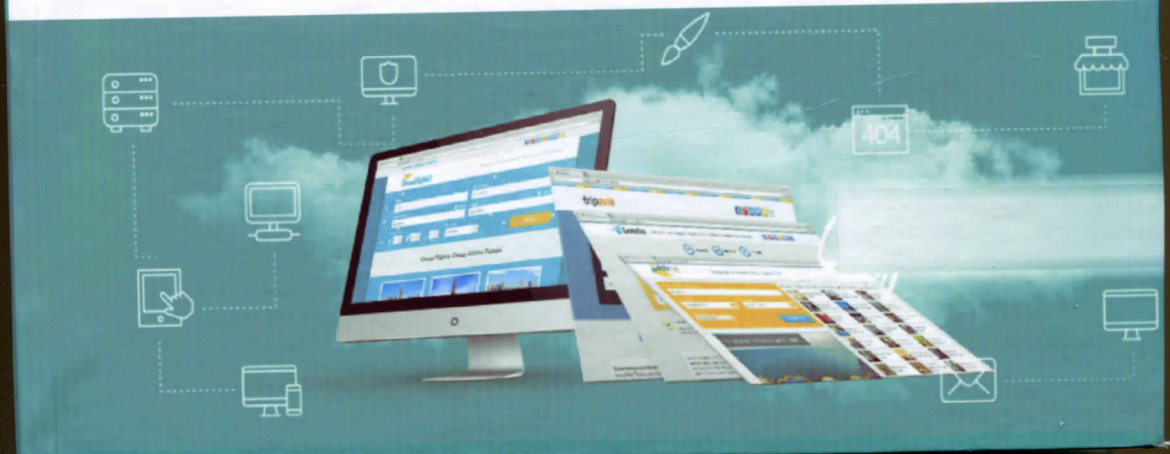
M.YU.XAYDAROVA,
N.M.QURBONOV,
O.U.MALLAYEV

VISUAL C++ DA KICHIK LOYIHALAR YARATISH



O‘ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI
VA KOMMUNIKASIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT
AXBOROT TEXNOLOGIYALARI UNIVERSITETI

M.YU.XAYDAROVA,
N.M.QURBONOV, O.U.MALLAYEV

VISUAL C++ DA KICHIK LOYIHALAR YARATISH

O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi
tomonidan o‘quv qo‘llanma sifatida tavsiya etilgan.

TOSHKENT – 2019

UO'K: 004.43(075.8)
KBK: 32.973.26-018.1

M.Yu.Xaydarova, N.M.Qurbonov, O.U.Mallayev. Visual C++ da kichik loyihalar yaratish. O'quv qo'llanma . – T.: «Aloqachi», 2019. – 224 b.

ISBN 978–9943–5896–2–9

O'quv qo'llanma C++ dasturlash tilining asosiy tushunchalari, obyektga yo'naltirilgan dasturlash tamoyillari, Visual C++ dasturlash muhiti komponentalari hamda ularning hossa va hodisalarini o'rgatish asnosida visual ko'rinishga ega bo'lgan dasturiy vositalarni yaratishga, shuningdek ma'lumotlar bazasi va SQL so'rovlaridan foydalanib, kichik loyihalarni ishlab chiqish ko'nikmalarini shakillantirishga bag'ishlangan.

O'quv qo'llanma 5330200-Informatika va axborot texnologiyalari, 5330500-Kompyuter injiniringi, 5330600-Dasturiy injiniringi, 5350100-Telekommunikatsiya texnologiyalari, 5350200-Televizion texnologiyalari, 5350300-Axborot-kommunikatsiya texnologiyalari sohasida iqtisodiyot va menejment, 5350400-Axborot-kommunikatsiya texnologiyalari sohasida kasb ta'limi, 5350500-Pochta aloqa texnologiyasi, 5350600-Axborotlashtirish va kutubxonashunoslik va 5330300-Axborot xavfsizligi ta'lim yo'nalishlarida tahsil olayotgan talabalar hamda mazkur sohaga aloqador professor-o'qituvchilar, ilmiy xodimlar, magistrlar va mustaqil o'rganuvchilar uchun mo'ljallangan.

UO'K: 004.43(075.8)
KBK: 32.973.26-018.1

Taqrizchilar:
N.Ravshanov; Z.Raxmonov.
Mas'ul muharrir:
N.O.Raximov.

O'quv qo'llanma Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti ilmiy-uslubiy kengashining qarori bilan chop etishga tavsiya etildi.

ISBN 978–9943–5896–2–9

KIRISH

Axborotlashtirish zamonaviy dunyo taraqqiyotining eng muhim yo'nalishlaridan biri bo'lib, jahon fan texnikasining iqtisodiy va ijtimoiy rivojlanishida erishilgan yutuqlarini o'zida mujassamlashtirgan. Axborotlar ko'lamining to'xtovsiz oshib borayotgani yangi asrimizning o'ziga xos xususiyatlaridan biridir.

Bugungi kunda axborot resurslari, axborot banklari, axborot biznesi, axborot xavfsizligi kabi axborotlashgan jamiyatga xos bo'lgan yangi tushunchalar paydo bo'ldi va ular doimiy ishlatilib yuriladigan so'zlar qatoriga kirib bormoqda. Jahondagi barcha ilg'or mamlakatlar an`anaviy turlar qatori axborot industriyasini rivojlantirish va uning mahsulotlaridan foydalanishni mamlakatni rivojlantirishning ustuvor yo'nalishlaridan biri sifatida qarashmoqda. Mazkur o'quv qo'llanma "C++ da dasturlash" fanining na'munaviy va ishchi dasturlari asosida tayyorlangan bo'lib, 4 ta bobdan iborat.

O'quv qo'llanmaning **"Kirish. Visual Studio dasturida Visual C++ muhitining loyiha platformalarini tanlash usullari"** deb nomlangan birinchi bobi **Visual Studio** dasturida loyiha platformalarini tanlash, **Visual C++** muhitida dastur platformalarini sozlash, loyiha xatoliklarini bartaraf qilish usullari, kutubxonalarni birlashtirish va shu kabi bir nechta **Visual Studio** dasturining **Visual C++** muhitida dastur tuzish shartlariga bag'ishlangan.

O'quv qo'llanmaning **"Visual C++ning Console Application muhiti va unda ishlash"** deb nomlangan ikkinchi bobida **Visual Studio** dasturini tizimga o'rnatish va **Visual C++ning Console Application** muhitida dastur yaratish usullari va shartlari, xatoliklar, ularnining turlari va ularni bartaraf etish usullari, kutubxonalarni e'lon qilish va ularni chaqirish turlari, maxsus kutubxona (Math::) funksiyalaridan foydalanish usullari, berilganlarni kiritish va chiqarish, sonlar jadvalini konsolga chiqarish, lokal va global o'zgaruvchilarni e'lon qilish va ularning qo'llanilishi, maxsuc String turi va u bilan ishlash shartlari, statik va dinamik massivlar bilan ishlash, fayllar bilan ishlovchi maxsus (IO) kutubxonasi funksiyalari va ular yordamida turli hil kengaytmali fayllar bilan ishlash, struktura (struct), birlashma (union), sinf (class) e'lon qilish va uning qo'llanilish usullari, satrlar bilash ishlovchi maxsus funksilar haqida batafsil ma'lumotlar keltirilgan.

Uchinchi bob **"Visual C++ ning Windows Form Application muhiti va unda ishlash"** deb nomlangan bo'lib, unda foydalanuvchi interfeysini sozlash usullari, boshqaruv elementlari – **ToolBar (windows Form)** haqida ma'lumotlar va komponentalar va ularni ishlatilishi yuzasidan quyida keltirilgan mavzular bo'yicha amaliy mashg'ulotlar keltirilgan:

3.1. Windows Form Application ilovasini yaratish;

3.2. Form, Button, Label komponentalari va MessageBox xabarlar oynasi;

3.3. MouseHower hodisasi;

3.4. TextBox va DataTimePicker komponentalari va ularning xossalari;

3.5. Forma (Form) ni formalarga bog'lash usullari;

3.6. CheckBox, CheckedListBox, ComboBox, ListBox komponentalari va ularning xossalari;

3.7. TabControl va RadioButton komponentalari;

3.8. Berilganlarini lug'at (Dictionary) yordamida strukturali saqlash;

3.9. Bir prosedura orqali bir nechta hodisalarga ishlov berish;

3.10. Turli tipdagi fayllarning manbalariga LinkLabel komponentasi yordamida murojaatlar;

3.11. Klaviatura hodisalarini qayta ishlash;

3.12. Matnli maydonga kiruvchi ma'lumotlarni bashqarish;

3.13. try …catch istisnosini qayta ishlash orqali matnli faylni o'qish va matnli faylga yozish;

3.14. OpenFileDialog va SaveFileDialog komponetalaridan foydalanib fayllarni ochish va saqlash;

3.15. Matnli xujjatni chop qilish;

3.16. Formaga grafik fayldagi tasvirni chiqarish;

3.17. Formada grafik shakllarni va funksiya grafiklarini chizish.

3.18. Formada sichqoncha ko'rsatgichi orqali chizish;

3.19. Jadvalli ma'lumotlar asosida Chart komponentasi yordamida grafik diagrammalar yaratish;

3.20. Veb brauzerda **HTML** jadvallarni tasvirlash va shakllantirish;

3.21. Visual C++da **MS Word** imkoniyatlaridan foydalanib, jadvallar yaratish, ularni **Word** faylga eksport qilish va taqdim etish;

3.22. Visual C++ da **MS Excell** imkoniyatlaridan foydalanib, diagrammalar yaratish va ularni turli kengaytmalarda saqlash.

3.23. Visual C++ ning **Windows Aplication** muhitida komponentalarning joylashish vaziyatlarini nazorat qilish.

To'rtinchi bob **"Ma'lumotlar bazasini yaratish va ularga ishlov berish usullari"** deb nomlangan bo'lib, unda bobda **MBBT, MS Access** dasturi, **SQL (Structured query language)** so'rovlar tili haqida ma'lumotlar va komponentalar va ularni ishlatilishi yuzasidan quyida keltirilgan mavzular bo'yicha amaliy mashg'ulotlar keltirilgan:

4.1. «MS Access» dasturi yordamida talabalarning bilimlarini monitoring qilish ma'lumotlar bazasini yaratish texnologiyasi (konstruktor rejimida jadval yaratish, konstruktor rejimida so'rovlar yaratish, konstruktor rejimida makroslar yaratish, konstruktor rejimida formalar yaratish, formaga komponentalar joylashtirish)

4.2. SQL so'rovlar tili (**select** komandasi, **Like** va mantiqiy operatorlar (**and, or, not**), tartiblash (**order by**) va qisqartitrish (**distinct**), **where** komandasi, jamlash operatorlari: **count, min, max, avg, sum va group by, SQL**da qism so'rovlar: **in, exists, not exists, SQL** funksiyalari: **lower** va **upper** haqida, jadvallarga yangi yozuv qo'shish: **INSERT** operatori, jadvallardan yozuvlarni o'chirish: **DELETE** operatori, **SQL** so'rovlarini bajarilishi bo'yicha maxsus jadval);

4.3. Command va **Datareader** sinf ob`ektlari yordamida **MS Accessda** yaratilgan ma'lumotlar bazasining jadvalidagi barcha ma'lumotlarini o'quvchi dastur yaratish;

4.4. Console Application muhitida **MS Access** ma'lumotlar bazasini yaratuvchi dastur yaratish;

4.5. Console Application muhitida **MS Access** ma'lumotlar bazasining jadvallariga ma'lumotlar yozuvchi dastur yaratish;

4.6. Command, DataReader sinf ob`ektlari va **DataGridView** komponentasi yordamida MBning jadvalidan ma'lumotlarni o'quvchi vizual dastur yaratish;

4.7. Command, Adapter va **DataSet** sinf ob`ektlari va **DataGridView** komponentasi yordamida MBning jadvalidan ma'lumotlarni o'quvchi vizual dastur yaratish;

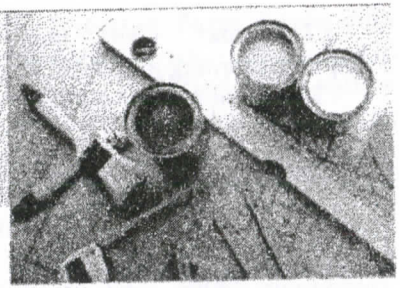
4.8. MS Access ning MBdagi jadval yozuvlarini yangilovchi vizual dastur yaratish;

4.9. MS Access ning MB dagi jadval yozuvlarini SQL so'rovlari va **"Command"** sinf ob`ekti yordamida o'chiruvchi vizual dastur yaratish;

4.10. Kichik loyihalar yaratish usullari va uning yuklanuvchi "инсталляционные" dasturlar paketini yaratish.

1- BOB. Visual Studio 2012 dasturida Visual C++ muhitining loyiha platformalarini tanlash usullari

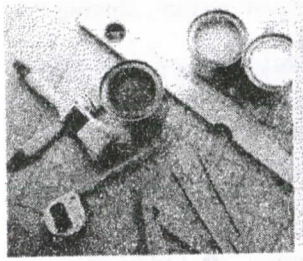
Visual C++ ga kirish



Bobning maqsadi

Algoritm, dasturlash, kompilyatsiya va kompilyatorlar, **Visual Studio 2012** dasturini tuzilishi, texnik tillar va yuqori darajadagi dasturlash tillari integrasiyalarini o`rganish, **Visual C++** muhiti bilan tanishish, **compile-time and run-time error** bilan tanishuv, algoritmlarni psevdokod bilan yoritish, dasturlash muammolarini o`rganish.

Bob mundarijasi



Loyihani o‘rganish jarayonida uskunalarni yig‘ish bilan birga yechimga olib keluvchi reja ishlab chiqing, bu bobda siz **Visual C++**ni o‘rganish uchun zaruriy tushunchalarga ega bo‘lasiz. Dasturiy ta`minot, dasturlash haqidagi qisqa tanishuvdan so‘ng, **Visual C++**da siz o‘zingizning ilk dasturingizni qanday yozish va ishga tushirishni o‘rganasiz. Bundan tashqari, Siz dasturlashdagi xatoliklarni qanday diagnoz qilish va tuzatishni, algoritmni tasvirlashda psevdokod ishlatish - dastur rejasiga muvofiq muammo yechimiga olib keluvchi ketma-ket tushuntirishlar bilan tanishasiz.

1.1. Dasturlash nima? Visual Studio dasturini tuzilishi

Kompyuterning eng asosiy ko‘rsatmalarini tezkor ketma-ketlikda amalga oshiradi.

Siz kompyuterni ish yoki vaqtichog‘lik uchun ishlatishingiz mumkin. Ko‘p insonlar kompyuterni har kungi ishlari, elektron pul o‘tkazish yoki kurs ishlarini yozish uchun ishlatadilar. Kompyuter shunday ishlar uchun kerak. Ular qo‘lda bajariladigan zerikarli yumushlarni bajarishda, zerikishsiz va charchamasdan sonlar yig‘indisini chiqarishda, sahifaga so‘zlar yozishda foydalaniladi.

Kompyuterning moslashuvchanligi juda ajablanarli hodisadir. Shu mashina yana ro‘yxat kitobini tekshirishda, kurs ishlari yaratishda va o‘yin o‘ynashda ham foydalaniladi. Boshqa mashinalar bilan solishtirganda, ular kamroq vazifalar bajarishadi, mashina haydashadi va tosterda non qizartirishadi. Kompyuter ko‘proq vazifalar bajaradigan, mahsus vazifalar buyuradigan dasturlarda ishlaydi.

Kompyuter dasturi qarorlar va ko‘rsatmalar ketma-ketligidir.

Kompyuter aslida ma’lumotlarni (sonlar, so‘zlar, raqamlar) saqlashga, qurilmalar (monitor, ovoz sistemasi va printer) bilan ulanishga va turli dasturlarni bajarishga xizmat qiladi. Kompyuter uchun yaratilgan dasturlar kompyuter tomonidan amalga oshirilishi kerak vazifalarni batafsil va aniq qadamlar yordamida bajarilishini ta’minlaydi. Kompyuterning barcha moddiy qismlari va qo‘shimcha qurilmalari uning apparat ta’minoti hisoblanadi. Kompyuterning ishlash va turli xil amallarning bajarishi hamda turli xil qurilmalar bilan aloqasini ta’minlash vazifasini dasturiy ta`minot amalga oshiradi.

Dasturlash kompyuterni bezatish va uni ijrosini ta`minlashdir.

Bugungi kundagi kompyuter dasturlari juda ham murakkab bo‘lib uning juda ham oddiy jarayonlardan tashkil topganiga ishonish qiyin.

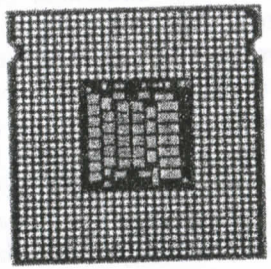
Ushbu kitobda, siz kompyuterda qanday dasturlashni, shuningdek amallar ijrosiga qanday yo'naltirishni o'rganasiz.

Kompyuter o'yinlarini harakatlar, ovoz effektlari va fantaziyalarga boy qilib yaratish kabi murakkab vazifa yuqori malakali dasturchilar jamoasini talab qiladi. Sizning dasturlashdagi ilk qadamlaringiz muvaffaqiyatli bo'lmasligi mumkin. Bu kitobda o'rganadigan tushuncha va ko'nikmalar Siz uchun asosiy fundament vazifasini o'taydi va ilk yaratgan dasturingiz Siz bilgan dasturlar bilan raqobatlasha olmasa xafsalangiz pir bo'lmasin. Sababi bu Sizning dasturlashdagi ilk qadamingiz.

Aslida, oddiy dasturlash jarayonlari ham ulkan zavq bag'ishlaydi. Uzoq muddatli mashaqqatli va zerikarli ishlar, zaruriy o'zgartirish va to'g'irlashlardan keyin kompyuterning berilgan buyruqni aniq va tez hamda Siz xoxlagandek amalga oshirishi juda ham zavqli taassurotdir.

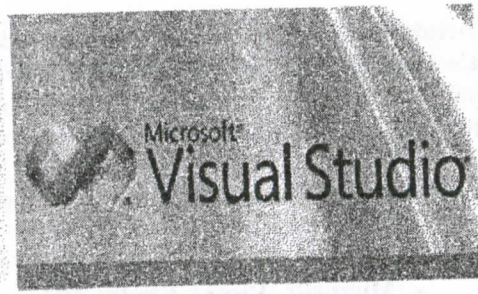
Dasturlash jarayonini tushunish uchun kompyuterni tashkil etgan qurilma bloklarini tushunishingiz kerak. Shaxsiy kompyuterni ko'rib chiqsak. Kompyuterning yuragi markaziy prosessor (CPU) (1.1-rasm) yagona chipdan yoki kichik birlikdagi chiplardan iborat. Kompyuter chipi metall yoki plastik komponentli korpusdan, metall ulagichlardan iborat, uzatkichlar ichki qismi esa kremniydan iborat. Prosessor ichki qismi juda murakkab tuzilgan.

Markaziy prosessor dasturni nazorat qiladi va bu orqali ma'lumotni qayta ishlaydi. Bunda kompyuter dastur nazoratini amalga oshiradi va ma'lumotlarning xotiradagi egallab turgan o'rnini aniqlaydi. Kesh xotira ma'lumotlar ustida qo'shish, ayirish, ko'paytirish va bo'lish kabi arifmetik amallarni bajaradi hamda tashqi xotira yoki qurilmalardagi qabul qilingan ma'lumotlarni qaytadan saqlay oladi.



1.1- rasm. Markaziy protsessor qurilmasi

Visual Studio dasturida Visual C++ muhitining loyiha platformalarini tanlash usullari



Microsoft kompaniyasi 2002-yil Sankt – Peterburg shahrida bo'lib o'tgan konferensiyasida Devid Chappel (David Chappell) .Net **Freamwork** dasturiy platformasiga bag'ishlab ma'ruza qildi. U .Ne platformasini yaratilishi **Windows** muhitida hamma narsani, ya'n dasturlash tillarini, interfeys va kutubxonalarni hamda ilovalarn o'zgartirib yuborishini aytib o'tdi. **.Net** markasi orqali quyidagi asosiy mahsulotlar yetkazib beriladi, bular:

-**.Net Framework** – amalga oshirish muhiti, unda yaratilga dasturiy komponentlar ishlatiladi. Bu muhit dasturiy kodlarn xavfsizligini ta'minlash, avtomatik ravishda keraksiz kodlarn yig'ishtirish va boshqa ishlar uchun mo'ljallangan.

-**Visual Studio. Net** – dasturchilar uchun yaratilgan muhit bo'lib, bitta kompilyatordan iborat, ya'ni **C++ kompilyatori**. C++ bunda yang o'zgartirilgan va integrallashtirilgan ishlab chiqish muhitidir. U dastu komponentlarini yaratishga mo'ljallangan. Bundan tashqari, bosho ko'pgina dasturlash tillarini qo'llab quvvatlaydi.

-**.Net Interprise Servers** (**.Net** korporativ serveri) – **SQL Serve 2000, Exchange 2000** va boshqalar.

.Net Framework ikkita komponentdan tashkil topgan. Uning ilo yaratuvchi asosiy instrumenti bu **Visual Studio.Net** hisoblanadi. Un har bir dasturlashtirish **.Net Framework** bilan umumiy interfeys orq aloqada bo'ladi. **Vs.Net** tarkibiga juda ko'p dasturlash tillari kiradi ulardan asosiysi **C++** dasturlash tili hisoblanadi.

Microsoft .Net (dot-net) dasturiy texnologiya bo'lib, u odd dasturlar kabi veb – ilovalarni yaratish uchun ishlatiladi (platfor sifatida birinchi bo'lib **Microsoft** firmasi tomonidan taklif kilingan).

Microsoft .Net ning asosiy maqsadi turli xil tilda yaratilgan har xizmat turlarini moslashtirishdir. Masalan **C++** tilida yozilgan das

Delphi tilida yozilgan kutubxona sinfi uslubida murojat etishi mumkin. **C++** tilida esa **Visual Basic.net**da yozilgan sinfga bog'langan sinf yaratish mumkin. **.Net** dagi har bir kutubxona o'z versiyasi haqidagi ma'lumotga ega bo'lishi, uning turli versiyalari orasidagi kelishmovchiliklarni bartaraf etadi.

Microsoft firmasining patentga ega texnologiyalari quyidagilar hisoblanadi:

- **.Net** – ilovalarini yaratish muhiti.
- **Microsift Visual Studio** (C++, Visual Basic.Net, Managed C++).
- **Borland Developer Studio** (Delphi For.Net , C++).
- **PascalABS.Net** va boshqalar.

.Net yaratish muhiti xuddi **Java** texnologiyasi kabi bayt – kod yaratadi. **.Net** ilova yaratuvchi model hisoblanadi. Uning asosiy maqsadi – qurilma va platformadan mustaqil bo'lgan ilovalar yaratishdir, bundan tashqari **Internet** orqali ma'lumotlarga murojaat etishni shakllantiradi.

.**Net** yadrosini quyidagi texnologiyalar tashkil etadi:

- **.Net Framework.**
- **.Net Enterprise Servers.**
- **. «quruvchi blok» xizmati.**
- **. Vs.Net.**

.Net platformasi klient operasion tizimlari, server va xizmatlar bilan integrallashgan va quyidagilardan iborat:

- **dastur modeli**, ya'ni XML – Web xizmati va ilova yaratish imkonini beradi;
- **xizmatlar to'plami** – «quruvchi blok», ya'ni maksimal samarali ilova yaratish imkonini beradi;
- **.Net Enterprise Servers** – serverlarni to'liq jamlanmasi bo'lib, u ilova yaratish uchun ishlatiladi;
- Shuningdek klient dasturiy ta'minoti (XP, CE) va Vs.Net kiradi.

.Net Framework quyidagilardan tashkil topgan:

- **CLR (Common Language Runtime);**
- **kutubxona sinfi** (Web va Windows formalari);

.Net quyidagi muammolarni yechadi:

- platformadan mustaqillik;
- .Net tillarini o'zaro bog'liqligini tashkil qiladi.

Microsoft Visual Studio – **Microsoft** kompaniyasining mahsuloti bo'lib, dasturiy ta'minot yaratish uchun integrallashgan muhitni va

boshqa instrumentlar qatorini taqdim etadi. Ushbu mahsulot kon ilovalar va grafik interfeys bilan ishlovchi ilovalar yaratish imkoniy beradi. **Windows Forms** texnologiyasini qo'llagan holda web-say web-ilovalar, web-xizmatlar, turli xil platformalar kodl boshqaruvchi (**Windows, Windows Mobile, .NET Wind Windows CE, .NET Framework, Xbox, Windows Phone, .I Compact Framework** va **Silverlight** texnologiyalari asosida) ar dasturlarni yaratish mumkin.

- **Tipi** – Integrallashgan ishlab chiqarish muhiti.
- **Ishlab chiqaruvchi** – Microsoft.
- **Dasturlash tillari** – C++ va C#.
- **Operasion tizim** – Windows.
- **Interfeys tili** – xitoy, ingliz, fransuz, portugal, nemis, ita yapon, koreys, ispan, rus tillari.
- **Oxirgi versiyasi** – Visual Studio 2017.
- **Lisenziya** – doimiy ravishdagi onlayn yangilanishga asosla
- **Sayt** – www.visualstudio.com.

Komponentlari. Visual Studio quyidagi bir q komponentlarni qo'llaydi:

- **Visual Basic .NET** – u Visual Basic asosida paydo bo'lgan.
- **Visual C++.**
- **Visual C#.**
- **Visual F#** (Bu Visual Studio 2010 dan boshlab ishlatiladi).

Boshqa variantlarni quyidagilarni yoqish orqali yetkazib mumkin:

Microsoft SQL Server yoki Microsoft SQL Server Express.

Visual Studioning oldingi tarkibiga quyidagi mahsulotlar kirit

- **Visual InterDev.**
- **Visual J++.**
- **Visual J#.**
- **Visual FoxPro.**
- **Visual SourceSafe** – fayl-Server tizimini boshqarish versiy

Versiyalari: Visual Studio 4.0 ning dastlabki versiyasida **Basic 3, Visual C++, Visual FoxPro va SourceSafe** ishlab chiq muhitlarini mustaqil paket sifatida yetkazib berilgan.

Visual Studio 97 – Visual Studioning birinchi ishlab ch versiyasi hisoblanib, turli xil muhitlarda dasturiy ta'minot maqsadida ilk bora ikki xil versiya (**Professional** va **Enterp**

ishlab chiqilgan. Ular **Visual Basic 5.0, Visual C++ 5.0, Visual J++ 1.1, Visual FoxPro 5.0** va ilk bora ASP ishlab chiqarish muhiti – **Visual InterDev** ni o'z ichiga oldi. Microsoft bu versiyasida ilk bora ko'plab tillarni: **Visual C++, Visual J++, Visual InterDev** va **MSDN** ni bir muhitda qo'llashga urinib ko'rdi va u **Developer Studio** deb nomlandi. **Visual FoxPro** va **Visual Basic**lar alohida ishlab chiqarish muhiti sifatida ishlatildi.

1.1-jadval. Visual Studio versiyalari ro'yxati

Rasmiy nomi	Kodli nomi	Ichki versiyasi	.NET Framework versiyasi	Ishlab chiqilgan sanasi
Visual Studio	N/A	4.0	N/A	1995 Aprel
Visual Studio 97	Boston	5.0	N/A	1997 Fevral
Visual Studio 6.0	Aspen	6.0	N/A	1998 Iyun
Visual Studio .NET (2002)	Rainier	7.0	1.0	13.02.2002
Visual Studio .NET 2003	Everett	7.1	1.1	24.04.2003
Visual Studio 2005	Whidbey	8.0	2.0, 3.0	07.11.2005
Visual Studio 2008	Orcas	9.0	2.0, 3.0, 3.5	19.11.2007
Visual Studio 2010	Dev10/Rosario	10.0	2.0, 3.0, 3.5, 4.0	12.04.2010
Visual Studio 2012	Dev11	11.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2	15.08.2012
Visual Studio 2013	Dev12	12.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2	17.10.2013
Visual Studio 2015	Dev14	14.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6	20.07.2015

Visual Studio 6.0 (1998). Visual Studio 6.0 – Windows 9x platformasida ishlaydigan Visual Studioning eng oxirgi versiyasi. Avvalgidek, ommaviy bo'lgan Visual Basic muhitidan foydalanildi. Ushbu versiyadan Microsoftning Windows platformasi ilovalari uchun .NET paydo bo'ldi.

Visual Studio .NET (2002). Visual Studio .NET (Kodli nomi **Rainner,** ichki versiyasi 7.0) – 2002 yil fevralda ishlab chiqarilgan (.NET Framework 1.0 ni qo'llaydi). Visual Studio .NET Servise Pack 1 uchun 2002 yil martida ishlab chiqildi.

Visual Studio .NET 2003. Visual Studio .NET (2003) (Kodli no **everett**, ichki versiya 7.1) – 2003 yilda ishlab chiqarilgan (**Framewo 1.1** qo'llaydi). Microsoft 2005 yilning aprelida maxsus muhit ishl chiqarilganini e'lon qildi, uning nomi **Microsoft Visual Studio .NE Professional Special Edition** deb ataldi. Bu maxsus ishlab chiqarilg muhit **Microsoft Visual Studio .NET 2003 Professional Edition** dasturiy ta'minotning Serverli komplektini va boshqa instrumentla qo'shadi (bu qismlarga operasion tizimlar: **Windows Server 20 Standard Edition** va **SQL Server 2000 Developer Editionlar** kira Muhitni bunday ko'rinishda rag'batlantirish ishlab chiqarishning ya bosqichiga ko'tarilishiga olib keldi va Microsoft boshqalar bi raqobatga kirisha oladigan dasturiy ta'minot yaratuvchini ishlab chiqd

Visual Studio .NET 2003 Service Pack 1 uchun 2006 yil sentyabrda ishlab chiqarilgan.

Visual Studio 2005. Visual Studio 2005 (kodli nomi **Whidb** ichki versiyasi 8.0) – 2005 yil oktyabrning oxirida ishlab chiqaril **(.NET Framework 2.0** ni qo'llaydi). Windows 2000 da ishlatiladi eng so'nggi rasmiy versiya. 2005 yil noyabr boshlarida ham bir qan seriyadagi mahsulotlarni Express tahririda ishlab chiqildi: Visual C 2005 Express, Visual Studio Basic 2005 Express, Visual C# 2 Express va boshqalar. 2006-yil 19 aprelda Express tahriri bepul q belgilandi. VS2005 Service Pack 1 uchun va barcha Express tahriri 2 yilning 14-dekabrida chiqarilgan. Windows Vista bilan birgali muammolarni hal qiladigan, SP1 uchun qo'shimcha patch 2007 yil 6-martida ishlab chiqarilgan.

Visual Studio 2008. Visual Studio 2008 (kodli nomi **Orcas,** i versiyasi 9.0) – 2007 yil 19-noyabrda, bir vaqtning o'zida .N Framework 3.5 bilan birgalikda ishlab chiqarilgan. Windows Microsoft Office 2007 va veb-ilovalarga ilovalar yaratish u mo'ljallangan (XP bilan birgalikda). LINQni qo'llagan holda C# Visual Basic tillarining yangi versiyalari ishlatilgan. J# Visual Stud kiritilmadi. 2008 yilning 28-oktyabrida birinchi marta versiyaga ru kiritildi.

Visual Studio 2010. Visual Studio 2010 (kodli nomi ***Ha*** **Ultimate** uchun – ***Rosario***, ichki versiya 10.0) - **.NET Framework** bilan birgalikda 2010 yil 12-aprelda yaratilgan. **Visual Studio C# 4. Visual Basic .NET 10.0** oldingi versiyalarida yo'q bo'lgan F# qo'llaydi.

Visual Studio 2012. Visual Studio 2012 - bu versiya ham 2010 yildagi tahrirdan tarqalgan, Visual Studio 2012 Expressga qilingan o'zgarishlar – barcha dasturlash tillari o'rnatildi, oldindan mavjud bo'lganlariga (Visual Basic 2010 Express va Visual C# 2010 Expresslarga) alohida paket sifatida beshta muhim versiya: **Visual Studio Express 2012 Web, Visual Studio Express 2012 Windows 8, Visual Studio Express 2012 Windows Desktop, Visual Studio Express 2012 Windows Phone** va **Visual Studio Express Team Foundation Server Express 2012** lar qo'shildi. Bu versiyalarning har biri alohida ilova sifatida tarqatiladi. Visual Studio express 2012 Windows 8 Windows Store uchun **Modern**-interfeysidagi ilovalarga ishlov berishga imkon bersa, Visual Studio Express 2012 Windows Desktop esa ishchi stol uchun ilovalarga "klassik" ishlov berishga imkon beradi. Visual Studio 2012 yordami bilan C++ da faqat Windows 7 SP1 va Windows 8 lar uchun ilovalarga ishlov berish mumkin.

Visual Studio 2013 2013 yil 17-oktyabrda **.NET 4.5.1** bilan birgalikda ishga tushirilgan. Biz ushbu kitobda VS 2012 versiyasi yordamida kerakli ilovalarni yaratishni ko'rib chiqamiz va uning imkoniyatlari haqida to'liqroq keyin to'xtalamiz.

Visual Studio 2015. 2014-yilning 12-noyabrida mahsulotning qat`iy varianti sifatida "Visual Studio 2015" ni qabul qildi. Visual Studio 2015 uchta tahrirni taqdim etdi: bepul hisoblangan **Community Edition** – barcha Express versiyalari, pullik hisoblangan kichik loyihalar yaratish uchun **Professional Edition** hamda katta loyihalar uchun **Enterprise Edition**. Birinchi STP 2014 yilning 2-iyunida ishlab chiqilgan, keyinroq 2015 yilning 29-aprelida **Release Candidate** ishlab chiqildi.

Yuqorida biz Visual Studioning bir qancha versiyalarini ko'rib o'tdik va e'tibor berdikki, ularning har biri qandaydir operasion tizim (OT)ning turlariga bog'liq ravishda ishlaydi. Shunday ekan, eng avvalo, Siz ulardan qaysidir birini tanlashingiz, kompyuteringizning OTiga, uning ichki qurilmalarining xotirasiga bog'liq bo'ladi. Siz ushbu versiyalarni Microsoftning sayti bo'lmish www.microsoft.comdan uning lisenziyalangan versiyasini ko'chirib olishingiz mumkin. SHuni aytib o'tish kerakki, ba'zi saytlarda uning noqonuniy versiyalari ("qaroqchi versiyalar") ham uchrab turadi. Bu paketlar to'liq holda bo'lmaydi, balki ishlashida ham muammolari bo'lishi mumkin. SHuning uchun faqatgina rasmiy nushasidan foydalangan ma'qul.

Microsoft Visual Studio dasturi **Microsoft** korporasiyasi tomonidan ishlab chiqilgan bo'lib, bu dastur dasturchilar uchun mo'ljallangandir. Bu dastur yordamida quyidagi dasturlash tillarida dasturlashni amalga oshirish mumkin:

- **Visual Basic .NET**
- **Visual C# .NET**
- **Visual C++ .NET**
- **Visual J# .NET**

Microsoft Visual Studio dasturi yordamida Windows muhiti uchun telefonlar uchun va tarmoqlar uchun Web dasturlarni yaratish mumkin **Microsoft Visual Studio** dasturida **Visual C#, Visual Basic, Visual J#** tillari yordamida Web dasturlarni va **Visual C#, Visual Basic, Visual J#, Visual C++** tillari yordamida **Windows** muhiti uchun dasturla yaratish mumkin.

1.2. Texnika kodi va dasturlash tillari

Visual C++ dasturlash muhiti

Hozirda ko'plab dasturlash tillari mavjud bo'lib, ular qo'llanilis sohasiga qarab turlicha bo'ladi, ya'ni har bir soha uchun mo'ljallanga dasturlash tillari mavjud. Ularning bir nechtasini sanab o'tish mumki Masalan, **C#, C++, Visual Basic, JavaScript, Delphi** va boshqalar. Quyida ushbu dasturlash tillari bilan tanishib chiqamiz:

Visual Basic .NET - Microsoft Visual Studio 2016 tarkibida samaradorligi katta dasturlash tillardan biri bo'lib, bu dasturlash tili to'liq ob`ektga yo'naltirilgan dasturlash tili deb aytishimiz mumki **Visual Basic .NET** dasturlash tili yordamida **Windows** ilovalarini Web ilovalarni yaratish mumkin.

