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 Access**da 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‘zingizninig 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

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

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 ishlashi 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 **.Net** platformasini yaratilishi **Windows** muhitida hamma narsani, ya`ni dasturlash tillarini, interfeys va kutubxonalarni hamda ilovalarni o‘zgartirib yuborishini aytib o‘tdi. **.Net** markasi orqali quyidagi asosiy mahsulotlar yetkazib beriladi, bular:

-.Net Framework – amalga oshirish muhiti, unda yaratilgan dasturiy komponentlar ishlatiladi. Bu muhit dasturiy kodlarni xavfsizligini ta`minlash, avtomatik ravishda keraksiz kodlarni yig‘ishtirish va boshqa ishlar uchun mo‘ljallangan.

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

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

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

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

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

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 konsolli ilovalar va grafik interfeys bilan ishlovchi ilovalar yaratish imkoniyatini beradi. **Windows Forms** texnologiyasini qo‘llagan holda web-saytlar, web-ilovalar, web-xizmatlar, turli xil platformalar kodlarini boshqaruvchi (**Windows, Windows Mobile, .NET Windows, Windows CE, .NET Framework, Xbox, Windows Phone, .NET Compact Framework** va **Silverlight** texnologiyalari asosida**)** amaliy dasturlarni yaratish mumkin.

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

Komponentlari. Visual Studio quyidagi bir qancha 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 berish mumkin:

Microsoft SQL Server yoki Microsoft SQL Server Express.

Visual Studioning oldingi tarkibiga quyidagi mahsulotlar kiritilgan:

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

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

Visual Studio 97 – Visual Studioning birinchi ishlab chiqilgan versiyasi hisoblanib, turli xil muhitlarda dasturiy ta`minot yaratish maqsadida ilk bora ikki xil versiya (**Professional** va **Enterprise)** da

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 nomi **everett**, ichki versiya 7.1) – 2003 yilda ishlab chiqarilgan (**Framework 1.1** qo‘llaydi). Microsoft 2005 yilning aprelida maxsus muhit ishlab chiqarilganini e`lon qildi, uning nomi **Microsoft Visual Studio .NET Professional Special Edition** deb ataldi. Bu maxsus ishlab chiqarilgan muhit **Microsoft Visual Studio .NET 2003 Professional Edition**ga dasturiy ta`minotning Serverli komplektini va boshqa instrumentlarni qo‘shadi (bu qismlarga operasion tizimlar: **Windows Server 2003 Standard Edition** va **SQL Server 2000 Developer Editionlar** kiradi). Muhitni bunday ko‘rinishda rag‘batlantirish ishlab chiqarishning yangi bosqichiga ko‘tarilishiga olib keldi va Microsoft boshqalar bilan raqobatga kirisha oladigan dasturiy ta`minot yaratuvchini ishlab chiqdi.

Visual Studio .NET 2003 Service Pack 1 uchun 2006 yil 13 sentyabrda ishlab chiqarilgan.

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

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

Visual Studio 2010. Visual Studio 2010 (kodli nomi ***Hawaii,*** **Ultimate** uchun – ***Rosario***, ichki versiya 10.0) - **.NET Framework 4.0** bilan birgalikda 2010 yil 12-aprelda yaratilgan. **Visual Studio C# 4.0 va Visual Basic .NET 10.0** oldingi versiyalarida yo‘q bo‘lgan F# tilini 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 dasturlar yaratish mumkin.

1.2. Texnika kodi va dasturlash tillari

Visual C++ dasturlash muhiti

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

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

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

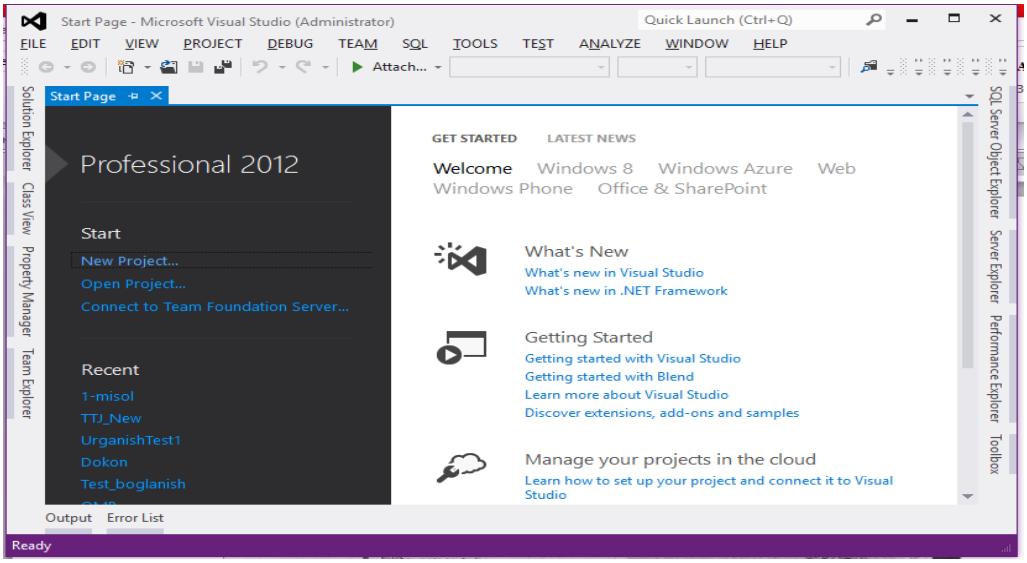
Visual C++ .NET dasturlash tili past sathdagi dasturchi uchun ilovalarni boshqarishda talab qilinadi. **.NET** platformasining **Visual C++** dasturini boshqa dasturlardan shu bilan farq qiladiki, bu dasturlash tili **.NET** platformasining kodli modeli (**managed code model**) va

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.

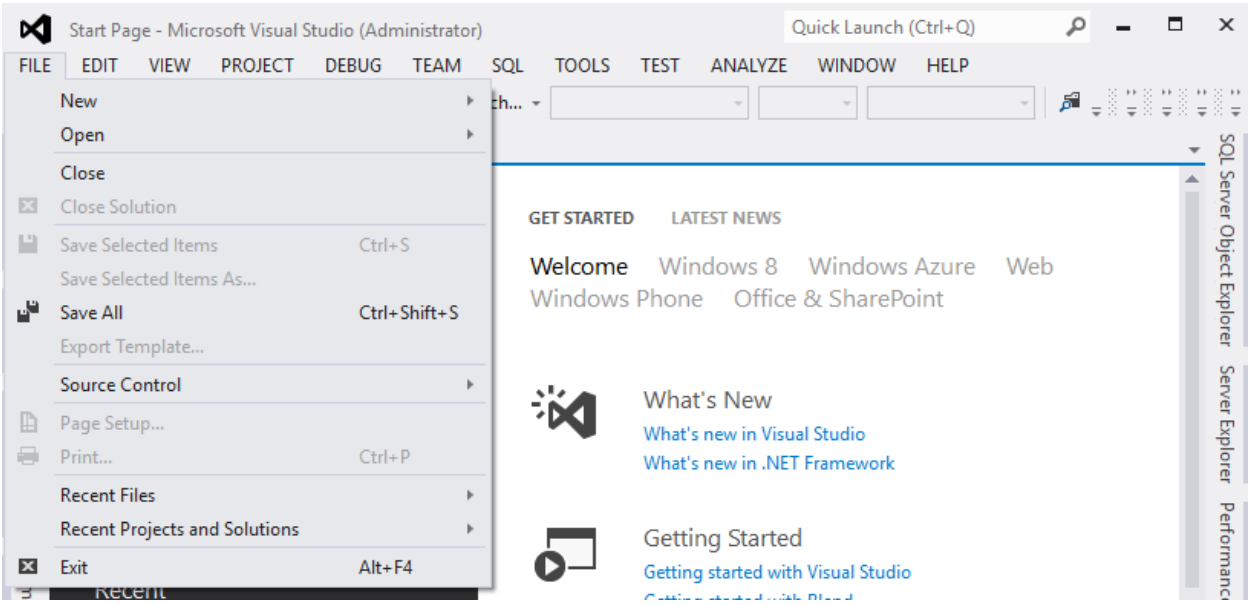
Visual C#.NET dasturlash tili. Zamonaviy dasturlash tillari orasida **Visual C#.NET** mukammal dasturlash tillaridan biri hisoblanadi. Ushbu dasturlash tili C++ asosida kelib chiqqan bo‘lib, u **.NET** Microsoft platformasi uchun yaratilgan va ob`ektli dasturlashga mo‘ljallangan. Uning yordamida internetda 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.

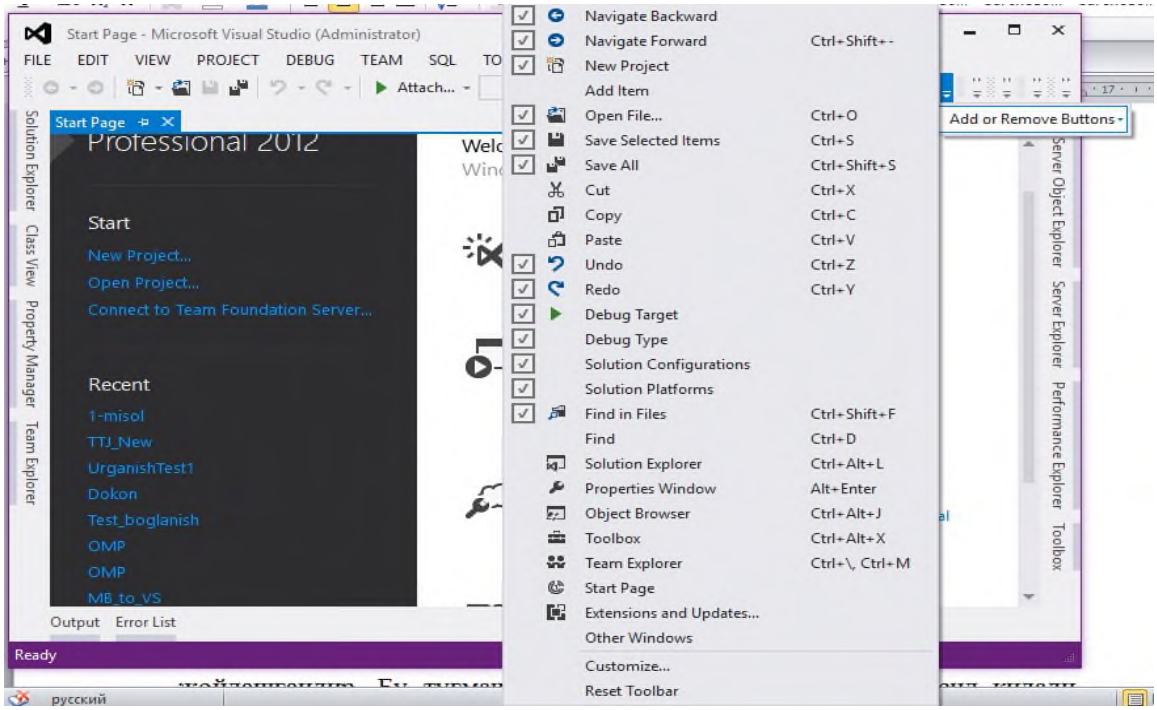


1.2- rasm. Visual Studio 2012 IDE oynasini ko‘rinishi

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

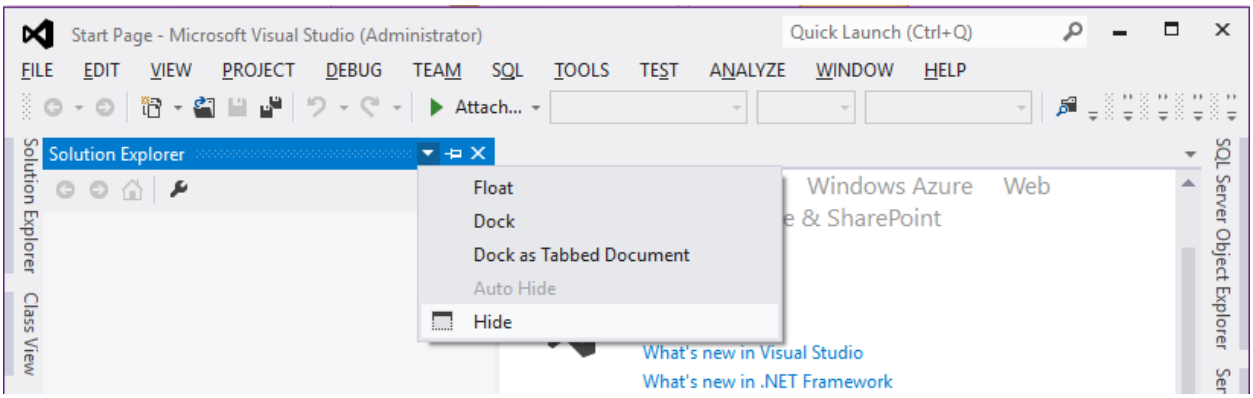


1.4- rasm. Tezkor tugmalar majmuasi

Pastki satrlarda asosiy menyuda **tezkor murojaatlar oynasi** joylashgandir. Bu tugmachalar suzib chiquvchi oynalarni hosil qiladi.

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.

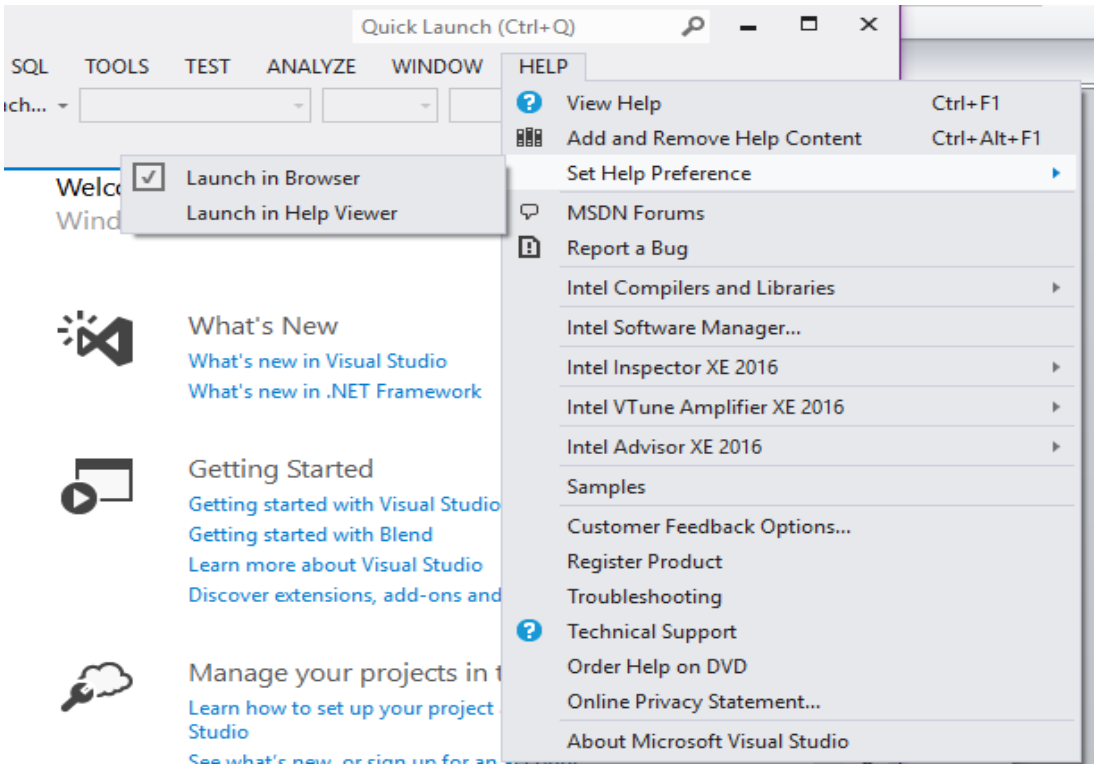
Misol uchun Toolbox va **Server Explorer** ma`lumot-oynasini qanday qilib o‘rnatish?

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

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 zarur bo‘lgan maxsus konstruksiyalarni qarab chiqamiz. Bu yerda ham konsolli ilovalarda o‘rganilgan tushunchalardan foydalaniladi. Bu va bundan keyingi bo‘limlarda biz murakkab ilovalarning grafik interfeysi bilan tanishamiz. Ilovalarni yaratishda forma tushunchasini ishlatamiz. Bosh oynaning chap qismida ikki bo‘lakli vkladka joylashgan. Birinchi qism ma`lumotlar bazasi bilan ishlashni tashkil etsa, ikkinchi qismda komponentalar ro‘yxati keltirilgan.

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

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

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

Bu oynalar bilan ishlash davomida go‘yoki mashhur dasturlardan biri bo‘lmish Photoshop oynalari yodga keladi. Bu oynalarning xossalari bir-biri bilan uzviy bog‘liq. Oynalarni keraklicha joylashtirishni muhitning *Buð* menyusi orqali ham amalga oshirish mumkin.

Tasodifiy fakt 1.3. Standartlovchi tashkilotlar

Amerika milliy standartlash instituti (ANSI - American National Standards Institute) va Halqaro standartlovchi tashkilot (ISO - International Organization for Standardization) hamkorlikda C++ tili 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

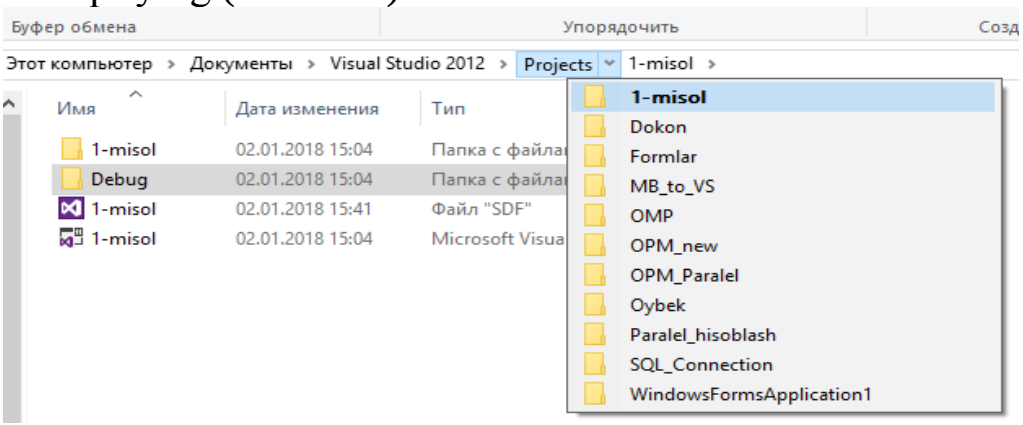
harflar farqlanadi, ayrimlarida esa yo‘q. Fayllar papkalar yoki kataloglarda saqlanadi. Papkalar o‘z ichiga fayllarni oladi va yana ichida fayllar va papkalar bor boshqa fayllarni ham saqlay oladi(1.7-rasm). Bu ierarxiya katta bo‘lishi mumkin va Siz uning barcha bo‘limlari bilan tanish bo‘lmasligingiz 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. Prosessordagi xabar xato xabar sababli dasturni tugatilishiga olib keladi. Misol uchun dasturingiz `Console::WriteLine("Hello\" World\"");` bo‘lsa "nol bilan ajratish" istisnosi bilan tugallanadi.

Dasturni rivojlantirish mobaynida, xatolikdan qochish qiyin. Agar dastur bir necha qatordan ko‘p bo‘lsa, kirgizish uchun aqlbovar qilmas diqqat e`tiborni talab qiladi. Siz vergul va qavslarni o‘zingiz bilmagan tarzda tushirib qoldirasiz, lekin kompilyator sizga ushbu muammolarni qidirib topib beradi.

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

RUN TIME ERROR muammolidir. Kompilyator ularni topa 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 Main funksiasiga aynan o‘xshashligini ko‘rib chiqmaydi, chunki Main katta

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 koʻrsatma 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 koʻrib chiqamiz: Siz bank hisob raqamingizga yiliga 5% foyda qiladigan $10,000 miqdordagi pulni qoʻydingiz. Dastlabki miqdor ikki baravar koʻpayishi uchun hisob balansi uchun qancha yil talab etiladi? Ushbu muammoni qoʻllar yordamida yecha olasizmi? Albatta, siz balansni quyidagicha hisoblaysiz:

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

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 taxmindan uzoq, aniq va loʻnda boʻlishi 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 qoʻshing. Foydani **balance x** 0.05 (ya`ni 5% foyda) deb hisoblang. Foydani balansga qoʻshing. Oxirgi yil qiymati javobligi toʻgʻrisida xisobot bering. Albatta, bu qadamlar kompyuter tushuna oladigan tilda emas, lekin siz tez orada ularni C++da qanday ifoda etishni oʻrganib olasiz. Bunday norasmiy tavsif psevdokod deb ataladi. Psevdokod uchun qat`iy talablar yoʻq, chunki uni kompyuter emas, odamlar oʻqishadi. Bu yerda bu kitobda foydlanadigan psevdokod operatorlarining turlari berilgan.

Har qaysi bosqichda nima qilish va keyin qaerga borish kerakligi toʻgʻrisida aniq koʻrsatmalar boʻlsa, bu aniq metoddir. Taxmin va yaratuvchanlikka joy yoʻq. Har qaysi bosqich amalda bajarilganda bu usul amalga oshuvchi metoddir. Agar bizdan yiliga 5 foizli aniq qiymatdan emas, yaqin yillarda olinadigan haqiqiy foiz qiymatidan foydalanish soʻralsa, bizning usul amalga oshadigan metod boʻlmaydi, chunki aynan oʻsha foyda qiymatini hech kim hech qanday usulda bila olmaydi. Agar usul nihoyasiga yetsa, u tugallanadigan metoddir. Bizning misolda metod uzluksiz davom etmasligini koʻrish ozgina fikrlashni talab etadi: Har qaysi bosqichda balans kamida $500 miqdoriga oshmoqda va shunday qilib u natijada $20,000 miqdoriga

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.

2-BOB

Visual C++ ning Console Application muhiti va unda ishlash

Bob maqsadlari

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.

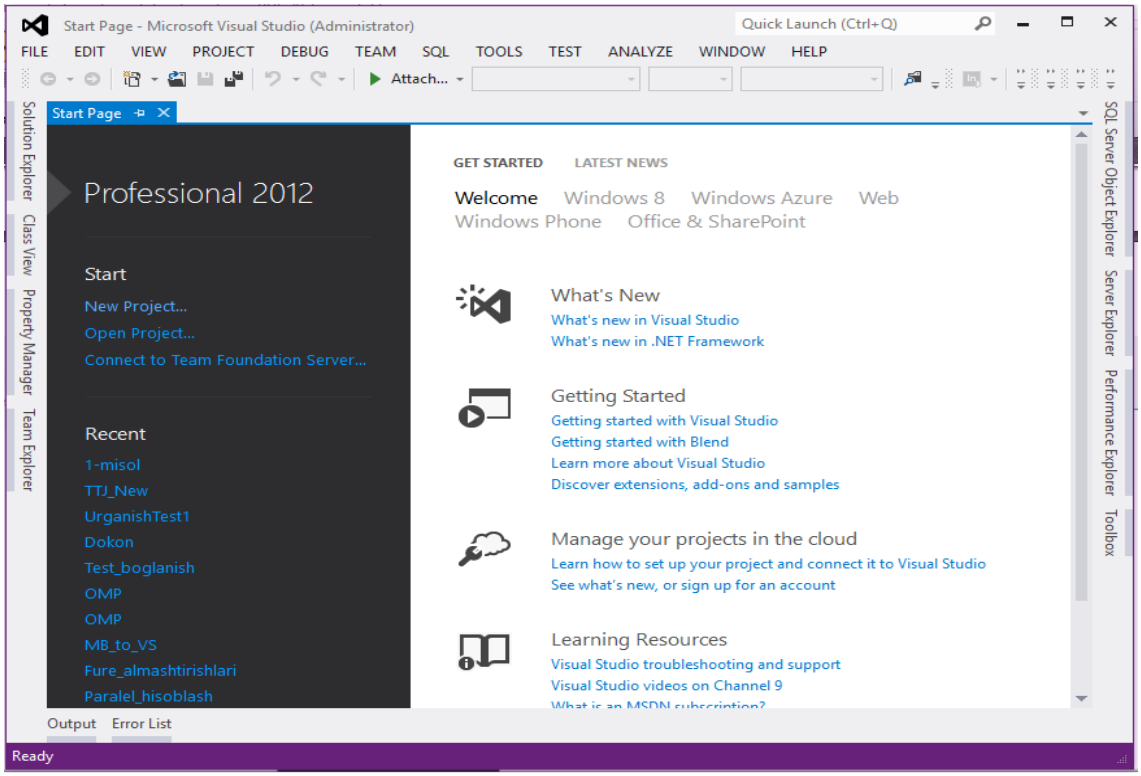
Microsoft Visual Studio 2012 ning o‘rnatiluvchi (инсталляционный) dasturi joylashgan papkadagi **vs_professional.exe** faylini ishga tushiriladi va ekranda quyidagi(2.1- rasm) oyna hosil bo‘ladi. Bu oynada dasturning lisenziya shartlariga roziligi so‘raladi. **“I agree to the License terms and conditions.”**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

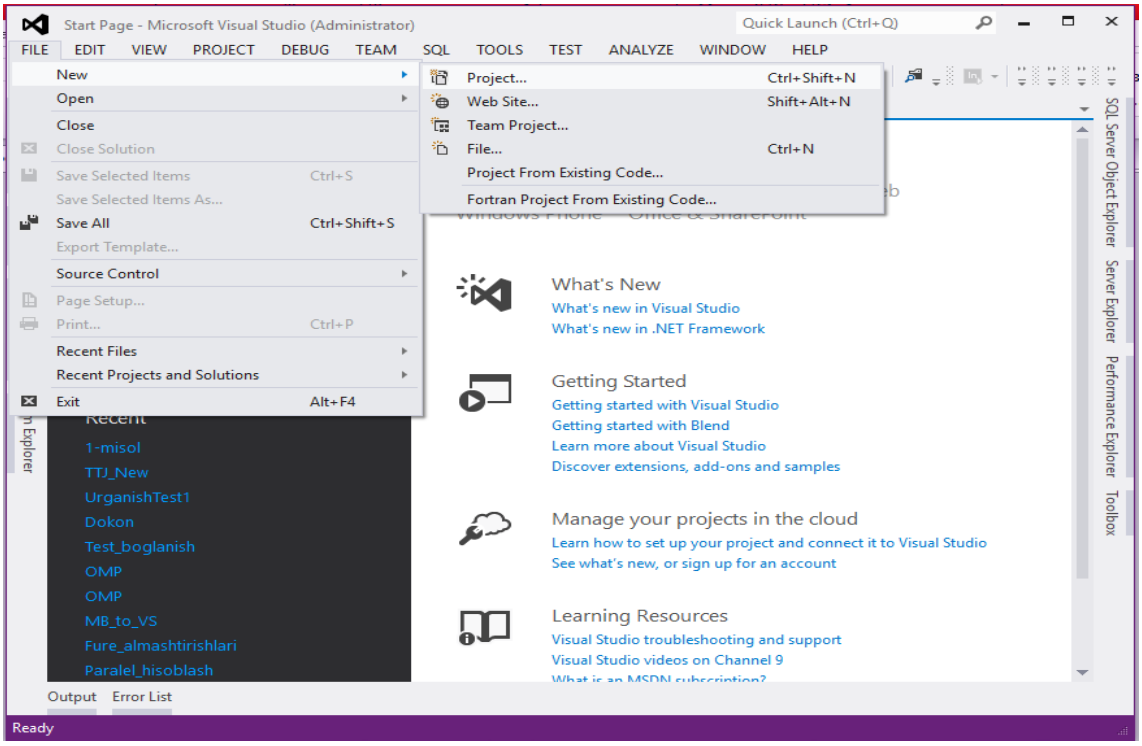


2.3- rasm. VS ning ishchi oynasining boshlang‘ich ko‘rinishi

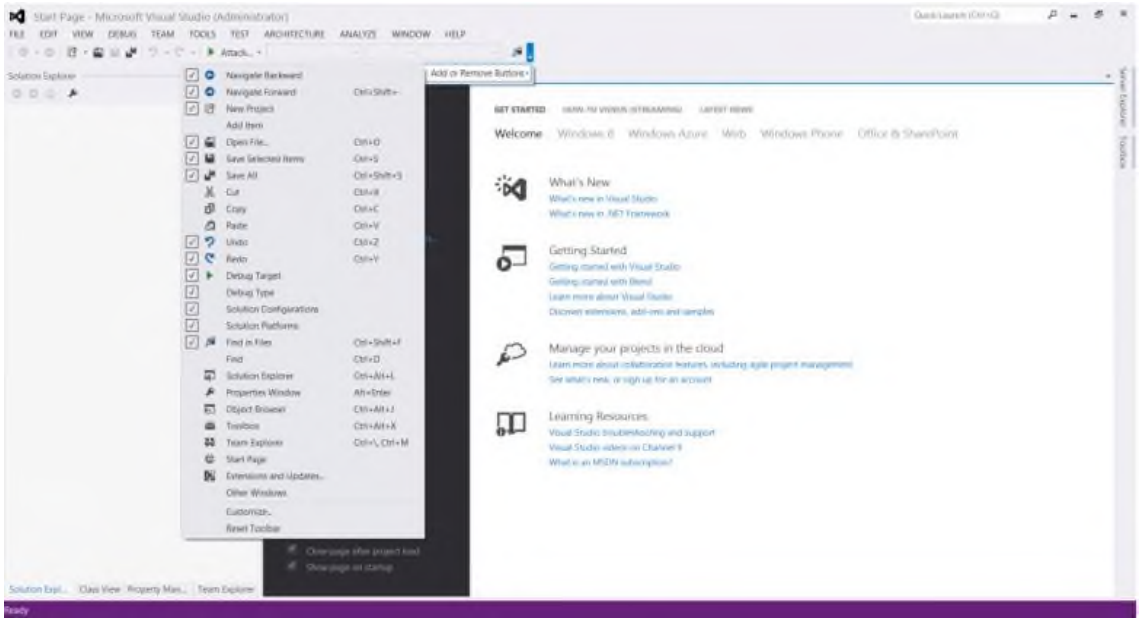
Interfeys – bu foydalanuvchiga muhit bilan qulay o‘zaro aloqani ta`minlovchi apparatdir. Kompyuteringizda **Microsoft Visual C++** dasturini o‘rnatganingizdan keyin uni Пуск/Программы bandidan ishga tushirishingiz mumkin. Qulaylik uchun o‘rnatilgan dasturiy mahsulot belgisini operasion tizim ishchi stolining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

Oynaning yuqori qismida *asosiy menyu komandalari* qatori (File, Edit, …komandalari) – gorizontal menyu qatori joylashgan. Bu komandalardan foydalanishda (ularni yana opsiyalar deb ham nomlashadi, ya`ni bir necha qiymatlar ichidan tanlanuvchi elementlar) “*tushuvchi menyu*” deb nomlanuvchi menyu ochiladi – bu menyu vertikal menyu bo‘lib, o‘zida ekranning yuqorisidan pastiga qarab joylashgan buyruqlar to‘plamini o‘zida namoyon qiladi.

Asosiy menyu qatori ostida joylashgan ikkinchi qatorda ba`zi bajariluvchi buyruqlarning *tezkor chaqiruv tugmalari* joylashgan. Bu tugmalarning barchasi suzuvchi izohga ega (sichqoncha kursorini tugmachaning ustiga olib borish va biroz kutish zarur, shundan so‘ng 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 konsol ilovalarni yaratish haqida suhbatlashamiz. Ba`zida, masalan, ilmiy hisoblashlar uchun qandaydir eng oddiy berilganlarni kiritish va kiritilgan ma`lumotlarni murakkab matematik hisoblashlar bilan qayta ishlash, olingan natijalarni tezlik bilan ekranga chiqarish talab qilinadi. Bunday vaziyatlar katta dasturlarni qismlarga ajratib yaratish vaqtida vujudga keladi. Dasturlashda konsol ilovali deb ataladigan turlicha dasturlarni tashkil qilish mumkin. Odatda konsol rejimida kompyuter ekrani va klaviatura nazarda tutiladi. Misol tariqasida, qandaydir ikkita sonni kiritish va hisoblashlar natijasini konsol ilovada chiqarish taklif qilinsin. Yangi (New Project) loyihaning nomi sifatida natijalar yig‘indisi deb nom berib, OK tugmasi bosiladi va dasturlash kodi yoziladigan muhitga o‘tiladi (2.7- rasm).

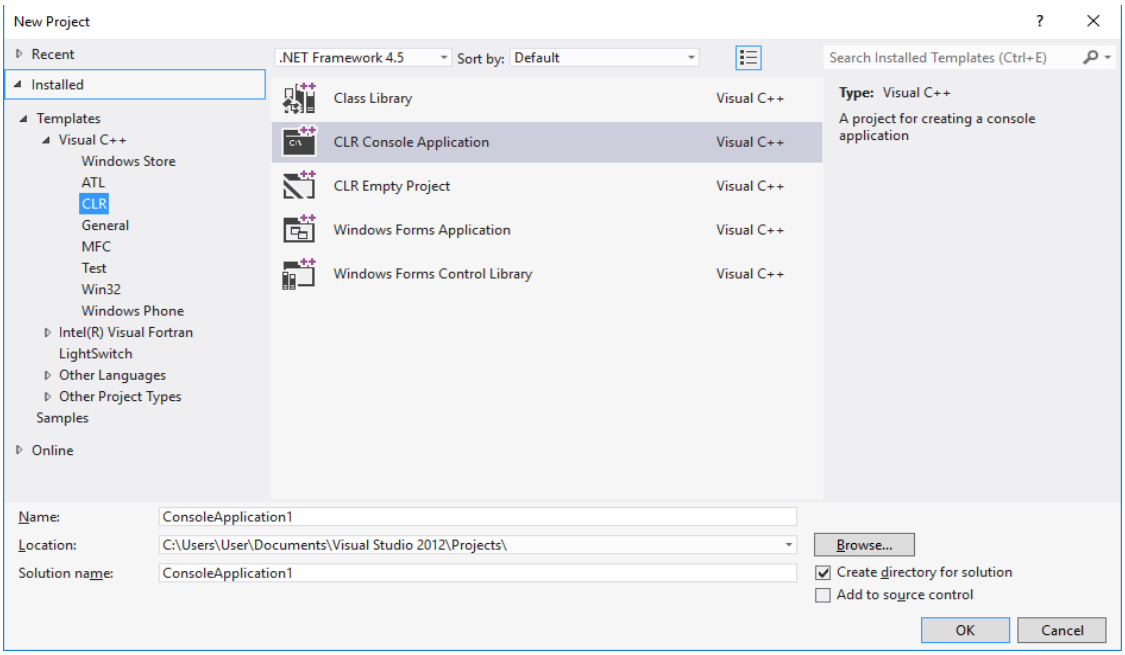
Visual Studio dasturining boshqarish muhiti bir necha qator dasturlash kodlarini hosil qiladi. Bu deyarli ish qobiliyatiga ega dastur, uning operatori orqali ketma-ket F10 tugmasini bosish orqali o‘tish mumkin. Konsol yoki Windows ilova ishga tushganda main() funksiyasi birinchi chaqiriladigan funksiya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

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

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

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

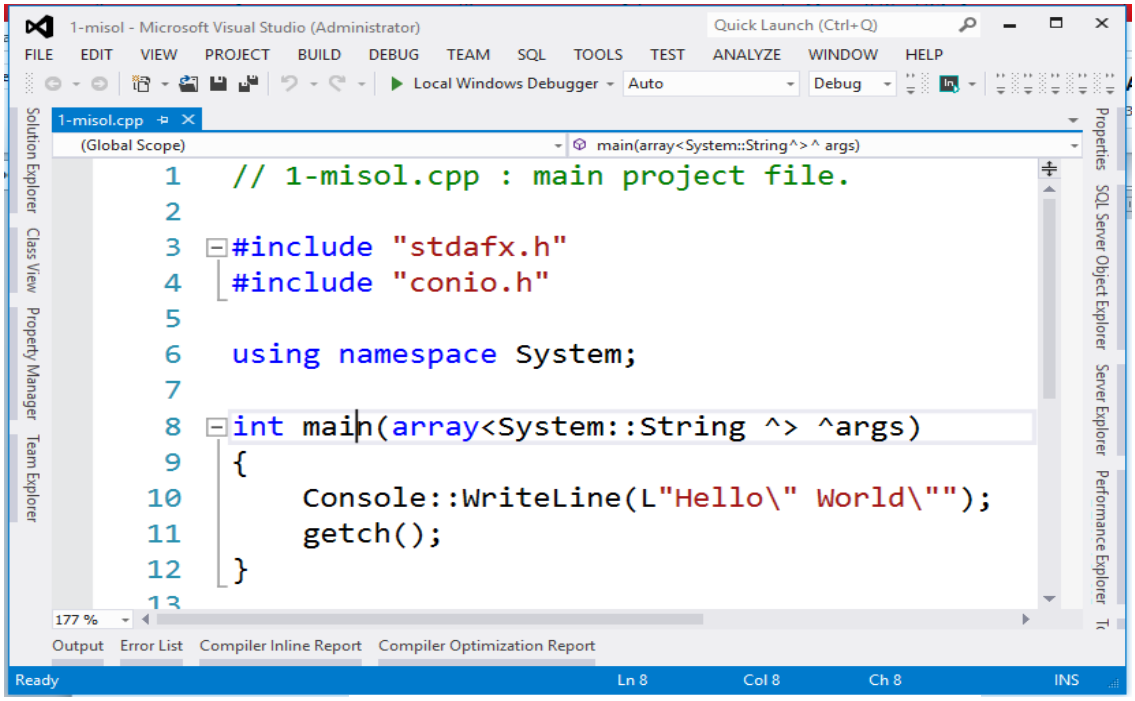
Bu ketma-ketlik bajarilgandan so‘ng, loyihaning tipini tanlash uchun *New Project* (2.6- rasm) oynasi ochiladi. Bu oynadagi ***Visual C++*** qismidan ***CLR*** bo‘limiga o‘tib, ***CLR Console Application*** punkti tanlanadi.



2.6- rasm. New Project oynasi

Name – qismiga loyiha nomi yoziladi (masalan "**1-misol"**);

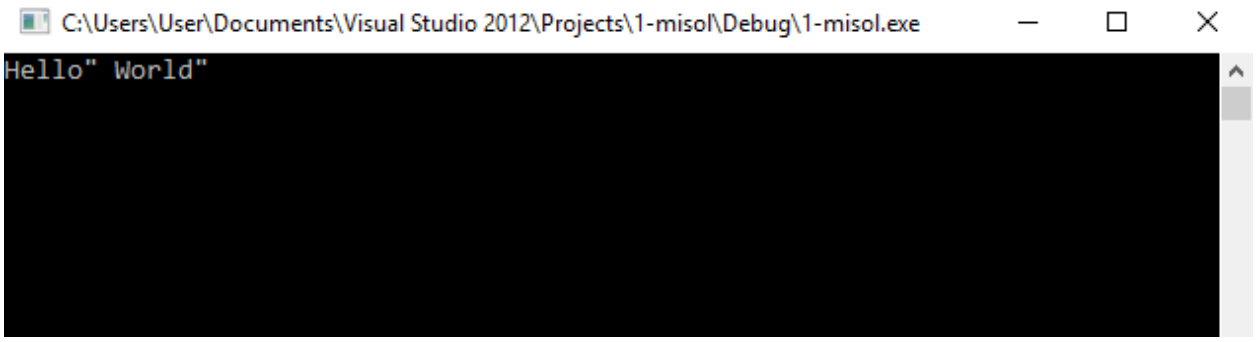
Location – qismiga loyihamiz saqlanadigan manzil ko'rsatiladi. (**Browse…** tugmasi yordamida). So'ngra **OK** tugmasi bosiladi. Natijada Visual C++ muhitining ishchi oynasi ochiladi. (2.7- rasm).



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

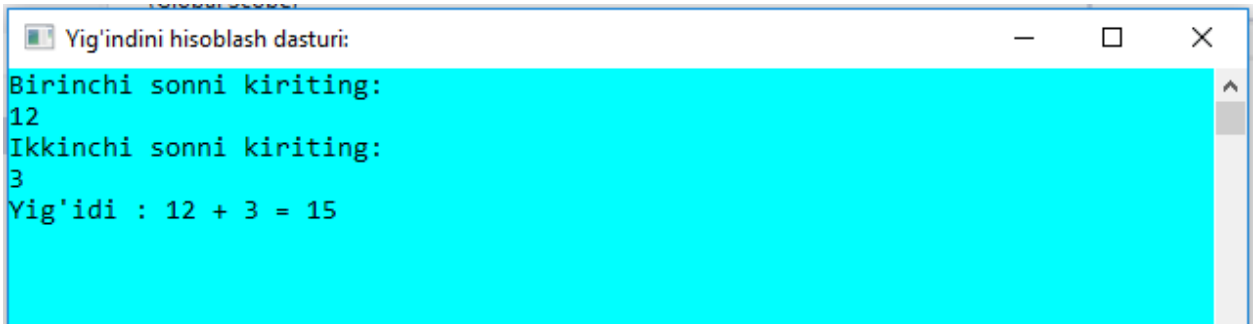
Konsol ilova berilganlarni kiritish va chiqarishga misol sifatida ikkita sonni kiritish va ularning yig‘indisini chiqarish dasturini tuzib, tahlil qilib chiqamiz.

```
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 boshlanadigan nuqta. Odatda konsol ilovalar qora fonli oynada bajariladi. Konsol

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);**

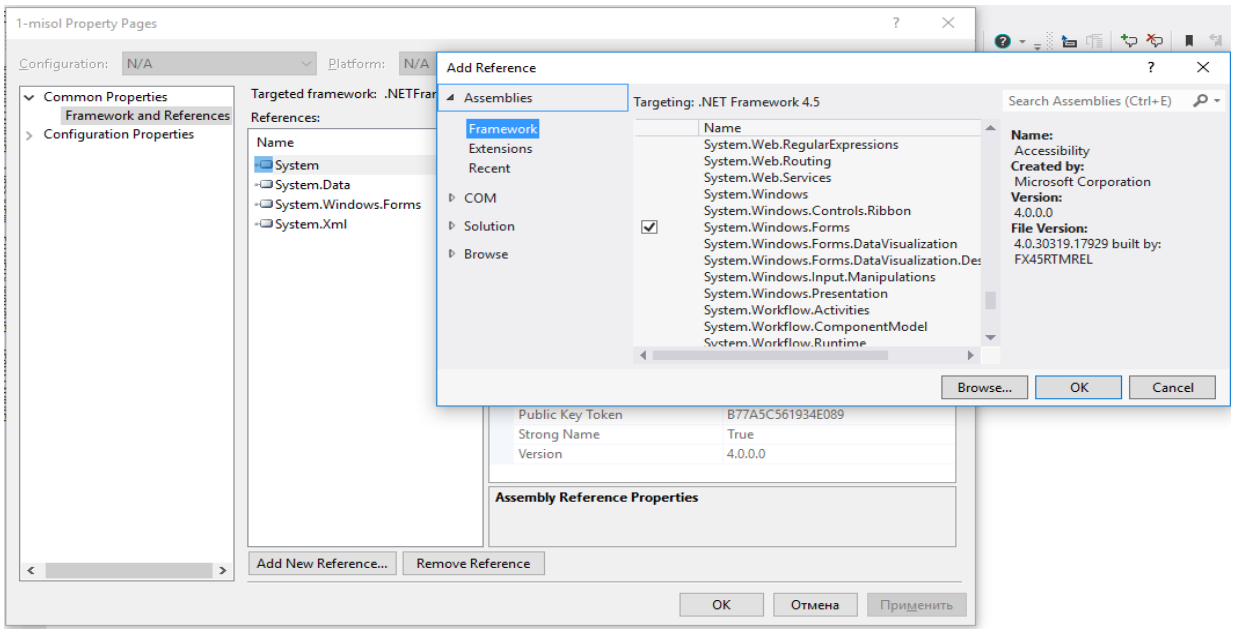
Belgi (simvol) bo‘lib xizmat qiluvchi funksiya tugaganida va hisoblashlar natijasi ekranga chiqqanidan so‘ng, **Beep** ovozli signalni beramiz. Dasturning oxirgi **Console::ReadKey();** qatori, foydalanuvchi tomonidan ixtiyoriy klaviatura bosilganda dastur to‘xtatilishini bildiradi. Agar dasturda bu 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**).



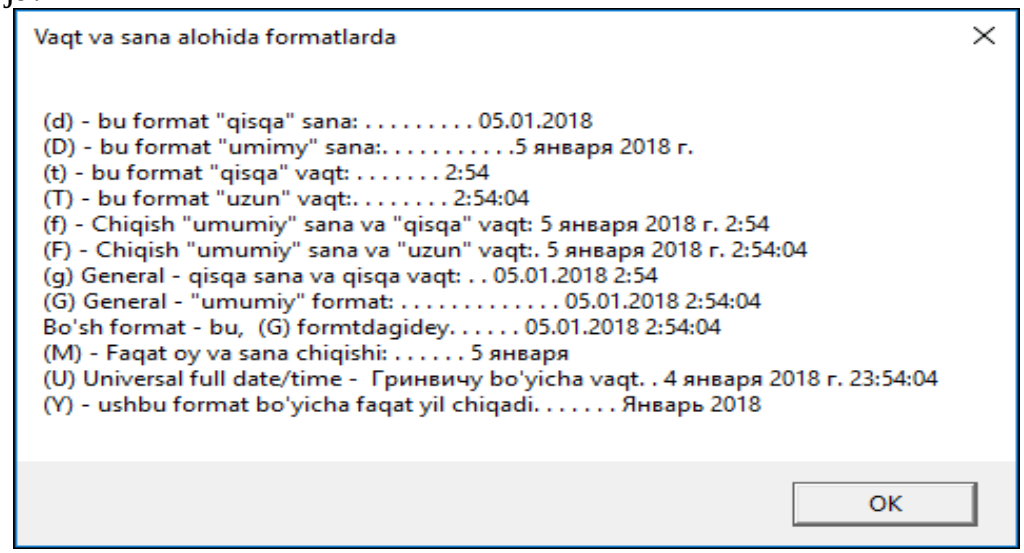
2.10- rasm. Add reference oynasi

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",
```

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

Natija:



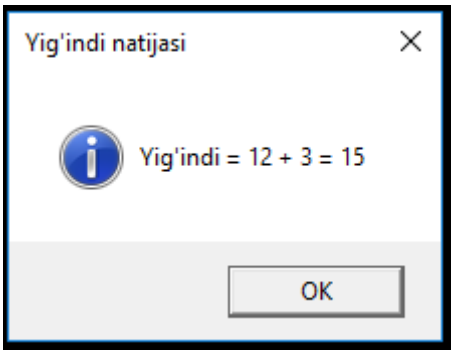
2.11- rasm. Natijalar oynasi

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:

```
#include "stdafx.h"
#include "conio.h"
using namespace System;
using namespace Microsoft::VisualBasic;
int main(array<System::String ^> ^args){
String^ satr;
for (; ; ){
satr = Interaction::
InputBox("Birinchi soni kiriting:", "Ikki son", "", 100, 100);
if (Information::IsNumeric(satr) == true) break;}
Single X = Single::Parse(satr); for (; ; ){
satr = Interaction::
InputBox("Ikkinchi soni kiriting:", "Ikkita son", "", 100, 100);
if (Information::IsNumeric(satr) == true) break; }
Single Y = Single::Parse(satr);
Single Z = X + Y;
satr = String::Format("Yig'indi = {0} + {1} = {2}", X, Y, Z);
// Natijani ekranga chiqarish
Interaction::MsgBox(satr, MsgBoxStyle::Information, "Yig'indi
natijasi"); _getch(); }
```

Natija:



2.12- rasm. MassegaBox natijalar oynasi

C++ dasturlash tili funksiyalar majmuidan iborat bo‘lganligi uchun **CLR Console Application** punkti orqali funksiyalarni tashkil etishda ishlatiladigan operatorlar va ba`zi bir tushunchalarga to‘xtalib o‘tamiz.

