1 do
Label4.caption:=Label4.caption+'
+IntToStr(rgad[i]); end
else begin
for c:=1 to N do
begin
if(map[s,c]<> 0)and(NOT incl[c])
then begin
road[p]:=c;
incl[c]:=TRUE;
step(c,f,p+1); incl[c]:=FALSE; rgad[p]:=0;
end;
end;
end;
end;
end;
Begin
Label1.caption:=' `';
for i:=1 to N do rgad[i]:=0;
for i:=1 to N do incl[i]:=FALSE;

for i:=1 to N do
for j:=1 to N do
if StringGrid1.Cells[i,j] <> ``
then map[i,j]:=StrToInt(StringGrid1.Cells[i, j])
else map[i,j]:=0;
start:=StrToInt(Edit1.text);
finish:=StrToInt(Edit2.text);
road[1]:=start;
incl[start]:=TRUE;

step(start,finish,2);
if not found
then Label1.caption:='Nuqtalar tutashtirilmagan';
end;
end.

```

### Eng qisqa yo'l topish

Eng qisqa yo'l topish uchun quyidagi usuldan foydalanish mumkin. Avval eng birinchi marshrut topib olinadi va u eng qisqa yo'l deb qaraladi. Yangi yo'l izlash davomida, biror nuqtani marshrutga qo'shish

natijasida marshrut uzunligi oldin topilgan yo'ldan oshib ketsa, bu nuqta o'tkazib yuborilib, keyingisi tekshiriladi. Shunday qilib, har bir topilgan yo'l oldingisidan qisqaroq bo'ladi.

Quyida shu usulni amalga oshiruvchi dastur matni berilgan:

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
const
```

```
N=10;
```

```
var
```

```
map:array[1..N,1..N] of integer;
```

```
road:array[1..N]of integer;
```

```
incl:array[1..N]of boolean;
```

```
start,finish:integer;
```

```
found:boolean; i,j:integer;
```

```
len:integer;
```

```
c_len:integer;
```

```
procedure step(s,f,p:integer);
```

```
Var
```

```
c:integer;
```

```
  i:integer;
```

```
begin
```

```
  if s=f then begin
```

```
    found:=TRUE;
```

```
    len:=c_len;
```

```
    Label4.caption:=Label4.caption+#13+' Yo'l';
```

```
    for i:=1 to p-1 do
```

```
      Label4.caption:=Label4.caption+'
```

```
      +IntToStr(road[i]);
```

```
      Label4.caption:=Label4.caption
```

```
      +', uzunlik:'+IntToStr(len)+#13;
```

```
    end
```

```
  else begin
```

```
    for c:=1 to N do
```

```
      begin
```

```
        if(map[s,c]<> 0)and(NOT incl[c])
```

```
          and((len=0)or(c_len+map[s,c]< len))
```

```
        then begin
```

```
          road[p]:=c;
```

```
          incl[c]:=TRUE;
```

```

    c_len:=c_len+map[s,c];
step(c,f,p+1); incl[c]:=FALSE; road[p]:=0;
    c_len:=c_len-map[s,c];
end;
end;
end;
end;
begin
Label1.caption:=' ';
for i:=1 to N do road[i]:=0;
for i:=1 to N do incl[i]:=FALSE;
for i:=1 to N do
for j:=1 to N do
if StringGrid1.Cells[i,j] <> ''
then map[i,j]:=StrToInt(StringGrid1.Cells[i, j])
else map[i,j]:=0;
    len:=0;
c_len:=0;
start:=StrToInt(Edit1.text);
finish:=StrToInt(Edit2.text);
road[1]:=start;
incl[start]:=TRUE;

step(start,finish,2);
if not found
then Label1.caption:="Nuqtalar tutashtirilmagan!";
end;

```

## S a v o l l a r

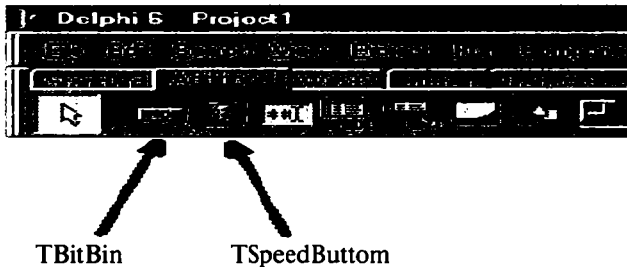
1. Yozuv qiymatini faylda saqlash uchun yozuv turidagi fayl qanday e'lon qilinadi?
2. Dinamik o'zgaruvchi deb qanday o'zgaruvchiga aytiladi?
3. Qaysi protsedura faylga yozuv qo'shishni amalga oshiradi?
4. Qaysi protsedura faylni ochib yozuvlarni ketma-ket o'qiydi?
5. Ko'rsaigich deb qanday o'zgaruvchiga aytiladi?
6. Dinamik o'zgaruvchilar deb qanday o'zgaruvchiga aytiladi?
7. Rekursiya nima va rekursiv funksiya deb qanday funksiyaga aytiladi?
8. Rekursiyaga misollar keltiring.

## VII. DELPHI QO‘SHIMCHA KOMPONENTALARI

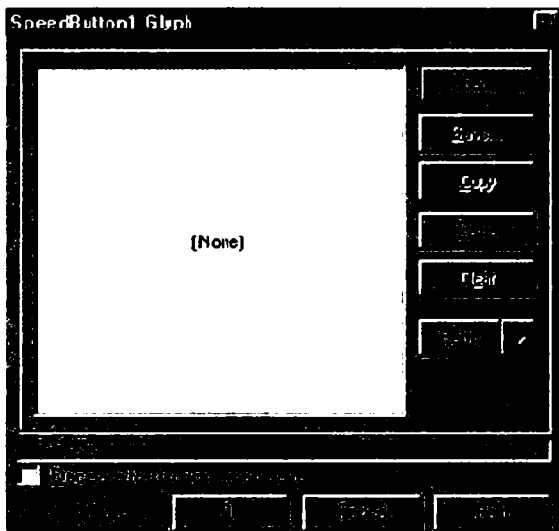
### 7.1. ADDITIONAL sahifasining komponentalari

#### TSpeedButton va TBitBtn tugmalari

Bu tugmalar *TButton* vazifalarini bajaradi. Yagona farqi matndan tashqari rasmlarni ham aks ettiradi. *TSpeedButton* tugmasi fokus olmaydi. Bu shuni bildiradiki, agar matn qatorida satr terib, bu tugma bosilsa, shu hodisa qayta ishlangandan so‘ng fokus yana matn qatoriga qaytib keladi. TAB tugmasi bilan bu tugmani ajratib bo‘lmaydi.

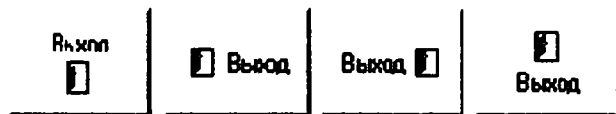


Tugmaga rasm o‘rnatish uchun ikki marta *Glyph* xossasi qatoriga chertish lozim. Natijada rasm paydo bo‘lgan yuklash oynasida *Load* tugmasini bosish lozim. Ko‘p rasmlar Program Files\Common Files\Borland Shared\Images\Buttons katalogida joylashgandir.





**TBitBtn** va **TSpeedButton** tugmalari deyarli bir xil xossalarga egadir. Ular uchun umumiy *Layout* xossasi rasm va matnning o'zaro joylashuvini o'zgartirishga imkon beradi. Quyidagi rasmda har xil qiymatlarga mos variantlari ko'rsatilgan:

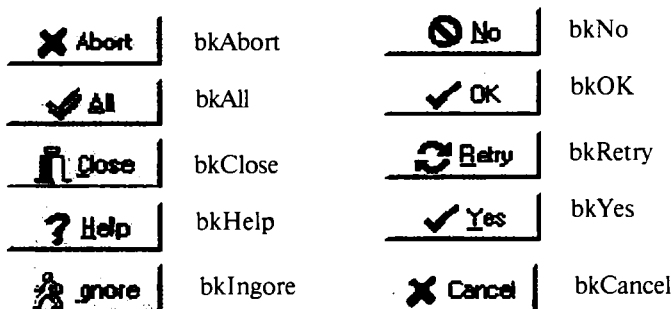


blGlyphBottom blGlyphLeft blGlyphRight blGlyphTop

**TBitBtn** tugmasining yana bir xossasi *Kind* bo'lib, oldindan tayyorlangan standart tugmalarni tanlash imkonini beradi. Quyidagi rasmda standart tugmalar va ularga mos qiymatlarni ko'rish mumkin.

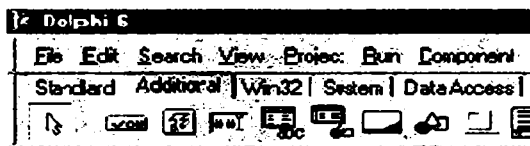
Yana bir xossa *ModalResult* – dialog oynasi uchun tugma qaytaradigan natijani tanlashga imkon beradi.

**TSpeedButton** tugmasining *GroupIndex* xossasi tugmalarni guruhlashga imkon beradi. Buning uchun bir guruhga tegishli tugmalarning *GroupIndex* xossasi bir xil qiymatga, masalan 1 ga teng bo'lishi kerak. Quruqlangan tugmalarning biri bosilsa, qolganlaridan ajralib qoladi. Buning uchun *Down* xossasi qiymati *true* ga teng bo'lishi kerak.



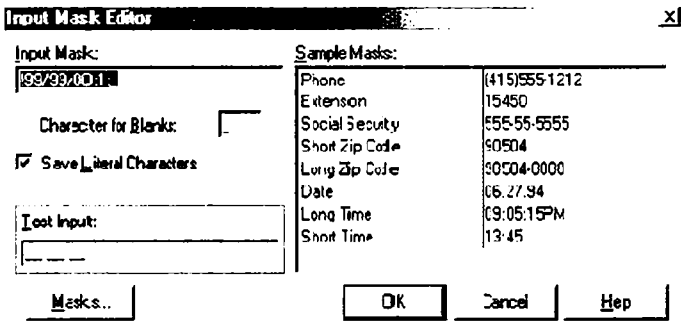
### Maskalangan kiritish qatori (TMaskEdit)

Bu komponent ma'lum formatga mos satr kiritishga imkon beradi.



TMaskEdit

Asosiy xossasi *EditMask* bo'lib, shu xossa qatoriga ikki marta chertilsa kiritish muharriri ochiladi.



*Input Mask* qatoriga maska kiritish mumkin. *Test Input* qatoriga maskani testlash mumkin.

Maska terish osondir. Agar qator to'rt raqamli son, tere va uch raqamli sonidan iborat bo'lishi kerak bo'lsa, *Input Mask* qatoriga 9999-999 kiritish mumkin.

### Siljitish yo'lchasiga ega panel (TScrollBox)

TScrollBox komponentasining oddiy Panel komponentasidan farqi siljitish yo'lchasiga ega bo'lishi mumkinligidir.

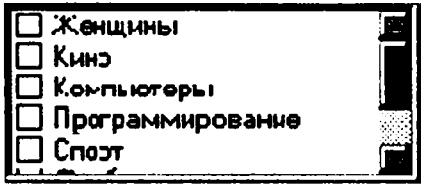
 - TScrollBox

Formaga *TScrollBox* komponentasini o'rnatib, uning ichiga (*TImage*) komponentasini o'rnatib. Endi *Image1* ga katta rasm joylab, *AutoSize* xossasiga *true* qiymatini bering. Agar *Image1* komponentasida rasm kattaligini olib *ScrollBox* chegarasiga sig'may qolsa, siljitish yo'lchalari paydo bo'ladi.

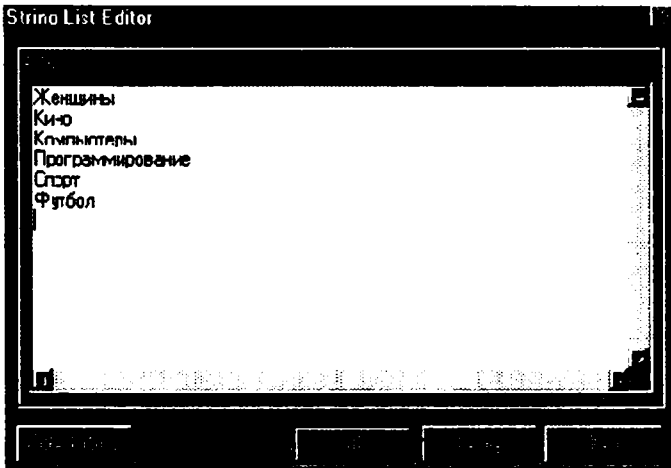


## Markirovka qilingan ro'yxat (TCheckBox)

*TCheckBox* komponentasi *TListBox* komponentasiga juda o'xshash, faqat har bir ro'yxat yonida *TCheckBox* komponentasidagi kabi ajratish to'rtburchagi mavjuddir.

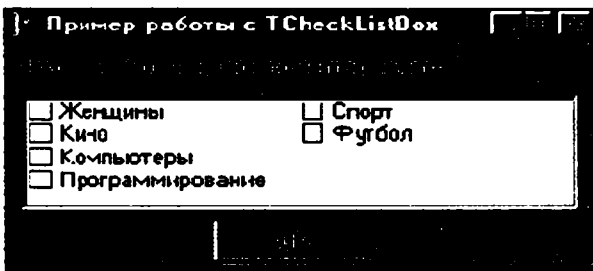


Ro'yxat kiritish uchun *Items* xossasi qatoriga ikki marta chertish lozim.



*TCheckBox* yana bir xossasi — *columns*, ya'ni ustunlar sonidir. Agar bu xossa qiymati birdan katta bo'lsa va ro'yxat bir ustunga sig'masa, ko'rsatilgan sonli ustunlarga ajratiladi.

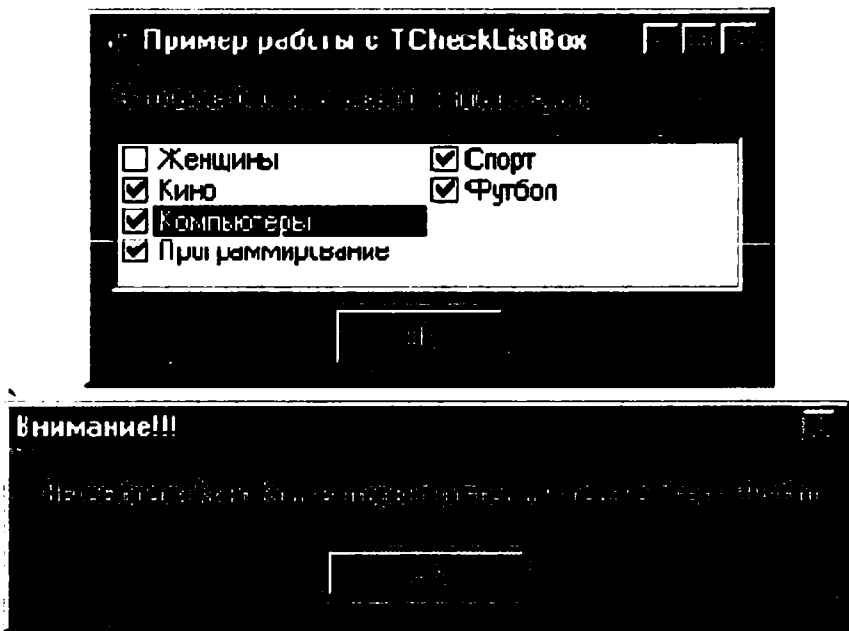
Quyida shu elementdan foydalanilgan dastur formasini keltiramiz:



OK tugmasining *OnClick* hodisasi uchun quyidagi protsedurani kiritamiz:

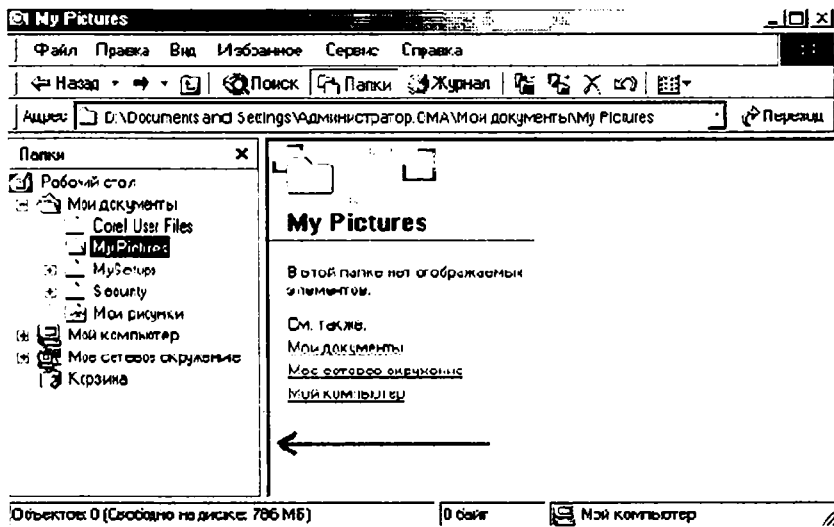
```
Procedure TForm1.OKButtonClick(Sender: TObject);  
var  
i:Integer;  
Str: `String`;  
begin  
Str:='Siz tanladingiz';  
for i:=0 to CheckListBox1.Items.Count-1 do  
if CheckListBox1.Checked[i] then  
Str:=Str+CheckListBox1.Items[i]+' ` `;  
Application.MessageBox(PChar(Str), 'Diqqat!!!');  
end;
```

Dasturni ishlash natijasiga misol:

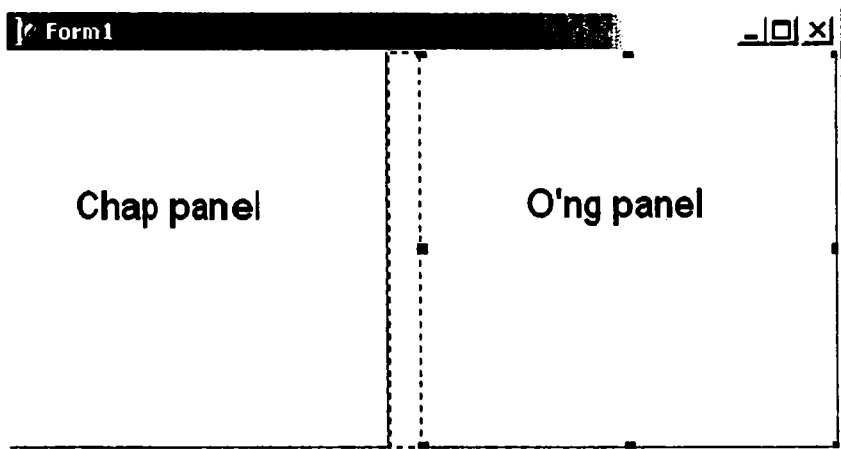


### Ajratish yo'lchasi (TSplitter)

Agar Windows Explorerni ochib ko'rsak oynasi ikkiga ajralgan bo'lib, o'rtasida siljitish mumkin bo'lgan yo'lchani ko'ramiz. Mana shu effekt **TSplitter** komponentasini yaratishga imkon beradi.



*TSplitter* komponentasidan foydalanishga misol. Formaga panel (*TPanel*) komponentasini joylashtirib *Align* xossasiga *alLeft* qiymatini beramiz va *Caption* xossasiga «Chap panel» qatorini kiritamiz. Formaga *TSplitter* joylashtirib *Align* xossasiga *alClient* qiymat beramiz. Yana bir panel joylashtirib *Align* xossasiga *alClient* qiymat beramiz va *Caption* xossasiga «O'ng panel» qatorini kiritamiz. Natija rasmda ko'rsatilgan.



Dasturni ishga tushirib, ajratish yo'lchasi sichqoncha bilan harakatlantirilsa panellar kattaligi o'zgaradi.

## Ko'p qatorli matn (TStaticText)

Ko'pincha dasturda bir necha qatorli matn chiqarishga to'g'ri keladi. Buning uchun formaga bir necha *TLabel* komponentasini o'rnatish mumkin. Lekin osonroq *TStaticText* komponentasini o'rnatib *AutoSize* xossasiga false qiymatini berishdir. Rasmda shu komponentadan foydalanishga misol keltirilgan.



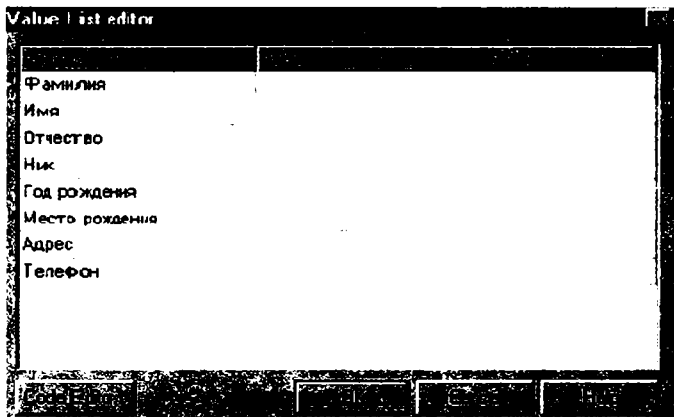
## Parametrlar muharriri (TValueListEditor)



Bu komponenta obyektlar inspektoridagi kabi xossalar muharririni yaratishga imkon beradi.

Asosiy xossalari:

- *DefaultColumnWidth* – ustunlarning ko'zda tutilgan kengligi;
- *DefaultColumnHeight* – ustunlarning ko'zda tutilgan kengligi;
- *DisplayOption* – komponentani akslantirish opsiyalari;
- *TitleCaptions* – sarlavha nomlari. Agar ikki marta chertilsa oddiy matn muharriri chiqadi.
- *FixedColor* – fiksirlangan ustun rangi.
- *FixedCols* – fiksirlangan ustun indeksi.
- *KeyOption* – kalitning maydon opsiyalari.
- *Strings* – xossalar nomlari. Shu qatorga ikki marta chertilsa xossalar muharriri chiqadi.



Formaning OnShow hodisasi uchun quyidagi protsedurani yaratamiz:

```
Procedure TForm1.FormShow(Sender: TObject);
begin
  ValueListEditor1.ItemProps[6].EditStyle:=esPickList;
  ValueListEditor1.ItemProps[6].PickList.Add('Moskva');
  ValueListEditor1.ItemProps[6].PickList.Add('Piter');
  ValueListEditor1.ItemProps[6].PickList.Add('Rostov-na-Donu');
  ValueListEditor1.ItemProps[4].EditMask:='99/99/9999';
end;
```

*ItemProps* xossasida ro'yxat elementlari xossasi joylashgan. Agar 3-element xossasini o'zgartirish lozim bo'lsa, *ValueListEditor1.ItemProps[2]*ni yozish kerak.


*EditStyle* – xossasi tahrirlash uslubini o'rnatadi. (*ValueListEditor1.ItemProps[6].EditStyle*):= *esPickList* instruksiyasi qator tugma bosilganda chiquvchi qatorga aylantiradi.

*ValueListEditor1.ItemProps[6].PickList.Add (текст элемента)* buyrug'i oltinchi qatorga satr qo'shadi.

*EditMask* – xossasi kiritish maskasini yaratishga imkon beradi.

## 7.2. WIN 32-sahifasining komponentalari

### Sarlavhalar ro'yxati (TTabControl)

 Bu komponenta formaga MS Word «Параметры» bo'limiga o'xshash sarlavhalar menyusini yaratishga imkon beradi.

Asosiy xossalari:

*TabHeioht* – Sarlavhalar balandligi. Agar 0 ko'rsatilsa ko'zda tutilgan qiymat olinadi.

*Tab Index* – ajratilgan sarlavha indeksining nomeri. Nomerlash 0 dan boshlanadi.

*TabPosition* – sarlavhalar pozitsiyasi.

Qiymatlari:

- *tpBottom* – pastda;
- *tpLeft* – chapda;
- *tpRight* – o'ngda;
- *tpTop* – yuqorida;

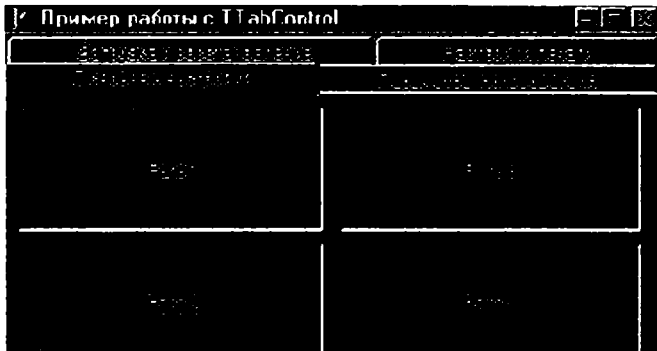
*Tabs* – sarlavha nomlari. Agar bu qatorga ikki marta chertilsa soddah muharrir chaqiriladi.

*MultiLine* – Agar bu xossaga *true* qiymati berilsa, sarlavhalar bir necha qatorga joylashadi.

*HotTrack* – Agar bu xossaga *true* qiymati berilsa, sarlavhalar sichqoncha keltirilganda ajraladi.

*Style* – sarlavhalarni akslantirish uslubi.

Quyidagi misolda bitta *TTabControl* komponentasi va 4 ta *Panel* komponentasidan foydalanilgan. Dastur formasining ko‘rinishi:



Hamma panellar *TTabControl* komponentasiga joylashtirilib, ularning *Align* xossasi *alClient* qiymatiga ega bo‘lishi kerak.

*TTabControl* komponentasi *OnChange* xossasi uchun quyidagi protsedurani yozishi lozim:

```
Procedure TForm1.OptionsTabChange(Sender: TObject);  
begin  
Panel1.Visible:=false;  
Panel2.Visible:=false;  
Panel3.Visible:=false;  
Panel4.Visible:=false;  
case OptionsTab.TabIndex of  
0: Panel1.Visible:=true;  
1: Panel2.Visible:=true;  
2: Panel3.Visible:=true;  
3: Panel4.Visible:=true;  
end;  
end;
```

### **Sahifalar to‘plami (TPageControl)**

Bu komponenta *TTabControl* komponenta xossalariga ega bo‘lib, qo‘shimcha imkoniyatlarga egadir.



Agar komponentada sichqonchani o'ng tugmachasi bosilsa, menyuni oynasi paydo bo'ladi.



Bu menyuni yuqorisida 4 ta punkt mavjud:

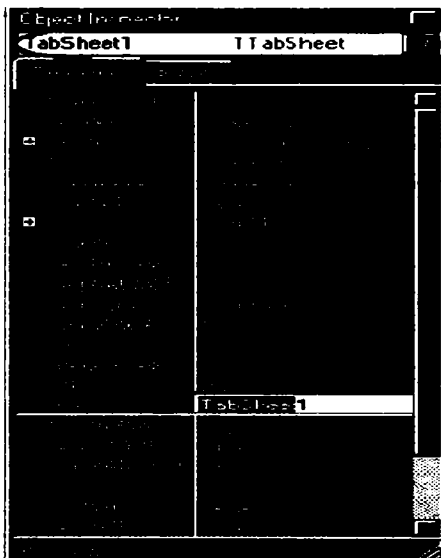
*New Page* – yangi sahifa yaratish;

*Next Page* – keyingi sahifaga o'tish;

*Previous Page* – oldingi sahifaga o'tish;

*Delete Page* – ajratilgan sahifani o'chirish.