Visual C# .NET - **Microsoft** korporasiyasi tomonidan ishl chiqilgan bo'lib, bu dasturlash tili aynan .NET platformasi uchun ishl chiqilgan. **Visual C# .NET** dasturlash tili imkoniyatlari boshqa ob`ekt yo'naltirilgan dasturlash tillari (**C, C++, Java va Delphi**)dan ancha ke bo'lib, bu dasturlash tilida **Visual Basic .NET** kabi **Windows** ilovalar va Web ilovalarni yaratish mumkin.

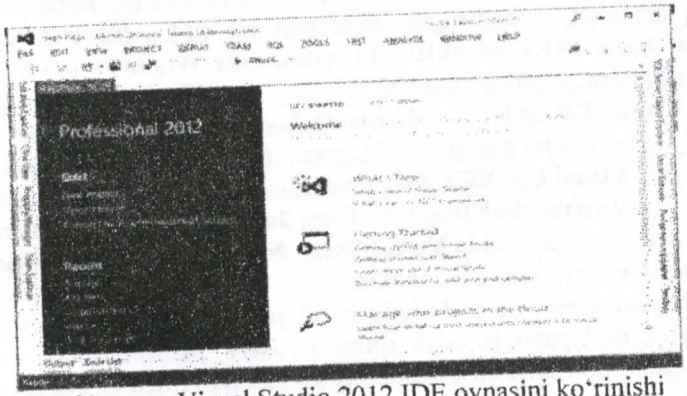
Visual C++ .NET dasturlash tili past sathdagi dasturchi uch ilovalarni boshqarishda talab qilinadi. **.NET** platformasining **Vis C++** dasturini boshqa dasturlardan shu bilan farq qiladiki, bu dasturl tili **.NET** platformasining kodli modeli (**managed code model**)

Windows (unmanaged native code model) kodli modelini qo'llab quvvatlaydi.

Visual J# .NET - Microsoft .NET platformasi uchun Web-servis va ilovalar yaratuvchi Java dasturchilari ishlatishi mumkin.

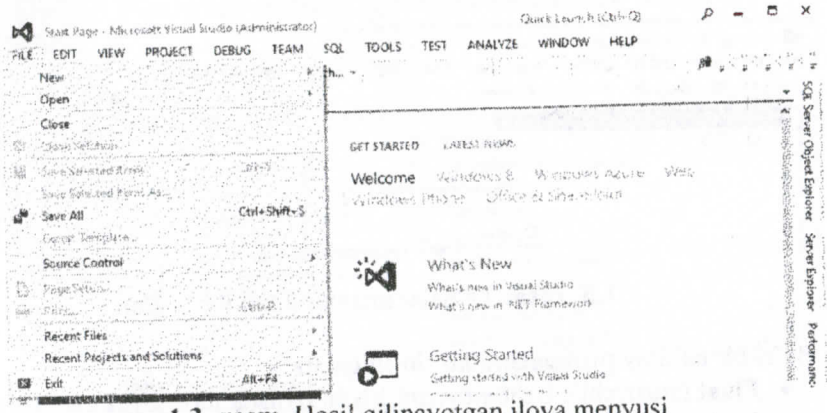
Visual C#.NET dasturlash tili. Zamonaviy dasturlash tillari orasida **Visual C#.NET** mukammal dasturlash tillaridan biri hisoblanadi. Ushbu dasturlash tili C++ asosida kelib chiqqan bo'lib, u **.NET** Microsoft platformasi uchun yaratilgan va ob`ektli dasturlashga mo'ljallangan. Uning yordamida internetda juda ommabop sahifalar, saytlar yaratsa bo'ladi va yaratilmoqda. **Visual C#.NET** dasturlash tilining afzalligi bu boshqa dasturlash tillarida yo'l qo'yilgan kamchiliklardan xoliligidir.

C++ tili C tilining rivojlangan holatidir. Shuning uchun C tilining barcha konstruksiyalari C++ dasturlash tilida o'z ifodasini topadi. Biroq C++ tilida C tiliga qaraganda juda ko'p sintaktik imkoniyatlar paydo bo'ldi (biz buni material bilan tanishish jarayonida ko'rib chiqamiz). Ushbu tilda dasturlash uchun biz **Visual Studio 2016** (ingliz tilidagi abriviaturasi – **IDE: Integrated Development Environment**) dasturi bilan yaqindan tanishishimiz zarur bo'ladi. Shu sababli dastur muhitining strukturasi va undagi interfeys bilan yaqindan tanishamiz. Interfeys – bu shunday qurilmaki, bunda foydalanuvchi muhit bilan muloqot jarayoni osonlashadi. **Visual Studio 2016** dasturini yuklab olib, uni o'rnatganingizdan so'ng -> **Пуск | Программы** paneliga kirib, uning .exe faylini topishingiz mumkin.

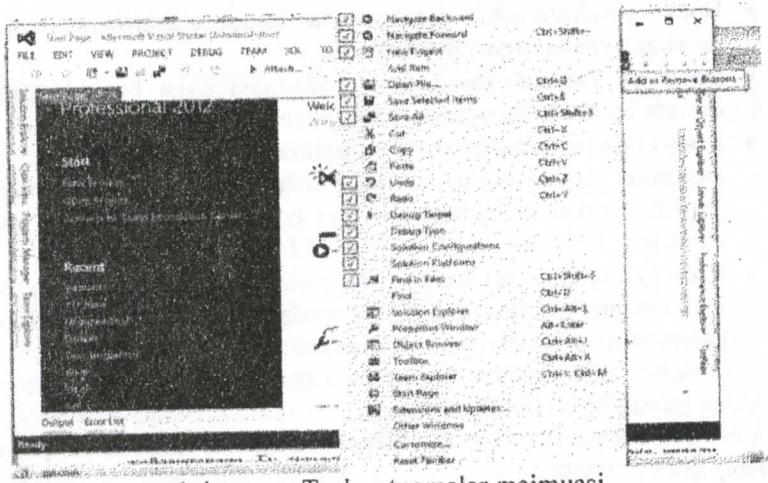


1.2- rasm. Visual Studio 2012 IDE oynasini ko'rinishi

Yuqori qismda **File, Edit** tugmalari joylashgan bo'lib – bu gorizontal menyu hisoblanadi. Ushbu buyruqlarni chaqirish imkoniyatlari juda keng bo'lib, "tushib qoluvchi" oynalar - vertikal menyu hisoblanadi hamda oynaning chap yuqori qismidan boshlab joylashadi va o'zida turli xil buyruqlar majmuasini saqlaydi.



1.3- rasm. Hosil qilinayotgan ilova menyusi

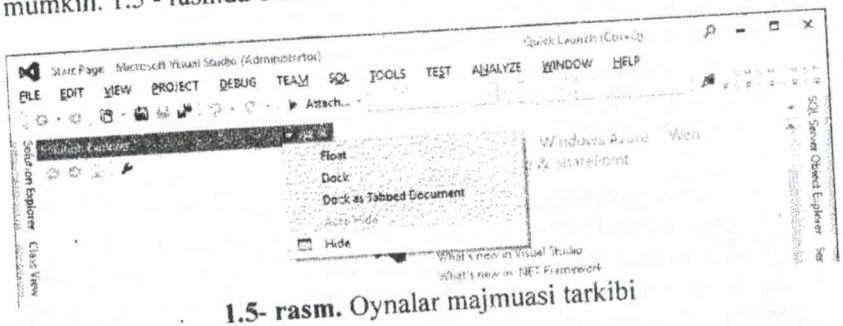


1.4- rasm. Tezkor tugmalar majmuasi

Pastki satrlarda asosiy menyuda **tezkor murojaatlar oynas** joylashganidir. Bu tugmachalar suzib chiquvchi oynalarni hosil qilad

Uning yonida kerakli qo‘shimcha tugmachalar ham joylashgandir. Asosiy oyna ko‘rinishi ilova turiga qarab o‘z holatini o‘zgartiradi.

Ishchi oyna oynalar majmuasidan iboratdir. Har bir oyna – bu odatda **windows** – oyna bo‘lib, u standart sarlavha qatoriga egadir. Bunda sichqoncha orqali sarlavha qatorlarini ixtiyoriy joyga surish mumkin. 1.5 - rasmda **Solution Explorer** oynasi ko‘rsatib o‘tilgan.



1.5- rasm. Oynalar majmuasi tarkibi

Ba’zi bir tarkibiy tuzilmalarni ko‘rib chiqamiz:

- **Float** (suzuvchi). Ushbu oynani ishchi oynada turli joyga cho‘zish mumkin.
- **Dock** (yondashuvchi). Ushbu oyna joyini o‘zgartirish mumkin (boshqa oynalar orqali fiksirlanadi).
- **Dock as Tabbed Document** (yopishgan oyna holatida asosiy oynada bo‘ladi). Bu oynada **Start page** varaqasi mavjud bo‘ladi.
- **Auto Hide** (avtomatik tarzda yo‘qolib qolish). Ushbu holatda oyna avtomatik tarzda yo‘qoladi va ishchi oynaning yon tomoniga yopishib turadi va sichqoncha olib borilganda u birdagina suzib chiqadi.
- **Hide** (yashirish). Bu holatda oyna ekrandan butunlay yo‘qoladi. Uni qaytarish uchun asosiy menyudan **View, Other Windows** parametridan foydalanish zarur bo‘ladi.

Oynalar ustida biror – bir operasiyalarni bajarish uchun ularning tarkibini yaxshi bilish kerak bo‘ladi.

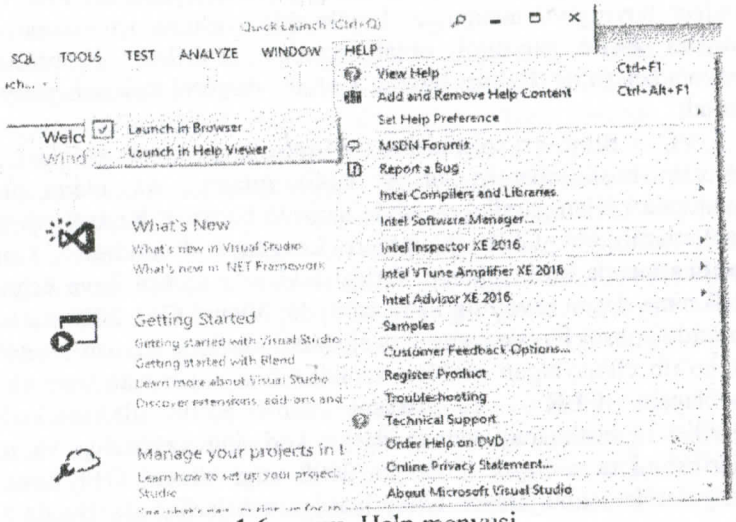
Misol uchun Toolbox va **Server Explorer** ma’lumot-oynasini qanday qilib o‘rnatish?

Ushbu oynalarda tarkibini o‘rnatish (sichqonchaning o‘ng tomonini bosish orqali). Shunda ekranda **Auto Hide** (avtomatik

berkinuvchi) chiqadi va shunga o‘xshash tugmachalarni ishlatish mumkin.

Help yordamchi tizimi haqida ma’lumot

Hammani quyidagi savol qiziqtirishi tabiiy: “Bu qanday amalga oshiriladi?”. Ma’lumki, yordamchi ma’lumotlar oynasi tizim dasturida mavjud. Buning uchun **Help** tugmasi tanlanadi (**MV Visual C++** 1.6 - rasm)



1.6- rasm. Help menyusi

Visual C++ dasturi strukturasi

Visual C++ dasturlash muhitidagi dasturlar ilovalar deyilad (ko‘rinib turibdiki, IDE dasturlash muhiti). Biz ularni kelgusida Visu C++ ilovalari deb ataymiz. Ushbu ilovalar konstruksiya ko‘rinishi tuziladi va ular - loyihalar deyiladi. Bu bir necha fayllarning ketma- k jamlanmasi hisoblanadi. C++ dasturlash tilidagi dasturlar bu funksiyal jamlanmasi hisoblanib, kerakli majburiyatlarga ega bo‘ladi. Ilova – asosiy funksiya bo‘lib, uning ichida shunday operatorlar mavjudki, dastur ishlashini realizatsiya qiladi. Har bir dastur o‘z ishini asos

funksiyadan boshlaydi va bu funksiyaning bir qismi dasturchi tomonidan tuziladi, qolgan qismi esa – kutubxona, ya'ni sarlavha funksiyalari tomonidan dasturlash jarayonida foydalanuvchiga uzatiladi. C/C++ tilini o'rganishda biz maxsus ilovalar turlaridan foydalanamiz – konsol rejimidagi ilovalarda ishlash, ya'ni oldindan hosil qilingan shablonlar asosida hosil qilinadigan holatlar bilan tanishamiz.

Konsol oynasi – bu grafik interfeysga ega bo'lmagan, dastur bilan foydalanuvchi orasidagi buyruqlar oynasi orqali hosil qilinadigan natija oynasidir. Buning uchun biz birinchi dialoglar oynasidan **File| New | Project** buyrug'ini tanlaymiz. Loyihaning yig'ilish va kompilyatsiya jarayoni **Build** buyrug'i orqali amalga oshiriladi. Kompilyatsiya jarayonidan so'ng **Debug** buyrug'i orqali dasturni bajarish jarayoniga o'tiladi.

VC++ tilini o'rganishni biz turli misollarda ko'rib chiqamiz, turli dasturlar hosil qilamiz, ularni tahlil qilamiz, va ularni ishlash strukturalari bilan parallel ravishda tanishib boramiz. Konsol rejimidagi shablonlarni biz CLR (Common Language Runtime) **Console Application** da hosil qilamiz. Oxirgi ikki so'z konsol ilova deganidir. **CLR** nima degan savol tug'ilishi tabiiydir. **Visual C++ 2008** dasturlash muhitida konsol ilovalarining 2 ta shabloni mavjud: ulardan birinchisi – biz ko'rib chiqayotgan shablon va ikkinchisi – abriviaturasiz shablon hisoblanadi. **CLR** – bu maxsus muhit bo'lib, dastur kodining bajarilishini boshqaradi hamda dastur kodining xavfsizligi va to'g'ri bajarilishini ta'minlaydi. 2005 va 2008 yilgi **Visual C++** dasturlash versiyalarida ushbu jarayon kompilyatsiyada bajarilar edi. Bunda dastur uchun "to'plam" degan xotira ajratilar edi: unda biz ob`ektni joylashtirish, xotirani bo'shatish, ob`ekt bilan ishlash vaqti kabi parametrlarni bajarar edik. Boshqa tomondan, **CLR** rejimi yoqilganda, bu ilova boshqa ilovalarga nisbatan tubdan farq qiladi, bunda ilovaga ulanish **System** muhiti orqali amalga oshiriladi, bunda ob`ektlarning xotiraga joylashishi avtomatik boshqariladi.

Regulyasiyalanuvchi ko'rsatkich – bu shunday ko'rsatkich turiki, bunda havola orqali ob`ektning "to'plam" xotiradagi o'rni ko'rsatiladi, bu holat sinov paytida amalga oshiriladi. Bunday ko'rsatkichlar uchun "*" o'rniga "^" belgisi ishlatiladi. Xullas, **CLR** o'zgaruvchilarni hosil qilish apparatini, ya'ni bir to'plamga tegishli xotira adresi va h.k. yaratib beradi. Shunday maxsus kutubxona borki, bu jarayonni boshqaradi. Lekin bu narsalar dasturlashni juda qiyinlashtirib yuboradi.

Biz bu bo'limda muhitning bosh oynasini hamda loyihalarda ishlatiladigan asosiy formalar va ularning sifatli chiqishi uchun zaru bo'lgan maxsus konstruksiyalarni qarab chiqamiz. Bu yerda han konsolli ilovalarda o'rganilgan tushunchalardan foydalaniladi. Bu v bundan keyingi bo'limlarda biz murakkab ilovalarning grafik interfeys bilan tanishamiz. Ilovalarni yaratishda forma tushunchasini ishlatamiz Bosh oynaning chap qismida ikki bo'lakli vkladka joylashgan. Birinch qism ma'lumotlar bazasi bilan ishlashni tashkil etsa, ikkinchi qismd komponentalar ro'yxati keltirilgan.

Bosh oyna (ishchi stol) ning har bir qismi to'g'risidagi to'liqro ma'lumotlarni biz keyingi boblarimiz davomida ko'rib chiqamiz.

Agar ishchi stolimizning strukturasiga e'tibor bersak, har bi oynaning sarlavhasi o'z tarkibida bo'lgan ko'plab funksiona masalalarga mos ravishda tanlangan. Bu oynalarni sichqonchaning cha tugmasi yordamida istalgan joyga joylashtirish mumkin. Oynalarn o'zaro birlashtirib, bir nechta vkladkalar ko'rinishida ham joylashtirs bo'ladi. Siz agar biror bir oynani tanlab, sichqoncha yordamida kerakl joyga o'rnatmoqchi bo'lsangiz, darhol sizga kerakli ob`ektni tanlas imkonini beruvchi chizmalar beriladi. Siz ulardan birini tanlab, kerakl joyga oynani o'rnatishingiz mumkin.

Demak, birinchi ish muhitning oynalarini o'zingizga qulay tarzd o'rnatishdir. Aslida bu ish oddiy ko'ringani bilan, lekin keyinchalik is unumdorligiga juda katta ta'sir qiladi, ya'ni vaqtdan yutish imkonin beradi. Loyihalaringizni tayyorlayotganda barcha oynalar sizning qo' ostingizda qulay tarzda joylashgan bo'lsa, ish sifati ham oshadi.

Bu oynalar bilan ishlash davomida go'yoki mashhur dasturlarda biri bo'lmish Photoshop oynalari yodga keladi. Bu oynalarning xossala bir-biri bilan uzviy bog'liq. Oynalarni keraklicha joylashtirishn muhitning *Bud* menyusi orqali ham amalga oshirish mumkin.

Tasodifiy fakt 1.3. Standartlovchi tashkilotlar

Amerika milliy standartlash instituti (ANSI - American Nationa Standards Institute) va Halqaro standartlovchi tashkilot (ISO - International Organization for Standardization) hamkorlikda C++ til uchun eng to'g'ri standartni vujudga keltirishdi. Nima uchun standart kerak? Siz standartlashni foydasi bilan har kuni to'qnash kelasiz

Lampochka sotib olayotganda u uyingizdagi lampochka chanog'iga mos kelishini bilasiz. Fakt shuki, siz qachondir nostandart fonar lampochkalarini xarid qilsangiz, standartlash qanchalik muhimligini bilib olasiz. Bunda fonar va lampochkalarni qayta almashtirish qiyin va qimmat bo'lishi mumkin.

ANSI va ISO standartlash tashkilotlari mashina balonlari va kredit kartalari shaklidan to C++ dasturlash tiligacha hamma narsalarni standartlashni yo'lga qo'ygan ishlab chiqarish mutaxassislarining birlashmasidir. Bu Siz bir sistemada o'rnatgan dasturni boshqa ishlab chiqaruvchi to'plamidagi boshqa dasturga qo'yganingizda ham uning ishlashiga amin bo'lishingizdir.

1.4. Dasturlash muhiti bilan tanishish

Ko'pchilik talabalar dasturchilarga kerak bo'ladigan qurilmalar ularga tanish bo'lgan dasturiy ta'minot qurilmalaridan farq qilishini yaxshi bilishadi. Siz alohida vaqt ajratib dasturiy muhit bilan tanishib chiqing. Chunki kompyuter tizimi keng farqlanadi, bu kitob sizga amal qilishingiz kerak bo'lgan bosqichlar yo'riqnomasini beradi. Turli xil amaliy jarayonlarda ishtirok etish yoki bilimga ega do'stingiz yo'rig'ini tinglash ham samaralidir. Kompyuter tizimlari bu borada katta farq qiladi. Ko'p kompyuterlarda yozish va dasturlarni sinash mumkin bo'lgan integrasiyalashgan ishlab chiqish muhiti mavjud. Boshqa kompyuterlarda esa C++ yo'riqnomalarini kiritish uchun avval so'z muharririni ishga tushirish lozim, keyin esa konsol oynasini ochish va bajarilishi kerak bo'lgan buyruqni kirgizishingiz kerak. Siz muhit bilan ishlashni o'rganishingiz lozim. Yangi dasturlash tilidagi yaratilgan ilk dastur – bu ko'pincha ekranda oddiy salomlashuv so'zi "Hello World!" ni chiqarish boshlanadi. Keling, bu an`anaga Biz ham amal qilaylik. Bu yerda C++ "Hello World!" dasturi:

```
1. // 1-misol.cpp : main project file.
2. #include "stdafx.h"
3. using namespace System;
4. int main(array<System::String ^> ^args) {
5. Console::WriteLine(L"Hello World");
6. return 0; }
```

Dasturlash maslaxati

1.5. Zahira ko'nikmalari

Biz keyingi bo'limda ushbu dasturni ko'rib chiqamiz. Qanday dasturiy muhitdan foydalanmang, dastur holatini muharrir oynasiga kiritishdan boshlaysiz.

Dastur buyruqlarini tepada berilgandek aniq kirgizing. Shu bilan bir qatorda, dasturdagi manbaa fayllarini elektron nushasini toping va muharriringizga qo'ying.

Siz ushbu dasturni yozar ekansiz, turli simvollarga yaxshilab e'tibor bering, Visual C++ da dastur tuzish nozik ish sanaladi. Siz xarflarni dastur satrida ko'ringanidek bosh va kichik xarflarda kirgizishingiz shart. **MAIN** yoki **CONSOLE** qilib yoza olmaysiz. Agar siz e'tiborli bo'lmasangiz odatiy xatoga yo'l qo'yasiz.

C++ da nozik jarayonlar. Siz katta va kichik xarflarni tanlab yozishda e'tiborli bo'lishingiz kerak

Ixtiyoriy C++ dasturi main funksiyaga ega bo'lishi shart.

Dasturingiz tuzilayotganda uning negizida nima borligi muhimdir. Birinchidan, kompilyator C++ manbaa kodini (siz kiritgan so'zlar) mashina ko'rsatmalariga o'girib beradi. Mashina kodi Siz yozgan so'zni kodga o'girib berilgan shaklidir. Dasturni faqatgina ishga tushirish yetarli bo'lmaydi. Kompyuter oynasida bir tizimni ko'rinishi uchun kam darajada bo'lsa ham faollik muhim. **Visual C++** da muhit amaliyotchilari **Console::WriteLine()** va uning vazifalarini o'z ichiga olgan kutubxona bilan ta'minlab beradi. Kutubxona bu boshqa bir inson tomonidan dasturlangan va tarjima qilingan, siz qo'llashingiz uchun tayyor dastur kodlari jamlanmasidir. Birlashtirgich nomli dastur mashina kodini va C++ kutubxonasidan kerakli qismlarni birlashtirib, amalga oshgan faylni yuzaga keltiradi. (1.7- rasmda ushbu bosqichlarning ko'rinishi tasvirlangan). Amalga oshgan fayl kompyuter tizimiga bog'liq ravishda **1-misol.exe** deb nomlanadi. Siz amalga oshgan faylni **Visual Studio** dasturidan chiqib ketganingizda ham ishga tushirishingiz mumkin.

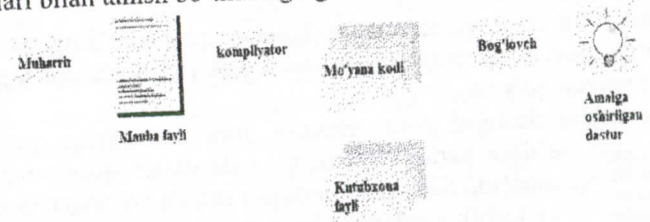
Kompilyator C++ dasturini mashina tiliga o'girib beradi

Qiymat ekranda ko'rinishi uchun **Console:: WriteLine()** funksiyasidan foydalaning.

Birlashtirgich mashina kodi bilan kutubxona kodini amalga oshgan dasturda birlashtiradi.

Ishingizni tashkillashtirish: dasturchi sifatida siz dastur tuzasiz sinaysiz va takomillashtirasiz, dasturlaringizni fayllarda saqlaysiz. Ayrim tizimlarda fayllarning nomlari orasida bo'sh joy qoldirishga ruxsat beriladi, ayrimlarida esa yo'q. Ayrimlarida katta va kichik

harflar farqlanadi, ayrimlarida esa yo‘q. Fayllar papkalar yoki kataloglarda saqlanadi. Papkalar o‘z ichiga fayllarni oladi va yana ichida fayllar va papkalar bor boshqa fayllarni ham saqlay oladi(1.7-rasm). Bu ierarxiya katta bo‘lishi mumkin va Siz uning barcha bo‘limlari bilan tanish bo‘lmasligingiz mumkin.

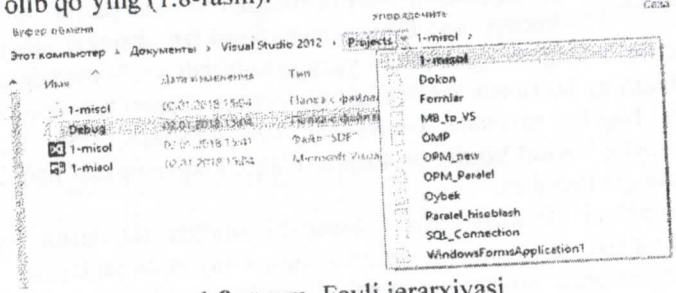


1.7- rasm. Amalga oshgan fayl manba kodi

Biroq siz ishingizni tashkil qilishingiz uchun papkalar yaratishingiz mumkin. Har bir dasturlash darsi uchun alohida papka ochish to‘g‘ri fikrdir. Papka ichida esa har bir topshiriq uchun alohida papka oching.

Ayrim dasturlash muhiti papkani o‘zingiz joylashtirmasangiz standart manzilga saqlaydi. U holda siz papka joylashgan joyini aniqlashingiz kerak bo‘ladi. Ierarxiya papkasida faylingiz qaerga joylashganini bilishingizga ishonch hosil qiling. Bu ma’lumot papkalarni guruhlayotganingizda va zahira nusxa olayotganingizda muhimdir.

Siz ko‘p vaqtingizni C++ dasturini yaratish va takomillashtirishga sarf qilasiz. Bunda ayrim fayllar kompyuter nosozligi uchun o‘chib ketishi mumkin. Qaytadan yo‘qolgan fayllarni tiklamaslik uchun ishlaringizni xotira kartasiga yoki boshqa kompyuter xotirasiga zahira nusxa olib qo‘ying (1.8-rasm).



1.8- rasm. Fayli ierarxiyasi

Odatiy xato

Nuqtali vergul tushib qolishi. C++ da har bir vaziyat nuqtali vergul bilan tugashi kerak. Nuqtali vergul tushib qolishi odatiy xato sanaladi. Bu kompilyatorni adashtiradi, chunki kompilyator qaerda yangi jarayon boshlanib, qaerda tugashini nuqtali vergul anglatib turadi. Kompilyator xech qachon jarayon so‘nggini anglatish uchun tugatuvchi satr yoki yopiluvchi qavs belgisini ishlatmaydi. Masalan, kompilyatorda yagona jarayon

```
Console::WriteLine(L"Hello World");
return 0;
```

agar siz quyidagicha yozgan bo‘lsangiz,

```
Console::WriteLine(L"Hello World"); return 0;
```

shundan so‘ng jarayonni tushunmaydi, chunki return so‘zini chaqiruvchi buyruq markazida turishini tushunmaydi. Masala yechimi oddiy. Har bir ingliz tilida berilgan gapda nuqtali vergul qo‘yilganini tekshiring.

Maxsus matn

Chiqarish ketma-ketligi: Qo‘shtirnoq belgisidan foydalanilgan zanjirni qanday ko‘rsatish mumkin

```
Console::WriteLine ("Hello World");
```

Lekin quyidagicha foydalana olmaysiz,

```
Console::WriteLine ("Hello"," World");
```

Kompilyator “Hello,“ ni o‘qiganidan so‘ng, u buni zanjir tugadi deb tushunadi va keyingi World xaqida chalkashlik yuzaga keladi. Kompilyator bir yo‘lli miyaga ega, u kiruvchi ma’lumotni oddiy tahlilini bajarib, oldinga yurmaydi va xato ko‘rsatadi. Inson bilan solishtirganingizda, u qo‘shtirnoq qaerda tizim so‘nggini anglatishini farqlaydi. Ekranda qo‘shtirnoq belgisini qanday ko‘rsatish mumkin? C++ dizaynerlari bu holatdan chiqish yo‘lini topishdi. Xar bir qo‘shtirnoq oldiga (\) belgisini qo‘yishgan.

Console::WriteLine() funksiyasini ishlatish uchun using namespace System; e’lon qilinishi kerak.

```
Console::WriteLine(L"Hello\" World\"");
```

Bu holat \" belgi aslida tizim tugamaganligini anglatadi. YAna bir qancha usulda ketma ketlikdan qochish mumkin. Agar siz (\) belsini ishlatmoqchi bo‘lsangiz, \\ ketma ketlikdan qochishdan foydalaning.

1.6. Xatolar

Dasturlash tili juda ham muhim konvensiyalarga asoslanadi. Siz bir inson bilan muloqot qilayotganingizda bir yoki ikki so'zni o'tkazib yuborsangiz yoki tushirib qoldirsangiz Siz bilan muloqotdagi inson nima demoqchi ekanligingizni tushuna oladi. Ammo **Visual C++** da xatoga yo'l qo'ysangiz kompilyator noto'g'ri tushunchani qabul qilmaydi. (Bu xaqiqatda yaxshi narsa, agar kompilyator noto'g'ri tushunchani qabul qilganda, u natijani ham noto'g'ri taqdim qilar edi. Bu esa falokatli oqibatlarga olib kelar edi.) Bu bo'limda siz dasturingizdagi xatolarni qanday bartaraf etishni o'rganasiz. **hello.cpp** dasturi bilan tajriba qilamiz. Biz quyidagi xatolarga yo'l qo'ysak nima sodir bo'lar edi. Birinchi holatda, kompilyator orqali nimani nazarda tutayotganingizni tushunmaganligidan arz qiladi. Yo'l qo'yilgan xatolikning aniq ta'rifi kompilyatorga bog'liq bo'ladi. Bu "**In defined symbol**" kabi ko'rinishda ham bo'lishi mumkin.

Bu **compile time** xatolik yoki sintaksis xatolik sanaladi. Imlo qoidasiga yo'l qo'yilsa ham kompilyator uni topadi. Agar kompilyator bir yoki undan ko'p xatolikni topsa, u holda dasturni mashina tiliga o'girmaydi va natijada ishga tushiriladigan dastur ham yaratilmaydi. Siz xatolikni bartaraf etish uchun uni qaytadan to'plashingiz kerak bo'ladi. Odatda ilk muvaffaqiyatli kompilyatsiyaga erishishdan oldin **compile time** xatolikni bartaraf etishdagi bir qancha jarayonlardan o'tiladi.

Dasturchi run-time error ga yo'l qo'ymaslik uchun tekshirishga javobgar.

Agar kompilyator xatolikni aniqlasa, u osonlikcha to'xtamaydi va rad etmaydi. U aniqlagan xatolikni hammasini ko'rsatadi, siz bunda hamma xatolikni bittada to'g'irlab olishingiz mumkin. Ayrim paytlarda bitta xatoni o'zi ham dasturni ishdan chiqarishi mumkin. Bunday xatolik ikkinchi satrda ham uchrashi mumkin. Dasturchi yopuvchi qavs belgisini ishdan chiqarsa, kompilyator satr oxirini qidirishda davom etadi. Bunday holatlarda kompilyator qo'shni qatorlarda sohta xatolikni ko'rsatadi. Siz kerakli qatorlardagi xatolikni to'g'irlab so'ng qaytadan kompilyatsiyalashingiz kerak. Uchinchi satrdagi xatolik boshqacha

bo'ladi. Dastur kompilyatsiyalanadi va ishga tushadi, ammo chiqarilgan ma'lumot noto'g'ri bo'ladi. **RUN TIME ERROR** sababli dastur mantiqiy nuqson aniqlaydi va bu nuqsonlar mantiqiy xatolik deyiladi. Ayrim **RUN TIME ERROR** xatoliklari jiddiy hisoblanadiki, ayrim istisnolarni keltirib chiqarishga ham sababchi bo'ladi. Prosessordag xabar xato xabar sababli dasturni tugatilishiga olib keladi. Misol uchun dasturingiz `Console::WriteLine("Hello\" World\"");` bo'ls "nol bilan ajratish" istisnosi bilan tugallanadi.

Dasturni rivojlantirish mobaynida, xatolikdan qochish qiyin. Aga dastur bir necha qatordan ko'p bo'lsa, kirgizish uchu aqlbovar qilmas diqqat e'tiborni talab qiladi. Si vergul va qavslarni o'zingiz bilmagan tarzda tushiril qoldirasiz, lekin kompilyator sizga ushb muammolarni qidirib topib beradi.

compile-time error -tomonidan to'xtatiladigan dasturlash tili qoidalari buzilishidir.

RUN TIME ERROR muammolidir. Kompilyator ularni top olmaydi, kompilyator gap tuzilishi xatosiz bo'lsa, istalgan dasturni tarjima qila oladi, lekin natijadagi dastur xato bo'ladi. Dastur muallifi **RUN TIME ERROR**ni topishga va dasturni tekshirishga ma'suldir. Dastur tekshiruvi ushbu kitobda muhim matnlardan biridir.

run-time error - dasturchi o'ylamagan xolatlarni keltirib chiqaradi.

1.7. Odatiy xato

Agar siz mabodo, so'zni noto'g'ri talaffuz qilsangiz, g'alati holat yuz berishi mumkin, va nima xatolik yuz bergani xaqidagi xabardan voqif bo'lmasligingiz mumkin. Quyida oddiy talaffuz xatolari qanday muammoga olib kelishiga misol berilgan:

```
// 1-misol.cpp : main project file.
1. #include "stdafx.h"
2. #include "conio.h"
3. using namespace System;
4. int Main(array<System::String ^> ^args)
5. {
6. Console::WriteLine("Hello\" World\""); getch(); }
```

Bu kod **Main** nomli funksiyani anglatadi. Kompilyator Mai funksiasiga aynan o'xshashligini ko'rib chiqmaydi, chunki Main katt

harf bilan yozilgan va C++ da nozik ish bu katta harf va kichik harf tubdan farq qiladi va kompilyatorga Main uchun Main ni tanlash yaxshiroq. Kompilyator **Main** funksiasidan arz qiladi, ammo bog'lovchi amalga oshgan faylni tuzishga tayyor bo'lganda Main funksiyasi yo'qligidan shikoyat qiladi. Albatta yo'qolgan **Main** funksiyasi xatolikni qaerdan topishni ko'rsatadi.

Agar xato xabar kompilyatorni noto'g'ri yo'ldaligini ko'rsatsa, talaffuzdagi xatolikni tekshirish va bosh harf bilan yozish kerak. C++ da ko'p nomlar kichik xarf bilan yoziladi. Agar nomlanishda talaffuz xatoligiga yo'l qo'ysangiz, (misol uchun **Console** o'rniga **console**), kompilyator "noaniq belgi" ko'rsatadi. Bu xatolik talaffuzdagi xatolik sanaladi.

1.8. Muammo yechimi: algoritm konstruksiyasi

Siz tez kunda hisoblarni va qaror qabul qilishni C++da qanday dasturlashni o'rganasiz. Biroq keyingi bobdagi hisoblarni amalda qo'llash mexanizmini ko'rib chiqishdan oldin, keling, ijrodan keyin keladigan rejalashtirish jarayonini ko'rib chiqamiz. Balki sizga mos umr yo'ldosh topib beruvchi kompyuterlashgan xizmat uchun sizni undaydigan e'longa duch kelgandirsiz.