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

1- usul:

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

Dastur natijasi:
12
4
C=16

2- usul:

```
// 2-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
// o‘zgaruvchilarga qiymat kiritish
cin>>a; cin>>b;
// hisoblash
int c=a+b;
//  C o‘zgaruvchini qiymatini chiqarish
cout<<"C= "<<c<<endl; getch(); }
```

Dastur natijasi:
12
4
C=16

3- usul:

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

Dastur natijasi:
12
4
C=16

- Operatorlarni qo‘llanilishi quyidagicha bo‘ladi:

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

```
1. // 4-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. cin>>a>>b;//o‘zgaruvchilarga qiymat kiritish
9. // if shart operatorini qo‘llanilishi
10.if(a>b){cout<<"max="<<a<<endl;}
11.else {cout<<"max="<<b<<endl;} getch(); }
```

Dastur natijasi:
12
4
max= 12

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

```
1. // 5-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; // o‘zgaruvchilarni e`lon qilish
7. // o‘zgaruvchilarga qiymat kiritish
8. a= Convert::ToInt32( Console::ReadLine());
9. switch (a){
10.case 1: Console::WriteLine("Dushanba"); break;
11.case 2: Console::WriteLine("Seshanba"); break;
12.default: Console::WriteLine("Xatolik"); break;}
13.getch(); }
```

Dastur natijasi:
2
Seshanba

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

```
1. // 6-misol.cpp : main project file.
2. #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(); }
```

Dastur natijasi:
12
a 10 dan katta

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();}
```

Dastur natijasi:
4
1
2
3
4
Yig’indi= 10

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(); }
```

Dastur natijasi:
4
1
2
3
4
Yig’indi= 10

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) {
```

Dastur natijasi:
4
1
2
3
4
Yig’indi= 10

```
// '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();
}
```

```
Dastur natijasi:
4
1
-4
77
-2
```

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();
}
```

```
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
```

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
```

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

10- *funksiya va uning qo‘llanilishi:*

```
1. // 13-misol.cpp : main project file.
2. #include "stdafx.h"
3. #include "conio.h"
4. using namespace System; int s;
5. float mas_sum(int *A, int n){
6. for(int i=0;i<n;i++){
7. s+=A[i];} return s; }
8. void mas_in(int *A, int n){
9. for(int i=0;i<n;i++){
10. A[i]=Convert::ToInt32(
11. Console::ReadLine());} }
12. int main(array<System::String ^> ^args) {
13. int n,*A = new int[100];
14. n= Convert::ToInt32( Console::ReadLine());
15. mas_in(A,n);
16. Console::WriteLine("Summa= "+mas_sum(A,n));
17. getch();
18. }
```

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

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

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

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

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

```
1. // 14-misol.cpp : main project file.
2. #include "stdafx.h"
3. #include "conio.h"
4. #include "iostream"
5. using namespace std; int s;
```

```
Dastur natijas
2
1 - talaba
Ismi= Oybek
Fam= Mallayev
Yoshi= 20
Stipendiyasi= 550000
2 - talaba
Ismi= Akbar
Fam= 19
```

```
struct talaba
{  char ism[20];
char fam[20];
int yosh;
float stipendiya; };
int main()
{ int n; talaba A[100];
cin>>n;
for(int i=0; i<n;i++){
cout<<i+1<<" - talaba"<<endl;
cout<<"Ismi= ";cin>>A[i].ism;
cout<<"Fam= ";cin>>A[i].fam;
cout<<"Yoshi= ";cin>>A[i].yosh;
cout<<"Stipendiyasi= ";
cin>>A[i].stipendiya;}
for(int i=0; i<n;i++){
cout<<i+1<<" - talaba"<<endl;
cout<<"Ismi= "<<A[i].ism<<endl;
cout<<"Fam= "<<A[i].fam<<endl;
cout<<"Yoshi= "<<A[i].yosh<<endl;
cout<<"Stipendiyasi= "<<A[i].stipendiya<<endl;
} getch(); }
```

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

```
// 15-misol.cpp : main project file.
#include "stdafx.h"
#include "conio.h"
#include "iostream"
using namespace std; int s;
class base {
int i, j;
public:
void set(int a, int b) { i=a; j=b; }
void show() { cout << i << " " << j << "g`n"; } };
class derived : public base { int k;
public:
derived(int x) { k=x; }
void showk() { cout << k << "g`n"; } };
int main(){
derived ob(3);
ob.set(1, 2); // asos sinf a'zosiga ruxsat ochiq
ob.show(); // asos sinf a'zosiga ruxsat ochiq
ob.showk(); // voris sinf a'zosidan foydalanish
getch(); }
```

```
1 2
3
```

2.3. Visual C++ muhitida strukturalar bilan ishlash

Struktura – bu ma`lumotlarni bir butun nomlangan elementlar

to‘plamiga birlashtirish. Struktura elementlari (maydonlar) har xil tipda 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. Struktura tipidagi o‘zgaruvchi quyidagicha ta`riflanadi:

<struktura_nomi > <o‘zgaruvchi>;

Struktura tipidagi o‘zgaruvchi ta`riflanganda inisializasiya qilinishi mumkin: **<struktura_nomi > <o‘zgaruvchi>=<inisializator>;**

Strukturani inisializasiyalash uchun uning elementlar qiymatlarini 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‘zgaruvchilar uchun o‘zlashtirish operasiyasi aniqlangan. Bunda har bir elementdan nusxa olinadi. Masalan:

Student ss=s;

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

<Struktura_nomi>.<element_nomi>

Masalan:

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

employee.rate – 10000 qiymatga ega bo‘lgan butun tipdagi o‘zgaruvchi.

Strukturaga ko‘rsatkichlar. Strukturaga ko‘rsatkichlar oddiy 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 birinchi indeksi chegarasi ko‘rsatilishi shart emas, lekin qolgan indekslar 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 ketma ketligidir. Satrli konstanta oxiriga avtomatik ravishda satr ko‘chirish ‘\n’ simvoli qo‘shiladi. Satr qiymati simvolli konstanta bo‘lgan simvolli 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 sifatida. Funksiyalarda massivlar argument sifatida ishlatilganda, ularning birinchi indeksi chegarasini ko‘rsatish shart emas, qolganlarini chegarasini ko‘rsatish shart. Massivlar ilova bo‘yicha uzatiladi, ya`ni 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 yakunlanadi. 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 qilish, kompilyatorga Mushuk sinfini mavjudligini, hamda unda qanday qiymatlar saqlashi mumkinligi (**intYosh**, **intsOgirlik***)* va u qanday amallarni bajarishi mumkinligi (Miyovlash() usuli) haqida xabar beradi. Bundan tashqari, bu e`lon qilish orqali kompilyatorga **Mushuk** sinfining o‘lchami, ya`ni har bir **Mushuk** sinfi ob`ekti uchun kompilyator qancha joy ajratishi lozimligi haqida ham ma`lumot beradi. Masalan, joriy misolda butun qiymat uchun to‘rt bayt talab qilinsa, **Mushuk** sinfi ob`ekti o‘lchovi sakkiz bayt bo‘ladi. (**intYosh** o‘zgaruvchisi uchun to‘rt bayt, **intsOgirlik** o‘zgaruvchisi uchun to‘rt bayt). **Miyovlash**() usuli xotiradan joy ajratishni talab qilmaydi.

Ob`ektni e`lon qilish

Yangi turdagi ob`ekt xuddi oddiy butun sonli o‘zgaruvchidek aniqlanadi. Haqiqatan ham ixtiyoriy butun sonli o‘zgaruvchi quyidagicha aniqlanadi:

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

Mushuk Frisky //Mushuk ob`ektini aniqlaymiz.

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

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

Sinf a`zolariga murojaat qilish imkoni

Mushuk sinfining real ob`ektini aniqlaganimizdan so‘ng bu ob`ektning a`zolariga murojaat qilish zaruriyati tug‘ilishi mumkin. Buning uchun bevosita murojaat (.) operatori qo‘llaniladi. Masalan, **Frisky** ob`ektining **Weight** o‘zgaruvchi - a`zosiga 50 sonini 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 uchun bu ishni kompilyatorning o‘zi bajaradi. Standart konstruktor va destruktorlar birorta argument qabul qilmaydi va hech qanday amal 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, u buni yangidan ixtiro qilmaydi, omborga (magazinga) borib, mos parametrlarga muvofiq kerakli detalni tanlaydi. Bu holda muhandis joriy rezistor qanday tuzilganligiga e`tiborini qaratmaydi, rezistor faqatgina zavod xarakteristikalariga muvofiq ishlasa yetarlidir. Aynan, shu tashqi konstruksiyada qo‘llaniladigan yashirinlik yoki ob`ektni yashirinligi yoki avtonomligi xossasi inkapsulyatsiya deyiladi. Inkapsulyatsiya yordamida berilganlarni yashirish ta`minlanadi. Bu juda yaxshi xarakteristika bo‘lib foydalanuvchi o‘zi ishlatayotgan ob`ektning ichki ishlari haqida umuman o‘ylamaydi. Haqiqatan ham, xolodilnikni

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 maydonlari ham, Miyovlash () usuli ham yopiq a`zo sifatida aniqlanadi. Dasturda yuqoridagi tartibda Mushuk sinfi e`lon qilingan bo‘lsa va bu sinf ekzemplyari bo‘lgan ob`ektning itsYosh a`zosiga main() funksiyaci tanasidan turib murojaat qilsak, kompilyator xatolik ro‘y berganligi haqida xabar beradi.

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

Statik elementlar hamda funksiyalar

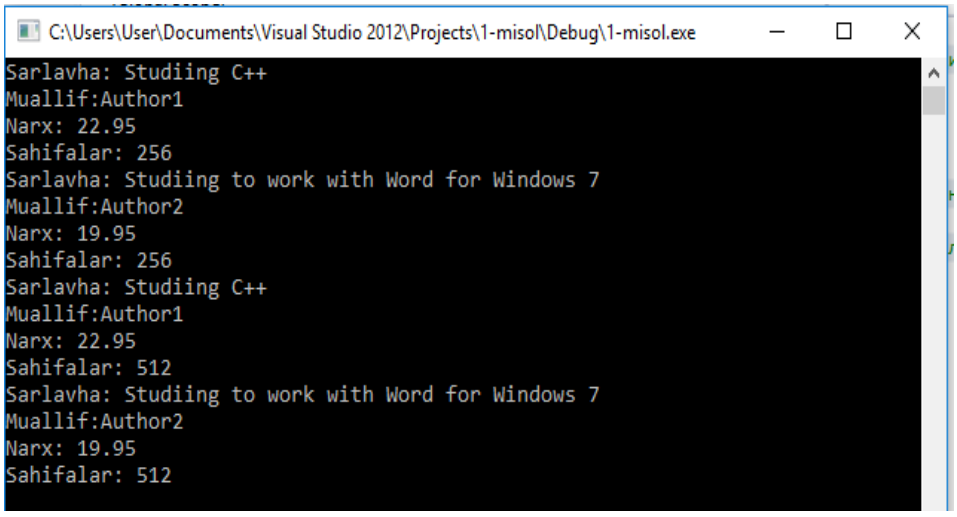
Shu paytgacha, har bir yaratilgan element o‘zining xususiy ma`lumotlar elementiga ega bo‘lar edi. Lekin, shunday holat bo‘ladiki, bitta sinf doirasidagi ob`ektlarning ba`zi elementlari o‘zaro bog‘langan bo‘ladi. Masalan, ish vaqti bir xil bo‘lgan 1000 ta ishchining oylik maoshini hisoblaydigan dastur tuzish taklif qilinayotgan bo‘lsin. Soliq stavkasini aniqlash uchun dastur har bir ishchining sharoitini bilishi kerak. Buning uchun aytaylik, state_of_employee nomli sinfdan foydalanamiz. Agar, ishchilar bir xil sharoitda ishlasa, demak, dastur barcha employee tipidagi ob`ektlar uchun (barcha ishchilar uchun) ushbu elementlardan o‘zaro moslikda foydalanadi. Ushbu holatda dastur, bitta axborotning 999 ta nusxasidan foydalanish bilan xotiradan foydalanish hajmini kamaytiradi.

Sinfning elementidan o‘zaro moslikda foydalanish uchun, ushbu element **static** (statik) deb e`lon qilinishi zarur. Agar, dastur ushbu elementga yangi qiymat o‘zlashtirsa, hamma ob`ekt elementi ushbu yangi qiymatni qabul qiladi. Sinf elementi statik deb e`lon qilinganidan 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:

```
#include "stdafx.h"
#include <string.h> //strcpy() uchun
#include <stdio.h> //printf() uchun
#include <conio.h> //_getch() uchun
using namespace std;
class book_series{
book_series();
public:
static void show_book(void); /*Funksiyani statis elementini chop etish uchun, ushbu atribut qo‘shiladi*/
static int page_count;
private:
char title [64];
char author[64];
float price; };
int book_series::page_count; /*O‘zgaruvchini global o‘zgaruvchi sifatida e`lon qilish*/
void book_series::show_book (void){
printf("Sahifalar soni=%d\n",page_count); }
int main(void){
book_series::page_count = 256;
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 qilishga zarurat tug‘iladi?
4. Sinf elementini qachon **public va static** ko‘rinishida e`lon qilishga zarurat tug‘iladi?
5. **private** ko‘rinishida himoyalangan elementga **int main()** funksiya

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:

```
#include "stdafx.h"
#include <string.h> //strcpy() uchun
#include <iostream> //cout uchun
#include <conio.h> //_getch() uchun
using namespace std;
class Parent{
public:
double F1(double x){ return x*x;}
 double F2(double x){
 return F1(x)/2; } };
class Child: public Parent
{ public:
double F1(double x){
return x*x*x; } };
int main() {
Child child;
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 aniqlaydi hamda funksiyalarni virtuallash vositasida ularning polimorf xulq-atvorini ta`minlaydi. Bazaviy va hosilaviy sinflarda virtual funksiyalarni 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:

```
#include "stdafx.h"
#include <string.h> //strcpy() uchun
#include <iostream> //cout uchun
#include <conio.h> //_getch() uchun
using namespace std;
class Parent{
public:
virtual double F1(double x){
return x*x; }
double F2(double x){
return F1(x)/2;} };
class Child:public Parent{
public:
virtual double F1(double x){
return x*x*x;} };
int main() {
Child child;
cout<<child.F2(3)<<endl;
_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. Sizning telefonga tuzilayotgan dasturingiz uchun siz **phone** bazaviy sinfiga ko‘rsatkich tuzasiz:

phone *poly_phone;

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

poly_phone=(phone*)&home_phone;

Qiymat berish operatoridan keyin keladigan (**phone***) belgilari turlarga keltirish operatori bo‘lib, bu operator C++ning kompilyatoriga hamma narsa joyida, bir turdagi o‘zgaruvchi (**touch_tone**) adresini boshqa turdagi o‘zgaruvchi (phone) ga berish zarurligini ma`lum qiladi. Dastur poly_phone ob`ekti ko‘rsatkichiga turli ob`ektlar adresini taqdim qilishi mumkin ekan, u holda bu ob`ekt ham o‘z shaklini o‘zgartirishi, 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 shaklini o‘zgartirib turar ekan, u polimorf bo‘ladi.

Do‘stona funksiyalar

Do‘stona funksiyalar, garchi ular biror bir sinfga mansub bo‘lmasalarda, tashqi sinf ma`lumotlarining barcha **private** va himoyalangan a`zolariga kirish huquqiga ega bo‘ladilar. Do‘stona funksiyalarning e`lon qilinish sintaksisini qaytarilayotgan tur ko‘rsatkichi oldidan turgan **friend** kalit so‘zi yordamida ko‘rib 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 parametri — **this** ko‘rsatkichini uzatish hisobiga o‘z sinfining barcha ma`lumotlariga avtomatik tarzda kirish huquqiga ega bo‘lsa, do‘stona funksiyalar ushbu parametrning ochiq-oydin spesifikasiyasini talab qiladi.

Darhaqiqat, X sinfida e`lon qilingan F do‘stona funksiya bu sinfga mansub emas, demakki, x.F va xptr->F (bu yerda x - X sinfining nusxasi, xptr - uning ko‘rsatkichi) operatorlari tomonidan chaqirib olina olmaydi. Bu o‘rinda F(&x) yoki F(xptr) murojaatlari sintaktik jihatdan to‘g‘ri (**correct**) bo‘ladi. Shunday qilib, do‘stona funksiyalar sinfning a`zo-funksiyalari vositasida ishga tushirilishi noqulay, qiyin va xatto 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 har bir shakl uchun alohida Area() va Print() usullarini qo‘llash kerak. 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 yaratishda undan meros bo‘lgan ma`lumot – a`zolari asosiy (bazaviy) sinf konstruktori orqali inisializasiyalanishi lozim. Asosiy sinf konstruktori avtomatik ravishda chaqiriladi va hosila sinfni konstruktoridan oldin bajariladi. Asosiy (bazaviy) sinfni konstruktorining parametrlari hosila sinfni konstruktorini aniqlashda ko‘rsatiladi. Shunday qilib argumentlarni hosila sinfni konstruktoridan asosiy (bazaviy) sinfni 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: avvalo asosiy(bazaviy), keyin esa komponent – ob`ektlar (agarda ular mavjud bo‘lsa), undan keyin esa hosila sinfning o‘zi. SHunday qilib, hosila sinfning ob`ekti quyi ob`ekt sifatida asosiy (bazaviy) sinf ob`ektini o‘z ichiga oladi. Ob`ektlar teskari tartibda o‘chiriladi: avvalo hosila, keyin uning komponent – ob`ektlari, undan keyin esa asosiy(bazaviy) ob`ekt.

Shunday qilib, ob`ektni o‘chirish tartibi uning konstruktorlash 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 boshqa sinflarni aniqlashda ishlatilsa, baza ro‘yxatidan chiqariladi. Ba`zi hollarda A sinf B sinfning bazasini ifodalasa va C uchun B baza bor bo‘lsa, u holda B sinf C uchun to‘g‘ridan-to‘g‘ri baza hisoblanadi, natijada A sinf C sinf uchun to‘g‘ri bo‘lmagan baza bo‘lib hisoblanadi.

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 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 ularni qo‘llanilishi, maxsus String turi va u bilan ishlash shartlari, statik va dinamik massivlar bilan ishlash, fayllar bilan ishlovchi maxsus (IO) kutubxonasi fuksiyalari va ular yordamida turli xil kengaytmali fayllar bilan ishlash, sinf (class) va struktura (struct) e`lon qilish turlari, ularning turli hil ko‘rinishlari, qo‘llanilish usullari, satrlar bilash ishlovchi maxsus funksiyalar haqida 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 tushuntirib 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.

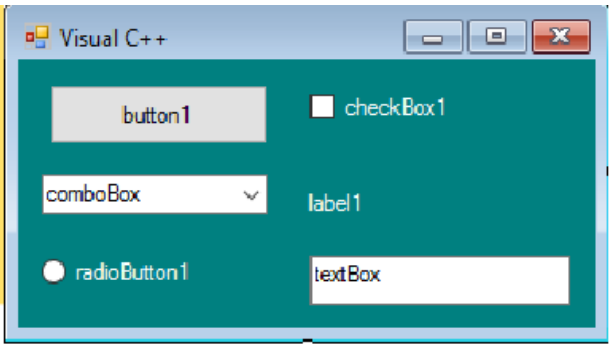
```
#include <iostream>
using namespace std;
class misol1{
private: int a;
int b;
public: void set(int a, int b);
public: void get();
};
void misol1::set(int a, int b){a=a; b=b;}
void misol1::get(){cout<<”a= ”<<a<<endl<<”b= ”<<b<<endl;}
int main()
{  misol1 A; int a=2, b=4;
A.set(a,b); A.get();
return 0;
}
```

16. Ushbu strukturaning vazifasini ayting.

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

3- B O B

Visual C++ ning Windows Application muhiti va unda ishlash



Bobning maqsadi

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

diagrammalar yaratish.

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.

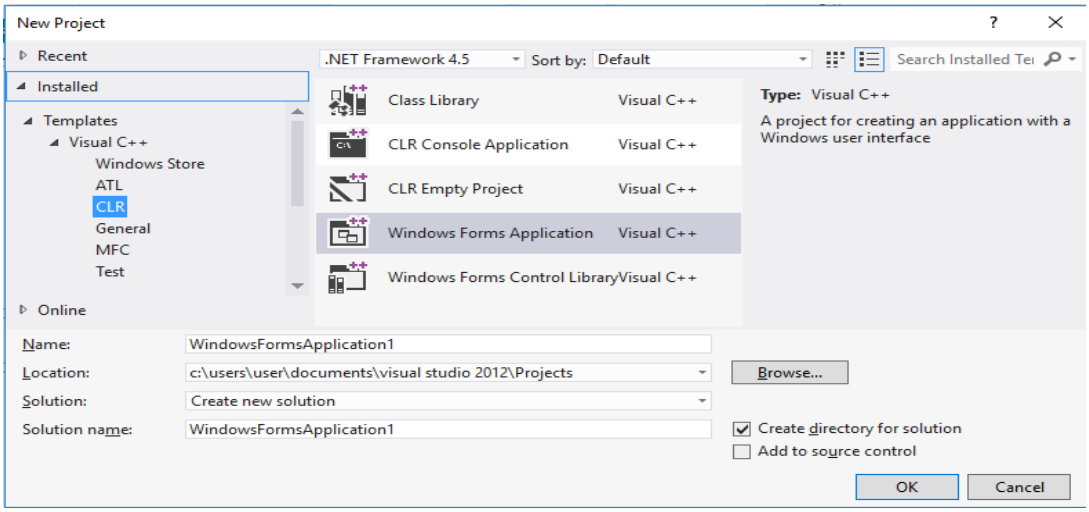
3.23. Visual C++ ning Windows Application muhitida komponentalarning joylashish vaziyatlarini nazorat qilish.

3.1. Windows Form Application muhitini yaratish va yordamchi oynalarini sozlash

Windows Form Application muhiti paketini o‘rnatish

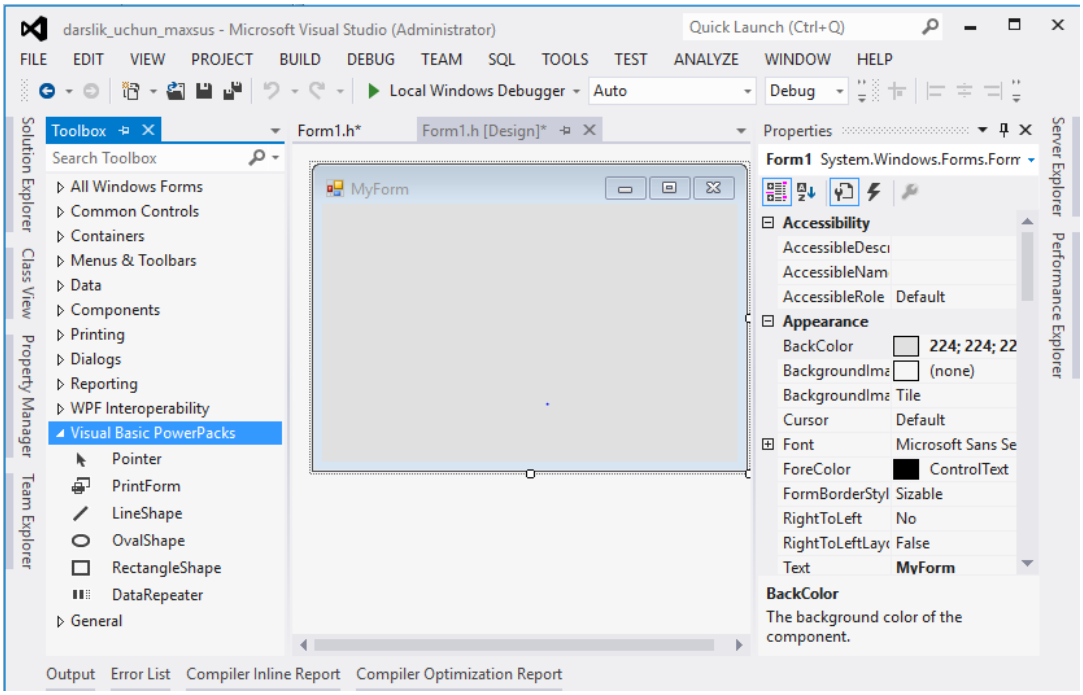
Visual Studio 2005 dan yuqori versiyalarida **Windows Form Application** joylashtirilmagan. Shuning uchun **Visual Studio 2012** dasturni tizimga o‘rnatgandan keyin **Windows Form Application** paketini quyidagi internet manzidan yuklab olinadi: http://www.outme.ru/visual-c-2012-sozdanie-formyi.html.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

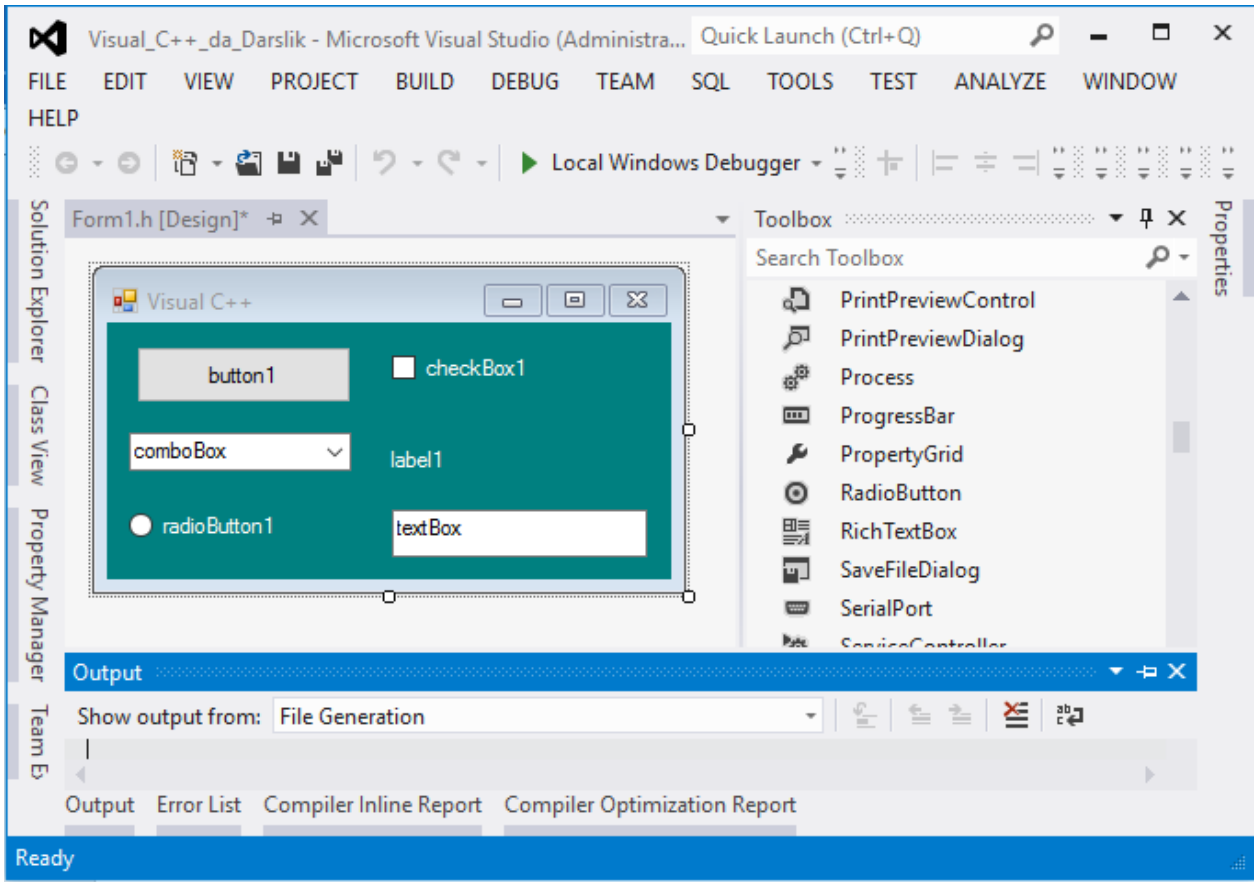


3.2- rasm. Windows Form Aplication ilova oynasi

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.

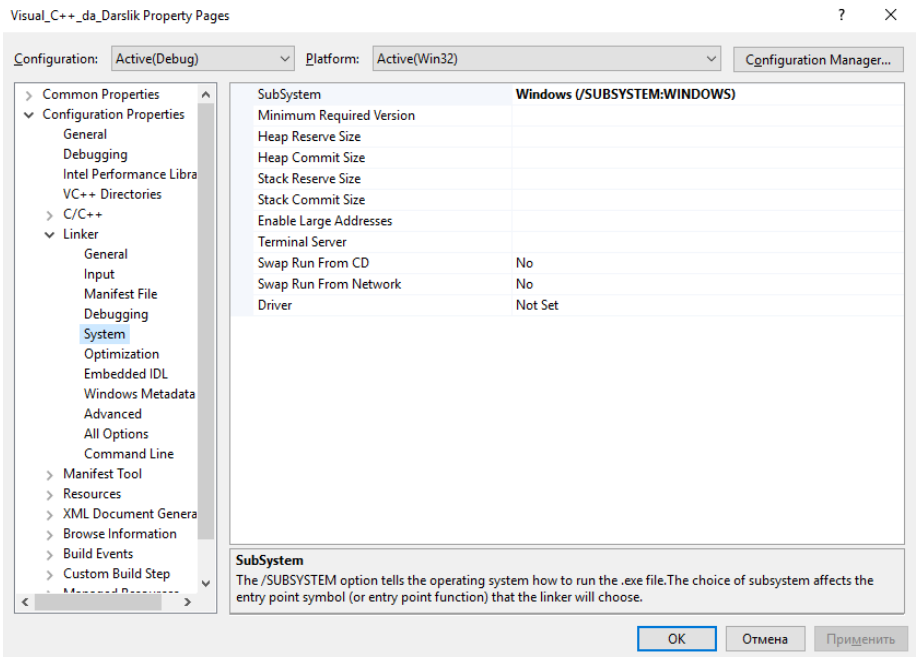


3.3- rasm. Foydalanuvchi interfeysini sozlash oynasi

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

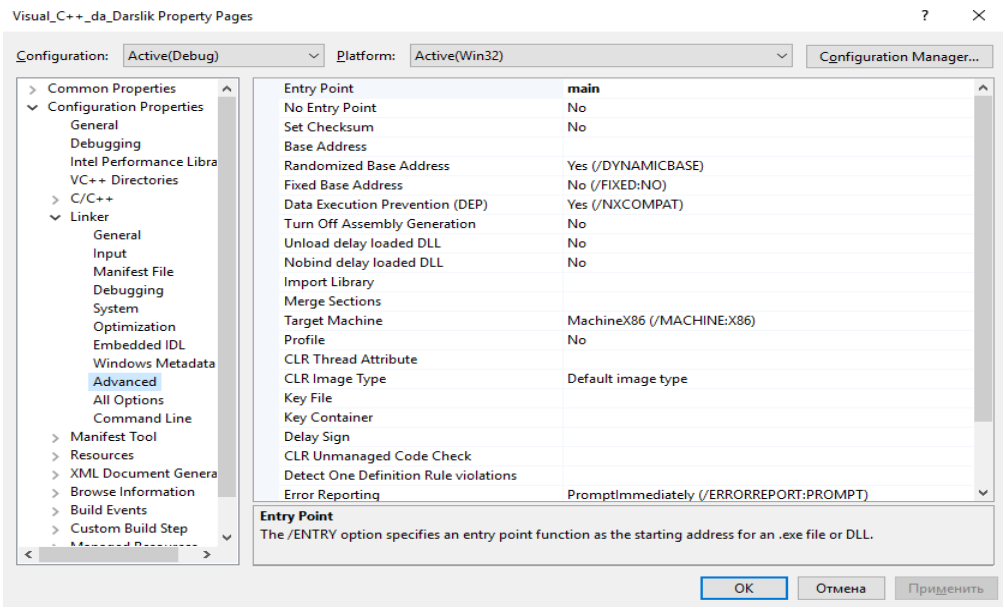
```
#include”MyForm.h”
using namespace System;
using namespace System::Windows::Forms;
[STAThread]
void main(array<String^>^ args){
Application::EnabledVisualStyles();
 Application::SetCompatibleTextRenderingDefault(false);
Project1::MyForm form;
Application::Run(%form); }
```

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.



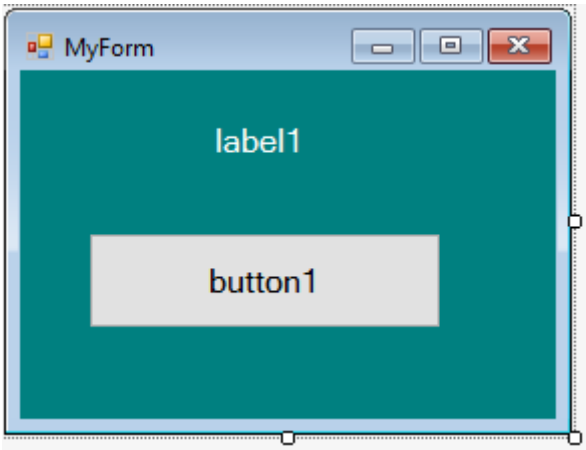
3.5- rasm. Formani sozlash oynasi

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

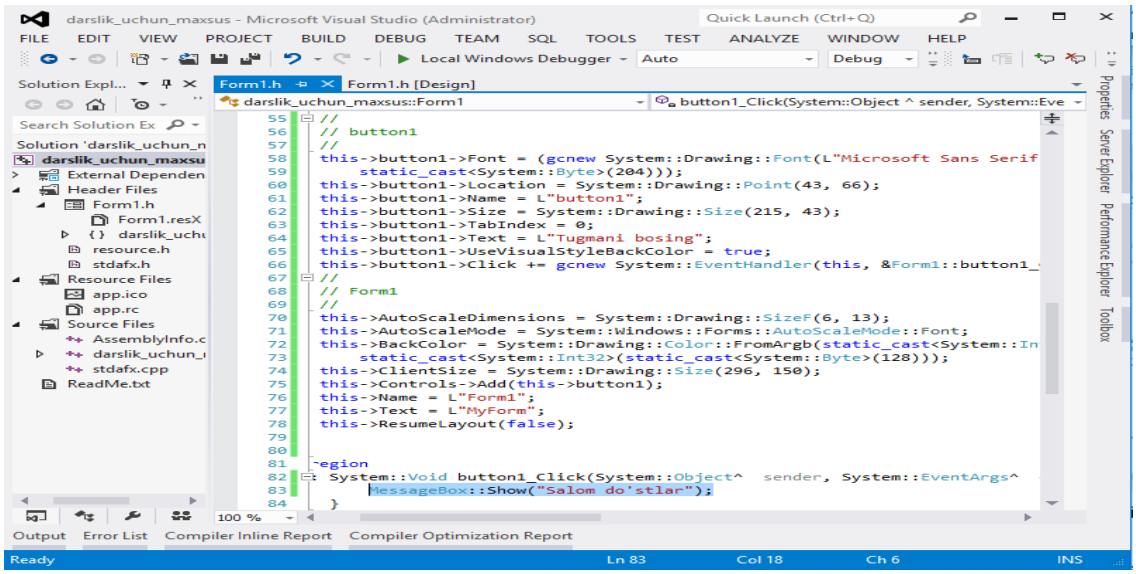
komponentni ToolBox panelidan sichqoncha yordamida tashlab tayyor vizual ob`ekt hosil qilinadi. Bunda har bir obyekt xossalarga (properties) ega ekanligini unutmaslik lozim. Masalan tugmaning xossalari quyidagilar: Tugma nomi (**Name**) – **button1,** tugma ustidagi matn (Text), tugmaning X,Y koordinata 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 button1 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){}

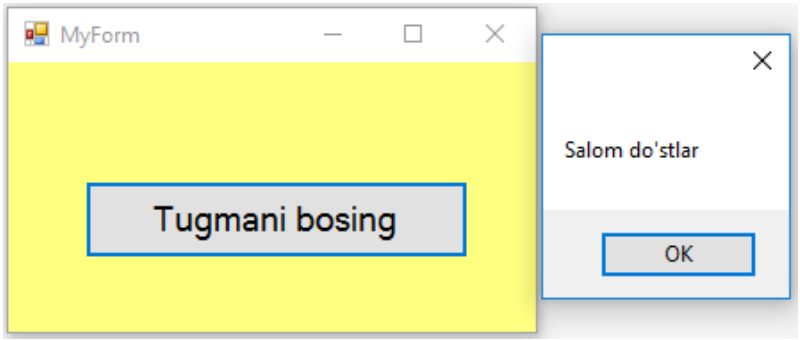


3.7- rasm. Dastur kodi qo‘yilmasi

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 Project1.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 xabarlar oynasiga chiqaring.
3. Button tugmasining Text xossasiga tugma bir marta bosilsa 1 raqamini, 2 marta bosilsa 2 raqamini chiqaring.
4. Button tugmasini har bosganda tugmaning rangini o‘zgartiring.
5. Button tugmasining Text xossasiga massivning elementlarini 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 tipdagai ma`lumotlarni chiqaradi?

3.3. MouseHower hodisasi

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

Bunda MyForm.h dastur kodi qo‘yilmasiga o‘tish amalga oshiriladi va uchta bosh hodisalarga ishlov beruvchilar hosil bo‘ladi. Masalan oxirgi hodisaga ishlov beruvchi quyidagi ko‘rinishda bo‘ladi:

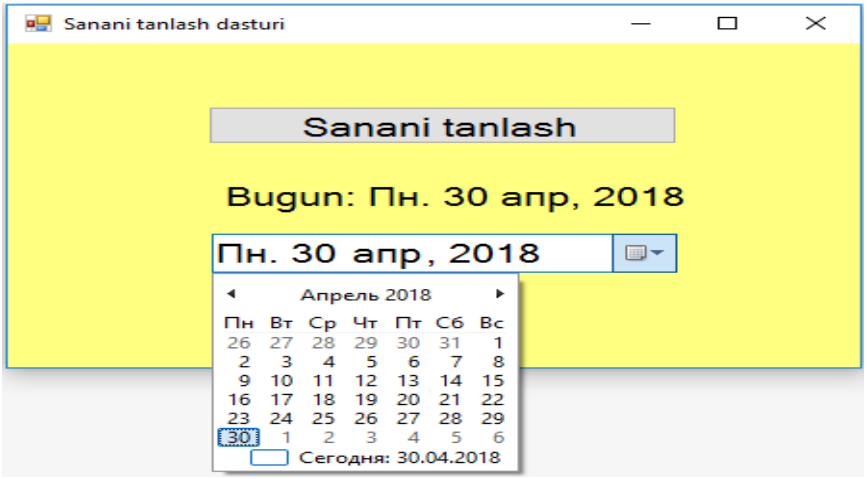
```
private:      System::Void      label1_MouseHover(System::Object^sender,
System::EventArgs^  e){}
```
Figurali qavslar orasiga muloqat oynasini chaqirish kodi joylashtiriladi. `MessageBox::Show("Salom do'stlar");`

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 kodi oynasiga o‘tiladi. Xuddi shunday usul bilan Button tugmasini hodisasi yaratiladi. Bundan tashqari ValueChanged hodisasi bilan ishlash zarurati yuzaga keladi. Uni hosil qilish uchun DataTimePicker ning ValueChanged hodisasida sichqoncha chap tugmasi ikki marta bosiladi. 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 aks ettirishning kerakli formati beriladi: birinchi 3 ta d harflari hafta kunini, 3 ta MMM harflari qisqa shaklda kun va oy nomini hamda 4 ta y harflari yilni anglatadi. Sanani o‘zgartiruvchi ValueChanged hodisasi bilan ishlashda label1 matnli nishonga tanlangan sana qiymati o‘zlashtiriladi. Bunda String::Format usulidan foydalanamiz. “Tanlangan sana :{0}” – foydalaniladigan format qiymatini anglatadi. Sanani tanlash hodisasi bilan ishlashda dateTimePicker1 boshqaruv elementiga fokus beriladi.

Kalendarni ochishda F4 klavishidan foydalanish mumkin. Bu vaziyatda Send usuli faol ilovaga mos klavish bosilishi haqida xabar yuboradi.

TextBox komponentasi va uning xossalari

Forma bilan ishlash jarayonida ko‘p hollarda berilganlarni TextBox matnli maydon orqali kiritish tashkil qilinadi. Misol tariqasida matnli maydon orqali raqam kiritiladigan, buyruq tugmasi bosilganda uning kvadrat ildizini hisoblovchi va natijani Label nishoniga chiqaradigan oddiy dasturni tuzishni ko‘rib chiqamiz. Agar raqam bo‘lmagan belgi kiritilsa, foydalanuvchiga bu haqida xabar berilsin.

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 nom beriladi va OK tugmasi bosiladi. Toolbox dizayneri panelidan formaga TextBox matn maydoni, label metrikasi va Button buyruq tugmasi joylashtiriladi. Ular ekran formatida 3.10-rasmda ko‘rsatilagandek joylashtiriladi. Formani sichqoncha chap tugmasini ikki marta bosilib, dasturning kodlar oynasida quyidagi kodlar yoziladi:

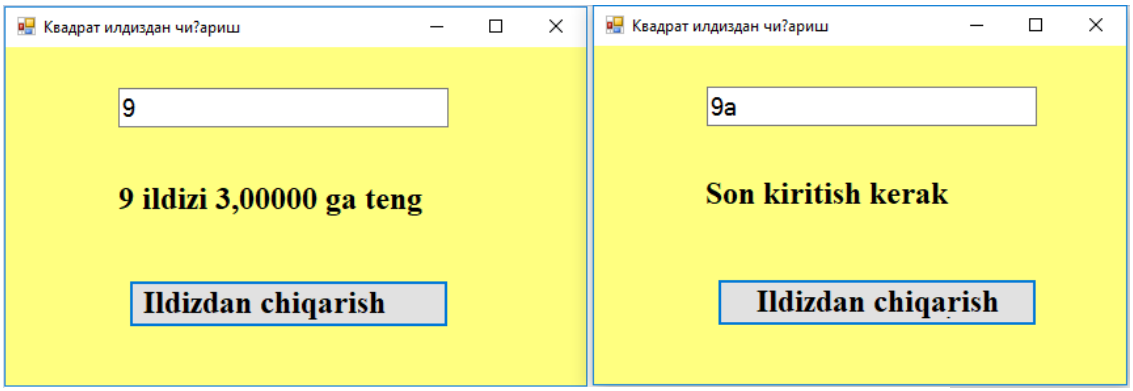
```
1. // Formani yuklash hodisasini hosil qilish
2. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {
3. this->Text="Kvadrat ildizdan chiqarish";
4. button1->Text = "ildizdan chiqarish";
5. label1->Text = String::Empty;
6. textBox1->Clear(); }
```

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

```
1. private: System::Void button1_Click_1(System::Object^  sender,
2. System::EventArgs^  e) { Single X;
3. // mantiqiy son o‘zgaruvchisiga faqat raqam o‘zlashtirilganda
   rost qiymat qabul qiladi
4. bool Son = Single::TryParse(textBox1->Text,
5. System::Globalization::NumberStyles::Number,
   System::Globalization::
6. NumberFormatInfo::CurrentInfo,X); if(Son == false){
7. // agar Son o‘zgaruvchisi raqamdan boshqa belgilarni qabul qilsa
8. // Son kiritish kerak xabarini chiqarib dastur ishini to‘xtatadi.
9. label1->Text = "Son kiritish kerak";
10.     label1-> ForeColor = Color::Red;
11.     //  label1  mertikasi rangi qizilga o‘zgartirildi
12.     return; }
```

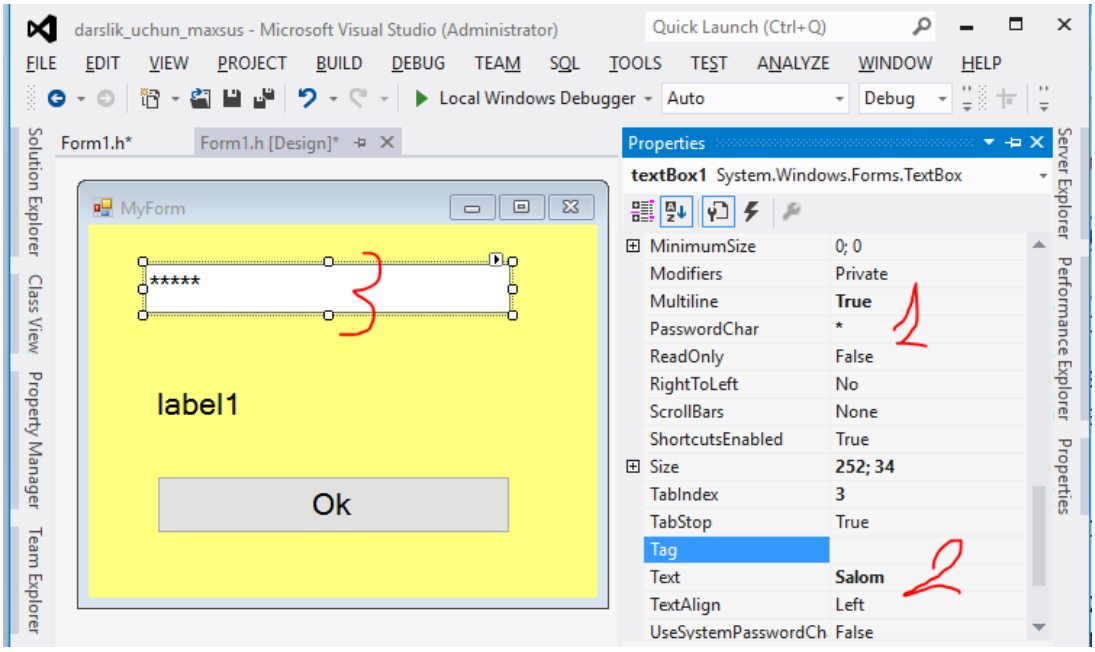
```
13.      Single Y = (Single)Math::Sqrt(X);
14.      label1-> ForeColor = Color::Black;
15. label1->Text = String::Format("{0}  ildizi {1:F5} ga teng ", X,Y
    ); }
```

TryParse usuli to‘rtinchi parametrik qayta ishlash natijasini qaytaradi(true yoki false).



3.10- rasm. Sondan ildiz chiqarish oynasi

Agar foydalanuvchi son kiritsa **Math::Sqrt(X)** kvadrat ildiz chiqarish operatori bajariladi. Visual Studio ning matematik funksiyalari **Math** sinfining usulidir. Qo‘llanilgan **"{0} ildizi {1:F5} ga teng "** format shuni anglatadiki, chiqarilayotgan nol elementni olish, ya`ni X o‘zgaruvchini fiksirlangan verguldan keyin beshta o‘nlik ko‘rinishida yozish.

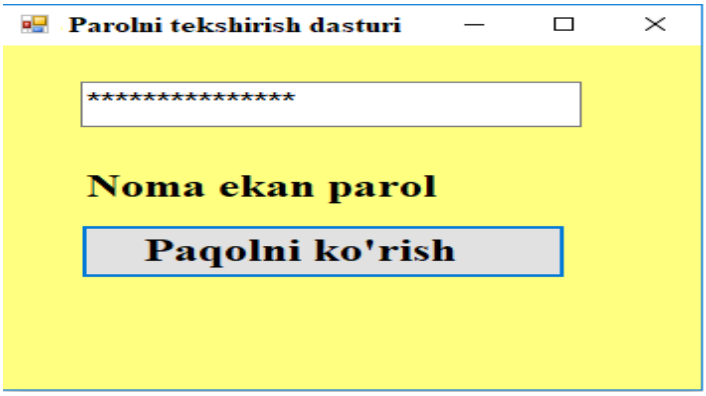


3.11- rasm. PasswordChar xossasini sozlash

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 shifirlangan ko‘rinishda chiqarish.

4. **TextBox** yordamida joriy parolni o‘zgartirish.

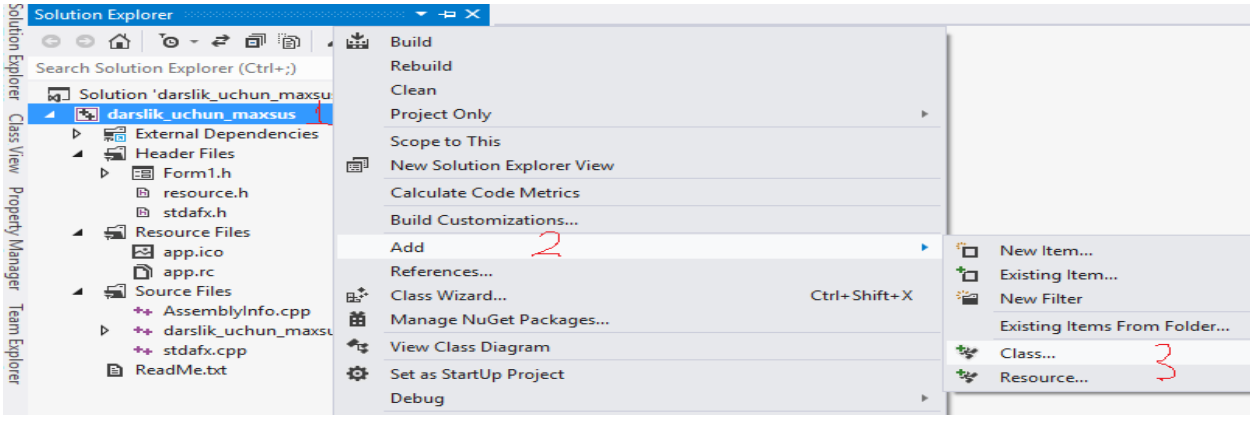
5. Parol 3 marta noto‘g‘ri terilgandan keyin TextBox ni ko‘rinmaydigan qilish.

Nazorat savollari:

1. Button komponentasining doubleClick hodisasining vazifasi nima?
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‘zgartirish mumkinmi?

