O’zbekistan Respublikasi baylanis, informatsiyalastiriw ha’m telekommunikatsiya texnologiyalari ma’mleketlik komiteti.
Tashkent informatsiyaliq texnologiyalari universiteti.
No’kis filiali.
Kompyuter injiniringi fakulteti.
Kompyuter injiniringi bag’dari.

C++ da programmalastiriw
pa’ninen tayarlang’an

Kurs jumisi

Tema: *Obektga yo’naltirilgan dasturlash tillari. C++ va Borloand C++Builder*

Ori’nlag’an: ***Allambergenov R***

Qabi’llag’an: ***Yadgarov Sh***

No’kis 2015-jil

Obektga yo’naltirilgan dasturlash tillari. C++ va Borloand C++Builder

Reja:

Kirish

1. *Asosiy bo’lim*

a) Komponentlar, xodisalar va xossalar

b) Turlar va C++ da o’zgaruvchilarni tavsiflash

c) Turlarni o’zgartirish protseduralari

d) Boshqaruvchi strukturalar

2. Xulosa

3. Amaliy ish

4. Foydalanilgan adabiyotlar

Kirish

Xozirgi vaqtga kelib komp`yuter olamida ko’plab dasturlash tillari mavjud. Paskal, C++ va boshqa dasturlash tillaridir. C++ dasturlash tili universal tildir. U UNIX sistemasi bilan bog`langan bo`lib, bu sistemada ishlatiladigan bir qancha dasturlar C++ tilida yozilgan. Paskal tili 1969 yil N. Virt tomonidan yaratilgan bo’lib, keyinchalik amerikaning Borland firmasi tomonidan qayta ishlandi va uni Turbo Pascal deb nomlangan. C++ Denis Ritchi tomonidan 1972 yili UNIX tipidagi operasion sistemalarini yaratish uchun loyihalashtirilgan.

Turbo Pascal ni qayta ishlash natijasida ob'ektli dasturlash yo’lga qo’yildi va 1995 yilda Borland kompaniyasi guruxi dastur tuzuvchilari Chack va Denny tomonidan Windows uchun mo’ljallangan dasturlash muxiti Borland Delphi dasturlash vositasi yaratildi.

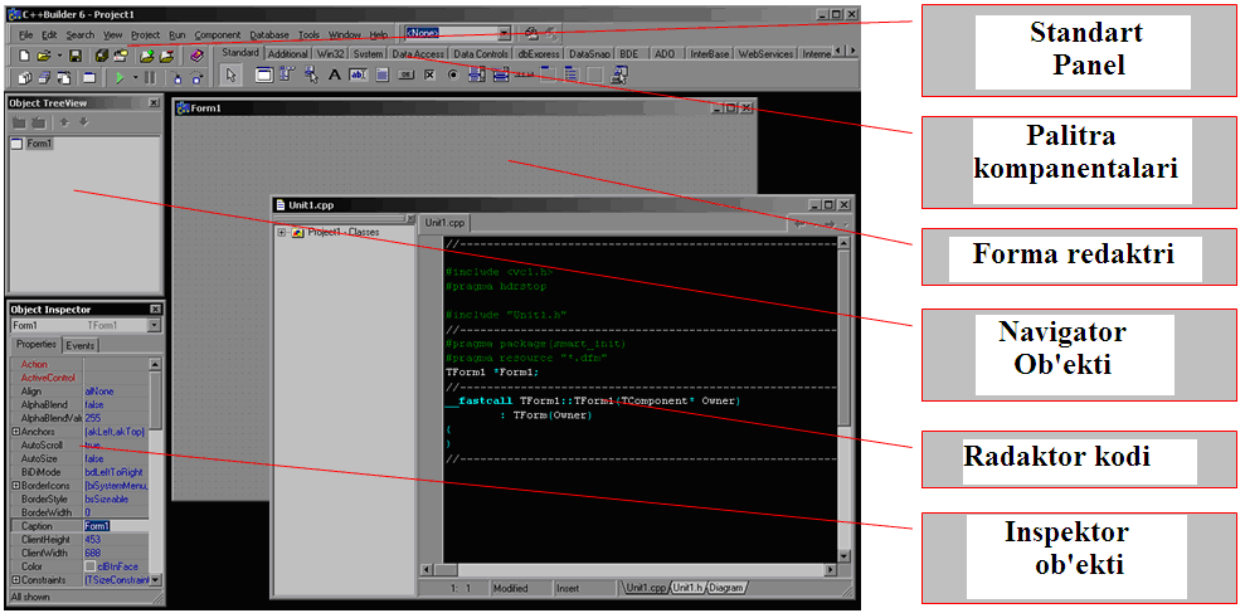
Borland C++ va Delphi dasturlash tili Windows uchun mo’ljallangan bo’lib, uning birinchi versiyasi Windows operatsion sistema qobig’ida ishlagan.

Borland C++ va Delphi dasturlash tili – bu dasturlarni qayta ishlash muxiti bo’lib, Windows operatsion sistemasida ishlaydi. Unda ob'ektli dasturlash tillari bo’lgan Object mujassamlashgan.

Borland C++ va Delphi vizual proektlar, turli xolat protseduralarini qayta ishlash va dasturlarni qayta ishlashda vaqtdan yutish va boshqalarni o’z ichiga oladi.

Dastur yaratish muhiti

Dastur yaratish umumlashgan muhiti Redaktor form – Shakllar muharriri, Inspektor ob’ektov – Ob’ektlar inspektori, Palitra komponentov – Komponentlar palitrasi, Administrator proekta – Proekt administratori va to’la umumlashgan Redaktor koda – Kodlar muharriri hamda kodlar va resurslar ustidan to’liq nazoratni ta’minlaydigan , dastur ilovalarini tezkor yaratadigan Otladchik - instrumentov - Sozlash-instrumentlari kabilarni birlashtiradi.



a) Komponentlar

Komponentalarni shaklga o'rnatish uchun komponentlar palitrasidagi kerakli piktogramma tanlanadi, so'ngra shaklning komponenta joylanishi kerak bo'lgan joyi tanlanadi. Shundan so'ng komponentalar xossalarini ob'ektlar inspektori yordamida tahrirlash mumkin. Properties bandida komponentalar xossalarining ro'yxati (chapda) va bu xossalarning qiymatlar ro'yxati (o'nggda) joylashgan.

Komponentalar ko'rinadigan (vizual) va ko'rinmaydigan (vizual bo'lmagan) larga bo'linadi. Vizual komponentalar bajarilish paytida proektlash paytidagidek paydo bo'ladi. Bunga knopkalar va tahrirlanuvchi maydonlar misol bo'la oladi. Vizual bo'lmagan komponentalar proektlan vaqtida shakldagi piktogramma ko'rinishida paydo bo'ladi. Ular bajarilish paytida hech qachon ko'rinmaydi, ammo ma'lum funksionallikga ega bo'ladi (masalan, berilganlarga murojatni ta'minlaydi, Windowsning standart muloqatlarini chaqiradi).

Xossalar

Xossalar komponentalarning tashqi ko'rinishi va tabiatini aniqlovchi atributlar hisoblanadi. Xossalar ustunidagi ko'p xossalar komponentalari oldindan o'rnatilgan (po umolchaniyu) qiymatlarga ega bo'ladi (masalan, knopkplar balandligi). Komponentalar xossalari xossalar varag'i (Properties) da aks ettiriladi. Ob'ektlar inspektori komponentalarning nashr etilgan (published) xossalarini aks

ettiriladi. published-xossalardan tashqari komponentalar umumiy (public), faqat ilovalarning bajarilish paytidagina murojat qilish mumkin bo'lgan nashr qilingan xossalarga ega bo'ladi. Xossalar ro'yxati ob'ektlar inspektori xossalar varag'ida joylahadi. Xossalarni proektlash paytida aniqlash mumkin yoki ilovalarning bajarilish paytida ko'rinishini o'zgartirish uchun kod yozish mumkin. Komponenta xossalarini proektlash paytida aniqlash uchun shakldagi komponenta tanlanadi, ob'ektlar inspektori xossalari varag'i ochiladi, aniqlanadigan xossa tanlanadi va zurur bo'lsa xossalar muharriri yordamida o'zgartiriladi (bu kiritish uchun oddiy maydon yoki son, osilib tushuvchi ro'yxat, ochiluvchi ro'yxat, muloqat paneli va boshqalar bo'lishi mumkin).