U qanday ishlashi mumkinligi to'g'risida o'ylab ko'ring. Siz anketa to'ldirasiz va uni jo'natasiz. Boshqalar ham shunday qiladilar. Ma'lumotlar kompyuter tomonidan ishlab chiqiladi. Kompyuter Sizga eng mos shaxsni topish vazifasini uddalay oladi deb o'ylash to'g'rimi? Deylik kompyuter emas, ukangizda barcha anketalar bor. Unga qanday ko'rsatmalar bera olardingiz? Siz unga "Konkida uchishni va internetda o'tirishni yoqtiradigan juda chiroyli bo'lgan qarama –qarshi jinsdagi shaxsni top", deb aytolmaysiz. Go'zal chehra borasida ob`ektiv standart yo'q va ukangizning fikri (yoki raqamli rasmni tahlil qilgan kompyuterning dasturining fikri) siznikidan farqli bo'lishi mumkin. Agar siz biror kimsaga muammoni hal etishi

uchun yozma ko'rsatma bera olmasangiz, uni kompyuterga tushuntura olmaysiz. Kompyuter faqatgina unga nima qilishni aytsangiz, shuni bajara oladi. U zerikmasdan yoki charchamasdan vazifani tezroq bajaradi. Endi quyidagi sarmoya borasidagi muammoni ko'rib chiqamiz. Siz bank hisob raqamingizga yiliga 5% foyda qiladigan $10,000 miqdordagi pulni qo'ydingiz. Dastlabki miqdor ikki baravar ko'payish uchun hisob balansi uchun qancha yil talab etiladi? Ushbu muammon qo'llar yordamida yecha olasizmi? Albatta, siz balansni quyidagicha hisoblaysiz:

Balans kamida $20,000 miqdoriga yetmaguncha siz hisobni davom ettirasiz. Yil ustunidagi oxirgi raqam javob bo'ladi.

Albatta, bu hisoblashni amalga oshirish siz va ukangiz uchun juda zerikarlidir. Lekin kompyuterlar takrorlanuvchi hisoblarni tez va mukammal bajarishda ustasi farangdirlar. Kompyuter uchun muhimi bu yechimni topishda bosqichlarning tavsifidir. Har qaysi qadam taxminda uzoq, aniq va lo'nda bo'lishi shart. Mana quyidagicha tavsif: Yil qiymatini 0 bilan, foyda va $10,000 lik balans uchun ustunlardan boshlang.

Balans kamida $20,000ga yetmaguncha quyidagi qadamlarni qaytaring. Yil qiymatiga birni qo'shing. Foydani **balance x** 0.05 (ya'ni 5% foyda) deb hisoblang. Foydani balansga qo'shing. Oxirgi yil qiymati javobligi to'g'risida xisobot bering. Albatta, bu qadamlar kompyuter tushuna oladigan tilda emas, lekin siz tez orada ularni C++da qanday ifoda etishni o'rganib olasiz. Bunday norasmiy tavsif psevdokod deb ataladi. Psevdokod uchun qat`iy talablar yo'q, chunki uni kompyuter emas, odamlar o'qishadi. Bu yerda bu kitobda foydlanadigan psevdokod operatorlarining turlari berilgan.

Har qaysi bosqichda nima qilish va keyin qaerga borish kerakligi to'g'risida aniq ko'rsatmalar bo'lsa, bu aniq metoddir. Taxmin yaratuvchanlikka joy yo'q. Har qaysi bosqich amalda bajarilganda usul amalga oshuvchi metoddir. Agar bizdan yiliga 5 foizli an qiymatdan emas, yaqin yillarda olinadigan haqiqiy foiz qiymatid foydalanish so'ralsa, bizning usul amalga oshadigan metod bo'lmay chunki aynan o'sha foyda qiymatini hech kim hech qanday usulda b olmaydi. Agar usul nihoyasiga yetsa, u tugallanadigan metodd Bizning misolda metod uzluksiz davom etmasligini ko'rish ozgi fikrlashni talab etadi: Har qaysi bosqichda balans kamida $5 miqdoriga oshmoqda va shunday qilib u natijada $20,000 miqdori yetishi shart.

Aniq, amalga oshuvchi va yakunlanuvchi bosqichlarning ketma - ketligi algoritm deyiladi. Sarmoyamizdagi muammoni hal qilish uchun algoritm ishlab chiqdik va u orqali dasturlashni amalga oshirimiz va muammoning yechimini topishimiz mumkin. Algoritmni mavjudligi vazifani dasturlashning muhim talabidir. Dasturlashni boshlashdan oldin birinchi bo'lib, Siz hal qilishni xohlagan vazifa uchun algoritm yaratishingiz va tavsiflashingiz kerak.

Qanday qilib

Psevdokodli algoritmni tavsiflash:

C++ dasturini yozishdan avval siz muayyan muammoga yechim topish usuli algoritmni ishlab chiqishingiz kerak. Psevdokodni algoritmda tavsiflang: Ingliz tilida ifodalangan aniq qadamlar ketma- ketligi.

Masalan, bu muammoni ko'rib chiqing. Sizda ikkita mashina sotib olish tanlovi bor. Biri yoqilg'idan foydalanish samaradorligini tejaydi, lekin qimmatroq. Sizga ikkalasining narhi va yoqilg'i sarfi ma'lum. Siz avtomobildan 10 yil davomida foydalanishni reja qilgansiz. Bir gallon uchun yoqilg'ining narxi $4 va yiliga 15,000 mil foydalanishni taxmin qiling. Siz mashina uchun naqd to'laysiz va xarajatlar haqida qayg'urmaysiz. Qaysi mashinani sotib olgan ma'qulroq?

1- qadam. Kiritish va chiqarish ma'lumotlarni aniqlang.

Namunadagi muammoda bizda ushbu kiritish ma'lumolari bor:

• **purchase price1** va **fuel efficiency1**, birinchi mashinaning narxi va yoqilg'i xarajati.

• **purchase price2** va **fuel efficiency2,** ikkinchi mashinaning narxi va yoqilg'i xarajati qaysi mashinani sotib olish yaxshiroqligini bilishni xoxlaymiz. Bu istalgan natija.

2- qadam. Muammoni kichik vazifalarga bo'ling.

Har bir mashinaning umumiy harajatini bilishimiz kerak. Keling bu hisobni alohida har biri uchun amalga oshiraylik. Har birining umumiy narxini bilgandan so'ng, biz qaysi birini sotib olish ma'qulroqligini aniqlashimiz kerak.

Har qaysi mashina uchun umumiy narx **purchase price** +

operating cost. Biz o'n yil doimiy foydalanish va yoqilg'ining narxini taxmin qilamiz, shunday qilib foydalanish xarajati mashinani bir yil davomidagi xarajatlariga bog'liq. Foydalanish xarajati 10 x **annual fuel cost.**

Bir yillik yoqilg'i xarajati **price per gallon x annual fuel consumed**. Bir yillik yoqilg'i xarajati **annual miles driven / fuel efficiency**. Masalan, siz mashinada 15,000 mil yursangiz va yoqilg'i samaradorligi 15 milg` gallon bo'lsa, mashina 1.000 gallon yoqilg'ini sarflaydi.

3- qadam. Har qaysi vazifani psevdokodda tavsiflang

Tavsifingizda qadamlarni shunday joylashtiringki, har qanday oraliq qiymatlar boshqa hisoblarga kerak bo'lishidan oldin hisoblansin. Masalan, **operating cost** ni hisoblab bo'lgandan so'ng qadamni sanab chiqing **total cost = purchase price + operating cost**

Qaysi mashinani sotib olish kerakligini qaror qilish algoritmi: For each car, compute the total cost as follows:

annual fuel consumed = annual miles driven / fuel efficiency

annual fuel cost = price per gallon x annual fuel consumed

operating cost = 10 x annual fuel cost

total cost = purchase price + operating cost

If total cost1 < total cost2

Choose car1. Else Choose car2.

4- qadam. Muammoni bajarib psevdokodingizni tekshiring.

Ushbu namunadagi qiymatlardan foydalanasiz:

Car 1: $25,000, 50 miles/gallon

Car 2: $20,000, 0 miles/gallon

Birinchi mashina uchun hisob - kitob:

ëannual fuel consumed = annual miles driven / fuel efficiency = 15000 / 50 = 300

annual fuel cost = price per gallon x annual fuel consumed = 4 x 300 = 1200

operating cost = 10 x annual fuel cost = 10 x 1200 = 12000

total cost = purchase price + operating cost = 25000 + 12000 = 37000

Xuddi shunday, ikkinchi mashinaning umumiy narxi $40,000 SHunday qilib, algoritmning natijasi birinchi mashinani tanlash.

Bob xulosasi

Algoritm va uning bosqichlari, dasturlash va uning turlari, kompilyatsiya tushunchalari, kompilyator va **Visual Studio 2012** dasturi haqida ma'lumotlar keltirildi. **Visual C++**da dastur tuzish va unda yuzaga keladigan xatoliklar bilan tanishildi.

Nazariy savollar va amaliy topshiriqlar

1. Kompyuter dasturidan foydalanish va kompyuterni dasturlash o'rtasidagi farqni tushuntiring.

2. Kompyuterning qaysi qismlari dastur kodini saqlay oladi? Qaysi biri foydalanuvchining ma'lumotini saqlaydi?

3. Kompyuterning qaysi qismlari foydalanuvchiga ma'lumot berish uchun xizmat qiladi? Qaysi biri foydalanuvchining kiritgan ma'lumotini oladi?

4. Toster bitta funksiyali qurilma, lekin kompyuter turli hil vazifalarni amalga oshirish uchun dasturlanishi mumkin. Sizning mobil telefoningiz bitta funksiyali qurilmami yoki u dasturlanganmi? (sizning javobingiz mobil telefoningizning modeliga bog'liq)

5. Mashina kodidan C++ning afzalligini tushuntiring.

6. Kompyuteringizda papka yoki katalog nomining aniq joyini toping.

7. Bu dastur nimani chop etadi?

```
#include <iostream>
using namespace std;
int main(){
 Console::WriteLine("6 * 7 = " + 6 * 7 );
 return 0;}
```

8. Bu dastur nimani chop etadi?

```
#include <iostream>
using namespace std;
int main(){
Console::WriteLine("Hello" << "World");  return 0;}
```

Bo'sh joylarga yaqindan diqqatingizni qarating.

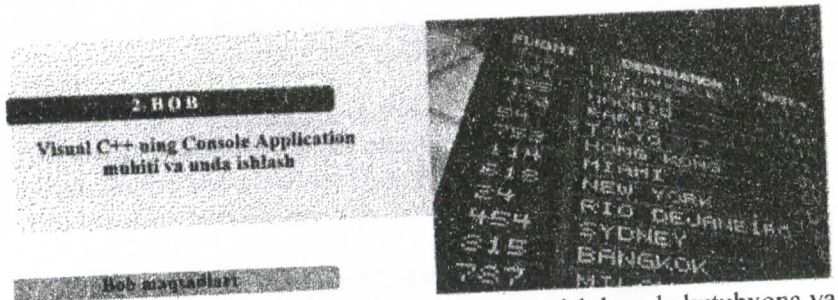
9. Bu dastur nimani chop etadi?

```
#include <iostream>
```



```
using namespace std;
int main(){
Console::WriteLine("Hello"+"\nWorld");  return 0; }
```

10. Turlicha kompilyatsiya xatosiga ega hello.cpp dasturining uch xil ko'rinishini yozing. Ishlashdagi xatoga ega ko'rinishini ham yozing.



Visual C++ muhitida konsol ilovalar yaratish hamda kutubxona va operatorlardan foydalanish ko‘nikmalarini egallash. Ob`ektga yo‘naltirilgan dasturlash bo‘yicha nazariy va amaliy ko‘nikmalarga ega bo‘lish.

Bob mundarijasi

2.1. Visual Studio 2012 dasturini tizimga o‘rnatish

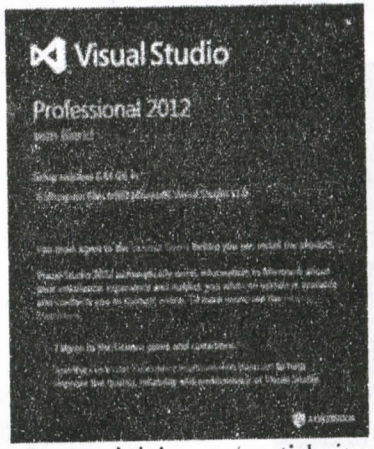
Visual Studio dasturiy paketi **Microsoft** firmasining mahsuloti hisoblanadi. U bir necha yillardan beri dasturchilarning asosiy ish quroli bo‘lib kelmoqda. Ushbu dastur orqali nafaqat C++ dasturlash tilida, balki boshqa tillar (C#, F#, FoxPro va boshqalar)da ham dastur tuzish mumkin bo‘ladi. Shu sababli bu dastur biroz og‘irroq yuklanadi (katta hajmdagi tezkor xotirani talab qiladi). Visual Studio dasturini bir necha xil versiyalari mavjud bo‘lib, hozirgi kunda **Visual Studio 2017** versiyasi chiqmoqda. Har bir versiyasiga bir necha modullar qo‘shilmoqda, bu esa o‘z navbatida dasturni yuklanishini sekinlashiga olib kelmoqda. Shularni inobatga olib, kompyuteringizni konfigurasiyasiga qarab (tezkor xotira hajmi, prosessor,...), kerakli

versiyani o‘rnatish tavsiya qilinadi. Zamondan orqada qolmay deb oxirgi versiyasini qo‘yib, kompyuteringizni qiynab qo‘ymang.

Visual Studio 2012 dastur yozish uchun **muhit** (joy) hisoblanadi Bu muhitda dastur kodlarini kompyuter kodlariga o‘tkazib beradigan **kompilyator** birlashtirilgan bo‘ladi. Har bir dasturlash turi uchun alohida kompilyator mavjud bo‘lib, ishlatilgan dasturlash tiliga qarab kompilyator ishga tushadi. Visual Studio dasturi o‘rnatilsa, barcha kompilyatorlar ham avtomat o‘rnatiladi.

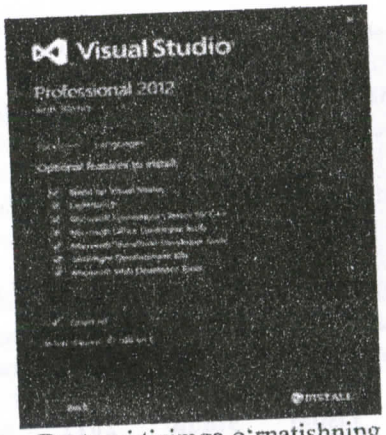
Demak dasturni o‘rnatishni boshlaymiz. Bunda **Microsoft Visual Studio 2012** dasturi o‘rnatiladi. Bu dasturning turli versiyalarini o‘rnatish bir-biridan biroz farqlanishi mumkin, lekin asos har doimgidek bir xil.

Microsoft Visual Studio 2012 ning o‘rnatiluvchi (инсталляционный) dasturi joylashgan papkadagi **vs_professional.exe** faylini ishga tushiriladi va ekranda quyidagi(2.1- rasm) oyna hosil bo‘ladi. Bu oynada dasturning lisenziya shartlariga roziligi so‘raladi. “**I agree to the License terms and conditions.**” bandi belgilanadi “**Continues**” tugmasi bosiladi.



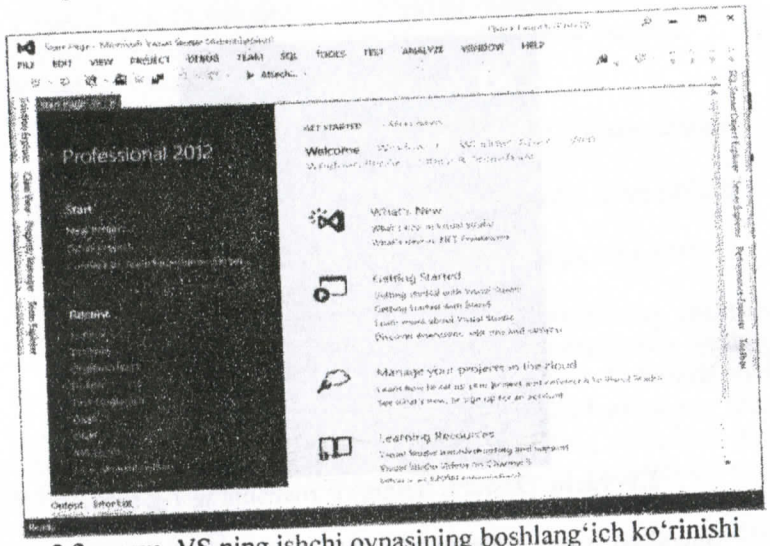
2.1- rasm. Dasturni tizimga o‘rnatishning 1-jaryoni

Shundan so‘ng dastur kerakli paketlarni o‘rnatish uchun tayyorgarlik ko‘radi (biroz kutamiz), yuqoridagi **Microsoft Corporation** ga axborot jo‘natishni ta’qiqlaymiz. **Next.**



2.2- rasm. Dasturni tizimga o‘rnatishning 2-jaryoni

“Select all” bandi tanlanadi va INSTALL tugmasi bosiladi. Dastur tizimga muvaffaqiyatli o‘rnatilgandan so‘ng dastur 2.3- rasmda ko‘rsatilganidek ishga tushadi.

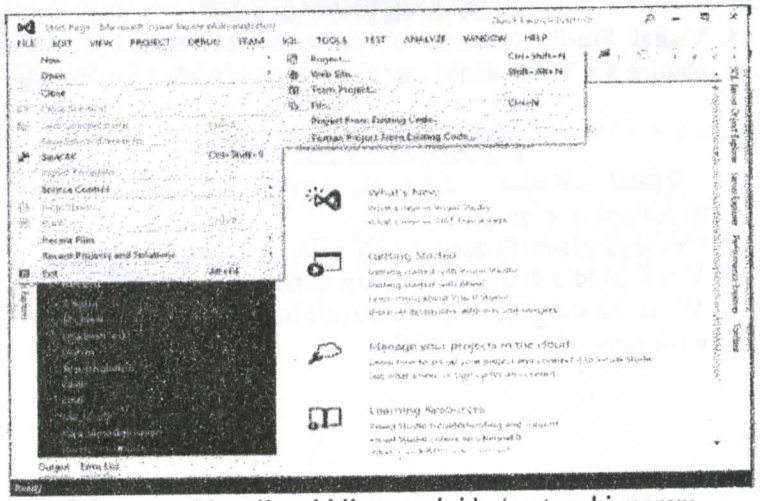


2.3- rasm. VS ning ishchi oynasining boshlang‘ich ko‘rinishi

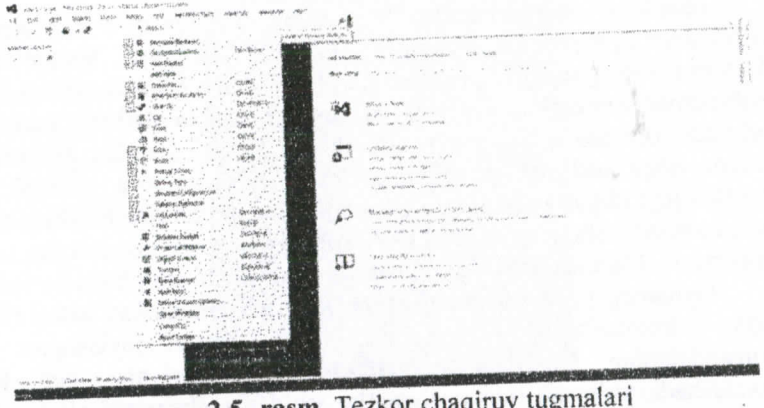
Interfeys – bu foydalanuvchiga muhit bilan qulay o‘zaro aloqa ta’minlovchi apparatdir. Kompyuteringizda **Microsoft Visual C+** dasturini o‘rnatganingizdan keyin uni Пуск/Программы bandidan ishg tushirishingiz mumkin. Qulaylik uchun o‘rnatilgan dasturiy mahsul belgisini operasion tizim ishchi stolining pastki qismida joylashga tezkor ishga tushirish paneliga joylab qo‘yish tavsiya qilinadi. B panelda joylashgan ixtiyoriy belgi sichqonchaning birgina bosilishi bila ishga tushadi. Shunday qilib, **Microsoft Visual C++** mahsulotini ishg tushiramiz. Ekranda asosiy oyna – dasturning ishchi stoli hosil bo‘ladi.

Oynaning yuqori qismida *asosiy menyu komandalari* qatori (Fil Edit, …komandalari) – gorizontal menyu qatori joylashgan. B komandalardan foydalanishda (ularni yana opsiyalar deb ha nomlashadi, ya’ni bir necha qiymatlar ichidan tanlanuvchi elementla *“tushuvchi menyu”* deb nomlanuvchi menyu ochiladi – bu meny vertikal menyu bo‘lib, o‘zida ekranning yuqorisidan pastiga qara joylashgan buyruqlar to‘plamini o‘zida namoyon qiladi.

Asosiy menyu qatori ostida joylashgan ikkinchi qatorda ba’ bajariluvchi buyruqlarning *tezkor chaqiruv tugmalari* joylashgan. B tugmalarning barchasi suzuvchi izohga ega (sichqoncha kursori tugmachaning ustiga olib borish va biroz kutish zarur, shundan so‘ ushbu tugma nima uchun mo‘ljallanganligi haqida izoh paydo bo‘ladi).



2.4- rasm. Yaratiluvchi ilova turini ko‘rsatuvchi menyu



2.5- rasm. Tezkor chaqiruv tugmalari

Bunday tugmalarning yonida ushbu tugmaning qo‘shimcha vazifalarini ochuvchi yana qo‘shimcha tugma bo‘lishi mumkin. Barcha tugmalarni ishchi stoliga joylashtirishning iloji yo‘qligi tufayli, ushbu tugmalar yashiringan holatda bo‘ladi. Ilova turini berish bilan ishchi soha o‘z ko‘rinishini o‘zgartiradi. Bular bilan keyingi tajriba mashg‘ulotlarida tanishamiz.

Topshiriq:

1. **Visual Studio** dasturini o‘rnating, ishchi muhiti bilan tanishib chiqing va “Hello world” so‘zini chiqaruvchi dasturini tuzing.

Nazorat savollari:

1. **Visual Studio** dasturlash muhitining boshqa dasturlash muhitlaridan afzalligi.
2. C++ dasturlash tili qanday tillar oilasiga kiradi?
3. Visual Studio dasturlash vositasi qanday o‘rnatiladi?
4. Visual Studio dasturlash vositasida loyiha yaratilayotganda konsol ilova nima uchun tanlanadi?

2.2. Console Aplication muhitida konsol ilovalar yaratish

Biz siz bilan mazkur mavzuda Visual C++ muhitida konso ilovalarni yaratish haqida suhbatlashamiz. Ba’zida, masalan, ilmi hisoblashlar uchun qandaydir eng oddiy berilganlarni kiritish v kiritilgan ma’lumotlarni murakkab matematik hisoblashlar bilan qayt ishlash, olingan natijalarni tezlik bilan ekranga chiqarish talab qilinad Bunday vaziyatlar katta dasturlarni qismlarga ajratib yaratish vaqtid vujudga keladi. Dasturlashda konsol ilovali deb ataladigan turlich dasturlarni tashkil qilish mumkin. Odatda konsol rejimida kompyute ekrani va klaviatura nazarda tutiladi. Misol tariqasida, qandaydir ikkit sonni kiritish va hisoblashlar natijasini konsol ilovada chiqarish takl qilinsin. Yangi (New Project) loyihaning nomi sifatida natijal yig‘indisi deb nom berib, OK tugmasi bosiladi va dasturlash ko yoziladigan muhitga o‘tiladi (2.7- rasm).

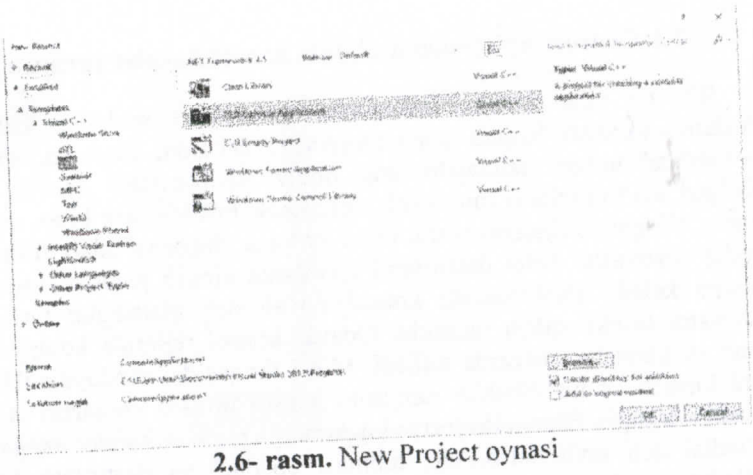
Visual Studio dasturining boshqarish muhiti bir necha qat dasturlash kodlarini hosil qiladi. Bu deyarli ish qobiliyatiga ega dastu uning operatori orqali ketma-ket F10 tugmasini bosish orqali o‘ti mumkin. Konsol yoki Windows ilova ishga tushganda main() funksiya birinchi chaqiriladigan funksiya hisoblanadi. main() dan key qo‘yiladigan figurali qavs ichiga dasturga oid kod joylashtiriladi.

Avvalo shuni aytishimiz kerakki, konsol ilovalarning ham b nechta platformalarda yaratish imkoniyati bor, lekin biz bu ishni **CL Console Application** bilan bajaramiz.

Dastlab, Visual Studio muhitini ishga tushiramiz. VS ish tushgandan so‘ng, quyidagi ketma-ketlikni bajaramiz:

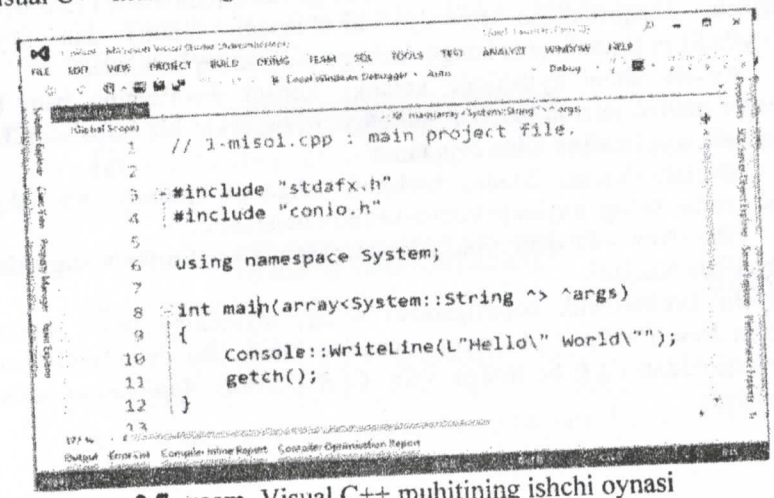
File->New->Project yoki klaviaturadan **Ctrl+Shift+N** tugmal birgalikda bosiladi.

Bu ketma-ketlik bajarilgandan so‘ng, loyihaning tipini tanla uchun *New Project* (2.6- rasm) oynasi ochiladi. Bu oynadagi ***Visu C++*** qismidan ***CLR*** bo‘limiga o‘tib, ***CLR Console Application*** pun tanlanadi.



2.6- rasm. New Project oynasi

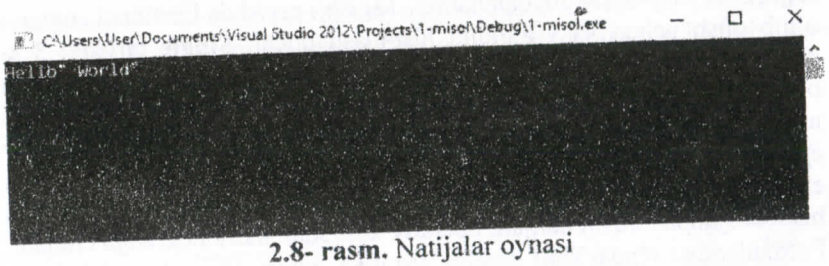
Name – qismiga loyiha nomi yoziladi (masalan "**1-misol**");
Location – qismiga loyihamiz saqlanadigan manzil ko'rsatiladi. (**Browse...** tugmasi yordamida). So'ngra **OK** tugmasi bosiladi. Natijada Visual C++ muhitining ishchi oynasi ochiladi. (2.7- rasm).



2.7- rasm. Visual C++ muhitining ishchi oynasi

Ushbu rasmdagi kodlarni birinchi bobda tushuntirilganini inobatga olib, ba'zi tushunchalarni berib ketamiz. 1- qatorda izoh

(kommentariya), 3- va 4- qatorlar kutubxonalar, 6- qatorda nomlar fazosi va 8-qatorda asosiy funksiya (**main()**) e'lon qilingan. Natijani chiqarish uchun **F5** yoki ishchi oydagi **Local Windows Debugger** tugmasi bosiladi. Natijalar oynasini ushlab turish uchun **getch()** funksiyasi ishlatilgan(2.8- rasm).



2.8- rasm. Natijalar oynasi

Berilganlarni kiritish va chiqarish, sonlar jadvalini konsolga chiqarish

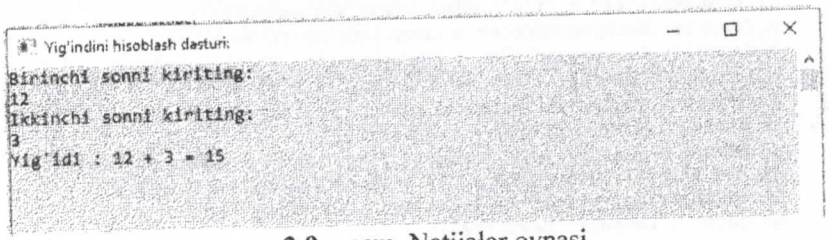
Konsol ilova berilganlarni kiritish va chiqarishga misol sifatida ikkita sonni kiritish va ularning yig'indisini chiqarish dasturini tuzib, tahlil qilib chiqamiz.

```
1. #include "stdafx.h"
2. #include "conio.h"
3. using namespace System;
4. int main(array<System::String ^> ^args){
5. Console::Title = "Yig'indini hisoblash dasturi:";
6. Console::BackgroundColor = ConsoleColor::Cyan;
7. Console::ForegroundColor = ConsoleColor::Black;
8. Console::Clear();// birinchi sonni o'qib olish
9. Console::WriteLine("Birinchi sonni kiriting:");
10. String^ qator = Console::ReadLine();
11. Single X,Y,Z; // satr turidagi o'zgaruvchini songa o'tkazish
12. X = Single::Parse(qator); // Ikkinchi sonni o'qib olish
13. Console::WriteLine("Ikkinchi sonni kiriting:");
14. qator = Console::ReadLine();
15. Y = Single::Parse(qator);// Yig'indini hisoblash
16. Z = X + Y;
17. Console::WriteLine("Yig'indi:{0}+{1}={2}", X, Y, Z);
    // 0.5 sekund 1000 Gs chastotali ovozli signal chiqarish
18. Console::Beep(1000, 500); Console::ReadKey(); _getch(); }
```

Ushbu dasturda **main()** – bajarilishi birinchi bo'lib boshlanadiga nuqta. Odatda konsol ilovalar qora fonli oynada bajariladi. Konso

ilovasining doimiy qora fonli oynasini boshqa rangga o'zgartirish uchun **Console::BackgroundColor** orqali, masalan, ko'kimtir yashil (**Cyan**) rangini o'rnatamiz, shrift rangi qora rangda bo'ladi. Qatorlarni konsol oynaga chiqarish uchun **WriteLine** metodidan, konsol oynada foydalanuvchi tomonidan kiritilgan qiymatlarni o'qib olish uchun **ReadLine** metodidan foydalanamiz. Tegishli ravishda kiritilgan sonlarni o'qib olish uchun va yig'indini hisoblash uchun **Single** turidagi 3 ta o'zgaruvchini e'lon qilamiz. Berilganlarni **Single** turida e'lon qilingan o'zgaruvchi butun va haqiqiy qismlardan iborat sonni qabul qilishi mukin. **Single** turida e'lon qilingan o'zgaruchi xotiradan 4 bayt joy egallaydi. Foydalanuvchi tomonidan kiritilgan satr belgilarining son ekanligini tekshirish uchun **Parse** metodidan foydalanamiz. Yig'indi hisoblangandan keyin natijani operativ xotiradan darhol ekranga chiqadi. Formatlangan chiqarishni amalga oshirish uchun **Console** ob`ektining **WriteLine** metodi qo'llaniladi. **Console::WriteLine("Yig'indi : {0} + {1} = {2}", X, Y, Z);**

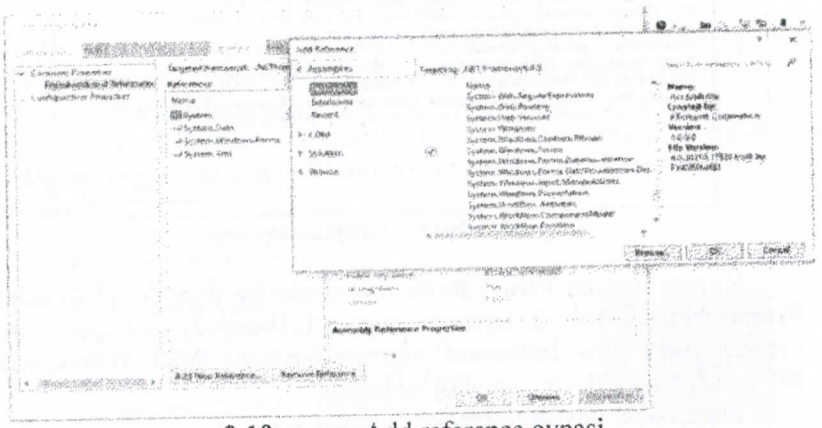
Belgi (simvol) bo'lib xizmat qiluvchi funksiya tugaganida va hisoblashlar natijasi ekranga chiqqanidan so'ng, **Beep** ovozli signalni beramiz. Dasturning oxirgi **Console::ReadKey();** qatori, foydalanuvchi tomonidan ixtiyoriy klaviatura bosilganda dastur to'xtatilishini bildiradi. Agar dasturda bu oxirgi qator qo'shilmasa hosil bo'lgan oyna darhol yo'qoladi va foydalanuvchi natijani ko'ra olmay qoladi. Dastur matni to'liq kiritilgandan keyin natijani ko'rish uchun **F5** klavishi bosiladi. Hosil bo'lgan oyna 2.9- rasmda ko'rsatilgan.



2.9- rasm. Natijalar oynasi

Konsol ilovada **MessageBox::Show** metodiga murojaat qilish uchun **using** direktivasidan foydalanamiz. **using namespace System::Windows::Forms;** Misol sifatida sana va vaqtni MessageBox orqali chiqarish dasturini ko'ramiz.

Birinchi navbatda, loyiha dasturiga **MessageBox** ob`ekti joylashgan kutubxonani qo'shish kerak. Buning uchun **Project** menyusidan References buyrug'i tanlanadi, ochilgan yangi oynadan **Add New Reference** tugmasi yoki **Alt+F7** klavishlari birgalikda bosiladi, **.NET** vkladkasidan **System.Windows.Forms.** havolasi tanlanadi. Natijada Properties oynasida ushbu ob`ekt qo'shilganligini ko'rishimiz mumkin. Ushbu havola quyidagi ko'rinishda bo'lishi ham mukin: **System.Windows.Forms.dll. (2.10- rasm).**



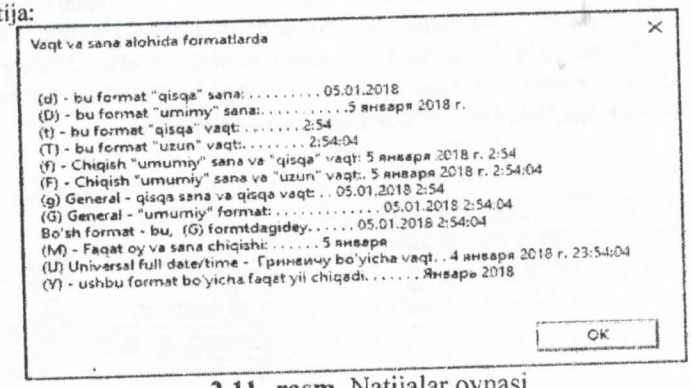
2.10- rasm. Add reference oynasi

Dastur kodi quyidagicha:

```
#include "stdafx.h"
#include "conio.h"
using namespace System;
using namespace System::Windows::Forms;
int main(array<System::String ^> ^args){
String^ Sana_va_vaqt = String::Format(
"(d) - bu format \"qisqa\" sana: . . . . . {0:d}\n" +
"(D) - bu format \"umimy\" sana:. . . . . .{0:D}\n" +
"(t) - bu format \"qisqa\" vaqt: . . . . . {0:t}\n" +
"(T) - bu format \"uzun\" vaqt:. . . . . {0:T}\n" +
"(f) - Chiqish \"umumiy\"sana va \"qisqa\" vaqt: {0:f}\n" +
"(F) - Chiqish \"umumiy\"sana va \"uzun\" vaqt:. {0:F}\n" +
"(g) General - qisqa sana va qisqa vaqt: . . {0:g}\n" +
"(G) General - \"umumiy\" format: . . . . . {0:G}\n" +
"Bo'sh format - bu,  (G) formtdagidey. . . . . {0}\n" +
"(M) - Faqat oy va sana chiqishi: . . . . . {0:M}\n" +
"(U) Universal full date/time - bo'yicha vaqt. . {0:U}\n" +
"(Y) - ushbu format bo'yicha faqat yil chiqadi. {0:Y}\n",
```

```
19. DateTime::Now);
20.  MessageBox::Show(ДатаВремя,"Vaqt va sana alohida formatlarda");
21. _getch(); }
```

Natija:



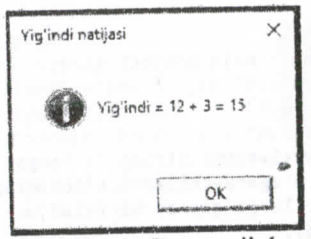
2.11- rasm. Natijalar oynasi

Konsol ilovada **Visual Basic** funksiyalariga murojat qilish uchun **Project** menyusidan References buyrug'i tanlanadi, ochilgan yangi oynadan **Add New Reference** tugmasi bosiladi, **.NET** vkladkasidan **Microsoft.VisualBasic.** havolasi tanlanadi.