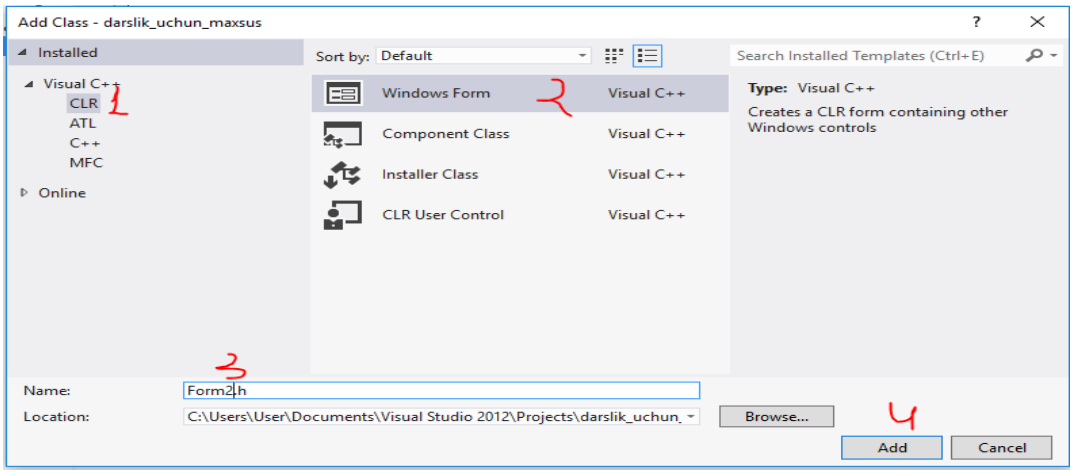
3.5. Forma (Form) lar bilan ishlash

Formalarni formalar bilan bog‘lovchi vizual dastur tuzish uchun quyidagi ketma - ketliklar amalga oshiriladi. **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 **label** metrikasi va **Button** buyruq tugmasi joylashtiriladi va formaga yangi forma qo‘shiladi(**3.13 va 3.14 - 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();      }
```

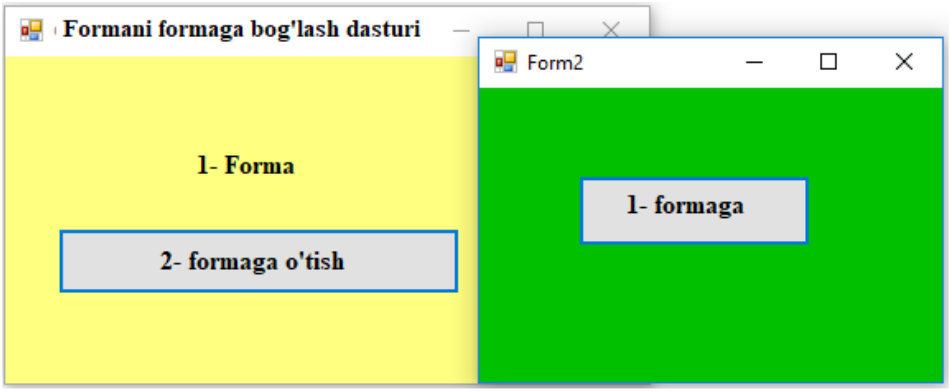
2-formadan 1- formaga qaytish uchun 2-formaning kodlar oynasiga quyidagi kodlar teriladi:

```
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:

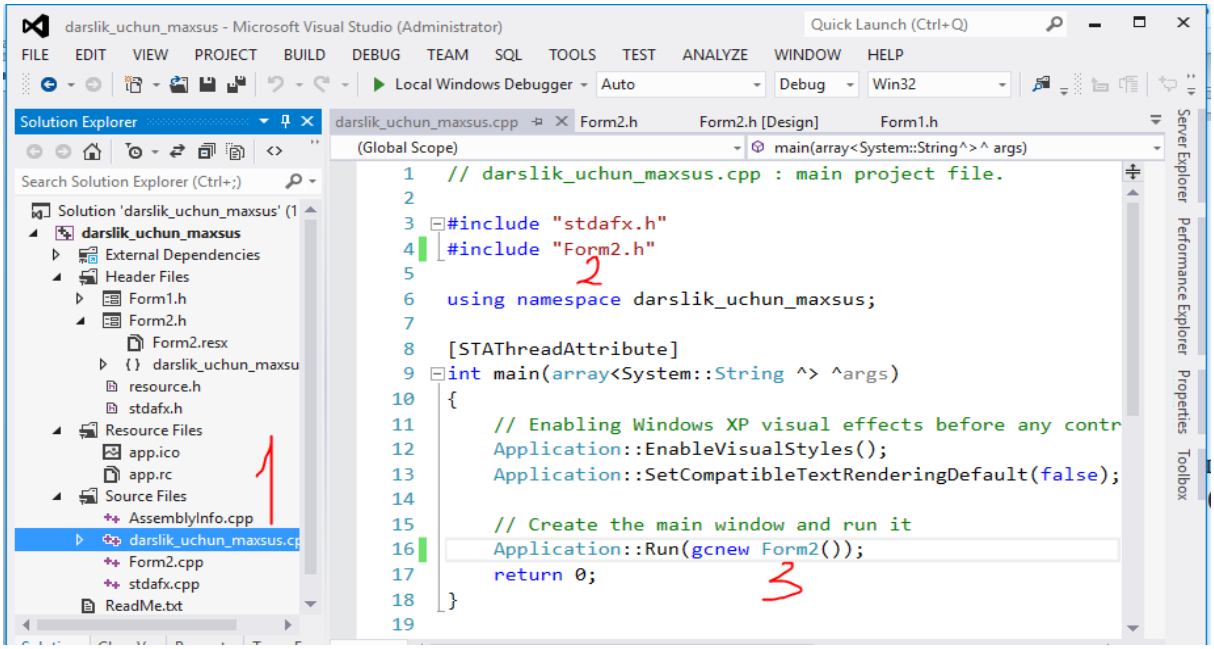
```
// Button buyruq tugmasini hodisasini hosil qilish
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^  e) { safe_cast<Form^>(Form1_ga)->Show();
3. this->Close();  }
```

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.

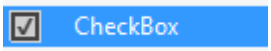
Nazorat savollari:

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?

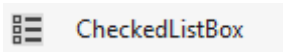
3.6. CheckBox, CheckedListBox, ComboBox, ListBox komponentalari va ularning xossalari

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

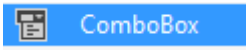
CheckBox, CheckedListBox va **ComboBox** komponentalari bir nechta variantlardan bittasini yoki bir nechtasini tanlash yo‘li bilan ishlovchi vizual dasturlar yaratish imkoniyatlarini beradi.

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=
```

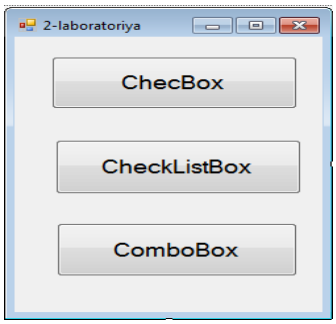
```
 checkedListBox1->CheckedIndices->GetEnumerator();     while( myEnum1-
 >MoveNext())    {
indexChecked[i] =  *safe_cast<Int32^>(myEnum1-> Current); i++; }
     }
index(indexChecked);  if ( indexChecked[0]==0 && indexChecked[1]==1)
{…}
     3) if(comboBox1->Text=="Tasodifiy sonlar bilan to'ldirish"){…}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

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

1) **1-formada** 3 ta button komponentalari joylashtiriladi va 1- button ning text xossasi **CheckBox** ga, 2- button ning text xossasi **CheckedListBox** ga, 3- buttonning text xossasi **ComboBox**ga o‘zgartiriladi. 1- buttonning **Click** hodisasiga Form2^ open=gcnew Form2(); this->Hide(); open->Show();, 2- button ning **Click** hodisasiga Form3^ open=gcnew Form3(); this->Hide(); open->Show();, 3- button ning **Click** hodisasiga Form31^ open=gcnew Form31(); this->Hide(); open->Show(); kodlari kiritiladi. Formalar bir biri bilan bog‘lanadi.
2) **2-formada** 1 ta **dataGridView,** 2 ta **TextBox,** 1 ta **button** va 6 ta **checkBox** komponentalari joylashtiriladi. **checkBox1** ning Text xossasi “Hisoblash” ga, **checkBox2** ning Text xossasi “Tasodifiy sonlar bilan to'ldirish” ga, **checkBox3** ning Text xossasi “DadaGridWiew dan kiritish” ga, **checkBox4** ning Text xossasi “Sariq rang”, **checkBox5** ning Text xossasi “Qizil rang” ga, **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 ta **CheckedListBox** komponentalarini joylashtiring. **CheckedListBox** ning Items xossasiga “Hisoblash”, “Tasodifiy sonlar bilan to'ldirish”, “DadaGridWiew dan kiritish”, “Sariq rang”, “Qizil rang”, “Yashil rang” iboralar kiritiladi va 2-jadvaldagi 3, 4 va 5- qadamlar bajariladi.
4) **4-formada** 1 ta **dataGridView,** 2 ta **TextBox,** 1 ta **button,** 4 ta **checkBox,** 1 ta **comboBox** komponentalarini joylashtiring. **checkBox1** ning Text xossasi “Hisoblash” ga, **checkBox4** ning Text xossasi “Sariq rang”, **checkBox5** ning Text xossasi “Qizil rang” ga, **checkBox6** ning Text xossasi “Yashil rang” ga o‘zgartiriladi. **comboBox** ning **Items** xossasiga “Hisoblash”, “Sariq rang”, “Qizil

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();
```

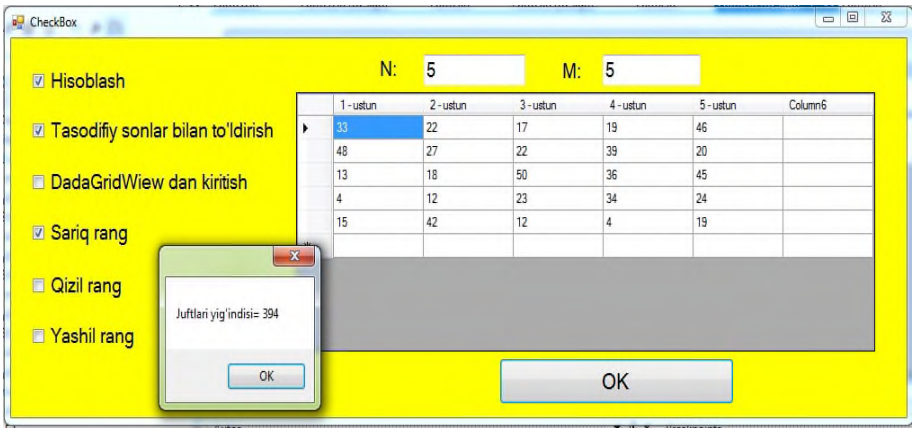
1-forma 3- formaga bog‘lanish uchun **ComboBox** tugmasining **Onclick** hodisasiga quyidagi kodlar kiritiladi:

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

2-qadam: 32 – variantdagi masalani **ye**chish 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());}
```

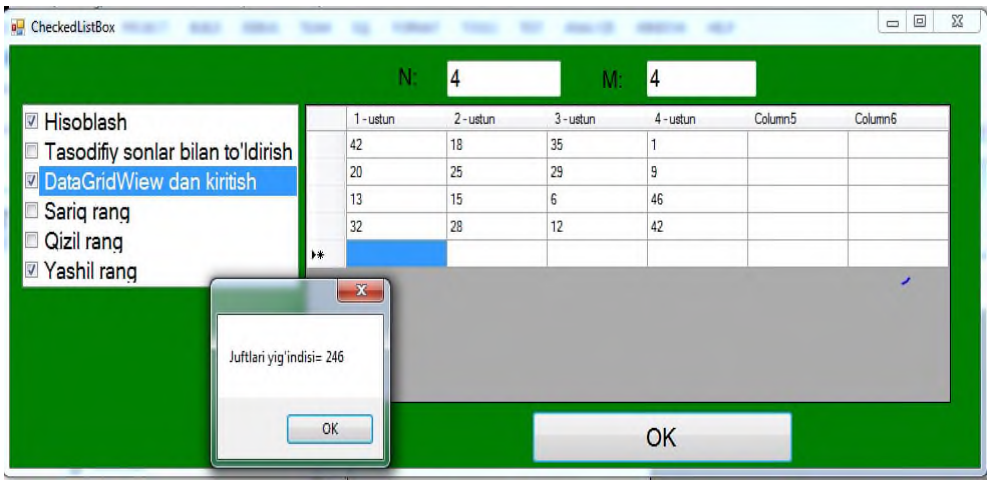


3.18-rasm. 2- formaning ko‘rinishi

3-qadam: 32 – variantdagi masalani **ye**chish uchun 3-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

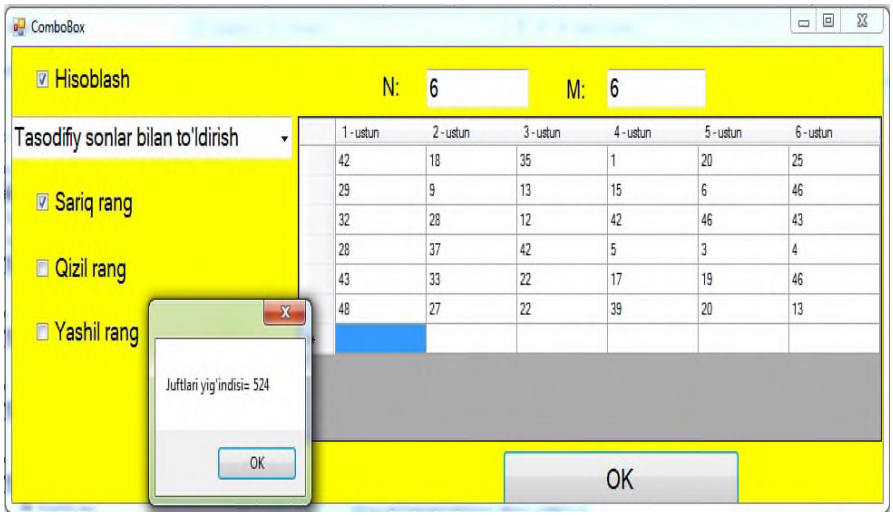
4-qadam: 32 – variantdagi masalani **ye**chish uchun 4-formada quyidagi kodlar kiritiladi:

```
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.      }
```



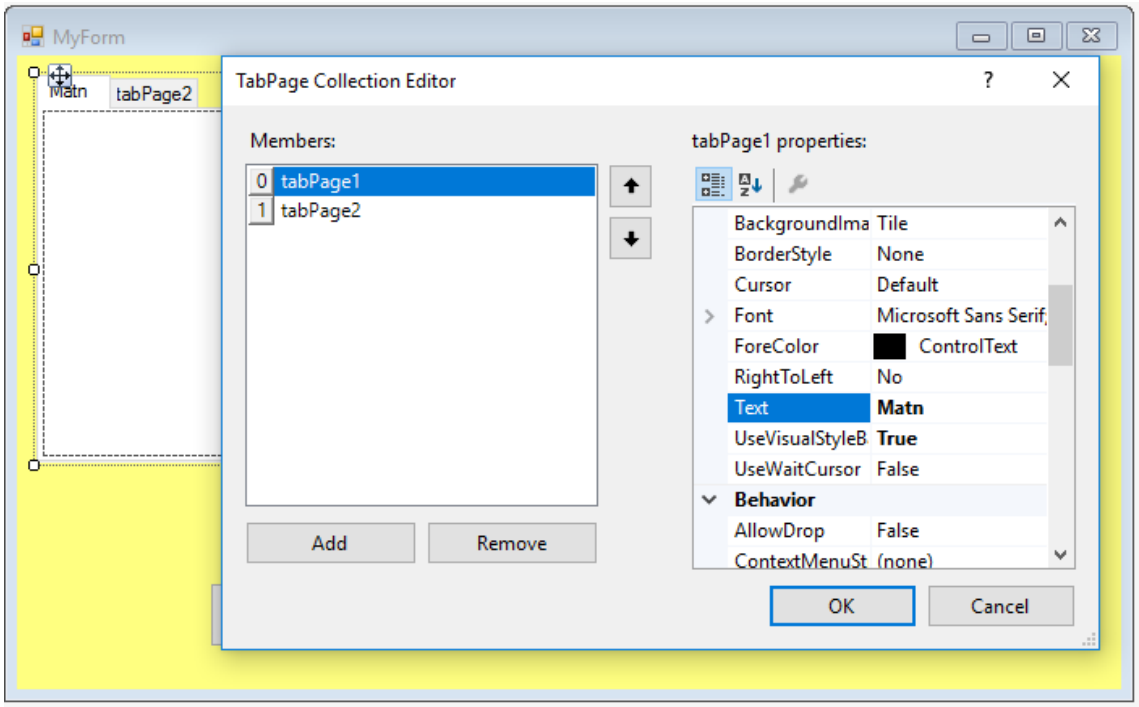
3.20-rasm. 4- formaning ko‘rinishi

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.

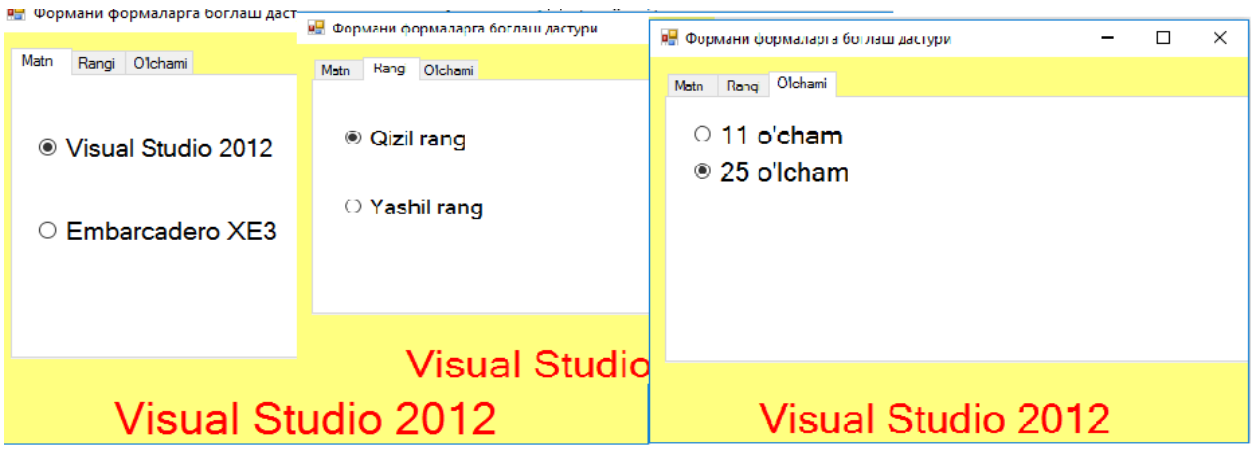
Uchta **TabControl** komponentalarida **RadioButton** tanlash komponentalaridan foydalangan holda matn uchun berilgan ikki variantdan birini tanlaydigan, matn uchun rang va o‘lcham beradigan dastur tuzish masalasi qo‘yilsin. Masalaning dastur ko‘rinishi 3.21-rasmda keltirilgan.

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.

```
1. #pragma endregion
2. // Formani yuklash xodisasini hosil qilish
3. private: System::Void Form1_Load(System::Object^  sender,
4. System::EventArgs^  e) {
5. this->Text="Formani formalarga boglash dasturi";
6. button1->Text = "2- formaga utish";
7. label1->Text = "1-forma";
8. auto TabPage3 = gcnew System::Windows::Forms::TabPage();
9. TabPage3->UseVisualStyleBackColor = true;
10.     this->tabControl1->Controls->Add(TabPage3);
11.     TabPage3->Controls->Add(this->radioButton5);
12.     TabPage3->Controls->Add(this->radioButton6);
13.     this->radioButton5->Location =
   System::Drawing::Point(20,15);
14.     this->radioButton6->Location =
   System::Drawing::Point(20,45);
15.     tabControl1->TabPages[0]->Text = "Matn";
16.     tabControl1->TabPages[1]->Text = "Rangi";
17.     tabControl1->TabPages[2]->Text = "O'lchami";
18.     radioButton1->Text    = "Visual Studio 2012";
19.     radioButton2->Text    = "Embarcadero XE3";
20.     radioButton3->Text    = "Yashil rang";
21.     radioButton4->Text    = "Qizil rang";
22.     radioButton5->Text    = "11 o'cham";
23.     radioButton6->Text    = "25 o'lcham";
24.     label1->Text = radioButton1->Text;
25.     radioButton5->Font = gcnew System::Drawing::Font(
26.     radioButton5->Font->Name,16);
27.     radioButton6->Font = gcnew System::Drawing::Font(
28.     radioButton6->Font->Name,16); }
29.     private: System::Void
   radioButton1_CheckedChanged(System::Object^
30.     sender, System::EventArgs^  e) { label1->Text =
31.     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…**

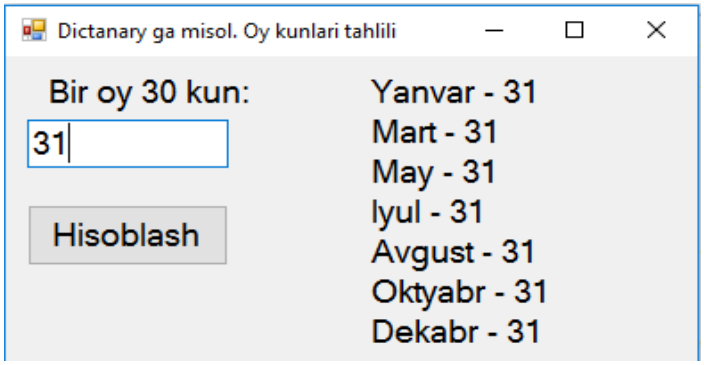
buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga “**Dictionary**” nomi beriladi va **OK** tugmasi bosiladi.

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

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

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

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.

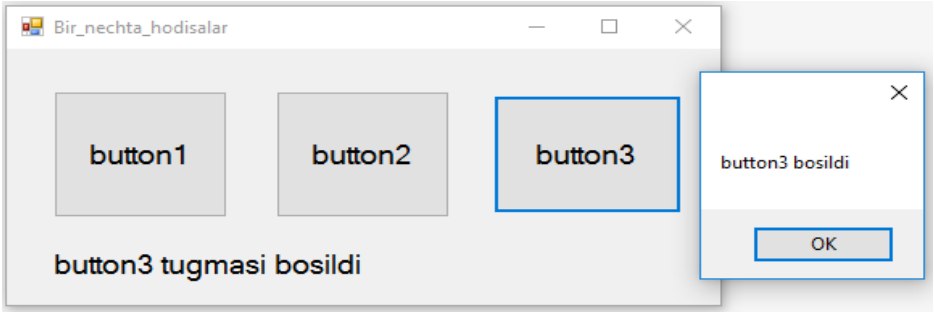
Buni qanday amalga oshirishni namuna misol orqali ko‘ramiz. Formaga ikkita tugma joylashtirib, ulardan ixtiyoriy biri bosilganda bitta prosedura ishlaydi. Bunda proseduraning sender parametridan foydalanib, sichqoncha yordamida aynan qaysi tugmaga bosilganligi aniqlanadi.

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()** prosedurasida **ikkichi_usul()** prosedurasi chaqirildi.



3.24- rasm. Bosilgan tugmani aniqlovchi dastur natijasi.

3.10. Turli tipdagi fayllarning manbalariga LinkLabel komponentasi yordamida murojaatlar

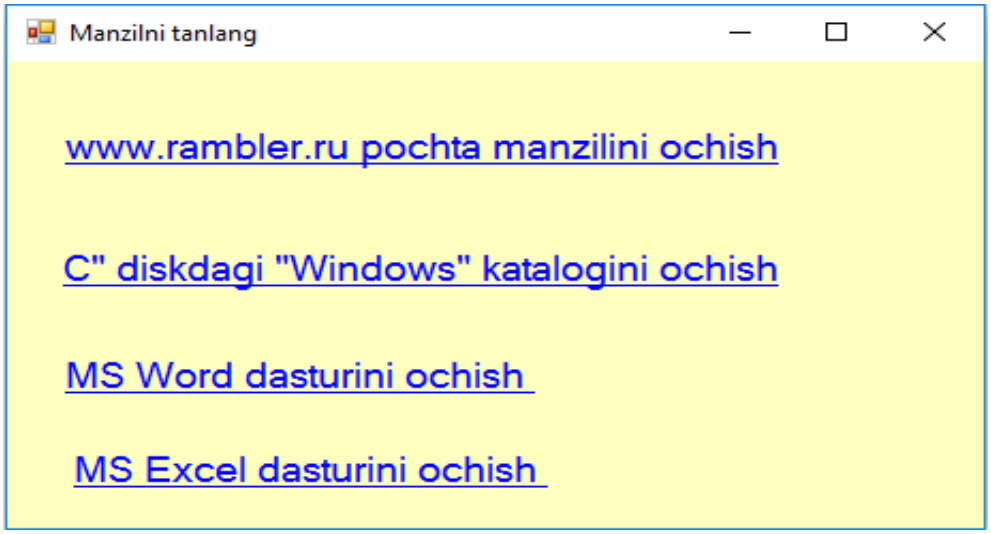
LinkLabel boshqaruv elementi formadan HTML xujjatlaridagi murojaatlarga o‘xshash veb sahifaga, dasturdagi qandaydir fayllarga, lokal disklarga, kataloglarga murojaat qilish imkoniyatini yaratadi.

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;  }   }  }; }
```



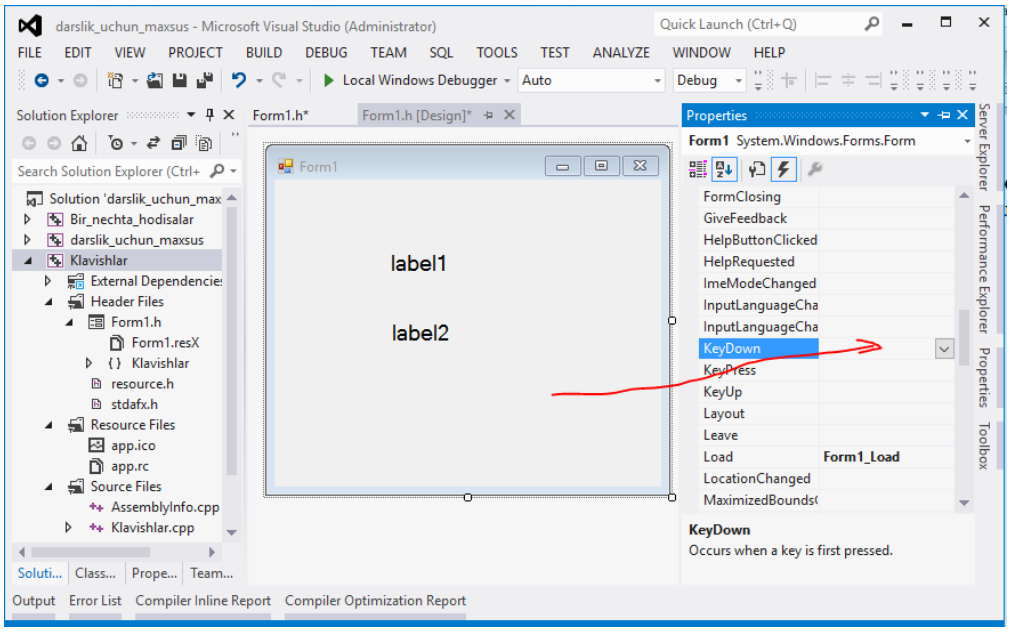
3.25-rasm. Manbalarga murojat dasturi natijasi.

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 yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga “**Klavishlar**” nomi beriladi va **OK** tugmasi bosiladi. Elementlar paneli **ToolBox** dan ikkita Label komponentasi joylashtiriladi va **Form1** ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar 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-rasm**) ustida sichqonchaning chap tugmasini ikki marta bosib, quyidagi kodlar 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; }   }
```

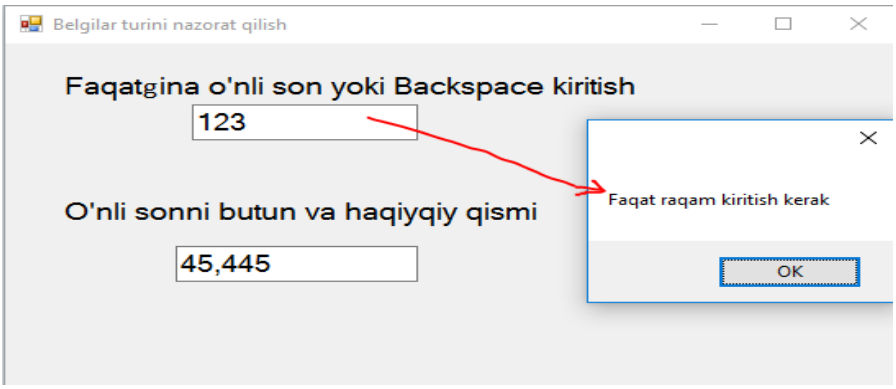
3.27- rasm. Klavyatura klavishlarini aniqlash dasturi

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:

```
#pragma endregion
String^ NuqtaVergul;
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
this->Text = "Belgilar turini nazorat qilish";
textBox1->Clear(); textBox2->Clear();
NuqtaVergul = Globalization::NumberFormatInfo::CurrentInfo-
>NumberDecimalSeparator; }
private: System::Void textBox1_KeyPress(System::Object^  sender,
     System::Windows::Forms::KeyPressEventArgs^  e) {
//Faqatgina o'nli son yoki Backspace kiritish masalasini hal qilish
if(Char::IsDigit(e->KeyChar)==true) return;
if(e->KeyChar == (char)Keys::Back) return;
else{MessageBox::Show("Faqat raqam kiritish kerak");}
// boshqa belgilar kiritilishini taqiqlash
e->Handled = true; } bool topildi;
private: System::Void textBox2_KeyPress(System::Object^  sender,
System::Windows::Forms::KeyPressEventArgs^  e) {
// Faqatgina o'nli son yoki Backspace kiritish masalasini hal qilish
if(Char::IsDigit(e->KeyChar)==true) return;
if(e->KeyChar == (char)Keys::Back) return;
else{MessageBox::Show("Faqat raqam kiritish kerak");}
if(textBox2->Text->IndexOf(NuqtaVergul) != -1)
topildi = true; if(topildi == true){ e->Handled = true; return;
}
if(e->KeyChar.ToString() == NuqtaVergul) return;
// boshqa belgilar kiritilishini taqiqlash
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:

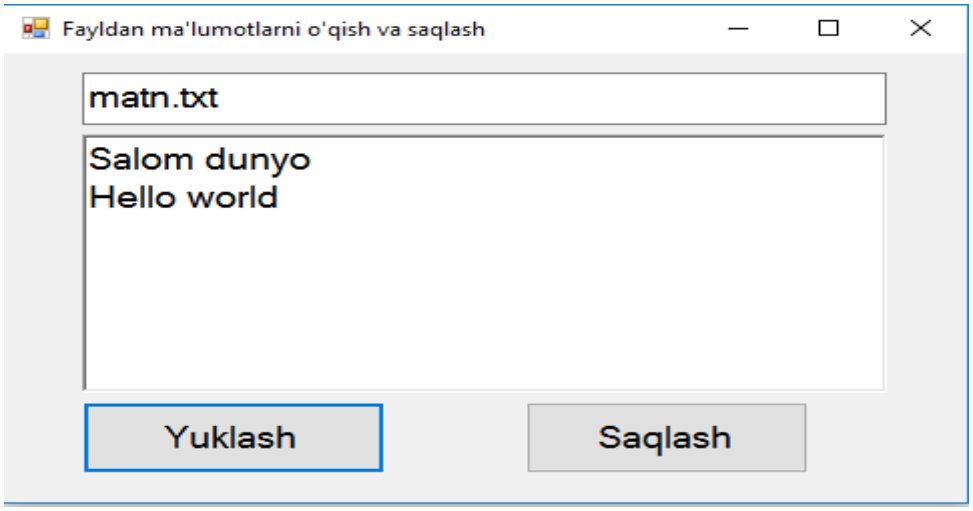
```
1. #pragma endregion
2. String^ fay_nomi; bool fayl(){
3. if(textBox1->Text =="" || textBox1->Text =="Fayl nomi"){
```

```
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‘tish lozim. Try qo‘llanilishining mazmuni quyidagicha: qandaydir masalani, misol uchun matnni o‘qishga *harakat qilish* (**try**). Agarda masala

noto‘g‘ri ishlangan, masalani fayli topilmagan bo‘lsa, u holda boshqaruvni qo‘lga olish (**catch**) va yuz bergan xolatni qayta ishlash (**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

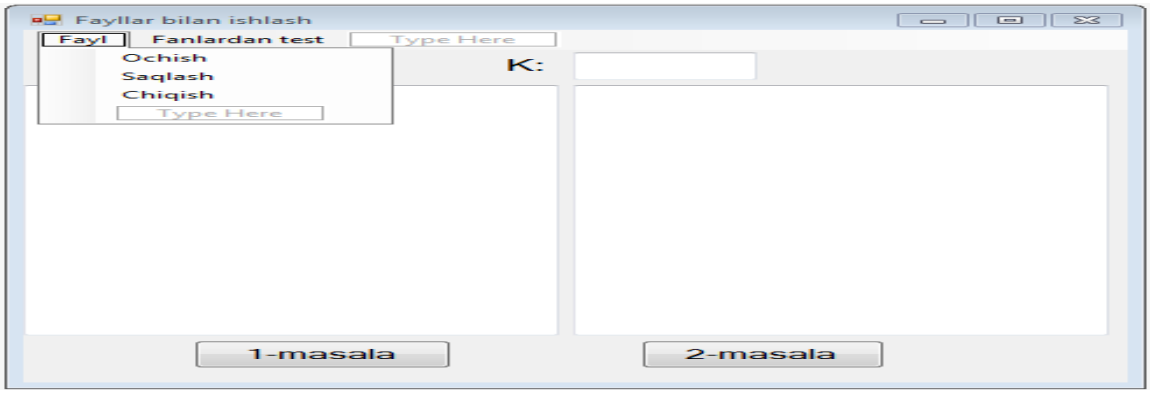
Berilgan matnli faylni nomini va kengaytmasini **textBox1** komponentasiga yozmaguncha **fayl()** funksiyasi **false** qiymat qaytaraveradi. Fayl manzili avtomatik tarzda C:\\ diskga to‘g‘irlangan. Fayl nomi va kengaytmasi (**txt**) to‘g‘ri yozilsa **fayl()** funksiyasi **true** qiymat qaytaradi va dasturning 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 beruvchi vizual dastur yaratishni ko‘rib chiqaylik. Ushbu vazifani bajaruvchi dasturni tuzish uchun

1-qadam. 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 “**Fayllar**” nomi beriladi va **OK** tugmasi bosiladi. Elementlar paneli **ToolBox** dan formaga ma`lumot kiritish va chiqarish uchun 2 ta **TextBox1, TextBox2, K** sonini kiritish uchun **TextBox3,** 1-masalani yechish uchun **Button1**, 2- masala uchun esa **Button2,** ma`lumotlarni ixtiyoriy nom bilan saqlash uchun **openFileDialog1, saveFileDialog1** va **menuStrip1** komponentalari joylashtiriladi. Dastur ko‘rinishi quyidagicha:



3.30- rasm. Dasturning umumiy ko‘rinishi

2-qadam. Button1 ning **click** hodisasiga quyidagi kodlar yoziladi:

```
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:

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

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

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

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

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

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

```
1. void Zapis(){
2. try{
3. auto Pisatel = gcnewIO::StreamWriter(saveFileDialog1->FileName, false,
4. System::Text::Encoding::GetEncoding(1251));
5. Pisatel->Write(textBox1->Text);
6. Pisatel->Close(); textBox1->Modified = false; }
7. catch (Exception^ Situasiya){
8. MessageBox::Show(Situasiya->Message, "Hatolik", MessageBoxButtons::OK,
9. 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:

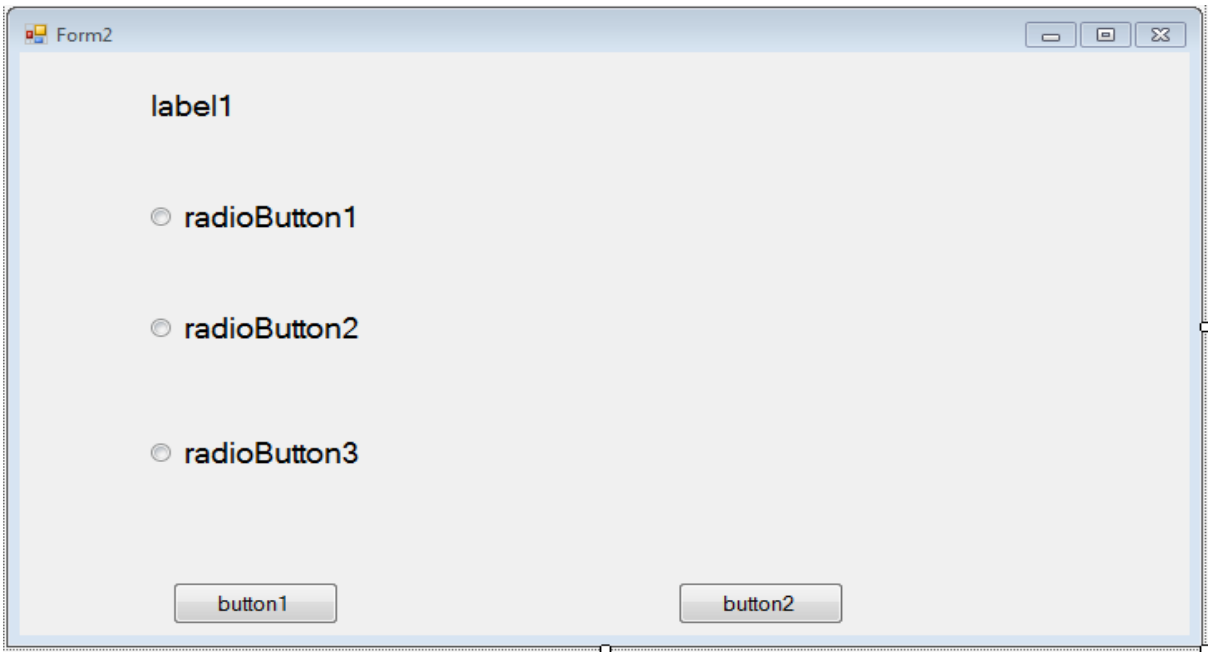
```
private: System::Void Form1_FormClosing(System::Object^  sender,
System::Windows::Forms::FormClosingEventArgs^  e) {
if (textBox1->Modified == false) return;
auto MBox = MessageBox::Show("Tekst bo`l izmenen. \nSoxranit
izmeneniya?","Prostoy redaktor", MessageBoxButtons::YesNoCancel,
MessageBoxIcon:: 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::OK)
     { Zapis(); return; }  else e->Cancel = true;  }}
```

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

8-qadam. Fanlardan test menyusidagi **Informatika** bo‘limining **onclick** hodisasining kodlari quyidagicha:

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

9-qadam. 2- formada **label1, radiobutton1, radiobutton2, radiobutton3, 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:

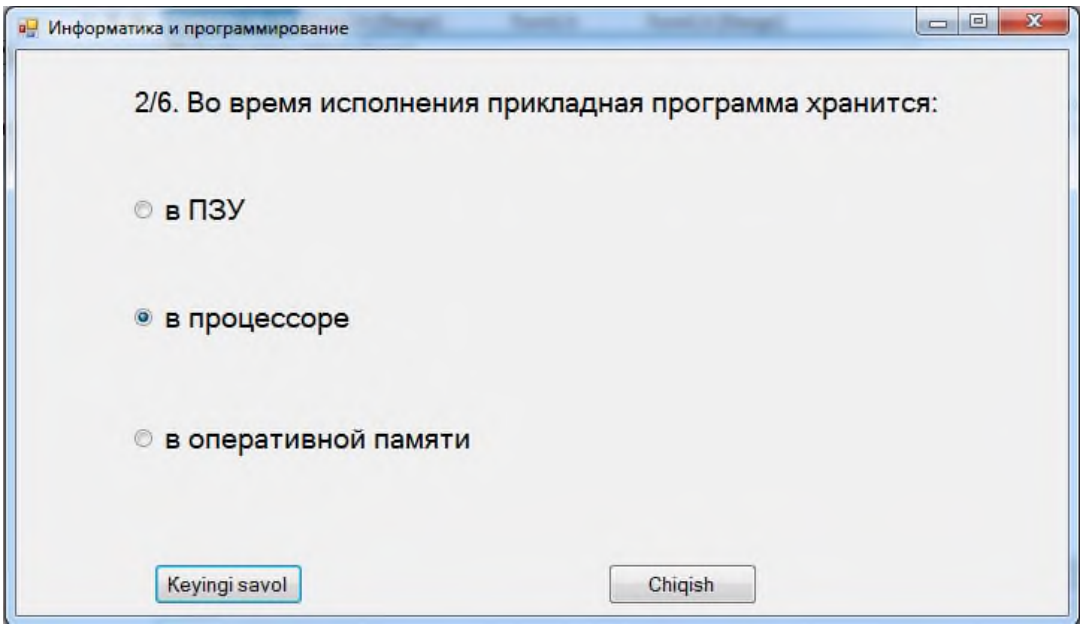
```
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^  e) {
3. if (Tanlangan_javob == TogriJavobNomeri) TogriJavob = TogriJavob + 1;
4. if (Tanlangan_javob != TogriJavobNomeri){
5. NoTogriJavob = NoTogriJavob + 1; NoTogri_javoblar[NoTogriJavob] =
6. label1->Text; }
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
```

```
int SavollarSoni;
int TogriJavob; int NoTogriJavob; array<String^>^
NoTogri_javoblar;
int TogriJavobNomeri; int Tanlangan_javob;
IO::StreamReader^ oqish;
private: System::Void Form2_Load(System::Object^  sender,
System::EventArgs^  e) {
button1->Text = "Keyingi savol";
button2->Text = "Chiqish";
radioButton1->CheckedChanged +=gcnew EventHandler(this,
    &Form2::Keyingi_savolga_otish);
    radioButton2->CheckedChanged +=gcnew EventHandler(this,
    &Form2::Keyingi_savolga_otish);
    radioButton3->CheckedChanged +=gcnew EventHandler(this,
    &Form2::Keyingi_savolga_otish);  Testni_boshalsh();}
    void Testni_boshalsh(){
    System::Text::Encoding^ Kodirovka=
    System::Text::Encoding::GetEncoding(1251);
    try{
    oqish = gcnew IO::StreamReader(
    IO::Directory::GetCurrentDirectory()+ "\\test.txt",
Kodirovka);
    this->Text = oqish->ReadLine();
    SavollarSoni = 0; TogriJavob = 0; NoTogriJavob = 0;
    NoTogri_javoblar = gcnew array<String^>(100);  }
    catch (Exception^ Situasiya)
    { MessageBox::Show(Situasiya->Message, "Hatolik",
    MessageBoxButtons::OK, MessageBoxIcon::Exclamation); }
    Key_savol_oqish(); }
    void Key_savol_oqish(){
    label1->Text = oqish->ReadLine();
    radioButton1->Text = oqish->ReadLine();
    radioButton2->Text = oqish->ReadLine();
    radioButton3->Text = oqish->ReadLine();
    TogriJavobNomeri = int::Parse(oqish->ReadLine());
    radioButton1->Checked = false; radioButton2->Checked= false;
    radioButton3->Checked = false;
    button1->Enabled = false;
    SavollarSoni = SavollarSoni + 1;
    if (oqish->EndOfStream == true) button1->Text = "Zavershit";
}
    private: Void Keyingi_savolga_otish(System::Object^ sender,
    System::EventArgs^ e){
    button1->Enabled = true; button1->Focus();
    RadioButton^ Pereklyuchatel = (RadioButton^)sender; String^
tmp = Pereklyuchatel->Name;
    Tanlangan_javob = int::Parse(tmp->Substring(11)); }
```

Talabalarni fanlardan baholash dasturini ishga tushirilgandan 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:

```
#pragma endregion
IO::StreamReader^ uqish;
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {this->Text = "Matn faylni chop qilish";
richTextBox1->Clear();
//richTextBox1->ScrollBars = ScrollBars::Vertical;
chopQilishToolStripMenuItem->Visible = false;
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(
```

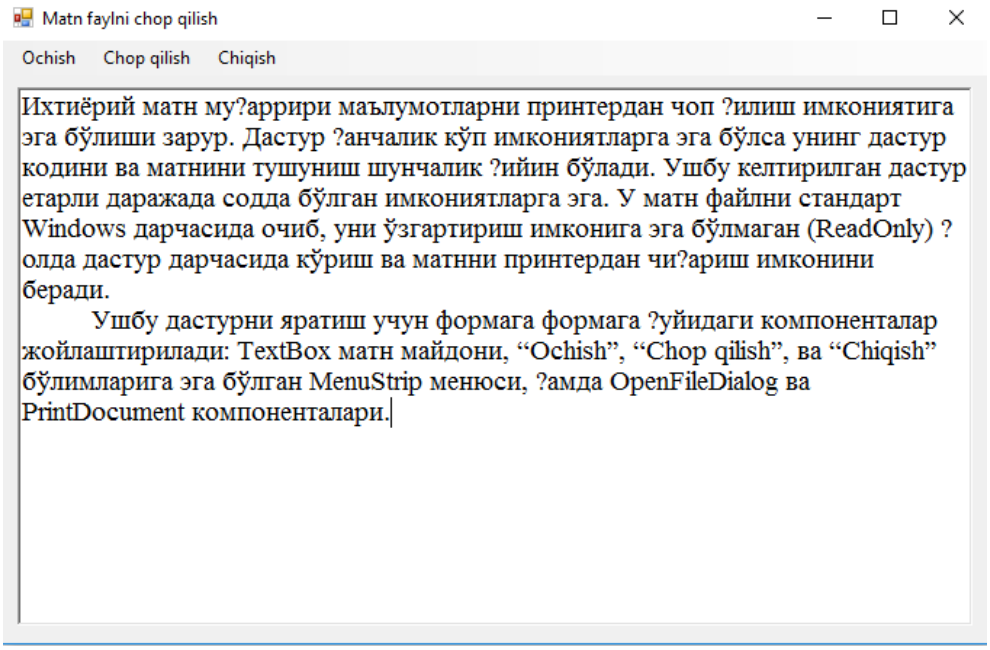
```
52.      e->Graphics);
53.      // Faylni har bir qatorini pechatga chiqarish
54.      int i=0;// qator soni
55.      while(i<sahifadagiQatorlar){
56.      if(Qator == nullptr) break; // sikldan chiqish
57.      Y =yuqoriTomon + i* shrift->GetHeight(e->Graphics);
58.      // Qatorni pechat qilish
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 berishda foydalanuvchiga matnli maydonni tahrirlash taqiqlanadi, ya`ni ReadOnly = true; Hamda chop qilish xossasiga **chopQilishToolStripMenuItem->Visible = false;** (formaning **Chop qilish** bandi) belgilanadi, ya`ni dastur ishlashining boshida **Chop qilish** bandi foydalanuvchiga koʻrinmaydi.

Formaning Ochish xossasiga ishlov berishda OpenFileDialog standart muloqot chaqiriladi va faylni oʻqishni StreamReader oqimini yaratish orqali tashkil qilinadi. Try…finally…catch bloklarda dastur yana bir marta StreamReader oqimini yaratadi, soʻngra, xujjatni chop qilish jarayonini ishga tushiradi printDocument1->Print. Agar faqat printDocument1->Print metodidan boshqasi dasturlashtirilmagan boʻlsa, u holda printer boʻsh 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 oʻzgaruvchilar 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) oʻqiydi, soʻngra uni **DrawString** buyrugʻi chop qiladi. Bu yerda “e” hodisasining argumentidan olinuvchi (Graphics) grafik ob`ektidan foydalaniladi. I oʻzgaruvchisida satrlar sanaladi. Agar satrlar miqdori sahifa satrlari sonidan katta boʻlsa, u holda dastur satr oʻzgaruvchisidagi 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 tasvirini formaga chiqarish masalasi qo‘yilgan bo‘lsin. Formada grafika bilan ishlashni turlicha amalga oshirish mumkin. Grafik bilan ishlashni OnPrint metodini qayta aniqlash orqali ko‘rib chiqamiz. OnPrint metodi Form sinfining a`zosi hisoblanadi.