Biror komponentaning xossalarini dasturning bajarilish paytida o'zgartirish uchun «Imya Komponenta» –> «Nazvanie svoystva» tavsifiga o'zaruvchidek murojat qilish kerak, ya'ni qiymatlarni o'zimiz hohlagandek o'qishimiz yoki almashtirishimiz mumkin.

Xodisalar

Ob'ektlar inspektorining xodisalar varag'i (Events) komponentalar tomonidan taniladigan xodisalar ro'yxatini ko'rsatadi. Har bir komnonenta o'zining shaxsiy xodisalarni qayta ishlovchi naborga ega bo'ladi. C++ Builder da xodisalarni qayta ishlovchi funksiyalarni yozish va xodisalarni bu funksiya bilan bog'lashga to'g'ri keladi. Biror bir xodisaga qayta ishlovchi yozib, siz dasturga bu xodisa ro'y berganda yozilgan funksiyaning bajarilishini topshirasiz.

Xodisani qayta ishlovchini qo'shish uchun shaklda xodisani qayta ishlovchi komponenta tanlanadi. So'ngra xodisalar varag'ida ob'ektlar inspektori ochilib (Event bandi) xodisaning qatoridagi qiymatlar ustunida sichqonning chap tugmasi ikki marta bosiladi. Bu bilan C++ Builder ni xodisalarni qayta ishlash prototipini generatsiya qilishga va uni kodlar muharririda ko'rinishiga majbur qiladi. Bu holda bo'sh funksiya nomi generatsiya qilinadi va muharrir kod kiritilishi zarur bo'lgan joyda ochiladi. Kursor buyruqlar qavslari ichiga joylashadi { … }. So'ngra xodisa

sodir bo’lganda bajarilishi kerak bo’lgan kod kiritiladi. Xodisalarni qayta ishlovchi funksiya nomidan keyin ko’rsatiladigan parametrlarga ega bo’lishi mumkin.

Quyida xodisalarni qayta ishlovchi protseduraning shunday bo’sh karkasi ko’rsatilgan:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{

}
```

b) Turlar va C++ da o’zgaruvchilarni tavsiflash

Har bir nom va har bir o’zgaruvchi ular ustida bajariluvchi amallar aniqlovchi turlarga ega bo’ladi. Masalan, **int i;** tavsiflash i o’zgaruvchi int turiga tegishli, ya’ni i butun o’zgaruvchi deb aniqlaydi. Tavsiflash - dasturga nom kirituvchi buyruqdir. Tavsiflash o’zgaruvchining turini aniqlaydi. Tur nom va ifodalardan to’g’ri foydalanishni aniqlaydi. Butun tur uchun quyidagi amallar aniqlangan: +, -, * va /.

Asosiy turlar

Bevosita apparat ta’minotiga javob beradigan asosiy turlar quyidagilar: char; short; int; long; float; double. Birinchi to’rtta tur butun kattaliklarni, oxirgi ikkitasi suzuvchi nuqtali, ya’ni kasr sonlarni tasvirlash uchun ishlatiladi.

char turidagi o’zgaruvchi mazkur kompyuterda belgilarni (odatda bayt) saqlash o’lchoviga ega, int turidagi o’zgaruvchi esa mazkur kompyuterdagi butun arifmetikaga mos o’lchovga ega (odatda so’z). Turlar bilan tasvirlangan butun sonlar diapazoni uning o’lchoviga bog’liq bo’ladi (uni sizeof buyrug’i yordamida hisoblash mumkin).

C++ da o’lchovlar char turidagi kattaliklar o’lchovi birligida o’lchanadi. Asosiy turlar o’rtasidagi munosabatlarni quyidagicha yozish mumkin:

1 = sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) = sizeof(float) <= sizeof(double).

Umuman, asosiy turlar xususida yana boshqa narsalarni faraz qilish ma’nosiz. Xususan, ko’rsatgichlarni saqlash uchun butun tur etarli, degan xulosa barcha kompyuterlar uchun to’g’ri emas. Asosiy turlarga const so’zini qo’shib tavsiflash mumkin. Bu boshlang’ich turga shu turning o’zini beradi, faqat bu holatda const turidagi o’zgaruvchilarning qiymatlari initsializatsiyadan so’ng o’zgarishi mumkin emas.

const float pi = 3.14;

const char plus = '+';

Bittalik qo’shtirnoqqa olingan belgilar belgi o’zgarmaslar hisoblanadi. Shunga e’tibor berish lozimki, bu usulda tavsiflangan o’zgarmaslar xotirada joy egallamaydi. uning qiymati talab qilingan joyda bevosita ishlatiladi. O’zgarmaslar initsializatsiya paytida tavsiflanishi shart. O’zgaruvchilar uchun initsializatsiya shartemas, ammo albatta tavsiya qilinadi. Lokal o’zgaruvchilarni initsializatsiyasiz kiritish asoslari juda ko’p.

Bu turlarning ixtiyoriy kombinatsiyasiga quyidagi arifmetik amallar qo’llanilishi mumkin:

+ (plyus, unar va binar);

- (minus, unar va binar);

* (ko’paytirish);

/ (bo’lish).

Hamda taqqoslash amallari:

== (teng);

!= (teng emas);

< (kichik);

> (katta);

<= (kichik yoki teng);

>= (katta yoki teng).

Agar operandlar qo’yilgan shartni qanoatlantirsa , u holda taqqoslash amallari natijada 1 qiymatni beradi, aks holda esa 0 qiymatni beradi.

Butunga bo’lish amali butun natijani beradi: 7/2 = 3. Butun kattaliklar ustida % - qoldiqni hisoblash amali bajariladi: 7%2 = 1.

O’zlashtirishda va arifmetik amallarda C++ ularni guruhlash uchun asosiy turlar o’rtasida barcha ma’noli almashtirishlarni bajaradi:

```
double d = 1;
int i = 1;
d = d + i;
i = d + i;
```

Satriy turlar

C++ da belgilarning biron-bir ketma-ketligi (massivlar) dan iborat matn qatorlarini xotirada saqlash uchun maxsus AnsiString ma’lumotlar turi qo’llaniladi.

«Stroka» - «Satr» turidagi o’zgaruvchilar barcha boshqa o’zgaruvchilar kabi e’lon va initsializatsiya qilinadi.

Kompilyatorga navbatdagi belgilar ketma-ketligi yangi o’zgaruvchining nomi emas, balki satr ekanligini bildirish uchun satrlar bittalik qo’shtirnoq ichiga olinadi.

Misol:

```
AnsiString st = ‘matn qatori’;
```

Satr turidagi o’zgaruvchilar ustida boshqa satr o’zgaruvchilar bilan qo’shish amali bajarilishi mumkin. Bu amal ikkita satrni ularning kelish tartibida birlashtirish deb tushuniladi.

Misol:

```
AnsiString s1 = ‘qatori’;
AnsiString s2 = ‘ matn’;
AnsiString s = s1 + s2;
```

Natijada s o’zgaruvchi s1 va s2 o’zgaruvchilardan tashkil topgan ‘stroka teksta’ degan qiymatni qabul qiladi.

Qo’shimcha turlar

Borland C++ da butun qiymatli o’zgaruvchilarning turlarini qo’shimcha ajratish imkoni mavjud. Bu holda o’zgaruvchilarning barcha tur nomlari quyidagicha yoziladi - int X, bu erda X o’zgaruvchiining bitlardagi maydon o’lchami. X quyidagi qiymatlardan birini qabul qilishi mumkin: 8, 16, 32 va 64. Bu turdagi o’zgaruvchilardan foydalanish standart turda aniqlangan o’zgaruvchilardan foydalanishdan farq qilmaydi.

Quyidagi jadvalda bunday turlar bilan ishlash yaqqol ko’rsatilgan.

Tur nomi	**O’zgaruvchini tavsiflashga misol**	**O’lcham**
__int8	__int8 c = 128;	8 bit
__int16	__int16 s = 32767;	16 bit
__int32	__int32 i = 123456789;	32 bit
__int64	__int64 big = 12345654321;	64 bit
unsigned __int64	unsigned __int64 huge = 1234567887654321;	64 bit

c) Turlarni o’zgartirish protseduralari

Standart turlarni o’zgartirish

C++ ning ma’lumotlarning turlari ustida qattiq nazorati tufayli imkoni boricha qiymatlarni saqlovchi, turlarni o’zgartirish amallari kiritilgan.

Boshqa o’zgaruvchidan ma’lum bir tur qiymatlarini olish uchun quyidagi konstruksiya ishlatiladi: **(yangi tur)o’zgaruvchi.**

Misol:

```
short S = 100;
int I = (int)S;
```

Bu misol ortiqcha buyruqlarga ega. C++ da ko’pgina tur o’zgaruvchilarining to’g’ridan-to’g’ri o’zlashtirilishi nazarda tutilgan, ammo ba’zi hollarda bu buyruqlar majburiy hisoblanadi (masalan, o’zgaruvchining qiymatini biror funksiyaga uzatishda).

Sonli qiymatlarni satrga almashtirish

C++ turlarning to'g'ridan-to'g'ri almashtirishda o'zgaruvchini uning o'nlik ko'rinishidan belgilar qatori ko'rinishiga yo'l qo'ymaydi, chunonchi, ular shakllarning ko'gina komponentalarida ishlatiladi. To'g'ridan-to'g'ri almashtirish faqatgina asosiy va qo'shimcha turlar uchun amalga oshiriladi. Massiv hisoblanadigan satr kattaliklar hosilaviy tur bo'lganligi sababli bunday almashtirishga yo'l qo'yilmaydi.

Bunday almashtirishlar uchun quyidagi standart almashtirish funksiyalari ishlatiladi: IntToStr, StrToInt, FloatToStr va boshqalar. Ko'pchilik ma'lumotlar turlari uchun shu kabi satrga va teskari o'tkazish funksiyalari mavjud.

Misol:

```
char S[10]; // belgilar massivi
int I = 100; // butun qiymatli o'zgaruvchi
S = IntToStr(I);   // o'tkazish
```

Shartli buyruq

Dasturda tarmoqlanishni amalga oshirish, ya'ni ba'zi faktorlarga bog'liq holda turli amallar bajarilishi uchun **if** buyrug'i ishlatiladi.

Buyruq quyidagi formatga ega**:**

if (ifoda){ 1 - operator;} [else { 2 - operator;}]

if buyrug'ining bajarilishi ifodaning qiymatini hisoblashdan boshlanadi. So'ngra ish quyidagi sxema asosida amalga oshiriladi:

- agar ifoda rost bo'lsa (ya'ni 0 dan farqli), u holda 1 - operator bajariladi.
- agar ifoda yolg'on bo'lsa (ya'ni 0 ga teng), u holda 2 - operator bajariladi.
- agar ifoda yolg'on va 2 - operator yo'q bo'lsa (kvadrat qavsga zarur bo'lmagan konstruktsiya kiritiladi), u holda if dan keyingi buyruq bajariladi.

Misol:

```
if (i < j)
{
        i++;
}
else
{
        j = i-3;
        i++;
}
```

Bu misol 1 - operatorning o'rnida ham, 2 - operatorning o'rnida ham murakkab konstruktsiya qatnashishi mumkinligini bildiradi. Ichma-ich if buyrug'ini ishlatish imkoniyati ham mavjud. if buyrug'i boshqa if buyrug'ining if yoki else konstruktsiyalari ichida qatnashishi ham mumkin.

Misollar:

```
int t = 2;
int b = 7;
int r = 3;
if (t>b)
{
        if (b < r)
        {
                r = b;
        }
}
else
{
        r = t;
        return (0);
```

}

Bu dastur bajarilganda r ning qiymati 2 ga teng bo'ladi.

1 - Misol.

Dastur tasnifi

Masala quyidagicha qo'yiladi: Standart o'lchovli (8x8) shaxmat taxtasiga bug'loy donlari quyidagicha Qo'yiladi: birinchi maydonga bitta don, keyingi har bir maydonga oldingi maydonga qo'yilgan donning ikki baravarida don qo'yiladi, ya'ni birinchi maydonga bitta, ikkinchi maydonga ikkita, uchinchiga to'rtta va hakazo. Taxtaning barcha maydonlaridagi donlarning umumiy sonini topmng.

Zarur ko'nikmalar

Mazkur dasturni yozish uchun quyidagi ko'nikmalargi ega bo'lish zarur:

1. Shakllar yaratish uchun kamida standart panelning oddiy komponentalaridan tashkil topgan dasturlar yaratish muhidan foydalanishni bilish. Taymer sistemali komponentadan foydalanishni bilish.
2. O'zgaruvchilar turlarini va ularning qiymatlari chegarasini bilish.
3. Sonli o'zgaruvchilarni satrga o'tkazuvchi standart protseduralarni bilish.
4. Shartli buyruqni ishlatishni bilish.

Muammolar

1. Mazkur dasturning asosiy muammosi **unsigned __int64** turidagi satr o'zgaruvchiga o'tkazuvchi standart funksiyani ishlatishning va shuncha katta sonning boshqa tur o'zgaruvchisida saqlash imkoniyatining yo'qligidadir. Bu muammoni hal qilish uchun **unsigned __int64** tur o'zgaruvchini ikkita **__int64** turiga va so'ngra mos ravishda satrga o'tkazuvchi standart funksiyalarni (ya'ni shu nom bilan, ammo boshqa tur uzatuvchi parametrlar bilan) yuklovchi funksiya tashkil qilingan. Dastur kodida bu funksiya alohida izoh bilan ajratib ko'rsatilgan.

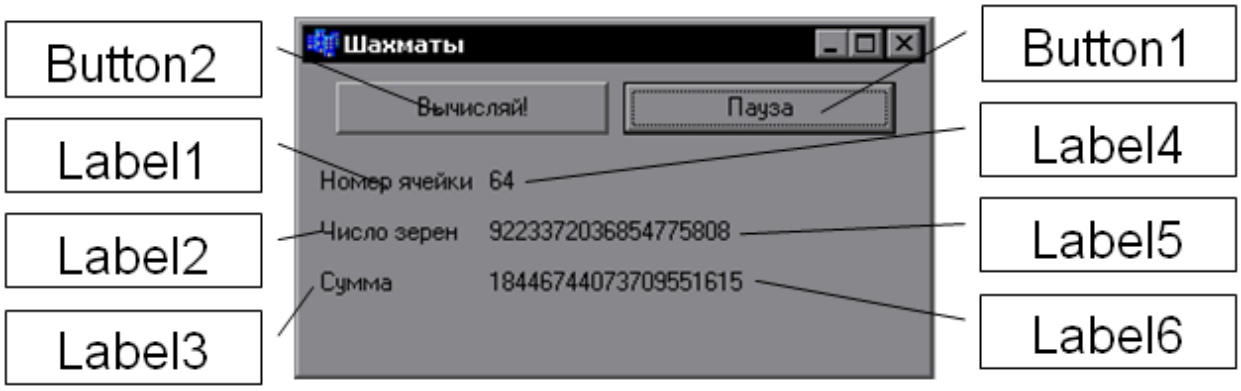
2. Mazkur dasturning ikkinchi muammosi quyidagilardan iborat: agar shaklga faqat natijaviy qiymatlarni chiqarsak, u holda dasturning ko’rsatmali ishlashi yo’qoladi. Bu esa bizni sikllardan foydalanishdan mahrum qiladi.

Bu muammoni sikllarni global o’zgaruvchilardan foydalanib ishlatishni simulyatsiya qilish yo’li bilan hal qilinadi. Shunday qilib, siklning tanasi alohida protsedura sifatida tashkil qilinadi va biror-bir xodisada u protsedura ishlatiladi, masalan, foydalanuvchi tomonidan tugmachalar bosilganda yoki taymerning holatlarida.

Masalaning echimi

Forma

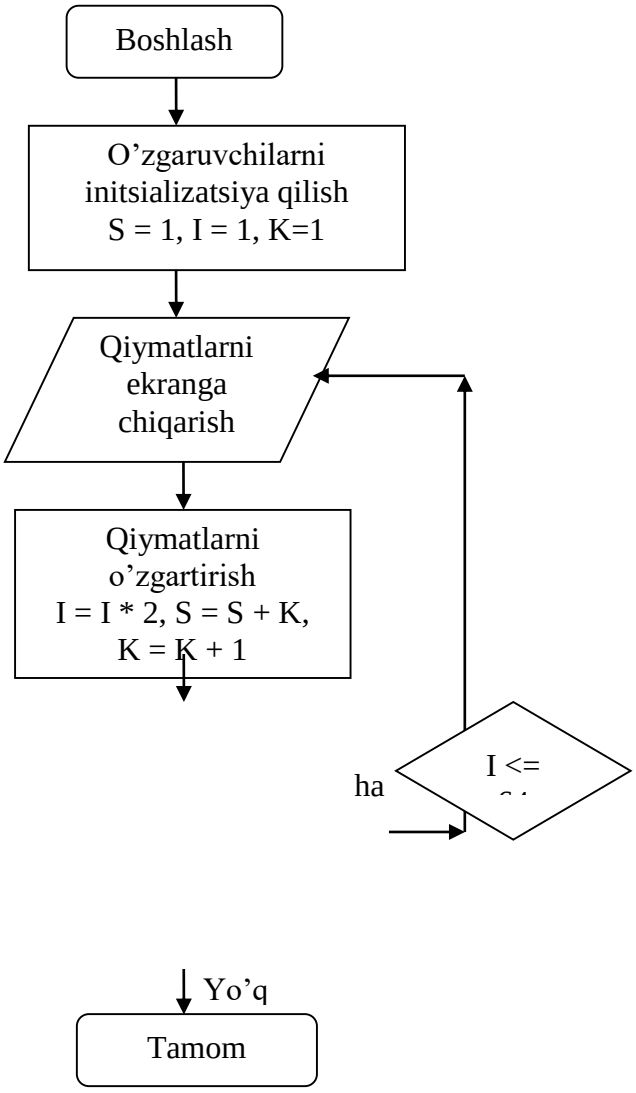
Bu dasturni amalga oshirish uchun quyidagi komponentalarishlatiladi: «Metka» (Label), «Knopka» (Button) va «Taymer» (Timer). Birinchi ikkita komponenta Standard bandida, taymer esa System bandida joylashgan.



Natijalarni tasvirlash uchun Label sinfi komponentalarining “Caption” xossasi qiymatlarini o’zgartirish zarur.

Tugmachalar uchun **onClick** xodisasining va taymer uchun **onTimer** xodisasining harakatlarni hosil qilinadi (dastur kodi yoziladi).

Blok cxema



Dastur kodi

```
/* __int64 tur bilan ishlatish uchun  IntToStr  funksiyasini oddiy yuklash */

AnsiString __fastcall IntToStr(unsigned __int64 Value)
{
 __int64 k = floor(Value/100000);
 __int64 l = Value - k*100000;

 if(k!=0)
  {return IntToStr(k)+IntToStr(l);}
```

```
 else
  {return IntToStr(l);}
}

//---------------------------------------------------------------------------

/* Global o'zgaruvchilar*/
unsigned __int64 s = 1, i = 1;
short j = 1;
char T = 0;

//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
 if(j<8*8)     //Maydonning navbatdagi =iymatini hisoblash
 {
  j++;
  i *= 2;
  s += i;
 }

 Label4->Caption = IntToStr(j); // Ularni shaklga chi=arish
 Label5->Caption = IntToStr(i);
 Label6->Caption = IntToStr(s);
}

//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
```

```
{
 T = !T;    // Taymerdan foydaldnishni o’zgartirish

 if(!T)     //Sarlavha
  {Button2->Caption = "Pusk";}
 else
  {Button2->Caption = "Pauza";}
}

//---------------------------------------------------------------------------

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 if(T){TForm1:Button1Click(Form1);}  // Taymerning tiki
}
```

d) Boshqaruvchi strukturalar

Siklik strukturaga kirish

Ko’pgina takrorlanuvchi elementlarga mos algoritmga mos dastur kodini yozish uchun quyidagi buyruqlar yordamida hosil qilinadigan siklik strukturalarni ishlatishga to’g’ri keladi.

for buyrug’i

for buyrug’i - sikllarni tashkil qilishning eng umumiy (ommaviy) usulidir. U quyidagi ko’rinishga ega:

for (1-ifoda; 2-ifoda; 3-ifoda) {tana}.

1-ifoda odatda siklni boshqaruvchi o’zgaruvchining boshlang’ich qiymatini o’rnatish uchun ishlatiladi. 2-ifoda sikl tanasi bajarilishi kerak bo’lgan shartni

ifodalaydi. 3-ifoda sikl tanasining bajarilganidan keyin o’zgaruvchining o’zgarishini boshqaradi.

for buyrug’ining bajarilish sxemasi quyidagicha:

1. 1-ifoda hisoblanadi.
2. 2-ifoda hisoblanadi.
3. Agar 2-ifoda noldan farqi (rost) bo’lsa, u holda sikl tanasi bajariladi. So’ngra 2-ifoda bajariladi va boshqarish 2-punktga uzatiladi. Agar 2-ifodaning qiymati nol (yolg’on) bo’lsa, u holda boshqarish for buyrug’idan keyingi buyruqqa uzatiladi.

Shu narsa ahamiyatliki, shartni tekshirish har safar sikl boshida bajariladi. Bu narsa esa bajarish sharti boshidayoq nolga teng bo’lganda sikl tanasining biror marta ham bajarilmasligini bildiradi.

Misol:

```
 int i, b;
 for (i=1; i<10; i++)
{
        b=i*i;
}
```

Bu misolda 1 dan 9 gacha bo’lgan sonlarning kvadratlari hisoblanadi.

Ba’zi hollarda siklni boshqaruvyai birnechta zo’garuvchilarni ishlatishning imkoniyati mavjudligi for buyrug’ining moslashuvchanligini oshiradi.

Misol:

```
int top,  bot;
char string[100],  temp;
for ( top=0, bot=100 ; top < bot ; top++, bot--)
{
        temp=string[top];
        string[bot]=temp;
```

}

Belgilar satrini teskari tartibda yozuvchi bu misolda siklni boshqarish uchun ikkita top va bot o’zgaruvchilari ishlatiladi. Shuni ta’kidlash lozimki, bu erda 1- va 2-ifodadal o’rnida ketma-ket bajariluvchi va bergul bilan adratilib yozilgan bir nechta ifodalar ishlatilgan.

for buyrug’ini ishlatishning boshqa varianti cheksiz sikl tashkil qilishdir. Bunday siklni tashkil etish uchun bo’sh shartli ifodalarni ishlatish mumkin. TSikldan chiqish uchun esa odatda qo’shimcha shartlar yoki break buyrug’i ishlatiladi (bu buyruq keyinroq ko’riladi).

Misol:

```
for (;;)
{
        ...
        ... break;
        ...
}
```

C tilining sintaksisiga binoan buyruq ham, for buyrug’ining tanasi ham bo’sh bo’lishi mumkin. Buyruqning shakli izlashlarni tashkil etishda qo’llanilishi mumkin.

Misol:

```
for (i=0; t[i]<10 ; i++);
```

Bu misolda sikl o’zgaruvchisi bo’lgan i o’zgaruvchi qiymati 10 dan kichik bo’lmagan t massiv birinchi elementi nomerining qiymatini qabul qiladi.

While buyrug’i

while sikl buyrug’i sharti oldindan berilgan sikl buyrug’i deyiladi va quyidagi ko’rinishga ega:

while (ifoda) {tana};

Ifoda sifatida C tilining ixtiyoriy ifodasini ishlatish mumkin. Tana sifatida ixtiyoriy buyruqni, jumladan bo'sh va tarkibli (murakkab) buyruqlarni ham, ishlatish mumkin. while buyrug'ining ishlash sxemasi quyidagicha:

1. Ifoda hisoblanadi.
2. Agar ifoda yolg'on bo'lsa while buyrug'ining bajarilishi tugallanadi va boshqarish navbatdagi buyruqqa uzatiladi, aks holda while buyrug'ining tanasi bajariladi.
3. Jarayon 1-punktdan davom ettiriladi.

Quyidagi ko'rinishdagi sikl buyrug'i

for (1-ifoda; 2-ifoda; 3-ifoda) {tana};

while buyrug'i bilan quyidagicha almashtiriladi:

```
1-ifoda;
while (2-ifoda)
{
        tana
        3-ifoda;
}
```

for buyrug'ining bajarilishidagi kabi while buyrug'ida ham avvalo shartning bajarilishi tekshiriladi. Shuning uchun ham buyruq tanasini bajarish shart bo'lmagan hollarda while buyrug'idan foydalanish qulay.

for va while buyruqlarining ichida ma'lum mos turlar bilan e'ln qilingan lokal o'zgaruvchilarni ishlatish mumkin.

do while buyrug'i

do while sikl buyrug'i sharti oxirida berilan sikl buyrug'i deyiladi va sikl tanasini kamida bir marta bajarish zarur bo'lgan hollarda ishlatiladi. Bu buyruq quyidagi ko'rinishga ega:

do {telo} while (vo’rajenie);

do while buyrug’ining bajarilish sxemasi:

1. TSikl tanasi bajariladi (tarkibli buyruq bo’lishi ham mumkin).
2. Ifoda hisoblanadi.
3. Agar ifoda yolg’on bo’lsa, u holda do while buyrug’ining bajarilishi tugallaniladi va navbatdagi buyruq bajariladi. Agar ifoda yolg’on bo’lsa, u holda bajarish 1-punktdan davom ettiriladi.

while va do while buyruqlari ichma-ich joylashgan bo’lishi ham mumkin.

Misol:

```
int i,j,k;
...
i=0;  j=0;  k=0;
do
{
        i++;
        j--;
        while (a[k] < i)
        {
                k++;
        }
}
while (i<30 && j<-30);
```

break buyrug’i

break buyrug’i birlashgan switch, do, for, while sikllardan eng ichkisining bajarilishi tugallanilishini ta’minlaydi. break buyrug’i bajarilgandan so’ng boshqarish bajarilishi tugallangan sikldan keyingi buyruqqa uzatiladi.

Shu yo’l bilan muddatidan avval sikldan chiqish ta’minlanadi.

Continue buyrug'i

continue buyrug'i ham break buyrug'i kabi faqatgina sikl buyruqlarinig ichida ishlatiladi. Ammo undan farqli ravishda bajarish bajarilishi tugatilgan sikldan keyingi buyruqdan emas, balki bajarilishi tugallangan sikldan boshlanadi.

Misol:

```
int a,b;
for (a=1,b=0; a<100; b+=a, a++)
{
        if (b%2 != 0) continue;
         ...
        /* juft yig'indilarni qayta ishlash */
}
```

Bu misolda ko'p nuqta bilan belgilangan amallar b ning toq qiymatlaridagina bajariladi. Chunki 1 dan a gacha sonlar yig'indisi toq bo'lganda continue buyrug'i qayta ishlash buyruqlarini bajarmasdan, boshqarishni for siklining tanasini navbatdagi qiymat uchun bajarishga uzatadi.

Continue buyrug'i ham break buyrug'i kabi ichma-ich sikllarning eng ichkisining ishini to'xtatadi.

2 - Misol.

Dasturning tasnifi

Bu masala progressiyani tasvirlovchi tenglama bilan tavsiflanadi.

Bu masalani tasvirlovchi tenglama quyidagicha yoziladi:

$$y_n = \begin{cases} n & \text{n juft bo'lganda} \\ \sum_{i=0}^{n} i^2 & \text{n toq bo'lganda} \end{cases}, \; n = \overline{0,100}.$$

Muammolar

Xudi oldingidagi kabi bu dasturda ham hisoblashlar ko'rsatmali bo'lishi uchun eng ichki sikl global o'zgaruvchilardan foydalanish bilan almashtirilgan. Uning tanasi esa tugmachalarni bosish yoki taymer yordamida chaqiriladigan alohida protseduraga ko'chirilgan.

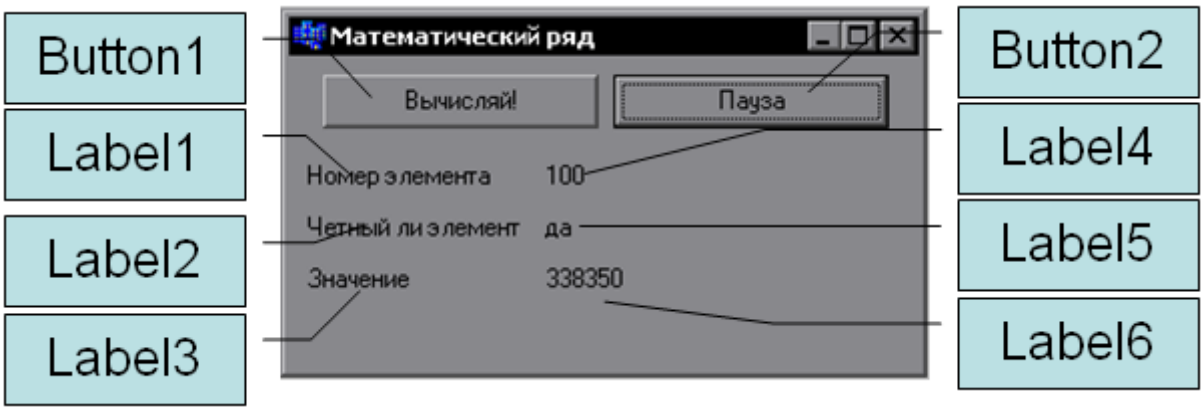
Zarur ko'nikmalar

Bu dasturni yozishda avvalgi dasturni yozihda orttirilgan bilimlardan tashqarimurakkab ifodalarni hisoblash uchun sikllarni ishlatishni ham talab qilinadi.

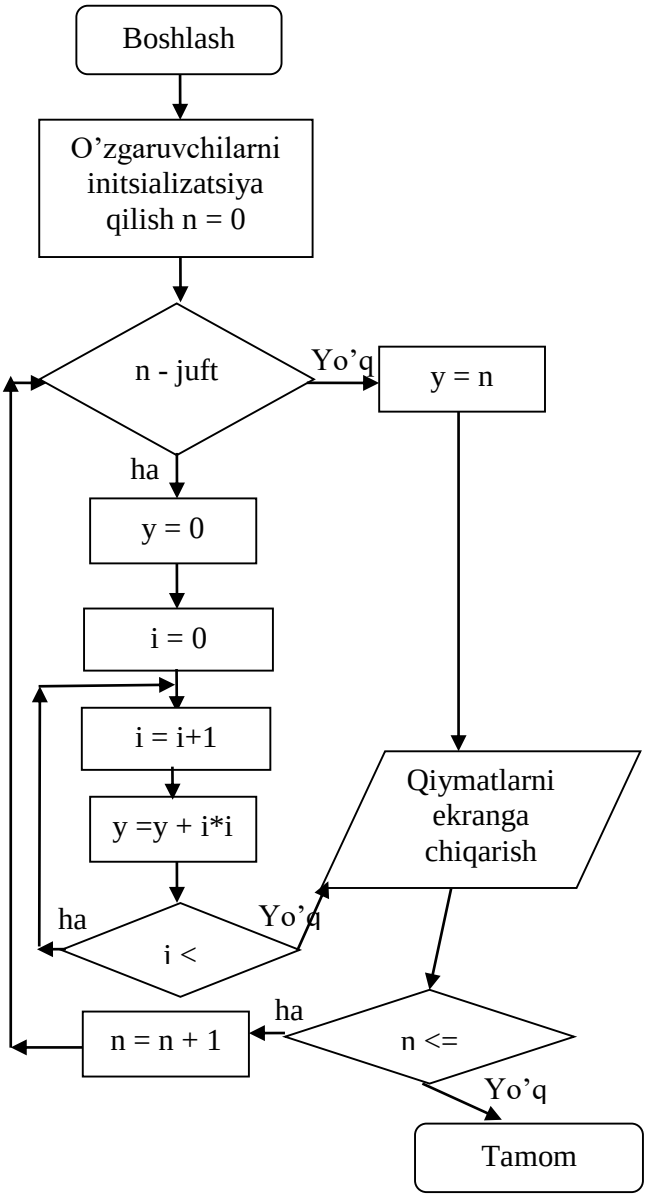
Yechish

Shakl

Bu masalani yechish uchun zarur bo'lgan shakl oldingi masalaning shakliga o'xshash bo'ladi va unda ba'zi elementlarning sarlavhalari (Caption xossasi) qatnashmaydi xolos.



Blok cxema



Natural sonlar kvadaratlarining yig’indisini hisoblash uchun for tsili ishlatildi.

Dastur kodi