Dastur kodi quyidagicha:

```
1. #include "stdafx.h"
2. #include "conio.h"
3. using namespace System;
4. using namespace Microsoft::VisualBasic;
5. int main(array<System::String ^> ^args){
6. String^ satr;
7. for (; ; ){
8. satr = Interaction::
9. InputBox("Birinchi soni kiriting:", "Ikki son", "", 100, 100);
10. if (Information::IsNumeric(satr) == true) break;}
11. Single X = Single::Parse(satr); for (; ; ){
12. satr = Interaction::
13. InputBox("Ikkinchi soni kiriting:", "Ikkita son", "", 100, 100);
14. if (Information::IsNumeric(satr) == true) break; }
15. Single Y = Single::Parse(satr);
16. Single Z = X + Y;
17. satr = String::Format("Yig'indi = {0} + {1} = {2}", X, Y, Z);
18. // Natijani ekranga chiqarish
19. Interaction::MsgBox(satr, MsgBoxStyle::Information, "Yig'indi
    natijasi"); _getch(); }
```


Natija:



2.12- rasm. MassegaBox natijalar oynasi

C++ dasturlash tili funksiyalar majmuidan iborat bo'lganligi uc
CLR Console Application punkti orqali funksiyalarni tashkil etis
ishlatiladigan operatorlar va ba'zi bir tushunchalarga to'xtalib o'tami

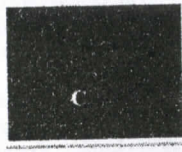
-O'zgaruvchilarni e'lon qilish, ularga qiymat kiritish chiqarish operatorlarini ishlatilishi quyidagicha bo'ladi:

1- usul:

```
1. // 1-misol.cpp : main project file.
2. #include "stdafx.h"
3. #include "conio.h"
4. using namespace System;
5. int main(array<System::String ^> ^args) {
6. int a, b; // o'zgaruvchilarni e'lon qilish
7. // o'zgaruvchilarga qiymat kiritish
8. a= Convert::ToInt32( Console::ReadLine());
9. b= Convert::ToInt32( Console::ReadLine());
10. // hisoblash
11. int c=a+b;
12. // 'S o'zgaruvchining qiymatini chiqarish
13. Console::WriteLine("C= "+c); getch(); }
```

2- usul:

```
1. // 2-misol.cpp : main project file.
2. #include "stdafx.h"
3. #include "conio.h"
4. #include "iostream"
5. using namespace std;
6. int main(array<System::String ^> ^args) {
7. int a, b; // o'zgaruvchilarni e'lon qilish
8. // o'zgaruvchilarga qiymat kiritish
9. cin>>a; cin>>b;
10. // hisoblash
11. int c=a+b;
12. // C o'zgaruvchini qiymatini chiqarish
13. cout<<"C= "<<c<<endl; getch(); }
```



3- usul:

```
// 3-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
#include "stdio.h"
int main(array<System::String ^> ^args) {
int a, b; // o'zgaruvchilarni e'lon qilish
// o'zgaruvchilarga qiymat kiritish
scanf("%d",&a);
scanf("%d",&b);
      // hisoblash
      int c=a+b;
      // C o'zgaruvchini qiymatini chiqarish
      printf("C= %d",c); getch(); }
```

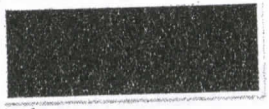
- Operatorlarni qo'llanilishi quyidagicha bo'ladi:

1- ***if shart*** *operatorini qo'llanilishi:*

```
// 4-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
#include "iostream"
using namespace std;
int main(array<System::String ^> ^args) {
int a, b; //o'zgaruvchilarni e'lon qilish
cin>>a>>b;//o'zgaruvchilarga qiymat kiritish
// if shart operatorini qo'llanilishi
if(a>b){cout<<"max="<<a<<endl;}
else {cout<<"max="<<b<<endl;} getch(); }
```

2- ***Switch()*** *tanlash operatorini qo'llanilishi:*

```
// 5-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
int a; // o'zgaruvchilarni e'lon qilish
// o'zgaruvchilarga qiymat kiritish
a= Convert::ToInt32( Console::ReadLine());
switch (a){
case 1: Console::WriteLine("Dushanba"); break;
case 2: Console::WriteLine("Seshanba"); break;
default: Console::WriteLine("Xatolik"); break;}
getch(); }
```

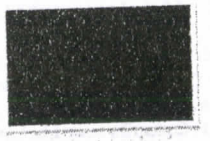


3- ***goto*** *shartsiz o'tish operatorini qo'llanilishi:*

```
// 6-misol.cpp : main project file.
#include "stdafx.h"
```

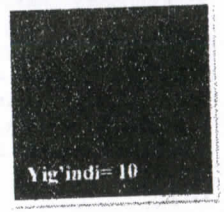


```
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
int a; // o'zgaruvchilarni e'lon qilish
// o'zgaruvchilarga qiymat kiritish
a= Convert::ToInt32( Console::ReadLine());
if(a>10){goto bir;}
Console::WriteLine("a 10 dan kichik");
bir: Console::WriteLine("a 10 dan katta");
getch(); }
```



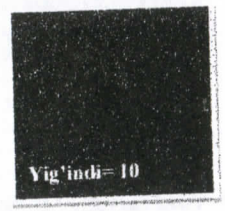
4- ***while()*** *sharti oldin tekshiriladigan sikl operatorini qo'llanilishi:*

```
// 7-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
// o'zgaruvchilarni e'lon qilish
int n,i=1, s=0;
// o'zgaruvchilarga qiymat kiritish
n= Convert::ToInt32( Console::ReadLine());
while (i<=n){
Console::WriteLine(i);
s+=i; i++;  }
Console::WriteLine("Yig'indi= "+s);
getch();}
```



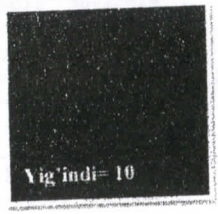
5- do ***while()*** *sharti keyin tekshiriladigan sikl operatorini qo'llanilishi:*

```
// 8-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
// 'o'zgaruvchilarni e'lon qilish
int n,i=1, s=0;
// 'o'zgaruvchilarga qiymat kiritish
n= Convert::ToInt32( Console::ReadLine());
do {
Console::WriteLine(i);
s+=i; i++;  } while (i<=n);
Console::WriteLine("Yig'indi= "+s);
getch(); }
```



6- for *sikl operatorini qo'llanilishi:*

```
// 9-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
```



```
// 'o'zgaruvchilarni e'lon qilish
int n,i=1, s=0;
// 'o'zgaruvchilarga qiymat kiritish
n= Convert::ToInt32( Console::ReadLine());
for(i=1;i<=n;i++){
Console::WriteLine(i);
s+=i; }
Console::WriteLine("Yig'indi= "+s);
getch();
}
```

7- *Massiv va uning qo'llanilishi:*

```
// 10-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
int n, A[]={1,-4,77,-2};
n= Convert::ToInt32( Console::ReadLine());
for(int i=0;i<n;i++){
Console::WriteLine("A["+i+"]= "+A[i]); }
getch();
}
```

```
3
A[0,0]= 1
A[0,1]= -4
A[0,2]= 75
A[1,0]= 2
A[1,1]= -6
A[1,2]= 9
A[2,0]= 4
A[2,1]= 8
A[2,2]= -3
```

8- *Ikki o'lchovli massiv va uning qo'llanilishi:*

```
// 11-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
int main(array<System::String ^> ^args) {
int n,A[][4]={{1,-4,75},{2,-6,9},{4,8,-3}};
n= Convert::ToInt32(Console::ReadLine());
for(int i=0;i<n;i++){
for(int j=0;j<n;j++){
Console::WriteLine("A["+i+","+j+"]= "+A[i][j]);
} } getch();
}
```

9- **void** *turidagi* funksiya *va uning qo'llanilishi:*

```
// 12-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System;
void mas_in(int *A, int n){
for(int i=0;i<n;i++){
A[i]=Convert::ToInt32(
Console::ReadLine());} }
void mas_out(int *A, int n){
for(int i=0;i<n;i++){
```

```
3
1
2
3
A[0]= 1
A[1]= 2
A[2]= 3
```



```
Console::WriteLine("A["+i+"]= "+A[i]);} }
int main(array<System::String ^> ^args)
{ int n,*A = new int[100];
n= Convert::ToInt32( Console::ReadLine());
mas_in(A,n);
mas_out(A,n); getch(); }
```

10- *funksiya va uning qo'llanilishi:*

```
// 13-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
using namespace System; int s;
float mas_sum(int *A, int n){
for(int i=0;i<n;i++){
s+=A[i];} return s; }
void mas_in(int *A, int n){
for(int i=0;i<n;i++){
A[i]=Convert::ToInt32(
Console::ReadLine());} }
int main(array<System::String ^> ^args) {
int n,*A = new int[100];
n= Convert::ToInt32( Console::ReadLine());
mas_in(A,n);
Console::WriteLine("Summa= "+mas_sum(A,n));
getch();
}
```

```
3
1
2
3
Summa= 6
```

11- *Fayllar bilan ishlovchi maxsus funksiyalar va uning qo'llanilishi:*

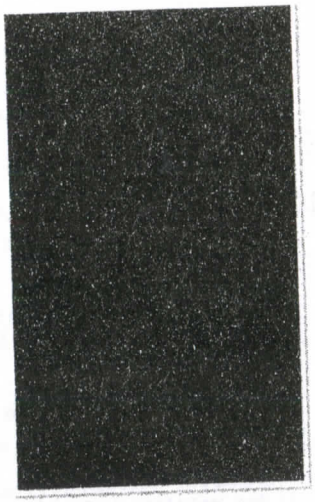
```
// 14-misol.cpp: matinli fayldan o'qish.
#include "stdafx.h"
#include "conio.h"
#include "iostream"
#include <stdio.h>
using namespace std;
int main(array<System::String ^> ^args){
FILE * fayl; int a=2; float b=4.5; char satr[20];
fayl=fopen("1.txt","r");// faylni o'qish uchun ochish
      if(fayl==NULL){cout<<"Fayl topilmadi"<<endl; exit(1);}
      fscanf(fayl," %d",&a); cout<<"a= "<<a<<endl;// int tipiga
      fscanf(fayl," %f",&b); cout<<"b= "<<b<<endl;// float tipiga
      fscanf(fayl,"%s",satr); cout<<"satr= "<<satr<<endl;//char
      _getch();}
```

```
a= 12
b= 2.5
satr= Salom
```

11- *Struktura*(**struct{}**) *va uning qo'llanilishi:*

```
// 14-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
#include "iostream"
using namespace std; int s;
```

```
6. struct talaba
7. {  char ism[20];
8. char fam[20];
9. int yosh;
10. float stipendiya; };
11. int main()
12. { int n; talaba A[100];
13. cin>>n;
14. for(int i=0; i<n;i++){
15. cout<<i+1<<" - talaba"<<endl;
16. cout<<"Ismi= ";cin>>A[i].ism;
17. cout<<"Fam= ";cin>>A[i].fam;
18. cout<<"Yoshi= ";cin>>A[i].yosh;
19. cout<<"Stipendiyasi= ";
20. cin>>A[i].stipendiya;}
21. for(int i=0; i<n;i++){
22. cout<<i+1<<" - talaba"<<endl;
23. cout<<"Ismi= "<<A[i].ism<<endl;
24. cout<<"Fam= "<<A[i].fam<<endl;
25. cout<<"Yoshi= "<<A[i].yosh<<endl;
26. cout<<"Stipendiyasi= "<<A[i].stipendiya<<endl;
27. } getch(); }
```



12- *Sinf*(**class{}**) *va uning qo'llanilishi:*

```
1. // 15-misol.cpp : main project file.
2. #include "stdafx.h"
3. #include "conio.h"
4. #include "iostream"
5. using namespace std; int s;
6. class base {
7. int i, j;
8. public:
9. void set(int a, int b) { i=a; j=b; }
10. void show() { cout << i << " " << j << "g`n"; } };
11. class derived : public base { int k;
12. public:
13. derived(int x) { k=x; }
14. void showk() { cout << k << "g`n"; } };
15. int main(){
16. derived ob(3);
17. ob.set(1, 2); // asos sinf a'zosiga ruxsat ochiq
18. ob.show(); // asos sinf a'zosiga ruxsat ochiq
19. ob.showk(); // voris sinf a'zosidan foydalanish
20. getch(); }
```

2.3. Visual C++ muhitida strukturalar bilan ishlash

Struktura – bu ma'lumotlarni bir butun nomlangan elementl to'plamiga birlashtirish. Struktura elementlari (maydonlar) har xil tipd bo'lishi mumkin va ular har xil nomlarga ega bo'lishi kerak.

Strukturali tip quyidagicha aniqlanadi:

struct { <ta'riflar ro'yxati > }

Strukturada albatta bitta komponenta bo'lishi kerak. Struktu tipidagi o'zgaruvchi quyidagicha ta'riflanadi:

<struktura_nomi > <o'zgaruvchi>;

Struktura tipidagi o'zgaruvchi ta'riflanganda inisializasiya qilini mumkin: **<struktura_nomi > <o'zgaruvchi>=<inisializator>;**

Strukturani inisializasiyalash uchun uning elementlar qiymatlar figurali qavslarda tavsiflanadi.

Misollar:

1. **struct Student {**
char name[20];
int kurs;
float rating; };
Student s={"Qurbonov",1,3.5};

2. **struct {**
char name[20];
char title[30];
float rate; }employee={"Ashurov", "direktor",10000};

Strukturalarni o'zlashtirish. Bitta tuzilma tipdagi o'zgaruvch uchun o'zlashtirish operasiyasi aniqlangan. Bunda har bir element nusxa olinadi. Masalan:

Student ss=s;

Struktura elementlariga murojaat. Struktura elementla murojaat aniqlangan ismlar yordamida bajariladi:

<Struktura_nomi>.<element_nomi>

Masalan:

employee.name – «Ashurov» satriga ko'rsatkich;

employee.rate – 10000 qiymatga ega bo'lgan butun tipd o'zgaruvchi.

Strukturaga ko'rsatkichlar. Strukturaga ko'rsatkichlar o ko'rsatkichlar kabi tasvirlanadi:

Student*ps;

Strukturaga ko‘rsatkich ta’riflanganda inisializasiya qilinishi mumkin:

```
Student *ps=&mas[0];
```

Ko‘rsatkich orqali struktura elementlariga ikki usulda murojaat qilish mumkin. Birinchi usul adres bo‘yicha qiymat olish amaliga asoslangan bo‘lib quyidagi shaklda qo‘llaniladi:

(* strukturaga ko‘rsatkich).element nomi;

Ikkinchi usul maxsus strelka (->) amaliga asoslangan bo‘lib quyidagi ko‘rinishga ega: strukturaga ko‘rsatkich->element nomi.

Struktura elementlariga quyidagi murojaatlar o‘zaro tengdir:

```
cin>>(*ps).name;
cin>>ps->title;
```

Bitli maydonlar

Bitli maydonlar strukturalarning xususiy holidir. Bitli maydon ta’riflanganda uning uzunligi bitlarda ko‘rsatiladi (butun musbat konstanta).

Misol:

```
struct {
int a:10;
int b:14}xx,*pxx;
. . . .
xx.a=1;
pxx=&xx;
pxx->b=8;
```

Bitli maydonlar ixtiyoriy butun tipga tegishli bo‘lishi mumkin. Bitli maydonlar adresini olish mumkin emas. Xotirada bitli maydonlarni joylashtirish kompilyator va apparaturaga bog‘liq.

Birlashmalar

Strukturalarga yaqin tushuncha bu birlashma tushunchasidir. Birlashmalar **union** xizmatchi so‘zi erdamida kiritiladi. Misol uchun: **union {long h; int i,j; char c[4]} UNI;** Birlashmalarning asosiy xususiyati shundaki, uning hamma elementlari bir xil boshlang‘ich adresga ega bo‘ladi.

Masalan:

```
union{
char s[10];
int x; }u1;
```


Quyidagi dastur yordamida bu xususiyatni tekshirish mumkin:

```
enum paytype{CARD,CHECK};
struct{
paytype ptype
union{
 char card[25];
 long check;
 };
}info;
switch (info.ptype)
{
case CARD:cout<<"\nKarta bilan to'lash:"<<info.card;break;
case CHECK:cout<<"\nChek bilan to'lash:"<<info.check;break;
}
```

Massivlar va satrlar

Massivlarni ta’riflash. Massiv indeksli o‘zgaruvchidir.

Massivning sodda ta’rifi:

```
<tip> <o'zgaruvchi_nomi>>[<konstanta_ifoda>] = <inisializator>;
```

Massiv indekslar qiymati har doim 0 dan boshlanadi.

Ko‘p o‘lchovli massiv inisializasiya qilinganda massivning bir indeksi chegarasi ko‘rsatilishi shart emas, lekin qolgan inde chegaralari ko‘rsatilishi shart. Misol uchun:

```
int a[6]; float b[8],c[100];
double d[] = {1, 2, 3, 4, 5};
int A [20][10];
int A [30][20][10];
int A [3][3] = {0,1,2,3,4,5,6,7,8,9,10,11};
int A[ ][3] = { {0,1,100}, {200,210,300}, {1000, 2000, 2100}};
```

Satrlar. Satrli konstanta ikkilik qavslarga olingan simvollar k ketligidir. Satrli konstanta oxiriga avtomatik ravishda satr ko‘chirish simvoli qo‘shiladi. Satr qiymati simvolli konstanta bo‘lgan sim massiv sifatida ta’riflanadi.

Misol uchun:

```
Char capital[]="TASHKENT";
Char capital[]={'T','A','S','H','K','E','N','T','\n'}; char A[ ][9] = {
"Tashkent", "Samarkand", "Xiva"};
```

Massivlar va satrlar funksiya parametrlari sifat

Funksiyalarda massivlar argument sifatida ishlatilganda, ular birinchi indeksi chegarasini ko‘rsatish shart emas, qolganl chegarasini ko‘rsatish shart. Massivlar ilova bo‘yicha uzatiladi, y

ularning qiymati funksiyada o'zgarishi mumkin.

Misol: //massiv elementlari summasini hisoblash

```
int sum (int n, int a[] ){
int i, int s=0;
for( i=0; i<n; i++ )
s+=a[i];
return s; }
```

Satrlar parametrlar sifatida **char**[] tipidagi bir o'lchovli massivlar sifatida uzatilishi mumkin. Bu holda satr uzunligini aniq ko'rsatish shart emas.

Misol: //simvollar sonini hisoblash

```
int strlen ( char a[]){
 int i=0; while(a[i++]);
return i; }
```

Dinamik massivlar

O'zgaruvchan o'lchamli massivlarni shakllantirish ko'rsatkichlar va xotirani dinamik taqsimlash vositalari yordamida tashkil etiladi. Xotirani dinamik taqsimlash uchun **new** va **delete** operasiyalardan foydalaniladi. Operasiya

new <tip_nomi> (<initsializator>)

tip ismi orqali belgilangan ma'lumotlar tipiga mos keluvchi o'lchamli bo'sh xotira qismini ajratish va unga murojaat etish imkonini beradi. Ajratilgan xotira qismiga inisializator orqali aniqlangan qiymat kiritiladi. Xotira ajratilsa xotira ajratilgan qismining bosh adresi qaytariladi, agarda xotira ajratilmasa NULL qaytariladi.

new operasiyasi orqali oldindan ajratilgan xotira qismi delete operasiyasi yordamida bo'shatiladi.

Misollar:

```
int *i; i=new int(10);
delete i;
```

Mazkur

new <tip_nomi> (<initsializator>)

operasiyasi o'zgaruvchilar massiviga xotira ajratishga imkon beradi.

Misollar:

```
int *mas=new[5];
delete [] mas;
```

Skalyar o'zgaruvchilarga xotira ajratilishi 1-misolda ko'rsatilgan. Matrisani shakllantirishda oldin bir o'lchovli massivlarga ko'rsatuvchi ko'rsatkich massivlar uchun xotira ajratiladi, keyin esa parametrli siklda bir o'lchovli massivlarga xotira ajratiladi.

Misol:

```
int n,m; cin>>n;
matr=new int*[n];
for (i=0;i<n;i++)
{cin>>m;
matr[i]=new int[m];
```

Xotirani bo'shatish uchun bir o'lchovli massivlarni bo'shatiruvchi siklni bajarish zarur.

```
for(int i=0;i<n;i++)
delete matr[i];
```

keyin esa matr ko'rsatkich ko'rsatgan xotira bo'shattiriladi.

```
delete [] matr;
```

Satr murakkab tip sifatida

String tipi. Satrlar bilan ishlash uchun standart kutubxonaga kiruvchi **string** murakkab turidan foydalanish qulaydir.

Bu tipdan foydalanish uchun quyidagi sarlavhali faylni ulash lozim:

```
#include <string.h>
```

Satrlarni ta'riflashga misollar:

```
string st( "BAXO \n" );//simvollar satri bilan inisiallash
string st2;  //bo'sh satr
string st3( st ); shu tipdagi o'zgaruvchi bilan inisiallash
```

Satrlar ustida amallar. Satrlar ustida quyidagi amallar aniqlangan:

- qiymat berish (=);
- ikki amal ekvivalentlikni tekshirish uchun (==) va (!=);
- konkatenatsiya yoki satrlarni ulash (+);
- qiymat berib qo'shish amali (+=)
- indeks olish ([]).

Usullar: Satr uzunligini aniqlash uchun size() funksiyacidan foydalaniladi (uzunlik tugallovchi simvolni xisobga olmaydi).

```
cout << "uzunlik "<< st << ": " << st.size();
```

Maxsus **empty()** usuli agar satr bo'sh bo'lsa, **true** qaytaradi, aks xolda **false** qaytaradi:

```
if ( st.empty() )//to'g'ri: bo'sh
```

Nazorat savollari:

1. Struktura deganda nimani tushunasiz?
2. Struktura qanday e'lon qilinadi?
3. Strukturaning massiv hamda to'plamdan farqi nimada?
4. Struktura elementiga qiymat qanday o'zlashtiriladi?

2.4. Visual C++ muhitida sinflar va ob`ektlar yaratish

Sinflar va sinf a'zolari

Yangi tip sinfni e'lon qilish bilan tuziladi. Sinf - bu bir – biri bilan funksiyalar orqali bog'langan o'zgaruvchilar va usullar to'plamidir. Sinflarga amaliyotdan ko'pgina misollar keltirish mumkin. Masalan, avtomobilni g'ildirak, eshik, o'rindiq, oyna va boshqa qismlardan tashkil topgan kolleksiya yoki haydash tezligini oshirish, to'xtatish, burish imkoniyatlariga ega bo'lgan ob`ekt deb tasavvur qilish mumkin. Avtomobil o'zida turli ehtiyot qismlarni va ularni funksiyalarini inkapsulyasiya qiladi. Avtomobil kabi sinfda ham inkapsulyatsiya qator imkoniyatlarni beradi. Barcha ma'lumotlar bitta ob`ektda yig'ilgan va ularga osongina murojaat qilish, ularni o'zgartirish va ko'chirish mumkin. Sizning sinfingiz bilan ishlovchi dasturiy qismlar, ya'ni mijozlar sizning ob`ektingizdan, uning qanday ishlashidan tashvishlanmasdan, bemalol foydalanishlari mumkin.

Sinf o'zgaruvchilarining ixtiyoriy kombinasiyasidan, shuningdek boshqa sinflar tiplaridan iborat bo'lishi mumkin. Sinfdagi o'zgaruvchilar o'zgaruvchi – a'zolar yoki xossalar deyiladi. Car sinfi o'rindiq, radiopriyomnik, shina va boshqa o'zgaruvchi -a'zolardan iborat. O'zgaruvchi – a'zolar faqatgina o'zlarining sinflarida yotadilar. G'ildirak va motor avtomobilning qanday tarkibiy qismi bo'lsa, o'zgaruvchi – a'zolar ham sinfning shunday tarkibiy qismidir.

Sinfdagi funksiyalar odatda o'zgaruvchi a'zolar ustida biror bir amal bajaradilar. Ular funksiya – a'zolar yoki sinf usullari deb aytiladi. Mashina sinfi usullari qatoriga **Haydash**() va **Tuxtatish**() usullari kiradi. Mushuk sinfi hayvonni yoshi va og'irligini ifodalovchi o'zgaruvchi – a'zolarga ega bo'lishi mumkin. Shuningdek, bu sinfning funksional qismi **Uxlash(), Miyovlash(), SichqonTutish()** usullaridan iborat bo'ladi.

Funsiya – a'zolar ham o'zgaruvchi a'zolar singari sinfda yotadi. Ular o'zgaruvchi a'zolar ustida amallar bajaradi va sinfni funksional imkoniyatlarini aniqlaydi.

Sinfni e'lon qilish

Sinfni e'lon qilish uchun **class** kalitli so'zi, undan so'ng ochiluvchi figurali qavs, so'ng xossalar va usullari ro'yxati ishlatiladi. Sinfni e'lon

qilish yopiluvchi figurali qavs va nuqtali vergul orqali yakunlan
Masalan, **Mushuk** sinfini quyidagicha e'lon qilish mumkin.

```
Class Mushuk {
    unsigned int itsYosh ;
    unsigned int itsOgirlik ;
    void Miyovlash() };
```

Mushuk sinfini e'lon qilishda xotira zahiralanmaydi. E'lon qil kompilyatorga Mushuk sinfini mavjudligini, hamda unda qan qiymatlar saqlashi mumkinligi (**intYosh**, **intsOgirlik**) va u qan amallarni bajarishi mumkinligi (Miyovlash() usuli) haqida xabar ber Bundan tashqari, bu e'lon qilish orqali kompilyatorga **Mus** sinfining o'lchami, ya'ni har bir **Mushuk** sinfi ob`ekti uc kompilyator qancha joy ajratishi lozimligi haqida ham ma'lumot ber Masalan, joriy misolda butun qiymat uchun to'rt bayt talab qili **Mushuk** sinfi ob`ekti o'lchovi sakkiz bayt bo'ladi. (**intY** o'zgaruvchisi uchun to'rt bayt, **intsOgirlik** o'zgaruvchisi uchun t bayt). **Miyovlash**() usuli xotiradan joy ajratishni talab qilmaydi.

Ob`ektni e'lon qilish

Yangi turdagi ob`ekt xuddi oddiy butun sonli o'zgaruvch aniqlanadi. Haqiqatan ham ixtiyoriy butun sonli o'zgaru quyidagicha aniqlanadi:

unsigned int MyVariable //ishorasiz butun sonni aniqlaymiz. sinfidagi ob`ekt esa quyidagicha aniqlanadi:

Mushuk Frisky //Mushuk ob`ektini aniqlaymiz.

Bu dasturiy kodlarda **unsigned int** tipidagi **MyVariable** n o'zgaruvchi va **Mushuk** sinfining **Frisky** nomli ob`ekti aniqlandi.

Ko'pgina hollarda sinf va ob`ekt tushunchalarini ishlatish chalkashlikka yo'l qo'yiladi. Shuning uchun, ob`ekt sinfning biror ekzemplyari(nusxasi) ekanligini yana bir bor ta'kidlash joiz.

Sinf a'zolariga murojaat qilish imkoni

Mushuk sinfining real ob`ektini aniqlaganimizdan so'ng ob`ektning a'zolariga murojaat qilish zaruriyati tug'ilishi mur Buning uchun bevosita murojaat (.) operatori qo'llaniladi. Mas **Frisky** ob`ektining **Weight** o'zgaruvchi - a'zosiga 50 s o'zlashtirmoqchi bo'lsak quyidagi jumlani yozishimiz lozim.

Fresky.Weight=50;

Meow() usulini chaqirish uchun esa

Frisky.Meow(); jumlasini yozish lozim. Qiymat sinfga emas ob`ektga o'zlashtiriladi. C++ tilida berilganlar tipiga qiymat o'zlashtirilmaydi. Qiymat faqatgina o'zgaruvchilarga beriladi. Masalan, quyidagi yozuv noto'g'ridir:

int=s // noto'g'ri

Kompilyator **int** tipiga qiymat o'zlashtirilishi xatolik ekanligi haqida xabar beradi. Xuddi shu nuqtai – nazardan quyidagi yozuv ham noo'rindir:

Cat.itsYosh= 5 // noto'gri

Nazorat savollari:

1. Ob`ektga yo'naltirilgan yondashuvning uchta asosiy tamoyilini aytib bering.
2. Sinf nima uchun yaratiladi va uni yaratishdan maqsad?
3. Sinf va struktura orasidagi farqlarni ayting.
4. Nima uchun bevosita sinf elementiga qiymat o'zlashtirish mumkin emas?
5. Sinf usuli nima?
6. Sinf xossasi, ya'ni xususiyati nima?

2.5. Visual C++ muhitida konstruktor hamda destruktor yaratish

Butun sonli o'zgaruvchini aniqlashning ikki xil yo'li bor. Birinchisi, oldin o'zgaruvchini aniqlash, keyin esa unga biror bir qiymat o'zlashtirishdir. Masalan,

```
int Ogirlik; // o'zgaruvchini aniqlash
             // bu yerda boshqa ifodalar bor
Ogirlik =7; // o'zgaruvchiga qiymat o'zlashtiramiz.
```

Ikkinchisi, o'zgaruvchi aniqlanishi bilan birga unga darhol qiymat o'zlashtiriladi, masalan:

```
int Ogirlik=7; //o'zgaruvchini e'lon qilamiz va unga qiymat
```

o'zlashtiramiz.

Qiymat berish amali o'zgaruvchi aniqlanishi bilan unga boshlang'ich qiymat o'zlashtirilishini anglatadi. Keyinchalik, bu o'zlashtirilgan qiymatni o'zgartirishingiz ham mumkin.

Sinfning o'zgaruvchi – a'zosiga qanday qiymat o'zlashtirildi?

Buning uchun sinfda konstruktor deb ataluvchi maxsus funksiya – a'zo ishlatiladi. Zaruriy vaqtda konstruktor bir nechta parametrni qabul qiladi. Lekin hech qanday tipdagi qiymat qaytarmaydi. Konstruktor – bu sinf nomi bilan ustma – ust tushadigan sinf usulidir.

Sinfda konstruktorni e'lon qilinishi bilan destruktorlar ham aniqlanishi lozim. Agarda konstruktor sinf ob`ektini tuzish va uning o'zgaruvchi – a'zolariga qiymat berish vazifasini bajarsa, destruktor mavjud ob`ektning xotiradan o'chiradi. Destruktorlar sinf nomi oldiga tilda (~) belgisini qo'yish orqali aniqlanadi. Destruktorlar hech qanday argument qabul qilmaydi va hech qanday qiymat qaytarmaydi.

Boshlang'ich berilgan konstruktor va destruktorlar

Agarda siz konstruktor yoki destruktorni aniqlamasangiz, siz uchu bu ishni kompilyatorning o'zi bajaradi. Standart konstruktor v destruktorlar birorta argument qabul qilmaydi va hech qanday ama bajarmaydi.

Nazorat savollari:

1. Konstruktor nima?
2. Konstruktor nima uchun tipsiz e'lon qilinadi?
3. Konstruktorni yaratishni necha xil usulini bilasiz?
4. Destruktor nima?
5. Konstruktorni destruktordan nima ajratib turadi?

2.6. Visual C++ muhitida inkapsulyatsiyani qo'llash

Inkapsulyatsiya

Agarda muhandis ishlab chiqarish jarayonida rezistorni qo'llasa, buni yangidan ixtiro qilmaydi, omborga (magazinga) borib, m parametrlarga muvofiq kerakli detalni tanlaydi. Bu holda muhandis jo rezistor qanday tuzilganligiga e'tiborini qaratmaydi, rezistor faqatgi zavod xarakteristikalariga muvofiq ishlasa yetarlidir. Aynan, shu tas konstruksiyada qo'llaniladigan yashirinlik yoki ob`ektni yashirinl yoki avtonomligi xossasi inkapsulyatsiya deyiladi. Inkapsulyats yordamida berilganlarni yashirish ta'minlanadi. Bu juda yax xarakteristika bo'lib foydalanuvchi o'zi ishlatayotgan ob`ektning ic ishlari haqida umuman o'ylamaydi. Haqiqatan ham, xolodilni

ishlatishda refrijeratorni ishlash tamoyilini bilish shart emas. Yaxshi ishlab chiqilgan dastur ob`ektini qo'llashda uning ichki o'zgaruvchilarining o'zaro munosabati haqida qayg'urish zarur emas.

Yana bir marta takrorlash joizki, rezistorni samarali qo'llash uchun uning ishlash tamoyili va ichki qurilmalari haqidagi ma'lumotlarni bilish umuman shart emas. Rezistorning barcha xususiyatlari inkapsulyatsiya qilingan, ya'ni yashirilgan. Rezistor faqatgina o'z funksiyasini bajarishi yetarlidir.

C++ tilida inkapsulyatsiya tamoyili sinf deb ataluvchi nostandart tiplarni(foydalanuvchi tiplarini) hosil qilish orqali himoya qilinadi.

Sinflar qanday tuzilishga ega ekanligi bilan keyinroq tanishib chiqamiz. To'g'ri aniqlangan sinf ob`ektini butun dasturiy modul sifatida ishlatish mumkin. Haqiqiy sinfning barcha ichki ishlari yashirin bo'lishi lozim. To'g'ri aniqlangan sinfning foydalanuvchilari uning qanday ishlashini bilishi shart emas, ular sinf qanday vazifani bajarishini bilsalar yetarlidir.

Sinf elementini e'lon qilishda bir nechta kalit so'zlardan foydalaniladi: **public, private, protected**.

Umumiy (**public**) komponentalar dasturni ixtiyoriy qismida murojaat xuquqiga ega. Ulardan ixtiyoriy funksiya ushbu sinf ichida va sinf tashqarida foydalansa ham bo'ladi.

Xususiy (**private**) komponentalar sinf ichida murojaat xuquqiga ega, lekin sinf tashqarisidan esa murojaat qilish mumkin emas. Komponentalardan ushbu ular tavsiflangan sinfdagi funksiya - a'zolari yoki "do'stona"- funksiyalar orqali foydalanish mumkin.

Ximoyalangan (**protected**) komponentalar sinf ichida va hosila sinflarda murojaat xuquqiga ega.