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 “**Grafik_rasmlar**” nomi beriladi va **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 chizish

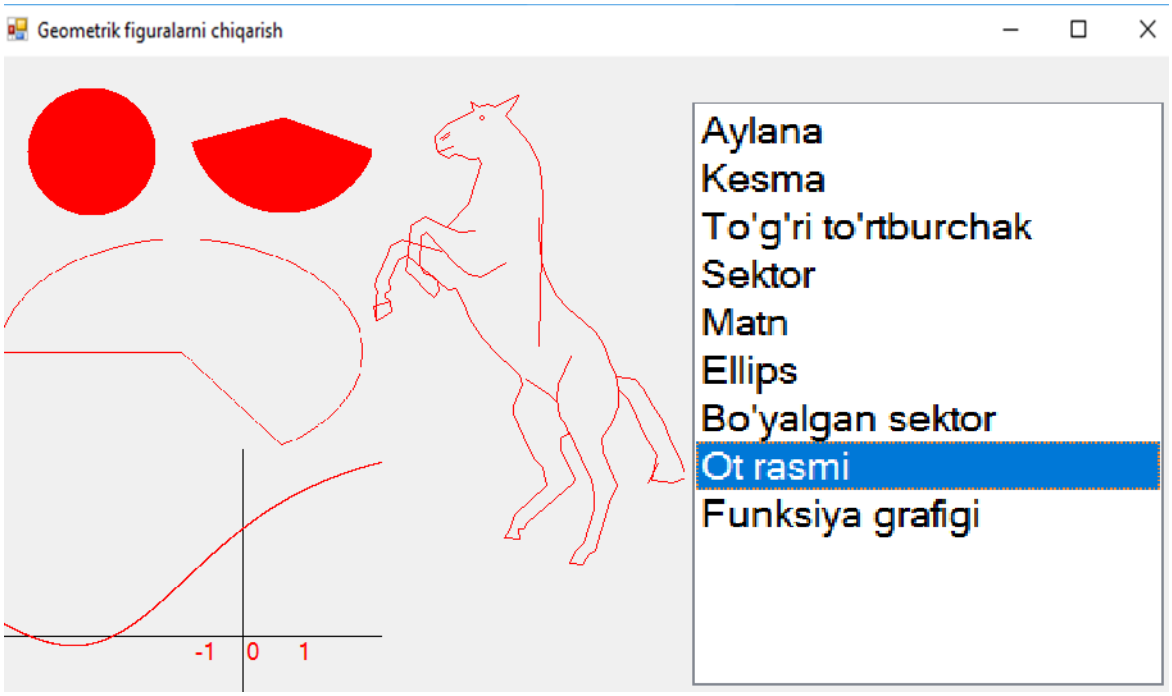
Vektorli chizmalarda quyidagi grafik shakllar chizmaning elementar tashkil etuvchilari deyiladi: kesma, yoy, belgi, aylana va h.k. Bunda rastrli grafika asosida koordinata bo‘yicha shakl chizadi. Visual Studioda koordinata tizimi quyidagicha aniqlanadi: koordinata boshi formaning chap yuqori burchagi, 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:

```
1. #pragma endregion
2. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {
3. this->Text = "Sodda figurali shakillar chizish";
4. listBox1->Items->AddRange(gcnew array<Object^>{"Aylana", "Kesma",
5. "To'g'ri to'rtburchak",    "Sektor", "Matn", "Ellips", "Bo'yalgan
   sektor",
6. "Ot rasmi","Funksiya grafigi"}); }
7. void Fun_gra(){ float x,y;
8. Graphics^Figura=this->CreateGraphics();
9. Pen^Qalam= gcnew Pen(Color::Red);
10.     Pen^qalam= gcnew Pen(Color::Black);
```

```
11.      Brush^MQalam =gcnew SolidBrush(Color::Red);
12.      Figura->DrawLine(qalam,0,250,400,250);
13.      Figura->DrawLine(qalam,200,0,200,800);
14.      Figura->DrawString("0",Font,MQalam,200,250);
15.      Figura->DrawString("-1",Font,MQalam,160,250);
16.      Figura->DrawString("1",Font,MQalam,240,250); x=-1;
17.      do    { y=(x-3)/(x*x+2);
18.      Point l=Point(100+(x*100),100-(y*100));
19.      Point k=Point(100+(x+0.001)*100,100-(y+0.001)*100);
20.      array<Point>^qizil={l,k};
21.      Figura->DrawPolygon(Qalam,qizil);
22.      x+=0.001; }while (x<=4);}
23.      private: System::Void
24.      listBox1_SelectedIndexChanged(System::Object^  sender,
25.      System::EventArgs^  e) {
26.      SolidBrush^ blueBrush = gcnew SolidBrush(Color::Blue );
27.      // point o'zgarmaslarini yaratish.
28.      Point point1 = Point(50,50); Point point2 = Point(100,25);
29.      Point point3 = Point(200,5); Point point4 = Point(250,50);
30.      Point point5= Point(300,100); Point point6 = Point(350,200);
31.      Point point7 = Point(250,250); array<Point>^ curvePoints =
32.      {point1,point2,point3,point4,point5,point6,point7};
33.      //Figurali obyektni hosil qilish
34.      Graphics^ Figura = this->CreateGraphics();
35.      //Figurali shakilni chizish uchun qalamni hosil qilish
36.      Pen^ Qalam = gcnew Pen(Color::Red);
37.      //Figurali shaklni bo'yash uchun mo'yqalamni hosil qilish
38.      Brush^ Mqalam = gcnew SolidBrush(Color::Red);
39.      //Figurali shakl chizish sohasini tozalash bilan birgalikda
   forma rangini berish
40.      Figura->Clear(SystemColors::Control);
41.      switch (listBox1->SelectedIndex){
42.      // Aylana chizish
43.      case 0: Figura->DrawEllipse(Qalam, 50, 50, 250, 150);
   break;
44.      // Kesma chizish
45.      case 1: Figura->DrawLine(Qalam, 50, 50, 200, 200); break;
46.      // To'g'ri to'rtburchak chizish
47.      case 2: Figura->DrawRectangle(Qalam, 50, 30, 150, 280);
   break;
48.      // Sektor chizish
49.      case 3: Figura->DrawPie(Qalam, 30, 50, 300, 200, 180, 225);
50.      break;
51.      // Matn chiqarish
52.      case 4: Figura->DrawString("Har narsaning iloji bor", Font,
53.      Mqalam, 10, 100); break;
54.      // ellips chiqarish
55.      case 5: Figura->DrawEllipse(Qalam, 30, 30, 150, 200); break;
56.      // Bo'yalgan sektor chiqarish
57.      case 6: Figura->FillPie(Mqalam, 20, 50, 150, 150, 20, 145);
```

```
58.      break;
59.      case 7:    Figura->DrawLine(Qalam, 167, 250, 165, 239);
   break;
60.      case 8: Fun_gra(); break;}}}; }
```



3.35- rasm. Geometrik shakillarni chizish dasturi

3.18. Formada sichqoncha ko‘rsatgichi orqali chizish

Ushbu mavzuda qanday qilib, formaga sichqoncha yordamida chizish ko‘rsatiladi. Ya`ni, sichqonchaning chap yoki o‘ng tugmasi bosilganda formaga chizish imkonini beruvchi dastur yozish masalasi qo‘yilgan. Agar foydalanuvchi sichqoncha tugmasini qo‘yib yuborsa, u holda chizish jarayoni to‘xtatiladi. Loyihalashtirilayotgan formada chizmani tozalash uchun mo‘ljallangan “**Tozalash**” tugmasini inobatga olish lozim.

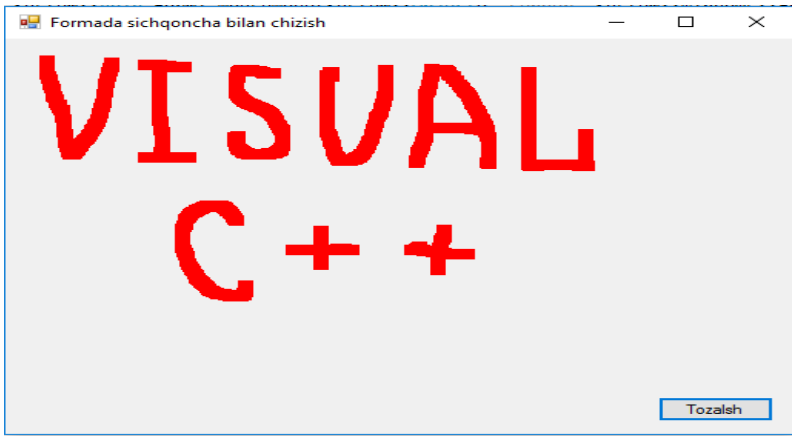
Ushbu vazifani bajaruvchi dasturni tuzish uchun **Visual Studio 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo‘limlaridan **File->New->Project…** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga “**Chizish**” nomi berilib, **OK** tugmasi bosiladi. **ToolBox** panelidan formaga **Button** komponentasi joylashtiriladi va quyidagi kodlar teriladi:

```
1. #pragma endregion
2. bool chizish;
3. private: System::Void Form1_Load(System::Object^ sender,
4. System::EventArgs^  e) {
```

```
this->Text = "Formada sichqoncha bilan chizish";
button1->Text = "Tozalsh"; chizish = false; }
private: System::Void Form1_MouseDown(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {
// Sichqoncha tugmasi bosilgan bo'lsa chizish
     chizish = true;      }
     private: System::Void Form1_MouseUp(System::Object^  sender,
     System::Windows::Forms::MouseEventArgs^  e) {
     // Sichqoncha tugmasi qo'yib yuborliganda chizmaslik
     chizish = false;    }
     private: System::Void Form1_MouseMove(System::Object^sender,
     System::Windows::Forms::MouseEventArgs^  e) {
     if(chizish==true) { Graphics^ Grafika = CreateGraphics();
     Grafika->FillRectangle(gcnew SolidBrush(Color::Red),
     e->X, e->Y, 10, 10); delete Grafika; }              }
     private: System::Void button1_Click(System::Object^  sender,
     System::EventArgs^ e) {Graphics^ Grafika = CreateGraphics();
     Grafika->Clear(this->BackColor); } }; }
```

Bu yerda, dasturning boshida false qiymatiga ega bo‘lgan mantiqiy turdagi chizish o‘zgaruvchisi e`lon qilingan. Ushbu o‘zgaruvchi yoki sichqonchani harakatlantirganda (MouseMove hodisasi) formaga chizishni (chizish=true), yoki chizishga ruxsatni man etadi (chizish=false). Sichqonchani tugmasini bosilgan holda harakatlantirilganda, GreateGraphics() metodidan foydalangan holda dastur nomlar sohasi System::Drawing Graphics grafik ob`ektni yaratadi va 10x10 piksel o‘lchamdagi, qizil rang bilan to‘ldirilgan FillRectangle() to‘g‘ri to‘rtburchak chizadi. e.X, e.U – sichqoncha ko‘rsatgichining koordinatasi, hamda to‘g‘ri to‘rtburchakning yuqori chap burchagi koordinatasi ham hisoblanadi.

Quyidagi rasmda formaga chizish namunasi keltirilgan. Barcha formaga chizilganlarni o‘chirish uchun Tozalash tugmasini bosish zarur.

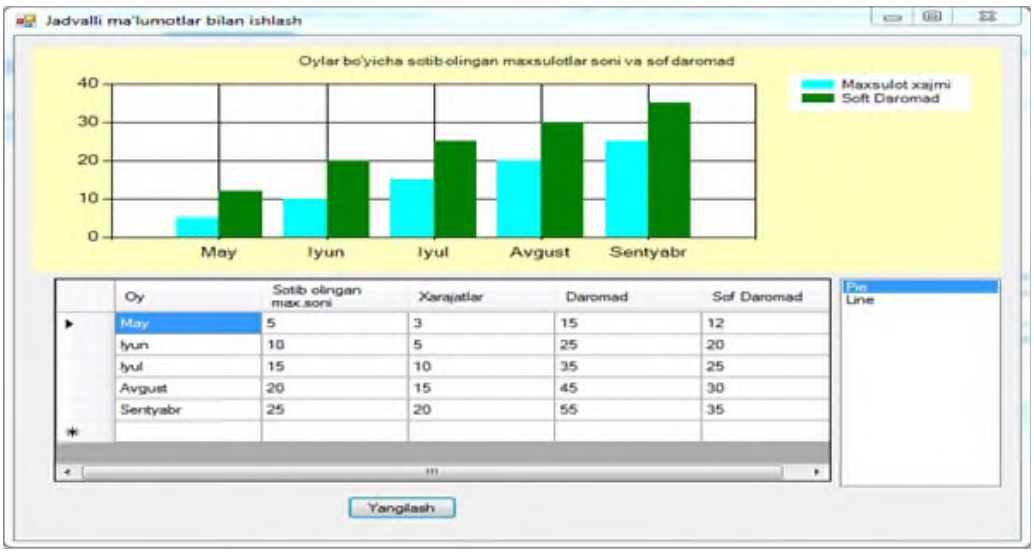


3.36- rasm. Formaga sichqoncha yordamida chizish

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

Jadval tipidagi ma`lumotlar yaratish uchun **DataTable** tipidan foydalaniladi. Bu tipda yaratilgan o‘zgaruvchi ustun va satrlardan tashkil topadi. Jadval ko‘rinishidagi ma`lumotlar asosida grafik diagrammalar qurish oson bo‘ladi. Formaning grafik imkoniyatlarini o‘zida jamlagan komponenta bu **Chart** komponentasidir. Ushbu komponenta yordamida jadvalli ma`lumotlarni ekranga grafik diagrammalar ko‘rinishida tasvirlash mumkin. Ushbu mavzuning mazmunini har oyda sotib olingan mahsulotlar miqdorlarini jadvallarda shakllantirib, grafik diagramma ko‘rinishida tasvirlovchi vizual dastur yaratish orqali tushuntirib o‘tamiz. Topshiriqdagi variantingiz masalasining dasturini tuzishda ushbu dastur namuna sifatida xizmat qiladi. Lekin diagrammalarni ko‘rinishlari va turlari har xil bo‘lganligi uchun ushbu turlardan ham foydalanib tuzilsa, dasturimizning tashqi ko‘rinishlari chiroyli va ko‘rgazmali bo‘ladi.

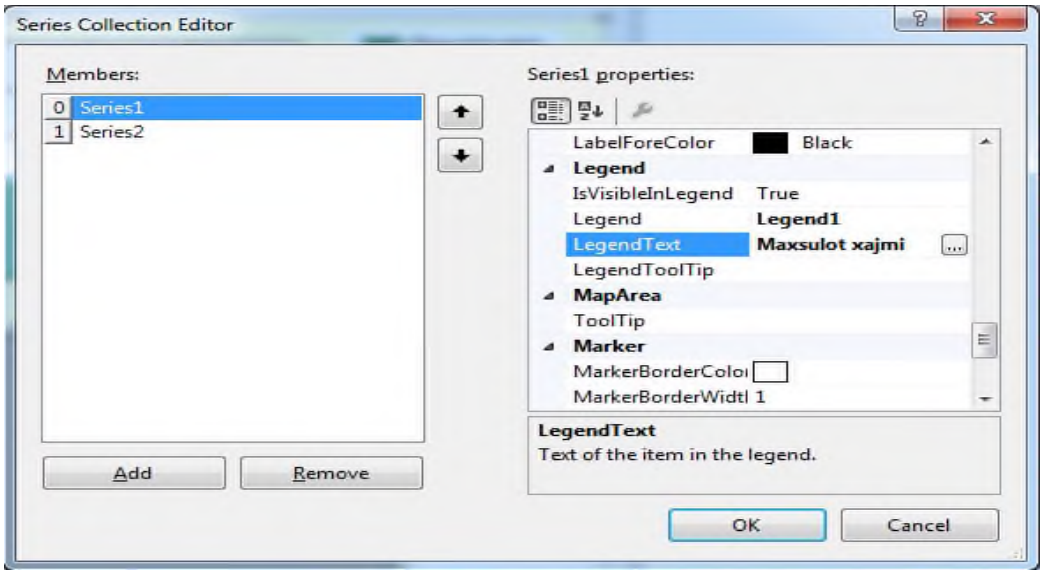
Ushbu vazifani bajaruvchi dasturni tuzish uchun **Visual Studio 2012** dasturi ishga tushiriladi, asosiy oyna menyu bo‘limlaridan **File->New->Project…** buyruqlari beriladi yoki **Ctrl+Shift+N** klavishalari bosiladi, ochilgan oynada loyihaga “**Jadvalli_malumotlar**” nomi berilib. **OK** tugmasi bosiladi. Dasturning umumiy ko‘rinishi quyidagicha bo‘ladi.



3.37- rasm. Dasturning umumiy ko‘rinishi oynasi

Dasturni tuzish quyidagi qadamlardan tashkil topgan:

1-qadam. Formaga **chart** komponentasi joylashtiriladi. Uning **Series** xossasiga kirib, 2- **Series** qo‘shiladi. **Series1** tanlanadi va uning **LegendText** xossasiga **“Maxsulot xajmi”** matni yoziladi. **Series2** tanlanadi va uning **LegendText** xossasiga **“Soft daromat”** matni yoziladi. Ushbu **Series** qo‘shish oynasi **3.38-** rasmda keltirilgan.



3.38- rasm. Series qo‘shish oynasi

2-qadam. Formaga **dataGridView, listBox** va **button** komponentalari joylashtiriladi. Ularning hech qanday xossasi o‘zgartirilmaydi.

3-qadam. Formaning **load** hodisasi kodlari quyidagicha:

```
1. DataTable ^ Jadval;
2. private: System::Void Form1_Load(System::Object^  sender,
3. System::EventArgs^  e) {chart1->Titles->Add("Oylar bo'yicha sotib
4. olingan maxsulotlar soni va sof daromat");
5. listBox1->Items->Add("Pie"); listBox1->Items->Add("Line");
6. this->Text = "Grafik ma'lumotlar";
7. Jadval = gcnew DataTable();
8. //Ushbu jadvalda 2 ta ustun "Oy"va "Sotib olingan //max.soni":
9. Jadval->Columns->Add("Oy", String::typeid);
10.      //C# da: Jadval.Columns.Add("Mesyas", //typeof(String));
11.      //2-ustun bo'yicha ma'lumotlar long tipida bo'lasi:
12.      Jadval->Columns->Add("Sotib olingan max.soni",
   Int64::typeid);
13.      Jadval->Columns->Add("Xarajatlar", Int64::typeid);
14.      Jadval->Columns->Add("Daromat", Int64::typeid);
15.      Jadval->Columns->Add("Sof daromat", Int64::typeid);
16.      //C# da: Jadval.Columns.Add("Ob`em prodaj", typeof(long));
17.      //Jadvalning 1- qatorini to'ldiramiz:
18.      DataRow ^ Qator = Jadval->NewRow();
```

```
19.      Qator["Oy"] = "May"; Qator["Sotib olingan max.soni"] =10;
20.      Qator["Xarajatlar"] =3; Qator["Daromat"] =15;
21.      Qator["Sof daromat"] =Convert::ToInt64(Qator["Daromat"])-
22.      Convert::ToInt64(Qator["Xarajatlar"]);
23.      //Convert::ToInt64( dataGridView1->Rows[0]->Cells[0]-
   >Value);
24.      Jadval->Rows->Add(Qator);
25.      //Jadvalning 2- qatorini to'ldiramiz:
26.      Qator = Jadval->NewRow();
27.      Qator["Oy"] = "Iyun"; Qator["Sotib olingan max.soni"] = 20;
28.      Qator["Xarajatlar"] =5; Qator["Daromat"] =25;
29.      Qator["Sof daromat"] =Convert::ToInt64(Qator["Daromat"])-
30.      Convert::ToInt64(Qator["Xarajatlar"]);
31.      Jadval->Rows->Add(Qator);
32.      //3- qatorni qo'shamiz:
33.      Qator = Jadval->NewRow();
34.      Qator["Oy"] = "Iyul"; Qator["Sotib olingan max.soni"] = 30;
35.      Qator["Xarajatlar"] =10; Qator["Daromat"] =35;
36.      Qator["Sof daromat"] =Convert::ToInt64(Qator["Daromat"])-
37.      Convert::ToInt64(Qator["Xarajatlar"]);
38.      Jadval->Rows->Add(Qator);
39.      //4- qatorni qo'shamiz:
40.      Qator = Jadval->NewRow();
41.      Qator["Oy"] = "Avgust"; Qator["Sotib olingan max.soni"]= 40;
42.      Qator["Xarajatlar"] =15; Qator["Daromat"] =45;
43.      Qator["Sof daromat"] =Convert::ToInt64(Qator["Daromat"])-
44.      Convert::ToInt64(Qator["Xarajatlar"]);
45.      Jadval->Rows->Add(Qator);
46.      //5- qatorni qo'shamiz:
47.      Qator = Jadval->NewRow();
48.      Qator["Oy"] ="Sentyabr"; Qator["Sotib olingan max.soni"]=50;
49.      Qator["Xarajatlar"] =20; Qator["Daromat"] =55;
50.      Qator["Sof daromat"] =Convert::ToInt64(Qator["Daromat"])-
51.      Convert::ToInt64(Qator["Xarajatlar"]);
52.      Jadval->Rows->Add(Qator);
53.      dataGridView1->DataSource = Jadval;
54.      chart1->DataSource = Jadval;
55.      chart1->Series["Series1"]->XValueMember = "Oy";
56.      chart1->Series["Series2"]->XValueMember = "Oy";
57.      chart1->Series["Series1"]->YValueMembers = "Sotib olingan
   max.soni";
58.      chart1->Series["Series2"]->YValueMembers = "Sof daromat";
59.      chart1->Series["Series1"]->ChartType =
60.      System::Windows::Forms::DataVisualization::Charting::
61.      SeriesChartType::Column;
62.      chart1->Series["Series2"]->ChartType =
63.      System::Windows::Forms::DataVisualization::Charting::
64.      SeriesChartType::Column;
65.      // Diogrammaning turi quyidagicha bo'lishi ham
66.      //mumkin. masalan: Pie, Line va bohqa.
```

```
67.      chart1->Series["Series1"]->Color = Color::Aqua;
68.      chart1->Series["Series2"]->Color = Color::Yellow;
69.      chart1->DataBind();} }
```

Button1 tugmasini **click** xodisasining kodlari quyidagicha:

```
1. private: System::Void button1_Click(System::Object^  sender,
   System::EventArgs^  e) {dio();        }
2. void dio(){
3. this->Text = "Grafik ma'lumotlar";
4. Jadval = gcnew DataTable();
5. Jadval->Columns->Add("Oy", String::typeid);
6. Jadval->Columns->Add("Sotib olingan max.soni", Int64::typeid);
7. Jadval->Columns->Add("Xarajatlar", Int64::typeid);
8. Jadval->Columns->Add("Daromat", Int64::typeid);
9. Jadval->Columns->Add("Sof daromat", Int64::typeid);
10. DataRow ^ Qator = Jadval->NewRow();
11. Qator["Oy"] = dataGridView1->Rows[0]->Cells[0]->Value;
   Qator["Sotib
    olingan max.soni"] =dataGridView1->Rows[0]->Cells[1]->Value;
12. Qator["Xarajatlar"] =dataGridView1->Rows[0]->Cells[2]->Value;
13. Qator["Daromat"] =dataGridView1->Rows[0]->Cells[3]->Value;
14. Qator["Sof daromat"] =Convert::ToInt64(dataGridView1->
15. Rows[0]->Cells[3]->Value) -Convert::ToInt64(
16. dataGridView1->Rows[0]->Cells[2]->Value);
17. Jadval->Rows->Add(Qator);
18. Qator = Jadval->NewRow();
19. Qator["Oy"] =dataGridView1->Rows[1]->Cells[0]->Value;
20. Qator["Sotib olingan max.soni"] = dataGridView1->Rows[1]-
   >Cells[1]->Value;
21. Qator["Xarajatlar"] =dataGridView1->Rows[1]->Cells[2]->Value;
22. Qator["Daromat"] =dataGridView1->Rows[1]->Cells[3]->Value;
23. Qator["Sof daromat"] =Convert::ToInt64(dataGridView1->
    Rows[1]->Cells[3]->Value) -Convert::ToInt64(
24. dataGridView1->Rows[1]->Cells[2]->Value);
25. Jadval->Rows->Add(Qator);  Qator = Jadval->NewRow();
26. Qator["Oy"] = dataGridView1->Rows[2]->Cells[0]->Value;
27. Qator["Sotib
28. olingan max.soni"] = dataGridView1->Rows[2]->Cells[1]->Value;
29. Qator["Xarajatlar"] =dataGridView1->Rows[2]->Cells[2]->Value;
30. Qator["Daromat"] =dataGridView1->Rows[2]->Cells[3]->Value;
31. Qator["Sof daromat"] =Convert::ToInt64(
    dataGridView1->Rows[2]->Cells[3]->Value) –
32. Convert::ToInt64(dataGridView1->Rows[2]->Cells[2]->Value);
33. Jadval->Rows->Add(Qator); Qator = Jadval->NewRow();
34. Qator["Oy"] = dataGridView1->Rows[3]->Cells[0]->Value;
   Qator["Sotib olingan max.soni"] = dataGridView1->Rows[3]-
   >Cells[1]->Value;
35. Qator["Xarajatlar"] =dataGridView1->Rows[3]->Cells[2]->Value;
36. Qator["Daromat"] =dataGridView1->Rows[3]->Cells[3]->Value;
```

```
37. Qator["Sof daromat"] =Convert::ToInt64(dataGridView1->Rows[3]-
    >Cells[3]->Value) -Convert::ToInt64(
38. dataGridView1->Rows[3]->Cells[2]->Value);
39. Jadval->Rows->Add(Qator); Qator = Jadval->NewRow();
40. Qator["Oy"] = dataGridView1->Rows[4]->Cells[0]->Value;
    Qator["Sotib olingan max.soni"] = dataGridView1->Rows[4]-
    >Cells[1]->Value;
41. Qator["Xarajatlar"] =dataGridView1->Rows[4]->Cells[2]->Value;
42. Qator["Daromat"] =dataGridView1->Rows[4]->Cells[3]->Value;
43. Qator["Sof daromat"] =Convert::ToInt64(
     dataGridView1->Rows[4]->Cells[3]->Value) -Convert::ToInt64(
     dataGridView1->Rows[4]->Cells[2]->Value);
44. Jadval->Rows->Add(Qator);
45. // chart1 komponentasiga Jadvalni eksport qilish
46. chart1->DataSource = Jadval;
47. //chart1 komponentasidagi Serie1 ning X kordinatasiga Oy so'zini
    yozish
48. chart1->Series["Series1"]->XValueMember = "Oy";
49. // chart1 komponentasidagi Serie1 ning X kordinatasiga Oy
    so'zini yozish
50. chart1->Series["Series2"]->XValueMember = "Oy//chart1
    komponentasidagi
51. //Serie1 ning Y kordinatasiga Sotib olingan max.soni so'zini
    yozish
52. chart1->Series["Series1"]->YValueMembers = "Sotib olingan
    max.soni";
53. // chart1 komponentasidagi Serie2 ning Y kordinatasiga Sof
    daromat so'zini yozish
54. chart1->Series["Series2"]->YValueMembers = "Sof daromat";
55. chart1->Series["Series1"]->ChartType = System::Windows::Forms::
56. DataVisualization::Charting::SeriesChartType::Column;
57. chart1->Series["Series2"]->ChartType = System::Windows::Forms::
58. DataVisualization::Charting::SeriesChartType::Column;
59. chart1->Series["Series1"]->Color = Color::Aqua;
60. chart1->Series["Series2"]->Color = Color::Green;
61. //chart1->Series["Series1"]->IsVisibleInLegend = false;
62. chart1->DataBind();
63. //DadaGridView1 komponentasiga Jadvalni eksport qilish
64. dataGridView1->DataSource = Jadval;        }
```

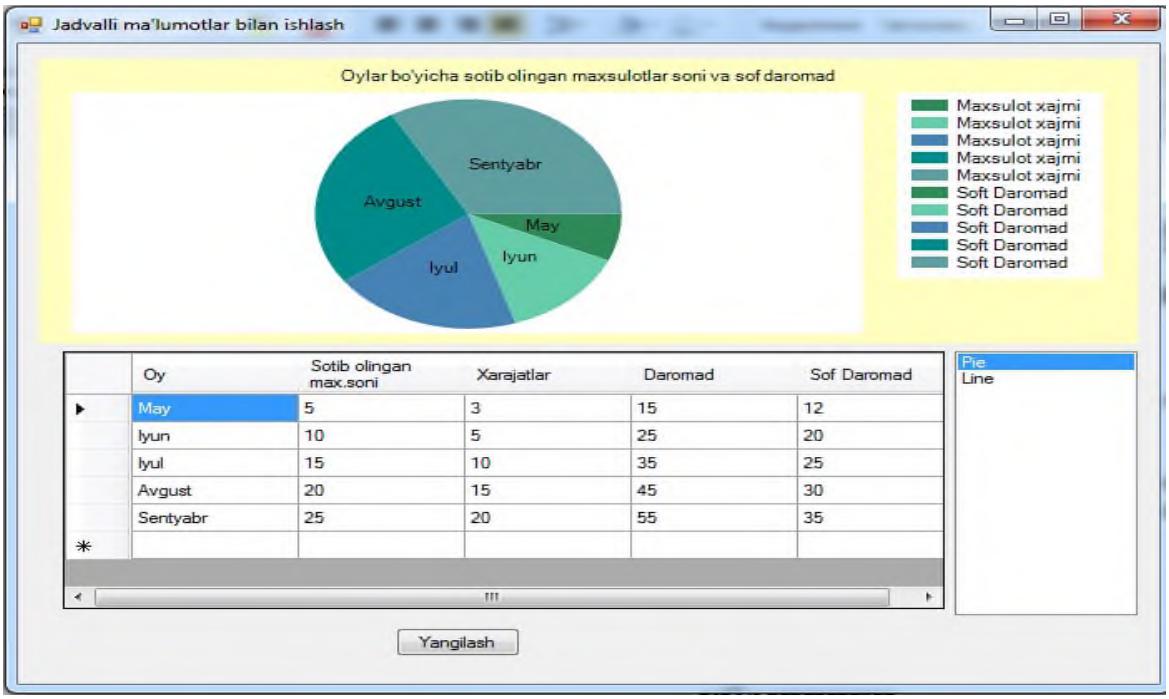
4-qadam. Jadvaldan kiritilgan qiymatlar natijaga darxol ta`sir qilishi uchun **dataGridView1** ning **CellEndEdit** xodisasiga **dio()** funksiyasi chaqiriladi. Ahamiyat berilgan bo‘lsa, **button1** ga ham chaqirilgan edi. Ushbu kod quyidagicha:

```
1. private: System::Void dataGridView1_CellEndEdit(System::Object^
2. sender, System::Windows::Forms::DataGridViewCellEventArgs^  e) {
3. dio();     }
```

5-qadam. listBox1da diagrammani ko‘rinishlarini o‘zgartirish mumkin. Buning uchun **listBox** ning **SelectedIndexChanged** hodisasining kodlari quyidagicha bo‘lishi kerak:

```
1. private: System::Void
   listBox1_SelectedIndexChanged(System::Object^
2. sender, System::EventArgs^  e) {
3. if (listBox1->Text == "Transparent") return;
4. this->chart1->Series["Series1"]->ChartTypeName=(listBox1->Text) ;
5. this->chart1->Series["Series2"]->ChartTypeName=(listBox1->Text) ;
   }
```

Dasturda diagrammani ko‘rinishi quyidagicha:



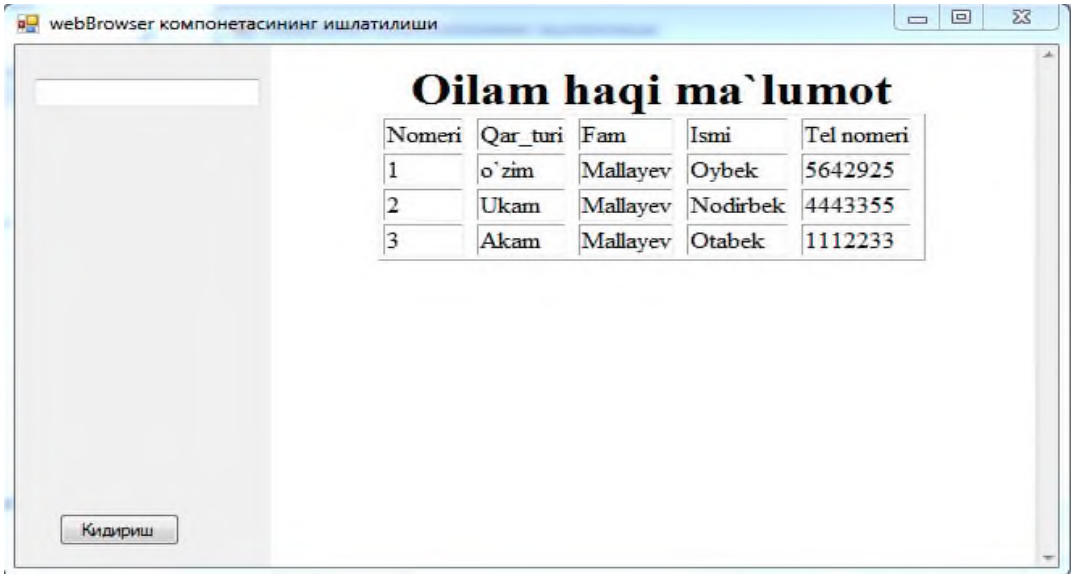
3.39- rasm. Diagrammani sozlash oynasi

3.20. Veb brouzerda HTML jadvallarni tasvirlash va shakllantirish

Bizga ma`lumki HTML teglari yordamida veb ilovalar yaratiladi va bu ilovalar jamlanib, veb sayt tashkil etiladi. Veb ilovalarni yoki global tarmoqga ulangan saytlarni amaliy dasturda ochish imkonini beruvchi komponenta bu WebBrowser komponentasi hisoblanadi. Ushbu komponenta orqali nafaqat saytlarni ochish, balki HTML kodlarida terilgan tekst (teg) larni ham chiqarish va tahrirlash imkoniyatlari mavjud. Quyidagi namunada tuziladigan dasturda HTML teglari ishlatilgan. Masalan: <h1>-matnni 16 shriftda ko‘rsatadi. <table>- jadval yaratadi. <tr>-jadvalning ustunini yaratadi. <td>-jadvalning satrlarini yaratadi. 8- jadvalda keltirilgan ma`lumotlar matnli

faylda tashkil qilinadi. Ushbu fayl o‘qish uchun ochiladi va uning ma`lumotlari ma`lum o‘zgaruvchilarga o‘zlashtirilgan holda, sikl asnosida **webBrowser** komponentasi yordamida ekranga tartiblangan ko‘rinishida chiqariladi.

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 “**HTML_xujjatlar**” nomi berilib **OK** tugmasi bosiladi. Dasturning umumiy ko‘rinishi quyidagicha bo‘ladi:



3.30- rasm. Dasturning umumiy ko‘rinish oynasi

Dasturni tuzish quyidagi qadamlardan tashkil topgan:

1-qadam. Formaga **webBrowser** komponentasini tashlang va **Form1**ning **load** hodisasining kodlari quyidagicha:

```
#pragma endregion
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
String^s="<h1 align=\"center\">Oilam haqi ma`lumot<br><table
border=\"1\"> ";
s+="<td>Nomeri<td>";  s+="<td>Qar_turi<td>";
s+="<td>Fam<td>";      s+="<td>Ismi<td>";
s+="<td>Tel nomeri<td>";
char qar_turi[20]=""; char Ismi[20]="";
char Fam[20]="";  char Tel[20]="";
     String^ss=""; FILE *oqi; int i=1,n; bool dd;
     oqi=fopen("C:\\1.txt","r");
     if(oqi==NULL){MessageBox::Show("Not found");}
     while(!feof(oqi)){
```

```
14.      fgets(qar_turi,20,oqi); fgets(Fam,20,oqi);
15.      fgets(Ismi,20,oqi); dd=fgets(Tel,20,oqi);
16.      if(dd==0){break;}
17.      String^ Fq=gcnew String(qar_turi);
18.      String^ Fi=gcnew String(Ismi);
19.      String^ Fm=gcnew String(Fam);
20.      String^ Ft=gcnew String(Tel);
21.      s+="<tr><td>"+i.ToString()+"<td>"; s+="<td>"+Fq+"<td>";
22.      s+="<td>"+Fm+"<td>"; s+="<td>"+Fi+"<td>";
23.      s+="<td>"+Ft+"<td>"; i++;} s+="</table> ";
24.      webBrowser1->Navigate("about:" + s); }
```

2-qadam. Formada saytlarni qidirish uchun **Button** va **TextBox** komponentalari joylashtiriladi va **Button1** tugmasining **click** hodisasining kodlari quyidagicha:

```
1. private: System::Void button1_Click(System::Object^  sender,
2. System::EventArgs^ e) { if (!this->textBox1->Text->Equals("") ){
3. this->webBrowser1->Navigate( this->textBox1->Text );} }
```

3-qadam. Manzillar oynasiga veb sayt manzili yoziladigan **TextBox1** hodisasi quyidagicha bo‘ladi:

```
1. private:System::Void textBox1_KeyDown(System::Object^ sender,
2. System::Windows:: Forms::KeyEventArgs^  e) {
3. if ( e->KeyCode == System::Windows::Forms::Keys::Enter &&
4. !this->textBox1->Text->Equals( "" ) ){ this->webBrowser1-
   >Navigate(
5. this->textBox1->Text );}
```

4-qadam. webBrowser1 komponentasining **Navigated** hodisasining kodlari quyidagicha:

```
1. private: System::Void webBrowser1_Navigated(System:: Object^
   sender,System::Windows::Forms:: WebBrowserNavigatedEventArgs^  e)
   {
2. this->textBox1->Text = this->webBrowser1->Url->ToString();  }
```

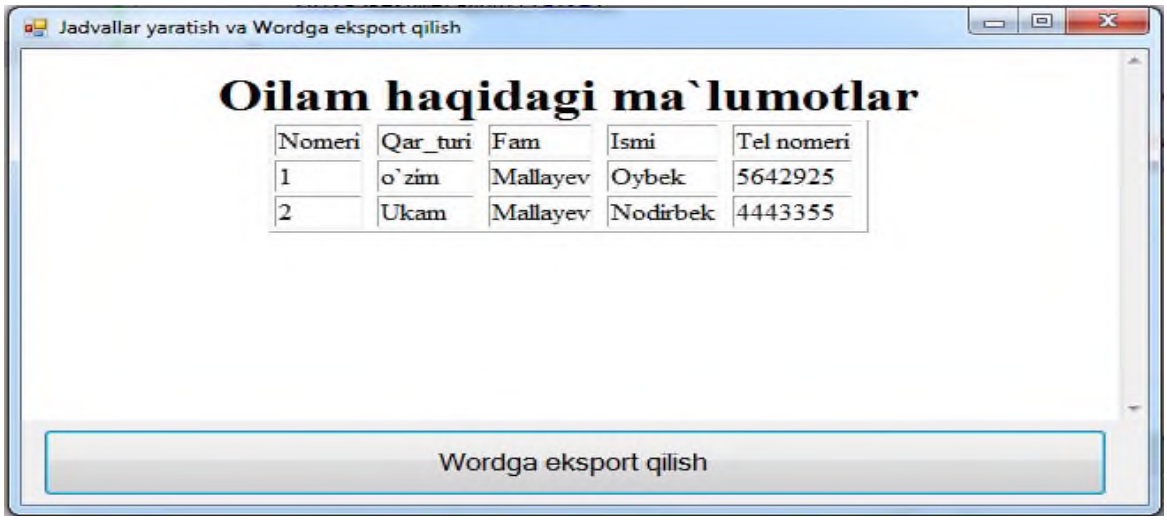
Agar kompyuter internetga ulangan bo‘lsa, sayt manzili maydoniga yozilgan sayt ochiladi.

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

Visual C++ da **Microsoft MS Office** paketidagi amaliy dasturlarning imkoniyatlaridan foydalanish imkoniyatlari mavjud.

Ushbu qo‘llanmaning 2-bobida Visual Basic tilidagi InputBox kiritish oynasi imkoniyatlaridan foydalanilgan edi. Ushbu bo‘limda esa Microsoft MS Office ning jadval yaratish usullari imkoniyatlaridan qanday foydalanish yo‘llari o‘rganiladi. Ish **Visual Studio 2012** dasturini ishga tushirishdan boshlanadi. File menyu bo‘limidan **New Projec**t tanladi va yangi loyiha yaratiladi. Ushbu loyihaga **MS Word** ning ob`ektlar kutubxonasi qo‘shiladi. Buning uchun **Project** menyu bo‘limidan **Add Reference** buyrug‘i tanlanadi yoki **Alt+F7** tugmalari bosiladi. Ochilgan **Property Pages** oynadan **Com** bo‘limiga o‘tiladi va **Add new Reference** tugmasi bosiladi. Ochilgan oynadan **Microsoft Word 14.0 Object Library 8.5** belgilanadi va OK tugmachasi bosiladi. So‘ngra ushbu kutubxona muvaffaqiyatli qo‘shilganligi haqida ma`lumot chiqadi. Natijada, dasturda ushbu kutubxona funksiyalaridan foydalanish imkoniyati vujudga keladi.

Dasturning umumiy ko‘rinishi quyidagicha bo‘ladi:



3.31- rasm. Dasturning umumiy ko‘rinishi oynasi

Dasturni tuzish quyidagi qadamlardan tashkil topgan:

1-qadam. Form1ga asosiy dastur oynasida matnli fayl ma`lumotlarini chiqarish uchun **webBrowser** komponentasi tashlanadi va **Form1** ning **load** hodisasi kodlari quyidagicha bo‘ladi:

```
#pragma endregion
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
button2->Text = "Wordga eksport qilish";
this->Text = "Jadvallar yaratish va Wordga eksport qilish";
String^s="<h1 align=\"center\">Oilam haqidagi
ma`lumotlar<br><table
```

```
border=\"1\"> "; s+="<td>Nomeri<td>";
s+="<td>Qar_turi<td>";
s+="<td>Fam<td>";     s+="<td>Ismi<td>"; s+="<td>Tel nomeri<td>";
char qar_turi[20]=""; char Ismi[20]="";       char Fam[20]="";
      char Tel[20]="";         String^ss="";
      /* fayllar bilan ishlash oqimi. fayldagi ma`lumotlarni
o‘zlashtirish uchun 4 ta char turidagi massiv e`lon qilish*/
      FILE *oqi;  int i=1,n;
      bool dd;  oqi=fopen("C:\\word.txt","r");
      // faylga yo‘l ko‘rsatish
      if(oqi==NULL){MessageBox::Show("Fayl topilmadi");}
      while(!feof(oqi)){
      // Matnli fayldan ma`lumotlarni o‘zlashtirish
      fgets(qar_turi,20,oqi); fgets(Fam,20,oqi);
      fgets(Ismi,20,oqi);    dd=fgets(Tel,20,oqi);
      if(dd==0){break;}
      // Matnli faylda ohirgi ma`lumotni aniqlash
      // char turidan String turiga o‘tkazish
      String^ Fq=gcnew String(qar_turi);     String^ Fi=gcnew
      String(Ismi);
      String^ Fm=gcnew String(Fam);     String^ Ft=gcnew
      String(Tel);
      // s o‘zgaruvchisiga o‘zlashtirish
      s+="<tr><td>"+i.ToString()+"<td>";
      s+="<td>"+Fq+"<td>";  s+="<td>"+Fm+"<td>";
      s+="<td>"+Fi+"<td>";   s+="<td>"+Ft+"<td>";i++;       }
      s+="</table> ";  webBrowser1->Navigate("about:" + s); }
```

2-qadam. **Form1**ga asosiy dastur oynasida matnli fayl ma`lumotlarini Word faylga eksport qilish uchun **Button** komponentasini tashlang va uning **OnClick** xodisasi kodlari quyidagicha bo‘ladi:

```
private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
// 4 ta satrli massiv e`lon kilish
array<String^> ^ Fi={"Ismi",
"","","","","","","","","","","","","","","","","","","","","","",""};
array<String^> ^ Ff={"Familiyasi"
,"","","","","","","","","","","","","","","","","","","","","","",};
array<String^> ^ Ft={"Telefoni",
"","","","","","","","","","","","","","","","","","","","","","",""};
array<String^> ^
Fq={"Qarin_turi","","","","","","","","","","","","","","","","","",};
FILE *oqi; /* fayllar bilan ishlash oqimi/fayldagi ma`lumotlarni
o‘zlashtirish uchun 4 ta char turidagi massiv e`lon qilish*/
char qar_turi[20]=""; char Ismi[20]="";
char Fam[20]="";      char Tel2[20]="";
int i=1,n;            bool dd;
oqi=fopen("C:\\word.txt","r");
```

```
// faylga yo‘l ko‘rsatish
if(oqi==NULL){MessageBox::Show("Fayl topilmadi");}
while(!feof(oqi)){
// Matnli fayldan ma`lumotlarni o‘zlashtirish
fgets(qar_turi,20,oqi);   fgets(Fam,20,oqi);
fgets(Ismi,20,oqi);       dd=fgets(Tel2,20,oqi);
if(dd==0){break;}
/*Matnli faylda ohirgi ma`lumotni aniqlash char turidan Stting turiga
o‘tkazish va massiv elementlariga o‘zlashtirish*/
Fq-> SetValue(gcnew String(qar_turi),i); Ff-> SetValue(gcnew
String(Fam),i);
Fi-> SetValue(gcnew String(Ismi),i); Ft-> SetValue(gcnew
String(Tel2),i); i++;}
// Word::_Application sinfi nushasini yaratish:
auto Vord1 = gcnew Microsoft::Office::Interop::Word::
Application();
     Vord1->Visible = true;
     // "bo‘sh" qiymatli o‘zgaruvchi:
     auto t = Type::Missing;
     // Yangi MS Word xujjat ochish:
     auto Dokument = Vord1->Documents->Add(t, t, t, t);
     // MS WORD xujjatiga joriy xolati bilan matnlar kiritish:
     Vord1->Selection->TypeText("Oilam haqida ma'lumotlar");
     // Katakchalar parametri va ularni chegaralarini
//ko‘rsatish:
     System::Object ^ t1 = Microsoft::Office::Interop::
     Word::WdDefaultTableBehavior::wdWord9TableBehavior;
     //Jadvaldagi katakchalar o‘lchamini o‘zgartirish:
     System::Object ^ t2 = Microsoft::Office::Interop::
     Word::WdAutoFitBehavior::wdAutoFitContent;
     // i ta satr va 5 ta ustunli jadval yaratish:
     Vord1->ActiveDocument->Tables->Add(Vord1->Selection->Range,
     i, 5, t1, t2);  int ii=i;
     // Jadval katakchalarini to‘ldirish:
     for (int i = 1; i <= ii; i++) {
     if(i>=2)Vord1->ActiveDocument->Tables[1]->Cell(i, 1)->
     default->InsertAfter((i-1).ToString());
     Vord1->ActiveDocument->Tables[1]->Cell(i, 2)->
     default->InsertAfter(Fq[i - 1]);
     Vord1->ActiveDocument->Tables[1]->Cell(i, 3)->
     default->InsertAfter(Ff[i - 1]);
     Vord1->ActiveDocument->Tables[1]->Cell(i, 4)->
     default->InsertAfter(Fi[i - 1]);
     Vord1->ActiveDocument->Tables[1]->Cell(i, 5)->
     default->InsertAfter(Ft[i - 1]);
     // C# tilida, quyidagicha yoziladi:
     // Vord1.ActiveDocument.Tables[1].Cell(i, 2).
     // Range.InsertAfter(Tel[i - 1]);
     }
     // MS Word xujjatlari ilovalarini belgilash:
```

```
64.     Object^t3=Microsoft::Office::Interop::Word::WdUnits::wdLine;
65.     //MS Word xujjatiga i ta satrni aniqlash :
66.     Object ^ stroka9 = i;
67.     Vord1->Selection->MoveDown(t3, stroka9, t);
68.     // Quyidagi matnni chop qilish:
69.     Vord1->Selection->TypeText("Kafedra mudiri N.Rahimov");
70.     /*Xujjatni avtomatik saqlash, lekin buni foydalanuvchi hal
   qilgani maqul*/:
71.     Object ^ ImyaFayla = "C:\\Natija.doc";
72.     Vord1->ActiveDocument->SaveAs(ImyaFayla, t, t, t, t, t, t,
   t,); }
```

Dastur tahlili:

Ushbu dastur kodida 4 ta **String** tipidagi massiv e`lon qilindi. Dasturning kutubxona e`lon qilish bo‘limiga fayllar bilan ishlash direktivasi (**#include<fstream>)** qo‘shildi.

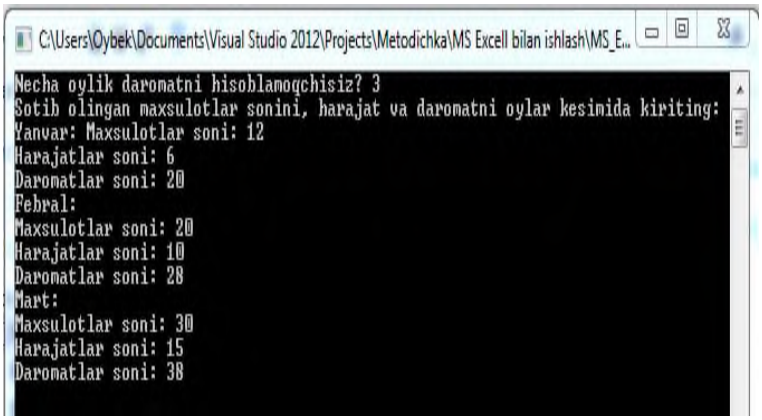
Word::Application sinf ob`ekti yaratildi. (**auto Vord1 =gcnew Microsoft::Office::Interop::Word::Application();)** va uning **Add** metodan foydalanib **Word** fayl ochildi. Matnli fayldagi ma`lumotlarni **FILE** tipidagi ko‘rsatgich yordamida 4 ta massivga o‘zlashtirildi. Massivdagi ma`lumotlar (**Vord1->ActiveDocument->Tables[1]->Cell(i,2)->default->InsertAfter(Fq[i - 1]);**) faylga o‘zlashtirildi.

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

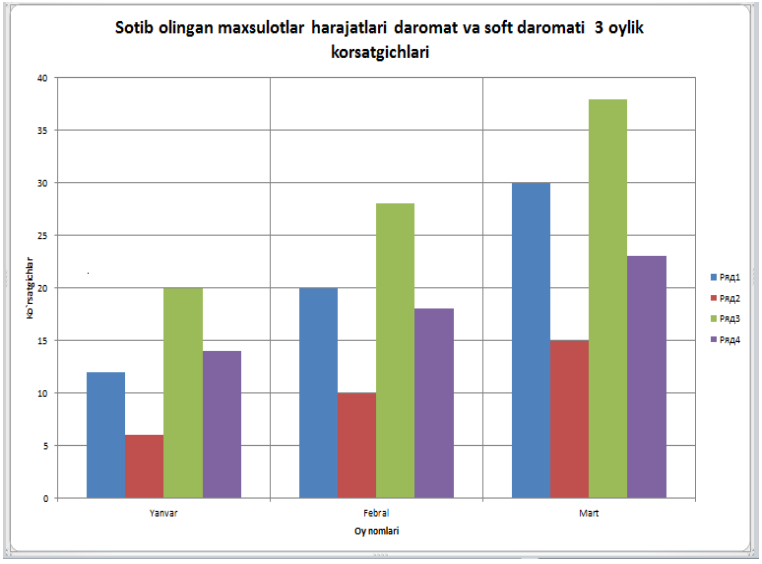
Visual C++ da **Microsoft MS Office** paketidagi amaliy dasturlarning imkoniyatlaridan foydalanish imkoniyatlari mavjud. Jumladan, jadvallar redaktori **MS Excell** da murakkab hisoblashlarni amalga oshirish imkoniyatlari mavjud. Ya`ni, dasturlash imkoniyatlarini ham oshiradi. Ushbu imkoniyatlardan qanday foydalanishni ushbu bo‘limda ko‘rib chiqiladi. Jumladan, **Microsoft MS Excell** ning diagrammalar yaratish usullari imkoniyatlaridan foydalanish yo‘llari o‘rganiladi. Ba`zi hisoblashlarni diagrammalar orqali namoyish qilish dasturning vizualligini orttiradi. Bu esa har doim **MS Excell** ning ma`lumotlari o‘zgarganda vizual diagrammani ham o‘zgarganini ko‘rish mumkin. Ish **Visual Studio** dasturini ishga tushirishdan boshlanadi. **File** menyu bo‘limidan **New Project** tanladi va yangi loyiha yaratiladi. Ushbu loyihaga **MS Excell** ning ob`ektlar kutubxonasi qo‘shiladi. Buning uchun **Project** menyu bo‘limidan **Add Reference** burug‘i

tanlanadi yoki **Alt+F7** tugmalari bosiladi. Ochilgan **Property Pages** oynasidan **Com** bo‘limiga o‘tiladi va **Add new Reference** tugmasi bosiladi. Ochilgan oynadan **Microsoft Excell 14.0 Object Library 8.5** belgilanadi va OK tugmachasi bosiladi. So‘ngra ushbu kutubxona muvaffaqiyatli qo‘shilganligi haqida ma`lumot chiqadi. Natijada, dasturda ushbu kutubxona funksiyalaridan foydalanish imkoniyati vujudga keladi.