```
/* Global o’zgaruvchilar*/
int n = 0, y = 0;
char T = 0;

void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
 if(n%2 == 0)     // juftlikka tekshirish
 { //ha - juft
  y = 0;
  for(int i=0;i<n+1;i++)
  {
   y += i*i;
  }
  Label5->Caption = "ha";
 }
 else
 { //yo'= - toq
  y = n;
  Label5->Caption = "yo'q";
 }

 Label4->Caption = IntToStr(n); // Shaklga chiqarish
 Label6->Caption = IntToStr(y);

 if(n<100){n++;} // nomerni oshirish
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 T = !T;
 if(!T)     // Taymerning Pusk/Pauza tugmasi
  {Button2->Caption = "Pusk";}
 else
  {Button2->Caption = "Pauza";}
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 if(T){TForm1:Button1Click(Form1);}    //Taymerning tiki
}
```

3. Massivlar bilan ishlash

Massiv tushunchasi

Massiv bir xil turdagi bir nechta o'zgaruvchilarning to'plamidan tashkil topadi (ba'zi adabiyotlarda ular jadvallar deb ham nomlanadi). Nomi **a** bo'lgan LENGTH elementdan iborat TYPE turidagi massiv quyidagicha e'lon qilinadi:

```
type a[length];
```

Bu buyruqda type turiga tegishli maxsus a[0], a[1], ..., a[length-1] nomli o'zgaruvchilar e'lon qilinadi. Massivning har bir elementi o'z nomeri - indeksga ega bo'ladi. Massivning x - elementiga murojat qilish indeksatsiya amali yordamida amalga oshiriladi:

```
int x = ... ;                  // Butun qiymatli indeks
TYPE value = a[x];             // x - elementni o'qish
a[x] = value;                  // x- elementga yozish
```

Indeks sifatida butun turdagi qiymat beruvchi ixtiyoriy ifodani ishlatish mumkin: char, short, int, long. C tilida massiv elementlarining indeksi 0 dan (1 dan emas) boshlanadi, LENGTH uzunlikdagi massivning oxirgi elementining indeksi esa LENGTH-1 (LENGTH emas). Shuning uchun ham massivning barcha elementlari bo'yicha sikl quyidagicha yoziladi:

```
TYPE a[LENGTH]; int indx;
for(indx=0; indx < LENGTH; indx++)
...a[indx]...;
```

Bu erda indx < LENGTH sharti indx <= LENGTH-1 ga teng kuchli. Massiv chegarasidan chiqish (mavjud bo'lmagan elementni o'qish/yozish) kutilmagan natijalarga va dastur ishida ham kutilmagan holatlarga olib kelishi mumkin. Bunday xatolar massivlar bilan ishlashdagi eng ko'p yo'l qo'yiladigan xatolar hisoblanadi.
Statik massivlarni uning elementlari qiymatlarini {} ichida vergul bilan ajratib yozish, ya'ni initsializatsiya qilish yo'li bilan ham e'lon qilish mumkin. Agar massiv uzunligidan kam elementlar berilgan bo'lsa, u holda qolgan elementlari nol deb hisoblanadi:

```
int a10[10] = { 1, 2, 3, 4 };          // va 6 ta nol
```

Agar massivlarni initsializatsiya qilishda uning o'lchovi berilmasa u kompilyator tomonidan hisoblanadi:

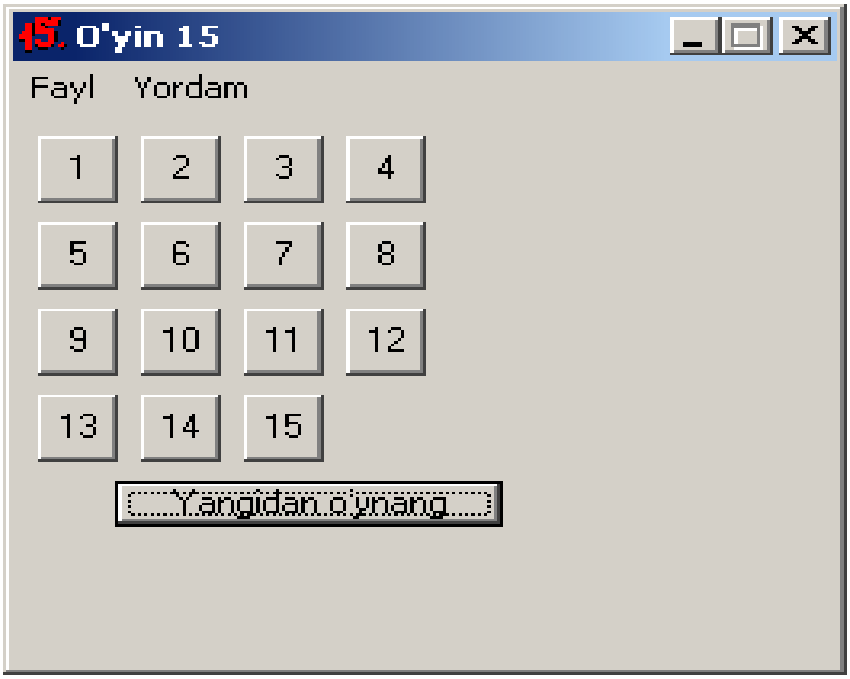
```
int a3[] = { 1, 2, 3 };          // Xuddi a3[3] kabi
```

Xulosa

Ushbu kurs ishida C++ haqida va C++ Builder muhitida ish olib borish haqida to’xtalib o’tdik.

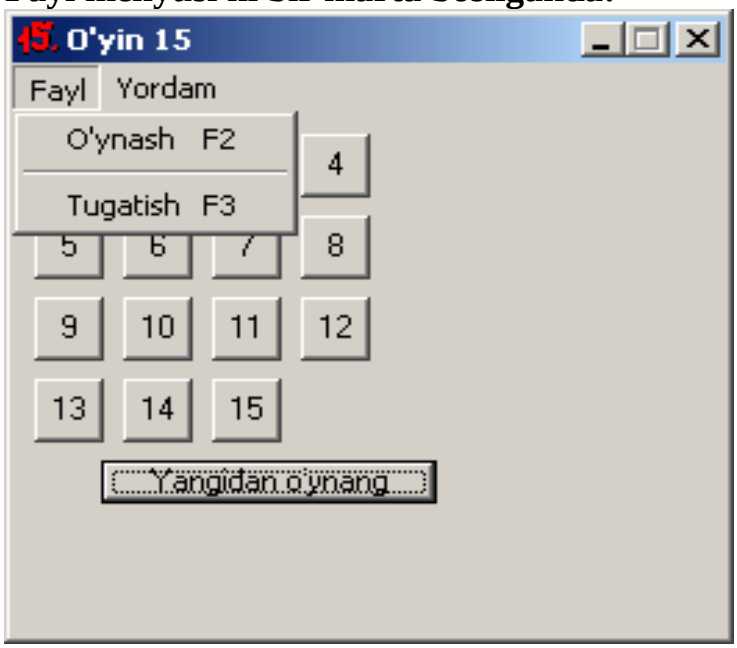
Bizga ayon bo’lganidek, zamon rivojlangan sari bizdan yanada tezkorlikni talab qilib, barcha jarayonlarni optimallashtirishni talab qila boshladi. Biz yangi dasturiy ishlanmalar ishlab chiqishimiz uchun yangi visual dasturlash muhitlari yaratildi. Bularning barchasi bizdan kam vaqt sarflab ko’proq natijalarga erishishimizga yordam beradi. Shuni aytib o’tishimiz joizki barcha dasturiy ishlanmalar iloji boricha insonlar vaqtini tejab ularning yengilini oson qilish uchun yaratiladi.

AMALIY ISH



“15 O’yini” asosiy oynasining asosiy ko’rinishi. Bu oynada Fayl va yordam menyulari joylashgan va yana o’yin maydonchasi, o’yin maydonchashi ko’rib turganimizdek 1 dan 15 gacha raqamlardan tarkib topgan bu oynada yana o’yinni boshqatdan (yangidan) boshlash tugmachasi ham bor.

Fayl menyusi ni bir marta bosilganda:



Fayl menyusida o’ynash (F2) va Tugatish (F3) (O’yindan chiqish) bo’limlari bor. Nomidan ham ko’rinib turibdi bu bo’limlar vazifasi o’yinni boshash va o’yindan chiqish vazifalarini bajaradi

Yordam menyusi bosilganda:

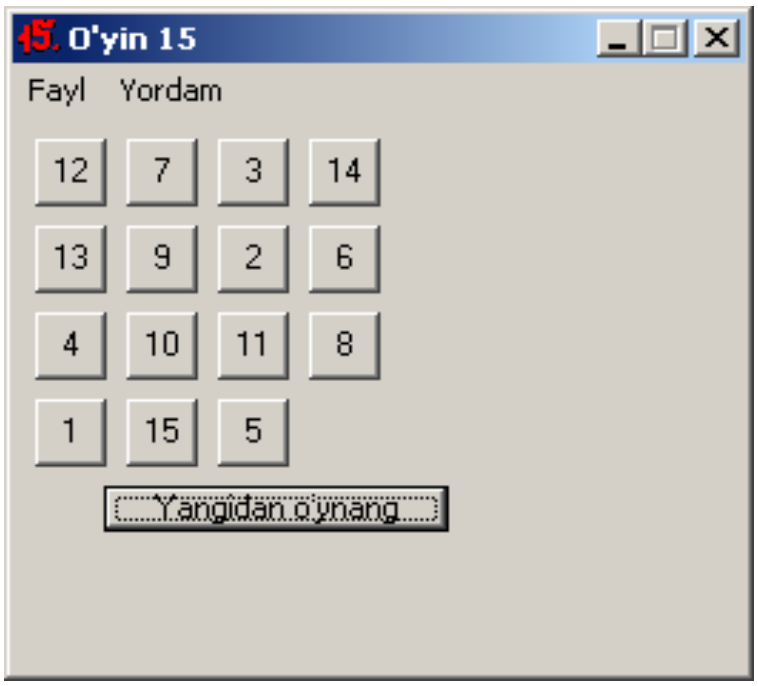
Muallif va o’yin qoidasi bandlari hosil bo’ladi. Bu menyu muallif bo’limida muallif haqida ma’lumotga ega bo’lish mumkin. F1 klavishasi bilan esa o’yin qoidasi haqida bilib olish mumkin.

O’yinni yutsangiz:

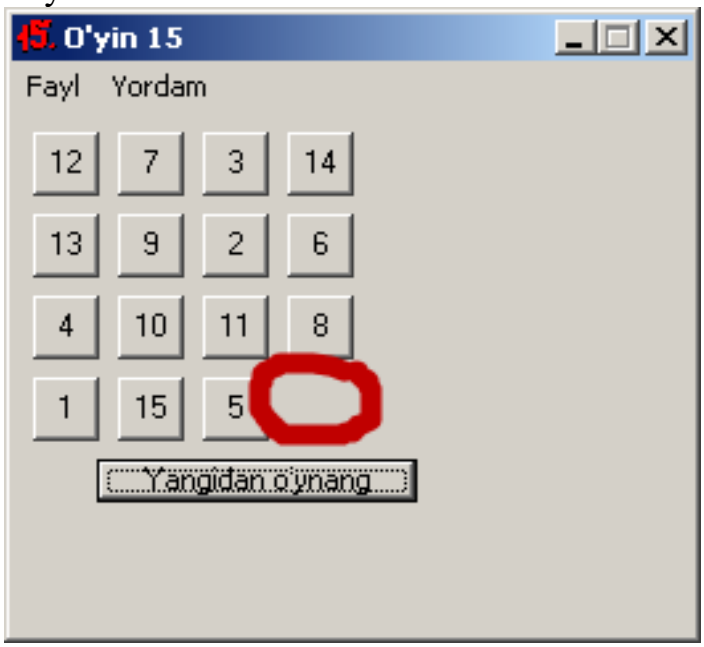
Shunga o’xshash ko’rinishga o’tadi ya’ni sonlar tartiblangan va qadamlar soni ko’rsatiladi.

Bu yerda qoyil! Va Qadamlar o’yinni yutganingizda ya’ni sonlarni tartiblashtirib bo’lganingizda chiqadi.

O’yinning boshlang’ich holati:

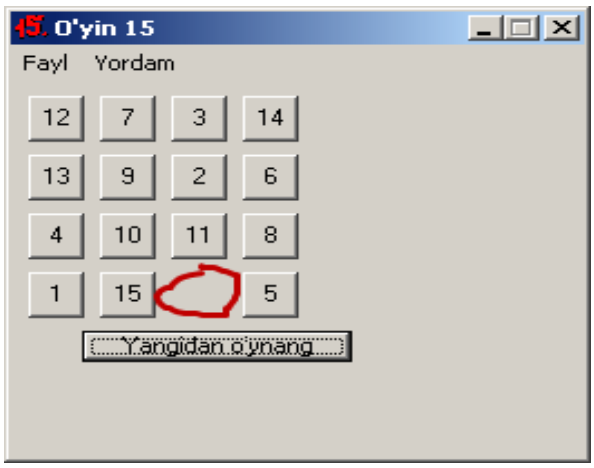


Bu ko’rinishga ya’ni o’yin boshi (sonlar tartiblanmagan holat) ga Yangidan o’ynash tugmasi bosilganidan keyin o’tish mumkin bo’ladi.
O’ynash uchun:



5 yoki 8 ustida sichqonchaning chap tugmasini bir marta bossangiz ulardan biriyanisizning tanlaganingiz qizil bilan belgilangan sohaga o’tadi va o’rnini bo’shatadi.

O’yin jarayoni:



Mana 5 ni ustiga bisilganda o’rnini bo’shatib bo’sh joyga o’tdi. Endi 15 yoki 11 yoki yana 5 ni bosganingizda ulardan biri ya’ni siz bosgan raqam qizil bilan belgilangan joyga o’tib o’z o’rnini bo’shatadi.

11 bosilgandagi holat:

G15.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "G15.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
Tgame15 *game15;
//---------------------------------------------------------------------------
__fastcall Tgame15::Tgame15(TComponent* Owner)
        : TForm(Owner)
{
 int i,j,k;
 for(k=0;k<15;k++)
  mass[k]=k+1;
```

```
 mass[15]=0;
 randomize();
 i0=3;j0=3;
 B=new TButton*[16];
 for(k=0;k<16;k++)
 {
  B[k]=new TButton(Owner);
  B[k]->Caption=mass[k];
  B[k]->Parent=game15;
  B[k]->Visible=1;
  i=k/4;j=k-4*i;
  B[k]->Height=25;
  B[k]->Width=25;
  B[k]->Top=8+i*32;
  B[k]->Left=8+j*32;
  B[k]->OnClick=_Cl_;
 }
 B[15]->Visible=0;
 Ur->Visible=0;
 Cnt->Visible=0;
 Ura->Visible=0;
 count=999;
}
//---------------------------------------------------------------------------
void __fastcall Tgame15::mixClick(TObject *Sender)
{
 int i,j,k,n;
 count=0;
 for(k=0;k<15;k++)
  mass[k]=k+1;
 mass[15]=0;
 for(k=0;k<2*N;k++)
 {
  do
  {
   i=random(15);
   j=random(15);
  }while(i==j);
  n=mass[i];mass[i]=mass[j];mass[j]=n;
 }
 i0=3;j0=3;
 for(k=0;k<15;k++)
 {
  B[k]->Caption=mass[k];
  B[k]->Visible=1;
```

```
 }
 B[15]->Visible=0;
 Ura->Visible=0;
 Ur->Visible=0;
 Cnt->Visible=0;
 Ura->Visible=0;
}
//---------------------------------------------------------------------------
void __fastcall Tgame15::_Cl_(TObject *Sender)
{
 int i,j,t,k;
 k=num(Sender);
 if(k!=-1)
 {
  i=k/4; j=k-4*i;
  if(!(i0==i&&j0==j))
  {
   if(i0==i)//ñîâïàëà ñòðî÷êà
   {
    if(j0<j)
    {
     for(t=j0+i0*4;t!=k;t++)
     {
      mass[t]=mass[t+1];
      B[t]->Caption=mass[t];
     }
    }
    else
    {
     for(t=j0+i0*4;t!=k;t--)
     {
      mass[t]=mass[t-1];
      B[t]->Caption=mass[t];
     }
    }
    mass[k]=0; B[k]->Visible=0;
    B[j0+4*i0]->Visible=1;
    j0=j;i0=i;count++;
   }
   else if(j0==j)
   {
    if(i0<i)
    {
     for(t=i0;t!=i;t++)
     {
```

```
      mass[t*4+j0]=mass[t*4+j0+4];
      B[t*4+j0]->Caption=mass[t*4+j0];
     }
    }
    else
    {
     for(t=i0;t!=i;t--)
     {
      mass[t*4+j0]=mass[t*4+j0-4];
      B[t*4+j0]->Caption=mass[t*4+j0];
     }
    }
    mass[k]=0; B[k]->Visible=0;
    B[j0+4*i0]->Visible=1;
    j0=j;i0=i;count++;
   }
  }
 }
 if((i0==3)&&(j0==3))
 {
  if(check())
  {
   Ura->Visible=1;
   Ur->Visible=1;
   Cnt->Visible=1;
   Cnt->Caption=count;
  }
  else
  {
   Ur->Visible=0;
   Cnt->Visible=0;
   Ura->Visible=0;
  }
 }
 else
 {
  Ur->Visible=0;
  Cnt->Visible=0;
  Ura->Visible=0;
 }
}
//---------------------------------------------------------------------------
int __fastcall Tgame15::num(TObject *Sender)
{
 for(int k=0;k<16;k++)
```

```
 {
  if(Sender==B[k]) return k;
 }
 return -1;
}
//---------------------------------------------------------------------------
int Tgame15::check(void)
{
 for(int k=0;k<15;k++)
  if(mass[k]!=k+1) return 0;
 if(mass[15]!=0) return 0;
 return 1;
}
void __fastcall Tgame15::mmfbegClick(TObject *Sender)
{
 mixClick(Sender);
}
//---------------------------------------------------------------------------
void __fastcall Tgame15::mmfendClick(TObject *Sender)
{
  Application->Terminate();
}
//---------------------------------------------------------------------------
void __fastcall Tgame15::mmhAboutClick(TObject *Sender)
{
        fmAbout->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall Tgame15::mmhRulesClick(TObject *Sender)
{
Form1->Show();
}
//---------------------------------------------------------------------------
```