Ulardan eng muhimlari **public** (ochiq) va **private** (yopiq) kalit so'zlari bo'lib, ular orqali ob`ektning a'zolariga murojaat qilish imkoniyati chegaralanadi. Sinfning barcha usullari va xossalari boshlang'ich holda yopiq deb e'lon qilinadi. Yopiq a'zolarga faqatgina shu sinfning usullari orqaligina murojaat qilish mumkin. Ob`ektning ochiq a'zolariga esa dasturdagi barcha funksiyalar murojaat qilishlari mumkin. Sinf a'zolariga murojaat qilish imkonini belgilash juda muhim xususiyat bo'lib, bu masalani yechishda uncha katta tajribaga ega bo'lmagan dasturlarchilar ko'pincha qiyinchiliklarga duch keladilar. Bu holatni batafsilroq tushuntirish uchun mavzuni boshida keltirilgan masalamizga qaytamiz.


```
Class Mushuk {
 unsigned int itsYosh;
 unsigned int itsOgirlik;
 void Miyovlash(); }
```

Bu tarzda sinfni e'lon qilishda itsYosh va itsOgirlik maydonla ham, Miyovlash () usuli ham yopiq a'zo sifatida aniqlanadi. Dastur yuqoridagi tartibda Mushuk sinfi e'lon qilingan bo'lsa va bu si ekzemplyari bo'lgan ob`ektning itsYosh a'zosiga main() funksiya tanasidan turib murojaat qilsak, kompilyator xatolik ro'y berganli haqida xabar beradi.

```
Mushuk Baroq;
Baroq.itsYosh = 5 // Xatolik!
// Yopiq a'zoga murojaat qilish mumkin emas.
```

Statik elementlar hamda funksiyalar

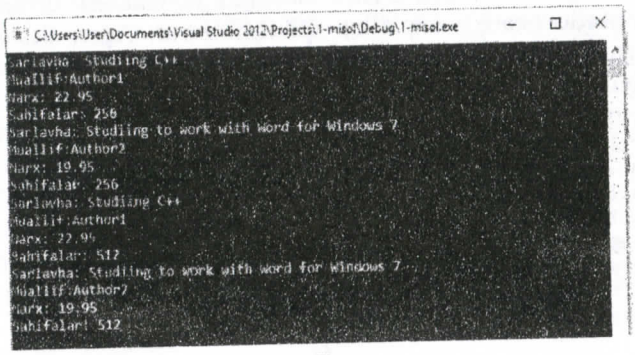
Shu paytgacha, har bir yaratilgan element o'zining xusu ma'lumotlar elementiga ega bo'lar edi. Lekin, shunday holat bo'lad bitta sinf doirasidagi ob`ektlarning ba'zi elementlari o'zaro bog'lan bo'ladi. Masalan, ish vaqti bir xil bo'lgan 1000 ta ishchining oy maoshini hisoblaydigan dastur tuzish taklif qilinayotgan bo'lsin. Sc stavkasini aniqlash uchun dastur har bir ishchining sharoitini bil kerak. Buning uchun aytaylik, state_of_employee nomli sinf foydalanamiz. Agar, ishchilar bir xil sharoitda ishlasa, demak, da barcha employee tipidagi ob`ektlar uchun (barcha ishchilar uch ushbu elementlardan o'zaro moslikda foydalanadi. Ushbu hola dastur, bitta axborotning 999 ta nusxasidan foydalanish bilan xotira foydalanish hajmini kamaytiradi.

Sinfning elementidan o'zaro moslikda foydalanish uchun, us element **static** (statik) deb e'lon qilinishi zarur. Agar, dastur us elementga yangi qiymat o'zlashtirsa, hamma ob`ekt elementi us yangi qiymatni qabul qiladi. Sinf elementi statik deb e'lon qilingani so'ng, u umumiy (global) o'zgaruvchi sifatida e'lon qilinishi zarur.

```
1. #include "stdafx.h"
2. #include <string.h> //strcpy() uchun
3. #include <stdio.h> //printf() uchun
4. #include <conio.h> //_getch() uchun
5. using namespace std;
6. class book_series{
7. public:
8. book_series(char *, char *, float);
```

```
9. void show_book(void); void set_pages(int);
10. private:
11. static int page_count; /*bu umumiy element hisoblanadi*/
12. char title[64]; char author[64];
13. float price; };
14. int book_series::page_count; /*sinfdan tashaoidaumumiy
    o‘zgaruvchini e’lon qilish*/
15. void book_series::set_pages(int pages){
16. page_count = pages; }
17. book_series::book_series(char *title, char *author, float price){
    /*Sinfning konstruktori*/
18. strcpy(book_series::title, title); /*string sinfiga ulanish uchun
    zarur bo‘lgan, strcpy() funksiyasi*/
19. strcpy(book_series::author, author);
20. book_series::price = price; }
21. void book_series:: show_book (void){
22. printf("Sarlavha: %s\n",title); printf("Muallif:%s\n",author);
23. printf("Narx: %.2f\n",price);
24. printf("Sahifalar: %d\n",page_count); }
25. void main(){
26. book_series programming("Studiing C++", "Author1", 22.95);
    /*programming ob`ektini konstruktor yordamida yaratish*/
27. book_series word( "Studiing to work with Word for Windows 7",
    "Author2", 19.95); /*word ob`ektini konstruktor yordamida
    yaratish*/
28. word.set_pages(256); /*Word o`ektining sahifalari soni beriladi,
    bu programmingga ham ta’sir qiladi */
29. programming.show_book ();
30. word.show_book() ;
31. programming.set_pages(512); /*page_countni o‘zgartirish*/
32. programming.show_book(); /*ob`ekt ma’lumotlarini ekranga
    chiqarish*/
33. word.show_book(); /*ob`ekt ma’lumotlarini ekranga chiqarish*/
34. _getch(); }
```

Natijasi:



Ob`ekt mavjud bo‘lmaganda, public static atributli elementlardan foydalanish

Sinfning barcha ob`ektlarida o‘zaro moslikda foydalaniladigan, elementi ***static*** sifatida e’lon qilinishi tushunarli bo‘ldi, lekin, shunday holat bo‘lishi mumkin: hech qanday ob`ekt yaratilmagan, ammo, ushbu elementdan foydalanish zarur. Dasturda bu elementdan foydalanish uchun, uni ***public*** hamda ***static*** deb e’lon qilish zarur. Ushbu dasturda xuddi shu holatga e’tibor qaratilgan.

Bu holatni ifodalaydigan dasturning kodi quyida ifodalangan:

```
1. #include "stdafx.h"
2. #include <string.h> //strcpy() uchun
3. #include <stdio.h> //printf() uchun
4. #include <conio.h> //_getch() uchun
5. using namespace std;
6. class book_series{
7. book_series();
8. public:
9. static void show_book(void); /*Funksiyani statis elementini chop
   etish uchun, ushbu atribut qo‘shiladi*/
10. static int page_count;
11. private:
12. char title [64];
13. char author[64];
14. float price; };
15. int book_series::page_count; /*O‘zgaruvchini global o‘zgaruvchi
    sifatida e’lon qilish*/
16. void book_series::show_book (void){
17. printf("Sahifalar soni=%d\n",page_count); }
18. int main(void){
19. book_series::page_count = 256;
20. book_series::show_book();_getch(); }
```

Natija: Sahifalar soni = 256

Nazorat savollari:

1. Sinf ichidagi ma’lumotlarni himoyalashning nechta xil usuli bor?
2. Sinfning **static** elementi qanday e’lon qilinadi?
3. Sinf elementini qachon **private static** ko‘rinishida e’lon qilishg zarurat tug‘iladi?
4. Sinf elementini qachon **public va static** ko‘rinishida e’lon qilishg zarurat tug‘iladi?
5. **private** ko‘rinishida himoyalangan elementga **int main()** funksiy

orqali qiymat o'zlashtirib bo'ladimi? **Static** atributi bilan e'lon qilingan elementgachi?

2.7. Visual C++ muhitida polimorfizmni qo'llash

Polimorfizm asoslari

Polimorfizm yunoncha so'z bo'lib, ikkita o'zakdan — **poly**(ko'p) va **morphos**(shakl) dan iborat bo'lib, ko'p shakllilikni bildiradi.

Polimorfizm — bu turdosh ob`ektlar (ya'ni bitta ajdod hosilasi bo'lgan sinflarga mansub ob`ektlar) ning dastur bajarilish vaqtida vaziyatga qarab o'zlarini turlicha tuta olish xususiyati. Ob`ektga mo'ljallangan dasturlash doirasida dasturchi ob`ekt xulq-atvoriga faqat bilvosita ta'sir ko'rsatishi, ya'ni dasturga kiritilayotgan usullari o'zgartirilishi hamda avlodlarga o'z ajdodlarida yo'q bo'lgan o'ziga xos xususiyatlarni baxsh etishi mumkin.

Usulni o'zgartirish uchun uni avlodda ortiqcha yuklash kerak, ya'ni avlodda bitta nomdagi usulni e'lon qilish va unda kerakli xatti-harakatlarni ishga solish kerak. Natijada ajdod-ob`ekt va avlod-ob`ektda bitta nomdagi ikkita usul amal qiladi. Bunda ushbu usullarning kodlari turlicha ishga tushiriladi va demakki, ob`ektlarga turlicha xatti-harakat baxsh etadi. Masalan, geometrik shakllar turdosh sinflarining tabaqalanishida (nuqta, to'g'ri chiziq, kvadrat, to'g'riburchak, doira, ellips va h.k.) har bir sinf Draw usuliga ega bo'lib, u ushbu shaklni chizib berish talabi qo'yilgan voqea-hodisaga tegishli javob berilishi uchun mas`uldir.

Polimorfizm tufayli avlodlar bitta voqeaga o'ziga xos tarzda munosabat bildirish uchun o'z ajdodlarining umumiy usullarini ortiqcha yuklashlari mumkin.

Virtual funksiyalar

Ob`ektga mo'ljallangan dasturlashda polimorfizmga nafaqat yuqorida tavsifi berilgan vorislik va ajdod usulini ortiqcha yuklatish mexanizmi vositasida erishiladi, balki virtuallash vositasida ham erishiladiki, u ajdod funksiyalarga o'z avlodlari funksiyalariga murojaat qilish imkonini beradi. Polimorfizm sinf arxitekturasi orqali ishga tushiriladi, biroq faqat a'zo-funksiyalar polimorf bo'lishlari mumkin.

C++da polimorf funksiya bitta nomdagi ehtimoliy funksiyalardan

biriga faqat bajarilish paytida, ya'ni unga sinfning aniq ob`ekti uzatilayotgan paytda bog'lab qo'yiladi. Boshqacha qilib aytganda, dastlabki dastur matnida funksiyaning chaqirilishi faqat taxminan belgilanadi, aynan qanday funksiya chaqirilayotgani aniq ko'rsatilmaydi. Bu jarayon kechikkan bog'lanish deb nom olgan. Navbatdagi misol oddiy a'zo - funksiyalarning polimorf bo'lmagan xulq-atvori nimaga olib kelishi mumkinligini ko'rsatadi:

```
1. #include "stdafx.h"
2. #include <string.h> //strcpy() uchun
3. #include <iostream> //cout uchun
4. #include <conio.h> //_getch() uchun
5. using namespace std;
6. class Parent{
7. public:
8. double F1(double x){ return x*x;}
9.  double F2(double x){
10. return F1(x)/2; } };
11.class Child: public Parent
12.{ public:
13.double F1(double x){
14.return x*x*x; } };
15.int main() {
16.Child child;
17.cout << child.F2(3)<<endl; _getch(); }
```

Natija: **4.5**

Parent sinfi F1 va F2 a'zo-funksiyalarga ega. Bunda F1 ni F2 chaqiradi. **Parent** sinfining hosilasi bo'lgan **Child** sinfi F2 funksiyasiga vorislik qiladi, biroq F1 funksiyasini oldindan belgilaydi. Kutilayotgan 13.5 natijasi o'rniga dastur 4.5 qiymatni chiqarib beradi. Gap shundaki kompilyator **child.F2(3)** ifodasini meros qilib olingan **Parent::F2** funksiya murojaatiga translyasiya qilib yuboradi, bu funksiya esa o'z navbatida **Child::F1** ni emas, **Parent**::F1 ni chaqiradi. Shunday bo'lganda edi, polimorf xulq-atvor qo'llab-quvvatlangan bo'lar edi.

C++ kechikkan bog'lanishni funksiya bajarilish paytida aniqlayd hamda funksiyalarni virtuallash vositasida ularning polimorf xulq atvorini ta'minlaydi. Bazaviy va hosilaviy sinflarda virtual funksiyalarn e'lon qilish sintaksisini umumlashtiradigan misolni ko'rib chiqamiz:

class className1{
//Boshqa a'zo-funksiyalar
virtual return Type functionName (<parametrlar ro'yxati>); }

class className2 : public className1 {
//Boshqa a'zo-funksiyalar
virtual return Type functionName (<>); }

Parent va **Child** sinflari ob`ektlarida F1 funksiyasining polimorf xulq-atvorini ta'minlash uchun uni virtual deb e'lon qilish zarur.

Quyida dasturning yangilangan matni keltiriladi:

```
1. #include "stdafx.h"
2. #include <string.h> //strcpy() uchun
3. #include <iostream> //cout uchun
4. #include <conio.h> //_getch() uchun
5. using namespace std;
6. class Parent{
7. public:
8. virtual double F1(double x){
9. return x*x; }
10.double F2(double x){
11.return F1(x)/2;} };
12.class Child:public Parent{
13.public:
14.virtual double F1(double x){
15.return x*x*x;} };
16.int main() {
17.Child child;
18.cout<<child.F2(3)<<endl;
19._getch(); }
```

Mana endi dastur kutilayotgan 13.5 natijasini chiqarib beradi. Kompilyator **child.F2(3)** ifodasini meros qilib olingan **Parent::F2** funksiya murojaatiga translyasiya qilib yuboradi, bu funksiya esa, o'z navbatida, **Child::F1** avlodining qayta aniqlangan virtual funksiyasini chaqirib oladi.

Agar funksiya bazaviy sinfda virtual deb e'lon qilingan bo'lsa, uni faqat hosila sinflarda qayta aniqlash mumkin, bunda parametrlar ro'yxati avvalgidek qolishi zarur. Agar hosila sinfning virtual funksiya parametrlar ro'yxatini o'zgartirgan bo'lsa, bu holda uning bazaviy sinfdagi (hamda uning barcha ajdodlaridagi) versiyasi kirib bo'lmas bo'lib qoladi. Boshida bunday vaziyat boshi berk ko'chaga kirib qolgandek ko'rinishi mumkin, amalda ortiqcha yuklanish mexanizmini qo'llab-quvvatlamaydigan ob`ektga mo'ljallangan dasturlash tillarida shunday bo'ladi ham. C++ bu muammoni virtual funksiyalardan emas, balki xuddi shu nomli, faqat boshqa parametr ro'yxatiga ega bo'lgan ortiqcha yuklangan funksiyalardan foydalangan holda hal qiladi.

Virtual deb e'lon qilingan funksiya, hosila sinflarda virtual kalit so'z bilan e'lon qilingani yoki qilinmaganidan qat`iy nazar, barcha hosila sinflarda virtual hisoblanadi.

Virtual funksiyalardan berilgan sinf ob`ektlarining o'ziga xos xulq-atvorini ishga solish uchun foydalaning. Barcha usullaringizni virtual deb e'lon qilmang, bu ularni chaqirishda qo'shimcha hisoblash sarflariga olib keladi. Hamma vaqt destruktorlarni virtual deb e'lon qiling. Bu sinflar tabaqalanishida ob`ektlarni yo'q qilishda polimorf xulq-atvorni ta'minlaydi.

Polimorf ob`ekt-telefonning yaratilishi

Aytaylik, sizning boshliqlaringiz sizga ob`ekt-telefoningiz diskli, tugmachali yoki to'lovli telefonlardan birini tanlab olib, emulyasiya qila olishi kerak dedi. Boshqacha qilib aytganda, ob`ekt-telefon bitta qo'ng'iroq uchun tugmachali apparat sifatida, boshqa qo'ng'iroq uchun to'lovli telefon sifatida va h.k. ishlashi kerak. Ya'ni bir qo'ng'iroqdan ikkinchisiga sizning ob`ekt-telefoningiz o'z shaklini o'zgartirishi lozim bo'ladi.

Turli sinflarga mansub bu telefonlarda faqat bitta farqlanuvchi funksiya mavjud — bu ideal usuli. Polimorf ob`ektni yaratish uchun siz avval bazaviy sinf funksiyalarini, ularning prototiplari oldidan virtual kalit so'zini qo'ygan holda aniqlaysiz. Bu bazaviy sinf funksiyalari hosila sinflar funksiyalaridan virtualligi bilan farqlanadi.

Keyin dasturda bazaviy sinf ob`ektiga ko'rsatkich tuziladi. Siznin telefonga tuzilayotgan dasturingiz uchun siz **phone** bazaviy sinfig ko'rsatkich tuzasiz:

phone *poly_phone;

Ob`ekt shaklini o'zgartirish uchun siz, quyida ko'rsatilganide ushbu ko'rsatkichga hosila sinf ob`ektining adresini berib qo'yasiz:

poly_phone=(phone*)&home_phone;

Qiymat berish operatoridan keyin keladigan (**phone***) belgil turlarga keltirish operatori bo'lib, bu operator C++ning kompilyatori hamma narsa joyida, bir turdagi o'zgaruvchi (**touch_tone**) adres boshqa turdagi o'zgaruvchi (phone) ga berish zarurligini ma'lum qila Dastur poly_phone ob`ekti ko'rsatkichiga turli ob`ektlar adresini taqd qilishi mumkin ekan, u holda bu ob`ekt ham o'z shaklini o'zgartiris demakki, polimorf bo'lishi mumkin.

Navbatdagi dastur bu usuldan ob`ekt-telefon yaratish uchun foydalanadi. Dastur ishga tushirilgach, **poly_phone** ob`ekti o`z shaklini diskli telefondan tugmachalisiga, keyin esa to`lovlisiga o`zgartiradi:

```
#include "stdafx.h"
#include <string.h> //strcpy() uchun
#include <iostream> //cout uchun
#include <conio.h> //_getch() uchun
using namespace std;
class phone{
public:
virtual void dial(char*number){
cout<<"Ulanish..."<<endl;}
phone(char*number){
strcpy(phone::number, number); };
protected:
char number[13]; };
class touch_tone:phone{
public:
void dial(char * number){
cout<<"Connecting by touch_tone..."<<endl;}
touch_tone(char*number):phone(number){} };
class pay_phone: phone{
public:
void dial(char *number){
cout<<"Iltimos! "<< amount <<" so`m to`lang"<<endl<<"Ulanmoqda...
  "<< number <<endl; };
pay_phone(char *number, int amount):phone(number){
pay_phone::amount = amount;}
private: int amount; };
int main(){
pay_phone city_phone("702-555-1212", 1500); /*to'lovli telefon
  obyekti*/
touch_tone home_phone("555-1212"); /*tugmachali telefon obyekti*/
phone rotary("201-555-1212") ; /* diskli telefon obyekti*/
  /* Obyekt diskli telefonga aylantirilsin*/
phone *poly_phone = &rotary;
poly_phone->dial("818-555-1212");
/* Obyekt shakli tugmachali telefonga o'zgartirilsin*/
poly_phone = (phone *) &home_phone;
poly_phone->dial("303-555-1210");
/*Obyekt shakli to'lovli telefonga o'zgartirilsin*/
poly_phone = (phone *) &city_phone;
poly_phone->dial("212-555-1212");
_getch(); }
```

Agar ushbu dastur kompilyatsiya qilinib ishga tushirilsa, ekranda quyidagi yozuv paydo bo`ladi:

Ulanish...

Connecting by touch_tone...
Iltimos! 1500 so`m to`lang
Ulanmoqda... 212-555-1212

Poly_phone ob`ekti dastur bajarilishi davomida o`z sha o`zgartirib turar ekan, u polimorf bo`ladi.

Do`stona funksiyalar

Do`stona funksiyalar, garchi ular biror bir sinfga ma bo`lmasalarda, tashqi sinf ma'lumotlarining barcha **private** himoyalangan a'zolariga kirish huquqiga ega bo`ladilar. Do` funksiyalarning e'lon qilinish sintaksisini qaytarilayotgan ko`rsatkichi oldidan turgan **friend** kalit so`zi yordamida k chiqamiz:

class className{
public: ~
className(); //YAshirish konstruktor
//friend **return type;**
friendFunction ning boshqa konstruktorlari(<parametrlar ro`yxati>) };

Agar oddiy a'zo-funksiyalar, sinf nusxasiga yashirish paramet **this** ko`rsatkichini uzatish hisobiga o`z sinfining barcha ma'lumotl avtomatik tarzda kirish huquqiga ega bo`lsa, do`stona funksiyalar u parametrning ochiq-oydin spesifikasiyasini talab qiladi.

Darhaqiqat, X sinfida e'lon qilingan F do`stona funksiya bu s mansub emas, demakki, x.F va xptr->F (bu yerda x - X sinf nusxasi, xptr - uning ko`rsatkichi) operatorlari tomonidan chaqirib olmaydi. Bu o`rinda F(&x) yoki F(xptr) murojaatlari sintaktik jih to`g`ri (**correct**) bo`ladi. Shunday qilib, do`stona funksiyalar sin a'zo-funksiyalari vositasida ishga tushirilishi noqulay, qiyin va mumkin bo`lmagan masalalarni ham hal qilishlari mumkin.

Nazorat savollari:

1. Polimorfizm deganda nimani tushunasiz?
2. Virtual funksiyalar nima maqsadda ishlatiladi?
3. Virtual funksiyalar qanday e'lon qilinadi?
4. Siz geometrik shakllar (aylana va to`g`ri to`rtburchak) va h shakl uchun alohida Area() va Print() usullarini qo`llash l Buni qanday amalga oshirasiz?

2.8. Visual C++ muhitida sinflar orasidagi munosabatni va merosxo'rlikni qo'llash

Vorislikda murojaat xuquqlarini boshqarish

Vorislik o'zining barcha ajdodlarining xususiyatlari, ma'lumotlari, metodlari va voqealarini meros qilib oladigan hosila sinfini e'lon qilish imkoniyatini beradi, shuningdek yangi tavsiflarni e'lon qilishi xamda meros sifatida olinayotgan ayrim funksiyalarni ortiqcha yuklashi mumkin. Bazaviy sinfning ko'rsatib o'tilgan tavsiflarini meros qilib olib, yangi tug'ilgan sinfni ushbu tavsiflarni kengaytirish, toraytirish, o'zgartirish, yo'q qilish yoki o'zgarishsiz qoldirishga majburlash mumkin.

Hosila sinfni e'lon qilishning umumlashgan sintaksisi:

class <sinf nomi>: [<kirish xuquqini beruvchi sertifikator >] <ajdod sinf nomi> {...}

Sinf o'zining bazaviy sinfidan yuzaga kelayotganida, uning barcha nomlari hosila sinfda avtomatik tarzda yashirin **private** bo'lib qoladi. Ammo uni, bazaviy sinfning quyidagi kirish spertifikatorlarini ko'rsatgan holda osongina o'zgartirish mumkin:

private. Bazaviy sinfning meros bo'lib o'tayotgan (ya'ni ximoyalangan va ommaviy) nomlari hosila sinf nushalarida kirib bo'lmaydigan bo'lib qoladi.

public. Bazaviy sinf va uning ajdodlarining nomlari hosila sinf nusxalarida kirib bo'ladigan bo'ladi, barcha himoyalangan nomlar esa himoyalangan bo'lib qolaveradi.

Agarda yangi sinf **class** kalitli so'z yordamida aniqlangan bo'lsa unda hosila sinfdagi meros komponentalar **private** kirish statusiga ega bo'ladi, **struct** yordamida esa **public** statusiga.

Me`roslikda ko'rsatilmagan kirish statusini asosiy(bazaviy) sinf ismini oldidan ko'rsatilgan **private, protected** va **public** kirish atributlari yordamida o'zgartirish mumkin. Agarda V sinf quyidagicha aniqlangan bo'lsa:

```
class B { protected: int t;
public: char u; };
```

unda quyidagi hosila sinflarni kiritish mumkin:

class M: protected B { ... }; //t va u **protected** sifatida merosxo'r.
class P: public B { ... }; // protected, va u- public sifatida merosxo'r.

class D: private B { ... }; //t va u private sifatida merosxo'r.
struct F: private B { ... }; //t i u private sifatida merosxo'r.
struct G: public B { ... }; t - **protected** va u – public sifatida merosxo'r

Konstruktor va destruktorlarda vorislik

Konstruktorlar meros bo'lmagani uchun, hosila sinfni yara undan meros bo'lgan ma'lumot – a'zolari asosiy (bazaviy) konstruktori orqali inisializasiyalanishi lozim. Asosiy sinf konstr avtomatik ravishda chaqiriladi va hosila sinfni konstruktoridan bajariladi. Asosiy (bazaviy) sinfni konstruktorining parametrlari sinfni konstruktorini aniqlashda ko'rsatiladi. Shunday argumentlarni hosila sinfni konstruktoridan asosiy (bazaviy) konstruktoriga uzatish vazifasi bajariladi.

Masalan.

```
class Basis{
 int a,b;
public: Basis(int x,int y){aqx;bqy;} };
class Inherit:public Basis
{int sum;
public:
Inherit(int x,int y, int s):Basis(x,y){sum=s;} };
```

Sinf ob`ektlari pastdan tepaga qarab konstruktorlanadi: a asosiy(bazaviy), keyin esa komponent – ob`ektlar (agarda ular n bo'lsa), undan keyin esa hosila sinfning o'zi. SHunday qilib, sinfning ob`ekti quyi ob`ekt sifatida asosiy (bazaviy) sinf ob`ekti ichiga oladi. Ob`ektlar teskari tartibda o'chiriladi: avvalo hosila, uning komponent – ob`ektlari, undan keyin esa asosiy(bazaviy) ob

Shunday qilib, ob`ektni o'chirish tartibi uning konstrukt tartibiga nisbatan teskari bo'ladi.

Ko'plikdagi vorislik va virtual sinflar

Bu sinf ketma-ket (to'g'ri-to'g'ri) baza sinfidir, agar u b sinflarni aniqlashda ishlatilsa, baza ro'yxatidan chiqariladi. hollarda A sinf B sinfning bazasini ifodalasa va C uchun B ba bo'lsa, u holda B sinf C uchun to'g'ridan-to'g'ri baza hisob natijada A sinf C sinf uchun to'g'ri bo'lmagan baza bo'lib hisob

Quyida keltirilgan sinflarni tasvirlashda bazalar ishlab chiqilgan. Xuddi shu tartibda yangi baza sinflarini kompilyator e'lon qiladi.

Sinflar bir nechta ketma-ket sinflardan tashkil topishi mumkin, sinf bazasida ixtiyoriy son yo'qolishi mumkin, misol uchun,

```
class X1 { ... };
class X2 { ... };
class X3 { ... };
class Y1: public X1, public X2, public X3 {
... };
```

Bir necha to'g'ri baza sinflari mavjud bo'lib, ular ko'plik vorislari deb nomlanadi. Ko'plik vorislarida ketma-ket bazada hech qanday sinf bittadan ortiq ishlatilishi mumkin emas. Bitta sinf to'g'ri bo'lmagan sinfda bir necha marta ishlatilishi mumkin:

```
class X { ...; f () ; ... };
class Y: public X { ... };
class Z: public X { ... };
class D: public Y, public Z { ... };
```

Bu misolda X va Z sinflari D sinfiga voris bo'ladi. Bir xil nomdagi ob`ektlarni bartaraf qilishda to'g'ri bo'lmagan sinf bazalarining ko'plik vorislari virtual deb e'lon qilinadi. Buning uchun sinf bazalari ro'yxatida oldingi sinf nomini **virtual** kalit so'zini ishlatish kerak. Misol uchun X sinfi virtual baza sinfi bo'l ko'rinishda quyidagicha yoziladi:

```
class X { ... f() ; ... };
class Y: virtual public X { ... };
class Z: virtual public X { ... };
class D: public Y, public Z { ... };
```

Abstrakt sinflar

Xech bo'lmasa bitta sof (bo'sh) virtual funksiyaga ega bo'lgan sinf abstrakt sinf deyiladi. Quyidagi tavsifga ega bo'lgan komponentali funksiya sof virtual funksiya deyiladi:

virtual <tip> <funksiya_nomi>(<formal_parametrlar_ro'yxati>) = 0;

Abstrakt sinf hosila sinf uchun asosiy (bazaviy) sinf sifatida ishlatilishi mumkin. Abstrakt sinflarning mexanizmi keyinchalik konkretizasiyalanadigan umumiy tushunchalarni tavsiflash uchun ishlab chiqilgan. Bu holda, sinflar ierarxiyasini yaratish quyidagi sxema bo'yicha bajariladi. Ierarxiya asosida abstrakt bazaviy sinf turadi. U interfeysni meros qilib olish uchun foydalaniladi. Hosila sinflar bu interfeysni konkretizasiyalaydi va amalga oshiradi. Abstrakt sinfda sof

virtual funksiyalar e'lon etilgan, ular aslida **abstrakt usullar.**

Ba'zi sinflar masalan shape sinfi, abstrakt tushunchalarni ifodalaydi va ular uchun ob`ekt yaratib bo'lmaydi. Bunday sinflar biror hosila sinfda ma'noga ega bo'ladi:

```
class shape {
//...
public:
 virtual void rotate(int) q =0; //sof virtual funksiya
 virtual void draw() = 0; // sof virtual funksiya
};
```

Abstrakt sinfni faqat boshqa sinf ajdodi sifatida ishlatish mumkin:

```
class circle : public shape {
 int radius;
public:
 void rotate(int) { }
//qayta ta'riflash shape::rotate
 void draw();
//qayta ta'riflash shape::draw
circle(point p, int r); };
```

Agar sof virtual funksiya hosila sinfda to'liq ta'riflanmasa, u hosila sinfda ham sof virtual bo'lib qoladi, natijada hosila sinf ham abstrakt sinf bo'ladi.

Abstrakt sinflar realizatsiya detallarini aniqlashtirmasdan faqat interfeysni ko'rsatish uchun ishlatiladi. Masalan operasion tizimda qurilma drayveri abstrakt sinf sifatida berilishi mumkin:

```
class character_device { public:
 virtual int open() = 0;
 virtual int close(const char*) = 0;
 virtual int read(const char*, int) = 0;
 virtual int write(const char*, int) = 0;
                virtual int ioctl(int ...) = 0; };
```

Drayverlar character_device sinfining ajdodlari sifatida kiritilishi mumkin.

Nazorat savollari:

1. Konstruktordan voris olish nima uchun kerak?
2. Destruktordan qanday voris olinadi?
3. Ko'plikdagi vorislik qanaqa bo'ladi?
4. Abstrakt sinflar nima uchun ishlatiladi?

2.9. Visual C++ muhitida standart qoliplar kutubxonasi bilan ishlash.

Qo'shimcha yuklash ta'rifi

Standart amallarni (masalan +) qo'shimcha yuklash biror sinf bilan birga qo'llashda mazmunini o'zgartirishdan iboratdir. Standart amallarni qo'shimcha yuklash maxsus funksiya – komponenta kiritish yo'li bilan amalga oshiriladi. Qo'shimcha yuklash til standartiga asosan amalga oshiriladi, amallar belgisi va operandlar soni o'zgarmaydi.

Amallarni qo'shimcha yuklash uchun quyidagi ta'rifdan foydalaniladi:

<operator amal> (<operandlar ro'yxati>)

quyidagi amallarni qo'shimcha yuklash mumkin:

+ - * / % ^ & | ~ !

= < > += -= *= /= %= ^= &=

|= << >> >>= <<= == != <= >= &&

|| ++ -- [] ()new delete

Bu amallar ustivorligi va ifodalar sintaksisini o'zgartirish mumkin emas. Masalan unar amal % yoki binar ! amalni kiritish mumkin emas. Funksiya amal har qanday funksiya kabi ta'riflanadi va chaqiriladi.

Standart tiplar uchun to'rt amal ("+", "-", "*" va "&") ham unar ham binar amal sifatida ishlatiladi va qo'shimcha yuklanadi. Hamma qo'shimcha yuklangan amallar uchun operator () amalidan tashqari, ko'zda tutilgan argumentlardan foydalanish mumkin emas.

Amallar xossalaridan ba'zilaridan foydalaniladi. Xususan operator, operator [], operator () va operator -> nostatik komponenta – funksiya bo'lishi lozim.

Operator - funksiya yoki sinf komponentasi bo'lishi kerak yoki juda bo'lmasa bitta parametri sinf ob`ekti bo'lishi kerak (**new** va **delete** amallarini qo'shimcha yuklovchi funksiyalar uchun bu shart emas).

Operator - funksiya, birinchi parametri asosiy turga tegishli bo'lsa, funksiya-komponenta bo'lolmaydi.

C++ tilida quyidagi amallarni qo'shimcha yuklash mumkin emas:

. sinf ob`ekti a'zosiga murojaat;

.* ko'rsatkich orqali murojaat;

?: shartli amal;

:: ko'rinish sohasini ko'rsatuvchi amal;

sizeof hajmni hisoblash amali;

preprosessor amali.

Nazorat savollari:

1. Binar amallarni qo'shimcha yuklash deganda nimani tushunasiz?
2. Unar amallarni qo'shimcha yuklash deganda nimani tushunasiz?
3. Inkrement va dekrement amallarini qo'shimcha yuklash deganda nimani tushunasiz?
4. Indekslash va funksiyani chaqirish amallarini qo'shimcha yuklash deganda nimani tushunasiz?
5. Qiymat berish va inisializasiya deganda nimani tushunasiz?

Bob xulosasi

Mazkur bobda **Visual Studio** dasturini tizimga o'rnatish va VC++ ning **Console Application** muhitida dastur yaratish usullari va shartlar xatoliklar, ularnining turlari va ularni bartaraf etish usullar kutubxonalarni e'lon qilish va ularni chaqirish turlari, maxsus kutubxon (Math::) funksiyalaridan foydalanish usullari, berilganlarni kiritish v chiqarish, sonlar jadvalini konsolga chiqarish, lokal va glob o'zgaruvchilarni e'lon qilish va ularni qo'llanilishi, maxsus String tu va u bilan ishlash shartlari, statik va dinamik massivlar bilan ishlas fayllar bilan ishlovchi maxsus (IO) kutubxonasi fuksiyalari va ul yordamida turli xil kengaytmali fayllar bilan ishlash, sinf (class) struktura (struct) e'lon qilish turlari, ularning turli hil ko'rinishla qo'llanilish usullari, satrlar bilash ishlovchi maxsus funksiyalar haqi batafsil ma'lumotlar bilan tanishildi.

Nazariy savollar va amaliy topshiriqlar

1. Struktura bilan sinfni farqini tushuntirib bering.
2. Ob`ekt nima va uning xossalarini sanab bering.
3. Nima uchun metod ham sinfda ham strukturada bir xil ishlatiladi.
4. Amallarni qayta yuklashdan maqsad nima ekanligini tushunti bering.
5. Oddiy funksiya bilan virtual funksiyaning farqlarini sanab berng.
6. Ob`ekt bilan metodni farqini aytib bering.
7. Do'st funksiyalar maqsadini ayting.

8. Merosxo'rlik bilan polimorfizm orasidagi farqlarni sanab bering.
9. Har xil tiplarni o'zida jamlagan sinf yarating.
10. Har xil tiplarni o'zida jamlagan struktura yarating.
11. Har xil tiplarni o'zida jamlagan birlashma yarating.
12. Inisializatsiyani maqsadini ayting.
13. Ushbu sinfning vazifasini ayting.

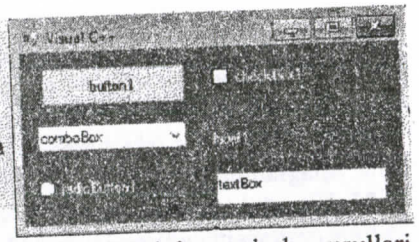
```
1. #include <iostream>
2. using namespace std;
3. class misol1{
4. private: int a;
5. int b;
6. public: void set(int a, int b);
7. public: void get();
8. };
9. void misol1::set(int a, int b){a=a; b=b;}
10. void misol1::get(){cout<<"a= "<<a<<endl<<"b= "<<b<<endl;}
11. int main()
12. {  misol1 A; int a=2, b=4;
13. A.set(a,b); A.get();
14. return 0;
15. }
```

16. Ushbu strukturaning vazifasini ayting.

```
1. #include <iostream>
2. using namespace std;
3. struct {
4. char ism[20];     char fam[20];     char sharf[20];
5. int Yoshi; float stependiya;  char guruh[10];
6. } talaba[100], A[100];
7. int main()
8. { char A; int n; cin>>n;
9. for(int i=0;i<n;i++)cin>>A[i].ism>>A[i].fam>>A[i].sharf>>A[i].yosh;
10. for(int i=0;i<n;i++)cout<<A[i].ism<<A[i].fam<<A[i].sharf<<A[i].yosh;
11. return 0;
12. }
```


3-BOB

Visual C++ ning Windows Application muhiti va unda ishlash



Windows application muhiti interfeysini sozlash usullari, boshqaruv elementlari – **ToolBar (windows Form)** haqida ma'lumotlar hamda komponentalar va ularni ishlatilishi yuzasidan amaliy ko'nikmalarga ega bo'lish.