Har bir sahifa alohida komponenta bo'lib, obyektlar inspektoriga xossalarni o'rnatish mumkin. Masalan, *TabSheet1* komponentasi xossalarning ko'rinishi:

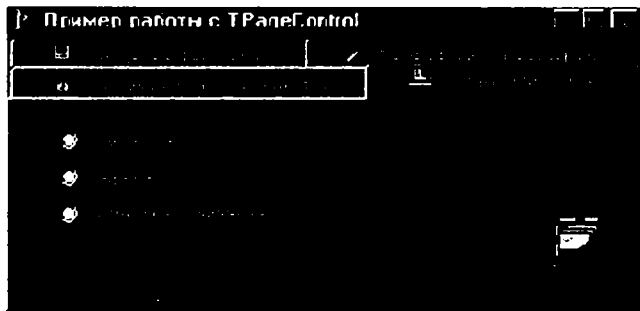


Har bir sahifa quyidagi asosiy xossalarga ega:

*Caption*, – sahifa nomi.

*ImageIndex* – rasm tanlashga imkon beradi. Buning uchun formaga *TImageList* komponentasini o‘rnatib, rasmlarni joylashtirish lozim. Shundan so‘ng *PageControl* komponentasi *Images* xossasida *ImageListni* ko‘rsatishi lozim. Shundan so‘ng sahifalardagi *Image Index* ro‘yxatida rasmlar paydo bo‘ladi.

Komponentadan foydalanishga misol ko‘ramiz. Dastur formasining ko‘rinishi:



### Rasmlar ro‘yxati (TImageList)

Bu komponenta rasmlarini qulay shaklda saqlash uchun ishlatiladi.



Asosiy xossalari:

*Height* – rasmlar kengligi;

*Width* – rasmlar balandligi.

Massivdan biror rasmni olish uchun *GetBitmap* usulidan foydalanish lozim. Misol uchun massivdan to‘rtinchi rasmni olish uchun:

**Var**

**bitmap:TBitmap;**

**begin**

**ImageList1.GetBitmap(3, bitmap);**

**end;**

Rasmlar massivda 0 dan boshlab nomerlanadi.

### Diapazonli tanlash (TTrackBar)

*TTrackBar* komponentasi biror diapazondan qiymat tanlash imkoniyatini yaratish uchun ishlatiladi.



Eng sodda ko‘rinishi.



Asosiy xossalari:

*Frequency* – chizish chastotasini ko‘rsatuvchi parametr;

*Max* – maksimal qiymat;

*Min* – minimal qiymat;

*Orientation* – ko‘rinishi. Bu xossa ikki qiymatga ega: *trHorizontal* (горизонтал) va *trVertical* (вертикал);

*Position* – joriy pozitsiya;

*SelStart* – ajratilgan bir necha qiymatning boshlanishi;

*SelEnd* – ajratilgan qiymatning oxiri;

*SliderVisible* – ko‘rsatgichni ko‘rsatish yoki ko‘rsatmaslik;

*TickMarks* – chizish o‘rni.

Qiymatlari:

- *tmBottomRight* – pastda;
- *tmBoth* – pastda va yuqorida;
- *tmTopLeft* – yuqorida.

*TickStyle* – chizish uslubi.

Qiymatlari:

- *tsAuto* – avtomatik chizish;
- *tsManual* – birinchi oxirgisini chizish;
- *tsNone* – chizishni man etish.

Quyidagi dasturda uchta *TTrackBar* komponentasi va uchta *TLabel* komponentasidan foydalanilgan.



Birinchi TTrackBar komponentasining *OnChange* hodisasi uchun quyidagi protsedurani yozamiz:

```
Procedure TForm1.TrackBar1Change(Sender: TObject);  
begin  
Label1.Caption:=IntToStr(TrackBar1.Position);  
end;
```

Bu misolda joriy pozitsiyani qatorga aylantirib Label1 ga yozamiz. Xuddi shunday protseduralarni qolgan TTrackBar komponentalari uchun ham yozish kerak.

### Jarayon indikator (TProgressBar)

Bu komponenta biror jarayonning qancha foizi bajarilganligini ko'rsatishga imkon beradi.



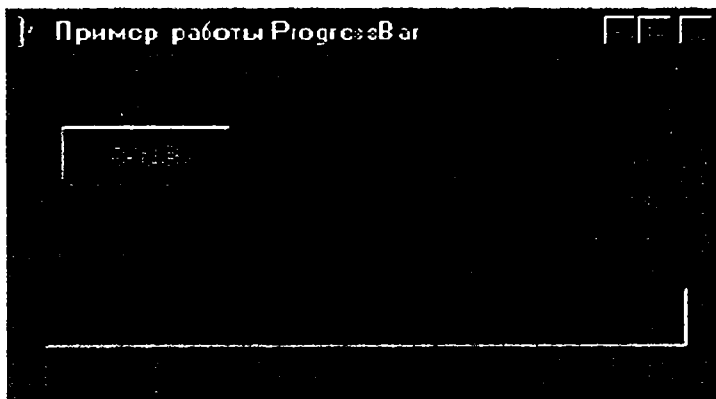
Asosiy xossalari:

*Max* – maksimal qiymat (alohida ko'rsatilmasa = 100).

*Min* – minimal qiymat (alohida ko'rsatilmasa = 0).

*Position* – pozitsiya.

Quyidagi dasturda 100 siklli hisoblash o'tkaziladi.



Boshqarish tugmasi uchun quyidagi protsedurani yozamiz:

```
Procedure TForm1.Button1Click(Sender: TObject);  
var  
i:Integer;
```

```

begin
for i:=0 to 20 do
begin
ProgressBar1.Position:=ProgressBar1.Position+5;
Sleep(100);
end;
ProgressBar1.Position:=0;
end;

```

Bu protsedurani quyidagicha yozish ham mumkin:

```

Procedure TForm1.Button1Click(Sender: TObject);
var
i:Integer;
begin
for i:=0 to 20 do
begin
ProgressBar1.Position:=i;
Sleep(100);
end;
ProgressBar1.Position:=0;
end;

```

### Chiqariluvchi sanalar ro'yxati (TDateTimePicker)

Bu komponenta chiqariluvchi ro'yxatga o'xshagan bo'lib, ro'yxat o'rniga kalendar joylashgan.



Quyidagi rasmda kalendar ko'rinishi berilgan.



Bu komponenta xossalari **TComboBox** xossalari bilan bir xildir, lekin quyidagi alohida xossalari mavjud:


*Date* – tanlangan sana;

*DateFormat* – bu xossaning ikki qiymati mavjud: *dfShort* – qisqa format va *dfLong* – uzun format;

*MaxDate* – maksimal sana;

*MinDate* – minimal sana.

## Kalendar (TMonthCalendar)

Bu komponenta formada doimiy kalendar ko'rsatishga imkon beradi. 

Kalendarning formada ko'rinishi quyidagicha:



Asosiy xossalari:

*FirstDayOfWeek* – birinchi sifatida ko'rsatilgan hafta kuni.

*Date* – tanlangan sana.

*MaxDate* – maksimal sana.

*MinDate* – minimal sana.


*MultiSelect* – oy sonlarining diapazonini tanlash imkoniyati;

*ShowToday* – joriy sanani ko'rsatish;

*ShowTodayCircle* – joriy sana doirasini ko'rsatish;

*WeekNumbers* – hafta nomerini ko'rsatish.

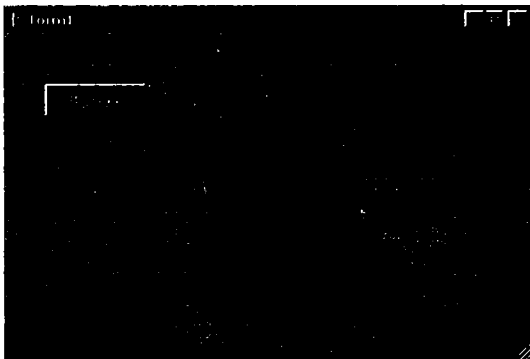
## Ma'lumotlar qatori (TStatusBar)

Bu komponenta formadagi biror komponenta ustiga sichqoncha ko'rsatkichi keltirilganda ma'lumotning qatori chiqishi uchun ishlatiladi. 

Komponenta formaga o'rnatilgandan so'ng quyidagilarni amalga oshirishi lozim:

1. Komponenta *Hint* xossasiga ma'lumotni kiritish.
2. Komponenta *ShowHint* xossasiga *true* qiymatini berish. Agar ajdod oynasining *ShowHint* xossasiga *true* qiymati va *Hint* xossasiga ma'lumot kiritilgan va *ParentShowHint* xossasiga *true* qiymati berilgan bo'lsa hamma komponentalar uchun ma'lumot chiqishi ta'minlanadi.
3. Ma'lumot hodisasi uchun protsedurani kiritish.

Formaga boshqarish tugmasini o'rnatamiz.



Boshqarish tugmasi *Hint* xossasiga «bu chiqish tugmasi» qatori yoziladi.

Dastur matni **private** bo'limiga *ShowHint* protsedurasi ta'rifini kiriting:

```
Private  
{ Private declarations }  
Procedure ShowHint(Sender: TObject);
```

Protsedura nomi o'zgacha bo'lishi mumkin, (masalan, *MyShowHint*) lekin parametri xuddi shunday bo'lishi kerak.

Endi Ctrl+Shift+C bosing. Delphi protsedura shablonini yaratadi:

```
Procedure TForm1.ShowHint(Sender: TObject);  
begin  
end;
```

O'zingiz ham { *TForm1* } satrdan so'ng shu shablonni kiritishingiz mumkin:

```
Implementation
```

```

{$R *.dfm}
{ TForm1 }
procedure TForm1.ShowHint(Sender: TObject);
begin
end;

```

Protsedura ichiga quyidagi kodni yozing:

```

Procedure TForm1.ShowHint(Sender: TObject);
begin
Status Bar1.SimpleText := Application.Hint;
end;

```

Endi *OnShow* hodisasi uchun qayta ishlovchi protsedura kiritamiz:

```

Procedure TForm1.FormShow(Sender: TObject);
begin
Application.OnHint := ShowHint;
end;

```

Bu vazifani soddaroq holda quyidagicha bajarish mumkin:

1. Dasturga *Additional* qatoridagi *TApplicationEvents* komponentasini qo'shish.
2. Bu komponentaning *Events* qatoridagi *OnHint* hodisasi uchun quyidagi satrni yozish «*StatusBar1.SimpleText := Application.Hint*».

### Uskunalar paneli (TToolBar i TControlBar)

Uskunalar paneli odatda menyudan so'ng joylashgan bo'ladi. Misol uchun MS Word uskunalar panelining ko'rinishi.



Bunday panellar TToolBar va TcontrolBar komponentalari yordamida yaratiladi.

ControlBar





## ToolBar

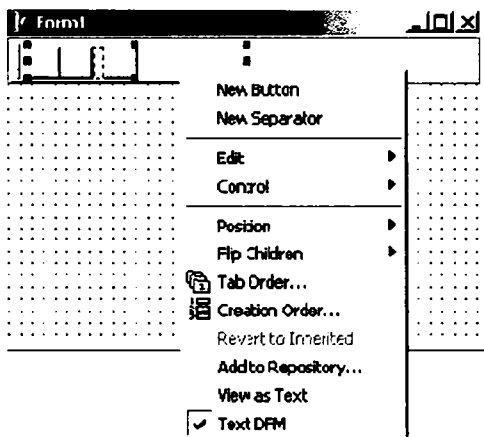


*ControlBar* komponentasining qulayligi shundaki, bu komponentaga uskunalar paneli joylashtirilsa, panelni komponenta ichiga siljitishi mumkin bo'ladi.

Ikkala komponenta uchun asosiy xossalar guruhi *EdgeBorders* bo'lib, bu guruhdagi *ebTop* xossasi qiymatiga *false* o'rnatilsa komponenta chetidagi hoshiya yo'qoladi.

DragMode	dmManual
<input type="checkbox"/> EdgeBorders	()
ebLeft	False
ebTop	False
ebRight	False
ebBottom	False
EdgeInner	esRaised
EdgeOuter	esLowered

*ToolBar* komponentasi ustiga sichqonchanning o'ng klavishi chertilsa, tugmalarni tahrirlash menyusi chiqadi. Bu menyuda *New Button* buyrug'i yangi tugma yaratadi. *New Separator* buyrug'i tugmalarni ajratuvchi oraliq yaratadi. Tugmani o'chirish uchun klaviaturadagi Del tugmasini ajratib bosish yetarli. Har bir yaratilgan tugma, yangi komponenta hisoblanib, obyektlar inspektorida xossalarni va hodisalarni ko'rish mumkin.

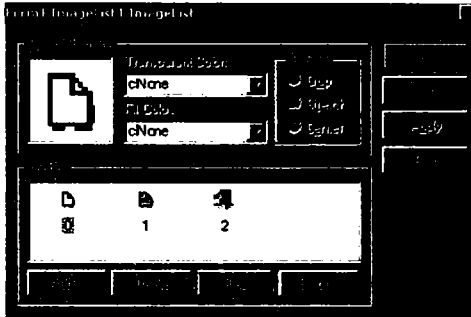


Agar uskunalar panelining *ShowCaptions* xossasiga *true* qiymati berilsa, tugmalarda rasmdan tashqari matnlarni aks ettirish mumkin bo'ladi.

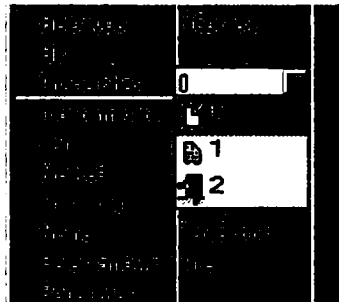
Agar *Flat* xossasiga *true*, qiymati berilsa, tugmalar chiroyliroq ko‘rinadi.

Tugmalarga rasm o‘rnatish uchun formaga *TImageList* komponentasi o‘rnatiladi.

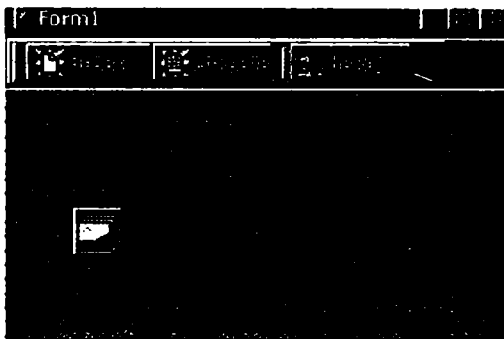
Uskunalar paneli *Images* xossasida rasmlar ro‘yxati ko‘rsatiladi.



Biror tugmadagi rasmni o‘zgartirish uchun *ImageIndex* xossasini o‘zgartirish lozim.



Rasm matndan chapda joylashishi uchun *List* xossasiga *true* qiymatini berish lozim.



Har bir tugma uchun chertish hodisasini qayta ishlovchi protsedura yaratiladi.

### 7.3. TTreeView va TListView komponentalari Elementlar daraxti (TTreeView)

WIN 32-sahifasiga tegishli elementlar daraxti murakkab, lekin ko'p ishlatiladigan komponentalardan biridir.

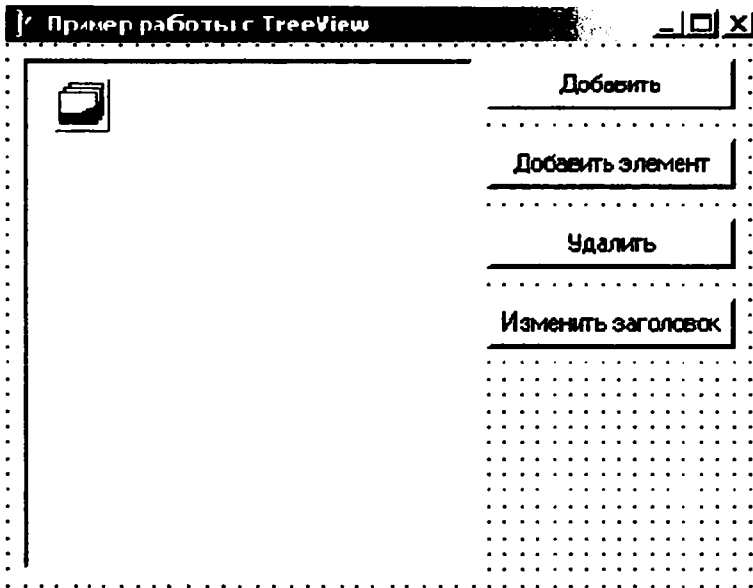


Komponentadan foydalanishni amaliy misolda ko'rib chiqamiz.

Yangi loyiha yaratib, formaga *TreeView* va *ImageList* komponentalari va uchta *TButton* tugmasini o'rnatib, ularning xossalarini quyidagicha o'zgartiring:

Caption	Name
1. Добавить	AddButton
2. Добавить элемент	AddChildButton
3. Удалить	DelButton
4. Изменить заголовок	EditButton

Forma quyidagi ko'rinishda bo'lishi kerak:



Daraxt Images xossasiga o'rnatilgan rasmlar ro'yxatini ulaymiz. «Добавить» tugmasi uchun quyidagi protsedurani yozamiz:

```
Procedure TTreeViewForm.AddButtonClick(Sender: TObject);  
var
```

```

CaptionStr:
NewNode:TTreenode;
begin
CaptionStr:='';
if not InputQuery('Ввод имени', 'Введите заголовок
элемента',CaptionStr) then exit;
NewNode:=TreeView1.Items.Add(TreeView1.Selected,
CaptionStr);
if NewNode.Parent<>nil then
NewNode.ImageIndex:=1;
end;

```

Bu yerda ikki o'zgaruvchi ishlatilgan: satr turidagi CaptionStr va TTreenode turidagi NewNode. TTreengde – daraxt elementining turidir.

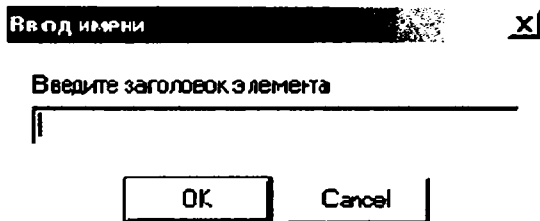
Birinchi qatorda CaptionStr qiymati bo'sh simvolga teng qilinadi. Ikkinchi qatorda element sarlavhasi kiritiladi.

```

if not InputQuery('Ввод имени', 'Введите заголовок элемента',
CaptionStr) then
exit;

```

Kiritish oynasining ko'rinishi.



Agar OK tugmasi bosilmagan bo'lsa protseduradan chiqiladi:

```

if not InputQuery(...) then
exit;

```

Keyingi qator yangi element qo'shadi:

```

NewNode:=TreeView1.Items.Add(TreeView1.Selected, CaptionStr);

```

TreeView1 komponentasining Items xossasida hamma elementlar saqlanadi.

Yangi element qo'shish uchun Add usuli chaqiriladi.

Add usuli ikki parametrga ega:

1. Yangi qo'shilayotgan element. Misolda ajratilgan element beriladi (TreeView1.Selected).

2. Yangi element sarlavhasi.

Natija *NewNode* o'zgaruvchida saqlanadi. Quyidagi satr rasmni o'zgartiradi:

```
if NewNode.Parent<>nil then
```

```
NewNode.ImageIndex:=1;
```

`Добавить элемент` tugmasi uchun quyidagi protsedurani yozamiz.

```
Var
```

```
CaptionStr:String;
```

```
NewNode:TTreeNode;
```

```
begin
```

```
CaptionStr:=``;
```

```
if not InputQuery(`Ввод имени подэлемента`,  
`Введите заголовок подэлемента`,CaptionStr) then exit;
```

```
NewNode:=TreeView1.Items.AddChild(TreeView1.Selected,  
CaptionStr);
```

```
if NewNode.Parent<>nil then
```

```
NewNode.ImageIndex:=1;
```

Bu protseduraning oldingisidan farqi *AddChild* usuli qo'llanilganligidir. Bu usulda avlod element qo'shadi. Qo'shilgan element ajdodi joriy elementdir.

`Удалить` tugmasi uchun protsedura:

```
if TreeView1.Selected<>nil then
```

```
TreeView1.Items.Delete(TreeView1.Selected);
```

Oldiniga ajratilgan element daraxtda mavjudligi tekshiriladi:

<http://www.vr-online.ru> 238

```
if TreeView1.Selected<>nil then
```

Agar mavjud bo'lsa *Delete* usuli qo'llaniladi:

```
TreeView1.Items.Delete(TreeView1.Selected);
```

`Изменить заголовок` tugmasi uchun quyidagi kodni kiritamiz:

```
Procedure TTreeViewForm.EditButtonClick(Sender: TObject);
```

```
var
```

```
CaptionStr:String;
```

```
begin
```

```
CaptionStr:=``;
```

```
if not InputQuery(`Ввод имени`,
```

```
`Введите заголовок элемента`,CaptionStr) then exit;
```

```
TreeView1.Selected.Text:=CaptionStr;
```

```
end;
```

*OnClose* hodisasi uchun quyidagi protsedurani yozamiz:

```
Procedure TTreeViewForm.FormClose(Sender: TObject;  
var Action: TCloseAction);  
begin  
  TreeView1.SaveToFile(ExtractFilePath(Application.ExeName)  
+ 'tree.dat');  
end;
```

Daraxtni saqlash uchun *SaveToFile* usuli chaqiriladi. To'la yo'lni ko'rsatish uchun quyidagi konstruksiyadan foydalaniladi:

```
ExtractFilePath(Application.ExeName) + 'tree.dat'
```

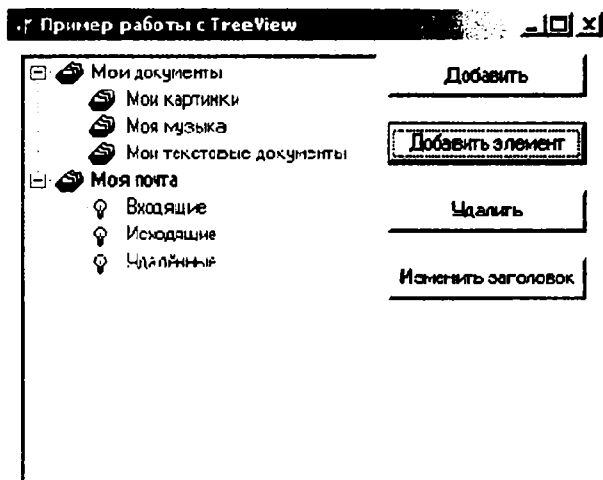
*Application.ExeName* – fayl nomi.

*ExtractFilePath* – faylga yo'l.


Endi saqlangan ma'lumotlarni yuklash uchun *OnShow* hodiasini qayta ishlovchi quyidagi protsedurani kiritamiz:

```
Procedure TTreeViewForm.FormShow(Sender: TObject);  
begin  
  if FileExists(ExtractFilePath (Application.ExeName) + 'tree.dat')  
then  
  TreeView1.LgadFromFile(ExtractFilePath(Application.ExeName)  
+ 'tree.dat');  
end;
```

Oldin *FileExists* funksiyasi yordamida fayl mavjudligi tekshiriladi. Agar mavjud bo'lsa *LgadFromFile* usuli chaqiriladi.



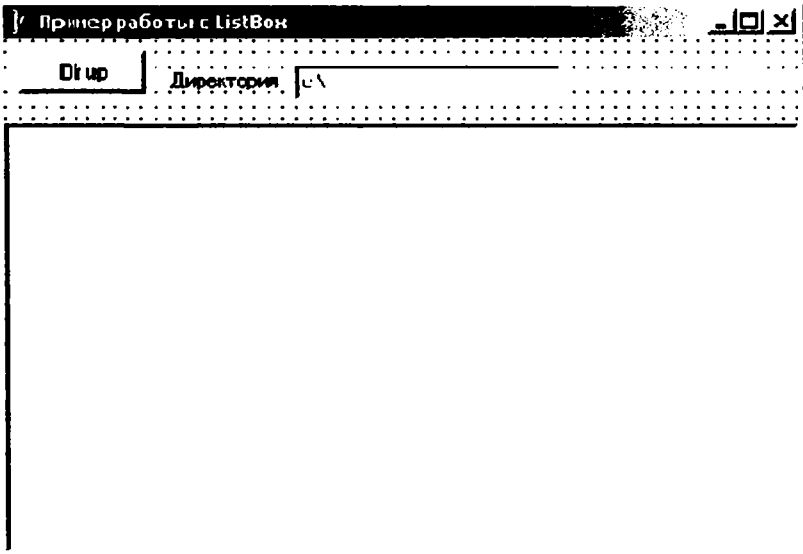
## Elementlar ro'yxati (TListView)

WIN 32-sahifasiga tegishli keyingi komponent Windows «Проводник» oynasining o'ng tomoniga o'xshagan ro'yxatni aks ettirishga imkon beradi. 

Bu komponentadan foydalanishni amaliy misolda ko'rib chiqamiz.

## Sodda fayl menedjeri

Loyiha formasiga bitta buyruq tugmasi, kiritish qatori va elementlar ro'yxatini o'rnatish.



Dastur uses bo'limiga shellapi modulini qo'shing.

Formaning OnCreate hodisasi uchun quyidagi protsedurani yozing:

```
Procedure TForm1.FormCreate(Sender: TObject);  
var  
SysImageList: uint;  
SFI: TSHFileInfo;  
begin  
  ListView1.LargeImages:=TImageList.Create(self);  
  ListView1.SmallImages:=TImageList.Create(self);  
  SysImageList := SHGetFileInfo("", 0, SFI,  
  SizeOf(TSHFileInfo), SHGFI_SYSICONINDEX or  
  SHOFI_LARGEICON);  
  if SysImageList <> 0 then  
  begin
```

```

ListView1.Largeimages.Handle := SysImageList;
ListView1.Largeimages.ShareImages := TRUE;
end;
SysImageList := SHGetFileInfo(`, 0, SFI; SizeOf(TSHFileInfo),
SHGFI_SYSICONINDEX or SHGFI_SMALLICON);
if SysImageList <> 0 then
begin
ListView1.Smallimages.Handle := SysImageList;
ListView1.Smallimages.ShareImages := TRUE;
end;
end;

```

Birinchi ikki qatorda kichik va katta ikonachalar ro'yxati yaratiladi. Katta ikonalar ro'yxatini hosil qilish uchun *SHGetFileInfo* funkstyasi chaqiriladi.