Dasturning umumiy ko‘rinishi quyidagicha bo‘ladi:



3.32- rasm. Dasturning kompilyatsiya vaqtidagi oynasi



3.33- rasm. Dasturning kompilyatsiyadan keyingi diagramma oynasi

Dasturni tuzish quyidagi qadamlardan tashkil topgan:

1-qadam. Dasturga forma muhiti talab qilinmaganligi uchun loyihani **Console Application** da tuziladi**. Visual C++** dasturi ishga tushiriladi. **File** menyu bo‘limidan **New Project** tanladi, **CLR** qismidan **Console Application CLR** muhitida yangi loyiha yaratiladi. MS Excell

kutubxonasi qo‘shiladi (**Add Reference->Com->Add new Reference->Microsoft Excell 14.0 Object Library 8.5**). Dastur kodlari quyidagicha bo‘ladi:

```
// MS_Excell_bilan_ishlash.cpp : main project file.
#include "stdafx.h"
#include "iostream";
using namespace System;
using namespace Microsoft::Office::Interop::Excel;
using namespace std;
int main(array<System::String ^> ^args) {
double harajat[12], daromat[12], Sof_daromat[12], maxsulot_soni[12];
int n; bir:Console::Write("Necha oylik daromatni hisoblamoqchisiz?");
std::cin>>n; Console::WriteLine("Sotib olingan maxsulotlar
     sonini, harajat va daromatni oylar kesimida kiriting:");
if(n<=12){
for(int i=0;i<n;i++){
if(i==0){Console::Write("Yanvar: ");}
else if(i==1){Console::WriteLine("Febral: ");}
else if(i==2){Console::WriteLine("Mart: ");}
else if(i==3){Console::WriteLine("Aprel: ");}
else if(i==4){Console::WriteLine("May: ");}
else if(i==5){Console::WriteLine("Iyun: ");}
else if(i==6){Console::WriteLine("Iyul: ");}
else if(i==7){Console::WriteLine("Avgust: ");}
else if(i==8){Console::WriteLine("Sentyabr: ");}
else if(i==9){Console::WriteLine("Oktyabr: ");}
else if(i==10){Console::WriteLine("Noyabr: ");}
else if(i==11){Console::WriteLine("Dekabr: ");}
Console::Write("Maxsulotlar soni: "); std::cin>>maxsulot_soni[i];
Console::Write("Harajatlar soni: ");std::cin>>harajat[i];
Console::Write("Daromatlar soni: ");std::cin>>daromat[i];
Sof_daromat[i]= daromat[i]-harajat[i];
if(i==n-1){ //  Excel::Application  klasidan nusxa olish:
_Application ^ XL1 = gcnew Application();
XL1->Visible = true;
// mavjud usulda foydalanish uchun parametr berish:
Object^t = Type::Missing;
// MS Excel da yangi varoq yaratish:
Workbook ^ Kniga = XL1->Workbooks->Add(t);
// Varoqni faylda e`lon qilish:
Sheets ^ Listi = Kniga->Worksheets;
// Birinchi listni tanlash:
_Worksheet ^ List = (_Worksheet ^)Listi->Item[1];
// Agar yana bitta list qo‘shishni kerak bo‘lsa quyidagi
//kodni terish lozim:
// _Worksheet^List = safe_cast<_Worksheet^>(List->Item[ (Object^)1 ]);
//Ma`lumotlarni joriy parametrlari bilan Diagramma yaratishga buyurtma berish:
_Chart ^ Grafik = (_Chart ^)XL1->Charts->Add(t, t, t, t);
if(n>=1){
List->Range["A1", t]->Value2 = "Oylar nomi";
List->Range["A2", t]->Value2 = "Yanvar";
List->Range["B1", t]->Value2 = "Sotib olingan maxsulot soni";
```

```
List->Range["B2", t]->Value2 = maxsulot_soni[0];
List->Range["C1", t]->Value2 = "Harajatlar";
List->Range["C2", t]->Value2 = harajat[0];
List->Range["D1", t]->Value2 = "Daromatlar";
List->Range["D2", t]->Value2 = daromat[0];
List->Range["E1", t]->Value2 = "Soft Daromatlar";
List->Range["E2", t]->Value2 = Sof_daromat[0];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E2"],
XlRowCol::xlColumns); }
if(n>=2){List->Range["A3", t]->Value2 = "Febral";
List->Range["B3", t]->Value2 = maxsulot_soni[1];
List->Range["C3", t]->Value2 = harajat[1];
List->Range["D3", t]->Value2 = daromat[1];
List->Range["E3", t]->Value2 = Sof_daromat[1];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E3"],
XlRowCol::xlColumns); }
if(n>=3){
List->Range["A4", t]->Value2 = "Mart";
List->Range["B4", t]->Value2 = maxsulot_soni[2];
List->Range["C4", t]->Value2 = harajat[2];
List->Range["D4", t]->Value2 = daromat[2];
List->Range["E4", t]->Value2 = Sof_daromat[2];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E4"],
XlRowCol::xlColumns); }
if(n>=4){List->Range["A5", t]->Value2 = "Aprel";
List->Range["B5", t]->Value2 = maxsulot_soni[3];
List->Range["C5", t]->Value2 = harajat[3];
List->Range["D5", t]->Value2 = daromat[3];
List->Range["E5", t]->Value2 = Sof_daromat[3];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E5"],
XlRowCol::xlColumns); }
if(n>=5){
List->Range["A6", t]->Value2 = "May";
List->Range["B6", t]->Value2 = maxsulot_soni[4];
List->Range["C6", t]->Value2 = harajat[4];
List->Range["D6", t]->Value2 = daromat[4];
List->Range["E6", t]->Value2 = Sof_daromat[4];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E6"],
XlRowCol::xlColumns); }
if(n>=6){
List->Range["A7", t]->Value2 = "Iyun";
List->Range["B7", t]->Value2 = maxsulot_soni[5];
List->Range["C7", t]->Value2 = harajat[5];
List->Range["D7", t]->Value2 = daromat[5];
List->Range["E7", t]->Value2 = Sof_daromat[5];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E7"],
XlRowCol::xlColumns); }
if(n>=7){
```

```
List->Range["A8", t]->Value2 = "Iyul";
List->Range["B8", t]->Value2 = maxsulot_soni[6];
List->Range["C8", t]->Value2 = harajat[6];
List->Range["D8", t]->Value2 = daromat[6];
List->Range["E8", t]->Value2 = Sof_daromat[6];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E8"],
XlRowCol::xlColumns); }
if(n>=8){
List->Range["A9", t]->Value2 = "Avgust";
List->Range["B9", t]->Value2 = maxsulot_soni[7];
List->Range["C9", t]->Value2 = harajat[7];
List->Range["D9", t]->Value2 = daromat[7];
List->Range["E9", t]->Value2 = Sof_daromat[7];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E9"],
XlRowCol::xlColumns); }
if(n>=9){
List->Range["A10", t]->Value2 = "Sentyabr";
List->Range["B10", t]->Value2 = maxsulot_soni[8];
List->Range["C10", t]->Value2 = harajat[8];
List->Range["D10", t]->Value2 = daromat[8];
List->Range["E10", t]->Value2 = Sof_daromat[8];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E10"],
XlRowCol::xlColumns); }
if(n>=10){
List->Range["A11", t]->Value2 = "Oktyabr";
List->Range["B11", t]->Value2 = maxsulot_soni[9];
List->Range["C11", t]->Value2 = harajat[9];
List->Range["D11", t]->Value2 = daromat[9];
List->Range["E11", t]->Value2 = Sof_daromat[9];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E11"],
XlRowCol::xlColumns); }
if(n>=11){
List->Range["A12", t]->Value2 = "Noyabr";
List->Range["B12", t]->Value2 = maxsulot_soni[10];
List->Range["C12", t]->Value2 = harajat[10];
List->Range["D12", t]->Value2 = daromat[10];
List->Range["E12", t]->Value2 = Sof_daromat[10];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E12"],
XlRowCol::xlColumns); }
if(n==12){
List->Range["A13", t]->Value2 = "Dekabr";
List->Range["B13", t]->Value2 = maxsulot_soni[11];
List->Range["C13", t]->Value2 = harajat[11];
List->Range["D13", t]->Value2 = daromat[11];
List->Range["E13", t]->Value2 = Sof_daromat[11];
//Grafika qurish uchun qiymatlar diapazonini berish:
Grafik->SetSourceData(List->Range["A2", "E13"],
XlRowCol::xlColumns); }
//Grafikni turini berish("stolbikovaya diagramma" (gistogramma)):
```

```
157.Grafik->ChartType = XlChartType::xlColumnClustered;
158.//Grafikni legandasini ekranga chiqarish:
159.Grafik->HasLegend = true;
160.// Grafikning sarlavhasini ekranga chiqarish:
161.Grafik->HasTitle = true;
162.Grafik->ChartTitle->Caption = "Sotib olingan maxsulotlar
163.harajatlari daromat va soft daromati  "+n.ToString()+" oylik
164.korsatgichlari";
165.//  osi X bo‘yicha imzo:
166.Axis ^ GorizontalnayaOs = (Axis^)Grafik->Axes(
167.XlAxisType::xlCategory, XlAxisGroup::xlPrimary);
168.GorizontalnayaOs->HasTitle = true;
169. GorizontalnayaOs->HasMajorGridlines=true;
170.GorizontalnayaOs->AxisTitle->Text = "Oy nomlari";
171.//  Y o’qi bo‘yicha imzo:
172.Axis ^ VertikalnayaOs = (Axis^)Grafik->Axes(
173.XlAxisType::xlValue, XlAxisGroup::xlPrimary);
174.VertikalnayaOs->HasTitle = true;
175.VertikalnayaOs->AxisTitle->Text = "Ko`rsatgichlar";
176.//Grafikni quyidagi kengaytmali fayl ko‘rinishida saqlash:
177.XL1->ActiveChart->Export("C:g`g`ExcelGrafik.jpg",t,t);
178.}    }        } else {Console::Write("1 yilda 12 oy bor
179.Qaytadan urunib ko`ring"); goto bir;} return 0;  }
```

Dastur tahlili:

Dasturda C++ ning kiritish (**cin**) operatori, **Visual C++** ning chiqarish (**Console::WriteLine()**) operatori**, Excel::Application** sinfidan nusxa olish**(_Application ^ XL1 = gcnew Application();**) operatori, **MS Excell** da yangi varoq yaratish (**Workbook**) operatori, varaqni faylda e`lon qilish (**Sheets**) operatori, **Excell** listini tanlash (**_Worksheet**) operatori, diagrammani yaratish (**_Chart**) operatori va diagrammani turli parametrlarini o‘zgartiruvchi bir nechta funksiyalar ishlatilgan. Algoritmi quyidagicha: 4 ta massiv (**harajat**[12], **daromat**[12], **Sof_daromat**[12], **maxsulot_soni**[12]) yaratilgan, ushbu massivlarga konsol muhitidan qiymat o‘zlashtiriladi, ushbu massiv qiymatlarini **Excell** ning varag‘idagi katakchalarga tartib bilan yoziladi, kataklar diapazoni belgilanadi va (**_Chart**) operatori yordamida grafik quriladi.

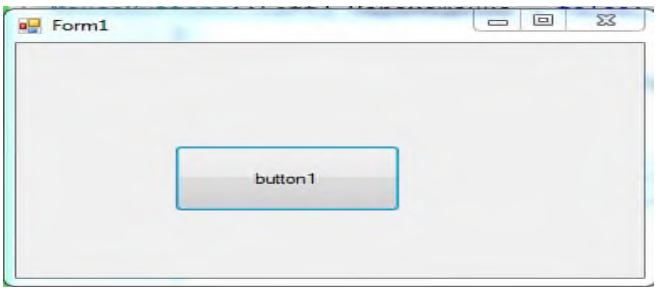
3.23. Visual C++ ning Windows Application muhitida komponentalarning joylashish vaziyatlarini nazorat qilish

1-dastur: Formada komponentalar dasturninng tashqi ko‘rinishlarini, unumdorligini oshirishda muhim ahamiyat kasb etadi. Bizga ma`lumki, dasturda ishlatiladigan komponentalarning joylashish

pozisiyalarini kompilyatsiyadan oldin o‘zgartirish mumkin. **Visual C++** muharririda kompilyatsiyadan keyin ham o‘zgartirish imkoniyati mavjud. Hozirgi yaratiladigan dastur, **Button** komponentasini forma bo‘ylab ko‘chirish imkoniyatlarini ochib beradi. Buning uchun **Windows Form Application** muhitida yangi loyiha yaratiladi. Formaga bitta **Button** komponentasi joylashtiriladi. Dastur kodlari quyidagicha:

```
#pragma endregion
bool Peremehenie;
int MouseDownX;  int MouseDownY;
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) { Peremehenie = false;}
private: System::Void button1_MouseDown(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) { if (e->Button ==
System::Windows::Forms::
MouseButtons::Left) { Peremehenie = true; MouseDownX = e->X;
MouseDownY = e->Y; }   }
private: System::Void button1_MouseUp(System::Object^
sender,System::Windows::Forms::MouseEventArgs^  e) {
if (e->Button == System::Windows::Forms::
MouseButtons::Left) Peremehenie = false; }
private: System::Void button1_MouseMove(System::Object^
sender,System::Windows::Forms::MouseEventArgs^  e) {
if (Peremehenie == true) {
auto Tochka = System::Drawing::Point();
Tochka.X = this->button1->Location.X + (e->X - MouseDownX);
Tochka.Y = this->button1->Location.Y + (e->Y - MouseDownY);
this->button1->Location = Tochka; } }
```

Dasturning tashqi ko‘rinishi quyidagicha:



3.34- rasm. Dastur natijasi

Dastur tahlili:

Dasturda uchta global o‘zgaruvchilar (**bool Peremeshenie;int MouseDownX;int MouseDownY;)** e`lon qilindi**. bool Peremeshenie** – mantiqiy o‘zgaruvchi bo‘lib, komponentani ko‘chirishga ruxsat bor yoki

yo‘qligini aniqlash uchun ishlatildi. Agar foydalanuvchi **Button1** tugmasini sichqonchaning chap tugmasi bilan bossa (**MouseDown, e->Button == Left),** u holda **Peremeshenie = true** bo‘ladi. Bu esa Button1 tugmasini ko‘chirish mumkinligigi anglatadi. Agar foydalanuvchi sichqonchaninig tugmasini qo‘yib yuborsa, (Vaziyat **MouseUp**) u holda **Peremeshenie = true** bo‘ladi. Qolgan 2 ta (**int MouseDownX; int MouseDownY;)** o‘zgaruvchilar joriy vaziyatni to‘ldirish uchun mo‘ljallangan.

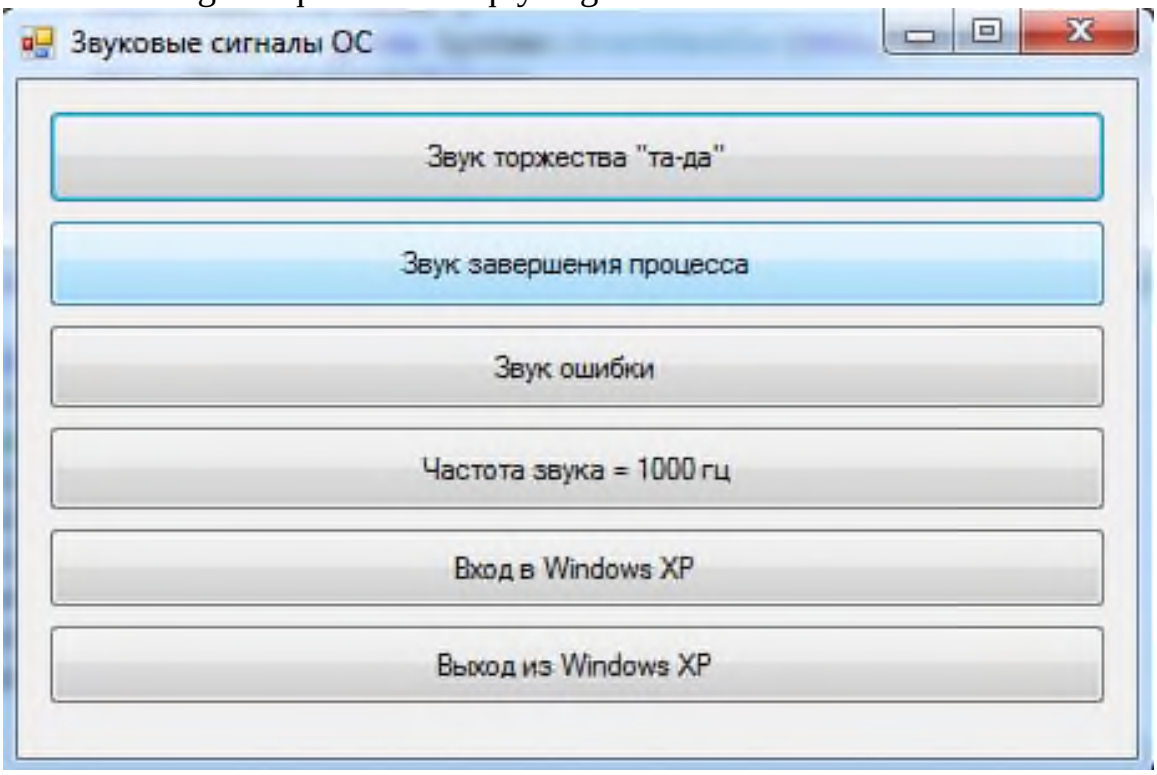
2-dastur: Operasion tizimning maxsus ovozli fayllarini yuklovchi vizual dastur

Bizga ma`lumki, operasion tizimda ishlaganda xatoliklar, amallarni yoki operasiyalarni bajarilishi, dasturlarni o‘rnatish va x.k.lar haqida ovozlarni eshitiladi. Ushbu ovozli fayllardan joriy dasturimizda foydalanishga harakat qilinadi. Buning uchun **Windows Form Application** muhitida yangi loyiha yaratiladi. Formaga 6 ta **Button** komponentasi joylashtiriladi. Dastur kodlari quyidagicha bo‘ladi:

```
1. #pragma endregion
2. System::Media::SoundPlayer ^ Pleer;
3. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e) {    this->Text = "Zvukovo`e signalo` OS";
4. Pleer = gcnew System::Media::SoundPlayer();
5. button1->Text = "Zvuk torjestva \"ta-dag`"";
6. button2->Text = "Zvuk zaversheniya prosessa";
7. button3->Text = "Zvuk oshibki";
8. button4->Text = "CHastota zvuka = 1000 gs";
9. button5->Text = "Vxod v Windows XP";
10. button6->Text = "Vo`xod iz Windows XP";                }
11. private: System::Void button1_Click(System::Object^  sender,
12. System::EventArgs^  e) {
13. Pleer->SoundLocation = "c:\\windows\\media\\tada.wav";
14. Pleer->Play();       }
15. private: System::Void button2_Click(System::Object^  sender,
16. System::EventArgs^  e) {
17. System::Media::SystemSounds::Asterisk->Play();        }
18. private: System::Void button3_Click(System::Object^  sender,
19. System::EventArgs^  e) {
20. System::Media::SystemSounds::Beep->Play();            }
21. private: System::Void button4_Click(System::Object^  sender,
22. System::EventArgs^  e) {
23. Console::Beep(1000, 500);       }
24. private: System::Void button5_Click(System::Object^  sender,
25. System::EventArgs^  e) {
```

```
26. Pleer->SoundLocation = "C:\\windows\\media\\Windows
    Startup.wav";
27. Pleer->Play();        }
28. private: System::Void button6_Click(System::Object^  sender,
29. System::EventArgs^  e) {
30. Pleer->SoundLocation = "C:\\windows\\media\\Windows User Account
    Control.wav";
31. Pleer->Play(); }
```

Dasturning tashqi ko‘rinishi quyidagicha bo‘ladi:



3.35- rasm. Dastur natijasi

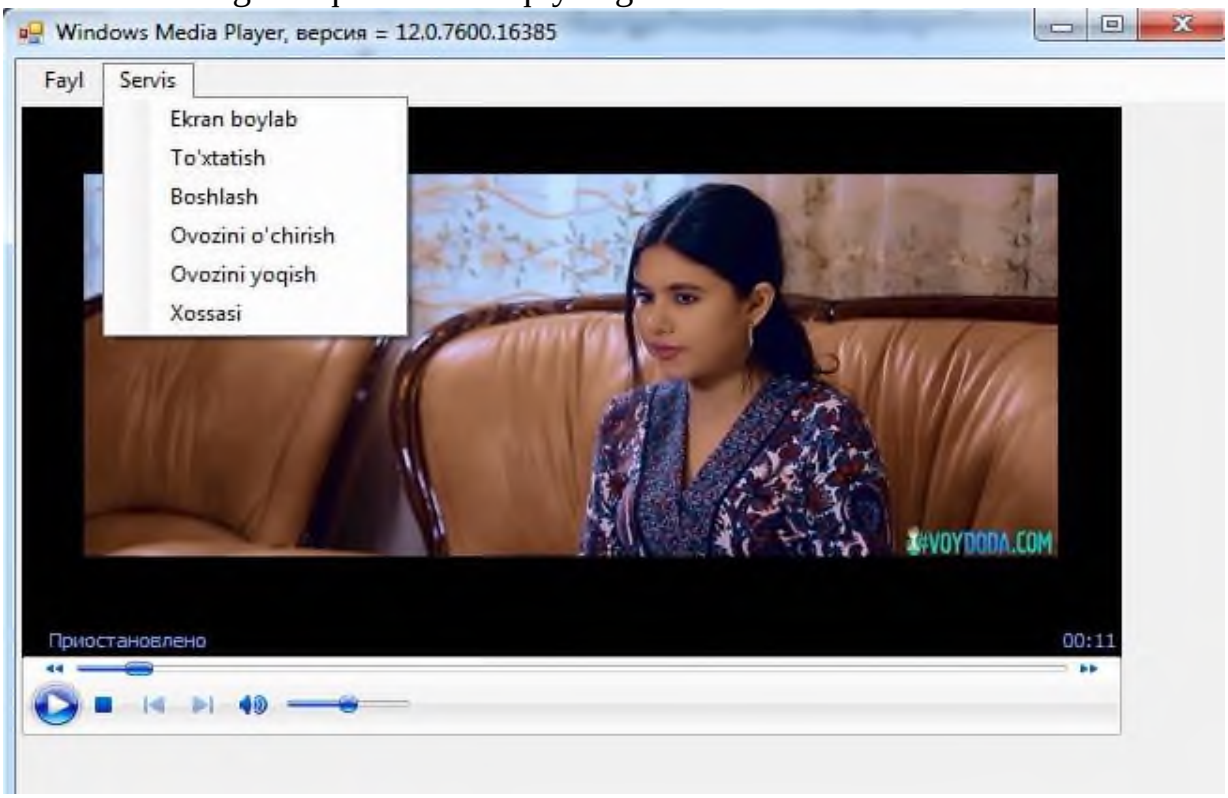
3-dastur: Operasion tizimning video fayllarini yuklovchi dastur.

Ushbu dasturni tuzishda **Windows Media Player** dasturi ishlatiladi. Buni amalga oshirish uchun **Windows Form Application** muhitida yangi loyiha yaratiladi. Formaga **menuStrip** va **Windows Media Player** komponentalari joylashtiriladi. Lekin **ToolsBox** asboblar panelida **Windows Media Player** komponentasi yo‘q. Uni o‘rnatish uchun **ToolsBox** asboblar panelida sichqonchaning o‘ng tugmasi bosiladi va kontekst menyudan **Choose Items** buyrug‘i tanlanadi. Ochilgan oynaning **COM** bo‘limidan **Windows.Media.Player** ni tanlab, **OK** tugmachasi bosiladi. Natijada **ToolsBox** asboblar panelida ushbu komponenta paydo bo‘ladi. **menuStrip** komponentasini sozlash **3.15-rasmda** keltirilgan.

Dastur kodlari quyidagicha:

```
#pragma endregion
OpenFileDialog ^ openFileDialog1;
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {openFileDialog1 = gcnew OpenFileDialog();
//  PLEERni versiyasi
this->Text = "Windows Media Player, versiya = " +
axWindowsMediaPlayer1->versionInfo; }
private: System::Void
ochishToolStripMenuItem_Click(System::Object^
sender, System::EventArgs^  e) { // PUNKT MENYUSI fOYDALANUVCHI
FAYLNI TANLAYDI:
 openFileDialog1->ShowDialog();
 // FAYLNING NOMINI PLEERGA JO‘NATISH
 axWindowsMediaPlayer1->URL = openFileDialog1->FileName;
 // axWindowsMediaPlayer1->URL = "C:\\WINDOWS\\Media\\tada.wav";
 // fayla FAYLNI YUKLASH KOMANDASI
 axWindowsMediaPlayer1->Ctlcontrols->play();
 //axWindowsMediaPlayer1->openPlayer(openFileDialog1->FileName);}
 private: System::Void
 yopishToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) { // KIRISH MENYU bo‘limi
 Application::Exit(); }
 private: System::Void
 ekranBoylabToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) {  // Ekran bo‘ylab MENYU bo‘limi.
 if (axWindowsMediaPlayer1->playState ==
WMPLib::WMPPlayState::wmppsPlaying)
 axWindowsMediaPlayer1->fullScreen = true;       }
 private: System::Void
 tuxtatishToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) {        // To‘xtatish MENYU bo‘limi
 axWindowsMediaPlayer1->Ctlcontrols->pause();          }
 private: System::Void
 boshlashToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) {        // Boshlash MENYU bo‘limi
 axWindowsMediaPlayer1->Ctlcontrols->play();    }
 private: System::Void
 ovoziniOchirishToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) {   // Ovozni yoqish MENYU bo‘limi
 axWindowsMediaPlayer1->settings->mute = true;  }
 private: System::Void
 ovoziniYoqishToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) { // Ovozni o‘chirish MENYU bo‘limi
 axWindowsMediaPlayer1->settings->mute = false; }
 private: System::Void
 xossasiToolStripMenuItem_Click(System::Object^  sender,
 System::EventArgs^  e) { // Fayl xossasi MENYU bo‘limi
 axWindowsMediaPlayer1->ShowPropertyPages(); }
```

Dasturning tashqi ko‘rinishi quyidagicha bo‘ladi:



3.36- rasm. Windows Media Player ni yuklash

Nazorat savollari:

1. **Visual C++** da **MS Excell** fayllariga eksport va import qanday amalga oshiriladi?
2. **Visual C++** da **MS Word** fayllariga eksport va import qanday amalga oshiriladi?
3. **Visual C++** da diagrammalar bilan ishlash komponentalarini bilasizmi?
4. **Visual C++** da satrlar bilan ishlovchi qanday funksiyalar bor?
5. **Visual C++** da jadvalli tip nima?
6. **DataGridWiew** komponentasining vazifasi nima?
7. **Visual C++** da **ovozli** fayllarga qanday murojaat qilinadi?
8. **Visual C++** da **video** fayllarga qanday murojaat qilinadi?
9. **Visual C++** da rasmli fayllarga qanday murojaat qilinadi?
10. **Visual C++** da dastur hisobotlari qanday tayyorlanadi?

Bob xulosasi

Ushbu bobda **Windows Form Application** ilovasini yaratish,

forma (Form) ni formalarga bog‘lash, berilganlarni lug‘at (Dictionary) yordamida strukturali saqlash, bir prosedura orqali bir nechta hodisalarga ishlov berish, klaviatura hodisalarini qayta ishlash, matnli maydonga kiruvchi ma`lumotlarni boshqarish, try … catch istisnosini qayta ishlash, matnli xujjatni chop qilish, formada grafik shakllarni va funksiya grafiklarni hamda sichqoncha ko‘rsatgichi orqali chizish, veb brouzerda HTML jadvallarni tasvirlash, Visual C++da MS Word imkoniyatlaridan foydalanib, jadvallar yaratish, ularni Word faylga eksport qilish, Visual C++ da MS Excell imkoniyatlaridan foydalanib, diagrammalar yaratish va Visual C++ ning Windows Application muhitida komponentalarning joylashish vaziyatlarini nazorat qilish usullari ko‘rib chiqildi. Ushbu usullar asosida amaliy dasturlar yaratish yo‘llari ko‘rsatildi. Har bir tuzilgan amaliy dastur tahlillari keltirildi. Ushbu vazifalarni bajarish uchun quyidagi komponentalardan foydalanildi: Form, Button, Label, TextBox va DataTimePicker, CheckBox, CheckedListBox, ComboBox, ListBox, TabControl, RadioButton, LinkLabel, OpenFileDialog, SaveFileDialog, Chart komponentalari. **MessageBox** va shu kabi xabarlar oynalaridan foydalanish qoidalari keltirildi. **MouseHower** hodisasi kabi hodisalar o‘rganildi, amaliy dasturlarda qo‘llanildi va tahlil qilindi.

Nazariy savollar va amaliy topshiriqlar

1. Ob`ektni avtomatlashtirilishi lozim bo‘lgan jarayonlarni topish. **Kutubxona,** dorixona, supermarket, firma va yuridik tashkilotlar misolida jarayonning strukturasi va matematik modelini ishlab chiqish.
2. Kutubxona kitoblarining joriy reyting olib boruvchi avtomatlashtirilgan tizimini matematik loyihalash. Kutilayotgan natijalarni rejalashtirish.

4-BOB

Ma'lumotlar bazasini yaratish va ularga ishlov berish usullari

Bobning maqsadi

Ushbu 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:

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)
2. **SQL** so‘rovlar tili (**select** komandasi, **Like** va mantiqiy operatorlar (**and, or, not**), tartiblash (**order by)** va qisqartirish (**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);
3. **Visual C++ da MS Access** ning ma`lumotlar bazasini **SQL** so‘rovlari asosida tahrirlash usullari;
4. **Command** va **Datareader** sinf ob`ektlari yordamida **MS Access**da yaratilgan ma`lumotlar bazasining jadvalidagi barcha ma`lumotlarni o‘quvchi dastur;
5. **Console Application** muhitida **MS Access** ma`lumotlar bazasini yaratuvchi dastur;
6. **Console Application** muhitida **MS Access** ma`lumotlar bazasining jadvallariga ma`lumotlar yozuvchi dastur;
7. Command, **DataReader** sinf ob`ektlari va **DataGridView** komponentalari yordamida MBning jadvalidan ma`lumotlarni

o‘quvchi vizual dastur;

8. **Command**, **Adapter** va **DataSet** sinf ob`ektlari hamda **DataGridView** komponentasi yordamida MBning jadvalidan ma`lumotlarni o‘quvchi vizual dastur;
9. **MS Access** ning MBdagi jadval yozuvlarini yangilovchi vizual dastur;
10. **MS Access** ning MB dagi jadval yozuvlarini **SQL** so‘rovlari va “**Command**” sinf ob`ekti yordamida o‘chiruvchi vizual dastur.

Bob mundarijasi

4.1. Ma`lumotlar bazasini boshqarish tizimlari. Microsoft Accessda ma`lumotlar bazasini yaratish

Mavzuning maqsadi

MS Access 2010 da jadvallar, so‘rovlar, formalar va makroslar yaratish usullarni o‘rganish. **MS Access 2010** dasturida “**Talabalarning bilimlarini monitoring qiluvchi ma`lumotlar bazasini**” yaratish.

Informasion tizimlarni yaratish bo‘yicha jadal harakatlar va ma`lumotlar hajmining tez sur`atlar bilan oshib borishi 60 yillar boshida maxsus “Ma`lumotlar bazasini boshqarish tizimi” (MBBT) deb ataluvchi dasturiy kompleksning yaratilishiga olib keldi.

Ma`lumotlar bazasi - biror sohaga oid o‘zaro bog‘langan ma`lumotlar yig‘indisining disk tashuvchidagi tashkiliy jamlanmasidir. Boshqacha qilib aytganda, ma`lumotlar bazasi - bu kompyuter xotirasiga yozilgan ma`lum bir strukturali, o‘zaro bog‘langan va tartiblangan ma`lumotlar majmui bo‘lib, u biror bir ob`ektning xususiyatini, holatini

yoki ob`ektlar o‘rtasidagi munosabatni ma`lum ma`noda tavsiflaydi. Ma`lumotlar bazasini boshqarish tizimi (MBBT) - bu dasturiy va apparat vositalarining murakkab majmui bo‘lib, ular yordamida foydalanuvchi ma`lumotlar bazasini yaratish va shu bazadagi ma`lumotlar ustida ish yuritish mumkin. MBBT o‘z maxsus dasturlash tillariga ham ega bo‘lib, bu tillarga buyruqli dasturlash tillari deyiladi.

MBBTga **Oracle, Clipper, Paradox, FoxPro, SQL, Access** va boshqalarni misol keltirish mumkin. MBBT asosiy xususiyatlari - bu nafaqat ma`lumotlarni kiritish va saqlashda ishlatiladigan proseduralar tarkibi bo‘lmasdan, ularning strukturasini ham tasvirlaydi.

Ma`lumki, MB tushunchasi fanga kirib kelgunga qadar, ma`lumotlardan turli ko‘rinishda foydalanish juda qiyin edi. Dastur tuzuvchilar ma`lumotlarini shunday tashkil qilar edilarki, u faqat qaralayotgan masala uchungina o‘rinli bo‘lardi. Har bir yangi masalani hal qilishda ma`lumotlar qaytadan tashkil qilinar va bu hol yaratilgan dasturlardan foydalanishni qiyinlashtirar edi.

Shuni qayd qilish lozimki, **MB**ni yaratishda ikkita muhim shartni hisobga olmoq zarur:

Birinchidan, ma`lumotlar turi, ko‘rinishi, ularni qo‘llaydigan dasturlarga bog‘liq bo‘lmasligi lozim, ya`ni **MB**ga yangi ma`lumotlarni kiritganda yoki ma`lumotlar turini o‘zgartirganda, dasturlarni o‘zgartirish talab etilmasligi lozim.

Ikkinchidan, MBdagi kerakli ma`lumotni bilish yoki izlash uchun biror dastur tuzishga hojat qolmasin.

Shuning uchun ham **MB**ni tashkil etishda ma`lum qonun va qoidalarga amal qilish lozim. Bundan buyon **axborot** so‘zini **ma`lumot** so‘zidan farqlaymiz, ya`ni **axborot** so‘zini umumiy tushuncha sifatida qabul qilib, **ma`lumot** deganda aniq bir belgilangan narsa yoki hodisa sifatlarini nazarda tutamiz.

Kompyuter xotirasida har bir fayl **yozuv** deb ataladigan bir xil turdagi qismlardan iborat bo‘ladi**. Yozuv** - o‘zaro bog‘langan ma`lumotlarning bir qismidir. Fayldagi **yozuvlar** soni, qaralayotgan ma`lumotning o‘lchoviga bog‘liq. Har bir **yozuv** esa **maydon** deb ataladigan bo‘laklardan tashkil topadi. **Maydon** ma`lumotlarning, imkoni boricha, qisqa to‘plamidan iborat bo‘lishi lozim. Har bir **maydon,** o‘zi ifodalaydigan ma`lumotlariga ko‘ra, biror nomga ega bo‘ladi. Fikrimizni misol bilan ifodalashga harakat qilamiz.

Masalan, biror oliy o‘quv yurtining aniq fakultetida tahsil olayotgan biror guruh talabalari to‘g‘risidagi ma`lumotlar kiritilgan quyidagi jadvalni ko‘raylik:

4.1- jadval. MB dagi jadval ko‘rinishi.

Familiyasi	**Ismi**	**Tug‘ilgan sanasi**	**Guruhi**	**Turar joyi**	**qiziqqan fani**
Ochilov	Alisher	2.05.1978	5-M	S-1,15	Matem.
Qobulov	Farxod	2.12.1982	6-E	I.Sino,1	Adabiyot
Aminov	San`at	3.06.1980	5-M	S-2,12	Tarix
Tolipov	Jasur	24.05.1979	6-E	Beruni,2	Iqtisod

Bu misolda **4 ta yozuv** bo‘lib, ularning har biri **6 ta maydon**dan iborat. Mazkur maydonlarning har biri mos ravishda «**Familiyasi», Ismi», «Tug‘ilgan sanasi», «Guruhi», «Turar joyi» va «qiziqqan fani**» deb nomlangan. Demak, **yozuv**dagi **maydon**lar soni **yozuv**ga kiritiladigan ma`lumotlar hajmiga bog‘liq. Fayldagi bu **yozuvlar birlamchi h**isoblanadi. Chunki biror **yozuv**dagi ixtiyoriy ma`lumotni boshqa **yozuv**dagi ma`lumotlar bilan taqqoslab aniqlash mumkin emas. Shuning uchun ham bizga kerakli bo‘ladigan ikkilamchi yozuvlarni esa faqat amaliy dasturlar yordamida olish mumkin bo‘ladi. Modomiki shunday ekan, **MB** tashkil qilish, ularga qo‘shimcha ma`lumotlarni kiritish va mavjud **MB**dan foydalanish uchun maxsus **MB**lar bilan ishlaydigan **dasturlar** zarur bo‘ladi. Bunday dasturlar majmui **ma`lumotlar bazasini boshqarish tizimi (MBBT**) deb yuritiladi. Aniqroq qilib aytganda, **MBBT** – bu ko‘plab foydalanuvchilar tomonidan **MB**ni yaratish, unga qo‘shimcha ma`lumotlarni kiritish va **MB**ni birgalikda ishlatish uchun zarur bo‘lgan dasturlar majmuidir. **MBBTnin**g tarkibida asosiy komponenti – bu **ma`lumotlar** bo‘lsa, boshqa komponenti–**foydalanuvchilar, Hardware -** texnik va **Software** - dasturiy ta`minoti hisoblanadi. **Hardware** tashqi qo‘shimcha xotiradan (disk, magnit lentasi) iborat bo‘lsa, dastur qismi esa **MB** bilan foydalanuvchi o‘rtasidagi muloqotni tashkil qilishni amalga oshiradi. **MB**ning tuzilishi o‘rganilayotgan ob`ektning ma`lumotlari ko‘rinishi, ma`nosi, tuzilishi va hajmiga bog‘liq bo‘ladi.

4.4.1. «MS Access» dasturi yordamida talabalarning bilimlarini monitoring qilish ma`lumotlar bazasini yaratish texnologiyasi

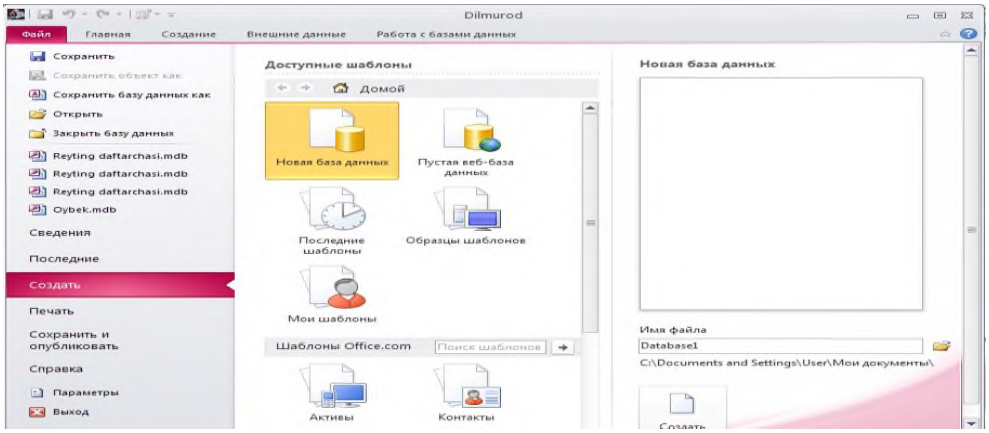
MBning umumiy ko‘rinishi quyidagicha:

4.1-rasm. MBning asosiy oynasi

Ushbu MBni yaratish algoritmi quyidagicha:

1- Har bir semestr uchun alohida jadvallar yaratiladi.
2- Hamma jadvallarni birlashtiruvchi umumiy baza yaratiladi.
3- Jadvallar birga ko‘p usulida bog‘lanadi.
4- Umumiy jadvaldan ma`lumotlarni qidirish uchun so‘rovlar (Familiyasi, Guruhi, Kursi va Guruh rahbari bo‘yicha) yaratiladi.
5- Hisobotlar tayyorlash uchun bitta makros yaratiladi va u (**Hisobot**) tugmasiga biriktiriladi.
6- Har bir semestr uchun alohida formalar yaratiladi.
7- Har bir semestr formasiga, jadvaldagi ma`lumotlarni ko‘rish uchun 3 ta (Familiyasi bo‘yicha, Guruhi bo‘yicha va Umumiysi bo‘yicha), guruh talabalarini fanlardan olgan baholarini ko‘rish uchun bitta, (Maxsus baholash) jadvalning **Excell** faylini yaratish uchun esa bitta (**Excell** faylini yaratish) tugmalar joylashtiriladi.
8- Asosiy oynaga **4.1-rasmda** keltirilgandek tugmalar joylashtiriladi.
9- Semestrlar bo‘yicha yaratilgan formalar ushbu tugmalarga biriktiriladi.

MBni yaratish uchun **“Microsoft Access 2010”** dasturida foydalaniladi:

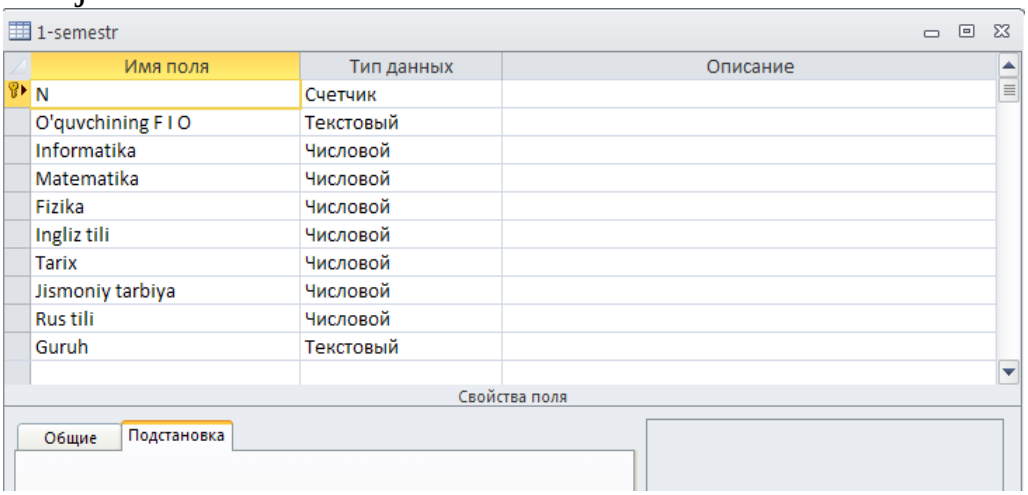


4.2-rasm. “MS Access 2010” dastur muxarriri

Bu oynadan: **“Доступные шаблоны”** bandidan **“Новая база данных”** buyrug‘i tanlanadi. **“Имя файла”** bandini tanlab yaratilishi kerak bo‘lgan ma`lumotlar bazasini nomi kiritilib, “**Создать”** tugmasi bosiladi. Agar uning formati **“.mbd”** bo‘lishini xohlasangiz **“Имя файла”** bandidagi “**папка**” piktogrammasi bosiladi va ochilgan oynadan fayl nomi va kengaytmasini ixtiyoriy tanlash imkoni hosil bo‘ladi.

I. “Konstruktor rejimida” jadval yaratish

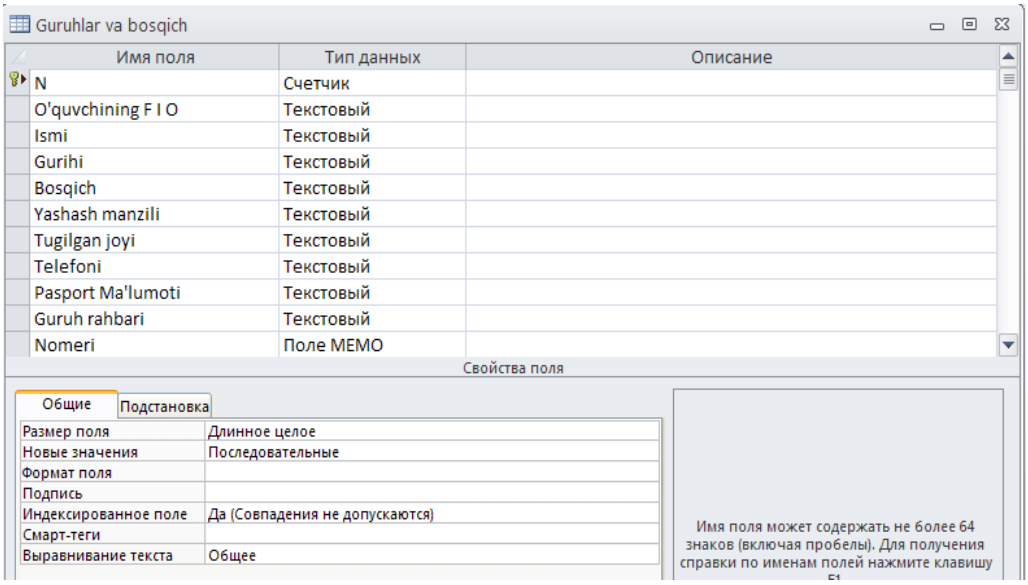
1-semestr jadvali:



4.3-rasm. Konstruktor rejimida 1-semestr jadvalini yaratish

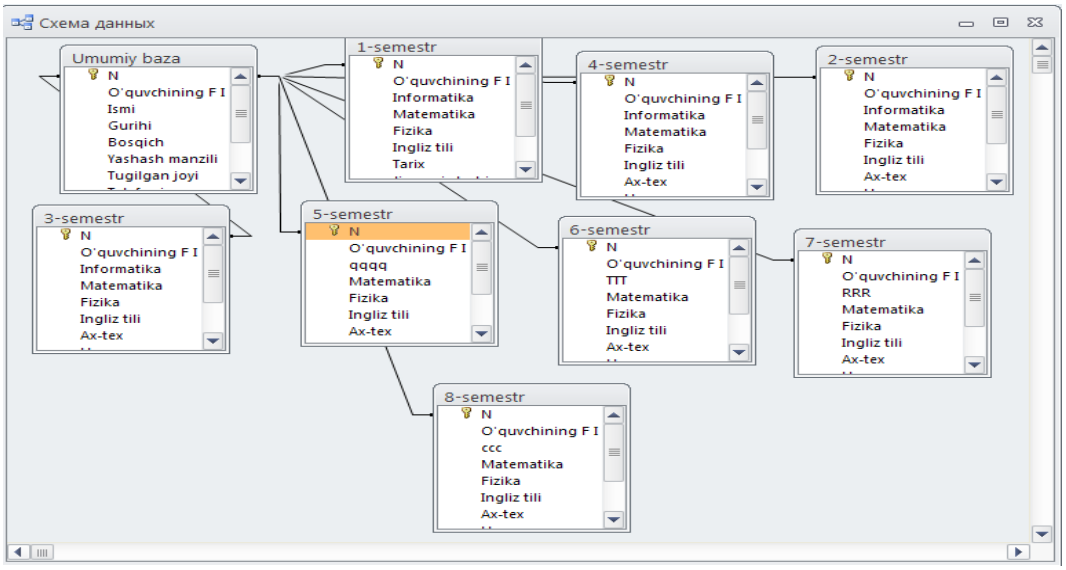
Shu kabi boshqa jadvallarni (**2-semestr, . . . 8-semestr**) yaratib olinadi, chunki har bir semestrda fanlar o‘zgarib boradi.