Bob mundarijasi

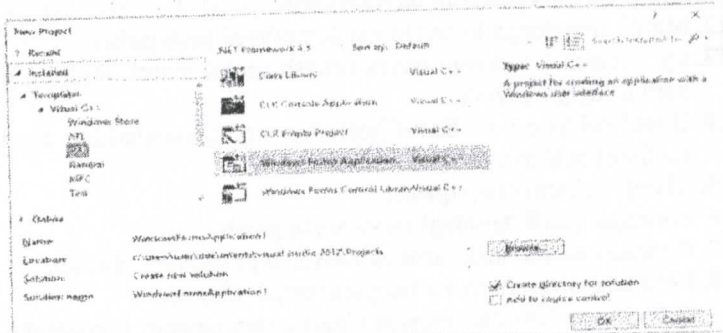
3.20. Veb brouzerda HTML jadvallarni tasvirlash va shakllantirish.
3.21. Visual C++da MS Word imkoniyatlaridan foydalanib, jadvallar yaratish va ularni Word faylga eksport qilish hamda taqdim etish.
3.22. Visual C++ da MS Excell imkoniyatlaridan foydalanib, diagrammalar yaratish va ularni turli kengaytmalarda saqlash.
3.23. Visual C++ ning Windows Application muhitida komponentalarning joylashish vaziyatlarini nazorat qilish.

3.1. Windows Form Application muhitini yaratish va yordamchi oynalarini sozlash

Windows Form Application muhiti paketini o'rnatish

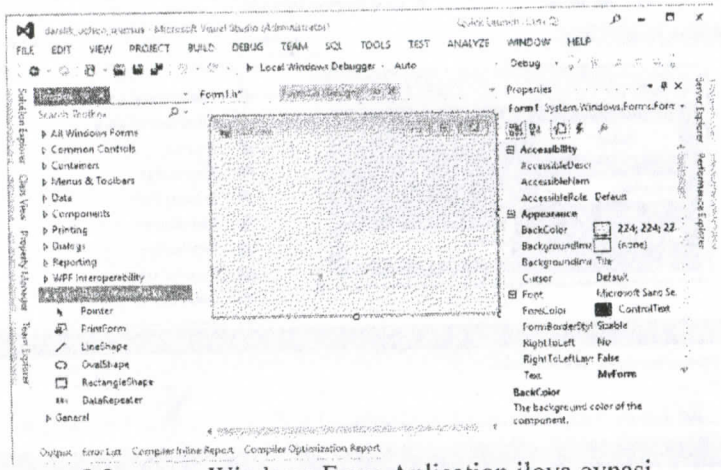
Visual Studio 2005 dan yuqori versiyalarida **Windows Form Application** joylashtirilmagan. Shuning uchun **Visual Studio 2012** dasturni tizimga o'rnatgandan keyin **Windows Form Application** paketini quyidagi internet manzidan yuklab olinadi: http://www.outme.ru/visual-c-2012-sozdanie-formyi.html. Paket yuklab olingandan keyin quyida ketma - ketliklar asosida tizimga o'rnatiladi:

1. Yuklab olingan arxiv faylni arxivdan chiqariladi.
2. Arxivdan chiqarilgan fayllar quyidagi manzilga tashlanadi: **C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\vcprojects\vcNET.**
3. **Visual Studio 2012** dasturi ishga tushiriladi va yangi loyiha yaratish oynasidan **Visual C++** va **CLR** tanlanadi(3.1- rasm).



3.1- rasm. Windows Form Aplication ilovasini yaratish oynasi

3.1-rasmda "Name" qismiga yaratiladigan loyiha nomi yoziladi (loyihaning nomi lotin harflaridan tashkil topgan bitta so'zdan iborat bo'lishi shart. Masalan: TUIT_loyihasi). **"Location"** qismida loyihani saqlanish joyi **"Browser"** tugmasi yordamida ko'rsatiladi. **"Solution"** qismida yangi ilova yaratish yoki yaratilgan loyihaga ilova qo'shish ko'rsatiladi. 3.1 - rasmda yangi ilova yaratish ko'rsatilgan. Qo'shish uchun esa **"Add to solution"** tanlanilishi kerak. **"Solution name"** qismiga yaratiladigan ilovaning nomi kiritiladi. **"OK"** tugmasi bosilsa, quyidagi **Windows Form Application** ilova oynasi ochiladi:

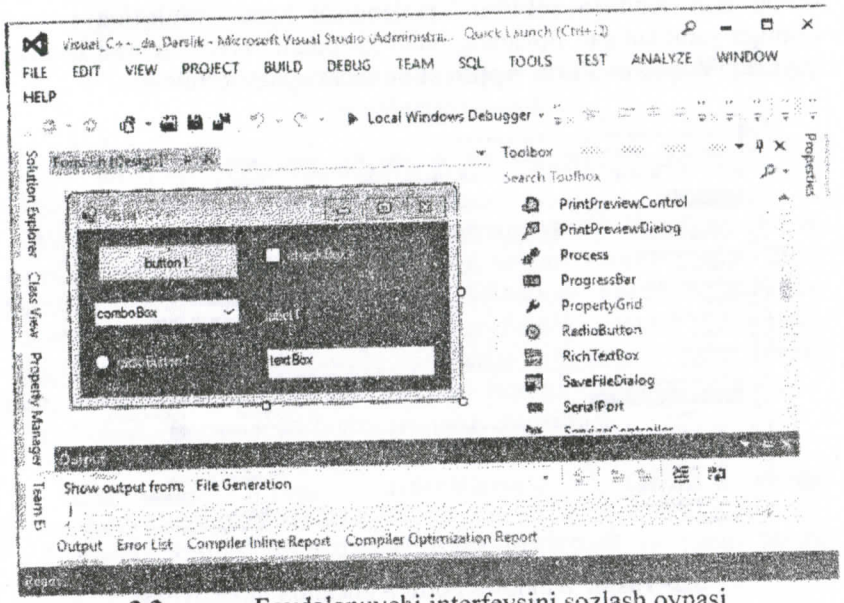


3.2- rasm. Windows Form Aplication ilova oynasi

3.2. Form, Button, Label komponentalari va MessageBox xabarlar oynasi

Biror masalaning dasturini tuzish uchun avval File menyusidan New Project buyrug'i ishga tushiriladi. Ochilgan New Project oynasining chap ustunida o'rnatilgan qoliplar (Installed Templates) ro'yxati turadi. Ular orasida Visual Studio muhitiga kiritilgan dasturlash tillarining qoliplari mavjud: Visual Basic, Visual C#, Visual C++, Visual F# va boshqalar. Ro'yxatdan Visual C++ tanlanadi. Visual C++ tugunidagi loyihalar turi ro'yxatining CLR muhitida loyiha nomi beriladi va oynaning o'ng qismida CLR Empty Project tanlanadi. Visual Studio muhiti oynasining o'ng qismida joylashgan loyihalar oynasida

sichqoncha o‘ng tugmasi bosilib, ochilgan muloqat oynasida Add new Items… buyrug‘i beriladi. Visual C++ qolipida UI qo‘yilmasining Windows Form satri tanlanadi. Forma nomi maydonida formaga nom beriladi (ko‘rsatilmaganda “MyForm.h”), ADD tugmasi bosiladi. Muhit ishchi maydonida quyidagi oyna hosil bo‘ladi.

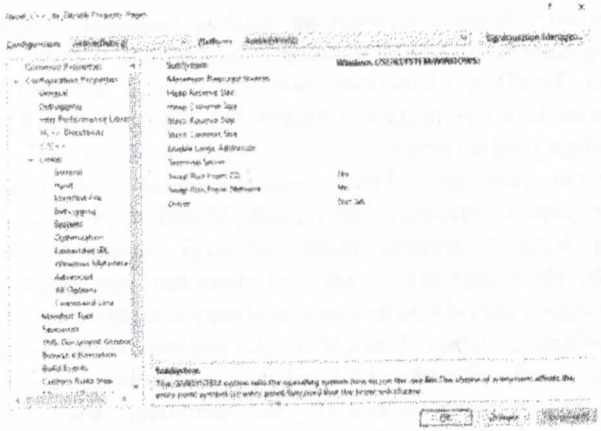


3.3- rasm. Foydalanuvchi interfeysini sozlash oynasi

Yaratilayotgan forma normal ishlashi uchun **MyForm.cpp** faylining mazmunini tahrirlaymiz. Buning uchun quyidagilarni yozish lozim bo‘ladi:

```
#include"MyForm.h"
using namespace System;
using namespace System::Windows::Forms;
[STAThread]
void main(array<String^>^ args){
Application::EnabledVisualStyles();
 Application::SetCompatibleTextRenderingDefault(false);
Project1::MyForm form;
Application::Run(%form); }
```


Bu yerda Project1 loyihaning nomi, MyForm esa forma qo‘shilganda berilgan nom. Loyiha oynasida Project1 nomida sichqoncha o‘ng tugmasi bosilib, Properties=>Linker=>System qo‘yilmasiga o‘tiladi va SubSystem maydoniga Windows(/SUBSYSTEM:WINDOWS) matni kiritiladi.



3.4- rasm. Formani sozlash oynasi

Advanced qo‘yilmasining Entery Point maydoniga main kalit so‘zi yoziladi.

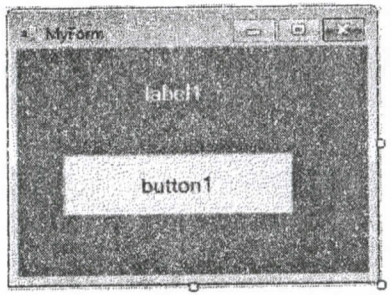
3.5- rasm. Formani sozlash oynasi

Zarur o‘zgaruvchilar kiritilgandan so‘ng OK tugmasi bosiladi. [F5] klavishini bosib, yaratilgan formani ishga tushirish va dastur natijasini ko‘rish mumkin.

Dasturchilar formaga foydalanuvchilar uchun grafik interfeysli komponentalarni, ya’ni boshqaruv elementlarini qo‘yadi. Bular matn kiritish uchun maydonlar (TextBox), buyruqli tugma (Button), formadagi matn satri (foydalanuvchi tomonidan o‘zgartirib bo‘lmaydigan)- belgilar (Label) va boshqa boshqaruv elementlaridir. Bunda eng zamonaviy vizual dasturlash qo‘llaniladi. Chunki sichqoncha yordamida ToolBox elementlar panelida mavjud bo‘lgan boshqaruv elementlarini formaga joylash mumkin. Bu o‘z navbatida dastur kodini kam yozishga yordam beradi.

Birinchi yaratilgan dastur bitta formadan tashkil topib, unda qandaydir yozuv, masalan “Microsoft Visual C++ 2013 ”, hamda “Tugmani bosing” matnini olgan buyruqli tugma bo‘lsin. Tugma bosilganda “Hammaga salom” xabarini olgan muloqot oynasi ochilsin.

Formaga yuqorida keltirilgan boshqaruv elementlarini qo‘yamiz. Buning uchun ToolBox boshqaruv elementlari paneli kerak bo‘ladi. Agar bu panel ekranda ko‘rinmayotgan bo‘lsa, [Ctrl]+[Alt]+[X] klavishlar kombinasiyasini bosish yoki menyudan View => ToolBox menyu osti buyrug‘ini tanlash lozim bo‘ladi. ToolBox panelining Label va Button elementlarida sichqoncha chap tugmasi ikki marta bosilib, formaga qo‘shiladi. So‘ngra elementlar 3.6- rasmda ko‘rsatilganidek joylashtiriladi.



3.6-rasm. Birinchi loyihaning formasi

Ixtiyoriy shunga o‘xshash ob`ektlarni o‘zingiz yaratishingiz yoki qoliplardan foydalanishingiz mumkin. Bu misolda formaga

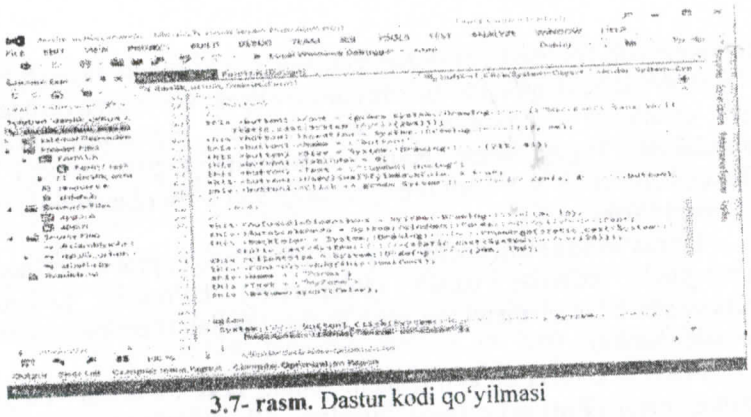
komponentni ToolBox panelidan sichqoncha yordamida tashlab tayyor vizual ob`ekt hosil qilinadi. Bunda har bir obyekt xossalarga (properties) ega ekanligini unutmaslik lozim. Masalan tugmaning xossalar quyidagilar: Tugma nomi (**Name**) – **button1,** tugma ustidagi matn (Text), tugmaning X,Y koordinata tekisligidagi o‘rni (**Location**), tugma o‘lchami (**Size**) va h.k.

Forma ichida sichqoncha o‘ng tugmasini bosib, properties menyusi tanlanganda xossalar paneli (**Properties**) ko‘rinadi. (3.3-rasm) Sichqoncha bilan elementlarni yoki formani tanlab ularning xossalarini ko‘rish mumkin.

Misolda **Label1** ob`ekti tanlanib, uning **Text** xossasiga “**Tugmani bosing**” matni (**Button1** o‘rniga) kiritiladi. Ob`ektlarda faqat xossalar mavjud emas, balki hodisalar ham qayta ishlanadi. Masalan tugma yoki forma ustida sichqoncha tugmasini bosish, dastur ishga tushganda formani tezkor xotiraga yuklanishi (Load) jarayonlari **hodisa** deb ataladi. Hodisalarni boshqarish uchun dastur kodida ularni qayta ishlovchi funksiyalar yoziladi. Buning uchun avval “Bo‘sh” hodisani qayta ishlovchi olishi kerak. Bu misolda tugma bosilganlik hodisasi qayta ishlanadi. Bo‘sh qayta ishlovchini olish uchun button tugmasining xossalar panelidan chaqmoq rasmli Events (hodisalar tugmasi bosilib, hodisalar ro‘yxatidan Click hodisasi tanlanadi (uning ustida sichqoncha tugmasi ikki marta bosiladi). Natijada MyForm.h dastur kodi qo‘yilmasiga o‘tiladi (3.4- rasm).

MyForm.h qo‘yilmasida Visual C++ tomonidan ko‘pgina kodlarning generatsiya qilinganligini ko‘rish mumkin. E’tibor bersangiz bu matnlar ichida siz Proporties panelida kiritilgan tugmalarni ko‘rishingiz mumkin. Masalan, button1 tugmasining Text xossasiga bergan qiymat uchun quyidagi kod generatsiya qilinadi: **this->button1 >Text = “tugmani bosing”**; 3.7 -rasmda dastur kodining bir qismi ko‘rsatilgan bo‘lib yuqorida kerak bo‘lgan hodisani qayta yuklovchi bo‘sh **button1_Click** funksiyasini ko‘rish mumkin.

```
private:System::Void button1_Click(System::Object^ sender
Ssytem::EventArgs^ e){}
```

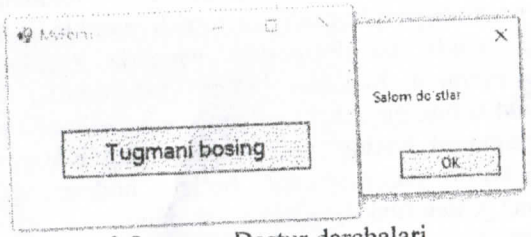


3.7- rasm. Dastur kodi qo'yilmasi

Figurali qavslar ichiga tugma bosilganda bajariluvchi dastur kodi yoziladi. Ahamiyat bergan bo'lsangiz yuqorida ikkita qo'yilma MyForm.h va MyForm.h[Disign] hosil bo'ladi. O'z navbatida, bular dastur kodi va dasturning vizual ko'rinishidir. Tugma bosilganda "**Salom do'stlar**" muloqot oynasi ochilishi uchun figurali qavslar ichiga quyidagi matn yoziladi:

```
MessageBox::Show("Salom do'stlar");
```

Bu yerda `MessageBox` ob`ektning `Show` funksiyasi ishga tushirilmoqda. Demak button1 ob`ektining bosilganlik (Click) hodisasi qayta ishlandi. [F5] tugmasi bosilib, dastur ishga tushiriladi. Natijada 3.8- rasmda ko'rsatilgan oyna ochiladi.



3.8- rasm. Dastur darchalari.

MS Visual C++ da tashqi ko'rinishga ega bo'lgan dastur yaratildi. Yaratilgan dasturni ochish uchun dastur yaratilgan katalogda Proje[illegible].sln faylida sichqoncha chap tugmasi ikki marta bosiladi.

Mustahkamlash uchun mashqlar

1. Uchta o'zgaruvchining yig'indisini xabarlar oynasiga chiqaring.
2. Tugmani N marta bosgandan keyin N ning qiymatini xabar oynasiga chiqaring.
3. Button tugmasining Text xossasiga tugma bir marta bosilsa raqamini, 2 marta bosilsa 2 raqamini chiqaring.
4. Button tugmasini har bosganda tugmaning rangini o'zgartiring.
5. Button tugmasining Text xossasiga massivning elementlar ketma-ket chiqaring.

Nazorat savollari:

1. Button komponentasining vazifasi nimadan iborat?
2. Label komponentasining vazifasi nimadan iborat?
3. Buttonning asosiy 5 ta xossa va hodisalarini sanang?
4. Labelning asosiy 5 ta xossa va hodisalarini sanang?
5. Xabarlar oynasi qanday tipdagi ma'lumotlarni chiqaradi?

3.3. MouseHower hodisasi

Masalani oldingi na'munadagidan bir mun murakkablashtiramiz. Label1 ob`ekti uchun sichqoncha MouseHower hodisasiga ishlov berish uchun qo'shamiz. MouseHo hodisasi – foydalanuvchi biror ob`ekt ustida sichqoncha ko'rsatg elementga olib borilganda ro'y beradi. Bundan tashqari, MouseE (kirish) hodisasi ham mavjud bo'lib, u sichqoncha ko'rsatg boshqaruv elementi sohasining ichiga kirganda ro'y beradi. Shu ta ushbu na'munadagi dastur, ekran formasida Label-matn nishoni button tugmasiga ega bo'lishi kerak. Formani jixozlashda Too panelidan Label nishoni va Button tugmasi joylashtiriladi. Dastur ko uchta hodisalarga ishlov beruvchilar qo'shiladi. Buning u Proporties panelida chaqmoq belgisini (Events) va ketma-ket forn yuklash Form_Load, tugmaga bosish button1-Click label1_MouseHover hodisalarida sichqonchaning chap tugmasi marta bosiladi.

Bunda MyForm.h dastur kodi qo'yilmasiga o'tish am oshiriladi va uchta bosh hodisalarga ishlov beruvchilar hosil bo' Masalan oxirgi hodisaga ishlov beruvchi quyidagi ko'rinishda bo'la

```
private:      System::Void      label1_MouseHover(System::Object^sender,
System::EventArgs^  e){}
```

Figurali qavslar orasiga muloqat oynasini chaqirish kodi joylashtiriladi.

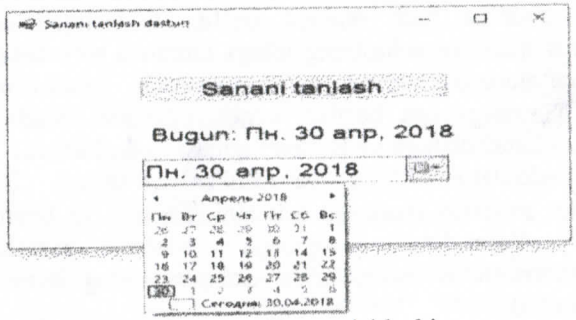
```
MessageBox::Show("Salom do'stlar");
```

Dastur imkoniyatlarini tekshirib ko‘rish uchun [F5] klavishasi bosiladi. Ob`ektlar xususiyatlari dastur kodida yoki forma komponentalari inisializatsiya qilingandan keyin yoki MyForm_Load hodisasini ishlab chiqayotganda aniqlanadi.

3.4. TextBox va DataTimePicker komponentalari

DataTimePicker komponentasi va uning xossalari

Zarur sanani tanlash uchun mos boshqaruv elementini bosilganda ekranda kalendar paydo bo‘lishi masalasini ko‘rib chiqaylik. Bu vazifani bajarish uchun Visual Studio 2012 dasturi ishga tushiriladi, asosiy oyna menyu bo‘limlaridan File->New->Project… buyruqlari beriladi yoki Ctrl+shift+n klavishalari bosiladi, ochilgan oynada loyihaga nom beriladi va OK tugmasi bosiladi. Toolbox dizayneri panelidan formaga Button buyruq tugmachasi, Label nishoni va DataTimePicker komponentalari joylashtiriladi. DataTimePicker komponentasi bevosita vaqtni tanlash funksiyasini bajaradi. Agar bu elementning ko‘rsatgich belgisi tanlansa, vaqtni tanlash uchun 3.9- rasmda ko‘rsatilgandek kalendar paydo bo‘ladi. Kerakli sana tanlanganidan so‘ng, Button tugmasini bosish orqali DataTimePicker elementining kalendari yopilishi va tanlangan vaqt Label nishoning matnli maydonida paydo bo‘lishi lozim.



3.9- rasm. Dastur ishlashi

Formada sichqonchaning chap tugmasini bosish orqali dastur ko oynasiga o‘tiladi. Xuddi shunday usul bilan Button tugmasini hodisa yaratiladi. Bundan tashqari ValueChanged hodisasi bilan ishlash zarura yuzaga keladi. Uni hosil qilish uchun DataTimePicker nir ValueChanged hodisasida sichqoncha chap tugmasi ikki marta bosilad Natijada kodlar oynasida quyidagi kodlar terilishi kerak:

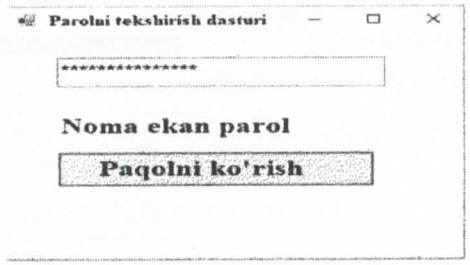
```
1- #pragma endregion
2- // Kerakli vaqtni tanlash dasturi
3- private: System::Void Form1_Load(System::Object^  sender,
4- System::EventArgs^  e) {
5- // Formani yuklash hodisasini hosil qilish
6- this->Text="Sanani tanlash dasturi";
7- dateTimePicker1->Format = DateTimePickerFormat::Custom;
8- dateTimePicker1->CustomFormat ="ddd. dd MMM, yyy";
9- button1->Text = "Sanani tanlash"; label1->Text =
   String::Format("Bugun:
10-     {0}", dateTimePicker1->Text);     }
11- private: System::Void
   dateTimePicker1_ValueChanged(System::Object^
12- sender, System::EventArgs^  e) {  // Sanani o‘zgartirish
   hodisasini hosil qilish
13- label1->Text = String::Format("Tanlangan sana: {0}",
   dateTimePicker1->Text);
14-     }
15-     private: System::Void button1_Click_1(System::Object^
   sender,
16-     System::EventArgs^  e) {
17-     //  dateTimePicker1 elementiga fokusni berish
18-     dateTimePicker1->Focus();
19-     // <F4> klavishi bosilishini imitasiya qilinadi
20-     SendKeys::Send("{F4}");} };
21-     }
```

Dasturda formani yuklash hodisasi bilan ishlashda sanani a ettirishning kerakli formati beriladi: birinchi 3 ta d harflari hafta kuni 3 ta MMM harflari qisqa shaklda kun va oy nomini hamda 4 ta y harfl yilni anglatadi. Sanani o‘zgartiruvchi ValueChanged hodisasi bil ishlashda label1 matnli nishonga tanlangan sana qiymati o‘zlashtirila Bunda String::Format usulidan foydalanamiz. “Tanlangan sana :{0}” foydalaniladigan format qiymatini anglatadi. Sanani tanlash hodisa bilan ishlashda dateTimePicker1 boshqaruv elementiga fokus berila

TextBox komponentasining PasswordChar xossasi kiritilayotgan qiymatlarni ma'lum bir ko'rinishga o'zgartirish imkonini beradi. Misol uchun TextBox dan kiritilgan parolni label da ko'rsatish dasturini tuzishni ko'ramiz. Buni amalga oshirishning 2 xil usuli mavjud: Birinchisi dizayn **[Design]** oynada TextBox komponentasi tanlanib, Properties darchasidan PasswordChar xosasiga * belgi yoki boshqa biron belgi kirtiladi (**3.11- rasm**). Ikkinchi usulini dastur kodida amalga oshiriladi. Buning uchun Button buyruq tugmasini sichqoncha chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. // Formani yuklash hodisasini hosil qilish
2. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {
3. this->Text="Parolni tekshirish dasturi";
4. button1->Text = "Parolni ko'rish";
5. label1->Text = String::Empty;
6. textBox1->Clear();
7. textBox1->PasswordChar ='*'; }
8. // Button buyruq tugmasini hodisasini hosil qilish
9. private: System::Void button1_Click_1(System::Object^  sender,
10.      System::EventArgs^  e) { label1->Text = textBox1->Text; }
11.}; }
```

Dastur natijasini ko'radigan bo'lsak, Forma yuklanganda **textBox1->PasswordChar ='*';** kodi, **"Parolni ko'rish"** tugmasi bosilganda **label1->Text = textBox1->Text;** kod bajarilmoqda(**3.12-rasm**).



3.12- rasm. Dastur natijasining oynasi

Mustahkamlash uchun mashqlar

1. **DataTimePicker** yordamida joriy sanani Label da chiqarish.
2. TextBox ga kiritilgan parolni xabarlar oynasida chiqarish.

3. Button tugmasining Text xossasiga joriy parolni shifirlang ko'rinishda chiqarish.
4. **TextBox** yordamida joriy parolni o'zgartirish.
5. Parol 3 marta noto'g'ri terilgandan keyin TextBox ko'rinmaydigan qilish.

Nazorat savollari:

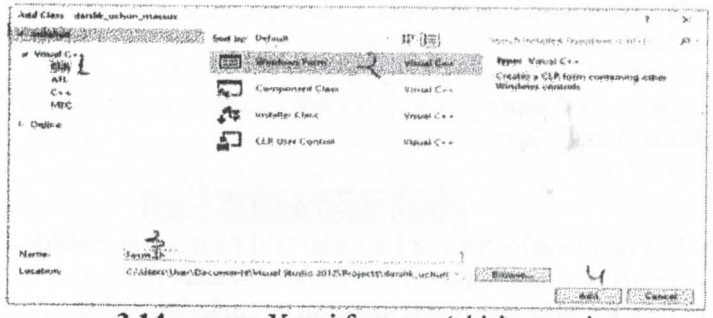
1. Button komponentasining doubleClick hodisasining vazifasi nim
2. TextBox komponentasining vazifasi nimadan iborat?
3. TextBox ning asosiy 5 ta xossa va hodisalarini sanang?
4. **DataTimePicker** ning asosiy 5 ta xossa va hodisalarini sanang?
5. **DataTimePicker** yordamida tizimni vaqtini o'zgartir mumkinmi?

3.5. Forma (Form) lar bilan ishlash

Formalarni formalar bilan bog'lovchi vizual dastur tuzish uc quyidagi ketma - ketliklar amalga oshiriladi. **Visual Studio 2** dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->N >Project...** buyruqlari beriladi yoki **Ctrl+shift+n** klavishalari bosil ochilgan oynada loyihaga nom beriladi va OK tugmasi bosil **Toolbox** dizayneri panelidan formaga **label** metrikasi va **Button** buy tugmasi joylashtiriladi va formaga yangi forma qo'shiladi(**3.13 va 3 - rasmlar**). Loyihaga yangi forma qo'shish:

3.13 – rasm. Yangi forma qo'shish oynasi

Yangi formaga **"Form2"** nomini berish:



3.14 – rasm. Yangi forma qo‘shish oynasi

Add tugmasi bosilgandan keyin dasturda **Form2** nomidagi yangi forma paydo bo‘ladi. 1-formaga quyida kodlar yoziladi:

```
#pragma once
#include "form2.h"
```

Button buyruq tugmasini sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. // Button buyruq tugmasini hodisasini hosil qilish
2. private: System::Void button1_Click_1(System::Object^  sender,
   System::EventArgs^  e) { Form2 ^Form2_ga = gcnew Form2(this);
3. Form2_ga->Show();      }
```

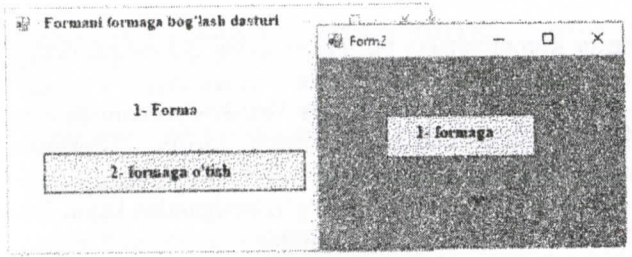
2-formadan 1- formaga qaytish uchun 2-formaning kodlar oynasiga quyidagi kodlar teriladi:

```
1. // Form2 ning private: metodini yaratish
2. private: Object^ Form1_ga;
3. public:
4. Form2(Object ^ ob){
5. Form1_ga = ob;
6. InitializeComponent();
7. //TODO: Add the constructor code here
8. }
```

2-formaga **Button** tugmachasi joylashtiriladi, uning ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

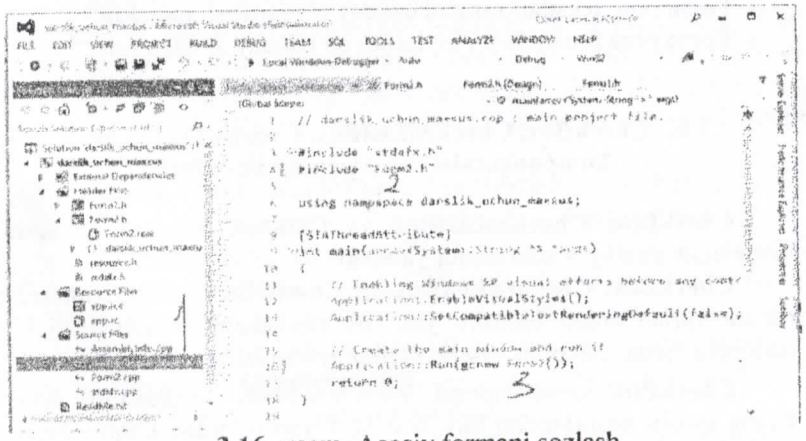
```
// Button buyruq tugmasini hodisasini hosil qilish
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^  e) { safe_cast<Form^>(Form1_ga)->Show();
3. this->Close();  }
```

Dastur natijasi quyidagicha:



3.15- rasm. Dastur natijasining oynasi

Agar loyihada bir nechta formalar bo‘lsa, ulardan 2- formani dastur ishga tushishida 1- bo‘lib ochilishi kerak bo‘lsa, loyiha strukturasi (**Solution Explorer**) dan quyidagi amallar bajariladi (**3.16- rasm**):



3.16- rasm. Asosiy formani sozlash.

Eslatib o‘tish kerakki, asosiy formani boshqa formalardan turib, **Close()** usuli bilan yopish tavsiya etilmaydi. Chunki asosiy forma yopilsa qolgan formalar ham avtomatik yopiladi. Ya’ni dastur to‘xtatiladi. Yangi formaga o‘tganda avvalgi forma ko‘rinmasligi uchun **Hide()** usuli ishlatilinadi.

Mustahkamlash uchun mashqlar

1. 1-formaning TextBox maydoniga kiritilgan matnni 2-formaning Label ida chiqarish;

2. Dastur kompilyatsiya bo'lganda avval 2-formani ochish va undan 2-formaga butun qiymatlar yuborish;

3. 2-formaning rangini 1-formada TextBox yordamida o'zgartirish;

4. Matrisa elementlarini 1-formada kiritib, 2-formada ularning yig'indisini chiqarish;

5. 1-formada parol 3 marta noto'g'ri terilgandan keyin 3-formani, aks holda 2-formani ochish.

Nazorat savollari:

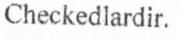
1. **Form** ning **onClosing()** hodisasining vazifasi nima?
2. **Form** ning **Load()** hodisasining vazifasi nimadan iborat?
3. **Form** ning asosiy 5 ta xossa va hodisalarini sanang?
4. **Form** ning tekstlari rangini qanday o'zgartirish mumkin?
5. **Form** orqa foniga rasm joylashtirish qanday amalga oshiriladi?

3.6. CheckBox, CheckedListBox, ComboBox, ListBox komponentalari va ularning xossalari

CheskBox, CheckedListBox va **ComboBox** komponetalaridan foydalanib, amaliy vizual dastur yaratish.

CheckBox, CheckedListBox va **ComboBox** komponentalari bir nechta variantlardan bittasini yoki bir nechtasini tanlash yo'li bilan ishlovchi vizual dasturlar yaratish imkoniyatlarini beradi.

CheckBox komponentasi  ko'rinishida bo'ladi. Uning asosiy xossalaridan biri Text va Checkedlardir. Ushbu xossalari yordamida tanlanganligi aniqlanadi.

CheckedListBox komponentasi  ko'rinishida bo'ladi. Uning asosiy xossalaridan biri Itemsdir. Ushbu xossasi yordamida tanlanganligi aniqlanadi.

ComboBox komponentasi  ko'rinishida bo'ladi. Uning asosiy xossalaridan biri **Items** dir. Ushbu xossasi yordamida tanlanganligi aniqlanadi.

Ushbu komponentalarining tanlash parametrlari turlicha bo'lib, ularni kodi quyidagicha:

```
1) if(checkBox1->Checked && checkBox2->Checked){…}
2) void index(int indexChecked[]){ int i=0; IEnumerator^ myEnum1=
```



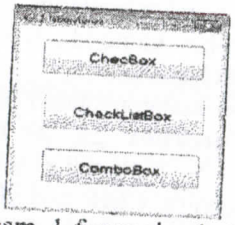
```
checkedListBox1->CheckedIndices->GetEnumerator();     while( myEnum1-
>MoveNext())    {
indexChecked[i] =  *safe_cast<Int32^>(myEnum1-> Current); i++; }
    }
index(indexChecked);  if ( indexChecked[0]==0 && indexChecked[1]==1)
{…}
    3) if(comboBox1->Text=="Tasodifiy sonlar bilan to'ldirish"){…}
```

Ushbu komponentalarning imkoniyatlarini to'liq ochib berish uchu amaliy vizual dastur tuzishni ko'raylik. Buni amalga oshirish uchu quyidagi ketma- ketliklarda bajariladi:

1) **1-formada** 3 ta button komponentalari joylashtiriladi va 1- butto ning text xossasi **CheckBox** ga, 2- button ning text xossas **CheckedListBox** ga, 3- buttonning text xossasi **ComboBoxg** o'zgartiriladi. 1- buttonning **Click** hodisasiga Form2^ open=gcnev Form2(); this->Hide(); open->Show();, 2- button ning **Clic** hodisasiga Form3^ open=gcnew Form3(); this->Hide(); open >Show();, 3- button ning **Click** hodisasiga Form31^ open=gcnev Form31(); this->Hide(); open->Show(); kodlari kiritiladi. Formala bir biri bilan bog'lanadi.
2) **2-formada** 1 ta **dataGridView,** 2 ta **TextBox,** 1 ta **button** va 6 t **checkBox** komponentalari joylashtiriladi. **checkBox1** ning Tex xossasi "Hisoblash" ga, **checkBox2** ning Text xossasi "Tasodifi sonlar bilan to'ldirish" ga, **checkBox3** ning Text xossas "DadaGridWiew dan kiritish" ga, **checkBox4** ning Text xossas "Sariq rang", **checkBox5** ning Text xossasi "Qizil rang" g **checkBox6** ning Text xossasi "Yashil rang" ga o'zgartiriladi va 2 jadvaldagi 3, 4 va 5- qadamlar bajariladi.
3) **3-formada** 1 ta **dataGridView,** 2 ta **TextBox,** 1 ta **button** va 1 **CheckedListBox** komponentalarini joylashtiring. **CheckedListBo** ning Items xossasiga "Hisoblash", "Tasodifiy sonlar bilan to'ldirish" "DadaGridWiew dan kiritish", "Sariq rang", "Qizil rang", "Yash rang" iboralar kiritiladi va 2-jadvaldagi 3, 4 va 5- qadaml bajariladi.
4) **4-formada** 1 ta **dataGridView,** 2 ta **TextBox,** 1 ta **button,** 4 **checkBox,** 1 ta **comboBox** komponentalarini joylashtirin **checkBox1** ning Text xossasi "Hisoblash" ga, **checkBox4** ning Te xossasi "Sariq rang", **checkBox5** ning Text xossasi "Qizil rang" g **checkBox6** ning Text xossasi "Yashil rang" ga o'zgartirilad **comboBox** ning **Items** xossasiga "Hisoblash", "Sariq rang", "Qiz

rang", "Yashil rang" iboralar kiritiladi va 2-jadvaldagi 3, 4 va 5-qadamlar bajariladi.