Endi *On.Show* hodisasi uchun protsedura kiritamiz. Bu protsedurada joriy direktoriyadan hamma fayllarni o'quvchi *AddFile* protsedurasini yaratamiz.

```

Procedure TForm1.FormShow(Sender: TObject);
begin
AddFile(Edit1.Text+'*. *',faAnyFile)
end;

```

Bu protsedura quyidagi ko'rinishga ega:

```

Function TForm1.AddFile(FileMask: string;
FFileAttr:DWORD): Boolean;
var
ShInfo: TSHFileInfo;
attributes: string;
FileName: string;
hFindFile: THandle;
SearchRec: TSearchRec;
function AttrStr(Attr: integer): string;
begin
Result
if (FILE_ATTRIBUTE_DIRECTORY and Attr) > 0 then
Result := Result + ` `;
if (FILE_ATTRIBUTE_ARCHIVE and Attr) > 0 then

```



```

Result := Result + `A`;
if (FILE_ATTRIBUTE_READONLY and Attr) > 0 then
Result := Result + `R`;
if (FILE_ATTRIBUTE_HIDDEN and Attr) > 0 then
Result := Result + `H`;
if (FILE_ATTRIBUTE_SYSTEM and Attr) > 0 then
Result := Result + `S`;
end;
begin
ListView1.Items.BeginUpdate;
ListView1.Items.Clear;
Result := False;
hFindFile := FindFirst(FileMask, FFileAttr, SearchRec);
if hFindFile <> INVALID_HANDLE_VALUE then
try
repeat
with SearchRec.FindData do
begin
if (SearchRec.Name = `.`) or (SearchRec.Name = `..`) or
(SearchRec.Name = ``) then continue;
FileName := SlashSep(Edit1.Text, SearchRec.Name);
SHGetFileInfo(PChar(FileName), 0, ShInfo, SizeOf(ShInfo);
SHGFI_TYPENAME or SHGFI_SYSICONINDEX);
Attributes := AttrStr(dwFileAttributes);
with ListView1.Items.Add do
begin
Caption := SearchRec.Name;
ImageIndex := ShInfo.iIcon;
SubItems.Add(IntToStr(SearchRec.Size));
SubItems.Add((ShInfo.szTypeName));
SubItems.Add(FileTimeToDateTimeStr(ftLastWriteTime));
SubItems.Add(attributes);
SubItems.Add(Edit1.Text + cFileName);
if (FILE_ATTRIBUTE_DIRECTORY and dwFileAttributes) > 0 then
SubItems.Add(`dir`)
else
SubItems.Add(`file`);
end;
Result := True;
end;

```

```

until (FindNext(SearchRec) <> 0);
finally
FindClose(SearchRec);
end;
ListView1.Items.EndUpdate;
end;

```

Bu protsedura o‘zgaruvchilarning e‘lon qilish bo‘limida lokal protsedura — **function AttrStr(Attr:Integer): string;** yaratilganidir.

Protsedura tanasida ListView1 ikki usuli chaqirilgan:

```

ListView1.Items.BeginUpdate;
ListView1.Items.Clear;

```

Birinchi usul *BeginUpdate* ro‘yxat elementlari o‘zgarishi boshlanganligi haqida xabar beradi. To *EndUpdate* chaqirilmaguncha o‘zgarishlar ekranda aks etmaydi.

Joriy ro‘yxat *ListView1.Items.Clear* usuli bilan tozalanadi.

Shundan so‘ng fayl izlash sikli boshlanadi:

**FindFirst** — izlashni boshlaydi.

Birinchi parametrlar izlash maskasi, masalan 'C:\\*. \*'. yoki 'C:\Fold\\*.exe'.

Ikkinchi parametrlar — fayl atributlari.

*faAnyFile* — ixtiyoriy fayllar.

*faReadOnly* — atributi *ReadOnly* bo‘lgan fayllar.

*faHidden* — berkitilgan fayllar.

*faSysFile* — sistema fayllari.

*faArchive* — arxiv fayllari.

*faDirectory* — direktoriyalarni izlash.

Oxirgi parametrlar izlash natijasi haqida ma‘lumot qaytaruvchi, ya‘ni fayl nomi, hajmi, yaratilish vaqti va hokazo.

Izlash funksiyasi nuqta yoki ikki nuqta qaytarishi mumkin. Bunday natija ko‘rilmaydi:

```

If (SearchRec.Name = '.') or (SearchRec.Name = '..') or
(SearchRec.Name = '') then continue;

```

Shundan so‘ng *SlashSep* funksiyasi chaqiriladi:

```

FileName := SlashSep(Edit1.Text, SearchRec.Name);

```

Bu funksiya va *FileTimeToDateTimeStr* var bo‘limida e‘lon qilingan:

```

var

```

```

Form1: TForm1;

```

```

function SlashSep(Path, FName: string): string;

```

```

function FileTimeToDateTimeStr(FileTime: TFileTime): string;

```

## implementation

*SlashSep* funksiyaning ko‘rinishi:

```
function SlashSep(Path, FName: string): string;  
begin  
if Path[Length(Path)] <> '\' then  
  Result := Path + '\' + FName  
else Result := Path + FName;  
end;
```

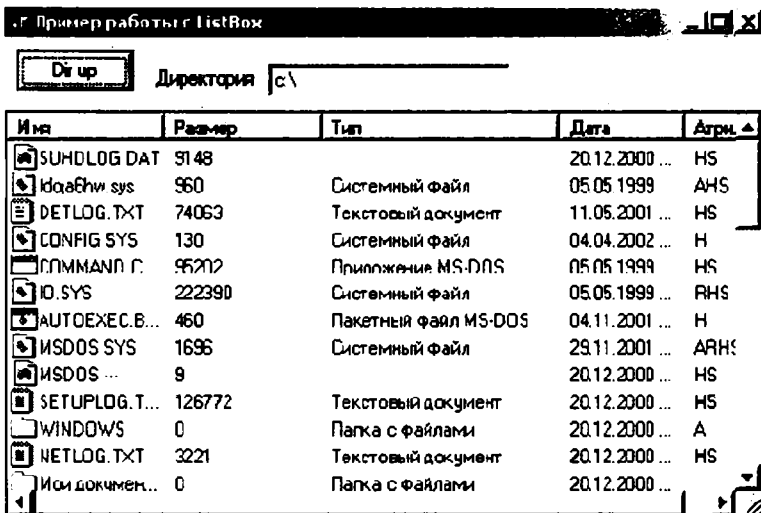
Bu funksiya faylning to‘la nomini hosil qiladi.

Shundan so‘ng SHGetFileInfo sistema funksiyasi chaqiriladi.

Keyingi satr ro‘yxatga yangi element qo‘shadi: *ListView1.Items.Add*.

Agar *ViewStyle* xossasiga *vsReport* qiymati berilsa jadval quyidagi ko‘rinishga ega bo‘ladi:

Yangi elementga qo‘shimcha ustunlar qo‘shish uchun *SubItems.Add* ('значение') bajarilishi lozim.



## Takomillashgan fayl menedjeri

Endi menedjerga direktoriyalarni aylanish va fayllarni ishga tushirish imkonini qo‘shamiz. Buning uchun *ListView* komponentasi *OnDoubleClick* hodisasi uchun quyidagi protsedurani yozamiz:

```
Procedure TForm1.ListView1DbClick(Sender: TObject);  
begin  
if (ListView1.Selected.SubItems[5] = 'dir') then
```

```

begin
Edit1.Text:=Edit1.Text+ListView1.Selected.Caption+'\';
AddFile(Edit1.Text+'*.*',faAnyFile)
end
else
ShellExecute(Application.MainForm.Handle, nil,
PChar(Edit1.Text+ListView1.Selected.Caption), "",
PChar(Edit1.Text), SW_SHOW);
end;

```

Agar ajratilgan qator direktoriya bo'lsa, joriy direktoriya nomini *Edit1.Text* da o'zgartiramiz va *AddFile* usulini chaqirib uni o'qiyamiz.

Agar ajratilgan qator fayl bo'lsa uni ishga tushiramiz. Buning uchun *ShellExecute* funksiyasini chaqiramiz. Uning quyidagi parametrlari mavjud:

1. Ilovani ishga tushirishga javob beruvchi dastur. Bu yerda nilni ko'rsatishi mumkin, lekin misolda dasturning asosiy oynasi ko'rsatilgan (*Application.MainForm.Handle*).

2. Bajarish kerak bo'lgan operatsiyani ko'rsatuvchi qator. Misolda nil ko'rsatilgan.

3. Faylga olib boruvchi yo'lni ko'rsatuvchi qator.

4. Buyruq qatorida dasturga uzatiluvchi parametrlar ro'yxati.

5. Ko'zda tutilgan direktoriya.

6. Akslantirish buyrug'i.

*SW\_SHOW* normal

*SW\_SHOWMAXIMIZED* maksimal

*SW\_SHOWMINIMIZED* minimal

*ShellExecute* funksiyasi *Shellapi* modulida e'lon qilingan, shuning uchun bu modulni uses bo'limida ko'rsatish lozim.

## S a v o l l a r

1. *TSpeedButton* va *TBitBtn* tugmalari vazifalari nima va ular *TButton* tugmasidan nima bilan farq qiladi?

2. Qaysi komponenta siljitiish yo'lchasini tashkil qiladi?

3. Maskalangan kiritish qatorini qaysi komponenta amalga oshiradi?

4. Ko'p qatorli matn kiritishni qaysi komponenta amalga oshiradi?

5. Qaysi komponenta sarlavhalar menyusini yaratishga imkon yaratadi?

6. Uskunalar panelini yaratishni qaysi komponentalar amalga oshiradi?

7. *TTreeView* va *TListView* komponentalariga tushintirish bering.

## VIII. MA'LUMOTLAR BAZASINING NAZARIY ASOSLARI

### 8.1. Ma'lumotlar bazasi haqida asosiy tushunchalar

Hozirgi kunda inson faoliyatida ma'lumotlar bazasi (**MB**) kerakli axborotlarni saqlash va undan oqilona foydalanishda juda muhim rol o'ynamoqda. Sababi jamiyat taraqqiyotining qaysi jabhasiga nazar solmaylik o'zimizga kerakli ma'lumotlarni olish uchun, albatta, MBga murojaat qilishga majbur bo'lamiz. Demak, MBni tashkil qilish axborot almashuv texnologiyasining eng dolzarb hal qilinadigan muammolaridan biriga aylanib borayotgani davr taqozosidir.

Informatsion texnologiyalarning rivojlanishi va axborot oqimlarining tobora ortib borishi, ma'lumotlarning tez o'zgarishi kabi holatlar insoniyatni bu ma'lumotlarni o'z vaqtida qayta ishlash choralarining yangi usullarini qidirib topishga undamoqda. Ma'lumotlarni saqlash, uzatish va qayta ishlash uchun MBni yaratish, so'ngra undan keng foydalanish bugungi kunda dolzarb bo'lib qolmoqda. Moliya, ishlab chiqarish, savdo-sotiq va boshqa korxonalar ishlarini ma'lumotlar bazasisiz tasavvur qilib bo'lmaydi.

Ma'lumki, **MB** tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishda foydalanish juda qiyin edi. Dastur tuzuvchilar ma'lumotlarni shunday tashkil qilar edilarki, u faqat qaralayotgan masala uchungina o'rinli bo'lardi. Har bir yangi masalani hal qilishda ma'lumotlar qaytadan tashkil qilinar va bu hol yaratilgan dasturlardan foydalanishni qiyinlashtirar edi.

Har qanday axborot tizimining maqsadi real muhit obyektlari haqidagi ma'lumotlarga ishlov berishdan iborat. Keng ma'noda ma'lumotlar bazasi – bu qandaydir bir predmet sohasidagi real muhitning aniq obyektlari haqidagi ma'lumotlar to'plamidir. Predmet sohasi deganda avtomatlashtirilgan boshqarishni tashkil qilish uchun o'rganilayotgan real muhitning ma'lum bir qismi tushuniladi. Masalan, korxonalar, zavodlar, ilmiy tekshirish instituti, oliy o'quv yurti va boshqalar.

Shuni qayd qilish lozimki, **MB**ni yaratishda ikkita muhim shartni hisobga olmoq zarur:

Birinchidan, ma'lumotlar turi, ko'rinishi ularni qo'llaydigan dasturlarga bog'liq bo'lmasligi lozim, ya'ni **MB**ga yangi ma'lumotlarni kiritganda yoki ma'lumotlar turini o'zgartirganda, dasturlarni o'zgartirish talab etilmasligi lozim.

Ikkinchidan, MBdagi kerakli ma'lumotni bilish yoki izlash uchun biror dastur tuzishga hojat qolmasin.

Shuning uchun ham MBni tashkil etishda ma'lum qonun va qoidalarga amal qilish lozim. Bundan buyon **axborot** so'zini **ma'lumot** so'zidan farqlaymiz, ya'ni **axborot** so'zini umumiy tushuncha sifatida qabul qilib, **ma'lumot** deganda aniq bir belgilangan narsa yoki hodisa sifatlarini nazarda tutamiz.

Ma'lumotlar bazasini yaratishda, foydalanuvchi axborotlarni turli belgilar bo'yicha tartiblashga va ixtiyoriy belgilar birikmasi bilan tanlanmani tez olishga intiladi. Buni faqat ma'lumotlarni tizimlashtirilgan holda bajarish mumkin.

*Tizilmalash tirish* – bu ma'lumotlarni tasvirlash usullari haqidagi kelishuvni kiritishdir. Agar ma'lumotlarni tasvirlash usuli haqida kelishuv bo'lmasa, u holda ular tizimlashtirilmagan deyiladi. Tizimlashtirilmagan ma'lumotlarga misol sifatida matn fayliga yozilgan ma'lumotlarni ko'rsatish mumkin.

Misol 1. Talabalar (sinov daftarchasining nomeri, familiyasi, ismi, otasining ismi, o'rtacha baho va stependiya miqdori) haqidagi axborotdan iborat tizimlashtirilmagan ma'lumotlar 1-rasmda ko'rsatilgan.

Reyting daftarcha nomeri 654311 Familiyasi Avazov Ismi Jamol  
Otasining ismi Aliyevich Tug'ilgan sana 15-yanvar. 1979-yil  
O'rtacha baho 4,78 Reyt. daft. nomeri 545712 F-yasi Ortiqov  
Ismi Akram Ota. ismi Salimovich Tug'ilgan sana 3/XI 1978-yil  
O'rta baho 4,61 Reyt. d. nomer 453225 F-yasi Lazizova Ismi  
Saida Otasi. Xo'jayevna Tug'ilgan sana 8.08.80 O'r. baho 4.52 R/  
d. nomeri 685564 Fam. Safarov Ismi Toshmurod Otasi Karimovich  
Tug'il. sana 12-apr., 81-y. O'rtacha b. 4,03 R/d. N 654786 F-ya  
Javlonov I. Alisher O. Ozodovich T. s 31/12/1982 O'. b. 3.69

*1-rasm.* Tizimlashtirilmagan ma'lumotlarga misol.

Tizimlashtirilmagan holda saqlanayotgan ma'lumotlardan zarur bo'lganini qidirib topish ancha murakkab, uni tartiblashni esa deyarli bajarib bo'lmaydi.

Bu ma'lumotlarni tizimlashtirish va qidirishni avtomatlashtirish uchun ma'lumotlarni tasvirlash usullari haqida ma'lum bir kelishuvni ishlab chiqish zarur.

Misol 2. 1-misolda ko'rsatilgan ma'lumotlarni oddiy jadval ko'rinishidagi tizilmaga solingandan so'ng u 2-rasmda tasvirlangan ko'rinishga ega bo'ladi.

Reyting daftarcha nomeri	Familiya	Ismi	Otasining ismi	Tug'ilgan sana	O'rtacha baho
654311	Avazov	Jamol	Aliyevich	15/01/1979	4,78
545712	Ortiqov	Akram	Salimovich	03/11/1978	4,61
453225	Lazizova	Saida	Xo'jayevna	07/07/1980	4,52
685564	Safarov	Toshmurod	Karimovich	12/04/1981	4,03
654786	Javlonov	Alisher	Ozodovich	31/12/1982	3,69

2-rasm. Tizilmalashtirilgan ma'lumotlarga misol.

Ma'lumotlar bazasidan foydalanuvchilar turli amaliy dasturlar, dasturiy vositalari, predmet sohasidagi mutaxassislar bo'lishi mumkin.

Ma'lumotlar bazasining zamonaviy texnologiyasida ma'lumotlar bazasini yaratish, uni dolzarb holatda yuritishni va foydalanuvchilarga undan axborot olishini ta'minlovchi maxsus dasturiy vosita, ya'ni ma'lumotlar bazasini boshqarish tizimi yordami bilan markazlashtirilgan holda amalga oshirishni nazarda tutadi.

Ma'lumotlar bazasi — EHM xotirasiga yozilgan ma'lum bir tuzilmaga ega, o'zaro bog'langan va tartiblangan ma'lumotlar majmuasi bo'lib, u biror-bir obyektning xususiyatini, holatini yoki obyektlar o'rtasidagi munosabatni ma'lum ma'noda ifodalaydi. MB foydalanuvchiga tuzilmalashtirilgan ma'lumotlarni saqlash va ishlatishda optimal qulaylikni yaratib beradi.

Ma'lumki ma'lumotlarni kiritish va ularni qayta ishlash jarayoni katta hajmdagi ish bo'lib, ko'p mehnat va vaqt talab qiladi. MB bilan ishlashda undagi ma'lumotlarning aniq bir tuzilmaga ega bo'lishi, birinchidan foydalanuvchiga ma'lumotlarni kiritish va qayta ishlash jarayonida undagi ma'lumotlarni tartiblashtirish, ikkinchidan kerakli ma'lumotlarni izlash va tez ajratib olish kabi qulayliklarni tug'diradi. MB tushunchasi fanga kirib kelgunga qadar, ma'lumotlardan turli ko'rinishlarda foydalanish juda qiyin edi. Bugungi kunda turli ko'rinishdagi ma'lumotlardan zamonaviy kompyuterlarda birgalikda foydalanish va ularni qayta ishlash masalasi hal qilindi. Kompyuterlarda saqlanadigan MB maxsus formatga ega bo'lgan muayyan tuzilmali fayl bo'lib, undagi ma'lumotlar o'zaro bog'langan va tartiblangan.

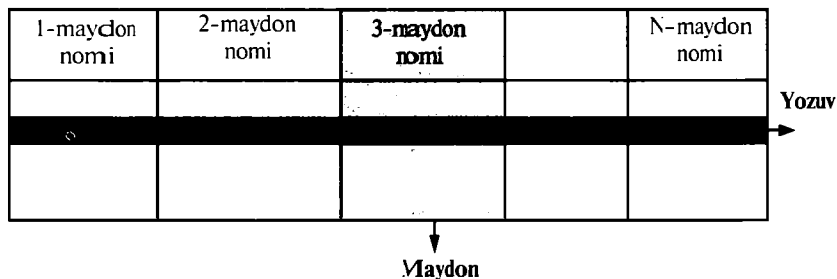
Demak, ma'lumotlar bazasi deganda ma'lum bir tuzilmada saqlanadigan ma'lumotlar to'plami tushuniladi. Boshqacha qilib aytganda, MB – bu ma'lum berilgan aniq bir tuzilmaga ega bo'lgan ma'lumotlarni o'z ichiga oluvchi maxsus formatga ega bo'lgan fayldir. Ma'lumotlarni tuzilmalashtirish – bu shunchaki ma'lumotlarni tasvirlashda qandaydir moslikni kiritish usulidir. Odatda, MB ma'lum bir obyekt sohasini ifodalaydi va uning ma'lumotlarini o'z ichiga oladi, ularni saqlaydi va foydalanuvchiga ma'lumotlarini qayta ishlashda undan foydalanish imkonini yaratib beradi.

**Ma'lumotlar bazasi** – bu ma'lum bir predmet sohasiga oid tuzimlashtirilgan (tuzilmalashtirilgan) ma'lumotlarning nomlangan to'plamidir.

**Ma'lumotlar bazasi** tushunchasi maydon, yozuv, fayl (jadval) kabi elementlar bilan chambarchas bog'liq.

**Maydon** – bu ma'lumotlarni mantiqiy tashkil etishning elementar birligi bo'lib, u axborotni eng kichik va bo'linmas birligi bo'lgan rekvizitga mos keladi. Maydonni tasvirlash uchun quyidagi tavsiflardan foydalaniladi:

*Maydon nomi*, masalan, familiyasi, ismi, tug'ilgan sana, lavozimi, ish staji, mutaxassisligi.



3-rasm. Ma'lumotlar bazasi tuzilmasining asosiy elementlari.

*Maydon turi*, masalan, son (числовой), simvol (символьный), sana/vaqt (data/время), mantiqiy (логический).

*Maydon uzunligi (o'lchami)*, masalan, eng ko'p simvollar sig'imi;

*Maydon aniqligi*, (son tipidagi ma'lumotlar uchun) masalan, sonning o'nlik kasr qismini aks ettirish uchun o'nlik raqamdan to'rtta.

**Yozuv** – bu mantiqiy bog'langan maydonlar to'plami. Yozuv tuzilishi uchun uning tarkibiga kiruvchi maydonlar tarkibi va joylashishi



ketma-ketligi bilan aniqlanib, ularni har biri ichida elementar **yozuvlarning nusxasi** deb ataladi. Yozuv obyektning biror-bir elementi haqida to'liq ma'lumotni ifodalaydi.

**Fayl (jadval)** – bu bir xil tuzilmaga ega bo'lgan yozuvning nusxalar to'plamidir. U o'zicha har bir maydonda qiymatga ega.

Misol 3. STUDENT faylidagi (jadvalidagi) yozuvlarning mantiqiy tuzilmasini tavsiflashga doir misol 4-rasmda ko'rsatilgan. Bu faylda 2-rasmda ko'rsatilgan ma'lumotlar tuzilmasi yechilgan. STUDENT faylidagi yozuvning tuzilishi chiziqli bo'lib, u o'zgarmas uzunlikdagi yozuvlardan iborat. Yozuv maydonlari takrorlanuvchi qiymatlar guruhiga ega emas. Maydon qiymatiga murojaat uning nomeri bo'yicha amalga oshiriladi.

Fayl nomi <i>STUDENT</i>					
<i>Maydon</i>		<i>Kalit belgisi</i>	<i>Maydon formati</i>		
<i>Nomni belgilash</i>	<i>To'liq nomlanish (rekvizit)</i>		<i>Tip</i>	<i>Uzunligi</i>	<i>Aniqligi</i>
Nomer	Talaba reyting daftarchasi nomeri	*	simvol	10	
Famil	Talaba familiyasi		simvol	10	
Ismi	Talaba ismi		simvol	8	
Ota.ismi	Talaba otasi ismi		simvol	10	
T_kun	Talabaning tug'ilgan sanasi		sana	8	
O'rt_a_baho	Talabaning o'rtacha bahosi		son	3	2

4-rasm. STUDENT faylidagi yozuvlarning mantiqiy tuzilishining tavsifi.

Har bir MB jadvali o'zining birlamchi kalitiga ega bo'lishi mumkin. **Birlamchi kalit** deganda yozuvlar qaytarilmasligini ta'minlovchi maydon (поля) yoki maydonlar guruhi tushuniladi. **Birlamchi kalit** sifatida ishlatiladigan maydon yoki maydonlar guruhi, bir xil yozuvga ega bo'lmaslik shartini bajarishi kerak. Masalan, yuqorida keltirilgan 2-rasmdagi jadvaldagi maydonlardan reyting daftarchasi nomeri birlamchi kalit bo'ladi. Boshqa maydonlarida bir xil yozuvlar takrorlanishi mumkin. Shu sabab ular birlamchi kalit bo'la olmaydi. Birlamchi kalit qisqa va sonli maydonlardan tashkil topishi maqsadga muvofiqdir. MB jadvaliga birlamchi kalitni kiritishdan maqsad, jadvaldagi ma'lumotlarni izlash, tartiblashtirish va tanlab olishda

qulaylikni beradi. Birlamchi kalitni kiritish yoki kiritmaslik foydalanuvchi tomonidan MB jadvali tuzilmasini tashkil qilishda aniqlanadi.

Bosh jadval yordamida qaram jadvaldagi mos ma'lumotlarni chaqirishni ta'minlash uchun qaram jadvalda tashqi kalit tashkil qilinadi. «Bitta-ko'pga» bog'lanish holatida tashqi kalit bosh jadvalda tashkil qilinadi. Birinchi va ikkinchi kalitlarni aniqlashda MBBT avtomatik ravishda jadvalda indekslarni quradi.

Indekslar kiritish mexanizmi MB jadvalidagi ma'lumotlarni tez topish va ajratib olish kabi imkoniyatlarni beradi.

Misol uchun, agar quyidagi jadval berilgan bo'lsa:

Yozuv №	Tovarning tushish vaqti	Tovar nomlari	Soni
1	10.01.2005	Shakar	10
2	12.01.2005	Kartoshka	50
3	12.01.2005	Sabzi	20
4	14.01.2005	Shakar	50
5	14.01.2005	Sabzi	10
6	16.01.2005	Olxo'ri	4

Mantiqiy nuqtai nazardan uning indeksleri quyidagicha bo'ladi.

Tovarning tushish vaqti		Tovar nomlari		Soni	
Sana	Yozuv №	Tovar	Yozuv №	Soni	Yozuv №
10.01.2005	1	Shakar	1	10	1
12.01.2005	2	Kartoshka	2	50	2
12.01.2005	3	Sabzi	3	20	3
14.01.2005	4	Shakar	4	50	4
14.01.2005	5	Sabzi	5	10	5
16.01.2005	6	Olxo'ri	6	4	6

Agar barcha yozuvlar ichidan faqat «shakar» nomli tovarni olishimiz kerak bo'lsa, jadvaldagi hamma ma'lumotlarni qarab ko'rish shart emas. U holda yozuv ko'rsatgichi to'g'ridan-to'g'ri «shakar» turgan birinchi yozuvga keladi va takrorlanadi. Agar jadvaldagi hamma «soni»>16 shartni qanoatlantiruvchi yozuvlarni chiqarish yoki ular ustida biror-bir operatsiya bajarish kerak bo'lsa, u holda yozuv

ko'rsatgich qiymatining «soni»>16 shartni qanoatlantiruvchi birinchi turgan yozuvga teng bo'ladi.

Bundan xulosa qilib aytganda jadvallarda «indeksatsiya» kiritish jadval ma'lumotlarini izlash, bir nechasini tanlab olish va ajratish kabi ishlarda qulaylikni tug'diradi. Bu esa, o'z navbatida jadval ustida har xil operatsiyalar olib borish foydalanuvchiga osonlashadi. Haqiqatan indekslashni tashkil etish ancha qiyin bo'lib, u MBni loyihalash jarayonida, uning tuzilmasini tashkil etish vaqtida aniqlanadi.

MBdan foydalanishda, ya'ni, undagi ma'lumotlar ustida har xil ishlar bajarishda bir necha usullar mavjud. Bu usullarga baza ma'lumotlaridan foydalanishga murojaat (доступ) usullari deyiladi. Quyidagi baza ma'lumotlaridan foydalanish usullarini ko'rib chiqamiz.