Umumiy baza quyidagicha tuziladi:



4.4-rasm. Konstruktor rejimida umumiy baza jadvalini yaratish

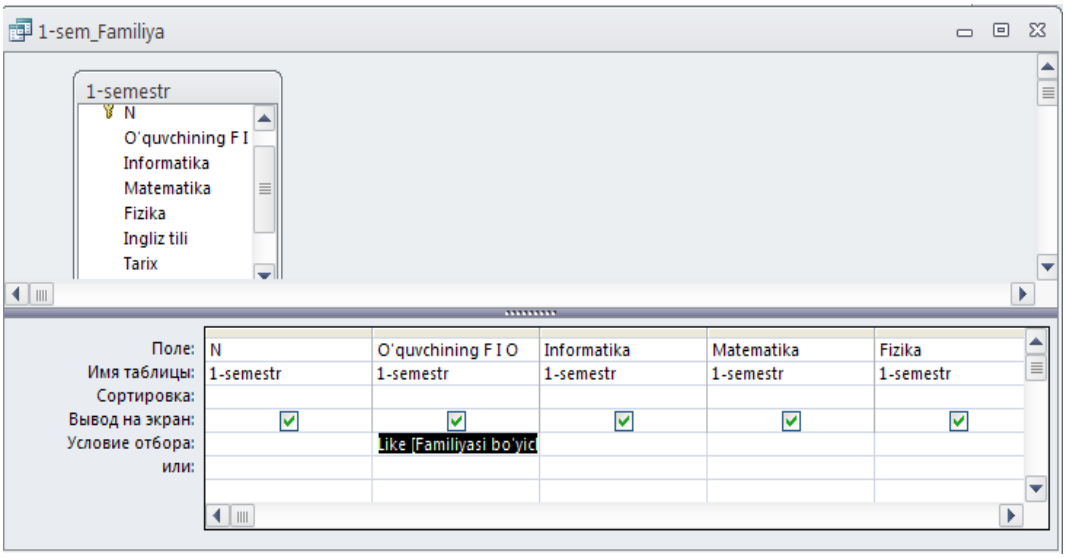
Bu jadval 1 - bosqichdan to 4 - bosqichgacha bo‘lgan talabalarning ma`lumotlarini o‘zida jamlaydi va boshqa yordamchi jadvalga avtomatik tarzda tarqatadi. Buni amalga oshirish uchun jadvallarni bog‘lash lozim bo‘ladi. U quyidagicha amalga oshiriladi:



4.5-rasm. Umumiy baza jadvaliga semestr jadvallarini bog‘lash

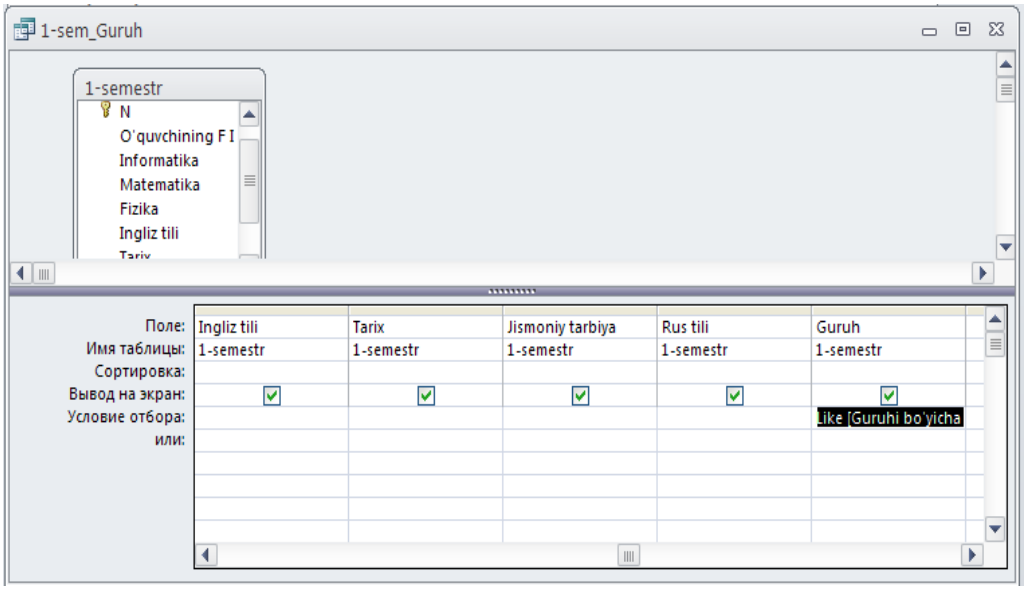
II. Konstruktor rejimida so‘rovlar yaratish

Har bir semestrda alohida jadval bo‘lgani uchun ularga alohida so‘rovlar yaratiladi. Ya`ni, familiyasi bo‘yicha qidirish so‘rovini yaratish uchun 4.6-rasmdagi amallar ketma-ketligi bajariladi:



4.6-rasm. Konstrurtor rejimida so‘rovlar yaratish

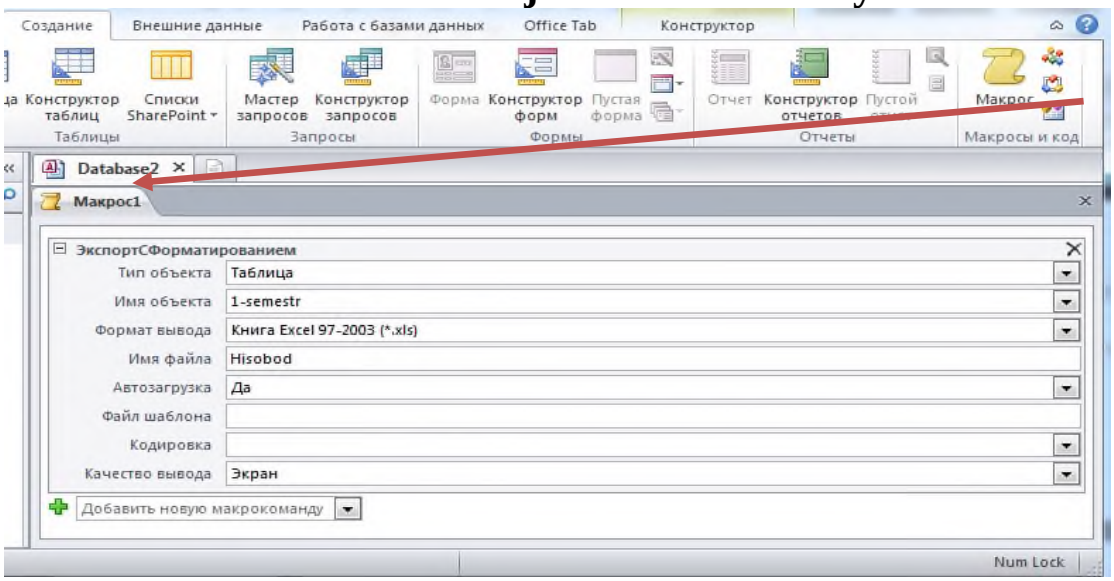
Guruhi bo‘yicha qidirish amalini bajarish uchun 4.7-rasmdagi amallar ketma-ketligi bajariladi:



4.7-rasm. Konstrurtor rejimida so‘rovlar yaratish

Shu kabi qolgan semestr so‘rovlari va umumiy jadvalning so‘rovlari yaratiladi.

III. Konstruktor rejimida makroslar yaratish



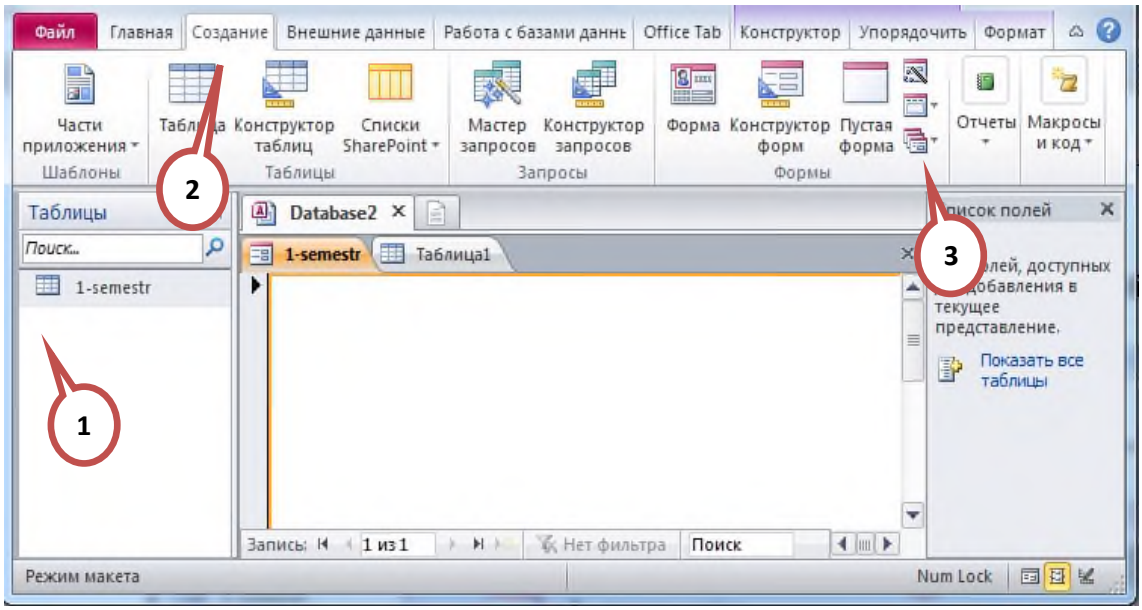
4.8-rasm. 1- semestr uchun makros yaratish

Ushbu makros **1-semestr** jadvalini **MS Excell** dasturiga eksport qiladi. Xuddi shu tartibda qolgan semestr jadvallari uchun ham makroslar yaratiladi.

IV. Konstruktor rejimida formalar yaratish

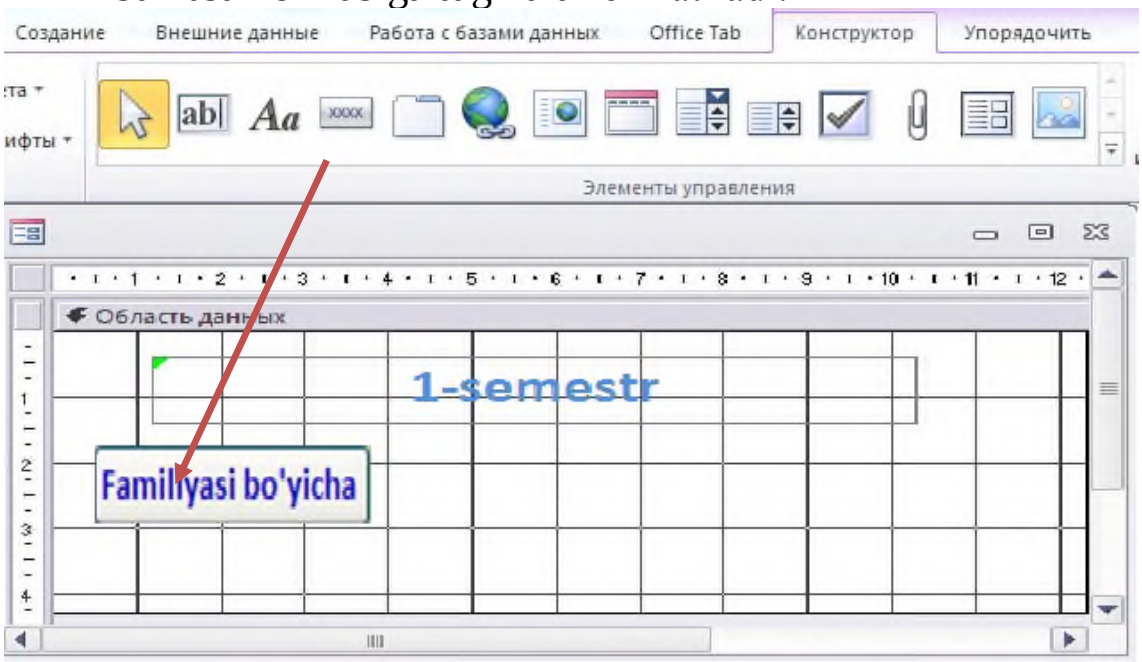
Har bir semestrda alohida jadval va so‘rovlar bo‘lgani uchun ularga alohida formalar yaratiladi. U quyidagicha:

1-semestr jadvali tanlanadi va “**Создание”** menyusidan “**Пустая формы”** buyrug‘i beriladi. Agar tanlangan jadvalni malumotlarini formaga chiqarmoqchi bo‘lsangiz, u holda “**Формы”** buyrug‘i beriladi.

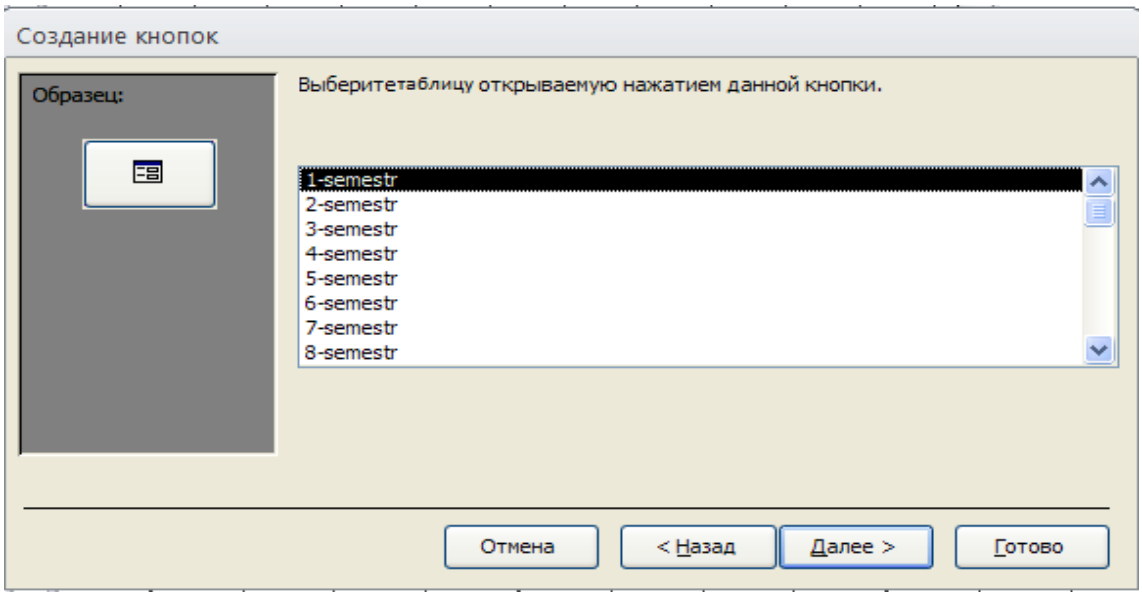


4.9-rasm. 1- semestr uchun forma yaratish

1-semestr formasiga tugmalar o‘rnatiladi.

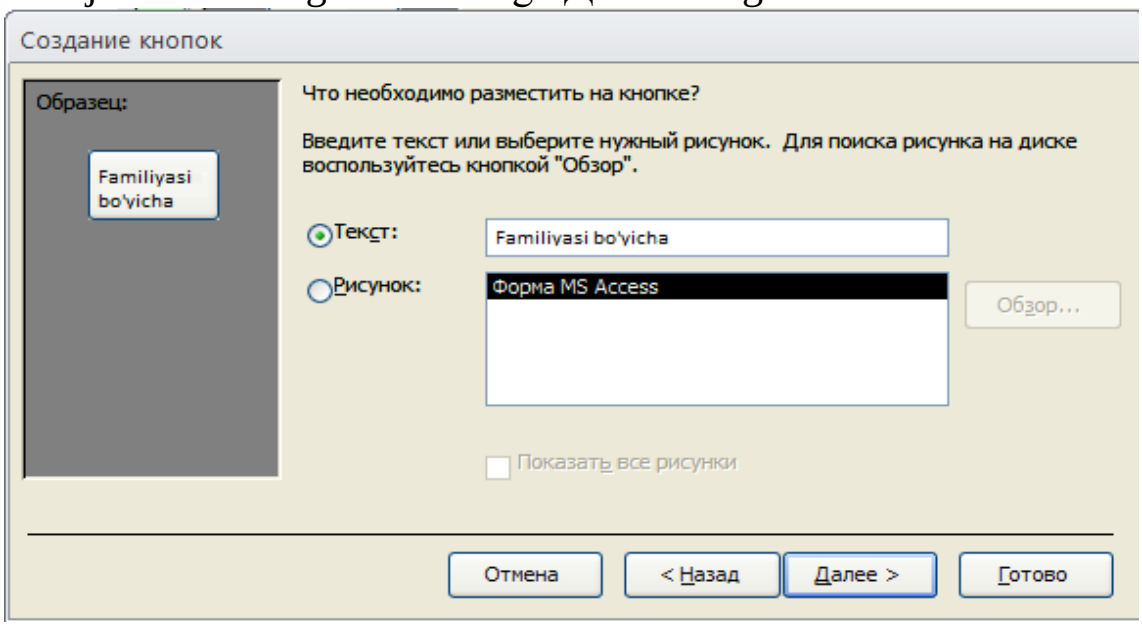


4.10-rasm. 1- semestr formasiga tugmalar o‘rnatish



4.11-rasm. Formaga formalarni biriktirish

Kerakli jadval tanlangandan so‘ng “**Далее”** tugmasi bosiladi:



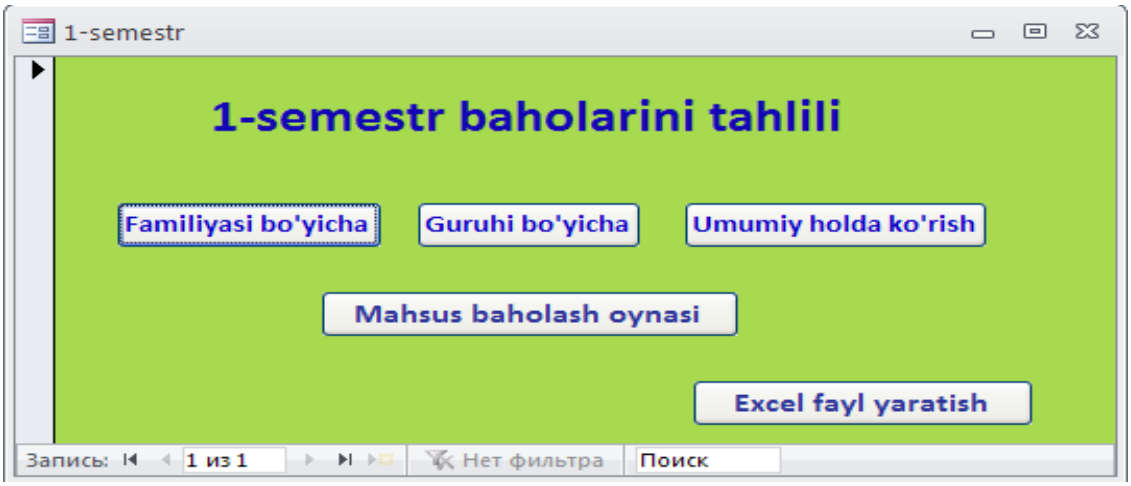
4.12-rasm. Tugma xossalarini belgilash

Qolgan to‘rtta tugmalar uchun ham shu amallar qaytariladi, faqat forma kerak bo‘lsa formalar bilan ishlash, makros bo‘lsa makroslar bilan ishlash va so‘rov bo‘lsa so‘rovlar bilan ishlash bo‘limi tanlanadi. Natijada formada hosil bo‘lgan tugma bosilganda nima tanlangan bo‘lsa ushbu buyruq bajariladi.

Eslatma: Har bir jarayonlar konstruktor rejimida bajariladi. Masalan, tayyor jadvallarni o‘zgartirish uchun jadval tanlanadi va

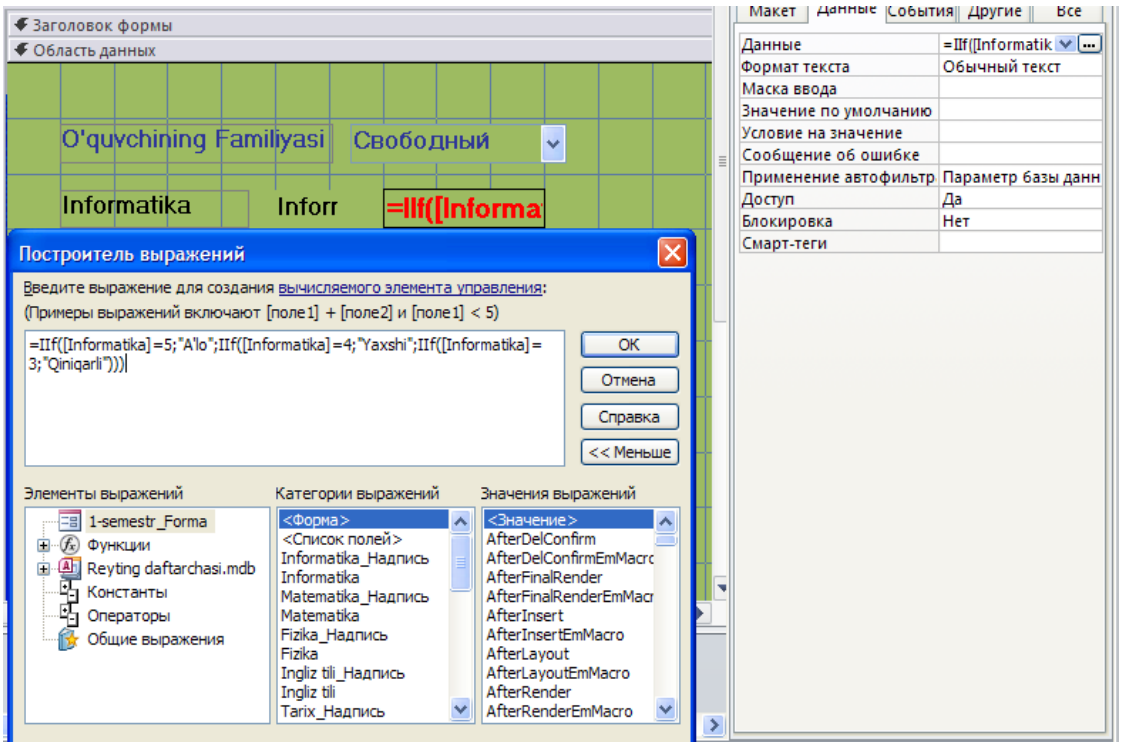
sichqonchaning o‘ng tugmasi bosilib, kontest menyudan “**Konstruktor”** buyrug‘i tanlanadi.

Formaga tashqi dizayn berilib, klaviaturadagi **F5** tugmasi bosilgandan so‘ng forma ishchi rejimga o‘tadi va quyidagi ko‘rinishga keladi:



4.13-rasm. 1-semestr forma oynasi

“Maxsus baholash oynasi” tugmasini yaratishda talabalarni baholar reytingi formasini har bir semestr uchun yaratiladi:



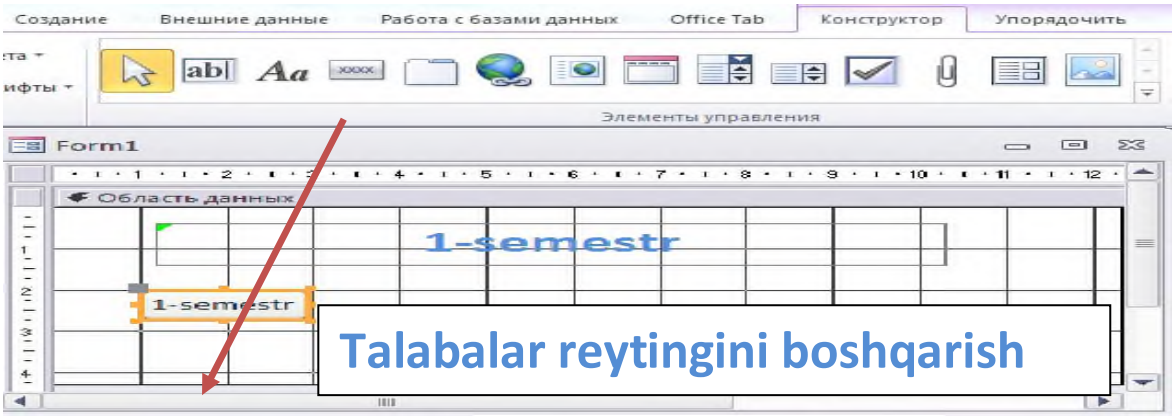
4.14-rasm. 1-semestr formasida formulalar bilan ishlash

4.15-rasm. 1- semestr uchun baholar reytingi formasi

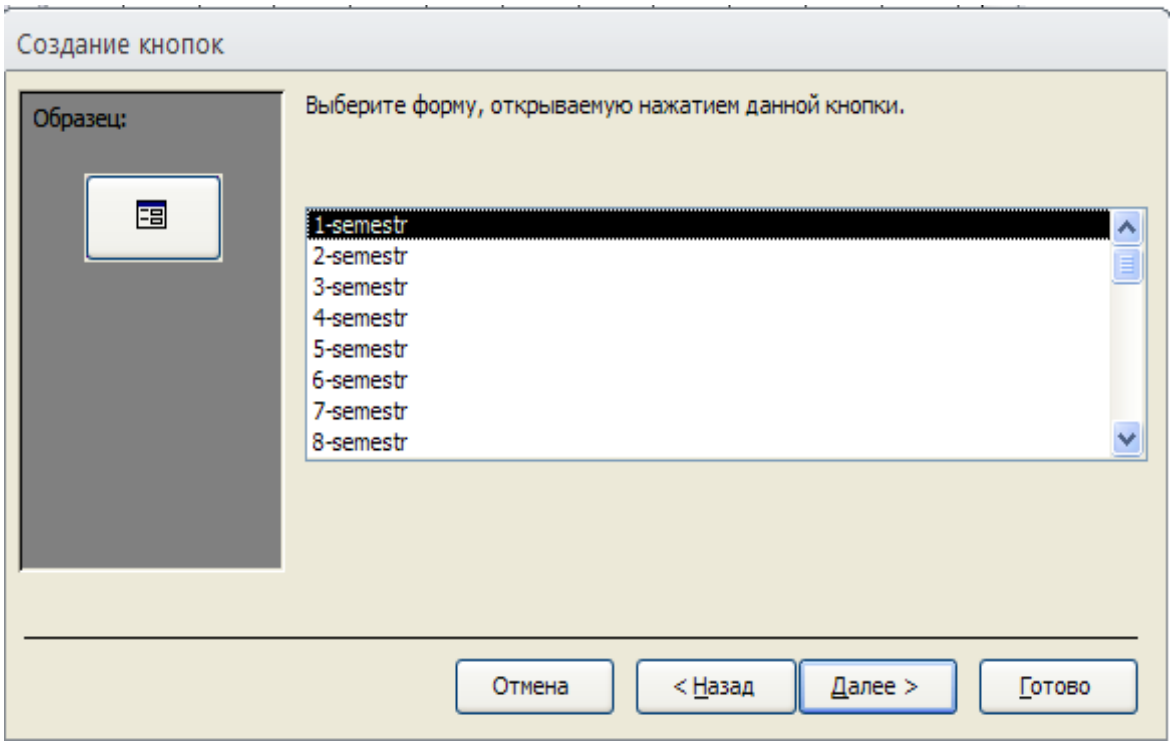
Ushbu amallar boshqa semestr jadvallari uchun ham bir xildir.

V. Formaga komponentalar joylashtirish

Asosiy oyna formasida tugmalarga semestr formalarini biriktirish. Forma yaratilganda “**konstruktor”** menyusi paydo bo‘ladi. Konstruktordan knopka tanlanadi.

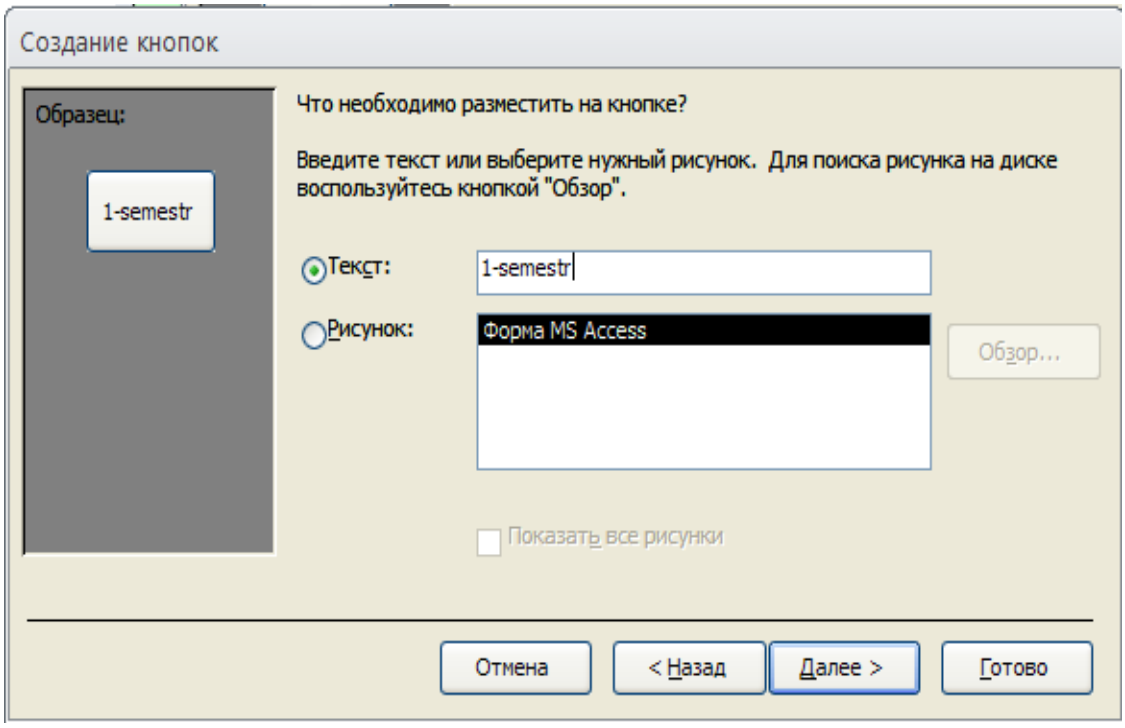


4.16-rasm. Asosiy formaga komponentalar joylashtirish.



4.17-rasm. Formaga formalarni biriktirish

Kerakli forma tanlangandan so‘ng “**Далее”** tugmasi bosiladi:



4.18-rasm. Tugma xossalarini belgilash oynasi

Yaratilgan MBda umumiy jadval yaratilgan. U orqali barcha jadvallar tashkil topadi. Bu jadvalda talabalarning umumiy ma`lumotlari

joylashgan. 1- bosqichdan to 4-bosqichgacha bo‘lgan talabalarning familiyasi, guruhi, bosqichi va guruh rahbari bo‘yicha qidirish imkoniyatlari to‘rtta **(familiyasi, guruhi, bosqichi va guruh rahbari bo‘yicha qidirish)** tugmalarga joylashtirilgan. “**Hisobot**” va “**Excell fayl yaratish**” tugmalarining vazifasi umumy jadvalda joylashgan ma`lumotlar asosida hisobotlar va **Excell** fayl yaratishdan iboratdir. Dasturdan chiqib ketish uchun “**Chiqish**” tugmasi joylashtirilgan. MBda 12 ta jadval, 30 ta so‘rov, 12 ta makros va 11 ta formadan tashkil topgan. Uning interfeysi quyidagicha:

4.19-rasm. MBning asosiy oynasi

Mustahkamlash uchun mashqlar

1. Mahalla ma`lumotlar bazasi;
2. Maktab o‘quvchilarini ma`lumotlar bazasi;
3. Shifoxonaning qabul bo‘limini ma`lumotlar bazasi;
4. Fakultetning ma`lumotlar bazasi.

Nazorat savollari:

1. MS Access 2010 dasturining maqsad va vazifalari?
2. MS Access 2010da jadval yaratishning nechta usuli bor?
3. MS Access 2010 da so‘rov yaratishning nechta usuli bor?
4. MS Access 2010 da forma yaratishning nechta usuli bor?
5. MS Access 2010 da makros yaratishning nechta usuli bor?
6. Formaga tugmachalar qaysi rejimda joylashtiriladi?

4.2. SQL(Structured Query Language) tili so‘rovlari. SQL haqida

SQL – bu so‘rov tili ko‘p operatorlardan tashkil topgan bo‘lib, bu operatorlar orqali foydalanuvchilar va dasturlar Oracle (MBBT) dagi ma`lumotlar bazasiga murojaatni amalga oshirishi mumkin. Oracle utilitalari yoki har xil dasturlar SQL operatorlarisiz bazaga murojaatni amalga oshirishi mumkin, lekin so‘rovlarni amalga oshirishda bu so‘rov tilidan foydalanmaslikning iloji yo‘q.

1970 yil iyun oyida E.F.Kodd o‘zining “A Relational Model of Data for Large Shared Data Banks” maqolasini ommaga taqdim etdi. Bu maqola “Communications of the ACM ” jurnalida chop etildi. Hozirgi kunda Koddning bu modeli “Relasion ma`lumotlar bazasini boshqarish tizimi (RMBBT)” ning yakuniy modeli deb qabul qilindi. Koddning modelini yo‘lga qo‘yish maqsadida IBM firmasi SEQUEL(Structured English Query Language) tilini ishlab chiqdi. Keyinchalik bu til SQL tiliga o‘zgartirildi, lekin haligacha “sikvel” deb ham yuritilmoqda. 1979 yil Relational Software (hozirgi vaqtdagi **Oracle**) korporasiyasi **SQL**ning birinchi tijoriy ishlanmasini ommaga taqdim etdi. Hozirgi kunda SQL tili RMBBTning standart tili hisoblanadi.

SQL tili so‘rov-natija ko‘rinishida ishlaydi. So‘rovlar har bir element uchun emas, butun bir guruh uchun beriladi va natija olinadi. **SQL** uchun ma`lumotlar bazasidagi ma`lumotlar qay shaklda, qay tartibda joylashganini umuman ahamiyati yo‘q, foydalanuvchilar ham bu ma`lumotlarni bilishi shart emas. Faqatgina operatorlarni to‘g‘ri yozish orqali istalgan ma`lumotlarni chiqarish mumkin bo‘ladi.

SQL tili barcha ma`lumotlar bazasini boshqarish tizimlari uchun umumiy standart til hisoblanadi. Bundan kelib chiqadiki, agar siz bu tilni bir marotaba o‘rganib olsangiz, istalgan MBBTlari bilan ishlay olasiz. Bitta **MBBT**da yaratilgan biror SQL operatorlar yig‘indisi (kichik so‘rov dasturi)ni istalgan **MBBT**ga ko‘chirish mumkin bo‘ladi.

SQL operatorlari orqali quyidagi vazifalarni bajarish mumkin:

- Ma`lumotlarni so‘rov orqali olish.
- Jadvalning qatorlariga ma`lumot qo‘shish, qatorlarini o‘chirish va yangilash.
- Ob`ektlarni yaratish, o‘zgartirish va o‘chirish.
- Ma`lumotlar bazasi va ob`ektlarga ruxsatlarni o‘rnatish.
- Ma`lumotlar bazasi foydalanuvchilarini hosil qilish va baza xavfsizligini ta`minlash.

Ikki xil turdagi **SQL** mavjud: interaktiv va o‘rnatilgan (встроенный). SQL ning bu ikki turini ishlashi bir xil, lekin har xil joyda ishlatiladi.

Interaktiv **SQL** deganda — ma`lumotlar bazasiga so‘rov orqali murojaat qilib, shu zahoti natijani olish tushuniladi. Ya`ni, bunda ketma-ketlik asosida jarayon sodir bo‘ladi. So‘rov-natija rejimda ishlaydi.

O‘rnatilgan SQL deganda – so‘rovlar yig‘indisi biror dasturlash tilida ishlatilishi tushuniladi. **Pascal, Delphi, Java** va **C++** tillarida bazaga murojaat qilib, natijani biror o‘zgaruvchiga yuklab qo‘yamiz va kerakli joyda bu natijani ishlatamiz. Ya`ni bunda so‘rov berib, darhol natijani ololmaymiz. Natija faqat dasturning davom etishi uchun olinadi va talab etilgan joyda ishlatiladi.

select komandasi

Misol sifatida quyidagi jadvaldan foydalanamiz.

NAME	SURNAME	YEAR	NUM
David	Beckham	7	1
Eric	Contana	17	2
Wayne	Rooney	10	3
Rio	Ferdinand	4	4
Nemane	Vidic	6	5

Bu jadval “**misol**” deb nomlanib, 4 ta ustun va 6 ta qatordan iborat. Yuqoridagi qatorda faqat qator nomlari aks ettirilgan, ular ma`lumot vazifasini bajarmaydi. 5 ta qatordan iborat ma`lumot, bizning jadvalimizda joylashgan va biz bu ma`lumotlarni SQL so‘rov tili orqali har xil ko‘rinishda chiqarib olishimiz mumkin. Albatta haqiqiy ma`lumotlar bazasida bunday kam ma`lumotlar saqlanmaydi, biz misollarni shu kichik ma`lumotlarda bajaramiz.

Demak birinchi komanda bu – **SELECT**.

select * from misol;

Bu so‘rovning ma`nosi, “**misol**” jadvalidagi barcha ma`lumotlarni chiqar deganidir. “*” barcha ustunlardagi ma`lumotlarni degani. “**select**” – esa ekranga chiqar degani. Natijani ko‘ramiz:

NAME	SURNAME	YEAR	NUM
David	Beckham	7	1
Eric	Contana	17	2
Wayne	Rooney	10	3
Rio	Ferdinand	4	4
Nemane	Vidic	6	5

Agar bizga barcha ustun ma`lumotlar emas, faqatgina ba`zi birlari kerak bo‘lsa, kerakli ustun nomlarini vergul orqali nomma-nom yozamiz.

select name, surname, year from misol;

NAME	SURNAME	YEAR
David	Beckham	7
Eric	Contana	17
Wayne	Rooney	10
Rio	Ferdinand	4
Nemane	Vidic	6

Ma`lumotlar bazasidan, ma`lumotlarni chiqarishda mantiqiy amallar orqali(+, -, *, /, ()) ma`lumotlarga o‘zgartirish kiritib, ekranga chiqarishimiz mumkin bo‘ladi.

select name, year+15, num*10 from misol;

Natijani ko‘rsangiz, so‘rov tili orqali, ma`lumotlarni o‘zgartirganimizni ko‘rishingiz mumkin bo‘ladi.

NAME	YEAR+15	NUM*10
David	22	10
Eric	32	20
Wayne	25	30
Rio	19	40
Nemane	21	50

yoki bo‘lmasa quyidagicha

select year+10*num, name from misol;

YEAR+10*NUM	NAME
17	David
37	Eric
40	Wayne
44	Rio
56	Nemane

Agar bazadagi ustun nomlari sizga yoqmasa, siz ularni o‘zingiz xohlaganday nom bilan ekranga chiqarishingiz mumkin bo‘ladi, bunda ma`lumotlar bazasidagi ustun nomlari o‘zgarmaydi, faqat ekranda o‘zgargan ko‘rinishi chiqadi. Bunda bizga “**as**” kalit so‘zi yordam beradi.

select name as ism, surname as familiya from misol;

ISM	FAMILIYA
David	Beckham
Eric	Contana
Wayne	Rooney
Rio	Ferdinand
Nemane	Vidic

Agar nomlarni qo‘shtirnoq ichiga olsangiz, qo‘shtirnoqdagi katta kichik harflar farqli bo‘lib qoladi.

select name as “IsM”, surname as “FamiliyA” from misol;

IsM	FamiliyA
David	Beckham
Eric	Contana
Wayne	Rooney
Rio	Ferdinand
Nemane	Vidic

Like va mantiqiy operatorlar (and, or, not)

Navbatdagi darsimizning asosiy qismi “Like” operatoriga bag‘ishlanadi. Bu orqali so‘rovlarni ma`lum bir shablon orqali qurishingiz mumkin bo‘ladi. Hozirgacha biz o‘rgangan so‘rovlar aniqlik asosida tuzildi va bitta harfga adashib so‘rov tuzib qo‘ysangiz, kerakli ma`lumotlarni olishga qiynalishingiz aniq. Bu darsda quyidagi “**test**” nomli jadvaldan foydalanamiz.

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Sir	Alex	50	
Phil	John_	16	12000

Agar siz biror “x” ustundagi ma`lumot bo‘yicha qolgan ustundagi ma`lumotlarni chiqarmoqchi bo‘lsangiz va “x” ustundagi ma`lumotni nomini to‘liq bilmasangiz, “like” operatoridan foydalanish sizga yengillik yaratadi. Demak nazariyadan amaliyotga o‘tamiz, misol “**test**” jadvalidagi “**ISM**” ustunidagi ma`lumotlar ichidan “**S**” harfidan boshlanadigan ismlarning to‘liq ismini va familiyasini chiqaruvchi so‘rov tuzish kerak:

```
select ism, yosh, familiya from test where ism like 'S%';
```

ISM	YOSH	FAMILIYA
Sir	50	Alex

Bu yerda “**like**” kalit so‘zidan keyin, biror shablon beriladi, **“%”** belgisi nol yoki bir necha simvol degan ma`noni beradi, bundan kelib chiqadiki, so‘rovda “S” harfidan boshlanuvchi (bu harfdan keyin istalgancha simvol bo‘lishi mumkin yoki umuman bo‘lmasligi ham mumkin) ismlarni chiqarib beradi. Y qorida keltirilgan belgilarni(“**%”,”_”**) aralash holda ham ishlatish mumkin. Misol,

“familiya” ustunidagi ma`lumotlar ichidan ikkinchi simvol “**e**” dan iborat bo‘lgan va bu simvoldan keyin istalgancha simvol bo‘lgan familiyani to‘liq holda chiqaruvchi so‘rov quyidagicha bo‘ladi:

select familiya from test where familiya like ‘_e%’;

FAMILIYA
Beckham

Bir necha shartlarni birlashtiruvchi mantiqiy operatorlarni boshlaymiz. Bu operatorlar “**where**” operatoridan keyin keladigan shartlarni mantiqiy tomondan birlashtirib beradi. Mantiqiy amallarda 2 ta tushuncha mavjud, bular: rost (**true**) va yolg‘on (**false**). Barcha shartning natijasi yoki rost yoki yolg‘on bo‘lishi mumkin. Shu natijaga qarab mantiqiy amallar ishlatiladi. Bu mantiqiy amallar 3 hildir.

1. **And** – mantiqiy “va” operatori. Bir necha shartlarning barchasi “**true**” (rost) bo‘lsa, oxirgi natija “**true**”bo‘ladi. Misol, jadvaldagi “**yosh**” ustuni 20 dan katta bo‘lgan va “maosh” ustuni 30000 dan katta bo‘lgan insonlarning ismi va familiyasini chiqaruvchi so‘rov:

select ism, familiya from test where (yosh>20) and (maosh>30000);

ISM	FAMILIYA
David	Beckham

Ikkala shart bajariladigan (albatta yoshi 20 dan katta bo‘lishi va maoshi 30000 dan katta bo‘lishi shart) ustunlar ekranga chiqariladi.

2. **Or** – mantiqiy “yoki” operatori. Bir necha shart orasidan istalgan bittasi “**true**” (rost) bo‘lsa, oxirgi natija rost bo‘ladi va shart bajarilgan hisoblanadi. Misol, jadvaldan ismi “**David**” bo‘lgan yoki maoshi 20000 dan kam bo‘lganlar haqida ma`lumot olish so‘rovi:

select * from test where (ism='David') or (maosh<20000);

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Phil	John_	16	12000

3. **Not** – inkor ma`nosini bildiruvchi mantiqiy amal. Barcha buyruqlarni inkorini ishlatadi. Misol, yoshi 16, 21, 30 ga teng bo‘lmagan insonlarning ismini chiqarish:

select ism as name from test where yosh not in (16, 21, 30);

NAME
David
Carlos
Sir

Agar “**not**” kalit so‘zini olib tashlasak, yoshi 16, 21, 30 ga teng bo‘lgan bo‘lib qolar edi.

Tartiblash (order by va qisqartirish(distinct))

Ma`lumotlar har xil va hamma insonlarga kerak, shu bilan birga bu ma`lumotlar har kimga har xil ta`sir qiladi. Shuning uchun korxonalarda hisobotlar har xilda tayyorlanadi. Ma`lumotlar bir xil bo‘lsa ham, so‘ragan insonlarga qarab har xilda taqdim etiladi. Bu, bir joyga to‘plangan ma`lumotlarni har xil ko‘rinishda saqlashga olib keladi, bu esa albatta ortiqcha ishni yuzaga keltiradi. Buni oldini olish uchun **SQL** so‘rov tilida tartiblash operatorlari ishlab chiqilgan. Bizga ushbu maqolada quyidagi “**test** ”jadvali kerak bo‘ladi:

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Sir	Alex	50	
Phil	John_	16	12000

SQL tilida tartiblash operatori vazifasini “**order by**” bajaradi. Misol ko‘ramiz, **FAMILIYA** ustuni bo‘yicha tartiblab, jadvaldagi ma`lumotlarni to‘liq ekranga chiqaruvchi so‘rov:

select * from test order by familiya;

ISM	FAMILIYA	YOSH	MAOSH
Pablo	Aimar	30	25000
Sir	Alex	50	
David	Beckham	21	55000
Phil	John_	16	12000
David	Luiz	23	30000
Carlos	Puyol	23	27000

Natijadan ko‘rinib turibdiki, “**FAMILIYA**” ustuni alfavit bo‘yicha tartiblanib, ekranga chiqmoqda.

“**Order by**” operatori ikki xil kalit so‘z bilan ishlatilishi mumkin. Bular: **desc, asc.** “**Desc**” kalit so‘zi, ustunni kamayish ketma-ketligida tartiblab ekranga chiqaradi, ya`ni eng katta qiymatdan boshlab eng kichikka qadar. Misol, “ISM” ustunini kamayish tartibida tartiblab chiqarish so‘rovi tuzish:

select * from test order by ism desc;

ISM	FAMILIYA	YOSH	MAOSH
Sir	Alex	50	
Phil	John_	16	12000
Pablo	Aimar	30	25000
David	Luiz	23	30000
David	Beckham	21	55000
Carlos	Puyol	23	27000

Natijadan ko‘rinadiki, alfavitning quyi qismidan yuqori qismiga qarab ko‘tarilib tartiblanmoqda. “**Ass**” kalit so‘zi esa kichikdan-katta (alfavit bo‘yicha yuqori harfdan quyi harfga qarab) tomonga qarab tartiblab chiqariladi. Agar hech qaysi kalit so‘z berilmasa, avtomat “ass” kalit so‘zini ishlatib, tartiblab chiqaradi.

Agar bir necha ustunni tartiblab chiqarish kerak bo‘lsa, “**order by**” operatoridan keyin ustunlar nomi vergul orqali ketma-ket yozib chiqariladi. Bunda natija, eng oldin yozilgan ustun bo‘ylab tartiblanib, keyin ikkinchi bo‘lib yozilgan ustun bo‘yicha tartiblab ekranga

chiqaradi. Misol, avval “**ISM**” ustuni, keyin “**Familiya**” ustuni bo‘yicha tartiblab, barcha ma`lumotlar ekranga chiqarilsin.

select * from test order by ism, familiya;

ISM	FAMILIYA	YOSH	MAOSH
Carlos	Puyol	23	27000
David	Beckham	21	55000
David	Luiz	23	30000
Pablo	Aimar	30	25000
Phil	John_	16	12000
Sir	Alex	50	

Bu tartiblash, agar birinchi tartiblanayotgan ustun bir necha turli hil ma`lumotlardan tashkil topgan bo‘lsa foydali hisoblanadi.

Keyingi operatorimiz bir hil ma`lumotlarni qisqartirib, natijani chiqaradigan operator “**distinct**” haqidadadir. Agar chiqarayotgan ma`lumotlar orasida bir xil ma`lumot bo‘lsa, bu ma`lumotni bitta qilib chiqaradi. Misol, “**ISM**” ustunidagi bir xil bo‘lmagan ismlarni chiqarish so‘rovini ko‘ramiz:

select distinct ism from test;

ISM
Pablo
Sir
David
Carlos
Phil

Bundan xulosa chiqarish mumkinki, “**distinc**” operatorini ishlatsak, bir xil qatorlarni umuman ko‘rmas ekanmiz, lekin ustundagi ma`lumotlar bir xil bo‘lishi mumkin. Bu operator faqat qator uchun ishlaydi.

SQL so‘rov tilida jadvallarni birlashtirish(UNION)

SQL so‘rov tili haqidagi maqolalarni kiritishni davom ettiramiz. Bunda biz ikki jadvalni bir-biri bilan birlashtirishni ko‘rsatib o‘tamiz, buning uchun bizga quyidagi ikkita(**test1, test2**) jadvallar kerak bo‘ladi.

NAME	FAMIL	YEAR
Phil	Jones	1992
Tom	Cleverley	1989
Adnan	Januzaj	1995
Wayne	Rooney	1985
Ashley	Young	1985

NAME	FAMIL	NUMBER
Phil	Jones	4
Ashley	Young	18
Ander	Herrera	21

Jadvallar bir-birlari bilan har xil ko‘rinishda birlashtirilishi mumkin. Dastlab ikki jadvalni to‘liq bir biri bilan birlashtiramiz, buning uchun biz «**Union**» kalit so‘zidan foydalanamiz. Bu birlashtirishda ikkita jadvaldagi barcha ma`lumotlar chiqariladi. Barcha ustundagi bir xil bo‘lgan ma`lumotlar esa, bir marotaba chiqariladi, ya`ni bir xil ma`lumotlar ekranga chiqmaydi.

```
select name, famil from test1
union
select name, famil from test2;
```

NAME	FAMIL
Adnan	Januzaj
Ander	Herrera
Ashley	Young
Phil	Jones
Tom	Cleverley
Wayne	Rooney

Yuqoridagi so‘rovda birinchi jadval (**test1**)ga ikkinchi jadval (**test2**) birlashtirilmoqda. So‘rovdan ko‘rib turibsizki, ekranga chiqarilayotgan ikkala jadval ustun nomi va tiplari bir xil bo‘lishi lozim, agar ustunlar bir xil bo‘lmasa quyidagi xatolik paydo bo‘ladi.

select name, famil, year from test1
union
select name, famil from test2;

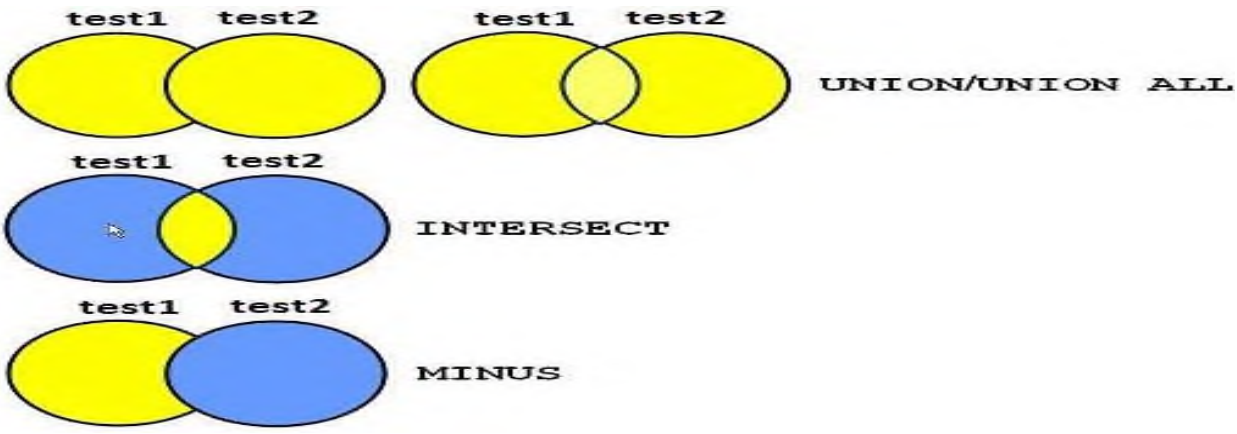
```
select name, famil, year from test1
*

ERROR at line 1:
ORA-01789: query block has incorrect number of result columns
```

Keyingi birlashtirish, ikkala jadvaldagi barcha ma`lumotlarni chiqarish uchun ishlatiladi, bunda bizga «**Union all**» kalit so‘zi yordam beradi. Bunda ikkala jadvaldagi hamma ma`lumotlar(bir xillari ham) chiqariladi.

select name, famil from test1
union all
select name, famil from test2;

NAME	FAMIL
Phil	Jones
Tom	Cleverley
Adnan	Januzaj
Wayne	Rooney
Ashley	Young
Phil	Jones
Ashley	Young
Ander	Herrera

Xulosa qiladigan bo‘lsak, barcha birlashtirishlarni quyidagi rasmda ko‘rish mumkin:



where komandasi.