1-qadam: 1- formaga boshqa formalarni bog'lash uchun quyidagi kodlar kiritiladi: `#include "Form2.h"; #include "Form3.h"; #include "Form31.h";`



3.17-rasm. 1-formaning ko'rinishi

1- forma 2- formaga bog'lanishi uchun **CheckBox** tugmasining **Onclick** hodisasiga quyidagi kodlar kiritiladi:

```
4. Form2^ open=gcnew Form2();
5. this->Hide(); open->Show();
```

1- forma 3- formaga bog'lanishi uchun **CheckListBox** tugmasining **Onclick** hodisasiga quyidagi kodlar kiritiladi:

```
1. Form3^ open=gcnew Form3();
2. this->Hide(); open->Show();
```

1-forma 3- formaga bog'lanish uchun **ComboBox** tugmasining **Onclick** hodisasiga quyidagi kodlar kiritiladi:

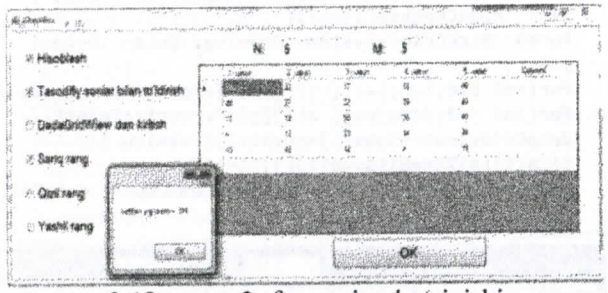
```
1. Form3^ open=gcnew Form3();
2. this->Hide(); open->Show();
```

2-qadam: 32 – variantdagi masalani yechish uchun 2-formada **button** ning **Onclick** hodisasiga quyidagi kodlar kiritiladi:

```
1. private: System::Void button1_Click(System::Object^
   sender,System::EventArgs^  e) {
2. int a[10][10]; s=0; int n,m;
3. if(textBox1->Text!="" && textBox2->Text!=""){
4. n=Convert::ToInt32(textBox1->Text);
5. m=Convert::ToInt32(textBox2->Text);}
6. if(checkBox1->Checked && checkBox2->Checked ){
7. if(checkBox4->Checked){Form2::BackColor=
8. System::Drawing::Color::Yellow; }
9. if(checkBox5->Checked){Form2::BackColor=
10.      System::Drawing::Color::Red; }
11.      if(checkBox6->Checked){Form2::BackColor=
12.      System::Drawing::Color::Green; }
```



```
13.      for(int i=0;i<n;i++){  if(g==0){dataGridView1->Rows->Add()
14.      for(int j=0;j<m;j++){a[i][j]=rand()%50+1;
15.  dataGridView1->Columns[j]->HeaderText= (j+1).ToString()+" -
   ustun";
16.      dataGridView1->Rows[i]->Cells[j]->Value =a[i][j].ToString
17.      if(a[i][j]%2==0){s+=a[i][j];}     }} g+=1;
18.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString());
19.      if(checkBox1->Checked && checkBox3->Checked ){s=0;
20.      for(int i=0;i<n;i++){ if(g==0){dataGridView1->Rows->Add()
21.      for(int j=0;j<m;j++){ a[i][j]=Convert::ToInt32(
22.      dataGridView1->Rows[i]->Cells[j]->Value );
23.      if(a[i][j]%2==0){s+=a[i][j];} }  }
24.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString());
```

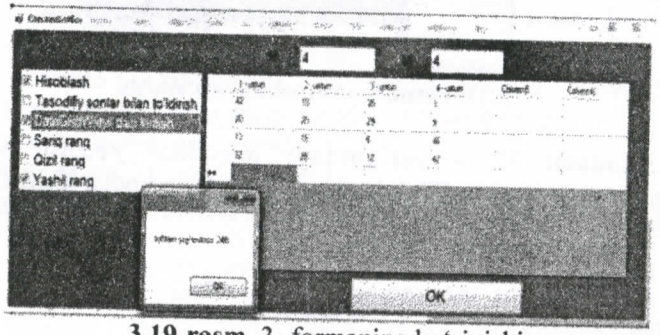


3.18-rasm. 2- formaning ko'rinishi

3-qadam: 32 – variantdagi masalani yechish uchun formada **button1** ning **Click** hodisasiga quyidagi kodlar kiritiladi:

```
1. private: System::Void button1_Click(System::Object^  sender,
2.  System::EventArgs^  e) { int g,s;
3. #pragma endregion
4. void index(int indexChecked[]){ int i=0;
5. IEnumerator^ myEnum1 = checkedListBox1->
6. CheckedIndices->GetEnumerator();
7. while( myEnum1->MoveNext()){
8. indexChecked[i] =  *safe_cast<Int32^>(myEnum1->Current); i++;
9. private: System::Void button1_Click(System::Object^  sender,
10.      System::EventArgs^  e) {
11.      int a[10][10];  int indexChecked[10]; s=0; int n,m;
12.      if(textBox1->Text!="" && textBox2->Text!=""){
13.      n=Convert::ToInt32(textBox1->Text);
14.      m=Convert::ToInt32(textBox2->Text);}  index(indexChecked)
15.      if (indexChecked[0]==0 && indexChecked[1]==1){
16.      if (indexChecked[2]==3{
17.      Form3::BackColor=System::Drawing::Color::Yellow; }
18.      if (indexChecked[2]==4){
19.      Form3::BackColor=System::Drawing::Color::Red; }
```

```
20.      if (indexChecked[2]==5){
21.      Form3::BackColor=System::Drawing::Color::Green; }
22.      for(int i=0;i<n;i++){ if(g==0){dataGridView1->Rows->Add();}
23.      for(int j=0;j<m;j++){ a[i][j]=rand()%50+1;
24.   dataGridView1->Columns[j]->HeaderText= (j+1).ToString()+" -
   ustun";
25.      dataGridView1->Rows[i]->Cells[j]->Value =a[i][j].ToString();
26.      if(a[i][j]%2==0){s+=a[i][j];} }} g+=1;
27.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString()); }
28.      if ( indexChecked[0]==0 && indexChecked[1]==2){
29.      if ( indexChecked[2]==3){
30.      Form3::BackColor=System::Drawing::Color::Yellow; }
31.      if ( indexChecked[2]==4){
32.      Form3::BackColor=System::Drawing::Color::Red; }
33.      if ( indexChecked[2]==5){
34.      Form3::BackColor=System::Drawing::Color::Green; }
35.      s=0;
36.      for(int i=0;i<n;i++){ if(g==0){dataGridView1->Rows->Add();}
37.      for(int j=0;j<m;j++){ a[i][j]=Convert::ToInt32(
38.      dataGridView1->Rows[i]->Cells[j]->Value );
39.      if(a[i][j]%2==0){s+=a[i][j];} }}
40.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString());}}
```



3.19-rasm. 3- formaning ko‘rinishi

4-qadam: 32 – variantdagi masalani yechish uchun 4-formada quyidagi kodlar kiritiladi:

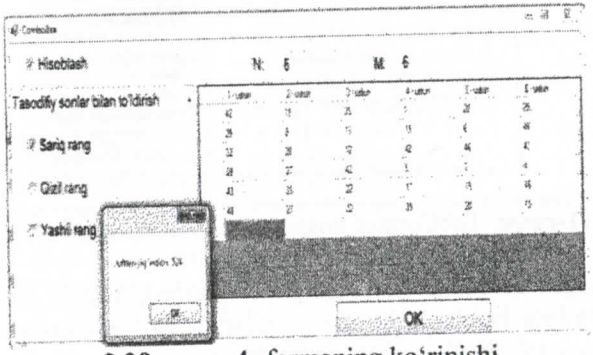
```
1. int g,s;
2. #pragma endregion
3. private: System::Void Form31_FormClosed(System::Object^  sender,
4. System::Windows::Forms::FormClosedEventArgs^  e) {
5. Application::Restart(); }
```

button1 ning **Click** hodisasiga quyidagi kodlar teriladi:

```
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^  e) {
```



```
3. int a[10][10]; int indexChecked[10]; s=0; int n,m;
4. if(textBox1->Text!="" && textBox2->Text!=""){
5. n=Convert::ToInt32(textBox1->Text);
6. m=Convert::ToInt32(textBox2->Text);}
7. if(comboBox1->Text=="Tasodifiy sonlar bilan to'ldirish"){
8. if(checkBox1->Checked){ if(checkBox4->Checked){
9. Form31::BackColor=System::Drawing::Color::Yellow; }
10.      if(checkBox5->Checked){
11.      Form31::BackColor=System::Drawing::Color::Red; }
12.      if(checkBox6->Checked){
13.      Form31::BackColor=System::Drawing::Color::Green; }
14.      for(int i=0;i<n;i++){
15.      if(g==0){ dataGridView1->Rows->Add();}
16.      for(int j=0;j<m;j++){ a[i][j]=rand()%50+1;
17.      dataGridView1->Columns[j]->HeaderText=
18.      (j+1).ToString()+" - ustun";
19.      dataGridView1->Rows[i]->Cells[j]->Value =a[i][j].ToString
20.      if(a[i][j]%2==0){ s+=a[i][j];}}} g+=1;
21.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString());
22.      if(comboBox1->Text=="DataGridWiew dan kiritish"){
23.      if(checkBox1->Checked){ if(checkBox4->Checked){
24.      Form31::BackColor=System::Drawing::Color::Yellow; }
25.      if(checkBox5->Checked){
26.      Form31::BackColor=System::Drawing::Color::Red; }
27.      if(checkBox6->Checked){
28.      Form31::BackColor=System::Drawing::Color::Green;} s=0;
29.      for(int i=0;i<n;i++){
30.      if(g==0){dataGridView1->Rows->Add();}
31.      for(int j=0;j<m;j++){
32.      a[i][j]=Convert::ToInt32( dataGridView1->Rows[i]->Cells[j
   >Value );
33.      if(a[i][j]%2==0){s+=a[i][j];} }}
34.      MessageBox::Show("Juftlari yig'indisi= " + s.ToString());
35.      }
```



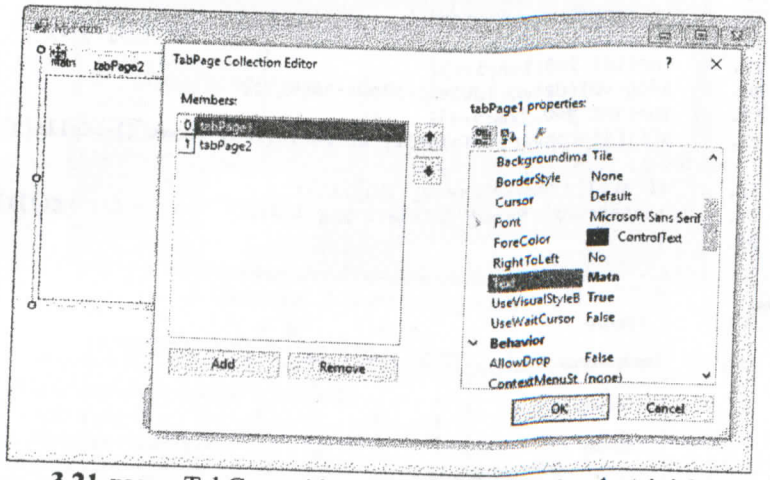
3.20-rasm. 4- formaning ko‘rinishi

3.7. TabControl va RadioButton komponentalari

Ushbu komponentalar boshqarishni osonlashtirish va ekranning ichki sohasidan optimal foydalanish uchun ishlatiladi. Ko'p miqdordagi boshqariluvchi ma'lumotlarni ko'rsatish kerak bo'lganda **TabControl** komponentasidan foydalangan ma'qul.

Uchta **TabControl** komponentalarida **RadioButton** tanlash komponentalaridan foydalangan holda matn uchun berilgan ikki variantdan birini tanlaydigan, matn uchun rang va o'lcham beradigan dastur tuzish masalasi qo'yilsin. Masalaning dastur ko'rinishi 3.21-rasmda keltirilgan.

Qo'yilgan masalani yechish uchun Visual Studio 2012 dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->New->Project...** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga "**Qo'yilmalar**" nomi beriladi va **OK** tugmasi bosiladi. ToolBox panelidan **TabControl** boshqaruv elementini olib kelinadi. **TabControl** boshqaruv elementining ko'rinishi va uning bo'limlarni sozlash quyidagicha (**3.21-rasm**):



3.21-rasm. TabControl boshqaruv elementining ko'rinishi

Kelishuv bo'yicha ikkita qo'yilma mavjud, masala shartiga ko'ra uchta qo'yilma kerak. Uchinchi qo'yilmani forma konstruktori orqali yoki dastur kodi orqali qo'shish mumkin.

3.22- rasm. Dastur natijasi

Dastur kodi. Qo'yilmalar bilan ishlash.

```
#pragma endregion
// Formani yuklash xodisasini hosil qilish
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
this->Text="Formani formalarga boglash dasturi";
button1->Text = "2- formaga utish";
label1->Text = "1-forma";
auto TabPage3 = gcnew System::Windows::Forms::TabPage();
TabPage3->UseVisualStyleBackColor = true;
     this->tabControl1->Controls->Add(TabPage3);
     TabPage3->Controls->Add(this->radioButton5);
     TabPage3->Controls->Add(this->radioButton6);
     this->radioButton5->Location =
System::Drawing::Point(20,15);
     this->radioButton6->Location =
System::Drawing::Point(20,45);
     tabControl1->TabPages[0]->Text = "Matn";
     tabControl1->TabPages[1]->Text = "Rangi";
     tabControl1->TabPages[2]->Text = "O'lchami";
     radioButton1->Text    = "Visual Studio 2012";
     radioButton2->Text    = "Embarcadero XE3";
     radioButton3->Text    = "Yashil rang";
     radioButton4->Text    = "Qizil rang";
     radioButton5->Text    = "11 o'cham";
     radioButton6->Text    = "25 o'lcham";
     label1->Text = radioButton1->Text;
     radioButton5->Font = gcnew System::Drawing::Font(
     radioButton5->Font->Name,16);
     radioButton6->Font = gcnew System::Drawing::Font(
     radioButton6->Font->Name,16); }
     private: System::Void
radioButton1_CheckedChanged(System::Object^
     sender, System::EventArgs^  e) { label1->Text =
     radioButton1->Text;}
```

```
32.     private: System::Void
   radioButton2_CheckedChanged(System::Object^
33.     sender, System::EventArgs^  e) {  label1->Text =
34.     radioButton2->Text;}
35.     private: System::Void
   radioButton4_CheckedChanged(System::Object^
36.     sender, System::EventArgs^  e) {  label1->ForeColor=
   Color::Red;}
37.     private: System::Void
   radioButton3_CheckedChanged(System::Object^
38. sender, System::EventArgs^ e) {  label1->ForeColor =
   Color::Green; }
39. private: System::Void
   radioButton5_CheckedChanged(System::Object^
40. sender, System::EventArgs^  e) { label1->Font = gcnew
41. System::Drawing::Font(label1->Font->Name,11);   }
42. private: System::Void
   radioButton6_CheckedChanged(System::Object^
43. sender, System::EventArgs^  e) {
44. label1->Font = gcnew System::Drawing::Font(label1->Font-
   >Name,25);  }};
45.}
```

3.8. Berilganlarni lug‘at (Dictionary) yordamida strukturali saqlash

Dictionary lug‘ati kalitlar va ularga mos qiymatlar to‘plamlarini o‘z ichiga oladi. Ya’ni, lug‘atga qo‘shiluvchi har bir element qiymat (**value**) va unga bog‘langan kalitdan (**Key**) tashkil topadi. Kalit yordamida qiymatni olib chiqishi juda ham tez amalga oshadi. Chunki **Dictionary <Key, Value>** sinfi xesh jadval kabi yaratiladi. Lug‘atdagi har bir kalit takrorlanmas bo‘lishi shart. **Dictionary** lug‘atiga yangi element qo‘shilayotganda komperator (solishtiruvchi) yangi kalitni lug‘atda yo‘qligini tekshiradi. Kalit bo‘sh (**null**) bo‘lishi mumkin emas, ammo qiymat turi ko‘rsatgichli bo‘lsa, qiymatga null berish mumkin. Lug‘at **Dictionary** algoritmning samaradorligini bir necha barobar oshirishi, dastur kodini osonlikcha tushunishiga va tez ishlashiga olib keladi.

Misol tariqasida, oy kunlarini tahlil qiluvchi vizual dastur tuzishni ko‘rish mumkin. Bunda **Dictionary** elementining kaliti sifatida oylar nomini va qiymat sifatida shu oylarning necha kunligi olingan.

Bu vazifani bajarish uchun Visual Studio 2012 dasturi ishga tushiriladi, asosiy oyna menyu bo‘limlaridan **File->New->Project...**

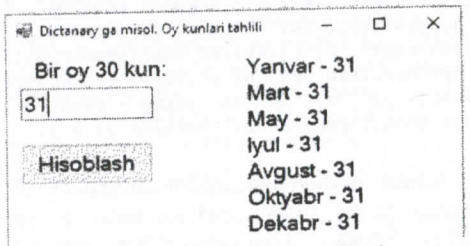
buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga “**Dictionary**” nomi beriladi va **OK** tugmasi bosiladi.

ToolBox panelidan 2 ta Label, Button va TextBox komponentalari tashlanadi.

Button tugmachasi ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. #pragma once
2. using namespace System::Collections::Generic;
3. #pragma endregion
4. private: System::Void button1_Click(System::Object^  sender,
   System::EventArgs^  e) {
5. textBox1->Focus(); label2->Text = String::Empty;
6. auto oylar=gcnew Dictionary<String^,int>(); oylar["Yanvar"]= 31;
7. oylar["Febral"]= 28; oylar["Mart"]= 31; oylar["Aprel"]= 30;
8. oylar["May"]= 31; oylar["Iyun"]= 30; oylar["Iyul"]= 31;
9. oylar["Avgust"]= 31; oylar["Sentyabr"]= 30; oylar["Oktyabr"]= 31;
10. oylar["Noyabr"]= 30; oylar["Dekabr"]= 31;
11.     for each (KeyValuePair<String^, int> oylar in oylar){
12.     if(oylar.Value ==Convert::ToInt32(textBox1->Text)){
13.     label2->Text = label2->Text + String::Format(
14.     "{0} - {1} \n", oylar.Key, oylar.Value); } }}
15.     private: System::Void Form1_Load(System::Object^  sender,
16.     System::EventArgs^  e) {textBox1->Focus();
17.     this->Text = "Dictanary ga misol. Oy kunlari tahlili";
18.     label1->Text = "Bir oy 30 kun:g`n";
19.     label2->Text = String::Empty;}
20.     };
21.     }
```

Dastur kodining boshida **Dictionary** sinfining ob`yekti yaratildi, ya’ni oy nomi uchun String turidagi maydon va oydagi kunlar soni uchun int turidagi maydonli lug‘at. Lug‘atda 30 kunlik yoki 31 kunlik oylarni izlashda for each sikl operatori ishlatilgan.



3.23- rasm. Dastur natijasi.

3.9. Bir prosedura orqali bir nechta hodisalarga ishlov berish

Visual Studio (C++, C#, F# va Visual Basic) zamonaviy tillarda elementlar massivda guruhlanishi mumkin emas. Bir nechta xodisalarni bir prosedurada tashkillashtirish yo'li, ushbu hodisalarga ishlov berishni bitta proseduraga yozishdan iborat. Til sintaksisi hodisaga ishlov beruvchi prosedurani ixtiyoriy ravishda nomlash imkonini beradi.

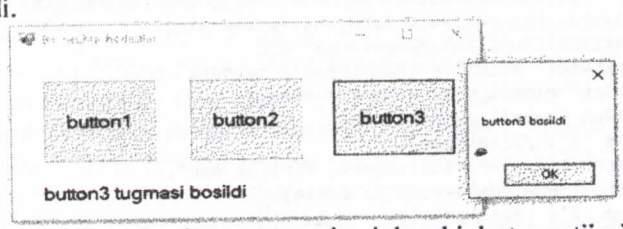
Buni qanday amalga oshirishni namuna misol orqali ko'ramiz. Formaga ikkita tugma joylashtirib, ulardan ixtiyoriy biri bosilganda bitta prosedura ishlaydi. Bunda proseduraning sender parametridan foydalanib, sichqoncha yordamida aynan qaysi tugmaga bosilganligi aniqlanadi.

Visual Studio 2012 dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->New->Project...** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga "**Bir_nechta_hodisalar**" nomi beriladi va **OK** tugmasi bosiladi. Elementlar panelidan ikkita tugma va matnli nishonni formaga joylashtiriladi. **Form** ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. #pragma endregion
2. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {
3. label1->Text = nullptr; button1->Click += gcnew
   EventHandler(this,
4. &Form1::Bosish); button2->Click += gcnew EventHandler(this,
5. &Form1::Bosish); button3->Click += gcnew EventHandler(this,
6. &Form1::Bosish); }
7. private: System::Void Bosish(System::Object^ sender,
   System::EventArgs^ e){
8. Button^ tugma = (Button^) sender;
9. label1->Text = tugma->Text + " tugmasi bosildi"; }
10. private: System::Void button3_Click(System::Object^  sender,
11. System::EventArgs^  e) { ikkichi_usul(sender,e); }
12. private: System::Void ikkichi_usul(System::Object^  sender,
13. System::EventArgs^  e){ Button^ tugma = (Button^) sender;
14. MessageBox::Show(tugma->Text+" bosildi"); } }; }
```

Formani yuklash hodisasiga ishlov berishda, hodisaga yozilish amalga oshiriladi, ya'ni hodisalar nomini **EventHandler** usuli yordamida hodisaga ishlov beruvchi Click prosedura nomi bilan

bog'landi. **button3_Click()** proseduraside **ikkichi_usul()** prosedurasi chaqirildi.



3.24- rasm. Bosilgan tugmani aniqlovchi dastur natijasi.

3.10. Turli tipdagi fayllarning manbalariga LinkLabel komponentasi yordamida murojaatlar

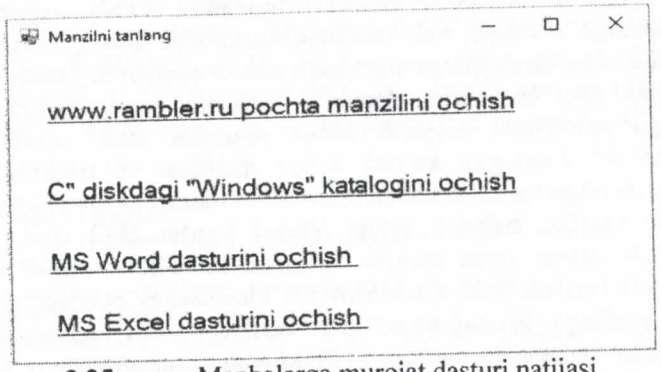
LinkLabel boshqaruv elementi formadan HTML xujjatlaridagi murojaatlarga o'xshash veb sahifaga, dasturdagi qandaydir fayllarga, lokal disklarga, kataloglarga murojaat qilish imkoniyatini yaratadi.

LinkLabel boshqaruv elementi yordamida www.mail.ru pochta serveriga tashriflarni aniqlash uchun murojaat bilan ta'minlaymiz, C:\Windows\ katalogini ko'rish uchun murojaat va Bloknot matnli redaktorini ishga tushirish uchun murojaatlar dasturi ko'rib chiqiladi.

Bu vazifani bajarish uchun **Visual Studio 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->New->Project...** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga "**Murojaatlar**" nomi beriladi va **OK** tugmasi bosiladi. Elementlar panelidan uchta LinkLabel komponentasi formaga joylashtiriladi. **Form1** ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. #pragma endregion
2. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) { this->Text = "Manzilni tanlang";
3. linkLabel1->Text = "www.rambler.ru pochta manzilini ochish";
4. linkLabel2->Text = "\C\" diskdagi \"Windows\" katalogini ochish";
5. linkLabel3->Text = "MS Word dasturini ochish ";
6. linkLabel4->Text = "MS Excel dasturini ochish ";
7. // Barcha hodilar 1 ta prosedura yordamida qayta ishlanadi
8. linkLabel1->LinkClicked += gcnew
   LinkLabelLinkClickedEventHandler(this,
9. &Form1::Murojaat); linkLabel2->LinkClicked += gcnew
10. LinkLabelLinkClickedEventHandler(this, &Form1::Murojaat);
11. linkLabel3->LinkClicked += gcnew
12. LinkLabelLinkClickedEventHandler(this, &Form1::Murojaat);
```

```
13. linkLabel4->LinkClicked += gcnew
14. LinkLabelLinkClickedEventHandler(this, &Form1::Murojaat); }
15. private: System::Void Murojaat(System::Object^  sender,
16. LinkLabelLinkClickedEventArgs^ e){
17. LinkLabel^ manzil = (LinkLabel^) sender;
18. String^ manzil_nomi = manzil->Name;
19. switch (manzil_nomi[9]) {
20. case '1': //linkLabel1
21. Diagnostics::Process::Start("firefox.exe",
22. "http://www.rambler.ru"); break;
23. case '2': //linkLabel2
24. Diagnostics::Process::Start("C:\\Windows\\"); break;
25. case '3': //linkLabel2
26. Diagnostics::Process::Start("winword.exe"); break;
27. case '4': //linkLabel2
28. Diagnostics::Process::Start("excel.exe"); break;
        break;  }   }  }; }
```



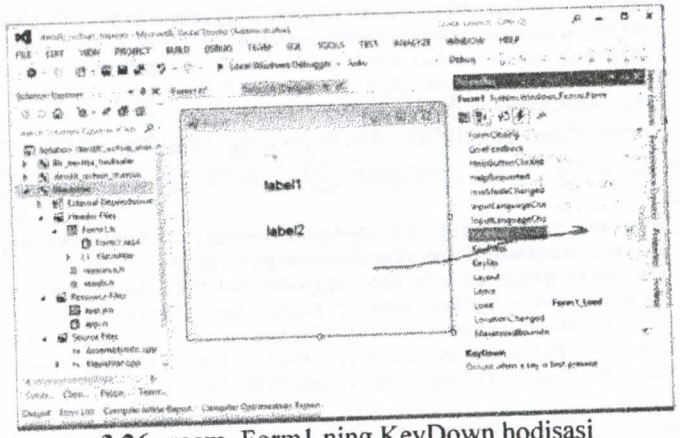
3.25-rasm. Manbalarga murojat dasturi natijasi.

3.11. Klaviatura hodisalarini qayta ishlash

Klaviatura hodisasi – klavishlar bosilgan yoki klavish bosib qo'yib yuborilgan vaziyatda sodir bo'ladi. Bundan farqli ravishda **KeyPress** hodisasi klavish bosilgan vaziyatda generatsiya bo'ladi. Klaviaturani bosib ushlab turgan holatda u uzluksiz ravishda bir necha takrorlanishlarni generatsiya qiladi. Masalan, **Alt, Shift** yoki **Ctrl** klavishlarini bosilganligini aniqlash uchun, KeyDown hodisasini yoki KeyUp hodisasini qayta yuklash kerak. **KeyDown** hodisasi klavish birinchi bosilgandagi vaziyatni generatsiya qiladi. KeyUp hodisasi esa klavish bosib qo'yib yuborilgandagi vaziyatni generatsiya qiladi.

Quyidagi misol klavishlar va klavishlar kombinasiyasi bosilganda foydalanuvchiga bu haqida xabar beruvchi dasturdir. Buni amalga oshirish uchun **Visual Studio 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->New->Project...** buyruqlari beriladi yok **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihag **"Klavishlar"** nomi beriladi va **OK** tugmasi bosiladi. Elementlar panel **ToolBox** dan ikkita Label komponentasi joylashtiriladi va **Form1** ustid sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodla teriladi:

```
1. private: System::Void Form1_Load(System::Object^  sender,
2. System::EventArgs^  e) {
3. this->Text = "Qanday klavish bosilganligini aniqlash";
4. label1->Text = String::Empty;
5. label2->Text = String::Empty;
6. label3->Text = String::Empty; label4->Text = String::Empty;
7. Form1::BackColor = Color::Yellow;}
```



3.26- rasm. Form1 ning KeyDown hodisasi

Proporties panelidan **Form1** ning **KeyDown** hodisasi(**3.26-ras** ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kod teriladi:

```
1. private: System::Void Form1_KeyDown(System::Object^  sender,
   System::Windows::Forms::KeyEventArgs^  e) {
2. // klavish bosilganda sodir bo'ladi
3. label1->ForeColor = Color::Green;
4. if(e->Alt==true){label1->Text=String::Empty; label1->Text +=
5. "Alt klavishasi bosildig`n";
6. label1->Text += String::Format("      Klavyatura kodilari:\n "
```

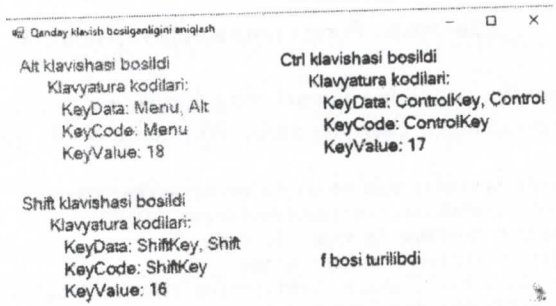
```
7. "        KeyData: {0} \n     KeyCode: {1}\n        KeyValue: {2}",
8. e->KeyData, e->KeyCode, e->KeyValue);}
9. label2->ForeColor = Color::Red; if(e->Shift==true){
10. label2->Text=String::Empty; label2->Text +="Shift klavish
    bosildig`n";
11. label2->Text += String::Format("      Klavyatura kodilari:\n "+
12. "       KeyData: {0} \n          KeyCode: {1}\n          KeyValue:
13. {2}", e->KeyData, e->KeyCode, e->KeyValue);}
14.     label3->ForeColor = Color::Black  ;
15.     if(e->Control==true){label3->Text=String::Empty;
16.     label3->Text+="Ctrl klavish bosildig`n";
17.     label3->Text += String::Format("   Klavyatura kodilari:\n "+
18.     "     KeyData: {0} \n        KeyCode: {1}\n        KeyValue:
19.     {2}", e->KeyData, e->KeyCode, e->KeyValue);} }
```

Proporties panelidan **Form1** ning **KeyPress** hodisasi ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. bool b;
2. private: System::Void Form1_KeyPress(System::Object^  sender,
3. System::Windows::Forms::KeyPressEventArgs^  e) {
4. //klavish bosib ushlab turilganda sodir bo'ladi
5. label4->Text = (e->KeyChar+" bosi turilibdi");
6. if(e->KeyChar=='t')b=true; }
```

Proporties panelidan **Form1** ning **KeyUp** hodisasi ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar teriladi:

```
1. private: System::Void Form1_KeyUp(System::Object^  sender,
2. System::Windows::Forms::KeyEventArgs^  e) {
3. //klavish bosib qo'yib yuborilganda sodir bo'ladi
4. label4->Text = String::Empty; Form1::Focused;
5. if(b){ label1->Text = String::Empty;
6. label2->Text = String::Empty; label3->Text = String::Empty;
7. label4->Text = String::Empty; b=false; }   }
```



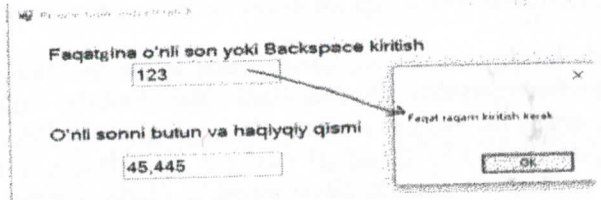
3.27- rasm. Klavyatura klavishlarini aniqlash dasturi

3.12. Matnli maydonga kiruvchi ma'lumotlarni boshqarish

Odatda kiritilgan sonli ma'lumotlarning tashhisini chiqarish uchun **TryParse** funksiyasidan foydalaniladi. Bu funksiya rost qiymat qaytaradi, agar kiritilayotgan ma'lumotlar sonli qiymat bo'lsa, aks holda false qiymat qaytaradi. Quyidagi dasturda foydalanuvchi tomonidan kiritilgan ma'lumotni nazorat qilish mumkin. Dastur kiritilayotgan har bir belgini tekshiradi. Bundan tashqari, foydalanuvchi matnli maydonga sonning haqiqiy qismini nuqta yoki verguldan keyin kiritishi mumkin.

Ushbu vazifani bajaruvchi dasturni tuzish uchun **Visual Studio 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo'limlaridan **File->New->Project...** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga "**Klavishlar**" nomi beriladi va **OK** tugmasi bosiladi. Elementlar paneli **ToolBox** dan 2 ta **TextBox** komponentasi joylashtiriladi. TextBox1 va TextBox2 komponentalarini KeyPress hodisalari qayta ishlanishi uchun quyidagi kodlar teriladi:

```
1. #pragma endregion
2. String^ NuqtaVergul;
3. private: System::Void Form1_Load(System::Object^  sender,
4. System::EventArgs^  e) {
5. this->Text = "Belgilar turini nazorat qilish";
6. textBox1->Clear(); textBox2->Clear();
7. NuqtaVergul = Globalization::NumberFormatInfo::CurrentInfo-
8. >NumberDecimalSeparator; }
9. private: System::Void textBox1_KeyPress(System::Object^  sender,
10.     System::Windows::Forms::KeyPressEventArgs^  e) {
11. //Faqatgina o'nli son yoki Backspace kiritish masalasini hal qilish
12. if(Char::IsDigit(e->KeyChar)==true) return;
13. if(e->KeyChar == (char)Keys::Back) return;
14. else{MessageBox::Show("Faqat raqam kiritish kerak");}
15. // boshqa belgilar kiritilishini taqiqlash
16. e->Handled = true; } bool topildi;
17. private: System::Void textBox2_KeyPress(System::Object^  sender,
18. System::Windows::Forms::KeyPressEventArgs^  e) {
19. // Faqatgina o'nli son yoki Backspace kiritish masalasini hal qilish
20. if(Char::IsDigit(e->KeyChar)==true) return;
21. if(e->KeyChar == (char)Keys::Back) return;
22. else{MessageBox::Show("Faqat raqam kiritish kerak");}
23. if(textBox2->Text->IndexOf(NuqtaVergul) != -1)
24. topildi = true; if(topildi == true){ e->Handled = true; return;
    }
25. if(e->KeyChar.ToString() == NuqtaVergul) return;
26. // boshqa belgilar kiritilishini taqiqlash
27. e->Handled = true; } }; }
```



3.28- rasm. Berilganlarning turlarini aniqlash dasturi

3.13. try ...catch istisnosini qayta ishlash orqali matnli faylni o‘qish va matnli faylga yozish

Berilganlarni xotiraga (diskga) matn formatida (ikkilikda emas) saqlash masalasi juda keng tarqalgan. Matnli va ikkilik formatga ajratish shartdir, chunki matnli fayllar ham, matnsiz fayllar ham aslida ikkilik ko‘rinishga ega. Masalan, matn turida bo‘lmagan faylni bloknotda ochganda “o‘qib bo‘maydigan aralashma” ga duch kelinadi. Odatda ma’lumotlarni diskka matnli formatda saqlashadi, sababi mazkur ma’lumotlarni o‘qish, bloknot, va **TextEdit** kabi istalgan matn muharririda tahrirlash mumkin bo‘lishi kerak.