1. Ketma-ket murojaat usuli. Ketma-ket murojaat usuli MB jadvaliga so'rovlarning bajarilishi uchun jadvaldagi hamma yozuvlarni qarab chiqadi. Masalan, bu usul yordamida birinchi yozuvdan oxirigi yozuvgacha kerakli yozuvlarni berilgan so'rov bo'yicha qarab chiqib, ularni ajratib chiqarish mumkin. Bu usulning uncha effektiv emasligi faqat uning tez izlab topishda vaqtning yoki hisoblash resurslarining ortiqcha ishlatilishidir.

2. Indeksli ketma-ket murojaat usuli. Indeksli ketma-ket murojaat usuli MB jadvaliga berilgan so'rov bajarilishi uchun yozuv ko'rsatgichini so'rov shartini bajaruvchi yozuvning birinchisiga olib kelib qo'yadi. Keyin yozuv ko'rsatgichi so'rov shartini qanoatlantiruvchi keyingi qatorga o'tadi. Shunday qilib, so'rov shartini qanoatlantiruvchi hamma qator o'qiladi. Demak, indeksli ketma-ket murojaat usulida jadvalda faqat so'rov shartini qanoatlantiruvchi yozuvlar o'qiladi.

3. To'g'ridan-to'g'ri murojaat usuli. Indeksli ketma-ket murojaat usulida jadvaldan bir yoki bir necha maydonlar guruhi qiymati bo'yicha to'g'ridan-to'g'ri kerakli yozuvlar olinadi. Bu usul ikkinchi usulning xususiy holi deb qaraladi. Chunki indeksli ketma-ket murojaat usuli to'g'ridan-to'g'ri usulni ishlatadi. Yozuv ko'rsatgichi so'rov sharti qanoatlantiradigan indeksning birinchi qatoriga qo'yiladi va kerakli shartni qanoatlantiruvchi indeksli yozuvlarni o'qib bo'lgandan so'ng ketma-ket murojaat usuli qator indeksi bo'yicha siljiydi.

Paradox va Dbase MBBSda indekslar boshqa faylda saqlanadi. Oracle, SyBase, InterBase va SqrServer MBBSda esa indekslar birgalikda baza faylining o'zida saqlanadi.

Ma'lumotlarga ishlov berish texnologiyasi bo'yicha ma'lumotlar bazasi markazlashtirilgan va taqsimlangan bazalarga bo'linadi.

Markazlashtirilgan ma'lumotlar bazasi hisoblash tizimining xotirasida saqlanadi. Agar bu hisoblash tizimi kompyuterlar tarmog'ining komponenti bo'lsa, u holda bunday bazaga tarmoq orqali kirish uchun ruxsat berilishi mumkin. Ma'lumotlar bazasidan bunday foydalanish usuli kompyuterlarning lokal tarmoqlarida ko'p uchraydi.

Taqsimlangan ma'lumotlar bazasi hisoblash tarmog'ining turli kompyuterlarida saqlanuvchi bir necha qismlardan iborat bo'lib, ular kesishuvi, hatto bir-birini takrorlashi mumkin. Bunday baza bilan ishlash ma'lumotlarni taqsimlangan bazasini boshqarish tizimining yordami bilan amalga oshiriladi.

Ma'lumotlar bazasidagi ma'lumotlardan foydalanish huquqi bo'yicha ular lokal kirish huquqiga ega bo'lgan ma'lumotlar bazasi va uzoq masofadan (tarmoqdan) kirish huquqiga ega bo'lgan ma'lumotlar bazasiga bo'linadi.

Tarmoqdan kirish huquqiga ega bo'lgan ma'lumotlar bazasini markazlashtirilgan tizimi bu kabi tizimlarning turli arxitekturasini nazarda tutadi:

- Fayl-server;
- Kliyent-server.

**Fayl-server.** Ushbu konsepsiyada tarmoqdan kirish huquqiga ega bo'lgan ma'lumotlar bazasi tizimining arxitekturasi markaziy EHM (fayllar serveri) sifatida tarmoq kompyuterlaridan birini ajratib ko'rsatishini nazarda tutadi. Bunday kompyuterda umumiy foydalanishga mo'ljallangan markaziy ma'lumotlar bazasi saqlanadi. Tarmoqdagi boshqa hamma kompyuterlar ishchi stansiyalari funksiyasini bajaradi. Ularning yordami bilan foydalanuvchi tizimdan markaziy ma'lumotlar bazasiga kirishi ta'minlanadi. Ma'lumotlar bazasi fayllari foydalanuvchi so'rovlariga mos ravishda ish stansiyalariga yuboriladi. Ma'lumotlarni qayta ishlash asosan ish stansiyalarida amalga oshiriladi. Ma'lumotlar bazasiga kirish intensivligi katta bo'lganda axborot tizimining unumdorligi pasayadi. Foydalanuvchilar ish stansiyalarida lokal ma'lumotlar bazasi yaratishlari va ulardan yakka tartibda foydalanishlari ham mumkin.

**Kliyent-server.** Bu konsepsiyada, markaziy ma'lumotlar bazasi maxsus kompyuterda (Ma'lumotlar bazasi serverida) saqlanishi bilan birga, ma'lumotlarni qayta ishlash masalalarining asosiy qismi bajarilishini ta'minlashi zarur.

Kliyent (ish stansiyasi) tomonidan ma'lumotlarga berilgan so'rov serverdagi ma'lumotlarni qidirish va topishga olib keladi. Olingan ma'lumotlar (ammo fayllar emas) tarmoq bo'yicha serverdan kliyentga

uzatiladi. Kliyent-server arxitekturasi o'ziga xos xususiyatlaridan biri SQL so'rovlar tilidan foydalanish hisoblanadi.

Ma'lumotlar bazasi – axborot tizimlarining eng asosiy tarkibiy qismi bo'lib hisoblanadi. Ma'lumotlar bazasidan foydalanish uchun foydalanuvchi ishini yengillashtirish maqsadida ma'lumotlar bazasini boshqarish tizimlari yaratilgan. Bu tizimlar ma'lumotlar bazasini amaliy dasturlardan ajratadi.

Ma'lumotlar bazasini boshqarish tizimi (MBBT) – bu dasturiy va apparat vositalarining murakkab majmuasi bo'lib, ular yordamida foydalanuvchi ma'lumotlar bazasini yaratishi va shu bazadagi ma'lumotlar ustida ish yuritishi mumkin.

Juda ko'p turdagi MBBT mavjud. Ular o'z maxsus dasturlash tillariga ham ega bo'lib, bu tillarga MBBT ning buyruqli dasturlash tillari deyiladi. MBBTga Oracle, Clipper, Paradox, FoxPro, Access va boshqalarni misol keltirish mumkin.

***Ma'lumotlar bazasini boshqarish tizimi – bu ma'lumotlar bazasini yaratish, ularni dolzarb holatini ta'minlash va undagi zarur axborotni topish ishlarini tashkil etish uchun mo'ljallangan dasturlar majmui va til vositasidir.***

## 8.2. Ma'lumotlar modellari

Mashina muhitida ma'lumotlarni tashkil etish ikki pog'onadan iborat bo'lib, **mantiqiy va fizik** pog'onalar bilan xarakterlanadi. Ma'lumotlarni bevosita fizik tashkil etishda ularni mashina «tashuvchisi»da joylashtirish usuli aniqlab olinadi. Zamonaviy amaliy dasturlar vositalarida ma'lumotlarni tashkil etishning bu pog'onasi avtomatik ravishda foydalanuvchi aralashuviz ta'minlanadi. Odatda, foydalanuvchi amaliy dasturlar vositalarining ma'lumotlarini mantiqiy tashkil etish haqidagi tushunchalar bilan operatsiyalar bajaradi. Mashina «tashuvchisi»da ma'lumotlarni mantiqiy tashkil etish, foydalanilayotgan dasturlash vositalaridan va mashina muhitidagi ma'lumotlar yuritishga bog'liq. Ma'lumotlarni tashkil etishning mantiqiy usulidan **foydalanilayotgan ma'lumotlar tuzilishining turi** va dasturiy vositalar orqali qo'llaniladigan **modelning shakli** aniqlanadi.

**Ma'lumotlarning modeli** – bu ma'lumotlar o'zaro bog'langan tuzilishlari va ular ustida bajariladigan operatsiyalar to'plamidir. Modelning shakli va undan foydalaniladigan ma'lumotlar tuzilishining turi dasturlash tizimi tilida foydalangan ma'lumotlarni tashkil etish va ishlov berish konsepsiyasini aks ettiradi.

Ma'lumki, aynan bir axborotni mashina ichki muhitida joylashtirish uchun ma'lumotlarning turli xil tuzilishlari va modellaridan foydalanish mumkin. Ulardan qaysi birini tanlash axborotlar bazasini yaratayotgan foydalanuvchining zimmasiga yuklatilgan bo'lib, u ko'plab omillarga bog'liq. Bu omillar qatoriga mavjud texnik va dasturiy ta'minotlar, hamda avtomatlashtirilayotgan masalalarning murakkabligi va axborotning hajmi kabilar kiradi.

Ma'lumotlar modeli quyidagi tarkibiy qismdan iborat:

1. Foydalanuvchining ma'lumotlar bazasiga munosabatini namoyish etishga mo'ljallangan ma'lumotlar tuzilmasi.

2. Ma'lumotlar tuzilishida bajarilish mumkin bo'lgan operatsiyalar. Ular ko'rib chiqilayotgan ma'lumotlar modeli uchun ma'lumotlar tilining asosini tashkil etadi. Yaxshi ma'lumotlar tuzilmasining o'zigina yetarli emas. Ma'lumotlarni aniqlash tili (MAT) va ma'lumotlar bilan amallar bajarish tilining (MABT) turli operatsiyalari yordamida bu tuzilma bilan ishlash imkoniga ega bo'lish zarur.

3. Yaxlitlikni nazorat qilish uchun cheklashlar. Ma'lumotlar modeli uning yaxlitligini saqlash va himoya qilishga imkon beruvchi vositalar bilan ta'minlangan bo'lishi lozim.

**Ma'lumotlarning iyerarxik va tarmoq modellari.** Mashina muhitidagi ma'lumotlarning murakkabroq modellari – *tarmoqli va iyerarxik* modellar bo'lib hisoblanadi. Bu modellar ularning o'zlariga xos turdagi ma'lumotlar bazasini boshqarish tizimida ishlatiladi. MBBTda ma'lumotlarni mantiqiy tashkil etish usuli ma'lumotlarning tarmoqli yoki iyerarxik modeliga mos holda ko'rsatiladi. Bunday model o'zaro bog'liq obyektlarning majmuidir. Ikki obyektning aloqasi ularning bir-biriga tobeligini aks ettiradi. Tarmoqli yoki iyerarxik modelida obyekt bo'lib, MBBT kiritilgan ma'lumotlar tuzilmasining asosiy turlari hisoblanadi. Turli MBBTlarda bu turdagi ma'lumotlarning tuzilmasi turlicha aniqlanishi va nomlanishi mumkin.

**Modellarda ma'lumotlarning tuzilmalari.** Ma'lumotlarning namunaviy tuzilmalariga quyidagilar kiradi: ma'lumotlarning elementi, ma'lumotlarning agregati, yozuv, ma'lumotlar bazasi va hokazo. Bu elementlar va agregatlar o'zaro aloqada bo'lgan tuzilma bilan tavsiflanadi. Shuning uchun yozuvning tuzilmasi iyerarxik xarakterga ega bo'lishi mumkin. Bir xil tuzilmaga ega bo'lgan yozuv nusxalari to'plamining hammasi yozuv turini tashkil etadi.

**Ma'lumotlarning elementi** – bu ma'lumotlar tuzilmasining nomlangan minimal birligi (faylli tizimlardagi maydonning o'xshashi).

**Ma'lumotlar agregati** – bu ma'lumotlar elementlarning quyi

to'plami yoki yozuvlar ichidagi boshqa agregatlarning nomlangan quyi to'plami.

**Yozuv** – umumiy holda agregat bo'lib, u boshqa agregatlarning tarkibiga kirmaydigan tarkibli agregatdan iborat.

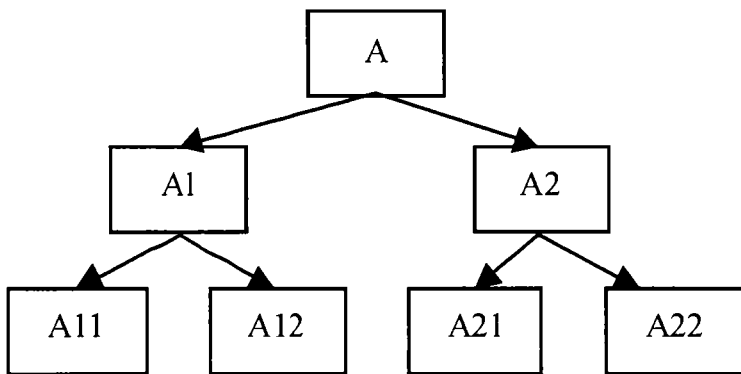
**Obyektlarning modellardagi aloqasi.** Ma'lumotlar modeli bir necha turdagi yozuvlarni (obyektlarni) o'z ichiga olishi mumkin. Ma'lumot modeli obyektlar o'rtasida aloqalar o'rnatadi. Qandaydir bir predmet sohasi uchun modelning o'zaro bog'langan muayyan obyektlar to'plami ma'lumotlar bazasini tashkil qiladi.

Ikki turdagi yozuvlarning (model obyektlari) o'rtasidagi aloqalar, ularning nusxalari o'rtasidagi guruh munosabatlari bilan aniqlanadi. Guruh munosabati – bu ikki turdagi yozuvlar o'rtasidagi qat'iy iyerarxik munosabat bo'lib, ular asosiy yozuvlar to'plami va tobe yozuvlar to'plamidan iboratdir.

Iyerarxik modellarda kalit bo'yicha bevosita kirish odatda, faqat boshqa obyektlarga tobe bo'lmagan eng yuqori pog'onadagi obyektgagina mumkin. Boshqa obyektlarga kirish modelning eng yuqori pog'onasidagi obyektidan aloqalar bo'yicha amalga oshiriladi. Tarmoqli modellarda esa, kalit bo'yicha bevosita ixtiyoriy obyektga kirish (uning modelda joylashgan pog'onasidan qat'i nazar) ta'minlanishi mumkin. Shuningdek, aloqalar bo'yicha har qanday nuqtadan kirish ham mumkin. Tarmoqli modellarda obyekt (yozuv, fayl)ning tuzilmasi ko'pincha chiziqli va kamroq hollarda esa iyerarxik bo'ladi. Quyi pog'onadagi ma'lumotlarning tuzilmasi ham o'z xususiyatiga va nomiga ega bo'lishi mumkin. Masalan, atribut bu ma'lumotlar elementining analogi. Chiziqli tuzilmaga ega bo'lgan obyekt faqat oddiy va kalitli atributlardan iborat. Iyerarxik modellardagi obyekt (yozuv, seoment) tuzilmasi iyerarxik yoki chiziqli bo'lishi mumkin.

Turli predmet sohalari uchun ma'lumotlarning tarmoqli modeli iyerarxik modeliga nisbatan mashinaning ish muhitida axborot tuzilmalarini aks ettiruvchi umumiy vosita hisoblanadi. Ko'plab predmet sohaslarining ma'lumotlari o'rtasidagi aloqalar tarmoqli ko'rinishga ega. Bu esa, ma'lumotlarning iyerarxik modeliga ega bo'lgan MBBTdan foydalanishni cheklab qo'yadi. Tarmoqli modellar ma'lumotlarning iyerarxik aloqasini ham aks ettirishga imkon beradi. Bundan tashqari, tarmoqli modellar bilan ishlash texnologiyasi foydalanuvchi uchun qulaydir, chunki ma'lumotlarga kirishni amalga oshirishda hech qanday cheklashlar yo'q va bevosita ixtiyoriy pog'onadagi obyektlarga kirish imkoni mavjud.

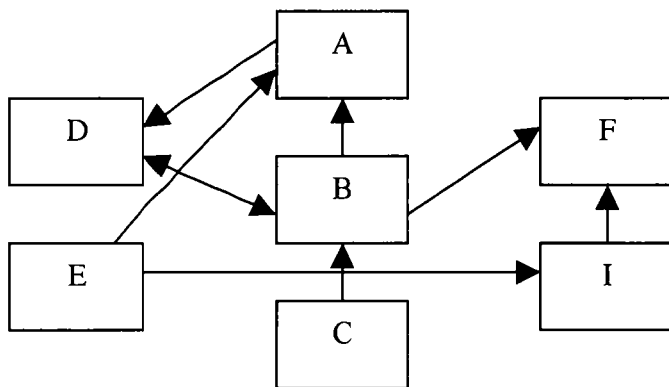
Iyerarxik ma'lumotlar bazasida – ma'lumotlar iyerarxiya (daraxt) ko'rinishida saqlanadi. Uning ko'rinishini quyidagicha tasvirlash mumkin:



Ma'lumotlarning iyerarxik modeli.

Masalan, bu yerda A12 tugunidagi ma'lumotni olish uchun, oldin MBdan A tugun, keyin A1 tugun va undan keyin A12 topiladi.

Tarmoq ma'lumotlar bazasi – ichki ma'lumotlar tuzilmasi, biri ikkinchisiga bog'liq ravishda bo'ladi. Uning ko'rinishini quyidagicha tasavvur qilish mumkin.



Ma'lumotlarning tarmoq modeli.



Iyerarxik va tarmoq modellarida ma'lumotlar tasvirining murakkabligi va bu ma'lumotlar orasidagi aloqani MBni loyihalashda aniqlash kerak bo'lib, bu esa, MBga so'rov berilganda relatsion MB jadvallari orasida aloqa o'rnatishni ta'minlab beradi.

**Ma'lumotlarning relatsion modeli.** Relatsion MB kuchli nazariy fundamentga ega bo'lib, u matematik munosabatlar (отношения) nazariyasiga asoslangan. Ma'lumotlarning relatsion modeli konsepsiyasi 1970-yilda Y.F.Kodd tomonidan taklif qilingan bo'lib, u ma'lumotlarni tavsiflash va tasvirlashning amaliy dasturlariga bog'liq bo'lmasligini ta'minlash masalasini hal qilish uchun xizmat qiladi.

Ma'lumotlarning relatsion modeli asosida «munosabat» tushunchasi yotib, u inglizcha relation so'zidan olingan. Ba'zi bir qoidalarga amal qilgan holda munosabatlarni ikki o'lchovli jadval ko'rinishida tasvirlash mumkin. Jadval har qanday odamga tushunarli va qulaydir.

Real dunyo obyektlari haqidagi ma'lumotlarni EHM xotirasida saqlash va ular orasidagi munosabatlarni modellashtirish uchun munosabatlar (jadval) to'plamidan foydalanish mumkinligini Y.F.Kodd isbotlab berdi. Masalan, «talaba» mazmunini saqlash uchun TALABA munosabatidan foydalaniladi. Bu mazmunning asosiy xususiyatlarini quyidagi jadvalning ustunlari tasvirlaydi:

Munosabat ustunlari atributlar deb ataladi va ularga nomlar beriladi. Munosabat atributlarining nomlaridan iborat ro'yxatni munosabatlar sxemasi deyiladi. Bizning misolimizdagi TALABA munosabatining sxemasi quyidagicha yoziladi: **TALABA (Familiyasi I.O., Tug'ilgan sana, Bosqich, Mutaxassisligi).**

**TALABA**

<b>Familiyasi I.O.</b>	<b>Tug'ilgan sana</b>	<b>Bosqich</b>	<b>Mutaxassisligi</b>
Karimov M.N.	15/01/1979	4	menejement
Avazov A.V	03/11/1978	4	Buxgalteriya
Ortiqov O.B.	07/07/1980	3	Buxgalteriya
Lazizova V.N.	12/04/1981	3	Muhandis pedagog
Safarov O.X	09/05/1980	3	Menejement
Xoliyorov N.A.	31/12/1982	2	Marketing
Javlonov X.B.	24/09/1983	1	Sug'urta ishi

**Ma'lumotlarning relatsion bazasi** – bu o'zaro bog'langan munosabatlar, ya'ni jadvallar to'plamidir. Har qanday munosabat (jadval) kompyuterlarning xotirasida fayl ko'rinishida joylashtiriladi. Ularning orasida quyidagi moslik mavjud:

Fayl	Jadval	Munosabat	Mazmuni
Yozuv	Satr	kortej	mazmunning nusxasi
Maydon	Ustun	atribut	atribut

Jadval hamma uchun juda qulay bo'lishi bilan bir qatorda ma'lumotlarni manipulyatsiya qilishning asosiy uch operatsiyasini bajarish uchun noqulaydir, ya'ni tartiblash, indekslarning qiymatlari bo'yicha guruhlash va daraxt ko'rinishidagi parametrlar bilan ishlash.

Jadvalda ushbu uch operatsiya bir-biri bilan chambarchas bog'langan. Bu esa, ba'zi bir operatsiyalarni bajarishda ma'lum bir qiyinchiliklarga olib keladi. Masalan, ma'lumotlarni bir parametr asosida tartiblash ikkinchi bir parametr bo'yicha tartiblashni buzib yuborishi tufayli zarur ma'lumotlarni izlab topish operatsiyasi bir parametr bo'yicha osonlashsa, boshqalari bo'yicha qiyinlashadi.

Kodd taklif qilgan usulining originalligi shundan iboratki, u munosabatlarga (jadvallarga) tadbiiq qilish uchun juda chiroyli qurilgan operatsiyalar tizimini ishlab chiqdi. Ularni amalga oshirish natijasida bir munosabatni boshqa munosabat orqali hisoblab chiqish imkoniyati paydo bo'ldi. Bu axborotlarni saqlanadigan va saqlanmaydigan (hisoblanadigan) qismlarga ajratish, hamda kompyuter xotirasini tejash zarur bo'lgan paytda axborotlarning saqlanmaydigan qismini saqlanadiganlar asosida hisoblab chiqish imkoniyatini beradi.

Ma'lumotlarning relatsion bazasidagi munosabatlar ustida bajariladigan asosiy operatsiyalar sakkizta bo'lib, ular quyidagilardan iborat:

– to'plamlar ustidagi an'anaviy (традицион) operatsiyalar, ya'ni to'plamlarning birlashmasi (yig'indisi), kesishmasi (ko'paytmasi), to'ldiruvchisi (ayirmasi), dekart ko'paytmasi, bo'lishmasi;

– maxsus relatsion operatsiyalar, ya'ni proyeksiyalash, bog'lanish (qo'shilish), birlashtirish (ulab qo'yish) va tanlash.

Har bir ma'lumotlar bazasini boshqarish tizimining samaradorligi ushbu operatsiyalarning borligi va ularni bajarish vositalarining qanchalik qulayligi bilan aniqlanadi. Relatsion MBBTda munosabatlar ustida operatsiyalar bajarish uchun mo'ljallangan tillarni ikki sinfga

ajratish mumkin: relatsion algebra tili (RAT) va relatsion hisob tili (RHT).

RAT relatsion algebraga (Kodd algebrasiga,  $\alpha$ -algebraga) asoslangan. Ma'lum tartib munosabatlar ustida operatsiyalarni ketma-ket yozish asosida xohlagan natijaga erishish mumkin. Shuning uchun RATni protsedurali til deyishadi.

RHT predikatlarini hisoblab chiqishning klassik usuliga asoslangan. Ular foydalanuvchilarga so'rovlarni yozish uchun ma'lum qoidalar to'plamini beradi. Bunday so'rovlarda faqat xohlagan natija haqidagi axborotlar bo'ladi, xolos. Ushbu so'rov asosida MBBT yangi munosabatlar hosil qilish yo'li bilan avtomatik tarzda zarur natijani beradi. Shuning uchun RHTni protseduralimas til deyishadi.

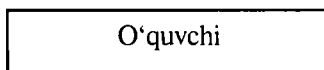
Ma'lumotlar bazasini loyihalashtirishda relatsion model bilan ishlash ancha noqulayliklarga olib keladi. Shu sabab ma'lumotlar bazasini loyihalashda har xil semantik modellar ham ishlatiladi. Ulardan eng ko'p tarqalganlaridan biriga – ER modeli deyiladi. Bu model inglizcha «**Entity–relation**» deyilib, ma'nosi «**Mohiyat–bog'lanish**» demakdir.

Bu model 1976-yil Piter Chen tomonidan kiritilgan bo'lib, u o'ziga bir qator grafik diagrammalarni oluvchi bir necha har xil turdagi komponentalarni birlashtirgan. Piter Chen mohiyatlar to'plami va ular orasida bog'lanish sifatida relatsion ma'lumotlar tuzilmasini interpretatsiya qilishni taklif qildi.

ER –modelining asosiy komponentalari mohiyat, bog'lanish va atribut (xossa) bo'lib hisoblanadi.

**Mohiyat** – bu ma'lumotlar bazasida saqlanishi kerak bo'lgan biror real yoki tasavvur qilingan obyektidir. ER modeli diagrammasida mohiyat odatda to'rtburchak shaklida tasvirlanib, uning ichiga mohiyat nomi qo'yiladi.

Masalan:



Mohiyat aniq ma'noga ega bo'lgan nomga ega bo'lib, u yagona bo'lishi kerak. Mohiyat turini uning nusxasi bilan farq qilish kerak. Mohiyat nomi uning nusxasiga emas, turiga beriladi. Mohiyat nusxasi – bu aniq bir xil turdagi narsalar, hodisalar va boshqalardir.

Masalan, yuqoridagi «O'quvchi» mohiyatida «O'quvchi» mohiyat turining nomi, mohiyat nusxasi esa aniq bir o'quvchidir. Masalan, Ahmedov, Toshmatov va boshqa.

**Bog‘lanish** – bu ikki yoki bir necha mohiyatlar birikmasidir. Bog‘lanish faqat ikkita har xil mohiyatlar orasida mavjud bo‘ladi. Oxirgi bog‘lanishga rekursiv deyiladi.

«Mohiyat–bog‘lanish» diagrammalarini ishlab chiquvchi har xil turdagi standart metodologiyalar mavjud. Masalan, IDEFIX, IE, DM. Bu usullar har qaysisining mohiyat–bog‘lanishni tasvirlashi uchun o‘z belgilari bor.

**Atribut (xossa)** – mohiyatni xarakterlovchi nomlardir. U o‘zida yagona murakab bo‘lmagan tuzilmani tasvirlab, mohiyat holatini xarakterlaydi. Masalan, «O‘quvchi» mohiyatining atributi – kod, familiya, ism, manzil, yosh va boshqalardir.

Mohiyat atributlar to‘plami cheksizdir. U axborot tizimlari bilan ishlaydigan foydalanuvchi talabiga va yechiladigan masalaga bog‘liqdir.

### **Ma‘lumotlar bazasi jadvallari orasidagi relatsion bog‘lanish.**

MBning ikki va undan ortiq jadvallarining orasi biri ikkinchisiga bog‘liq bo‘lishi mumkin. Agar ikkinchi jadval birinchi jadvalga qaram bo‘lsa, birinchi jadvalga bosh jadval, ikkinchi jadvalga esa qaram jadval deyiladi. Bosh jadvaldagi bitta yozuvga qaram jadvalda unga mos bir necha yozuv mavjud bo‘lishi mumkin.