About.h

```
//---------------------------------------------------------------------------
#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
```

```
#include <jpeg.hpp>
//---------------------------------------------------------------------------
class TfmAbout : public TForm
{
__published:        // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
    TImage *Image1;
private:    // User declarations
public:     // User declarations
        __fastcall TfmAbout(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TfmAbout *fmAbout;
//---------------------------------------------------------------------------
#endif
```

Foydalanilgan adabiyotlar

1. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль.- М.:Наука, 1988.-
   320с.
2. Абрамов С.А.,Гнезделова Капустина Е.Н.и др. Задачи по программированию. - М.: Наука,
   1988.
3. Кэнту М. Delphi 5 для профессионалов.- СПб: Питер, 2001. -944 с.
4. Пилшиков В.Н. Упражнения по языку Паскал-М.: МГУ, 1986.
5. Б. Керниган. Д. Ритчи «Язык програмирования Си» М. Финансы и статистика 1992 г.
6. Д. Маслова «Введение в языку Си» М. 1991 г.
7. Л. Амераль «Програмирование графики на Турбо Си» М., «Сол систем» 1992 г.
8. P.Karimov, S.Irisqulov, A.Isabaev «Dasturlash» Toshkent-2003 yil.
9. А. Нейбауэр. «Моя первая программа на Си/Си++». Петербург. 1995 г.
10. А. А. Мот, Дж. Ратклифф и др. «Секреты программирование игр». Петербург. 1995 г.

Internet resurslari:

www.dasturchi.uz

codeprojekt.com

www.ziyonet.uz