Matnli faylni o‘qish jarayoni bir necha satrlarda amalga oshadi.

```
1. // Fayldan o'qish uchun StreamReader nushasini yaratish
2. IO::StreamReader^ Readder = gcnew
   IO::StreamReader("D:\\matn.txt");
3. // Matnli fayl tarkibini satrga o'tkazish
4. String^ Matn =Readder->ReadToEnd();  Readder->Close();
```

Biroq ayrim farqlar ham bor. Forma tarkibida matn maydoni va 2 ta buyruq tugmachasi bo‘lgan dastur yoziladi. Birinchi tugmacha bosilganda matnli faylning Unicode kodida matn maydoniga o‘qilish yuz beradi. Ikkinchi tugmacha bosilganda foydalanuvchi tomonidan matn maydonida tahrir qilingan matn diskdagi faylga saqlanadi. Matn maydoni uchun **Properties** oynasida **Multiline** xossasi uchun **true** qiymat ko‘rsatiladi. Bu orqali kiritilayotgan matn bir satrga emas, balki sichqoncha orqali belgilab ko‘rsatilgan maydon o‘lchamida joylashadi. Dastur matni quyida ko‘rsatilgan:

```
1. #pragma endregion
2. String^ fay_nomi; bool fayl(){
3. if(textBox1->Text =="" || textBox1->Text =="Fayl nomi"){
```

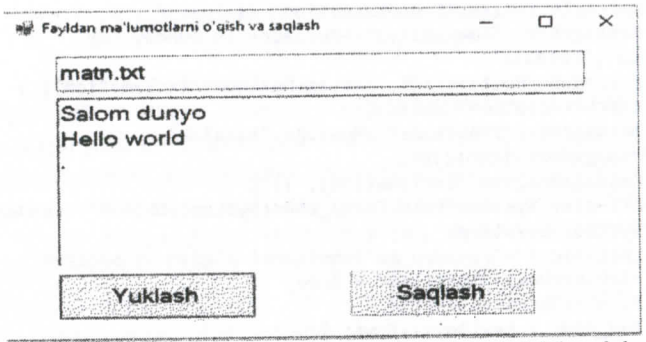


```
4. MessageBox::Show("Fayl nomini kiriting"); textBox1-
   >Focus();return 0; }
5. else{ fay_nomi="C:\\"+ textBox1->Text; return 1; } }
6. private: System::Void button1_Click(System::Object^  sender,
7. System::EventArgs^  e) {// Yozish tugamasi bosilganda
8. if(fayl()){
9. // Ruscha harflarni ham o'qish uchun kodirovka obyekti e'lon qilinadi.
10. System::Text::Encoding^ kodirovka =
11. System::Text::Encoding::GetEncoding(1251);
12. try{     auto uqish = gcnew IO::StreamReader(fay_nomi);
13. richTextBox1->Text = uqish->ReadToEnd();
14. uqish->Close(); // faylni yopish
15. IO::File::ReadAllLines(fay_nomi); }
16. catch(IO::FileNotFoundException^ vaziyat){
17. MessageBox::Show(vaziyat->Message+"\n bunday fayl
    yuq","Xatolik",
18. MessageBoxButtons::OK, MessageBoxIcon::Exclamation); }
19. catch(Exception^ xolat){
20. MessageBox::Show(xolat->Message,"Xatolik",
    MessageBoxButtons::OK,
21. MessageBoxIcon::Exclamation); }} }
22. private: System::Void Form1_Load(System::Object^  sender,
23. System::EventArgs^  e) {
24. this->Text ="Fayldan ma’lumotlarni o'qish va saqlash";
25. richTextBox1->Multiline = true;
26. richTextBox1->Clear();
27. fay_nomi = textBox1->Text; }
28. private: System::Void button2_Click(System::Object^  sender,
29. System::EventArgs^  e) {
30. // Saqlash tugamasi bosilganda
31. if(fayl()){
32. try{
33. auto kodirovka = System::Text::Encoding::GetEncoding(1251);
34. auto yozishv = gcnew IO::StreamWriter(fay_nomi, false);
35. yozishv->Write(richTextBox1->Text);
36. yozishv->Close(); MessageBox::Show("Faylga saqlandi"); }
37. catch(Exception^ xolat){
38. MessageBox::Show(xolat->Message,
    "Xatolik",MessageBoxButtons::OK,
39. MessageBoxIcon::Exclamation); } } }
40. private: System::Void textBox1_Click(System::Object^  sender,
41. System::EventArgs^  e) { textBox1->Text = String::Empty; } }; }
```

Berilgan dastur kodida qo‘llanilgan try bloki haqida to‘xtalib o‘ti lozim. Try qo‘llanilishining mazmuni quyidagicha: qandaydir masala misol uchun matnni o‘qishga *harakat qilish* (**try**). Agarda masa noto‘g‘ri ishlangan, masalani fayli topilmagan bo‘lsa, u hol boshqaruvni qo‘lga olish (**catch**) va yuz bergan xolatni qayta ishla

(**Exception**) mumkin. Istisno holatlarini qayta ishlash foydalanuvchini chalkashliklardan ogohlantiradi.

"**Ochish**" tugmasini bosish jarayonini qayta ishlashda **fayl()** funksiyasi tashkil qilingan. Odatda bunday holda fayl tanlovi uchun **OpenFileDialog** komponentasidan foydalaniladi. Keyingi o'rinda fayldan o'qish uchun **Reader** ob`ekti hosil qilindi. Faylni **ReadToEnd()** metodi orqali **RichTextBox1** komponentasiga o'tkazish va **Close()** metodi bilan yopadi. Quyidagi rasmda faylning ish jarayonidan lavha berilgan.



3.29- rasm. Unicod kodida matnli faylni o'qish/yozish

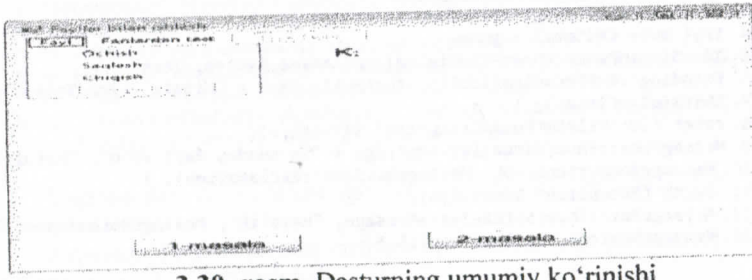
Berilgan matnli faylni nomini va kengaytmasini **textBox1** komponentasiga yozmaguncha **fayl()** funksiyasi **false** qiymat qaytaraveradi. Fayl manzili avtomatik tarzda C:\\ diskga to'g'irlangan. Fayl nomi va kengaytmasi (**txt**) to'g'ri yozilsa **fayl()** funksiyasi **true** qiymat qaytaradi va dasturning keyin qadamlariga o'tiladi.

3.14. OpenFileDialog va SaveFileDialog komponentalaridan foydalanib, fayllarni ochish va saqlash

Bizga ma'lumki, matnli fayllarga ma'lumotlarni yozuvchi va o'quvchi maxsus funksiyalar mavjud. Shunday bo'lsa ham **Visual C++** muharririning ham maxsus shunday funksiyalari mavjud. Ushbu funksiyalarni vizual komponentalar bilan ishlatib, kichik matn muharririni yaratish mumkin. Ushbu mavzuda fayllarga ishlov berishning bir necha usullari bilan tanishib chiqiladi. **TextBox, MenuStrip, OpenFileDialog** va **SaveFileDialog** komponentalaridan

foydalanib, matnli va ***.rtf** kengaytmali fayllarga qayta ishlov beruvc vizual dastur yaratishni ko'rib chiqaylik. Ushbu vazifani bajaruvc dasturni tuzish uchun

1-qadam. Visual Studio 2012 dasturi ishga tushiriladi, aso oyna menyu bo'limlaridan **File->New->Project...** buyruqlari berila yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyiha "**Fayllar**" nomi beriladi va **OK** tugmasi bosiladi. Elementlar pan **ToolBox** dan formaga ma'lumot kiritish va chiqarish uchun 2 **TextBox1, TextBox2, K** sonini kiritish uchun **TextBox3,** 1-masal yechish uchun **Button1**, 2- masala uchun esa **Button2,** ma'lumotla ixtiyoriy nom bilan saqlash uchun **openFileDialog1, saveFileDialog1 menuStrip1** komponentalari joylashtiriladi. Dastur ko'rini quyidagicha:



3.30- rasm. Dasturning umumiy ko'rinishi

2-qadam. Button1 ning **click** hodisasiga quyidagi kodlar yozila

```
1. private: System::Void button1_Click(System::Object^  sender,
   System::EventArgs^  e) {
2. textBox2->Text="";
3. char satr[200]=""; int a[100], k=1,l=0; a[l]=0;
4. for(int i=0; i<textBox1->Text->Length; i++)
5. { satr[i]=textBox1->Text[i]; }
6. strcat(satr," "); int m=0,min=0, max=0;
7. bool b=true;
8. for(int i=0; i<strlen(satr); i++) {
9. if(satr[i]==' '){
10. a[l]=k-1; if(b){max=a[l]; b=false;}
11. if(a[l]<=max){max=a[l]; m=i-k; min=k;} l++; k=0;} k++; }
12. char natija[100]="";
13. strncpy(natija,satr+m,min);
14. String^ s=gcnew String(natija);
15. textBox2->Text=s;}
```

3-qadam. Button2 ning **click** hodisasiga quyidagi kodlar yoziladi:

```
private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) { int K;
K=Convert::ToInt64(textBox3->Text); textBox2->Text="";
char satr[200]=""; int a[100], k=1,p=1,l=0; a[l]=0;
for(int i=0; i<textBox1->Text->Length; i++)
{ satr[i]=textBox1->Text[i]; }
char natija[100]="";
strcat(satr," "); int m=0,min=0, max=0;
for(int i=0; i<strlen(satr); i++) {
      if(satr[i]==' '){  if(k-1==K ){
strncat(natija,satr+(i-k),k); strcat(natija," ");  } k=0; } k++; }
String^ s=gcnew String(natija);  textBox2->Text=s;}
```

4-qadam. 3.30- rasmdagi **File** menyusidagi **Ochish** bo'limining **onclick** hodisasiga quyidagi kodlar yoziladi:

```
private:  System::Void  ochishToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
openFileDialog1->ShowDialog();
if (openFileDialog1->FileName == nullptr) return;
try{ auto CHitatel = gcnew
IO::StreamReader(openFileDialog1->FileName,System::Text::
Encoding::GetEncoding(1251)); textBox1->Text = CHitatel->ReadToEnd();
CHitatel->Close(); }
catch (IO::FileNotFoundException^ Situasiya){
MessageBox::Show(Situasiya->Message + "\n bundey fayl yo`q", "Hatolik",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation); }
catch (Exception^ Situasiya){
MessageBox::Show(Situasiya->Message, "Hatolik", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);} }
```

5-qadam. 3.30- rasmdagi **File** menyusidagi **Saqlash** bo'limining **onclick** hodisasiga quyidagi kodlar yoziladi:

```
private: System::Void saqlashToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) {
saveFileDialog1->FileName = openFileDialog1->FileName;
if (saveFileDialog1->ShowDialog() ==Windows::Forms:: DialogResult::OK)
Zapis(); }
```

bu yerda **Zapis()** funksiyasi chaqirilgan. **Zapis()** funksiyasi quyidagicha:

```
void Zapis(){
try{
auto Pisatel = gcnewIO::StreamWriter(saveFileDialog1->FileName, false,
System::Text::Encoding::GetEncoding(1251));
Pisatel->Write(textBox1->Text);
Pisatel->Close(); textBox1->Modified = false; }
catch (Exception^ Situasiya){
MessageBox::Show(Situasiya->Message, "Hatolik", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);}
```

"Chiqish" tugmasi kodi quyidacha:

```
private: System::Void chiqishToolStripMenuItem_Click(System::Object^
```



```
sender, System::EventArgs^  e) {  this->Close();  }
```

6-qadam. Formaning **Load** hodisasining kodlari quyidagicha:

```
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^  e) {  textBox1->Multiline = true;
textBox1->Clear(); textBox1->Size = Drawing::Size(230, 299);
this->Text = "Oddiy matn muharriri";
openFileDialog1->FileName = "C:\\Text2.txt";
openFileDialog1->Filter = "Matnli fayllar (*.txt)|*.txt|All files
(*.*)|*.*";
saveFileDialog1->Filter = " Matnli fayllar (*.txt)|*.txt|All files
(*.*)|*.*";}
```

7-qadam. Formaning **FormClosing** xodisasi kodlari quyidagicha:

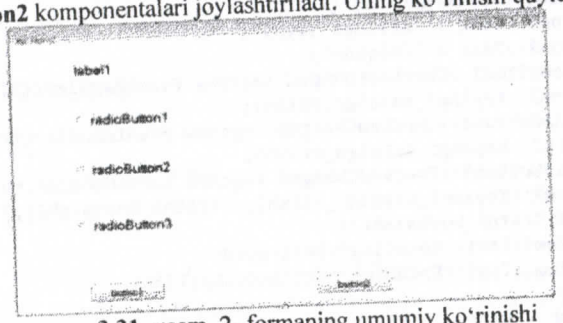
```
private: System::Void Form1_FormClosing(System::Object^  sender,
System::Windows::Forms::FormClosingEventArgs^  e) {
if (textBox1->Modified == false) return;
auto MBox = MessageBox::Show("Tekst bo`l izmenen. \nSoxranit
izmeneniya?","Prostoy redaktor", MessageBoxButtons::YesNoCancel,
MessageBoxIcon:: Exclamation);
if (MBox == Windows::Forms::DialogResult::No) return;
if (MBox == Windows::Forms::DialogResult::Cancel) e->Cancel =true;
if (MBox == Windows::Forms::DialogResult::Yes){
if (saveFileDialog1->ShowDialog() ==Windows::Forms:: DialogResult::
{ Zapis(); return; }  else e->Cancel = true;  }}
```

7-qadamdan keyin, talabalarni har xil fanlardan baholash uchun y
forma yaratiladi.

8-qadam. Fanlardan test menyusidagi **Informatika** bo'limining **on**
hodisasining kodlari quyidagicha:

```
private: System::Void informatikaToolStripMenuItem_Click(System::Objec
sender, System::EventArgs^  e) {
Form2^ op=gcnew Form2(); this->Hide(); op->Show(); }
```

9-qadam. 2- formada **label1, radiobutton1, radiobutton2, radiobut**
butto1, button2 komponentalari joylashtiriladi. Uning ko'rinishi quyidagicha



3.31- rasm. 2- formaning umumiy ko'rinishi

10-qadam. 2- formadagi **Button1** ning **click** hodisasidagi kodlar quyidagicha:

```
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^  e) {
3. if (Tanlangan_javob == TogriJavobNomeri) TogriJavob = TogriJavob + 1;
4. if (Tanlangan_javob != TogriJavobNomeri){
5. NoTogriJavob = NoTogriJavob + 1; NoTogri_javoblar[NoTogriJavob] =
6. label1->Text; }
7. if (button1->Text == "Testni boshidan boshlash") {
8. button1->Text = "Keyingi savol";
9. radioButton1->Visible = true; radioButton2->Visible = true;
10. radioButton3->Visible = true; Testni_boshalsh(); return; }
11. if (button1->Text == "Yakunlash") { oqish->Close();
12. radioButton1->Visible = false; radioButton2->Visible =
13. false; radioButton3->Visible = false;
14. label1->Text = String::Format("Test yakunlandi.g`n" + "Togri
15. javoblar: {0} iz {1}.\n" +
16. "5 balli tizimda baholash: {2:F2}.", TogriJavob, SavollarSoni,
17. (TogriJavob * 5.0F) / SavollarSoni); button1->Text = "Testni
18. boshidan boshlash";
19. String^ Str = "Siz javob bergan noto`g`ri savollar ro`yhati
20. :\n\n";
21. for (int i = 1; i <= NoTogriJavob; i++)
22. Str = Str + NoTogri_javoblar[i] + "\n";
23. if (NoTogriJavob != 0)
24. MessageBox::Show(Str, "Test yakunlandi"); }
25. if (button1->Text == "Keyingi savol") Key_savol_oqish(); }
```

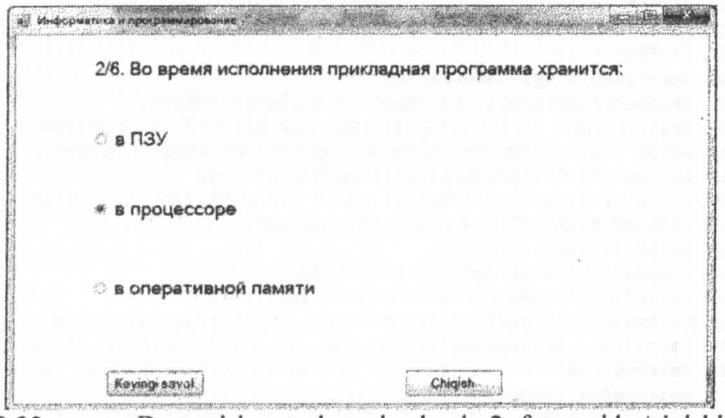
11-qadam. 2- formaning **Formload** hodisasi va unda chaqirilgan bir nechta funksiyalarning kodlari quyidagicha:

```
1. #pragma endregion
2. int SavollarSoni;
3. int TogriJavob; int NoTogriJavob; array<String^>^
   NoTogri_javoblar;
4. int TogriJavobNomeri; int Tanlangan_javob;
5. IO::StreamReader^ oqish;
6. private: System::Void Form2_Load(System::Object^  sender,
   System::EventArgs^  e) {
7. button1->Text = "Keyingi savol";
8. button2->Text = "Chiqish";
9. radioButton1->CheckedChanged +=gcnew EventHandler(this,
10. &Form2::Keyingi_savolga_otish);
11. radioButton2->CheckedChanged +=gcnew EventHandler(this,
12. &Form2::Keyingi_savolga_otish);
13. radioButton3->CheckedChanged +=gcnew EventHandler(this,
14. &Form2::Keyingi_savolga_otish);  Testni_boshalsh();}
15. void Testni_boshalsh(){
16. System::Text::Encoding^ Kodirovka=
17. System::Text::Encoding::GetEncoding(1251);
18. try{
19. oqish = gcnew IO::StreamReader(
```



```
20. IO::Directory::GetCurrentDirectory()+ "\\test.txt", Kodirovka)
21. this->Text = oqish->ReadLine();
22. SavollarSoni = 0; TogriJavob = 0; NoTogriJavob = 0;
23. NoTogri_javoblar = gcnew array<String^>(100);  }
24. catch (Exception^ Situasiya)
25. { MessageBox::Show(Situasiya->Message, "Hatolik",
26. MessageBoxButtons::OK, MessageBoxIcon::Exclamation); }
27. Key_savol_oqish(); }
28. void Key_savol_oqish(){
29. label1->Text = oqish->ReadLine();
30. radioButton1->Text = oqish->ReadLine();
31. radioButton2->Text = oqish->ReadLine();
32. radioButton3->Text = oqish->ReadLine();
33. TogriJavobNomeri = int::Parse(oqish->ReadLine());
34. radioButton1->Checked = false; radioButton2->Checked= false;
35. radioButton3->Checked = false;
36. button1->Enabled = false;
37. SavollarSoni = SavollarSoni + 1;
38. if (oqish->EndOfStream == true) button1->Text = "Zavershit"; }
39. private: Void Keyingi_savolga_otish(System::Object^ sender,
40. System::EventArgs^ e){
41. button1->Enabled = true; button1->Focus();
42. RadioButton^ Pereklyuchatel = (RadioButton^)sender; String^ tm
    = Pereklyuchatel->Name;
43. Tanlangan_javob = int::Parse(tmp->Substring(11)); }
```

Talabalarni fanlardan baholash dasturini ishga tushirilganc keyingi ko‘rinishi quyidagicha:



3.32- rasm. Dastur ishga tushgandan keyin 2- formani ko‘rinishi

3.15. Matnli xujjatni chop qilish

Ixtiyoriy matn muharriri ma'lumotlarni printerdan chop qilish imkoniyatiga ega bo'lishi zarur. Dastur qanchalik ko'p imkoniyatlarga ega bo'lsa, uning dastur kodini va matnini tushunish shunchalik qiyin bo'ladi. Ushbu keltirilgan dastur yetarli darajada sodda bo'lgan imkoniyatlarga ega. U matn faylni standart Windows darchasida ochib, uni o'zgartirish imkoniga ega bo'lmagan (ReadOnly) holda dastur darchasida ko'rish va matnni printerdan chiqarish imkonini beradi.

Ushbu dasturni yaratish uchun formaga quyidagi komponentalar joylashtiriladi: TextBox matn maydoni, "**Ochish**", "**Chop qilish**", va "**Chiqish**" bo'limlariga ega bo'lgan MenuStrip menyusi, hamda **OpenFileDialog** va **PrintDocument** komponentalari.

Dastur kodi quyida berilgan:

```
1. #pragma endregion
2. IO::StreamReader^ uqish;
3. private: System::Void Form1_Load(System::Object^  sender,
4. System::EventArgs^  e) {this->Text = "Matn faylni chop qilish";
5. richTextBox1->Clear();
6. //richTextBox1->ScrollBars = ScrollBars::Vertical;
7. chopQilishToolStripMenuItem->Visible = false;
8. openFileDialog1->FileName = nullptr;
9. richTextBox1->Font =gcnew Drawing::Font("Times New Roman",
   14.0F);}
10. private: System::VoidochishToolStripMenuItem_Click(
   System::Object^  sender, System::EventArgs^  e) {
11. // Ochish menyusi buyrug'i tanlanganda:
12. openFileDialog1->Filter = "Mant fayllar (*.txt)|*.txt|All files
13. (*.*)|*.*";
14. openFileDialog1->ShowDialog();
15. if(openFileDialog1->FileName == nullptr) return;
16. try{//Fayldan o'qish uchun StreamReader potokini hosil qilish
17. uqish = gcnew IO::StreamReader(openFileDialog1->FileName,
18. System::Text::Encoding::GetEncoding(1251));
19. // bu yerda kril alifbosini oqish uchun Win1251 ni o'qish
20. richTextBox1->Text = uqish->ReadToEnd();
21. uqish->Close();
22. chopQilishToolStripMenuItem->Visible = true; }
23. catch(IO::FileNotFoundException^ Istisno){
24. MessageBox::Show(Istisno->Message+ "\n Bunday fayl yo'q",
25. "Xatolik", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);}
26. catch(Exception^ Istisno){// Boshqa xatoliklar haqida xabar
27. MessageBox::Show(Istisno->Message, "Xatolik",
28. MessageBoxButtons::OK, MessageBoxIcon::Exclamation);} }
29. private: System::Void chopQilishToolStripMenuItem_Click(
```



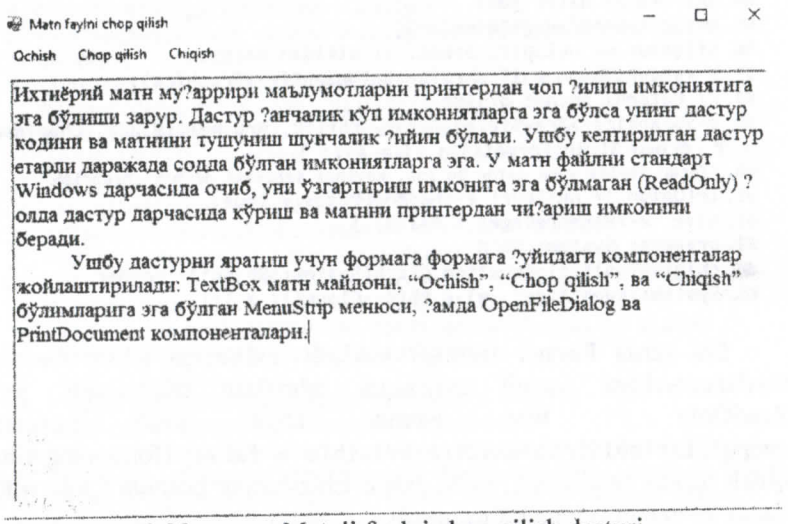
```
30. System::Object^  sender, System::EventArgs^  e) {
31. // Chop qilish menyu buyrug'i tanlanganda
32. try{
33. uqish = gcnew IO::StreamReader(openFileDialog1->FileName,
34. System::Text::Encoding::GetEncoding(1251));
35. try{
36. printDocument1->Print();
37. }
38. finally{uqish->Close();} }
39. catch(Exception^ Istisno){ MessageBox::Show(Istisno->Message);}
   }
40. private: System::Void printDocument1_PrintPage(System::Object^
41. sender, System::Drawing::Printing::PrintPageEventArgs^  e) {
42. // Sahifani pechatga chiqarish hodisasi
43. Single sahifadagiQatorlar = 0;
44. Single Y = 0;
45. Single chapTomo = (Single) e->MarginBounds.Left;
46. Single yuqoriTomon = (Single) e->MarginBounds.Top;
47. String^ Qator = nullptr;
48. Drawing::Font^ shrift =
49. gcnew Drawing::Font("Times New Roman", 12.0F);
50. // Bitta sahifadagi qatorlar soni
51. sahifadagiQatorlar = e->MarginBounds.Height/shrift->GetHeight(
52. e->Graphics);
53. // Faylni har bir qatorini pechatga chiqarish
54. int i=0;// qator soni
55. while(i<sahifadagiQatorlar){
56. if(Qator == nullptr) break; // sikldan chiqish
57. Y =yuqoriTomon + i* shrift->GetHeight(e->Graphics);
58. // Qatorni pechat qilish
59. e->Graphics->DrawString(Qator, shrift, Brushes::Black, chapTomo
   Y, gcnew StringFormat());  i = i+1; }
60. //Agar faylda yana qator bo'lsa, keyingi sahifani pechat chiqarish
61. if(Qator != nullptr) e->HasMorePages = true;
62. else  e->HasMorePages = false; }
63. private: System::Void
64. chiqishToolStripMenuItem_Click(System::Object^  sender,
65. System::EventArgs^  e) { this->Close(); } };}
```

Bu yerda **Form1** formani yuklash hodisasiga ishlov berish foydalanuvchiga matnli maydonni tahrirlash taqiqlanadi, ya' ReadOnly = true; Hamda chop qilish xossasi **chopQilishToolStripMenuItem->Visible = false;** (formaning **Ch qilish** bandi) belgilanadi, ya'ni dastur ishlashining boshida **Chop qili** bandi foydalanuvchiga ko'rinmaydi.

Formaning Ochish xossasiga ishlov berishda OpenFileDial standart muloqot chaqiriladi va faylni o'qishni StreamReader oqimi

yaratish orqali tashkil qilinadi. Try…finally…catch bloklarda dastur yana bir marta StreamReader oqimini yaratadi, so‘ngra, xujjatni chop qilish jarayonini ishga tushiradi printDocument1->Print. Agar faqat printDocument1->Print metodidan boshqasi dasturlashtirilmagan bo‘lsa, u holda printer bo‘sh sahifani chop qiladi. Printer matnni chop qilishi uchun, printDocument1 ob`ektni yaratuvchi Printpage hodisasiga ishlov berish zarur.

PrintDocument1_PrintPage hodisasiga ishlov berish MSDN da keltirilgan. Birinchi navbatda o‘zgaruvchilar e’lon qilingan, ularning ba’zi birlari “e” hodisasining argumentidan olingan. Chop qilish shrifti Times New Roman, 12 punktda deb belgilanadi. While siklida dastur Chitate->ReadLine() fayldan har bir satrni (line) o‘qiydi, so‘ngra uni **DrawString** buyrug‘i chop qiladi. Bu yerda “e” hodisasining argumentidan olinuvchi (Graphics) grafik ob`ektidan foydalaniladi. I o‘zgaruvchisida satrlar sanaladi. Agar satrlar miqdori sahifa satrlari sonidan katta bo‘lsa, u holda dastur satr o‘zgaruvchisidagi qiymat orqali tahlil qilib, aniqlashtiradi. Agar undagi qiymat **nullptr** qiymatidan farqlansa (Satr != nullptr), u holda argument o‘zgaruvchisi bo‘lgan **e.HasMorePage true** qiymat oladi.



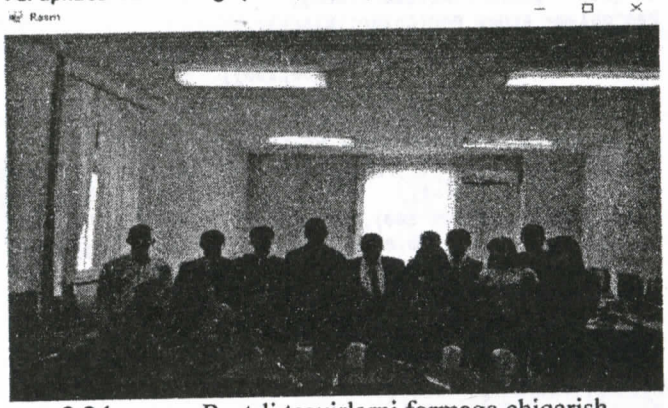
3.33- rasm. Matnli faylni chop qilish dasturi

3.16. Formaga grafik fayldagi tasvirni chiqarish

Rastrli grafikadagi BMP, JPEG, PNG formatidagi fayllar tasvi formaga chiqarish masalasi qo‘yilgan bo‘lsin. Formada grafika bi ishlashni turlicha amalga oshirish mumkin. Grafik bilan ishlas OnPrint metodini qayta aniqlash orqali ko‘rib chiqamiz. OnPrint met Form sinfining a’zosi hisoblanadi.

Ushbu vazifani bajaruvchi dasturni tuzish uchun **Visual Stu 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo‘limlaridan **F >New->Project…** buyruqlari beriladi yoki **Ctrl+Shift+N** klavisha bosiladi, ochilgan oynada loyihaga “**Grafik_rasmlar**” nomi beriladi **OK** tugmasi bosiladi va quyidagi kodlar teriladi:

```
1. #pragma endregion
2. private: System::Void Form1_Paint(System::Object^  sender,
   System::Windows::Forms::PaintEventArgs^  e) {
3. this->Text = "Rasm";
4. // Formning o'lchami
5. this->Width = 640; this->Height = 480;
6. // Rasm bilan ishlash uchun obyekt hosil qilinadi
7. Image^ rasm = gcnew Bitmap("rasm.png");
8. // rasmni formaga chiqarish
9. e->Graphics->DrawImage(rasm, 5, 5);} }; }
```



3.34- rasm. Rastrli tasvirlarni formaga chiqarish

3.17. Formada grafik shakllarni va funksiya grafiklarini chizis

Vektorli chizmalarda quyidagi grafik shakllar chizma elementar tashkil etuvchilari deyiladi: kesma, yoy, belgi, aylana va

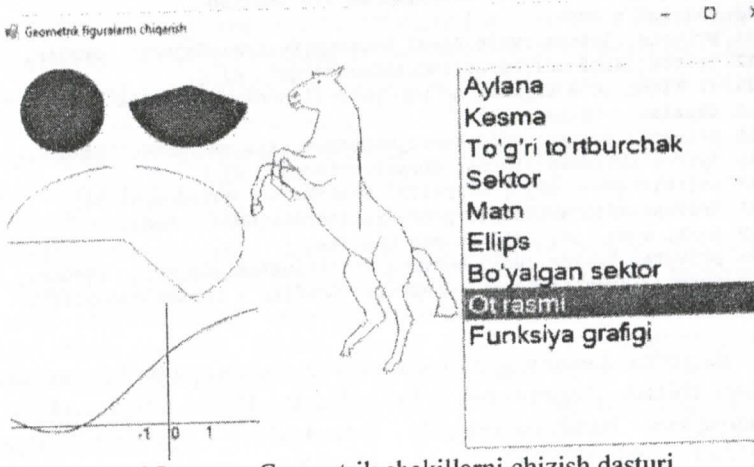
Bunda rastrli grafika asosida koordinata bo‘yicha shakl chizadi. Visual Studioda koordinata tizimi quyidagicha aniqlanadi: koordinata boshi formaning chap yuqori burchagi, Ox o‘qi o‘ngga, Oy o‘qi pastga yo‘naltirilgan.

Qo‘yilgan vazifa formada aylana, kesma, to‘rtburchak, sektor, matn, ellips, bo‘yalgan sektor va matematik funksiya grafiklarini chizish. U yoki bu grafik shaklni tanlashni **ListBox** komponentasi orqali amalga oshirish mumkin. Shuni hisobga olish kerakki, boshqa shaklni chizayotganda oldingisini o‘chirish kerak bo‘ladi. Vazifani amalga oshirish uchun formaga ListBox komponentasini joylashtirib, quyidagi kodlar teriladi:

```
#pragma endregion
private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {
this->Text = "Sodda figurali shakillar chizish";
listBox1->Items->AddRange(gcnew array<Object^>{"Aylana", "Kesma",
"To'g'ri to'rtburchak",     "Sektor", "Matn", "Ellips", "Bo'yalgan
   sektor",
"Ot rasmi","Funksiya grafigi"}); }
void Fun_gra(){ float x,y;
Graphics^Figura=this->CreateGraphics();
Pen^Qalam= gcnew Pen(Color::Red);
Pen^qalam= gcnew Pen(Color::Black);
Brush^MQalam =gcnew SolidBrush(Color::Red);
Figura->DrawLine(qalam,0,250,400,250);
Figura->DrawLine(qalam,200,0,200,800);
Figura->DrawString("0",Font,MQalam,200,250);
Figura->DrawString("-1",Font,MQalam,160,250);
Figura->DrawString("1",Font,MQalam,240,250); x=-1;
do   { y=(x-3)/(x*x+2);
Point l=Point(100+(x*100),100-(y*100));
Point k=Point(100+(x+0.001)*100,100-(y+0.001)*100);
array<Point>^qizil={l,k};
Figura->DrawPolygon(Qalam,qizil);
x+=0.001; }while (x<=4);}
private: System::Void
listBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
SolidBrush^ blueBrush = gcnew SolidBrush(Color::Blue );
// point o'zgarmaslarini yaratish.
Point point1 = Point(50,50); Point point2 = Point(100,25);
Point point3 = Point(200,5); Point point4 = Point(250,50);
Point point5= Point(300,100); Point point6 = Point(350,200);
Point point7 = Point(250,250); array<Point>^ curvePoints =
{point1,point2,point3,point4,point5,point6,point7};
//Figurali obyektni hosil qilish
```



```
Graphics^ Figura = this->CreateGraphics();
//Figurali shakilni chizish uchun qalamni hosil qilish
Pen^ Qalam = gcnew Pen(Color::Red);
//Figurali shaklni bo'yash uchun mo'yqalamni hosil qilish
Brush^ Mqalam = gcnew SolidBrush(Color::Red);
//Figurali shakl chizish sohasini tozalash bilan birgalikda
   forma rangini berish
Figura->Clear(SystemColors::Control);
switch (listBox1->SelectedIndex){
// Aylana chizish
case 0: Figura->DrawEllipse(Qalam, 50, 50, 250, 150);  break;
// Kesma chizish
case 1: Figura->DrawLine(Qalam, 50, 50, 200, 200); break;
// To'g'ri to'rtburchak chizish
case 2: Figura->DrawRectangle(Qalam, 50, 30, 150, 280);  break
// Sektor chizish
case 3: Figura->DrawPie(Qalam, 30, 50, 300, 200, 180, 225);
break;
// Matn chiqarish
case 4: Figura->DrawString("Har narsaning iloji bor", Font,
Mqalam, 10, 100); break;
// ellips chiqarish
case 5: Figura->DrawEllipse(Qalam, 30, 30, 150, 200); break;
// Bo'yalgan sektor chiqarish
case 6: Figura->FillPie(Mqalam, 20, 50, 150, 150, 20, 145);
break;
case 7:    Figura->DrawLine(Qalam, 167, 250, 165, 239); break
case 8: Fun_gra(); break;}}}; }
```



3.35- rasm. Geometrik shakillarni chizish dasturi