SQL so‘rov tilining asosiy shart qo‘yish operatori (**where**) ni o‘rganishda bizga “**test**” nomli quyidagi jadval kerak bo‘ladi.

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Wayne	Rooney	18	21000
Sir	Alex	50	

Shu jadvalga so‘rovlar berish orqali “**where**” operatorini o‘rganamiz.

“**where**” operatori so‘rovni ikkiga bo‘lib beruvchi hisoblanadi. Undan so‘ng so‘rovga qo‘yiladigan shartlar yoziladi. Uning umumiy ko‘rinishi:

select * from test where …;

Har doim jadval nomidan so‘ng “**where**” ishlatiladi. Misol ko‘ramiz, “**test**” jadvalidagi, “maosh” ustuni 25000 ga teng bo‘lgan barcha qatorlarni chiqarish kerak.

select * from test where maosh=25000;

ISM	FAMILIYA	YOSH	MAOSH
Pablo	Aimar	30	25000

Agar sonli ustun emas, balki biror matn yo so‘zga tegishli bo‘lgan ustunga shart qo‘yib chiqarmoqchi bo‘lsak, matn yoki so‘z apostrof ichiga olinadi, aks holda xatolik yuzaga keladi.

select * from test where ism='David';

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000

Yuqoridagi shartlardan tashqari, yana quyidagi solishtirish belgilari orqali ham shart qo‘yish mumkin bo‘ladi.
= - tenglik belgisi
> - kichik belgisi
>= - kichik yoki tenglikni anglatuvchi belgi
< - katta belgisi
<= - katta yoki teng belgisi
<> - teng emasni anglatuvchi belgi.

Yuqoridagi belgilarga misol ko‘ramiz. “test” jadvalidagi yoshi 20 dan kichik bo‘lgan insonlarning familiyasini chiqaruvchi so‘rov quyidagicha bo‘ladi.

select familiya from test where yosh<20;

FAMILIYA
Rooney

Yoki bo‘lmasa, ismi “David” bo‘lmagan insonlar haqida ma`lumotni ekranga chiqarish so‘rovi.

select * from test where ism<>'David';

ISM	FAMILIYA	YOSH	MAOSH
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Wayne	Rooney	18	21000
Sir	Alex	50	

Agar biz biror aniq qiymatni bilmasak, u holda qandaydir oraliq orqali bizga kerak qiymatni topib olishimiz mumkin bo‘ladi. So‘rovda biror oraliqni ishlatish uchun “**between**” va “**and**” operatorlaridan foydalanamiz. Misol, yoshi 20 dan 25 gacha bo‘lgan insonlarning maoshini chiqaruvchi so‘rov tuzish kerak.

select maosh from test where yosh between 20 and 25;

MAOSH
30000
55000
27000

Oraliq qiymatlar emas aksincha bir necha aniq qiymatlar berib, shu orqali kerakli ma`lumotlarni chiqarish kerak bo‘lsa, “in” operatoridan foydalanamiz. Misol, yoshi 18 va 30 ga teng bo‘lgan ism va familiyalarni ekranga chiqaruvchi so‘rov quyidagicha bo‘ladi

select ism, familiya from test where yosh in(18, 30);

ISM	FAMILIYA
Pablo	Aimar
Wayne	Rooney

Agar biror ustunga hech qanday ma`lumot kiritilmagan bo‘lsa, shu bo‘sh ustunga tegishli qolgan ma`lumotlarni chiqarish kerak bo‘lsa, “**NULL**” kalit so‘zidan foydalanamiz. Misol, maoshi kiritilmagan ustunga tegishli bo‘lgan boshqa ustun ma`lumotlarini chiqaramiz.

select * from test where maosh is null;

ISM	FAMILIYA	YOSH	MAOSH
Sir	Alex	50	

Jamlash operatorlari: count, min, max, avg, sum va group by

Hisobotlarni tuzishda ishlatiladigan va ma`lumotlarni ma`lum bir ko‘rinishlarda jamlash uchun foydalaniladigan, “SQL” ning guruhli operatorlarini ishlatishda, bizga quyidagi “**test**” nomli jadval kerak bo‘ladi.

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Sir	Alex	50	
Phil	John_	16	12000

Odatda hisobotlar tuzishda, ma`lum bir jadvaldan foydalaniladi va bu jadvaldagi ma`lumotlarning umumiy hajmi katta rol o‘ynaydi. Hajm

– jadvaldagi qatorlar sonidir. Bu so‘rovni topishda bizga “**count**” operatori yordam beradi. Misol, “**test**” jadvalidagi barcha qatorlarning sonini topish so‘rovi tuzilsin:

```
select count(*) from test;
```

COUNT(*)
6

Ko‘rib turganingizdek qiyin emas. Bu operatorni “**distinct**” operatori bilan ham ishlatish mumkin. Misol, yoshi har xil bo‘lgan insonlarning soni qanchaligini aniqlovchi so‘rov tuzilsin.

```
select count(distinct yosh) from test;
```

COUNT(DISTINCTYOSH)
5

“**distinct**” operatori bo‘lgani uchun, 23 yoshi bir marotaba hisoblanmoqda.

Keyingi operatorimiz “**sum**” deb nomlanadi va faqat raqam tipidagi ustundagi ma`lumotlar uchun ishlatiladi. Barcha ustundagi sonlarning yig‘indisini topib beradi. Misol, ismi “**David**” bo‘lgan insonlarning maoshi yig‘indisi topilsin.

```
select sum(maosh) from test where ism='David';
```

SUM(MAOSH)
85000

Statistika ma`lumotlari bilan shug‘ullanuvchi korxonalarda, raqamlarning o‘rtacha qiymati katta ahamiyatga ega bo‘ladi. SQL bu masalani ham birgina operator bilan yechmoqda, bu operator “AVG” operatori. Misol, ismi 5 ta harfdan iborat bo‘lgan insonlarning o‘rtacha yoshi hisoblansin.

```
select avg(yosh) from test where ism like '_____';
```

AVG(YOSH)
24,6666667

Keyingi operatorlarimiz yuqoridagi operatorlardan ham sodda. Bu operatorlarimiz, sonli ustunlarning maksimal va minimal qiymatlarini aniqlab beradi. Misol, yoshi eng katta va eng kichik insonni topish so‘rovi tuzilsin.

select max(yosh) from test;

MAX(YOSH)
50

select min(yosh) from test;

MIN(YOSH)
16

Agar so‘rovda guruhli operatorlardan boshqa operatorlar ham ishlatilsa, guruhli operatorlarni “guruh” ekanligini aytuvchi “**group by**” operatoridan foydalanish kerak. Agar bir xil ma`lumotlar bo‘lsa, guruhlanadi, aks holda natijalar guruhlanmagan holda chiqariladi. Misol, ism bo‘yicha guruhlab, guruhlangan qatorlarni o‘rtacha yoshini chiqaruvchi so‘rov tuzish kerak.

select avg(yosh), ism from test group by ism;

AVG(YOSH)	ISM
30	Pablo
50	Sir
22	David
23	Carlos
16	Phil

Yana bitta misol ko‘ramiz, bu misolni yaxshilab o‘rganib chiqing. Jadvaldan insonlarning yoshi, ismi va familiyasiga oid ma`lumotlar

chiqarilsin, agar yuqoridagi ma`lumotlari bir hil bo‘lgan insonlar mavjud bo‘lsa, ularning kichigi natija sifatida chiqarilsin.

```
select min(yosh), ism, familiya from test group by ism, familiya;
```

MIN(YOSH)	ISM	FAMILIYA
23	Carlos	Puyol
30	Pablo	Aimar
50	Sir	Alex
16	Phil	John_
21	David	Beckham
23	David	Luiz

So‘rovdan ko‘rinib turibdiki, natijadagi ism va familiya ustuni guruhlanmoqda, lekin bir hil ismli va familiyali inson bo‘lmagani uchun, “avg(yosh)==yosh” ga bo‘lmoqda va so‘rov quyidagi oddiy so‘rovga aylanmoqda.

```
select yosh, ism, familiya from test;
```

SQLda qism so‘rovlar: *in, exists, not exists.*

SQL so‘rov tilida qism so‘rovlar tuzishda SQLning kalit so‘zlaridan foydalanamiz. Buning uchun bizga quyidagi jadval kerak bo‘ladi.

ID	NAME	YEAR
1	Marcos	1990
2	Antonio	1985
3	Jonny	1988
4	Shinji	1989
5	Rafaeli	1990
6	Rafaeli	1990

Dastlabki so‘rovda «**in**» kalit so‘zidan foydalanamiz. Qism dasturdan chiqqan natijalar, asosiy so‘rovda, «**in**» kalit so‘zi orqali tanlanib, mos kelganlari ekranga chiqariladi. Misol ko‘ramiz,

```
select * from test where year in(select year from test where year>1988);
```

ID	NAME	YEAR
6	Rafaeli	1990
5	Rafaeli	1990
1	Marcos	1990
4	Shinji	1989

Bu so‘rovni qism so‘rovidan natija chiqqach, quyidagicha kichik so‘rovga aylanib qoladi (bazadagi axborotlardan kelib chiqqan holda).

select * from test where year IN(1989, 1990);

ma`nosi, 1989 va 1990 yillarda tug‘ilgan o‘yinchilarni ekranga chiqaring.

Qism so‘rov tuzishda ishlatiladigan navbatdagi kalit so‘zimiz «***Exists***» deb nomlanadi. Bu kalit so‘z orqali biror ustun emas, butun boshli so‘rov tekshiriladi. Agar qism dasturdan «**true**» natija chiqsa, ya`ni qism so‘rovdan jadvalda mavjud bo‘lgan biror axborot chiqsa, asosiy so‘rov ishlaydi, aks holda hech qanday so‘rov ishlamasdan, ekranga hech narsa chiqmaydi.

select * from test where exists(select year from test where year=1990);

ID	NAME	YEAR
1	Marcos	1990
2	Antonio	1985
3	Jonny	1988
4	Shinji	1989
5	Rafaeli	1990
6	Rafaeli	1990

agar qism so‘rovdan natija olinmasa,

select * from test where exists(select year from test where year=1990);

asosiy so‘rov ishlamaydi. Bu operator uchun qanday natija chiqqani qiziq emas, balki natija chiqqan yoki chiqmagani qiziq. Bunga teskari

bo‘lgan operator «***Not exists***» deb nomlanib, ishlashi ham «**exists**»ning teskarisi.

Yuqoridagi operatorlar oddiy so‘rovlar uchun ham, qism so‘rovlar uchun ham ishlataveriladi, lekin navbatdagi operatorlar faqat qism so‘rovlar bilan ishlatiladi. Ular quyidagilardan iborat: ***ANY, ALL, SOME..***

«***ANY***» va «***SOME***» bir xil operator bo‘lib, istalganini ishlatishingiz mumkin, ikkisida ham bir xil natija chiqadi. Bu operatorlarning ishlashi, qism so‘rovdan chiqqan natijani istalgani asosiy so‘rov uchun qiymat bo‘lsa, asosiy so‘rov natija beraveradi. Bu operator «**in**» operatori bilan bir xil, faqat bularda «=» dan boshqa «<,<=,>, >=,<>» belgilarni ham ishlatish mumkin(**in** da faqat «=»).

select * from test where id>any(select id from test where id>4);

ID	NAME	YEAR
6	Rafaeli	1990

«**ALL**» qism so‘rovdagi barcha natijalar, asosiy so‘rovdagi shartni bajarishi lozim, shunda barcha shartdan o‘tgan natija ekranga chiqadi.

select * from test where id<all(select id from test where id>3);

ID	NAME	YEAR
1	Marcos	1990
2	Antonio	1985
3	Jonny	1988

SQL funksiyalari: *lower, upper* haqida

So‘rov tuzishda juda ko‘p ishlatiladigan va juda kerakli bo‘lgan ba`zi funksiyalar so‘rov kodlarida tez-tez ishlatilib turiladi va shu bilan birga o‘rganish ham qiyin emas. Bu kichik funksiyalarni o‘rganishda yana o‘sha biz bilgan “**test**” jadvali yordam beradi.

ISM	FAMILIYA	YOSH	MAOSH
David	Luiz	23	30000
David	Beckham	21	55000
Pablo	Aimar	30	25000
Carlos	Puyol	23	27000
Sir	Alex	50	
Phil	John_	16	12000

Demak boshladik, dastlabki funksiyamiz “**LOWER**” deb ataladi. Yozuv ko‘rinishidagi ustunlarni, biror qiymat bilan solishtirishda, yozuvning katta yoki kichik harfda ekanligini bilish muhim, aks holda kerakli natijaga erishish qiyinchilik tug‘diradi (sababi, katta kichik yozuvlar qo‘shtirnoq ichida farqlidir). Bu funksiya, solishtirilayotgan yozuvni barcha harflarini kichik harflarga o‘tkazib beradi va siz bemalol kichik harflar bilan solishtirishingiz mumkin bo‘ladi. Misol, familiyasi ***BECKHAM*** bo‘lgan inson haqida ma`lumot chiqaramiz (biz, bazada bu familiya katta yoki kichik harfda yozilganini bilmaymiz va familiya ustunidagi barcha familiyalarni kichik harfga o‘tkazib tekshiramiz, ya`ni ***Beckham*** => ***beckham,*** ***Aimar*** => ***aimar,*** ***Luiz*** => ***luiz*** ko‘rinishiga o‘tkazilib, so‘ng solishtiriladi):

```
select * from test where lower(familiya)='beckham';
```

ISM	FAMILIYA	YOSH	MAOSH
David	Beckham	21	55000

Barcha familiyalarni katta harfga o‘tkazib solishtiradi(***Beckham*** => ***BECKHAM,*** ***Aimar*** => ***AIMAR,*** ***Luiz*** => ***LUIZ***,… ko‘rinishiga o‘tkazilib solishtiriladi).

```
select * from test where upper(familiya)='BECKHAM';
```

ISM	FAMILIYA	YOSH	MAOSH
David	Beckham	21	55000

Having bo‘limi group by bo‘limi bo‘lgan hollarda ishlaydi. Having ni ishlatilishi deyarli **where** bilan bir hil. Faqat having bo‘limidagi so‘rov butun jadvallarga emas, **where** bo‘limi orqali olingan natijaga yoziladi. Quyida birdan ortiq takrorlanuvchi familiyalar va ularning takrorlanishlar sonini chiqaruvchi so‘rov keltirilgan:

Select familiya, count(familiya) as soni
From test Group by familiya
Having count(familiya)>2

Jadvallarga yangi yozuv qo‘shish **INSERT** operatori orqali amalga oshiriladi:
INSERT INTO <jadvallar nomi>(<Maydonlar ro‘yxati>)
VALUE(<qiymatlar ro‘yxati>)
Misol:

INSERT INTO test(familiya, ismi) Value(“Mallayev”,”Oybek”);

Mavjud yozuvni tahrirlash(qiymatini yangilash) **UPDATE** operatori orqali amalga oshiriladi.

UPDATE <jadval nomi> SET <maydon>=<qiymat> Where <shart>

Misol: Malllaev Oybekning tug‘ilgan kunini 1989 ga o‘zgartirish.

Update test set tug_yili=1989 Where familiya=”Mallayev” and ism=”Oybek”;

Jadvallardan yozuvlarni o‘chirish **DELETE** operatori orqali amalga oshiriladi.
Delete from <jadval nomi>
Where <shart>
Misol: Ismi Oybekga teng bo‘lgan qatorni o‘chirish.

Delete from test Where ism=”Oybek”

Bu so‘rov orqali Ismi Oybek bo‘lgan barcha qatorlar o‘chiriladi.

SQL so‘rovlarini baholanilishi bo‘yicha maxsus jadval

2-jadval

MBBT NOMI					
QISQACHA MAZMUNI					**MAX BALL**
Jadvallar uchun (Har bir jadval uchun 1 ballgacha)					**4**
Axborot modeli uchun (real bog‘langanligini ko‘rsatish, to‘g‘ri va teskari)					**1**
Yozuvlar soni uchun (har bir jadval uchun 10 ta yozuv, asosiy jadvalda 50 ta yozuv)					**1**
To‘planadigan umumiy ball					**6**
AMALIY QISM					
t.r.	**operator turi**	**so‘-rov soni (min)**	**qisqacha mazmuni**		**ball (max)**
ASOSIY OPERATORLAR					
1	INSERT INTO	3	turli tipdagi va bir nechta yozuvlarni qo‘shish		
2	SELECT	5	cheklangan sondagi yozuvlari chiqarish		
3	DELETE	2	shart orqali o‘chirish, hammasini o‘chirish		
4	UPDATE	3	turli tipdagi va bir nechta yozuvlarni hamda hammasini		
AMALLAR					
5	matematik amallar	2	kamida ikkitasi birga bo‘lsin		
6	munosabat amallar	3	kamida ikkitasi birga bo‘lsin		
7	mantiqiy amallar	2	kamida ikkitasi birga bo‘lsin		
8	arralash	2			
OPERATORLAR					
9	DISTINCT	2			
10	WHERE	2			
11	BETWEEN	2			
12	IN	2			
13	LIKE	4	fakat boshidan, oxiridan, ixtiyoriy joydan		
14	ISNULL	2	not bilan ham		
15	NOT	3	murakkab amallar bilan		
16	ORDER BY	2	xamma variantlar uchun		
17	HAVING	2			
STATIK FUNKSIYALAR					
18	MAX	1			
19	MIN	1			
20	COUNT	1			
21	SUM	1			
22	COUNT(*)	1			
23	AVG	1			
24	**ARRALASH**	3			

YORDAMCHI SO‘ROVLAR (SUBQUERY - PODCHINENNO`E ZAPROSO`) BILAN ISHLASH					
25	YORDAMCHI SO‘ROVLAR	3	kamida 3 ta statik funksiyalar, amallar, operatorlardan foydalang		
26	IN	2			
27	EXISTS	2			
28	ANY	2			
	ALL	2			
MAXSUS OPERATORLAR					
29	CAST	1			
30	CONVERT	2			
31	CASE	3			
32	COALESCE	1			
33	NULLIF	1			
BOG‘LANISHLAR					
34	ODDIY	2			
35	INNER JOIN	2			
36	LEFT OUTER JOIN	2			
37	RIGHT OUTER JOIN	2			
38	FULL OUTER JOIN	2			
39	CROSS JOIN	2			
40	UNION	2			
YOZUV FUNKSIYALARI					
41	FIRST	1			
42	UCASE	1			
43	LCASE	1			
44	UPPER	1			
45	LOWER	1			
46	LAST	1			
47	MID	1			
48	SUBSTRING	1			
49	LEN	1			
50	NOW	1			
51	SARLAHANI NOMLASH	1			
52	FORMAT	1			

SQL so‘rovlarini bajarilishi bo‘yicha maxsus jadval

3-jadval

	INSERT INTO GA DOIR
1	**INSERT INTO** Millat (millat, fullInfo)VALUES
2	**INSERT INTO** Millat (millat, fullInfo)VALUES (","), (",")
3	**INSERT INTO** Fuqaro(fuqaro, ishStaji, maoshi, ishManzili, uyManzili, tugilganSana,

	lavozimi, millati) VALUES(",12,1230000,1,1, '',1,1)
	SELECT GA DOIR
4	**SELECT** id, millat, fullInfo **FROM** Millat
5	**SELECT** TOP N id, millat, fullInfo **FROM** Millat
6	**SELECT** fuqaro, ishStaji, maoshi **FROM** FUQARO
	DELETE GA DOIR
7	**DELETE FROM** Millat
8	**DELETE FROM** Millat WHERE millat = ''
9	**INSERT INTO** Millat(millat,fullInfo) VALUES ('TATAR' , 'TATAR')
	UPDATE ga dior
10	**UPDATE** Millat SET millat = 'tatar'
11	**UPDATE** Millat SET millat = 'tatar' WHERE millat = 'TATAR'
12	**UPDATE** geografikManzil SET geografik M1 'Tashkent sh.', pochtaIndex = '100' WHERE geografik M1 = 'Tashkent'
	Amallar
13	**SELECT** fuqaro, maoshi*12 **FROM** Fuqaro
14	**SELECT** fuqaro, ishStaji **FROM** Fuqaro where Fuqaro.maoshi>130000*4
15	**SELECT** fuqaro Fuqaro.maoshi*.0065 **FROM** Fuqaro
16	**UPDATE** Fuqaro SET maoshi = maoshi*1.1
17	**SELECT** fuqaro ,maoshi,ishStaji **FROM** Fuqaro where (maoshi>1300*5 and ishStaji >=7)
18	**SELECT** fuqaro ,ishStaji **FROM** Fuqaro where not(maoshi<1300*5)
19	DISTINCT
20	**SELECT** DISTINCT geografik M1 **FROM** geografikManzil
21	BETWEEN
22	**SELECT** fuqaro ,maoshi ,ishStaji **FROM** Fuqaro WHERE ishStaji between 5 and 12;
	IN ga dior
23	**SELECT** fuqaro ,maoshi ,ishStaji **FROM** Fuqaro WHERE ishStaji not in(5,10)
	LIKE ga doir
24	**SELECT** fuqaro ,maoshi ,ishStaji **FROM** Fuqaro WHERE fuqaro like 'A%'
25	**isNUll**
26	**SELECT** fuqaro ,maoshi ,ishStaji **FROM** Fuqaro WHERE tugilganSana is not null
27	**SELECT** geografik M1, kucha uy **FROM** geografikManzil WHERE pochtalndex is null
	NOT ga doir
28	**SELECT** fuqaro ,maoshi,ishStaji **FROM** Fuqaro where not maoshi>1300*5 and ishStaji >=7
	ORDER BY ga doir
29	**SELECT** fuqaro **FROM** Fuqaro order by fuqaro desc
30	**SELECT** fuqaro **FROM** Fuqaro order by fuqaro
31	group by
32	**SELECT** millati **FROM** Fuqaro group by millati
	HAVING ga doir
33	**SELECT** millati **FROM** Fuqaro group by millati having millati>1
34	Statik funksiyalar
35	**SELECT** 'MAX'=MAX(maoshi),'MIN'=MIN(maoshi) **FROM** Fuqaro

36	**SELECT** COUNT(fuqaro) **FROM** Fuqaro where ishStaji >= 5
37	**SELECT** SUM(maoshi) **FROM** Fuqaro
38	**SELECT** COUNT(distinCT millati) **FROM** Fuqaro
39	**SELECT** COUNT(*) **FROM** Fuqaro
40	**SELECT** AVG(maoshi) **FROM** Fuqaro
	IN ga doir
41	**SELECT** fuqaro **FROM** Fuqaro WHERE millati in (**SELECT** id **FROM** millat where millat = 'uzbek')
42	EXISTS
43	**SELECT** Fuqaro.fuqaro **FROM** Fuqaro WHERE EXISTS (**SELECT** id **FROM** millat where millat = 'turman')
44	ANY and ALL
45	**SELECT** millati **FROM** Fuqaro group by millati having millati = ANY(**SELECT** id **FROM** Millat)
46	**SELECT** millati **FROM** Fuqaro group by millati having millati = ALL(**SELECT** id **FROM** Millat)
	CONVERT and CAST ga doir
47	**SELECT** geografik M1 **FROM** geografikManzil where CONVERT(int,telefonHome) > 0
48	**SELECT** CAST (millat as char) **FROM** Millat
	CASE ga doir
49	**SELECT** fuqaro, case when maoshi<=500000 then 'PAST' when (maoshi>500000 and maoshi<1000000) then 'URTACHA' else 'YAXSHI' end **FROM** Fuqaro;
	COALESCE ga doir
50	**SELECT** geografik M1 coalesce'telefonHome, - 1 **FROM** geografikManzil
51	NULLIF
52	**SELECT** geografik M1 NULLIF telefonHome ") AS NNN **FROM** geografikManzil
	Bog‘lanishlar
53	**SELECT** Fuqaro, millat **FROM** Fuqaro, Millat where Fuqaro.millati = Millat.id
54	**SELECT** Fuqaro, millat **FROM** Fuqaro inner join Millat ON Fuqaro.millati = Millat.id
55	**SELECT** Fuqaro, millat **FROM** Fuqaro Full outer join Millat ON Fuqaro.millati = Millat.id
56	**SELECT** Fuqaro, millat **FROM** Fuqaro Left outer join Millat ON Fuqaro.millati = Millat.id
57	**SELECT** Fuqaro, millat **FROM** Fuqaro RIGHT outer join Millat ON Fuqaro.millati = Millat.id
58	**SELECT** Fuqaro, millat **FROM** Fuqaro,Millat
59	**SELECT** Fuqaro, millat **FROM** Fuqaro cross join Millat
60	**SELECT** Fuqaro, millat **FROM** Fuqaro cross Join Millat where Fuqaro.millati = Millat.id

	UNION ga doir
61	(**SELECT** Fuqaro **FROM** Fuqaro where maoshi < 6*130000) UNION (**SELECT** Fuqaro **FROM** Fuqaro inner join Millat ON Fuqaro.millati = Millat.id where Millat.millat ='uzbek')
62	YOzuv funksiyalari
63	**SELECT** First(fuqaro)**FROM** fuqaro
64	**SELECT** Ucase(fuqaro) **FROM** Fuqaro
65	**SELECT** LCASE(fuqaro) **FROM** Fuqaro
66	**SELECT** UPPER(fuqaro) **FROM** Fuqaro
67	**SELECT** LOWER(fuqaro) **FROM** Fuqaro
68	**SELECT** LAST(fuqaro) **FROM** Fuqaro
69	**SELECT** MID(fuqaro,1,1) **FROM** Fuqaro
70	**SELECT** fuqaro **FROM** Fuqaro where SUBSTRING(fuqaro,1,1)= 'A'
71	**SELECT** len(fuqaro) **FROM** Fuqaro
72	**SELECT** fuqaro, Now() as SANA **FROM** Fuqaro
73	**SELECT** Round(maoshi) **FROM** Fuqaro
74	**SELECT** fuqaro, FORMAT(Now(),'YYYY-MM-DD') AS PerDate **FROM** Fuqaro
	Commit RollBack ga doir
75	BEGIN TRANSACTION W1; **INSERT INTO** MILLAT(MILLAT,FULLINFO) VALUES ('TATAR12 5','TATAR12 5') ; **UPDATE** GEOGRAFIKMANZIL SET GEOGRAFIK M1 = 'TOSHKENT', POCHTAINDEX = '100' WHERE GEOGRAFIK M1 = 'TASHKENT'; COMMIT TRANSACTION W1;
76	BEGIN TRANSACTION WORK; **INSERT INTO** MILLAT(MILLAT,FULLINFO) VALUES ('TATAR-','TATAR-3'); **UPDATE** GEOGRAFIKMANZIL SET GEOGRAFIK M1 = 'TASHKENT', POCHTAINDEX = '100' WHERE GEOGRAFIK M1 = 'TSHKENT'; ROLLBACK WORK;
77	BEGIN TRANSACTION WORK; **INSERT INTO** MILLAT(MILLAT, FULLINFO) VALUES('KKKK1',''); **DELETE FROM** MILLAT WHERE MILLAT LIKE 'KK'; DECLARE @MAX1 INT; **SELECT** @MAX1 = MAX(MAOSHI) **FROM** FUQARO; IF(@MAX1 < 1300000) COMMIT WORK ELSE ROLLBACK WORK;
	Procedure lar bilan ishlash
78	ALTER PROCEDURE TEST as BEGIN DEclare @I iNT; **SELECT** @I = COUNT(fuqaro) **FROM** Fuqaro **SELECT** @I; END

	EXEC TEST
79	ALTER PROCEDURE TEST as BEGIN DEclare @I iNT; **SELECT** @I = COUNT(fuqaro) **FROM** Fuqaro return @I; END DECLARE @return_value int EXEC @return value = TEST if @return value) < 10 Print N'10 dan kam' else Print N'10 dan ko'p'

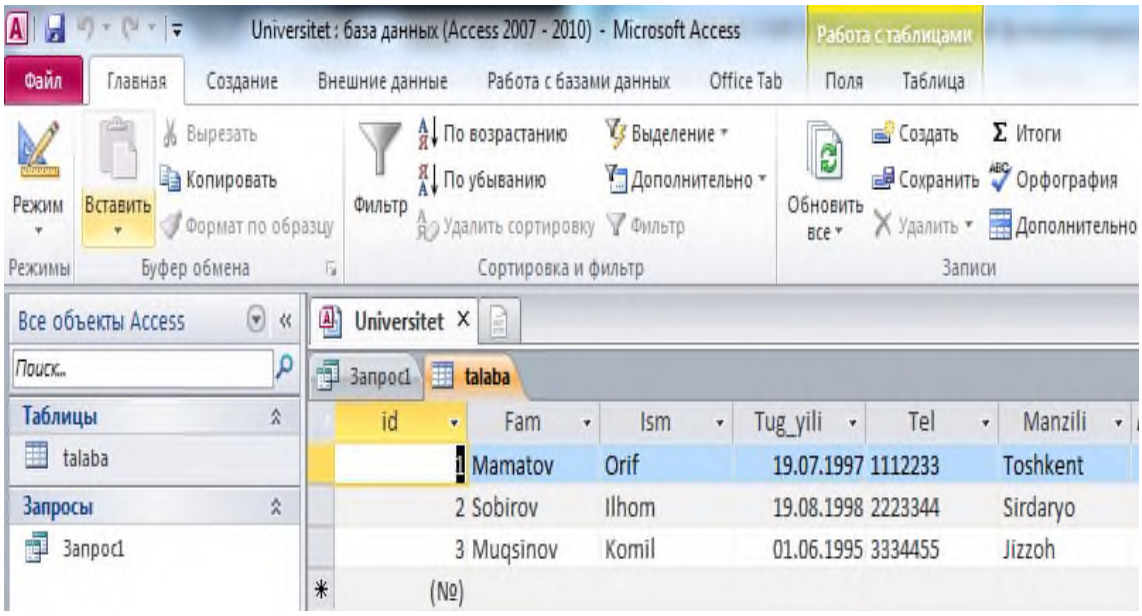
4.3. Ma`lumotlar bazasini yaratish bo‘yicha uslubiy ko‘rsatmalar

MS Access dasturida SQL so‘rovlaridan foydalnish:

Oldingi bo‘limlarda **MS Access** dasturida ma`lumotlar bazasini yaratish ko‘rsatib o‘tilgan edik. Endi esa MB dagi jadvallar asosida SQL so‘rovlarni amalga oshirish ko‘rsatiladi.

Masalan: **talaba** jadvalida familiyasi M harfidan boshlanuvchi talabalarni chiqaruvchi so‘rov yaratish.

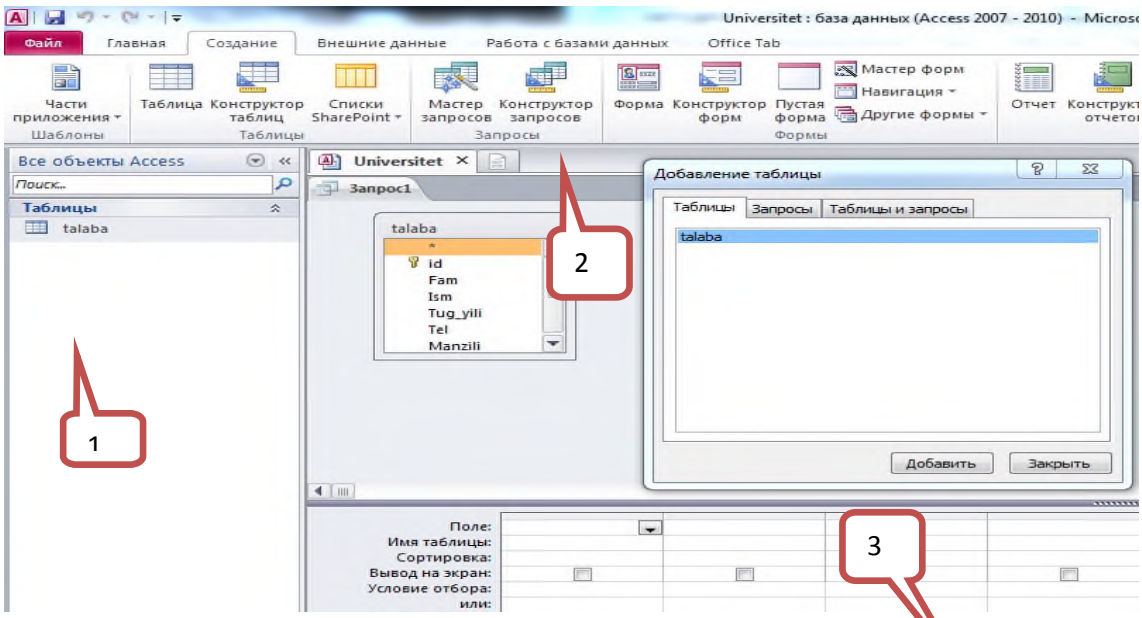
1. **MS Access** dasturi ishga tushiriladi va unda **Universitet** nomli MB yaratiladi;
2. **MB** da **talaba** jadvali yaratiladi;
3. **talaba** jadvalida **Fam, Ism, Tug_yili, Tel, Manzili** nomli maydonlari tashkil etiladi va to‘ldiriladi:



4.20-rasm. Talaba jadvalining ma`lumotlari

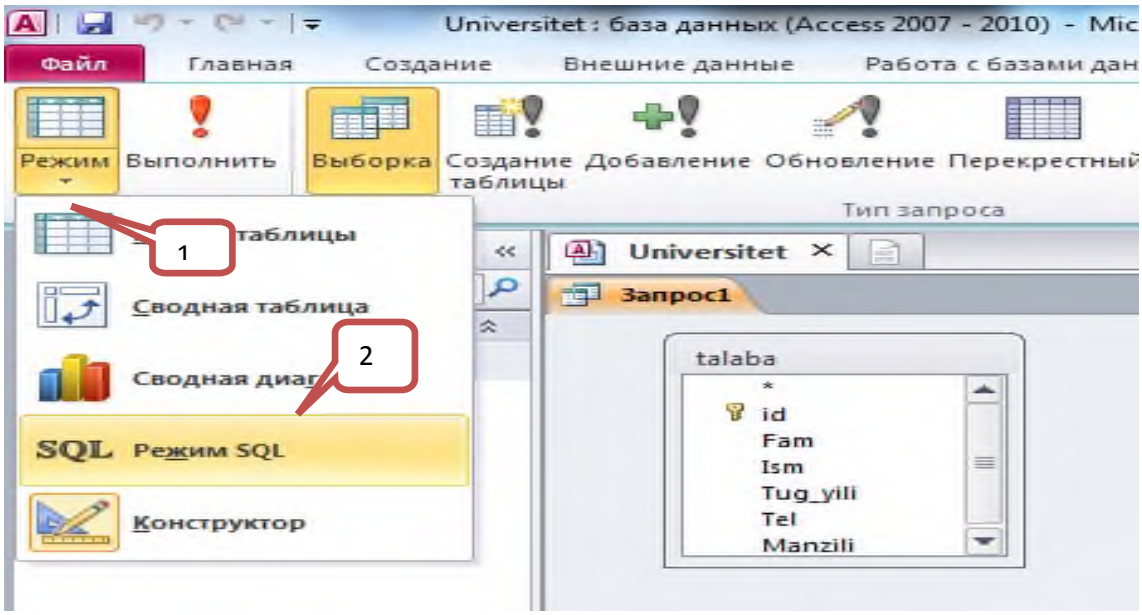
4. **talaba** jadvalidan SQL so‘rovlarni yaratiladi:

“Создание” menyu bo‘limidan **“Конструктор запросов”** tanlanadi va 4.21- rasmda ko‘rsatilgan ketma – ketliklar bajariladi.



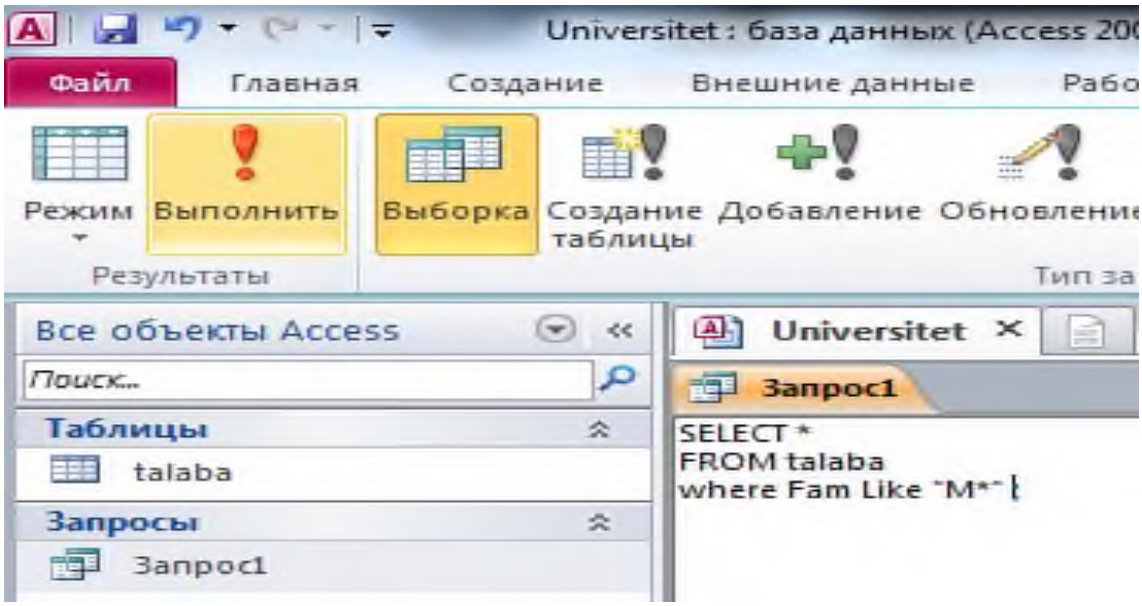
4.21-rasm. Konstruktor rejimida so‘rov yaratish

Jadval tanlanangandan keyin, **“Режим”** aktivlashadi va 4.22- rasmda ko‘rsatilgan ketma – ketliklar bajariladi.



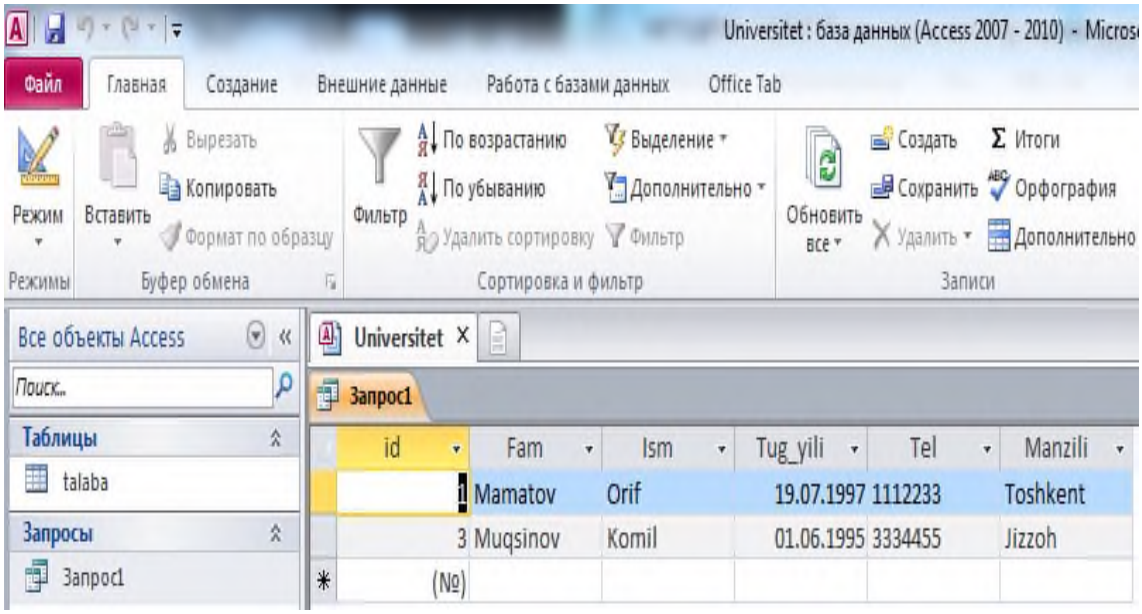
4.22-rasm. SQL so‘rovlarini yaratish

So‘rov yozilgandan keyin “**Выполнить”** tugmasi bosiladi.



4.23-rasm. SQL so‘rovlarini bajarish

MS Accessda SQL so‘rov yaratishda % o‘rniga * qabul qilingan. “**Выполнить”** tugmasi bosilgandan keyin quyidagi jadval paydo bo‘ladi:



4.24-rasm. SQL so‘rovining natijasi

Nazorat savollari:

1. SQL nima?
2. SQL ning maqsad va vazifasi nimalardan iborat?
3. Ma`lumotlarni chaqirish operatori nima?
4. Ma`lumotlarni yangilash operatori nima?

5. Ma`lumotlarni yozish operatori nima?
6. Ma`lumotlarni o‘chirish operatori nima?

4.4. Visual C++ da MS Access ning ma`lumotlar bazasini SQL so‘rovlari asosida tahrirlash usullari

Mavzuning maqsadi

Visual C++ ning **Console Application** muhitida **MS Access** ma`lumotlar bazasini yaratuvchi, unga ma`lumotlarni yozib va undan ma`lumotlarni o‘qib hamda o‘zlashtirilgan ma`lumotlarni tahrirlab ekranga chiqaruvchi dastur tuzish.

1-dastur: Command va **Datareader** sinf ob`ektlari yordamida **MS Access**da yaratilgan ma`lumotlar bazasining jadvalidagi barcha ma`lumotlarni o‘quvchi dastur

1-dastur: Shunday dastur tuzaylikki, kam kodlarni ishlatib, ma`lumotlar bazasi (MB) ning jadvalidagi barcha ma`lumotlarni ekranga chiqarsin. Buning uchun zamonaviy **ADO.NET** texnologiyasi ishlatiladi. Bunga 4 ta ob`ekt kerak bo‘ladi. 1. **Connection** ob`ekti –MB bilan ulanishni ta`minlaydi. 2. **Command** ob`ekti – MB bilan **SQL** ifodalarini bog‘lashni ta`minlaydi. 3. **Dataset** va **Datareaders** ob`ektlari yordamida so‘rovlar natijasini ko‘rish imkonini beradi.

Dasturda MB bo‘yicha 4 ta asosiy xarakatlarni ko‘rib chiqiladi: **Select** (MB jadvalidan yozuvlarni tanlash), **Insert** (bazaga yozish), **Update** (MBning jadvalidagi ma`lumotlarni yangilash), **Delete** (Jadvaldan ma`lumotlarni o‘chirish uchun).

Visual Studio ni ishga tushiramiz va **Console Application CLR** muhitida ishlovchi loyiha yaratiladi va quyidagi kodlar kiritiladi:

```
1. // Access_console.cpp : main project file.
2. #include "stdafx.h"
3. using namespace System;
4. using namespace System::Data::OleDb; // SqlServerCe;
5. int main(array<System::String ^> ^args){
6. //konsolning matniga rang berish:
7. Console::ForegroundColor = ConsoleColor::Yellow;
8. // Connection sinfi ob`ektini yaratish
9. auto Ulanish = gcnew OleDbConnection();
10. // Uni ulanish qatoriga yuborish:
```

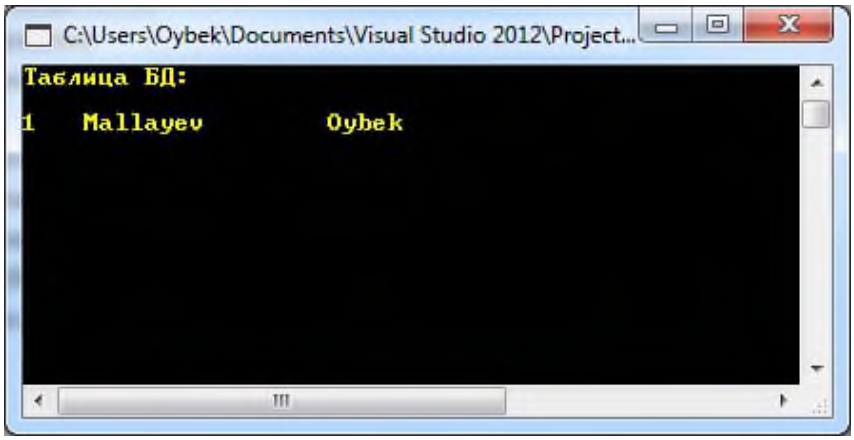
```
Ulanish->ConnectionString = "Data Source=\"Oybek.mdb\";User " +
"ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";";
Ulanish->Open();
// Command sinfi ob`ektini yaratish:
auto Komanda = gcnew OleDbCommand();
Komanda->Connection = Ulanish;
// Uni SQL-komandalariga jo‘natish :
Komanda->CommandText = "Select * From [Talaba]";
// Xamma yozilganlar tanlanadi va FIO ustuni bo‘yicha
saralanadi:
// Komanda->CommandText = "Select * From [BD telefonov] order by
FIO";
//"Nomer p/p" usutuni bo‘yicha esa:
// Komanda->CommandText = "Select * From [BD telefonov] ORDER BY
'Nomer p/p'";
// SQL-komandasini bajarish:
OleDbDataReader ^ CHitatel = Komanda->
 ExecuteReader(System::Data::CommandBehavior::CloseConnection);
Console::WriteLine("Jadval BD:\n");
while (CHitatel->Read() == true)
// sikl xali xamma yozuvlarni o‘qigani yo‘q
// CHitatel->FieldCount - qatorlardagi maydonlar soni.
// Bu yerda 3 ta maydon: 0, 1 i 2.
// Minus prijimaet stroku vlevo:
Console::WriteLine("{0,-3} {1,-15} {2,-15}",
CHitatel->GetValue(0),
CHitatel->GetValue(1), CHitatel->GetValue(2));
CHitatel->Close(); Ulanish->Close();
// qandaydir tugmani bosilguncha kutib turadi
Console::ReadKey(); return 0;}
```

Dastur tahlili:

Dastur kodidan ko‘rinib turibdiki, avval **Connection** sinfning Ulanish ob`ekti yaratildi va bog‘lanish satriga yuborildi. Bog‘lanish satri **mdb** faylga hamma ruxsatni oldi. Bu esa fayl va kataloglarning yo‘llarini aniq o‘qish uchun mo‘ljallangan. Keyin **Command** sinf ob`ekti yaratildi va uni SQL komandasiga yuborildi.

Select * From [**BD telefonov**]

Bu holda BD telefonov jadvalidagi hamma yozuvlar tanlanadi. FIO bo‘yicha (**ORDER BY FIO**) va nomeri bo‘yicha (**ORDER BY 'Nomer p/p'**) saralanadi. Keyin SQL buyruqlarini bajarish uchun **DataReader** sinf ob`ekti ishlatildi. Dastur natijasi 4.25- rasmda keltirilgan:



4.25-rasm. MBning jadvalini konsol muhitida chiqarish

2-dastur: Console Application muhitida MS Access ma`lumotlar bazasini yaratuvchi dastur tuzish

Visual Studio 2012 ning **Console Application** muhitida **new_BD.mdb** nomli fayl (Ma`lumotlar bazasi)ni yaratishda avval ushbu bazada hech qanday jadval bo‘lmaydi. Jadvallarni dasturlash kodi yordamida yaratiladi. Ushbu dasturda **ADO.NET** texnologiyasi ishlatilmaydi.