MB jadvallari orasida uchta har xil aloqa bo‘lishi mumkin: «bitta-ko‘pga»; «bitta-bittaga»; «ko‘p-ko‘pga».

«**Bitta-ko‘pga**» bog‘lanish. «Bitta-ko‘pga» bog‘lanish bo‘ladi, qachonki bosh jadvaldagi bitta yozuv qaram jadvaldagi bir necha yozuvga aloqasi bo‘lsa.

### **M i s o l :**

«Fakultetlar» jadvali

«Talabalar» jadvali

№	Fakultetlar nomi	Talabalar soni	Yuqori ball olgan talabalar	Fakul. nomeri	Ball
1	Mexanika	500	Botirov Sh.	1	100
2	Yengil sanoat	700	Rustamov F	1	95
3	Pedagogika	1000	Ahmedova T.	1	93
4	Axborot texnologiyalari	600	Mirsodiqov I	2	100
			Sobirov M	2	97
			Ismatullayev F	4	98
			Rahimova A	4	93

Relatsion MB uchun «bitta-ko'pga» bog'lanish holati eng ko'p ishlatiladi.

«**Bitta-bittaga**» bog'lanish. «Bitta-bittaga» bog'lanish bo'ladi, qachonki bosh jadvaldagi bitta yozuv qaram jadvaldagi faqat bitta yozuvga aloqasi bo'lsa.

**Misol:**

«O'qituvchilar haqida ma'lumotnoma» jadvali

№	F.I.O	Lavozimi	Kafedra	№	Tug'il. yili	Bolasi	...
1	Alimov S	Dotsent	Mexanika	1	1950	3	...
2	Ikromov R.	Katta o'qituvchi	Fizika	2	1952	1	...
3	Rustamov A	Assistent	Fizika	3	1960	2	...
4	.....	.....	.....	4	.....	.....	...

«**Birga-bir**» bog'lanish qattiq yoki yumshoq bo'lishi mumkin. Agar bosh jadvaldagi bitta yozuvga qaram jadvaldan hammavaqt faqat bitta yozuv to'g'ri kelsa qattiq bog'lanish bo'ladi. Agar bosh jadvaldagi bitta yozuvga qaram jadvalda bitta yozuv bo'lish yoki bo'lmaslik sharti bo'lsa, u holda bog'lanish yumshoq bo'ladi.

«**Ko'p-ko'pga**» bog'lanish. «Ko'p-ko'pga» bog'lanish quyidagi hollarda bo'lishi mumkin.

a) bosh jadvaldagi yozuvga qaram jadvalda bittadan ortiq yozuv to'g'ri kelsa.

b) qaram jadvaldagi yozuvga bosh jadvalda bittadan ortiq yozuv to'g'ri kelsa.

**Misol:**

«Guruhlar va predmetlar» jadvali

«O'qituvchilar» jadvali

Guruh	Fan nomi	O'qit. nomeri	O'qit. nomer	O'qit. I.F	Kafedra nomi
22-03	Dasturlash	10	10	Sobirov	AT
4-02	Axborot texnologiyalari	10	12	Karimov R	Ximiya
3p-03	Mexanika	13	62	Ikromov	Tarix
7-02	Falsafa	62	78	Naimov T	Fizika
18-03	Tarix	62	85	Zoirov S	EI
.....	.....	.....	.....	.....	.....

Bitta jadvaldagi yozuvlar ham bir-biri bilan aloqada bo'lishi mumkin. Quyidagi misolni qaraymiz. Relatsion MB quyidagi daraxt ko'rinishidagi tuzilmaga ega bo'lsin.

### **Avtomatlashtirish departamenti**

#### 1. Texnik boshqarmasi:

- Tarmoq bo'limi;
- Remont bo'limi;
- ATS.

#### 2. Sistemali programma boshqarmasi:

- Eksploatatsiya bo'limi:
  - Axborot tayyorlash guruhi;
  - Administrativ guruhi;
  - Dispatcher byurosi.
- Ishlab chiqarish bo'limi:

Buni jadval ko'rinishida quyidagicha tasavvur qilish mumkin.

Bo'limlar №	Bo'limlar nomlari	Bo'lim dar.
1	Avtomatlashtirish departamenti.	0
2	Texnik boshqarmasi:	1
3	Sistemali programma boshqarmasi:	1
4	Tarmoq bo'limi;	2
5	Remont bo'limi;	2
6	ATS.	2
7	Eksploatatsiya bo'limi:	3
8	Ishlab chiqarish bo'limi.	3
9	Axborot tayyorlash guruhi;	7
10	Administrativ guruhi;	7
11	Dispatcher byurosi.	10

Bunday ko'rinishdagi jadval ma'lumotlarini MBBT avtomatik ravishda boshqara olmaydi, uni dasturli boshqarishga to'g'ri keladi.

### **8.3. Ma'lumotlar bazasini loyihalashtirish**

Ma'lumotlar bazasini (MB) ishlab chiqish (loyihalash)ning asosiy maqsadi uning mantiqiy tuzilishini aniqlashdan iboratdir. MBni ishlab chiqish predmet sohasini tavsiflash asosida amalga oshiriladi. Bu tavsiflash MBga kiruvchi hamma ma'lumotlarni o'z ichiga oluvchi hujjatlar majmuini va predmet sohasini ifodalovchi obyekt va jarayonlar haqidagi boshqa ma'lumotlarni o'z ichiga oladi.

MBni yaratishni loyihalashdan boshlash lozim. Loyihalash natijasida relatsion bazaning tuzilishi, ya'ni relatsion jadvallar tarkibi, ularning tuzilishi va mantiqiy aloqadorligi aniqlanadi. Relatsion jadvalning tuzilishi esa uning ustunlari tarkibi, ularning ketma-ketligi, ustun ma'lumotlarining turi va o'lchami, shuningdek, jadval kaliti bilan aniqlanadi.

### **Predmet sohasini tahlil qilish**

Ixtiyoriy tipdagi MBni loyihalashtirishning birinchi bosqichi predmet sohasini aniqlash bo'lib, u axborot tuzilmasini (konseptual sxemalar) tuzish bilan yakunlanadi. Bu bosqichda foydalanuvchining so'rovlari tahlil qilinadi, axborot obyektlari va uning xarakteristikallari tanlanadi, hamda o'tkazilgan tahlil asosida predmet sohasi tuzilmashtiriladi. Predmet sohasini tahlil qilish umumiy bosqich bo'lib, MB ishlashini amalga oshiradigan dasturiy va texnik vositalarga bog'liq emas.

Predmet sohasini tahlil qilishni uch pog'onaga bo'lish maqsadga muvofiq:

- konseptual talablar va axborot ehtiyojlarini tahlil qilish;
- axborot obyektlari va ular orasidagi aloqalarni aniqlash;
- predmet sohasining konseptual modelini qurish va MBni konseptual sxemasini loyihalashtirish.

Konseptual talablar va axborot ehtiyojlarini tahlil qilishda quyidagi masalalarni hal qilish kerak:

- foydalanuvchilarning MBga bo'lgan talablarini tahlil qilish (konseptual talablar);
- MBdan o'rin olishi lozim bo'lgan axborotlarga ishlov berish bo'yicha mavjud masalalarni aniqlash (tadbiqni tahlil qilish);
- kelajakda hal qilinishi lozim bo'lgan masalalarni aniqlash (perespektiv tadbiqning tahlili).
- tahlil natijalarini hujjatlashtirish.

Ishlab chiqarilayotgan MBga foydalanuvchilarning talablari so'rovlar bilan ularning intensivligi ko'rsatilgan ro'yxat va ma'lumotlarning hajmidan iborat. MBni ishlab chiqaruvchilar bu ma'lumotlarni uning bo'lajak foydalanuvchilari bilan suhbat o'tkazish natijasida aniqlaydilar. Shu yerda axborotlarni kiritishga, yangilashga va o'zgartirishga bo'lgan talablar ham aniqlanadi. Mavjud va perespektiv tadbiqlarni tahlil qilish natijasida foydalanuvchilar talablari aniqlashtiriladi va to'ldiriladi.

Turli predmet sohaslarini tahlil qilishda so'rovnomaning taxminiy tarkibini quramiz:

**1-misol.** Institut talabalarini hisobga olish uchun MBni ishlab chiqish taklif qilinyapti.

Predmet sohasini tahlil qilish:

1. Institutda qancha talaba ta'lim oladi?
2. Institutda qancha fakultet va kafedralar bor?
3. Fakultet bo'limlari va kurslari bo'yicha talabalar qanday taqsimlangan?
4. Har bir kursda har bir mutaxassislik bo'yicha qancha fanlar o'qitiladi?

5. Institutda qancha o'qituvchi bor?

6. Boshqa shahardan kelgan qancha talaba yotoqxonada, xususiy xonadonlarda (ijarachi sifatida) yashaydi?

8. Ma'ruza va amaliy mashg'ulotlar o'tkazish uchun qancha auditoriyalar hamda qancha laboratoriyalar bor?

9. Va boshqalar.

Predmet sohasini tahlil qilishning ikkinchi pog'onasi axborot obyektlarini tanlash, har bir obyekt uchun zarur xossalarni berish, obyektlar orasidagi aloqalarni aniqlash, axborot obyektlariga qo'yiladigan cheklashlarni aniqlash, axborot obyektlari orasidagi aloqalarning turlarini va axborot obyektlarining tavsifnomalarini aniqlashdan iborat.

Masalan, axborot obyektlarini tanlayotganda quyidagi savollarga javob berish kerak bo'ladi:

1. MBda saqlanishi lozim bo'lgan ma'lumotlarni qanday sinflarga ajratish mumkin?

2. Har bir ma'lumotlar sinfiga qanday nom berish mumkin?

3. Har bir ma'lumotlar sinfi uchun qanday eng muhim tavsifnomalarni (foydalanuvchining nuqtai nazaridan) ajratish mumkin?

4. Tanlangan tavsifnomalar to'plamlariga qanday nomlarni berish mumkin?

Axborot obyektlarini aniqlash itaratsion jarayon. U axborotlar oqimining tahlili va iste'molchilar bilan suhbat o'tkazish asosida amalga oshiriladi. Axborot obyektlarining tavsifnomalari ham xuddi shu usullar bilan aniqlanadi.

Predmet sohasining axborot ehtiyojini va tizimdan foydalanuvchilarning axborot bilan ishlashdagi manfaatini hisobga olgan holda predmet sohasini tuzilmashtirish uchun konseptual model qo'llaniladi.



Har bir MB miqyosida konseptual talablar konseptual modelda umumlashtiriladi. Konseptual model abstrak vositalar yordamida quriladi va predmet sohasidagi hamma axborot obyektlarini qurish imkoniyatini beradi. Bunda predmet sohasini qanchalik keng, aniq va chuqurroq qurishimiz biz tanlagan modelga bog'liq bo'ladi. Minimal imkoniyatlarga ega bo'lgan model ma'lumotlarni va ular orasidagi o'zaro aloqalarni berish imkoniyatini ta'minlashi zarur. Konseptual modelning semantik quvvati uning yordamida aniqlanishi mumkin bo'lgan qo'shimcha xarakteristikalarining sonini ortishiga mos ravishda ortadi.

Ma'lumki, har bir model ma'lum bir cheklanishlarga ega bo'lib, o'zida faqat ma'lum bir xossalarni aks ettiradi. Shu sababli konseptual loyihalashtirish uchun model tanlashda, real dunyoni o'zida to'liq aks ettiradigan ideal modelni topish juda katta muammo ekanligini hisobga olish zarur. Avvalambor predmet sohasining xususiyatlari va MBga bo'lgan foydalanuvchining talablari modelni tanlash uchun asos bo'ladi. Boshqa muhim talab sifatida konseptual modelning MBBTga bog'liq emas, uni konseptual sxema qurilgandan so'ng tanlash kerak. Predmet sohasini tahlil qilishda qo'llaniladigan model xilma-xil bo'lishi mumkin.

### **Mantiqiy loyihalash**

MB yaratishning eng zaruriy va mas'uliyatli bosqichlaridan biri – bu mantiqiy loyihalashtirishdir. Uning asosiy masalasi tanlangan MBBT uchun mo'ljallangan holda MB mantiqiy sxemasini ishlab chiqishdan iborat. Mantiqiy loyihalashtirish bosqichi konseptual loyihalashtirishdan farqli ravishda, u kompyuterning dasturiy vositasini to'liq hisobga olgan holda amalga oshiriladi. Ish mazmuni bo'yicha mantiqiy loyihalashtirish axborot tizimini va uni tashkil etuvchi qismlarni real MBBTga mos shaklda modellash-tirishdan iborat.

Mantiqiy loyihalashtirish jarayoni quyidagi bosqichlardan iborat:

1. Aniq bir MBBTni tanlash.
2. Konseptual sxemani mantiqiy sxemaga o'tkazish.
3. Zarur kalitlarni tanlash.
4. So'rov tilini tavsiflash.

Aniq bir MBBTni tanlash protsedurasini batafsil qaraymiz. MB loyihalashtirishni amalga oshirish uchun MBBTni tanlash juda katta mas'uliyat talab qiladi. Bu bir tomondan MBBTlarning juda ko'pligi

bo'lsa, ikkinchi tomondan ko'p sonli xarakteristikalar bo'yicha MBBTni baholash va ularning orasidan aynan shunday tizimni tanlash kerakki, u foydalanuvchi va ishlab chiqaruvchilar talablarini to'liq qanoatlantirishi mumkin bo'lsin. Chunki MBda axborotdan foydalanish va ishlov berish samaradorligi MBBTning qanchalik to'g'ri tanlashga bog'liq bo'ladi.

MBBTni tanlashning asosiy me'yorlaridan biri—bu ma'lumotlarni ishlatadigan ichki modelining konseptual sxemasini tavsiflash uchun qanchalik samarador ekanligini baholashdan iborat. Shaxsiy kompyuterlar uchun mo'ljallangan MBBTlarning ko'pchiligi, odatda, ma'lumotlarning relatsion yoki tarmoq modeliga tayangan holda ishlaydi. Zamonaviy MBBTlarning juda katta qismi relatsion model asosida yaratilgan. Agar relatsion tizim tanlangan bo'lsa, u holda MBning konseptual sxemasini relatsionga akslantirish (o'tkazish) oldinda turibdi.

Ishning mazmuni bo'yicha ma'lumotning tanlangan modeli (relatsion, tarmoq va iyerarxik) ma'lumotlar tizilmasini tavsiflash uchun vosita beradi. Protседuralar MBBT yadrosiga kiradigan ma'lumotlarni tavsiflash tilida bajariladi.

MBBTning ikkinchi tarkibiy qismi ma'lumotlarni manipulyatsiya qilish tilidan iborat. Undan MBni turli tadbirlar uchun ishlatishda foydalaniladi. Ko'p hollarda ma'lumotlarni manipulyatsiya qilish tili (MMT) dasturlashtirish tiliga o'rnatilgan (kiritilgan) bo'ladi. MMT turli imkoniyatlarga ega bo'lishi mumkin: quyi pog'onadagi til va yuqori pog'onadagi til. Odatda quyi pog'onadagi til protsedurali, yuqori pog'onadagisi esa deklarativ til bo'ladi. Protседurali tillardan foydalanish ma'lum tayyorgarlikni talab qiladi, deklarativ til bo'lsa ko'proq professional bo'lmagan foydalanuvchilar uchun yaroqli. Shuning uchun ma'lum MMTga ega MBBTni tanlash maxsus tayyorgarligi bo'lmagan foydalanuvchi uchun juda muhimdir. Bundan tashqari, MBBTga servis dasturlar va amaliy masalalarni yechish uchun vositalar kiradi.

### **Axborot obyektlari va ular orasidagi aloqalarni aniqlash**

Predmet sohasini tahlil qilishning ikkinchi pog'onasi axborot obyektlarini tanlash, har bir obyekt uchun zarur xossalarni berish, obyektlar orasidagi aloqalarni aniqlash, axborot obyektlariga qo'yiladigan cheklashlarni aniqlash, axborot obyektlari orasidagi aloqalarning turlarini va axborot obyektlarining tavsifnomalarini aniqlashdan iborat.

Axborot obyektlarini tanlayotganda quyidagi savollarga javob berishga harakat qilamiz:

1. MBda saqlanishi lozim bo'lgan ma'lumotlarni qanday sinflarga ajratish mumkin?

2. Har bir ma'lumotlar sinfiga qanday nom berish mumkin?

3. Har bir ma'lumotlar sinfi uchun qanday eng muhim tavsif-nomalarni (foydalanuvchining nuqtai nazaridan) ajratish mumkin?

4. Tanlangan tavsifnomalar to'plamlariga qanday nomlarni berish mumkin?

Axborot obyektlarini aniqlash – itaratsion jarayon. U axborotlar oqimining tahlili iste'molchilar bilan suhbat o'tkazish asosida amalga oshiriladi. Axborot obyektlarining tavsifnomalari ham xuddi shu usullar bilan aniqlanadi.

Keyin axborot obyektlari orasidagi aloqalarni aniqlaymiz. Bu jarayon borishida quyidagi savollarga javob berishga harakat qilamiz:

1. Axborot obyektlari orasidagi aloqalar turi qanday?

2. Har bir tur aloqalarga qanday nom berish mumkin?

3. Keyinchalik foydalanish mumkin bo'lgan qanday turdagi aloqalar bo'lishi mumkin?

4. Aloqa turlarining biror-bir kombinatsiyasi ma'noga egami?

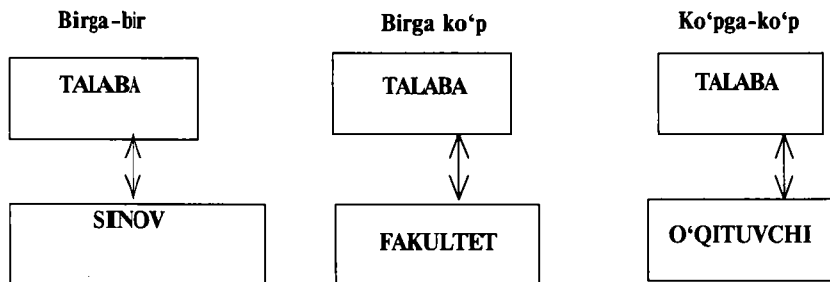
Obyektlarga, ularning tavsiflariga va aloqalariga cheklanishlar berishga harakat qilamiz. Buning uchun quyidagi savollarga javob berish zarur:

1. Sonli tavsifnomalar uchun qiymatlarning o'zgarish sohasi qanday?

2. Bir axborot obyektining tavsiflari orasida qanday funksional bog'lanishlar bor?

3. Har bir tur aloqalarga qanday turdagi akslantirishlar mos keladi?

Axborot obyektlarining o'zaro aloqasiga misol sifatida TALABA, SINOVA DAFTARI, FAKULTET, O'QITUVCHI kabi axborot obyektlarini qarash mumkin.

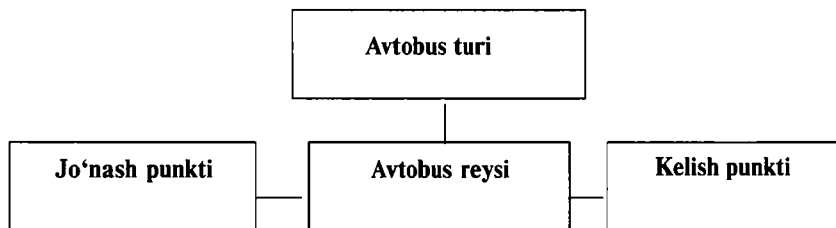


TALABA, SINOVA DAFTARI, FAKULTET,  
O'QITUVCHI axborot obyektlari o'rtasidagi o'zaro aloqa.

## Axborot tuzilmalarini qurish

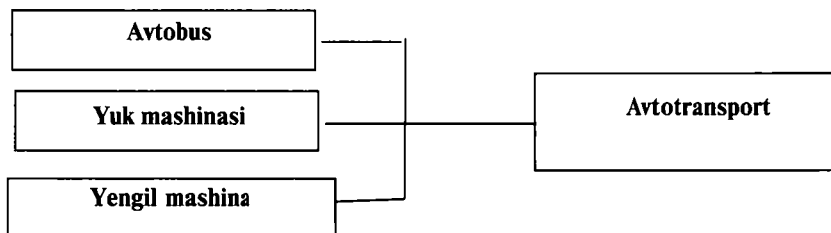
Predmet sohasini tahlil qilishning oxirgi bosqichi uning axborot tuzilmalarini (yoki konseptual sxemalarni) loyihalashtirishdan iboratdir. Predmet sohasini tavsiflash yoki konseptual sxemalarni loyihalashtirishni, buni maqsadlar uchun maxsus yaratilgan modellardan foydalangan holda amalga oshirish mumkin.

Ko'p hollarda konseptual sxemalarni qurish uchun an'anaviy agregatsiya va umumlashtirish usullaridan foydalaniladi. Agregatsiya qilishda axborot obyektlari (ma'lumotlar elementlari) ularning orasidagi semantik aloqalarga mos ravishda bir obyektga birlashtiriladi. Masalan, «DEAWOO» turidagi avtobus yo'lovchilarni jo'nash punktidan kelish punktiga tashiydi. Agregatsiya usuli bilan quyidagi atributlarga ega bo'lgan REYS axborot obyektini hosil qilamiz. «Avtobus turi», «jo'natish punkti», «kelish punkti», «avtobus reysi».



AVTOBUS REYSIning axborot obyekti.

Axborot obyektlarini (ma'lumotlar elementlarini) birlashtirishda ularni sinfdosh obyektlarga birlashtiriladi.



## **Axborot obyektlarini sinfdosh obyektlarga birlashtirish**

MB tizimini qurish masalasi quyidagi hollarda paydo bo'ladi:

1. Agar turli tadbirlar qandaydir miqdorda umumiy axborot obyektlarini talab qilsak, ammo hamma axborot obyektlari va ular orasidagi hamma aloqalarni bitta MBda amalga oshirib bo'lmasa.

2. Agar axborot obyektlari umumiy aloqaga, munosabatga va atributlarning umumiy turiga ega bo'lsa.

MB tizimining axborot tuzilmasini ko'rishga misol sifatida avtomobilsozlikda ilmiy-tadqiqot va tajriba konstruktorlik ishlarini axborot ta'minoti tizimining MB uchun axborot tuzilmasini ko'rish jarayonini qaraymiz.

Ushbu predmet sohasini tahlil qilish jarayonida quyidagi axborot obyektlari ajratildi:

1. Fizik effektlar. Ushbu predmet sohasidagi fizik effektlar texnik obyektlarning ishlash (harakatlanishi) tamoyili sifatida ko'rsatiladi.

2. Texnik qarorlar.

Bu qurilmaning ishlash usuli yoki modda.

3. Mahsulot.

Masalan, dvigatel.

4. Mahsulotni oluvchi obyekt.

Masalan, dvigatel avtomashinaga o'rnatiladi.

5. Usullar va uslubiyat.

6. Asboblar va stendlar.

7. Texnologik jarayonlar va uskunalar.

8. Tashkil qilish va mutaxassislar.

9. Me'yorlar.

Bu obyektlar orasidagi aloqalar qarab chiqilib ma'lumotlarning aloqalarini aniqlovchi va tasvirlovchi jarayonlar yoki hodisalar ajratiladi.

### **S a v o l l a r**

1. Tizimlashtirilmagan va tizimlashtirilgan axborotlarga misollar toping va ularning farqini tushuntirib bering.

2. Ma'lumotlar bazasi nima?

3. Ma'lumotlar bazasining maydon, yozuv va fayl elementlariga tushuntirish bering.

4. Maydonga tavsif bering.

5. Ma'lumotlar bazasini boshqarish tizimi qanday vosita vazifasini bajaradi?

6. Qanday MBBTlarni bilasiz?

# IX.DELPHI VIZUAL DASTURLASH VOSITASIDA MA'LUMOTLAR BAZASINI YARATISH TEXNOLOGIYALARI

## 9.1.MBni boshqaradigan ilovalar tuzish uchun Delphi vositalari

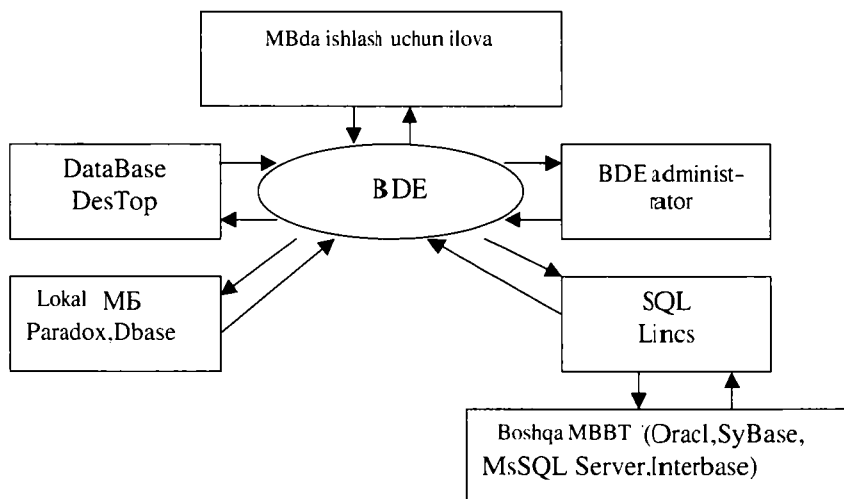
Delphi tarkibiga kiruvchi MBni boshqarishda ishlatiladigan ilovalarni tuzish va ishlatish vositalariga quyidagilar kiradi:

– BDE (Borland Database Engine). Borland ma'lumotlar bazasining mashinasi. Uning tarkibiga dasturlar to'plami kiritilgan bo'lib, ular lokal va kliyent server toifasining MBga murojaat qilishni va undan foydalanishni tashkil qilib beradi.

– SQL LINKS. Boshqa MBBT bilan (masalan, SysBase, Oracl, MsSQL Server) ishlash uchun drayverlar. Delphi tizimi Paradox va Dbase MBBT lari uchun SQLni ishlatmaydi, BDE yordamida bajaradi.

– DBE administrator – bu utilita bo'lib, MBga psevdonimlar, parametrlar va MB o'rnatish uchun ishlatiladi. Delphida tuzilgan ilova yordamida MB bilan ishlash vaqtida Mbdan foydalanish uning psevdonimi bo'yicha amalga oshiriladi.

– DataBase Desctop (DBD). Mbni ko'rish, tahrirlash va tashkil qilish uchun maxsus vosita (utilita). Bu utilita asosan Paradox va Dbase MBBT uchun uning jadvallari bilan ishlashga yo'naltirilgan bo'lib, ayrim hollarda boshqa tashqi MBBT jadvallari bilan ishlashda ham foydalaniladi. Ilovalarning o'zaro bog'lanishining umumiy modeli quyidagi sxemada berilgan:



– DataBase Explorer (SQL Explorer). MB psevdonimi, konfiguratsiyasi va tuzilmasini ko'rish, hamda MB jadvaliga so'rov berish utilitalarini o'z ichiga oladi.

– SQL monitor. SQL so'rovlarini bajarish vositasi.

– Visual Query Builder. Delphi tarkibiga kiruvchi vosita bo'lib, SQL –so'rovlarini avtomatlashtirishni tashkil etadi.

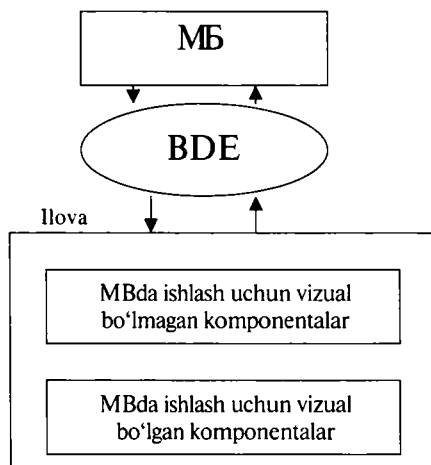
– Data Dictionary –ma'lumotlar lug'ati. MB jadvali maydonlari atributini saqlaydigan vosita.

– MB bilan ishlash uchun vizual bo'lmagan komponentalar. Vizual bo'lmagan komponentalar ilovalar bilan MB jadvalini bog'lashga xizmat qiladi. Bu komponentalar Data Access komponentalar palitrasida joylashgan.

– MB bilan ishlash uchun vizual komponentalar. Delphi vizual komponentalari ma'lumotlar to'plami yozuvlarini (masalan, komponenta TDBGrid) akslantirish uchun va foydalanuvchiga qulay interfeys yaratish uchun ishlatiladi. Bu komponentalar Data Controls komponentalar palitrasida joylashgan.

– Hisobotlar tuzish uchun komponentalar. Bu komponentalar 20 dan ortiq bo'lib, ular Qreport komponentalar palitrasida joylashgan hisobotlarni tuzish uchun ishlatiladi.

MB tayyor ilovasi bilan ishlash uchun vositalarning umumiy tarkibi quyidagi sxemada keltirilgan:



Bu sxemaga asosan, biz quyidagi ketma-ketlik zanjiriga ega bo'lamiz.

**Ilova => BDE => MB**

Vizual bo'lmagan komponentalardan BDE ga to'g'ridan-to'g'ri chiqiladi, u foydalanuvchi interfeysini ta'minlaydi. Delphi yordamida

MB bilan ishlash uchun yaratilgan dasturlarning asosiy xususiyati ularda BDEning ishlatilishidir. BDE ning asosiy vazifasi dasturlar bilan MB o'rtasida bog'lovchi ko'prik vazifasini bajaradi.

## 9.2.MB bilan ishlash uchun Delphi komponentalari

Delphi MB bilan ishlashda yetarlicha katta guruh komponentalariga ega.

**Data Access** (ma'lumotlarga ruxsat yoki ma'lumotlardan foydalanishga ruxsat) sahifasida MBni boshqa ma'lumotlar bilan o'zaro ta'sirida ishlatiladigan komponentalar mavjud. Ularning ko'pi vizual bo'lmagan (ko'rinmaydigan) bo'lib, o'z ichiga jadval, so'rov, ko'rish, o'zgartirish va boshqalar tavsifini oladi.

**Data Controls** (ma'lumotlar bilan bog'liq elementlarni boshqarish) sahifasida asosan vizual komponentalar bo'lib, ularga ma'lumotlar bilan bog'liq komponentalar deyiladi.

Delphi ma'lumotlar bazasi bilan muloqot (unga murojaat) qilishi uchun Data Source komponentasini ishlatadi. Bu komponenta to'g'ridan-to'g'ri ma'lumotlarni belgilamaydi, u Data Set komponentasiga murojaat qiladi. Quyidagi jadvallarda vizual va vizual bo'lmagan komponentalar tavsifi berilgan:

### MB bilan ishlash uchun vizual bo'lmagan asosiy komponentalar haqida ma'lumotlar

Komponent	Vazifasi
Tsession	MB bilan aloqa o'rnatish seansi bo'lib, MB ochishda, yopishda va uni parametrlari boshqarishda ishlatiladi
Tdatabase	MB. Bu komponenta himoyalashgan MB bilan birlashtirish jarayonini boshqarish uchun xizmat qiladi
TDatasource	Ma'lumotlar manbai. Ma'lumotlarga murojaat qilish komponentalari bilan ma'lumotlarni aks ettirish komponentalari o'rtasida bog'lovchi element vazifasini bajaradi
TDataSet	Kliyentlar ma'lumotlar to'plamiga, ma'lumotlarga MBni mashinasidan foydalanmasdan murojaat qilish vositasi sifatida ishlatiladi. MB bilan ishlashda xossa va metodlarni aniqlaydi
TTable	MBning jadvaliga (fayliga) kirish vosita sifatida xizmat qiladi
Tquery	So'rov. Ma'lumotlar jadvalini SQL tili yordamida tanlab olish imkonini beradi
TIndex Dets	MB jadvali indeksleri haqida ma'lumotni beradi
TField Dets	MB jadvali maydonlari haqida ma'lumotni beradi
TBatch Move	Bir MB to'plamini boshqasiga ko'chirishda ishlatiladi



## MB bilan ishlash uchun vizual komponentalar haqida ma'lumotlar

Komponent	Vazifasi
TDBtext	Ma'lumotlar to'plami maydonining joriy yozuvini ko'rsatadi
TDBEDIT	Joriy yozuvni ko'rish va maydon qiymatlarini o'zgartirishni ta'minlaydi (tahrirlash)
TDBCheckBox	Mantiqiy turga (Boolean) ega bo'lgan maydonlarning joriy yozuvini ko'rish va uning qiymatlarini tahrirlashni ta'minlaydi
TDBMenu	Menyu — maydon(посл комментария) qiymatlarini matn muharriri rejimida ko'rish va o'zgartirishni ta'minlaydi
TDBGrid	MB jadvali. Ma'lumotlar to'plamini jadval ko'rinishida chiqarishni ta'minlaydi
TDBNavigator	MB navigatori. MB yozuvlarini yengillashtirishni ta'minlaydi. Shuningdek, yozuvlarni qo'yish, olib tanlash va tahrirlash imkonini beradi
TDBChart	Ma'lumotlarni grafik ko'rinishda tasvirlashda ishlatiladi

Vizual va vizual bo'lmagan komponentalar bir-birlari bilan xossalar yordamida bog'lanadi. Xossalar asosan ilovalarni ishlab chiqishda aniqlanadi.

### 9.3.BDE administrator utilitasi bilan ishlash

Delphida ilova yordamida MBdan foydalanish uchun

**Ilova => BDE => MB**

ketma-ketligi bajariladi. Bu shuni bildiradiki, har qanday MBga ilovadan murojaat qilinganda uning aniq adresi BDEga uzatiladi. BDE o'zining maxsus funksiyalarini MB bilan bog'lanishda ishlatadi. BDE aniq bir MB bilan ishlaganda quyidagilarni bilishi kerak.

- MB qaysi joyda joylashganligini;
- MB parametrlari haqidagi ma'lumotni.

MB parametrlari va uning joylashishi MB psevdonimida aniqlanadi. Psevdonim – MB ga berilgan biror nom bo'lib, MBga mantiqiy murojaat qilinganda ishlatiladi.

MB psevdonimi BDE administrator utilitasi yordamida aniqlanadi.

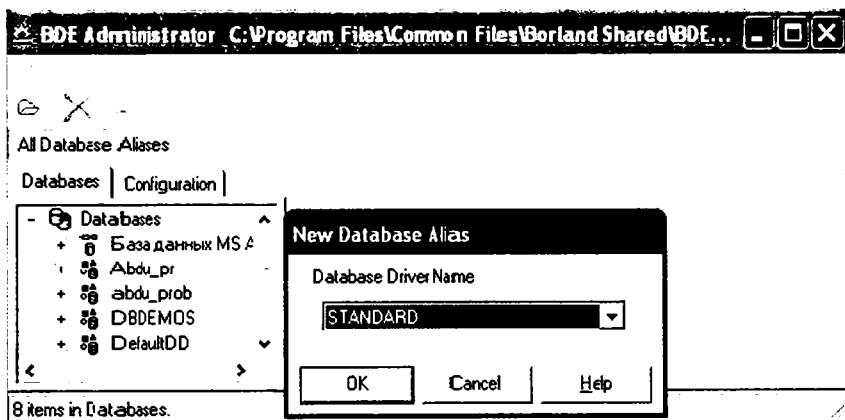
MBning har bir jadvali uchun fayl tuziladi. Xuddi shunday jadval indeksleri va memo maydonlari uchun maxsus fayllar tuziladi. MBga dastur va utilitalardan murojaat qilinish MBning psevdonimida amalga oshiriladi. Psevdonim BDE administrator utilitasi yordamida

aniqlanadi. Pseudonim – biror-bir nom bo‘lib, MBga Delphi komponenti ilovalari (masalan Ttable va Tquery) tomonidan mantiqiy murojaat qilishda ishlatiladi.

Aytaylik, biz tashkil qilishimiz kerak bo‘lgan MB «C:\PROBA\» katalogida joylashsin. Biz tashkil qiladigan pseudonim nomi aytaylik «Proba» bo‘lsin. BDE administrator utilitasini ishga tushiramiz. Asosiy menyudan Object/New buyruqlarini tanlaymiz. Hosil bo‘lgan darchadan tuziladigan MB turini aniqlash uchun «Standart» parametrini o‘zgartirmasdan Ok tugmasini bosamiz. Chap oynada (MB administratori oynasida) biz «Standart1» nomini ko‘ramiz va uni «Proba»ga o‘zgartiramiz. O‘ng oynada MB parametrlari berilgan, u yerda faqat Path (yo‘l) o‘zgartiriladi. Bu parametr MB katalogiga yo‘lni ko‘rsatadi. Path nomini ko‘rsatib, o‘ng darchadan uch nuqtali tugmani va keyin chap oynadan «Proba»ni tanlab Ok tugmasini bosamiz. MB uchun aniqlangan pseudonimni saqlash uchun chap oynaga kelib, sichqonchanning o‘ng tugmasini chiqillatib dialog oynasidan «Apply» elementini tanlaymiz va Ok tugmasini bosamiz. Keyin esa BDE administratoridan chiqamiz.

MBni pseudonimini tuzishni yanada chuqurroq ko‘rib chiqaylik. Uning algoritmi quyidagicha:

1. BDE administrator utilitasi ishga tushiriladi.
2. Menyudan Object=>New buyrug‘i beriladi.
3. Darchadan Standart turi o‘zgartirilmasdan Ok tugmasi bosiladi.

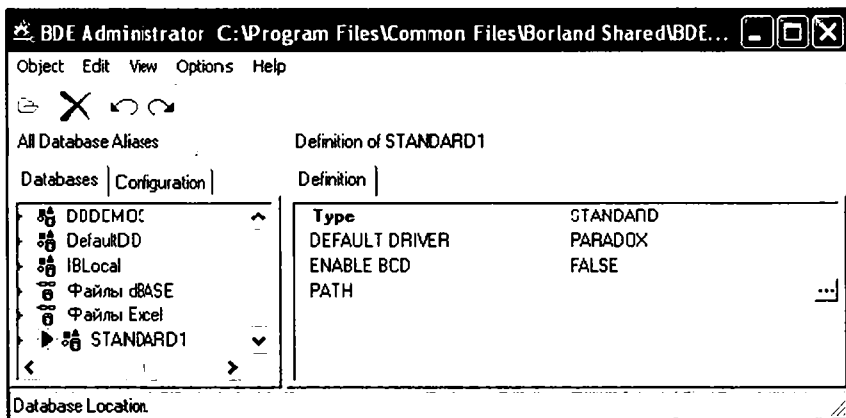


4. Chap oynadan Standart1 nomi yangi pseudonim nomiga o‘zgartiriladi.

5. O‘ng oynadan Path qatoriga o‘tilib, uch nuqtali tugmacha va keyin Ok bosiladi.

6. Menyudan Object=>Apply=>Ok buyrug'i beriladi.

7. BDE administrator dan chiqiladi.



## 9.4. MB jadvalini tuzish

MB jadvalini tuzish uchun DataBase Desktop (DBD) utilitasini ishga tushirish zarur. Utilitani ishga tushirgandan keyin ishchi psevdonimi o'ratiladi. Ishchi psevdonimni o'rnatish uchun bosh menyudan File=>Working Directory buyrug'ini berib, keyin ro'yxatga kiruvchi «Aliases» tanlanib psevdonim nomi «Proba» ko'rsatiladi va Ok tugmasi bosiladi.

MB jadvalini tuzish uchun asosiy menyudan File=>New=>Table buyrug'i bajariladi. Paydo bo'lgan Create Table oynada MB turini aniqlab, (masalan Paradox 7 yoki DBase Windows) Ok tugmasi bosiladi. Yangi paydo bo'lgan oynadan MB jadvali tuzilmasi aniqlanadi.

Ma'lumotlar bazasini yaratish uchun avvalambor uning modelini, ya'ni, uning tuzilmasini ishlab chiqarish zarur. Buning uchun MBga maydon tushunchasi kiritilgan bo'lib, uning tuzilmasini aniqlaydi. Maydonni aniqlash quyidagi elementlardan tashkil topadi.

- ◆ maydon nomi (Fields Name);
- ◆ maydon turi (Type);
- ◆ maydon o'lchami (Size);
- ◆ maydon kaliti (Key).

Maydon nomi – 10 tagacha lotin harfi va sonlardan tashkil topgan bo'lishi mumkin.

Maydon turlari matnli, sana, mantiqiy, izohli (примечание) va sonli bo'lishi mumkin.

Maydon o'Ichami – simvol yoki sonlarning maksimal soni.

Masalan, DBase MBBT uchun maydon turlari quyidagilar bilan aniqlanadi.

– «C» simvolli (matnli) qiymatlarni bildirib, uning uzunligi 255 tagacha bo'lishi mumkin;

– «N» sonli qiymatlarni bildirib, uning diapazoni –  $10^{307}; +10^{308}$  ;

– «D» sana (data) qiymatini bildirib, uning o'zgarish diapazoni 01.01.9999 dan 31.12.9999 gacha;

– «L» mantiqiy qiymatni bildirib, «True» va «False» qiymatlarni qabul qiladi;

– «M» memo (izohli) bo'lib, qatorli qiymatlarni qabul qiladi, har bir qator 255 tagacha simvol olishi mumkin. Memo ma'lumotlari maxsus Dbt kengaytmali faylda saqlanadi.

MB ning turli fayllari orasidagi munosabat bir xil nomli maydonlar orqali o'rnatiladi. Har bir MB DBF kengaytirilgan nom bilan kompyuter xotirasida saqlanadi.

**Misol 1.** Misol tariqasida xodimlarni hisobga olishni avtomatlashtirish masalasini qaraylik. MBni tashkil qilish uchun oldin uning kartatekasini ishlab chiqarish lozim: hisobga olish tartib raqami; ismi va familiyasi; tug'ilgan yili; maoshi va hokazo. Bularni hisobga olgan holda MBning tuzilmasini ishlab chiqamiz.

*1-jadval*

Kodx	N	3	-	Hisobga olish tartib raqami
Famx	C	20	-	Xodim ismi va familiyasi
Tyx	D	8	-	Tug'ilgan yili
Omx	N	10	2	Oylik maoshi
MUT	C	15	-	Mutaxassisligi
Lavozim	C	15	-	Lavozimi
BULIM	C	15	-	Bo'lim nomi
Ichl	D	8	-	Ishga qabul qilingan kun

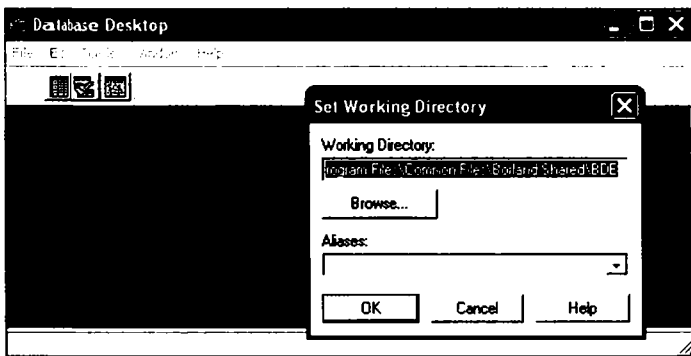
MBning jadvalini yaratish algoritmi quyidagi ketma-ketlikda bajariladi:

1. DataBase DeskTop utilitasi ishga tushiriladi. Buning uchun quyidagi buyruqlar ketma-ket bajariladi.

**Пуск=>Программы=>Borland Delphi=>DataBase DeskTop**

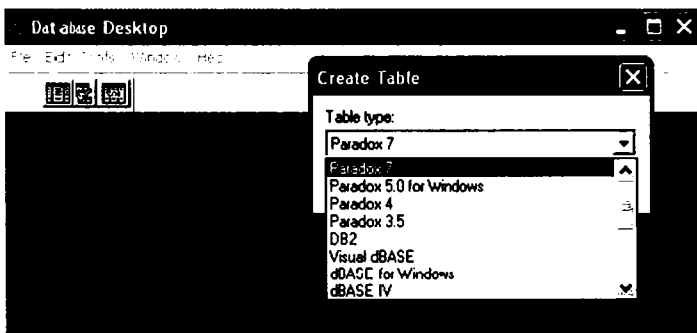
2. DataBase DeskTop oynasining bosh menyusidan quyidagi buyruq bajariladi. **File=>Working Directory**

3. Hosil bo'lgan Set Working Directory muloqot darchasining Aliasesidan MB psevdonimi aniqlanadi.

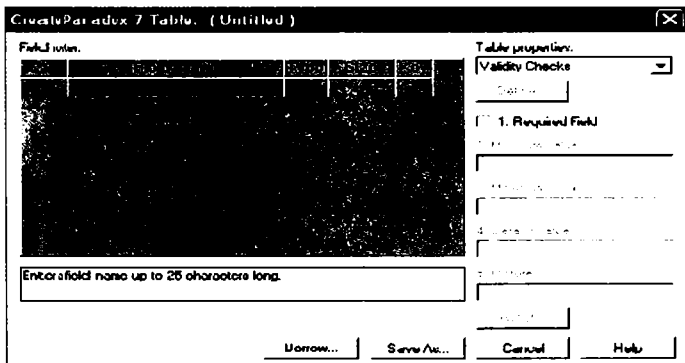


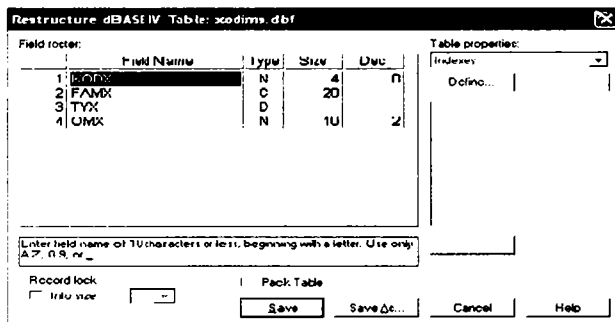
4. Menyudan File=>New=>Table buyrug'i bajariladi.

5. Hosil bo'lgan Create Table muloqot darchasidan MB turi aniqlanadi (masalan, Paradox yoki Dbase Windows). Bu darchada MBning bir qancha turlari mavjud.



6. MB tuzilmasi tashkil qilinadi.





7. Save As buyrug' i berilib, jadval nomi kiritiladi va Ok tugmasi bosilib saqlanadi.

8. Agar MBga ma'lumotlar kiritish kerak bo'lsa menyudan File=>Open buyrug' i berilib, keyin fayl tanlanadi va ma'lumotlar kiritiladi.

## 9.5. MB jadvali uchun oddiy dastur ilovalarini tuzish

Delphi dasturlash vositasida tuzilgan ilovalarni saqlash uchun yuqoridagi tuzilgan «Proba» katalogida «App» nomli podkatalog tashkil qilamiz. Delphi tizimini ishga tushirib, uning komponentalar palitrasi qatoridan Data Accessni ishga tushirib, undan vizual bo'lmagan komponentasi Ttable olinadi (bu komponentani darchaga olish uchun u ko'rsatilib, sichqonchani chap tugmasini ikki marta tez-tez chiqillatmoq kerak bo'ladi). Ttable komponenti ma'lumotlarni saqlash va undan foydalanishda ishlatilib, u ma'lumotlarni akslantirishda vizual komponentalar DTBGrid, Tedit va boshqalar bilan birgalikda ishlatiladi.

Ttable komponentasini formaga (darchaga) joylashtirgandan keyin, Object Inspectorida uning xossalari quyidagi ketma-ketlikda o'rnatiladi:

- Ttable komponentasi ajratiladi (belgilanadi);
- DataBase Name MB psevdonimi xossasi o'rnatiladi, Proba psevdonimi ro'yxatdan olinadi yoki kiritiladi;
- Table Name (MB jadvali nomi) xossasi o'rnatiladi (bu yerda MB jadvali nomi ro'yxatdan olinadi yoki kiritiladi);
- Active xossasi o'rnatiladi («True» qiymati tanlanadi).

Bu bajarilgan buyruqlardan keyin Ttable komponentasi bilan MB jadvali orasida aloqa to'liq o'rnatiladi. Ttable komponentasi kabi endi formaga TDataSource komponentasini joylashtiramiz. Bu

komponenta vizual va vizual bo'lmagan komponentalar o'rtasida aloqa o'rnatish uchun xizmat qiladi. Shu tufayli TdataSource komponentasiga ma'lumotlar manbai deyiladi. TdataSource komponentasi uchun DataSet (ma'lumotlar to'plamining nomi) xossalari o'rnatiladi (Table1 nomi olinadi).

Data Controls menyu qatoridan foydalanib formaga TdbGrid komponentasini joylashtiramiz va uning DataSource xossasini o'rnatamiz (DataSource1 qiymat bilan). Bu TdbGrid komponentasi ma'lumotlar to'plamining yozuvlarini jadval ko'rinishida akslantirishda xizmat qiladi.

Ishlab chiqilgan loyihani saqlash uchun menyudan quyidagi buyruqlar ketma-ket bajariladi. File=>Save Project As. Oldin loyiha formasi (masalan, Appl1.pas nomi bilan), keyin loyihaning o'zi (masalan Appl.dpr nomi bilan) saqlanadi.

Delphi tizimidan chiqmasdan turib tuzilgan ilovani ishga tushirish uchun F9 tugmasini bosish kifoya. Ilovani tizimdan tashqarida ishlatish uchun esa oldin tizim ichida Ctrl+F9 tugmasini bosish kerak bo'ladi. Bu holda ilovani tizimdan tashqarida ishlatish uchun maxsus .EXE kengaytmali fayl avtomatik ravishda tashkil etiladi (masalan, Appl.exe). Bu faylni tizimdan tashqarida ishlatganda tuzilgan ilova ishga tushadi. Mbga qo'shimcha yozuv kiritish uchun oxirgi yozuvga kelib Insert tugmasini bosish, kiritilayotgan yozuvdan voz kechish uchun ESC tugmasini bosish, yozuvni to'liq o'chirish uchun esa Ctrl+Del tugmasini bosish kerak bo'ladi.

MB jadvali bilan ishlash uchun oddiy ilova yaratish algoritmi quyidagi ketma-ketlikda bajariladi:

- 1.Delphi tizimi ishga tushirilib, BDE komponentalar palitrasidan Ttable komponentasi formaga qo'yiladi.

- 2.Formadagi Ttable komponentasi belgilanib, DataBase Name xossasida Mbning psevdonimi aniqlanadi.

- 3.TableName xossasidan MB jadvali nomi aniqlanadi.

- 4.Active xossasi True qiymat bilan o'rnatiladi.

- 5.Data Access komponentalar palitrasidan TdataSource komponentasi formaga qo'yiladi.

- 6.TdataSource xossasi Table1 nomi bilan o'rnatiladi.

- 7.Data Controls komponentalar palitrasidan TdbGrid komponentasi formaga qo'yiladi.

- 8.DataSource xossasi DataSource1 nomi bilan o'rnatiladi.

- 9.Menyudan File=>Save Project As buyrug'i berilib, oldin forma keyin loyiha saqlanadi.

## O'qituvchilar haqida ma'lumotnoma

KODX	FAMX	TYX	OMX
136	Karimov Muzaffar	02/08/1981	100000
137	Sobirov Naimjon		80000
138	Zufarov Zafarjon		120000
139	Usmonov Sobir		80000
140	Ahmedov Sahob		100000
141	Naimov Akbarjon		110000
142	Mahmudov Tohirjon		110000

10.Loyihani ishga tushirish uchun F9 tugmasi bosiladi. Natijada quyidagi formaga ega bo'lamiz.

Form1

### O'qituvchilar haqida ma'lumotnoma

KODX	FAMX	TYX	OMX
136	Karimov Muzaffar	02/08/1981	100000
137	Sobirov Naimjon		80000
138	Zufarov Zafarjon		120000
139	Usmonov Sobir		80000
140	Ahmedov Sahob		100000
141	Naimov Akbarjon		110000
142	Mahmudov Tohirjon		110000

**TDBNavigator komponenti.** MB jadvalida ma'lumotlarni surish, o'chirish, yozuvni siljitish va tahrirlash uchun Data Controls komponentalar palitrasida maxsus TDBNavigator komponentasi mavjud.

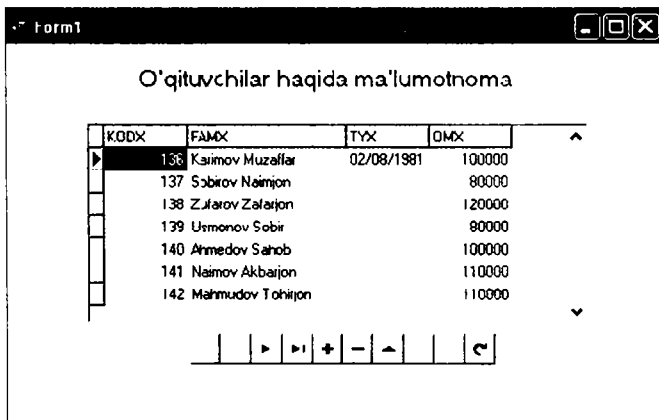
Bu komponentani formadagi MB jadvaliga quyidagi tartibda o'rnatish mumkin.

- 1.MB jadvali formasi ekranga chaqiriladi.
- 2.Data Controls komponentalar palitrasidan TDBNavigator komponentasi formaga joylashtiriladi.
- 3.TDBNavigator komponentalar xossasidan DataSource xossasi DataSource1 nomi bilan o'rnatiladi.
- 4.Menyudan File=>Save Project As buyrug'i berilib, oldin forma keyin loyiha saqlanadi.



5. Loyihani ishga tushirish uchun F9 tugmasi bosiladi.

Natijada jadvaldagi ma'lumotlarni surish, o'chirish, yozuvni siljitish va tahrirlash kabi tugmachalarga ega bo'lgan quyidagi formaga ega bo'lasiz.



Kompyuter quyidagi dastur kodlarini avtomatik ravishda tuzadi:

**Unit xodim;**

**Interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,**

**Dialogs, StdCtrls, Grids, DBGrids, DB, DBTables, ExtCtrls, DBCtrls;**

**Type**

**TForm1 = class(TForm)**

**Label1: TLabel;**

**Table1: TTable;**

**DataSource1: TDataSource;**

**DBGrid1: TDBGrid;**

**DBNavigator1: TDBNavigator;**

**Private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation**

```
{SR *.dfm}
```

```
end.
```

## Ikkita jadvalda ishlash uchun ilova tuzish

Bitta formada ikkita bog‘liq ma‘lumotlar to‘plami uchun ilova tuzishni ko‘rib chiqaylik. Forma ilovasini «App21.pas» fayliga, loyiha ilovasini «App2.drp» fayliga yozib qo‘yaylik.

Ilovaga Ttable komponentasini qo‘shamiz (Table2 nomi bilan). Проба Мbning Приход jadvali bilan ishlash uchun uning xossalarini qo‘yamiz (xossalar qiymati Table1 komponentasi kabi bo‘lib, Table Name ga jadval nomi Приход.dbf beriladi). Table2 Aptive xossasini «True» qiymatiga o‘rnatamiz. Formaga TdataSurce komponentasini joylashtiramiz (DataSource2 nom bilan). Bu komponenta uchun DataSet xossasini Table2 qiymati bilan o‘rnatamiz. TDBGrid komponentasini formaga joylashtiramiz (DBGrid2 nomi bilan) va DataSource xossasini o‘rnatamiz (DataSource1 qiymati bilan). Ilovani ishga tushiramiz.

## 9.6.Ma‘lumotlarni izlash va filtrlash

Komponenta TDataset va uning davomchilari ma‘lumotlar bilan ish yuritishda maxsus usullarga ega:

- ◆ maydon qiymati bo‘yicha ma‘lumotni izlash;
- ◆ ma‘lumotlarni filtrlash;
- ◆ zakladka qo‘yish va unga o‘tish.

**Ma‘lumotni izlash.** Ma‘lum belgilangan yozuvlarni ma‘lumotlar to‘plamidan izlab topish uchun ikkita usul mavjud: Locate va Lookup.

**Locate** – usuli biror maydonning berilgan yozuvi bo‘yicha kerakli yozuvni topish imkonini beradi. Uning umumiy ko‘rinishi quyidagicha:

**Function Locate(Const KeyFields: String; Const KeyValues: Variant; Options: TLocateOptions): Boolean;**

Bu yerda

**KeyFields** – ma‘lumotlarni izlashda qatnashadigan maydon nomlari. Ular bir-biridan nuqta, vergul bilan ajratiladi.

**KeyValues** – bir yoki bir necha izlanadigan maydon qiymatlari. Agar izlanadigan qiymatlar bir necha bo‘lsa, massiv variant funksiyasi qilib berish zarur.

**Options** – izlanadigan parametrlar to‘plami. U quyidagi qiymatlarni saqlashi mumkin:

loCaseInsensitive. Registrni hisobga olmasdan izlash.

loPartialKey. Maydon qiymatini to'liq berilmagan holda izlash. Masalan, 'So' boshlanadigan familiyalar izlanadigan bo'lsa. U holda ma'lumotlar to'plamidan So bilan boshlanadigan familiyalar 'Sobirov' va 'Soatov' lar topiladi.

Agar izlanayotgan yozuv topilsa, funksiya Locate – true qiymatni qaytaradi.

**Misol 2.** Misol tariqasida yuqorida tuzilgan o'qituvchilar MBni olaylik va MBdan kerakli o'qituvchini izlab topish uchun ilova yarataylik.

**Ilovani yaratish algoritmi:**

1. Delphini ishga tushuramiz.

2. Formaga Label1 komponentasini o'rnatamiz va uning Caption xossasini «Ma'lumotlar bazasidan izlash» so'ziga almashtiramiz.

3. Formaga DataSource (Ma'lumotlar manbai), Query (So'rov) va DBGrid komponentalarini joylashtiramiz. Ularning quyidagi xossalarini o'rnatamiz.

DataSource1 komponenti

Xossa	Qiymati
DataSet	Query1

Query1 komponenti

Xossa	Qiymati
DataBaseName	ABDU_PR
RequestLive	True
SQL	Select * From Xodims

DataSource1 komponenti

Xossa	Qiymati
DataSource	DataSource1

4. Yuqoridagilarni to'g'riligini tekshirish uchun Query1 komponentasining Active xossasini True qilib o'rnatamiz.

5. Data Controls komponentalar palitrasidan TDBNavigator komponentasini formaga joylashtiramiz.

6. TDBNavigator komponentalar xossasidan DataSource xossasini DataSource1 nomi bilan o'rnatamiz.

7. Button1 komponentasini formaga joylashtiramiz va uning Caption xossasini «Chiqish» so'ziga almashtiramiz.

8. Edit1 komponentasini formaga joylashtiramiz.

9. Button2 komponentasini formaga joylashtiramiz va uning Caption xossasini «Kod bo'yicha izlash» so'ziga almashtiramiz.

Natijada quyidagi formaga ega bo‘lamiz:

KODX	FAMX	TYX	OMX
136	Karimov Muzaffarjon	02/08/1981	100000
138	Zufarov Zafarjon	03/09/1970	120000
139	Usmonov Sobir	04/02/1968	90000
140	Ahmedov Sahob	02/07/1969	100000
141	Naimov Akbarjon		110000

10.«Kod bo‘yicha izlash» tugmasini ikki marta tez-tez chiqillatib quyidagi dastur kodini kiritamiz.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Query1.Locate('Kodx', Edit1.Text, [LopartialKey,  
        LoCaseInsensitive]);  
end;
```

Bu yerda 'Kodx' –xodimlar kodini tasvirlovchi maydon nomi.

11.Ilovani ishga tushiramiz.

Natijada quyidagi forma ilovasiga ega bo‘lamiz:

Bu formani ishga tushirgandan so‘ng Edit1 tahrirlash qatoridan kerakli o‘qituvchi kodi kiritilib «Kod bo‘yicha izlash» tugmasi bosiladi.

Bu yerda izlash Kodx maydoni bo'yicha amalga oshirilayapti. Izlash 140 kodi kiritilib «Kod bo'yicha izlash» tugmasi bosilayapti. Natijada shu kodli xodim topilgan, chunki kursor shu kod to'g'risida turibdi. Buni familiya va ism bo'yicha amalga oshirish ham mumkin. Uning uchun `Kodx` maydoni o'rniga `Famx` maydonini yozish kifoya.

MBdan izlashni ikkita maydon bo'yicha ham tashkil qilish mumkin, buning uchun ikkinchi tahrirash maydonini kiritish kerak bo'ladi.

Tuzilgan dastur kodlari quyidagicha bo'ladi:

**Unit Xodim1;**

**interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,**

**Dialogs, StdCtrls, Orids, DBORids, DB, DBTables, ExtCtrls, DBCtrls;**

**Type**

**TForm1 = class(TForm)**

**DataSource1: TDataSource;**

**Query1: TQuery;**

**DBGrid1: TDBGrid;**

**Button1: TButton;**

**Button2: TButton;**

**Edit1: TEdit;**

**Label1: TLabel;**

**DBNavigator1: TDBNavigator;**

**procedure Button2Click(Sender: TObject);**

**private**

**{ Private declarations }**

**public**

**{ Public declarations }**

**end;**

**Var**

**Form1: TForm1;**

**implementation**

**{ \$R \*.dfm }**

**procedure TForm1.Button2Click(Sender: TObject);**

**begin**

**Query1.Locate('kodx',edit1.Text,[LopartialKey,LoCaseInsensitive]);**

**end;**

**end.**

Endi TDataSet obyektini davomchisi bo'lgan ikkinchi LookUp izlash funksiyasini ko'rib chiqamiz. Bu funksiya ham Locate funksiyasiga juda o'xshash bo'lib, uning ko'rinishi quyidagichadir:

**Function LookUp(Const KeyFields: String; Const KeyValues: Variant; const ResultFields: String): Variant;**

Bu funksiyaning Locate funksiyasidan farqi shundaki, u topilgan yozuvni joriy deb aniqlamaydi, u topilgan yozuvning berilgan maydon qiymatini qaytaradi. Xuddi oldingi holdagi kabi uning parametrlari quyidagilarni aniqlaydi.

**KeyFields** – ma'lumotlarni izlashda qatnashadigan maydon nomlarining ro'yxati. Ular bir-biridan nuqta, vergul bilan ajratiladi.

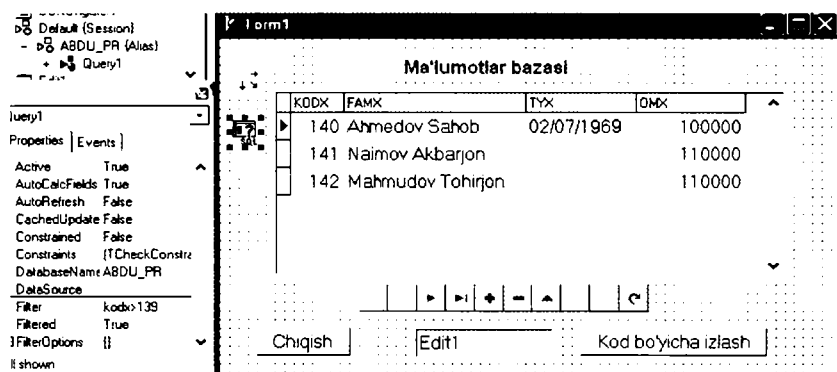
**KeyValues** – bir yoki bir necha izlanadigan maydon qiymatlari. Agar izlanadigan qiymatlar bir necha bo'lsa massiv varianti funksiyasi qilib berish zarur.

### Ma'lumotlarni filtrlash

Ma'lumotlarni filtrlashning ikkita asosiy usuli mavjud:

- filtr xossasini ishlatish;
- OnFilterRecord obrabotchigini (qayta ishlovchisini) tashkil qilish.

Filter xossasi MB yozuvini qanoatlantiruvchi shartni o'z ichiga oladi. Shart solishtirish va mantiqiy operatorlarni o'z ichiga olishi mumkin. Masalan, Kodx>139 And Omx>100000. Bu shart bizning yuqorida yaratgan MB uchun kodi 139 dan katta va maoshi 100000dan katta xodimlarni ekranga chiqaradi. Quyidagi oynada kodi 139 dan katta xodimlarni ekranga chiqarish tasvirlangan.



Filter xossasiga shartni kiritish uchun oldin Query1 komponentasi belgilanadi va keyin obyekt inspektoridan Filter xossasi topilib, uning qatoridan Kodx>139 sharti kiritiladi va keyin Filtered xossasi true qiymatiga almashtiriladi.

## 9.7. So'rovlar hosil qilish

Zamonaviy ma'lumotlarni boshqarish tizimlari kerakli ma'lumotlarni so'rovlar yordamida tanlab olishga imkon beradi. Foydalanuvchi ma'lum qoidalarga asosan so'rov hosil qiladi, tizim bo'lsa shu so'rovga mos keluvchi yozuvlarni ajratib beradi.

Ma'lum talablarga javob beruvchi yozuvlarni ajratib olish uchun Query komponentasidan foydalaniladi.

Query komponentasi xossalari:

Xossa	Ta'rifi
Name	Komponenta nomi. Datasource komponentasi tomonidan so'rov natijalarini, yozuvlarni ko'rishga imkon beruvchi komponentalar, misol uchun DBGrid bilan bog'lash uchun ishlatiladi.
SQL	SQL tilida yozilgan so'rov.
Active	Xossaga True qiymati berilganda so'rovni bajarish aktivlashadi.

Umumiy holda jadvaldan yozuvlarni tanlash uchun so'rov quyidagi ko'rinishga ega bo'ladi:

```
SELECT Maydonlar ro'yxati FROM Jadval WHERE (Shart)
ORDER BY Maydonlar ro'yxati.
```

Bu yerda ORDER BY – yozuvlarni tartiblash parametri.

Misol uchun:

```
SELECT Fam, Name FROM `:Maktab:school.db` WHERE
(Class = `10a`) ORDER BY Name, Fam
```

Bu so'rov «Maktab» ma'lumotlar bazasidan (School.db jadvalidan) 10-a sinfi talabalari ro'yxatini hosil qiladi.

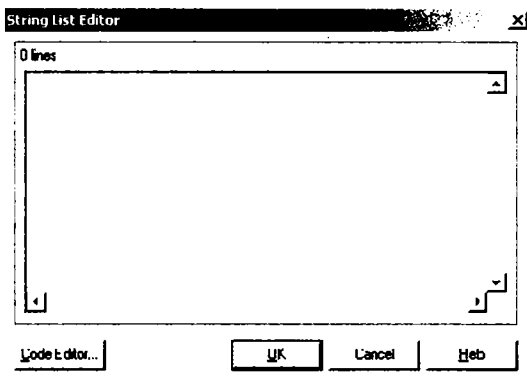
Misol uchun:

```
SELECT Fam, Name FROM `:Maktab:school.db` WHERE
(Fam > `K`) and (Fam < `L`) ORDER BY Name, Fam
```

Bu so'rov familiyasi K harfidan boshlanuvchi talabalar ro'yxatini hosil qiladi.

So'rov SQL xossasiga forma yaratish yoki dastur bajarilishi jarayonida yozilishi mumkin.

Forma yaratish jarayonida SQL xossasiga yozuv yozish uchun satrlar qatori muharriridan foydalaniladi. Bu muharrir Object Inspector oynasidagi SQL xossasi qatoridagi uch nuqtali tugmani bosish natijasida ochiladi.



SQL xossasi satrlar ro'yxatidan iborat. Dastur bajarilish jarayonida so'rov hosil qilish uchun Add usulidan foydalanib, SQL ro'yxatiga qatorlarni qo'shishi lozim.

Buning uchun avval joriy so'rovni berkitish, satrlar ro'yxatini o'zlashtirish lozim:

```
Query1.Close;
```

```
Query1.SQL.Clear;
```

Delphi so'rovni qayta ishlab, natijani jadval shaklida qaytarishi uchun quyidagi usulni chaqiradi:

```
Query1.Open;
```

Quyida konkret shaxs to'g'risida ma'lumotni qidirish uchun so'rov hosil qiluvchi dastur qismi berilgan. Qidiruv sharti Fam maydoni qiymati fam o'zgaruvchi qiymatiga teng bo'lishi kerak:

```
with form1.Query1 do begin
```

Close; закрыт файл — результат выполнения предыдущего запроса;

```
SQL.Clear; удалит текст предыдущего запроса
```

```
записываем новый запрос в свойство SQL;
```

```
SQL.Add('SELECT Fam, Name, Class');
```

```
SQL.Add('FROM `:Школа:school.db`');
```

```
SQL.Add('WHERE');
```

```
SQL.Add('(Fam = `` + fam + ``)');
```

```
SQL.Add('ORDER BY Name, Fam');
```

```
Open; // активизируем выполнения запроса
```

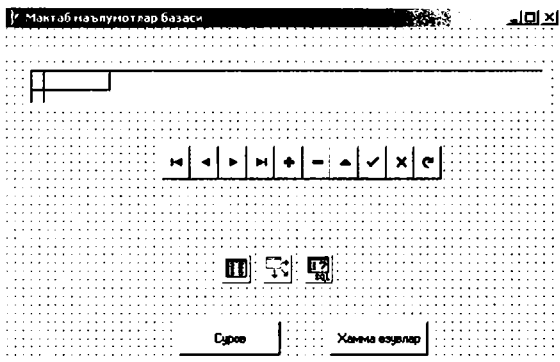
```
end;
```

Quyidagi dastur hamma o'quvchilar yoki bir qismi haqidagi ma'lumotlarni so'rovni bajarish orqali chiqarishga imkon beradi.



Ma'lumotlar bazasida so'rov natijalarini ko'rish uchun DBGrid1 komponentasi ishlatilib, DataSource1 komponentasi orqali Table1 yoki Query komponentasi bilan bog'lanadi.

Dastur formasini:



**Dastur matni:**

**Unit school2\_**

**interface**

**uses**

**Windows, Messages, SysUtils, Classes,**

**Graphics, Controls, Forms,**

**Dialogs, Orids, DBGrids, Db,**

**DBTables, ExtCtrls, DBCtrls, StdCtrls;**

**Type**

**TForm1 = class(TForm)**

**Table1: TTable; // таблица (вся база данных)**

**Query1: TQuery; // запрос (записи БД, удовлетворяющие критерию выбора)**

**DataSource1: TDataSource; // источник данных – таблица или запрос**

**DBGrid1: TDBGrid; // таблица для отображения БД или результата выполнения**

**DBNavigator1: TDBNavigator**

**DBText1: TDBText**

**Button1: TButton; // кнопка запрос**

**Button2: TButton; // кнопка Все записи**

**procedure Button1Click(Sender: TObject);**

**procedure Button2Click(Sender: TObject);**

**procedure FormActivate(Sender: TObject);**

```

private
{ Private declarations }
public
{ Public declarations }
end;
Var
Form1: TForm1;
implementation
{$R *.DFM}
// шелчок на кнопке Запрос
procedure TForm1.Button1Click(Sender: TObject);
Var
fam: string[30];
begin
fam:=InputBox('Выборка информации из БД'
'Укажите фамилию и щелкните на ОК')
if fam <> S'' // пользователь ввел фамилию
then
begin
with form1. Query1 do begin
Close; // закрыт файл-результат выполнения предыдущего
запроса
SQL.Clear; // удалит текст предидущего запроса
// записываем новый запрос с свойство SQL
SQL.Add('SELECT Fam, Name, Class');
SQL.Add('FROM `Школа:school.db`');
SQL.Add('WHERE');
SQL.Add('(Fam = `+ fam + `)');
SQL.Add(«ORDER BY Name, Fam`);
Open; // активизируем выполнение запроса
end;
{ *** другой вариант изменения критерия запроса
begin
Query1.Close;
Query1.SQL[3]:= `(Fam=`+ fam + `)`;
Query1.Open;
DataSource1.DataSet:=Query1;
end;
}
if Query1.RecordCount <> 0 then

```

```

DataSource1.DataSet:=Query1 // отобразит рез-т выполнения
запроса
else begin
ShowMessage('В БД нет записей, удовлетворяющих критерию
запроса.');
```

```

DataSource1.DataSet:=Table1;
end;
end;
end;
// щелчок на кнопке Все записи
procedure TForm1.Button2Click(Sender: TObject);
begin
DataSource1.DataSet:=Table1; // источник данных – таблица
end;
// активизация формы
procedure TForm1.FormActivate(Sender: TObject);
begin
DataSource1.DataSet := Table1;
Table1.Active := True;
end;
end.
```

TForm1.Button1Click protsedurasi so'rov tugmasi bosilganda bajariladi. U foydalanuvchidan qator (familiya) qabul qilib, SQL xossasiga yozish orqali so'rov matnini hosil qiladi. So'ngra bu protsedura Open usulini chaqirish bilan so'rovni bajarilishini aktivlashtiradi.

TForm1.Button2Click protsedurasi hamma yozuvlar tugmasini bosish orqali chaqirilib, DataSource1 komponentasini Table1 komponentasi bilan bog'laydi va butun bazani ko'rish rejimiga o'tishini ta'minlaydi.

Agar so'rov SQL xossasiga formani yaratish jarayonida yozilgan bo'lsa, dastur bajarilish jarayonida so'rov shartini so'rov matniga mos 5 qatorini almashtirish yo'li bilan o'zgartirish mumkin.

Masalan:

```

SELECT DISTINCT Fam, Name, Class FROM ``:Школа
:school.db`` WHERE
```

```

(Class= '10a') ORDER BY Name, Fam
```

so'rov matnini o'zgartiruvchi instruksiya quyidagi ko'rinishga ega bo'lishi mumkin:

```

form1.Query1.SQL[3]:='(Fam=`` + fam+ ``)'
```

Shuni e'tiborga olish lozimki, SQL xossasi TString tipidagi tuzilma bo'lib, qatorlar nomerlari noldan boshlanadi.

## Dinamik yaratuvchi psevdonimlar

Ma'lumotlar bazasiga murojaat uchun psevdonimdan foydalanish dasturning ma'lumotlar bazasini va dasturni har xil disklarda joylashtirishga imkon beradi. Shu bilan birga sodda ma'lumotlar bazalari dastur bilan birga bir katalogda joylashadi. Bunday holda BDE Administrator yordamida psevdonim yaratishdan voz kechib, dastur bajarilish jarayonida psevdonim yaratilishi mumkin.

Quyidagi dasturda «Maktab» ma'lumotlar bazasi bilan bog'lanish uchun dinamik yaratiluvchi psevdonimdan foydalaniladi.

### «Maktab» ma'lumotlar bazasi(psevdonim dinamik yaratiladi)

```
unit school3_;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,  
Dialogs, Grids, DBGrids, Db, DBTables, ExtCtrls, DBCtrls,  
StdCtrls;
```

```
type
```

```
TForm1 = class(TForm)
```

```
Table1: TTable; // таблица (вся база данных)
```

```
Query1: TQuery; // запрос (записи БД, удовлетворяющие  
критерию выбора)
```

```
DataSource1: TDataSource; // источник данных – таблица  
или запрос
```

```
DBGrid1: TDBGrid; // таблица для отображения БД или  
результата выполнения запроса
```

```
DBNavigator1: TDBNavigator;
```

```
DBText1: TDBText;
```

```
Button1: TButton; // кнопка запрос
```

```
Button2: TButton; // кнопка Все записи
```

```
procedure Button1Click(Sender: TObject);
```

```
procedure Button2Click(Sender: TObject);
```

```
procedure FormActivate(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```

end;
var
Form1: TForm1;
implementation
{$R *.DFM}
// шелчок на кнопке Запрос
procedure TForm1.Button1Click(Sender: TObject);
var
fam: string[30];
begin
fam:=InputBox('Выборка информации из БД',
'Укажите фамилию и щелкните на ОК.', '');
if fam <> «» // ползователь ввел фамилию
then
begin
with form1.Query1 do begin
Close; // закрыт файл-результат выполнения предыдущего
запроса
SQL.Clear; // удалит текст предыдущего запроса
// записываем новый запрос в свойство SQL
SQL.Add('SELECT Fam, Name, Class');
SQL.Add('FROM `:Школа:school.db`');
SQL.Add('WHERE');
SQL.Add('(Fam = `'+ fam + `)`');
SQL.Add(«ORDER BY Name, Fam»);
Open; // активизируем выполнения запроса
end;
if Query1.RecordCount <> 0 then
DataSource1.DataSet:=Query1 // отобразит рез-т выполнения
запроса
else begin
ShowMessage('В БД нет записей, удовлетворяющих критерию
запроса.');
```

```

DataSource1.DataSet:=Table1;
```

```

end;
```

```

end;
```

```

end;
```

```

// шелчок на кнопке Все записи
```

```

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

```

begin
```

```

DataSource1.DataSet:=Table1; // источник данных – таблица
```

```

end;
// активизация формы
procedure TForm1.FormActivate(Sender: TObject);
begin
with Session do
begin
ConfigMode := cmSession;
try
{Если файл данных находится в том же каталоге, что и
выполняемые файл Программы, то в Программе путь к файлу
данных может быть получен из командной строки при помощи
функции Extract File Path (Param Str(o)).
В приведенном примере файл данных находится в
подкаталоге DATA каталога Программы. }
// создадим временный псевдоним для базы данных
AddStandardAlias( `Школа`,
ExtractFilePath(ParamStr(0))+`DATA\`,
`PARADOX`);
Table1.Active:=True; // откроем базу данных
finally
ConfioMode := cmAll;
end;
end;
end;
end.

```

Bu dasturda ma'lumotlar bazasi dastur joylashgan katalogning DATA ostki katalogida joylashgan deb hisoblanadi. Psevdonim TForm1.FormActivate protsedurasida yaratiladi. Psevdonim yaratishni AddstandardAlias protsedurasiga bajaradi. Protседuraga parametr sifatida psevdonim nomi va unga mos katalog nomi beriladi. Katalog nomi ParamStr(0) va ExtractFilePatch funksiyalarga murojaat qilish yordamida aniqlanadi. Birinchi funksiya qiymati – dastur faylining to‘liq nomi, ikkinchisi shu faylga yo‘l. Shunday qilib – AddstandardAiias protsedurasiga ma'lumotlar bazasi katalogining to‘liq nomi beriladi.

## 9.8. DataSet muharriri

DataSet muharriri TTable va TQuery obyektlari yordamida chaqirilishi mumkin. Muharrir bilan ishlash uchun TQuery obyektini

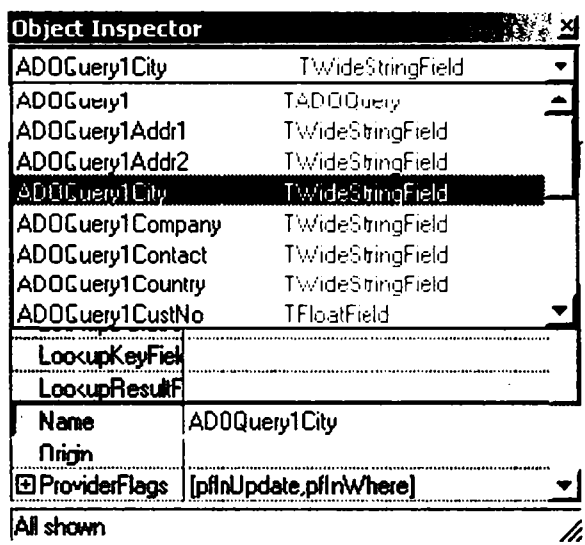


formaga joylashtirib, DBDEMOS psevdonimini o'rnatib, ``select \* from customer`` SQL so'rovini kiritib, Active xossasiga True qiymatini bering. Obyektlar inspektori yuqorisida ikki komponent: TForm i TQuery joylashgan.

TQuery obyektida sichqonchani o'ng klavishasini bosib va kontekstli menyuda «Fields Editor» punktini tanlab va **DataSet** muharririni ekranga chiqaring. Muharrir ekranida sichqonchani o'ng klavishasini

bosib va menyudan Add buyrug'ini tanlab – Add Fields dialog oynasi ekranga chiqadi.

Ko'zda tutilgani bo'yicha hamma maydonlar tanlangan. Hamma maydonni tanlash uchun OK tugmasini bosib va muharriri berkitib. Obyektlar inspektorida yangi obyektlar paydo bo'ladi.



Bu yangi obyektlar CUSTOMER jadvalini tasvirlaydi.

Query1CustNo: TFloatField;  
 Query1Company: TStringField;  
 Query1Addr1: TStringField;

```
Query1Addr2: TStringField;  
Query1City: TStringField;  
Query1State: TStringField;  
Query1Zip: TStringField;  
Query1Country: TStringField;  
Query1Phone: TStringField;  
Query1FAX: TStringField;  
Query1TaxRate: TFloatField;  
Query1Contact: TStringField;
```

Agar biz Query1 obyektini Customer deb o'zgartirsak quyidagi nomlar hosil bo'ladi:

```
CustomerCustNo  
CustomerCompany
```

Har bir yangi yaratilgan obyekt TField sinfining avlodi hisoblanadi. Ajdodning turi ma'lumotlar turiga bog'liq. Masalan, CustNo maydonining turi TFloatField, Query1City tipi TStringField.

Bu sinflarning eng asosiy xossasi Value deb ataladi. Bu xossaga quyidagicha murojaat qilish mumkin:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
d: Double;  
S: string;  
begin  
d := Query1CustNo.Value;  
S := Query1Company.Value;  
d:=d+1;  
S := `Zoo`;  
Query1CustNo.Value := d;  
Query1Company.Value := S;  
end;
```

Value qiymatining turi har doim o'zi tegishli bo'lgan maydon turiga mos keladi. Masalan, TStringFields uchun – string, TCurrencyFields uchun – double.

Agar joriy DataSetda maydonlar nomini olish lozim bo'lsa, FieldName xossasidan ikki usulda foydalanish lozim:

```
S := Query1.Fields[0].FieldName;  
S := Query1CustNo.FieldName;
```

Agar maydon bilan bog'liq obyekt nomini olish lozim bo'lsa, Name xossasidan foydalanish lozim:



S := Query1.Fields[0].Name;

S := Query1CustNo.Name;

CUSTOMER jadvali uchun birinchi misol ``CustNo`` qatorini, ikkinchi misol ``Query1CustNo`` qatorini qaytaradi.

## Hisoblanuvchi maydonlar

Hisoblanuvchi maydonlar yaratish DataSet muharririning asosiy avfzalliklaridan biridir. Quyida shunday maydon yaratishga misol ko‘ramiz. Formaga Query, DataSource, DBGrid obyektlarini joylashtirib, DBDEMOS psevdonimini o‘rming.

Query1 uchun SQL xossasiga quyidagi tekstni kiriting:

```
select * from Items I, Parts P  
where (I.PartNo=P.PartNo)
```

Query1 obyektini aktivlashtirib, DataSet (Fields Editor) muharririni chaqiring, Add Fields oynasidan OrderNo(buyurtma nomeri), PartNo(tovar nomeri), Qty(son) va ListPrice(narx) maydonlarini qo‘shing.

Muharrir ekranida sichqonchanning o‘ng klavishasini bosib va menyudan NewField buyrug‘ini tanlab, dialog oynasini ekranga chiqaring:

**New Field** [X]

**Field properties**

Name: [ ] Component: [ ]

Type: [ ] Size: [ ]

**Field type**

Data  Calculated  Lookup

**Lookup definition**

Key Fields: [ ] Dataset: [ ]

Lookup Keys: [ ] Result Field: [ ]

OK Cancel Help

Name qatoriga Total so‘zini kiriting. Type xossasiga CurrencyField qiymatini bering. Calculated tanlanganligini t kshiring. Ok tugmasini bosib, DataSet muharririni berkiting.

Hisoblanuvchi maydon yaratish uchun obyektlar inspektoridan Query1 uchun hodisalar (Events) ro'yxatini oching va OnCalcFields qatoriga ikki marta cherting. Usulni quyidagicha to'ldiring:

```
procedure TForm1. Query1 CalcFields(DataSet: TDataset);
begin
  Query1 Total.Value:= Query1 Qty.Value* Query1 ListPrice.Value;
end;
```

Agar dasturni ishga tushirsangiz Total maydoni kerakli qiymatga ega bo'ladi.

Form1

OrderNo	I.PartNo	Qty	ListPrice	Total
1005	900	10	3999.95	39 999.50p.
1020	900	4	3999.95	15 999.80p.
1024	900	3	3999.95	11 999.85p.
1027	900	8	3999.95	31 999.60p.
1034	900	8	3999.95	31 999.60p.
1043	900	4	3999.95	15 999.80p.
1047	900	7	3999.95	27 999.65p.

Total maydonidagi hamma qiymatlar yig'indisini quyidagicha hisoblash mumkin:

```
procedure TForm1.Button1Click (Sender: TObject);
var
  R : Double;
begin
  R:=0;
  with Query1 do begin
    DisableControls;
    Close;
    Open;
    repeat
      R:=R+ Query1 Total.Value;
    Next;
  until EOF;
  First;
```

EnableControls;

end;

end;

*DisableControls* usuli DBGridni qayta chizishni man qilish uchun chaqiriladi.

## 9.9.MB bilan ishlashda Delphi grafikasi

MBning jadvali ma'lumotlari asosida grafiklar qurish uchun Delphi komponentalar politrasing DataControls sahifasidagi TDBChart komponentasi ishlatiladi. Bu komponenta yordamida grafiklar yaratish uchun ma'lumotlar manbai, ya'ni MB yaratilgan bo'lishi zarur. Grafik yoki gistogrammalar yaratish ketma-ketligi quyidagi qadamlarda bajariladi:

1. Ishlab chiqilgan MBning loyiha formasi ekranga chiqariladi.

2.TDBChart komponenta formasi asosiy formaga joylashtiriladi. Buning uchun TDBChart piktogrammasi sichqonchada ko'rsatilib ikki marta chiqillatiladi.

3.Grafik muharriri chaqiriladi. Forma sarlavhasi ustiga sichqoncha ko'rsatgichini olib kelib ikki marta tez-tez chiqillatiladi yoki o'ng sichqoncha tugmachasi bosilib lokal menyudan Edit Chart buyrug'i beriladi. Edit Chart muharririga quyidagi parametrlarni o'rnatish mumkin, ularning tavsifi quyidagicha:

Series – bir necha grafiklar variantlarini tavsiya etadi;

General – umumiy parametrlarni o'rnatish mumkin, masalan, grafik o'lchamini kattalatish (kichraytish);

Axis – grafik koordinata o'qlarini aniqlaydi;

Show Axis – chap, o'ng, past va yuqori o'qlarni tanlaydi;

Scales – koordinata o'qi masshtabining xossalari qiymatini aniqlaydi;

Automatic – ma'lumotlarni avtomatik masshtablashtiradi;

Title – o'qlar bo'yicha matn yozishni aniqlab, uning joylashishini va shriftlarini aniqlaydi;

Titles – grafik sarlavhasi matnini yozishni aniqlab, uning joylashishi va shriftlarini aniqlaydi;

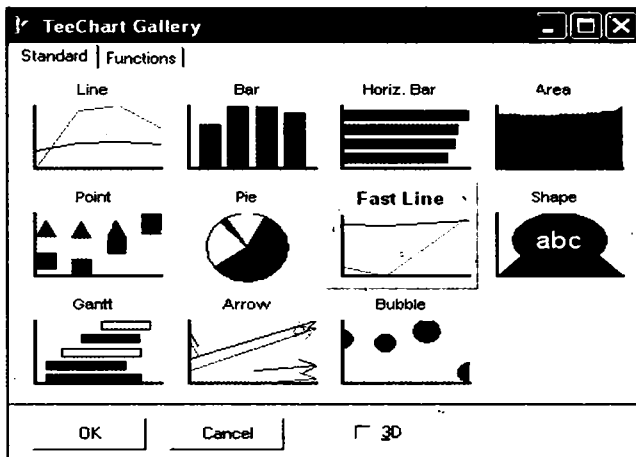
Ligent – grafiklarga tushuntirish ma'lumotlarini beradi;

Panel – parametrlar panelini joylashtiradi;

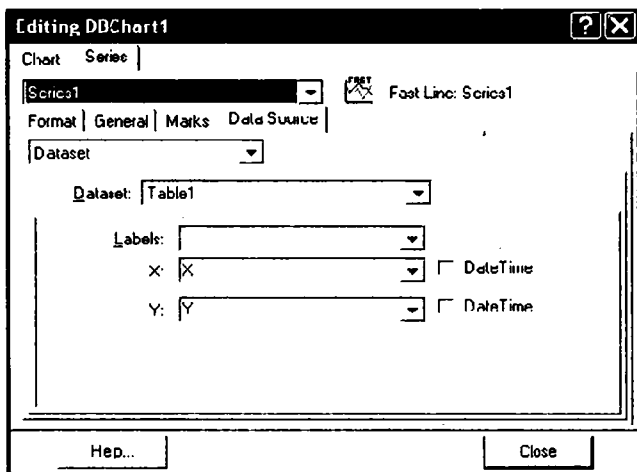
Paging – ko'pqatorli grafikning parametrlarini aniqlaydi.

4.Grafik variantlarini chiqarish va tanlash. Kerakli variantni tanlash Chart sahifasidagi Series bo'limining ADD tugmachasini sichqonchada chiqillatish bilan amalga oshiriladi. Bu oynadagi boshqa tugmachalarning vazifalari:

- Delete – tanlangan joriy variantni o‘chiradi;
- Title – har bir tanlangan variantga sarlavha qo‘yishni bajaradi;
- Clone – grafikdan nusxa tayyorlaydi;
- Change – joriy variantning turini o‘zgartiradi.



5. Grafik qurish uchun ma'lumotlar manbai tanlanadi. Buning uchun tanlangan variant ikki marta sichqonchada chiqillatiladi va u yerdan DataSource sahifasiga kirilib, DataSet qatoridan Table1 jadvali tanlanadi. Keyin Tables belgisidan (metkasidan) kerakli maydon olinadi va x, y bo'yicha koordinata o'qlari aniqlanadi. Close tugmachasi bosilib, oyna yopiladi.



6. Formaga grafik joylashtiriladi.

**M i s o l 1.** Funksiyaning jadval qiymatlari yordamida uning grafigini quring. Argument  $x$  va funksiya  $y$  qiymatlari quyidagi jadvalda berilgan.

X	2	3	4	5	6	7	8	9	10	11
Y	10	15	19	22	25	29	33	38	45	55

### Y e c h i s h

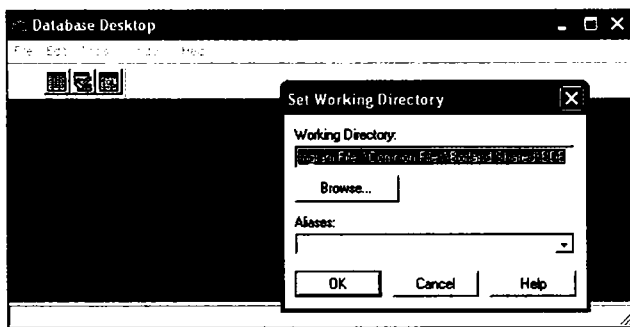
Argument « $x$ » va funksiya « $y$ » nomli maydonlardan iborat MBni tuzamiz:

1.DataBase DeskTop utilitasini ishga tushiramiz.

Пуск=>Программы=>Borland Delphi=>DataBase DeskTop

2.DataBase DeskTop oynasining bosh menyusidan quyidagi buyruqni beramiz. **File=>Working Directory**

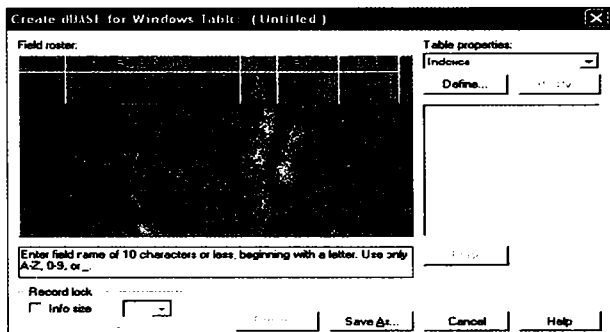
3.Hosil bo'lgan Set Working Directory muloqot darchasining Aliasesidan MB psevdonmini aniqlaymiz.



4.Menyudan File=>New=>Table buyrug'ini beramiz.

5.Hosil bo'lgan Create Table muloqot darchasidan MB turini aniqlaymiz.

6.MB tuzilmasini tashkil qilamiz.



7. Save As buyrug'ini berib, jadval nomini «fun» deb kiritamiz va Ok tugmasini bosamiz.

8. Agar MBga ma'lumotlar kiritish kerak bo'lsa DataBase DeskTop oynasi menyusidan File=>Open buyrug'ini berib, keyin faylni tanlab va Edit Data tugmachasini bosib ma'lumotlarni kiritamiz.

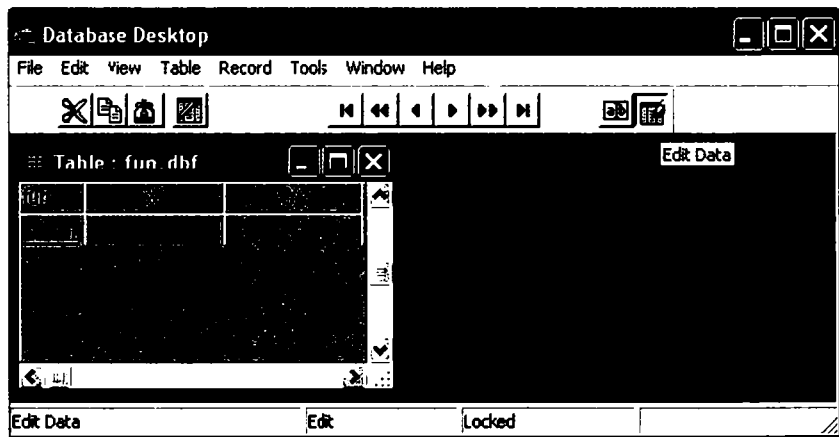


Table oynasida x va y maydonlariga misolda keltirilgan ularning qiymatlarini kiritamiz. Bu qiymatlarni ilova tuzilgandan so'ng kiritisa ham bo'ladi.

Endi tuzilgan MBni boshqarish uchun ilova yaratishga kirishamiz:

1. Delphi tizimini ishga tushirib BDE komponentalar palitrasidan Ttable komponentasini formaga joylashtiramiz.

2. Formadagi Ttable komponentasi uchun DataBase Name xossasida Mbning psevdonimini aniqlaymiz.

3. TableName xossasidan MB jadvalining nomini aniqlaymiz.

4. Active xossasini True qiymati bilan o'rnatamiz.

5. Data Access komponentalar palitrasidan TdataSource komponentasini formaga qo'yamiz.

6. TdataSource xossasini Table1 nomi bilan o'rnatamiz.

7. Data Controls komponentalar palitrasidan TDbGrid komponentasini formaga qo'yamiz.

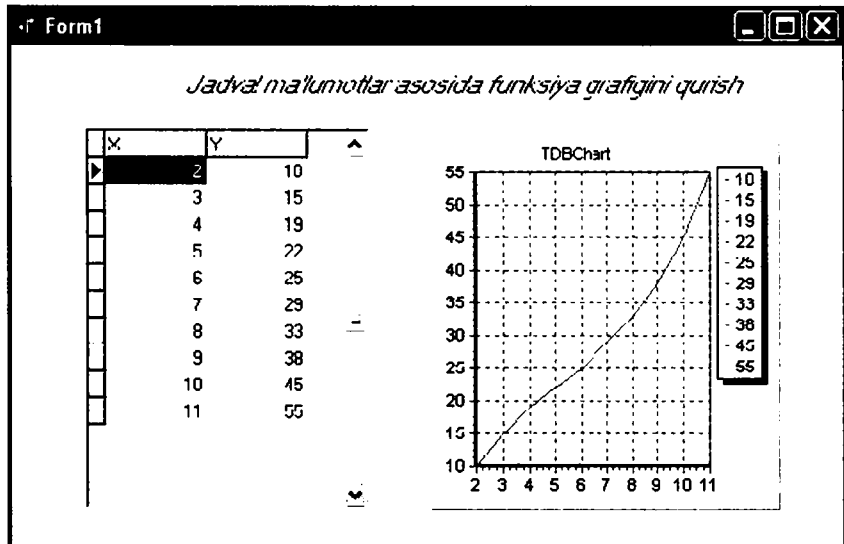
8. DataSource xossasini DataSource1 nomi bilan o'rnatamiz.

9. Label1ga forma sarlavhasini yozamiz.

10. Formaga grafikni joylash uchun yuqorida keltirilgan grafikni qurish algoritmidan foydalanamiz.

11. Menyudan File=>Save Project As buyrug'ini berib, oldin forma keyin loyihani saqlaymiz.

12. Loyihani ishga tushirish uchun F9 tugmasini bosamiz. Natijada, quyidagi formaga ega bo'lamiz.



Bu oynada berilgan funksiyaning jadval qiymatlari o'zgarsa grafik ham mos ravishda o'zgaradi.

Kompyuter quyidagi dastur kodlarini avtomatik ravishda tuzadi:

Unit Funk;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, DBGrids, DB, DBTables, StdCtrls, TeEngine, Series,

ExtCtrls, TeeProcs, Chart, DbChart;

type

TForm1 = class(TForm)

Label1: TLabel;

Table1: TTable;

DataSource1: TDataSource;

DBGrid1: TDBGrid;

DBChart1: TDBChart;

Series1: TFastLineSeries;

```

private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
implementation
  {$R *.dfm}
end.

```

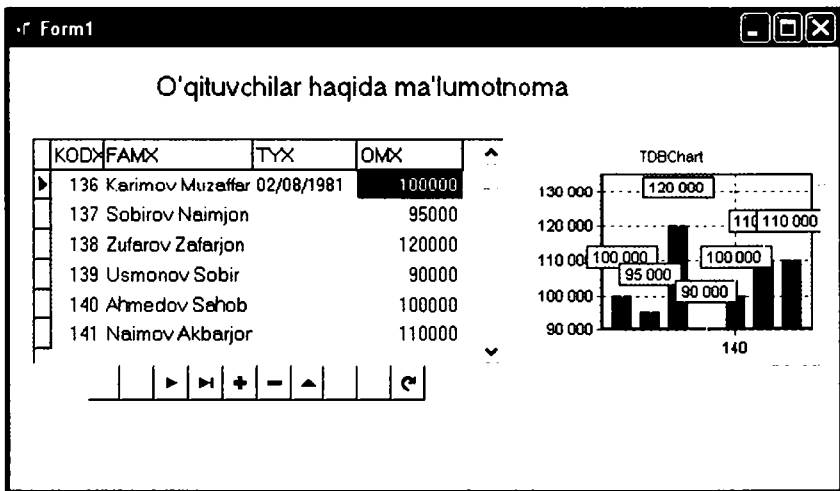
**Misol 2.** Yuqoridagi o'qituvchilar haqidagi MBdan foydalanib, o'qituvchilar ish haqining gistogrammasini quring.

### Yechish

1. Yuqoridagi o'qituvchilar haqidagi MB uchun tuzilgan ilova loyahasini ekranga yuklaymiz.

2. Formaga grafikni joylash uchun yuqorida keltirilgan grafikni qurish algoritmidan foydalanamiz.

3. Loyihani ishga tushiramiz va natijada quyidagi ishchi formaga ega bo'lamiz.



Formadagi baza ma'lumotlarining o'zgarishi mos ravishda grafik o'zgarishlariga ham olib keladi.



## Savollar

1. Delphi MBni boshqarishda qanday vositalarni ishlatadi?
2. BDE nima va u qanday ishlarni bajaradi?
3. DBE administrator utilitasi vazifasini tushuntirib bering.
4. MB bilan ishlash uchun vizual bo'lmagan komponentalar haqida ma'lumotlar bering.
5. MB bilan ishlash uchun vizual komponentalar haqida ma'lumotlar bering.
6. MBni psevdonimi nima va uni tuzish ketma-ketligini aytib bering.
7. MB jadvali tuzilmasi qanday ketma-ketlikda bajariladi?
8. Maydon nima va u qanday elementlardan iborat?
9. MB jadvali bilan ishlash uchun oddiy ilova yaratish qanday ketma-ketlikda bajariladi?
10. Ma'lumotni izlashda qanday usullar mavjud?
11. Ma'lumotlarni filtrlashning qanday usullari bor?
12. Ma'lumotlar bazasiga so'rovlar qanday tashkil qilinadi va qanday buyruqlar ishlatiladi?

## ADABIYOTLAR:

1. *А.Файсман*. Профессиональное программирование на Турбо Паскале. 1992.
2. *М.В.Култин*. Программирование в Турбо Паскале и Делпхи, Санкт-Петербурге, 2002.
3. *С.Р. Кондзуба, В.И. Оромов*. Делпхи 6/7. База данных и приложения. М.— Санкт-Петербург — Киев, 2002.
4. [WWW.Intuit.ru](http://WWW.Intuit.ru). Интернет-Университет информационных технологий. Москва.

Kirish.....	3
-------------	---

## **I. Asosiy tushunchalar**

1.1.Algoritm va dastur tushunchasi.....	5
1.2.Dasturlash tilining elementlari.....	7
1.3.O'zgarmlar, o'zgaruvchilar va standart funksiyalar.....	8
1.4.Ma'lumotlar turlari.....	11

## **II. Operatorlar, protsedura va funksiyalar**

2.1.Ma'lumotlarni kiritish va chiqarish operatorlari.....	18
2.2.Delphining konsol ilovasini yaratish.....	20
2.3.Shartli o'tish operatori.....	23
2.4.Shartsiz o'tish va tanlash operatorlari.....	25
2.5.Sikl operatorlari.....	27
2.6.Massivlar.....	30
2.7.Qism dasturlari.....	34
2.8.Modullar.....	38
2.9.Fayllar bilan ishlash funksiyasi va protseduralari.....	44

## **III. Delphi vizual dasturlash muhiti haqida asosiy tushunchalar**

3.1.Delphi dasturlash muhiti.....	49
3.2.Delphi tizimining oynasi va uning elementlari.....	50
3.3.Delphi loyihasining tuzilmasi.....	52
3.4.Sinflar va obyektlar.....	54
3.5.Vizual komponentalar bibliotekasi.....	63
3.6.VCL tarkibiga kiruvchi sinflar usullari.....	64
3.7.Delphining forma komponentalari.....	67
3.8.Asosiy xossalr va hodisalar.....	69

#### **IV. Delphi vizual dasturlash muhitida komponentalar bilan ishlash texnologiyalari**

4.1. Label, Edit, Memo matn komponentalari va Button tugmachasi.....	73
4.2. Boshlang'ich forma ilovasini yaratish.....	75
4.3. Tanlash tugmalarini o'rnatish.....	82
4.4. ListBox va ComboBox komponentalari.....	87
4.5. StringGrid jadval komponentasi.....	94
4.6. Muloqot oynalarini yaratish.....	97
4.7. Ilovalar uchun menyu yaratish.....	107
4.8. Bir necha formalar bilan ishlash.....	113

#### **V. Delphi muhitida grafika va multimedia**

5.1. Delphi ning grafik imkoniyatlari.....	118
5.2. Grafik komponentalar.....	131
5.3. Delphi ning multimedia imkoniyatlari.....	137
5.4. Bosmaga chiqarish.....	147

#### **VI. Delphi tilining imkoniyatlari**

6.1. Yozuvlarni faylga yozish va fayldan o'qish.....	151
6.2. Dinamik tuzilmalar.....	158
6.3. Rekursiya.....	167
6.4. Graflarga rekursiyani qo'llash. Yo'l izlash.....	177

#### **VII. Delphi qo'shimcha komponentalari**

7.1. ADDITIONAL sahifasining komponentalari.....	184
7.2. WIN 32-sahifasining komponentalari.....	191
7.3. TTreeView va TListView komponentalari.....	203

#### **VIII. Ma'lumotlar bazasining nazariy asoslari**

8.1. Ma'lumotlar bazasi haqida asosiy tushunchalar.....	213
8.2. Ma'lumotlar modellari.....	221
8.3. Ma'lumotlar bazasini loyihalashtirish.....	230

#### **IX. Delphi vizual dasturlash vositasida ma'lumotlar bazasini yaratish texnologiyalari**

9.1. MBni boshqaradigan ilovalar tuzish uchun Delphi vositalari.....	238
9.2. MB bilan ishlash uchun Delphi komponentalari.....	240

9.3.BDE administrator utilitasi bilan ishlash.....	241
9.4.MB jadvalini tuzish.....	243
9.5.MB jadvali uchun oddiy dastur ilovalarini tuzish.....	246
9.6.Ma'lumotlarni izlash va filtrlash.....	250
9.7.So'rovlar hosil qilish.....	255
9.8.DataSet muharriri.....	262
9.9.MB bilan ishlashda Delphi grafikasi.....	267
Adabiyotlar.....	274

*O'quv qo'llanma*

Sh.A. Nazirov, M.M. Musayev,  
A.N. Ne'matov, R.V. Qobulov

**DELPHI TILIDA DASTURLASH ASOSLARI**

Kasb-hunar kollejlari uchun o'quv qo'llanma

Muharrir *Mavjuda Nasriddinova*

Musavir *Anatoliy Bobrov*

Badiiy muharrir *Rustam Zufarov*

Texnik muharrir *Tatyana Smirnova*

Musahhih *Dono To'ychiyeva*

Kompyuterda sahifalovchi *Akmal Sulaymorov*

IB № 4472

Bosishga 11.07.07- y.da ruxsat etildi. Bichimi 60x90<sup>1</sup>/<sub>16</sub>.  
Tayms garniturası. Ofset bosma. 17,5 shartli bosma toboq.  
19,0 nashr tobog'i. Jami 5048 nusxa. 239 raqamli buyurtma.  
8-2007 raqamli shartnoma. Bahosi shartnoma asosida.

O'zbekiston Matbuot va axborot agentligining  
G'afur G'ulom nomidagi nashriyot-matbaa ijodiy uyi.  
100129. Toshkent, Navoiy ko'chasi 30.  
100128. Toshkent, Usmon Yusupov ko'chasi, 86.

**Bizning internet manzilimiz: [www.iptdgulom.uz](http://www.iptdgulom.uz)**

Delphi tilida dasturlash asoslari: Kasb-hunar kollejlari uchun o'quv qo'll./ Tuzuvchilar: Sh.A.Nazirov va boshq.-T.: G'afur G'ulom nomidagi nashriyot-matbaa ijodiy uyi, 2007. — 280 b.

I. Nazirov Sh.A.

Oxirgi yillarda dasturlashga bo'lgan qiziqish tobora ortib bormoqda. Bu kompyuter texnologiyasining kun sayin rivojlanib borishi bilan bog'liqdir. Ayniqsa, vizual dasturlash texnologiyalaridan foydalanib dasturlar yaratish kompyuter texnologiyasining rivojlanishiga katta ta'sir etmoqda.

Ushbu o'quv qo'llanmada Pascal tili va Delphi dasturlash vositasida vizual dasturlash texnologiyalari haqida o'quvchilar to'liq tasavvurga ega bo'ladigan barcha kerakli ma'lumotlar berilgan.

O'quv qo'llanma kollej o'quvchilari, oliy texnika o'quv yurtlari talabalari, o'qituvchilar va kursni mustaqil o'rganuvchilar uchun mo'ljallangan.

**BBK 32.973.202-018.1я722**