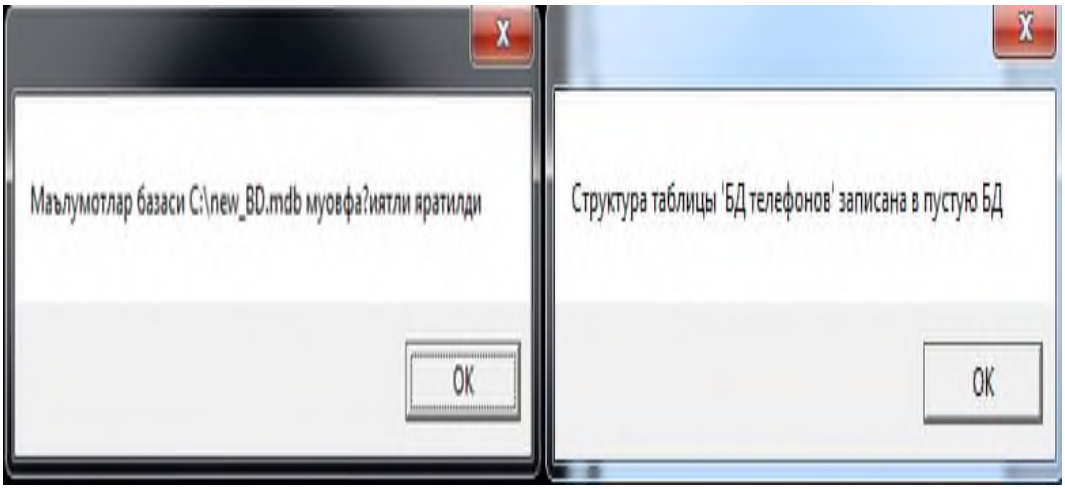
Visual Studio dasturi ishga tushiriladi. **New Project** oynasidan **CLR** bo‘limi tanlanadi va **Console Application CLR** muhitida yangi loyiha dasturi yaratiladi**. ADOX DLL** kutubxonasini loyihaga qo‘shish uchun dasturning **Project** mensidan **Properties Add Reference** tanlanadi, ochilgan oynadan **COM** bo‘limiga o‘tiladi va **Microsoft ADO Ext. 2.8 for DDL and Security** belgilanib OK tugmachasi bosiladi. Natijada dasturdan **ADOX** kutubxonasi funksiyalariga murojaat qilish imkoniyatlari ochiladi.

```
1. // Access_mb_yaratish_Consol.cpp : main project file.
2. #include "stdafx.h"
3. using namespace System;
4. using namespace System::Windows::Forms;
5. using namespace System::Data::OleDb;
6. using namespace System::Data::SqlClient;
7. int main(array<System::String ^> ^args){
8. ADOX::Catalog ^ Katalog = gcnew ADOX::Catalog();
9. try {
10. Katalog->Create("Provider=Microsoft.Jet." +  "OLEDB.4.0;Data
   Source=C:\\new_BD.mdb");
11. MessageBox::Show("Ma`lumotlar bazasi C:\\new_BD.mdb
   muvovfaqiyatli yaratildi");   }
12. catch (System::Runtime::InteropServices::COMException ^
   Situasiya)
```

```
13. { MessageBox::Show(Situasiya->Message); }
14. finally
15. { Katalog = nullptr; }
16. auto Ulanish = gcnew OleDbConnection(
17. "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\new_BD.mdb");
18. // podklyucheniya ni ochish:
19. Ulanish->Open();
20. // Command sinfining obe`ktini SQL so‘rovlari bilan yaratish
21. auto Komanda = gcnew OleDbCommand("CREATE TABLE [" +
22. "BD telefonov] ([Nomer p/p] counter, [FIO] ch" +
    "ar(20), [Nomer telefona] char(20))", Ulanish);
23. try{ // SQL buyrug‘larini bajarilishi:
24. Komanda->ExecuteNonQuery();
25. MessageBox::Show("Struktura tablisi 'BD telefonov' zapisana v
    pustuyu BD");}
26. catch (Exception ^ Situasiya)  { MessageBox::Show(
    Situasiya->Message); }
27. Ulanish->Close();     return 0; }
```

Dastur tahlili:

MessageBox ob`ektiga murojaat qila olinishi uchun dasturga yana bitta **DLL** kutubxona qo‘shildi. Buning uchun oldingi holatda tushuntirilganidek menyuning **Project->Properties->Add Reference** buyrug‘i berildi va **NET** bo‘limidan **System.Windows.Forms.dll** belgilandi. Dastur kodiga **Using namespace System::Windows::Forms;** havolasi kiritildi. Dastur quyidagi ko‘rinishda ishlaydi: **ADOX::catalog** sinfi yaratiladi, uning **Great** funksiyasi bilan MB yaratiladi. **try...catch** bilan esa qayta ishlanadi. Agar dastur ishga tushsa **new_BD.mdb** nomli ma`lumotlar bazasi yaratiladi. Lekin uning ichida hech qanday jadval yo‘q va quyidagi xabarnoma chiqadi.



4.26-rasm. MB ni yaratilganligi haqida dasturning natija oynasi

3-dastur: Console Application muhitida MS Access ma`lumotlar bazasining jadvallariga ma`lumotlar yozuvchi dastur tuzish

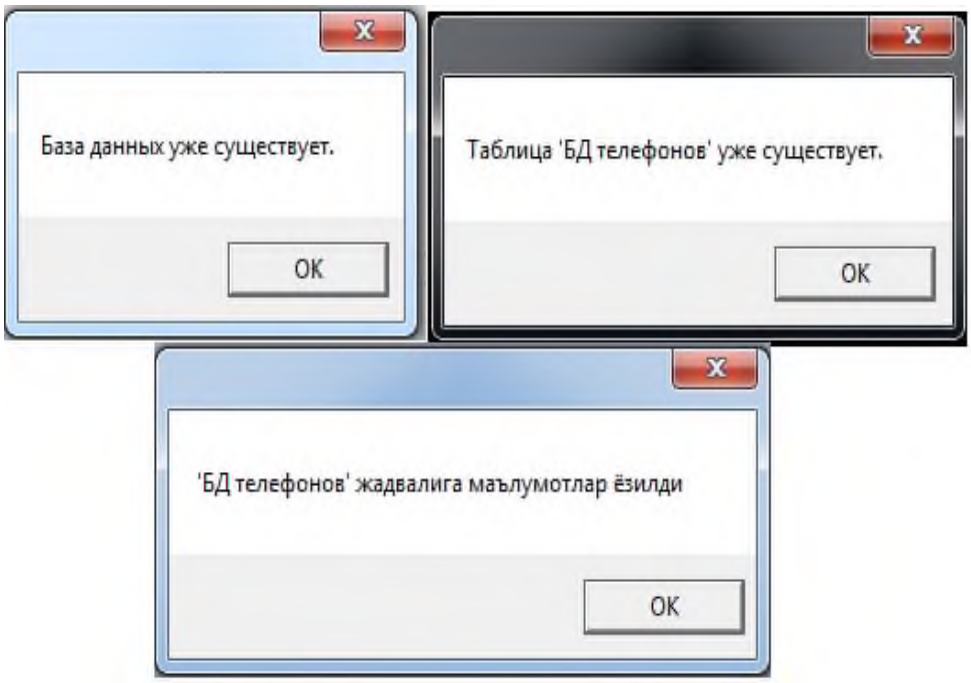
Ma`lumotlar ba`zasi bilan muloqot qilish uchun 2- dasturni ishlatib, xohlagancha so‘rovlarni amalga oshirish mumkin. Masalan MB ning jadvallariga **INSERT** so‘rovi yordamida ma`lumot yozish mumkin. Buning uchun **Command** ob`ekti yaratiladi va unga **SQL** so‘rovlari beriladi. Buning uchun **insert** so‘rovi komanda ob`ektiga uzatiladi: “**Komanda->Connection = Ulanish”;**

Dastur kodi quyidagicha:

```
// Access_mb_yaratish_Consol.cpp : main project file.
1. #include "stdafx.h"
2. using namespace System;
3. using namespace System::Windows::Forms;
4. using namespace System::Data::OleDb;
5. using namespace System::Data::SqlClient;
6. int main(array<System::String ^> ^args){
7. ADOX::Catalog ^ Katalog = gcnew ADOX::Catalog();
8. try {
9. Katalog->Create("Provider=Microsoft.Jet." +  "OLEDB.4.0;Data
10. Source=C:\\new_BD.mdb");
11. MessageBox::Show("Ma`lumotlar bazasi C:\\new_BD.mdb
    muvaffaqiyatli yaratildi"); }
12. catch(System::Runtime::InteropServices::COMException ^
    Situasiya){
13. MessageBox::Show(Situasiya->Message); }
14. finally  { Katalog = nullptr; }
15. auto Ulanish = gcnew OleDbConnection("Provider=
    Microsoft.Jet.OLEDB.4.0;Data Source=C:\\new_BD.mdb");
16. // podklyucheniya ni ochish:
17. Ulanish->Open();
18. //Command sinf obe`ktini SQL so‘rovlari bilan yaratish
19. auto Komanda = gcnew OleDbCommand("CREATE TABLE [" + "BD
    telefonov] ([Nomer p/p] counter, [FIO] ch" +  "ar(20), [Nomer
    telefona] char(20))", Ulanish);
20. try{ // SQL buyrug‘larini bajarilishi:
    Komanda->ExecuteNonQuery();  MessageBox::Show("Struktura
    tabliso` 'BD telefonov' zapisana v pustuyu BD");  }
21. catch (Exception ^ Situasiya) { MessageBox::Show(
22. Situasiya->Message); }
23. Komanda = gcnew OleDbCommand( "INSERT INTO [BD telefonov] (" +
    "FIO, [Nomer telefona]) VALUES ('Sveta-X', '521-61-41')");
24. // MB Jadvaliga yozish uchun ushbu komanda albatta //bo‘lishi
    kerak:
25. Komanda->Connection = Ulanish;
26. // SQL buyruqlarini bjarish:
27. Komanda->ExecuteNonQuery();
```

```
28. MessageBox::Show(" 'BD telefonov' jadvaliga ma`lumotlar
    yozildi");
```

Dastur natijasi quyidagicha:



4.27-rasm. Ma`lumotlar yozilganligi haqida dasturning natija oynasi

Mustahkamlash uchun mashqlar

a) Quyidagi topshiriqlar asosida MB jadvallaridan bittasini tanlang va uni “**Console Application”** muhitida yarating.

b) Ushbu jadvalni SQL so‘rovlari (**select, insert, update**) ni ishlatgan holda ma`lumotlar bilan to‘ldiring, yangilang va ushbu ma`lumotlarni “**Console Application”** muhitida chiqaruvchi dastur tuzing.

1. Mahalla ma`lumotlar bazasi;
2. Maktab o‘quvchilarining ma`lumotlar bazasi;
3. Shifoxonaning qabul bo‘limini ma`lumotlar bazasi;
4. Fakultetning ma`lumotlar bazasi;

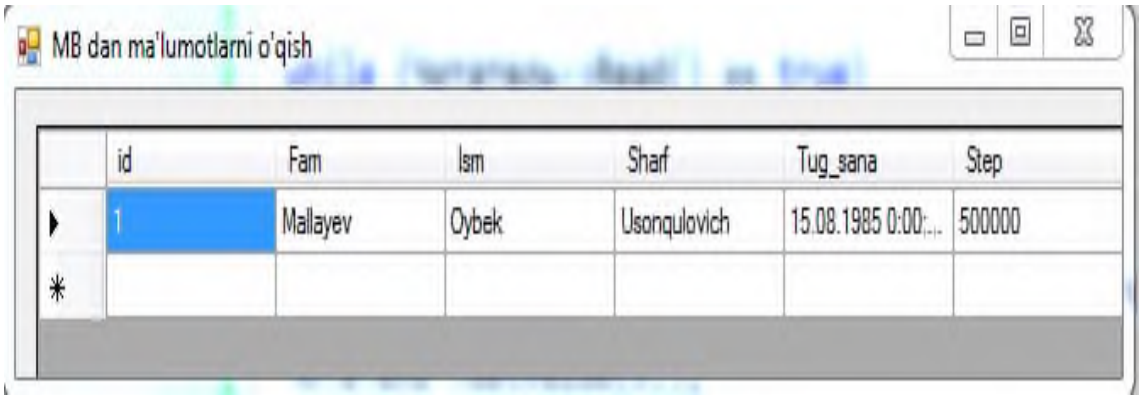
4-dastur: Command, DataReader sinf ob`ektlari va DataGridView komponentasi yordamida MBning jadvalidan ma`lumotlarni o‘quvchi vizual dastur

Ushbu dasturni yaratishda **Command, DataReader** sinf ob`ektlari va **DataGridView** komponentasi kerak bo‘ladi. **DataGridView**

komponentasidan MB ga ma`lumotlarni o‘qish uchun foydaniladi. Buning uchun **Windows Application** muhitida yangi loyiha yaratiladi va formaga **DataGridView, Button** komponentasi tashlanadi va **Form1** da dastur kodi quyidagicha bo‘ladi:

```
#pragma endregion
private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^e) {
this->Text="MB dan ma'lumotlarni o'qish";
auto Ulanish = gcnew OleDb::OleDbConnection("Data
Source=\"C:\\Oybek.mdbg`"; User " +
"ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";");
Ulanish->Open();
// Command obyektini yaratib, unga SQL-komandasini yuborish
auto Komanda = gcnew OleDb::
OleDbCommand("Select * From [talaba]", Ulanish);
// SQL-komandasini bajarish
auto CHitatel = Komanda-> ExecuteReader();
// (CommandBehavior.CloseConnection)
auto Jadval = gcnew DataTable();
// Jadvalni to’ldirish
for(int i=0;i<6;i++)
{Jadval->Columns->Add(CHitatel->GetName(i));}
while (CHitatel->Read() == true)
// Jadval katakchasini to’ldirish
Jadval->Rows->Add(CHitatel->GetValue(0),
CHitatel->GetValue(1), CHitatel->GetValue(2),
CHitatel->GetValue(3), CHitatel->GetValue(4),
CHitatel->GetValue(5));
// bu yerda 6 maydon: 0, 1, 2,3, 4, 5 bor
CHitatel->Close(); Ulanish->Close();
dataGridView1->DataSource = Jadval; }}; }
```

Dastur ko‘rinishi quyidagicha:



4.28-rasm. DataGridView komponentasiga MBdan ma`lumotlarni chiqarish

5-dastur: Command, Adapter va DataSet sinf ob`ektlari va DataGridView komponentasi yordamida MBning jadvalidan ma`lumotlarni o‘quvchi vizual dastur

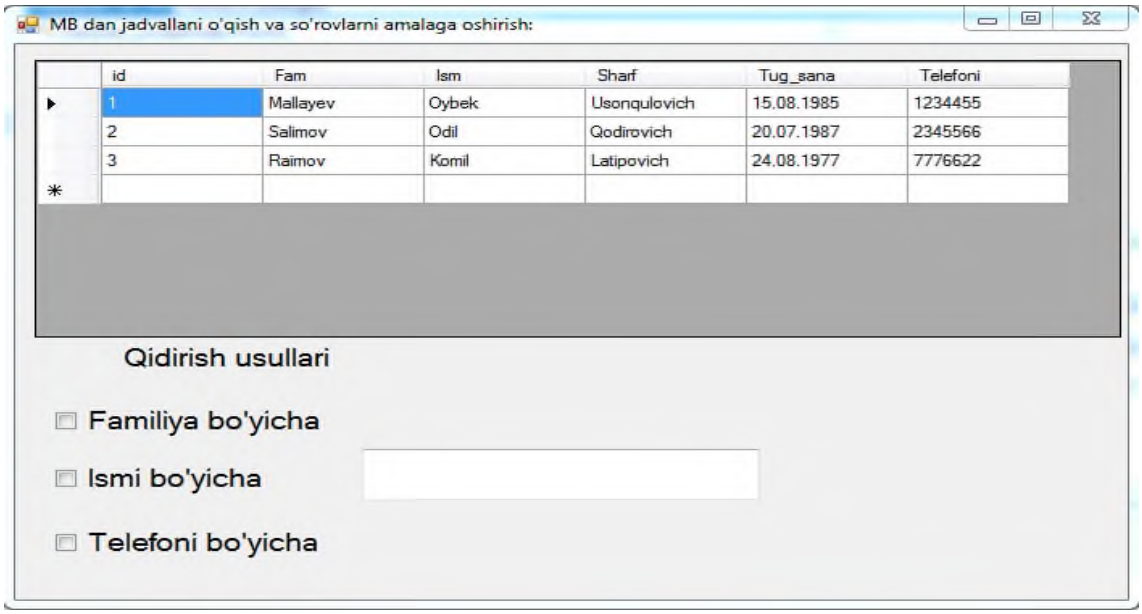
MB dan **Adapter** sinf ob`ekti bilan jadvallarnini muhim ma`lumotlarini o‘qish va ularni **DataSet** sinf ob`ektiga jo‘natishni ko‘rib chiqiladi. **DataGridView** komponentasidan foydalanib MB ning jadvallaridagi ma`lumotlarni **DataSet** dan o‘qish dasturiga juda qulay hisoblanadi. Misol sifatida quyidagi dasturni ko‘rish mumkin:

Dasturning kodi:

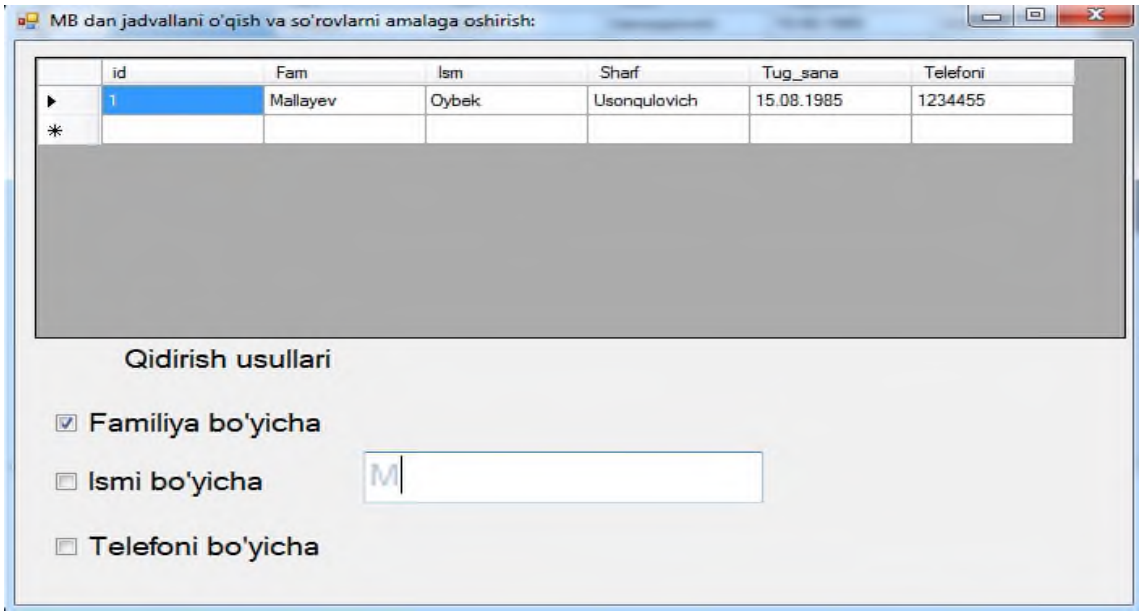
```
#pragma endregion
void DBSet(String ^s){
this->Text = "MB dan jadvallani o'qish va so'rovlarni amalaga oshirish:";
auto Ulanish = gcnew OleDb::OleDbConnection( "Data Source=\"C:\\Oybek.mdbg`"; User " +
"ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";");
Ulanish->Open();
auto Komanda =  gcnew OleDb::OleDbCommand(s, Ulanish);
// Adapter sinfi ob`ektini yaratamiz va SQL-so‘rovini //amalga oshiramiz
 auto Adapter = gcnew OleDb::OleDbDataAdapter(Komanda);
// DataSet sinfi ob`ektini yaratamiz
auto NaborDannix = gcnew DataSet();
// DataSet ni SQL-so‘rovini natijalari bilan //to‘ldiramiz
Adapter->Fill(NaborDanno`x, "talaba");
// DataSet da satrlar ko‘rinishi uchun XML ko‘rinishiga //o‘tadi:
auto StrokaXML = NaborDanno`x->GetXml();
// Komponentaga ma`lumot manbasini ko‘rsatamiz:
dataGridView1->DataSource = NaborDannix;
// Qidiriladigan malumotlarning jadvalini nomini //ko‘rsatamiz:
dataGridView1->DataMember = "talaba"; Ulanish->Close();        }
private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e) {
String ^satr="SELECT * FROM [talaba] WHERE (Fam LIKE '%M%')";
DBSet(satr); }
private: System::Void textBox1_TextChanged( System::Object^ sender, System::EventArgs^  e){ if(checkBox1->Checked){
DBSet("SELECT * FROM [talaba] WHERE (Fam LIKE '"+ textBox1->Text+"%')");}
else if(checkBox2->Checked){ DBSet("SELECT * FROM [talaba] WHERE (Ism LIKE '"+ textBox1->Text+"%')");}
else  if(checkBox3->Checked){ DBSet("SELECT * FROM [talaba] WHERE (Telefoni LIKE '"+textBox1->Text+"%')");}
```

```
27. else { MessageBox::Show("Qidirish turlaridan bittasini
    tanlang");
28. } } };
```

Dasturning ko‘rinishi quyidagicha:



4.29-rasm. MBdan barcha ma`lumotlarni chiqarish holati



4.30-rasm. MBdan barcha ma`lumotlarni qidirish holati

Dastur tahlili:

Dasturdan ko‘rinib turibdiki, **Connection** sinf ob`ekti yordamida MB bilan bog‘lanish amalga oshirildi. Jadvaldan hamma yozuvlarni

o‘qish va qidirish uchun **SQL** so‘rovlari (**Select, Like**)dan foydalanildi. Dastur formasiga 3 ta **CheckBox** va **TextBox** komponentalari turli parametrlar asosida qidirishni tashkil etish maqsadida joylashtirildi. Dastur kodida DBSet() nomli funksiya yaratildi va uning yordamida MBdan parametrli qidirish amalga oshirildi. **TextBox** komponentasining **onchange** xodisasiga ma`lum shartlar (Fam, Ism, Tel bo‘yicha) asosida DBSet() funksiyasi chaqirildi (**if(checkBox1->Checked){ DBSet("SELECT * FROM [talaba] WHERE (Fam LIKE '"+textBox1->Text+"%')");}**).

6-dastur: MS Access ning MBdagi jadval yozuvlarini yangilovchi vizual dastur

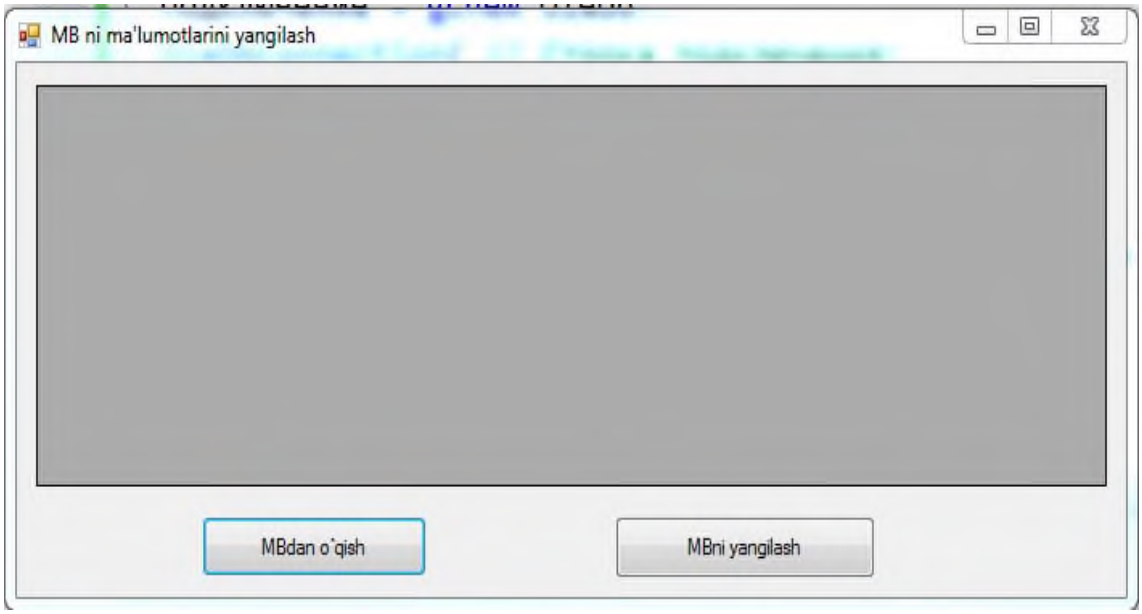
Ushbu dasturni tuzish uchun **Visual C++** dasturida **Windows Form Application** muhitida yangi loyiha yaratiladi. Formaga 2 ta **Button** va **DataGridView** komponentalari joylashtiriladi.

Dasturning kodlari quyidagicha:

```
1. #pragma endregion
2. DataSet ^ NaborDannix;
3. OleDb::OleDbDataAdapter ^ Adapter;  OleDb::OleDbConnection ^
   Ulanish;
4. OleDb::OleDbCommand ^ Komanda;
5. private: System::Void Form1_Load(System::Object^  sender,
   System::EventArgs^  e){
6. NaborDannix = gcnew DataSet();
7. Ulanish = gcnew OleDb::OleDbConnection( "Data
   Source=\"C:\\Oybek.mdb\";User "
   +"ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";");
8.  Komanda = gcnew OleDb::OleDbCommand();
9.  button1->Text = "MBdan o`qish";button1->TabIndex = 0;
10. button2->Text = "MBni yangilash"; this->Text="MB ni
   ma'lumotlarini yangilash";}
11. private: System::Void button1_Click(System::Object^  sender,
    System::EventArgs^  e){
12. // MB dan o‘qish:
13. if (Ulanish->State ==  ConnectionState::Closed) Ulanish->Open();
14. Adapter = gcnew OleDb::OleDbDataAdapter("Select * From
   [talaba]", Ulanish);
15. // DataSet ni SQL so‘rovlari natijasi bilan to‘ldirish
16. Adapter->Fill(NaborDannix, "talaba");
17. String ^ StrokaXML = NaborDannix->GetXml();
18. dataGridView1->DataSource = NaborDannix;
19. dataGridView1->DataMember = "talaba";
20. Ulanish->Close();                }
```

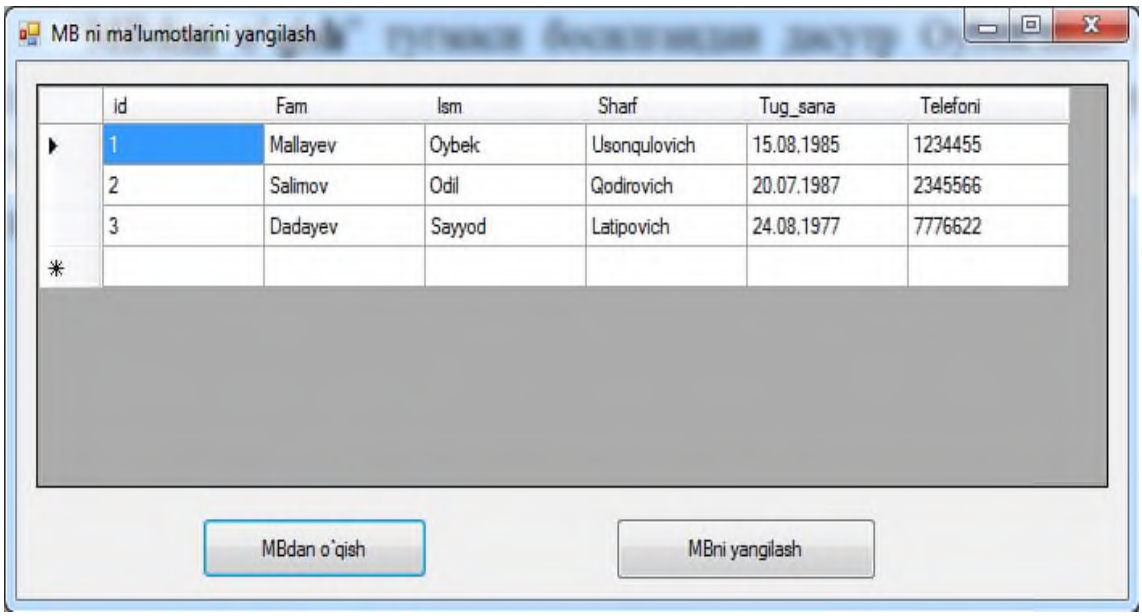
```
21. private: System::Void button2_Click(System::Object^  sender,
    System::EventArgs^ e) {// MB ni yangilash
22. Komanda->CommandText = "UPDATE [talaba] SET [Fam] = ?, Ism = ?,
    Sharf = ?, "+ "Tug_sana = ?, Telefoni = ? WHERE ([id] = ?)";
23. // Ustun nomlari, tiplari, o‘lchami, ustun nomlari
24. Komanda->Parameters->Add("Fam", OleDb::OleDbType::VarWChar, 50,
    "Fam");
25. Komanda->Parameters->Add( "Ism", OleDb::OleDbType::VarWChar, 50,
   "Ism");
26. Komanda->Parameters->Add( "Sharf", OleDb::OleDbType::VarWChar,
   50, "Sharf");
27. Komanda->Parameters->Add( "Tug_sana", OleDb::OleDbType::Date ,
   50, "Tug_sana");
28. Komanda->Parameters->Add( "Telefoni", OleDb::OleDbType::VarChar
   , 50, "Telefoni"); Komanda->Parameters->Add(gcnew
29. OleDb::OleDbParameter("Original_id",
30. OleDb::OleDbType::Integer, 0, System::Data::ParameterDirection::
31. Input, false, (Byte)0, (Byte)0, "id",
   System::Data::DataRowVersion::
32. Original, nullptr));
33. Adapter->UpdateCommand = Komanda;   Komanda->Connection =
   Ulanish;
34. try { // Update o‘zgargan qatorlar sonini qaytaradi:
35. int kol = Adapter->Update(NaborDanno`x, "talaba");
36. MessageBox::Show("Obnovleno " + kol + " zapisey"); }
37. catch (Exception ^ Situasiya)
38. { MessageBox::Show(Situasiya->Message, "Nedorazumenie"); }
39. }
```

Dasturning tashqi ko‘rinishi quyidagicha:



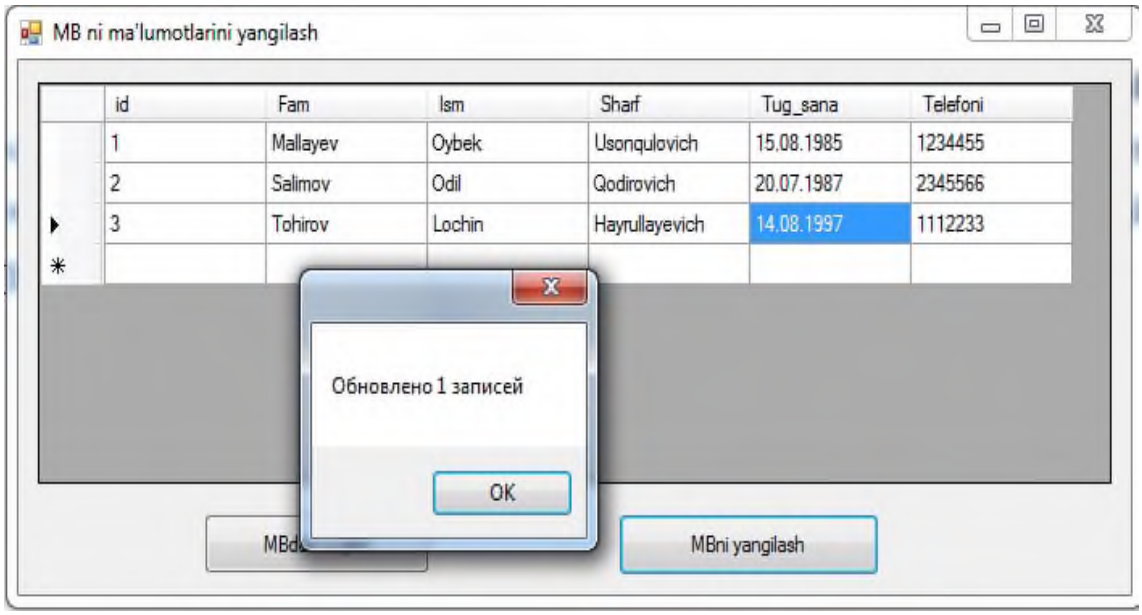
4.31-rasm. Dasturning boshlang‘ich xolati

“**MBdan o’qish**” tugmasi bosilgandan dastur “**Oybek.mdb”** nomli MB bilan bog‘lanadi va uning “**talaba**” nomli jadvalidagi barcha ma`lumotlarni ekran “**dataGridView** komponentasi”ga chiqaradi va quyidagi ko‘rinishga keladi:



4.32-rasm. MB ga bog‘lanib, yozuvlarini chiqargan xolati

Jadvaldagi ixtiyoriy qatorni o‘zgartirib, “**MBni yangilash**” tugmasi bosilsa dastur “**Oybek.mdb”** nomli MB bilan bog‘lanadi, uning “**talaba**” nomli jadvalidagi o‘zgartirilgan qator ma`lumotlarini yangilaydi va quyidagi ko‘rinishga keladi:



4.33-rasm. Jadvaldagi yozuvlarni yangilagan holati

Dastur tahlili:

MB dan ma`lumotlarni o‘qish va yangilash dasturini tuzishda birinchi navbatda **SQL** so‘rovi (**Select * From[talaba]**)ni tanlandi. Hamda **Adapter** sinf ob`ekti yordamidan foydalanildi. Keyin “**dataGridView1”** komponentasining “**DataSource”** hossasiga olingan barcha yozuvlar ko‘rsatildi. Natijada MBdan yozuvlarni o‘qish va ularni o‘zgartiruvchi vizual dastur yaratildi.

7-dastur: MS Access ning MB dagi jadval yozuvlarini SQL so‘rovlari va “Command” sinf ob`ekti yordamida o‘chiruvchi vizual dastur

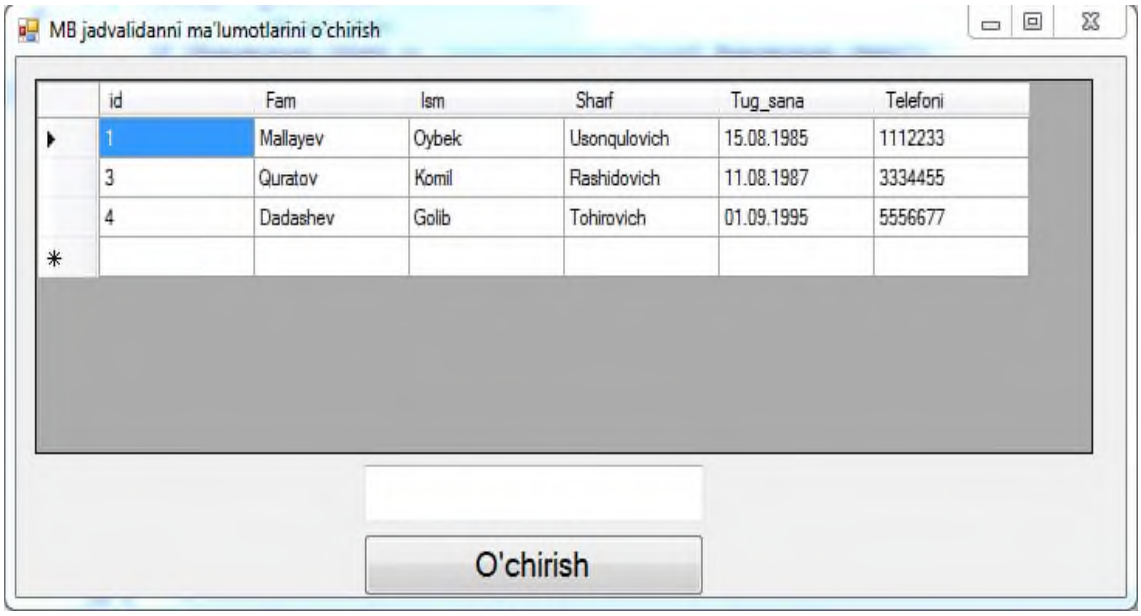
SQL so‘rovlaridan maqsadli foydalanib, MB ning jadvalidagi yozuvlarini o‘chirish mumkin. Buning uchun “**Command”** sinf ob`ekti kerak bo‘ladi. Ushbu dasturni tuzish uchun “**Visual Studio”** dasturida “**Windows Form Application”** muhitida yangi loyiha yaratiladi. Formaga bitta “**Button”, “TextBox”** va “**DataGridView”** komponentalari joylashtiriladi.

Dasturning kodlari quyidagicha:

```
1. #pragma endregion
2. DataSet ^ NaborDannix;
3. OleDb::OleDbDataAdapter ^ Adapter;
4. OleDb::OleDbConnection ^ Ulanish;
5. OleDb::OleDbCommand ^ Komanda;
6. void MBdan_uchirish(){
7. Ulanish = gcnew OleDb::
8. OleDbConnection( "Data Source=\"C:\\Oybek.mdb\";User "+
   "ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";");
9.  if (Ulanish->State ==  ConnectionState::Closed) Ulanish->Open();
10. Komanda = gcnew Data::OleDb::OleDbCommand(
    "Delete Fam,Ism,Sharf,Tug_sana,Telefoni From [talaba] Where " +
    "Fam = '"+textBox1->Text+"'", Ulanish);
11. int i = Komanda->ExecuteNonQuery();
12. // i - kolichestvo udalennix zapisey
13. if (i > 0){ MessageBox::Show("Familiya ustunidagi bo`yicha
    topilgan yozuvlar '"+textBox1->Text+"*', o`chirildi");
14. MBdan_uqish();  }
15. if (i == 0) MessageBox::Show("Familiya ustunidagi bo`yicha
    topilgan yozuvlar '"+textBox1->Text+"*', topilmadi");  }
16. void MBdan_uqish(){
```

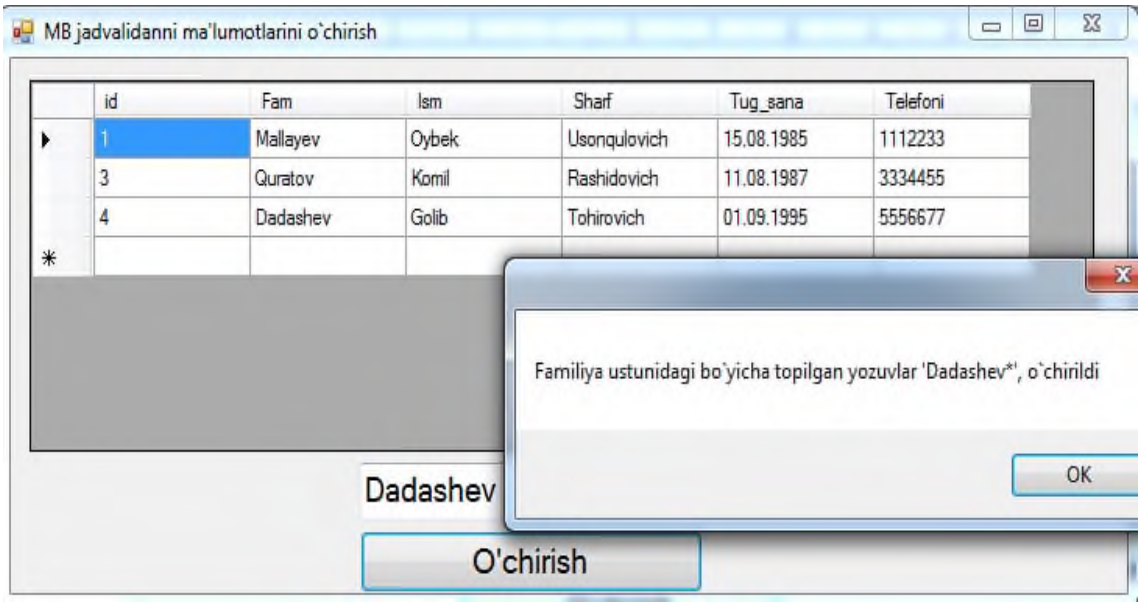
```
17. NaborDannix = gcnew DataSet();
18. Ulanish = gcnew OleDb:: OleDbConnection( "Data
    Source=\"C:\\Oybek.mdb\";User " +
19. "ID=Admin;Provider=\"Microsoft.Jet.OLEDB.4.0\";");
20. Komanda = gcnew OleDb::OleDbCommand();
21. // MB dan o‘qish:
22. if (Ulanish->State ==  ConnectionState::Closed) Ulanish->Open();
23. Adapter = gcnew OleDb::OleDbDataAdapter("Select * From
   [talaba]", Ulanish);
24. // DataSet ni SQL so‘rovlari natijasi bilan to‘ldirish
25. Adapter->Fill(NaborDanno`x, "talaba");
26. String ^ StrokaXML = NaborDanno`x->GetXml();
27. dataGridView1->DataSource = NaborDanno`x;
28. dataGridView1->DataMember = "talaba"; }
29. private: System::Void Form1_Load(System::Object^  sender,
    System::EventArgs^  e) {
30.  this->Text="MB jadvalidanni
    ma'lumotlarini o`chirish"; MBdan_uqish();  }
31. private: System::Void button1_Click(System::Object^  sender,
    System::EventArgs^  e) { MBdan_uchirish(); }
```

Dastur natijasi:



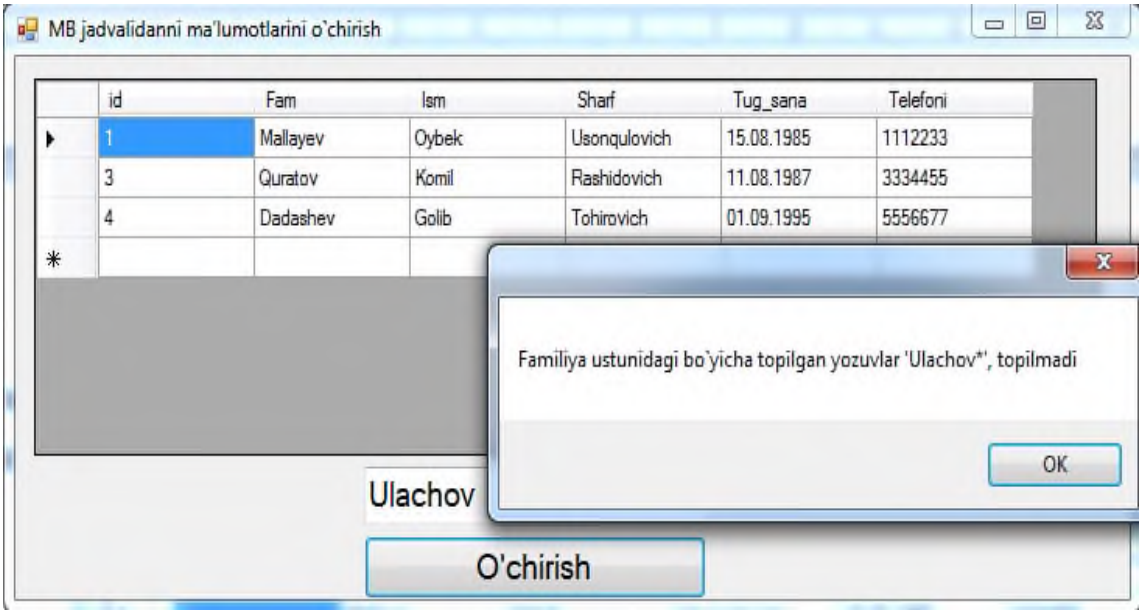
4.34-rasm. Dasturning boshlang‘ich xolati.

“**O’chirish**” tugmasi tepasidagi bo‘sh maydonchaga “Fam” ustunida mavjud yozuvlardan biri kiritiladi. “**O’chirish**” tugmasi bosilganda Agar kiritilgan yozuv “**Fam”** ustunining qaysidir qatorida mavjud bo‘lsa, o‘sha qator o‘chiriladi.



4.35-rasm. Talaba jadvalidan topilgan yozuvni o‘chirish holati.

Aks holda, “**Bunday yozuv topilmadi**” degan xabarnoma chiqadi. Xabarnomadagi **OK** tugmasi bosilsa, dastur yana Oybek.mdb nomli MB bilan bog‘lanib, uning “**talaba**” nomli jadvalidagi barcha ma`lumotlarni ekran (**dataGridView** komponentasi)ga chiqaradi va quyida ko‘rinishga keladi:



4.36-rasm. MB dan yozuvlar topilmagan holat.

Dastur tahlili:

Dasturda 2 ta funksiya yaratildi: 1- **void MBdan_uqish()** deb nomlanadi. Uning vazifasi **SQL** so‘rovi (**Select * From[talaba]**)ni tanlab, **Adapter** sinf ob`ekti yordamida “**Oybek.mdb**” nomli MB ga ulanishni ta`minlashdan iborat. Bunda ulanishni global o‘zgaruvchilar orqali tashkil qilindi. CHunki ikkita ta funksiya uchun alohida sinf ob`ektlaridan e`lon qilinmasligi kerak edi. Bu esa dastur hajmini kamaytirib, samaradorligini oshiradi.

2- **void MBdan_uchirish()** deb nomlanadi. Uning vazifasi nomidan kelib chiqib, “**Oybek.mdb**” nomli MB ning “**talaba**” jadvaldagi yozuvlarni, foydalanuvchi qidirayotgan yozuv asosida o‘chirishdan iborat. YOzuvlarni o‘chirishda quyidagi texnologiyadan foydalanildi:

```
Komanda = gcnew Data::OleDb::OleDbCommand("Delete
Fam,Ism,Sharf,Tug_sana,Telefoni From [talaba] Where " + "Fam =
'"+textBox1->Text+"'", Ulanish);
```

Mustahkamlash uchun mashqlar

a) Quyidagi topshiriqlar asosida 4, 5, 6 va 7- dasturlardan na`muna sifatida foydalnib, “**Windows Application”** muhitida ishlovchi vizual dastur yarating;

b) Dasturda **SQL** so‘rovlari(**select, insert, update**)ni ishlatgan holda MB ni jadvallarini ma`lumotlar bilan to‘ldirish, yangilash va o‘chirishning har hil usullaridan foydalanish imkoniyatlari mavjud bo‘lsin;

s) MB dagi har bitta jadval, dasturda alohida formada joylashishi lozim va ularning tashqi ko‘rinishlariga va ishlash qulayliligiga katta ahamiyat qarating.

1. Mahalla ma`lumotlar bazasi;
2. Maktab o‘quvchilarini ma`lumotlar bazasi;
3. SHifoxonaning qabul bo‘limini ma`lumotlar bazasi;
4. Fakultetning ma`lumotlar bazasi;

Nazorat savollari:

1. Ma`lumotlar bazasiga ta`rif bering?

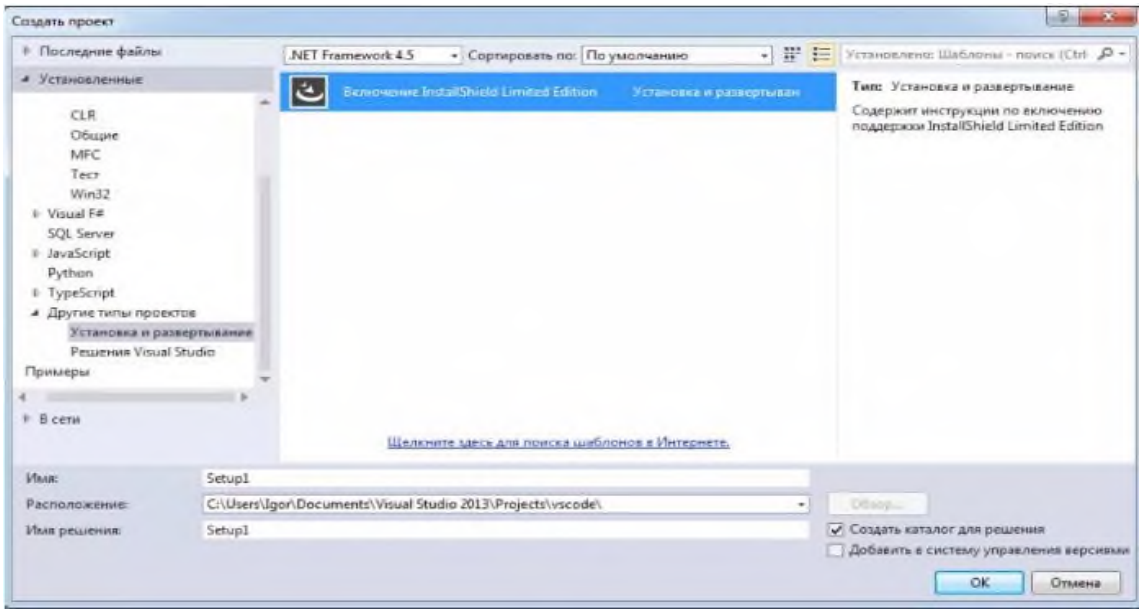
2. Ko‘pchilik foydalanuvchilar tomonidan MBni yaratish, to‘ldirish va birgalikda foydalanish uchun mo‘ljallangan dasturiy vositalar majmuasi nima deyiladi?
3. Fayllarni boshqarishda nima yordam beradi?
4. MB adminstratori.
5. Zamonaviy MBBTlar fayl tizimining qaysi muammosini hal qiladi?
6. MBBT dagi foydalanuvchilarga yaqin abstraksiya bosqichini ko‘rsating.
7. MBBT arxitekturasining bosqichi bo‘lmagan javob variantini ko‘rsating.
8. MBBT dagi MBning barcha mantiqiy strukturasini ko‘rsatuvchi abstraksiya bosqichini ko‘rsating.
9. MBBTning vazifasiga nimalar kirmaydi?
10. Ma`lumotlar bazasi tizimi nimalarni o‘z ichiga oladi?
11. MBBT dagi axborotni jismonan saqlashga yaqin abstraksiya bosqichini ko‘rsating.
12. Ma`lum bir sust strukturaga ega va qiyin shakllanuvchi fan sohasining sun`iy intellekt tizimi nima deb ataladi?

4.5. Loyihalarning yuklanuvchi “инсталляционные” dasturlar paketini yaratish

Bizga ma`lumki, avvalgi bo‘limlarda yaratilgan kichik loyihalarni boshqa kompyuterlarda sinash yoki amaliyotga tadbiq qilish uchun ularning yuklanuvchi paketlarini yaratish kerak. Yuklanuvchi paketlar - bu loyihaning dastur kodlari yaratuvchisida saqlangan holda uning “**exe**” formati (tizimga yuklanuvchi)dir. Yuklanuvchi paket o‘zida bir nechta (**setup.exe, *.msi, *.dll, *.tlb**)kengaytmali fayllarni saqlaydi. Kichik loyihalarning operasion tizimga yuklanuvchi paketlarini yaratish uchun dasturlarni instalyasiyalovchi maxsus dasturlar bor. Masalan **InstallShild, InnoSetup** va boshqalar. Lekin bunday dasturlarni internet orqali yuklab olishni va aktivlashtirishni talab qiladi.

Yuklanuvchi paketlarni yaratish uchun **Visual Studio** dasturi ishga tushiriladi. Yangi loyiha yaratiladi(**New Project**), **Other Project Types** bo‘limidan **Setup and Deployment (Установка и распространение)** qismiga o‘tiladi va **Visual Studio Installer va Setup Project** tanlanadi.

Location qismiga yuklanuvchi paket saqlanishi kerak bo‘lgan diskdan joy ko‘rsatiladi va OK tugmachasi bosiladi. Masalan: **S:\New_instal.** Misol uchun, bir loyihani yuklanuvchi paketini yaratib ko‘raylik. **“Visual Studio”** dasturi ishga tushiriladi. Yangi loyiha yaratiladi “**New Project**”, “**Other Project Types”** bo‘limidan “**Setup and Deployment” (“Установка и распространение”)** qismiga o‘tiladi:



4.37-rasm. Yuklanuvchi paketlarni yaratishning 1- bosqichi

1-qadam. Veb brouzerda quyidagi sahifa ochiladi. Bu sahifadan “**Шаг 2”** havolasi bosiladi.

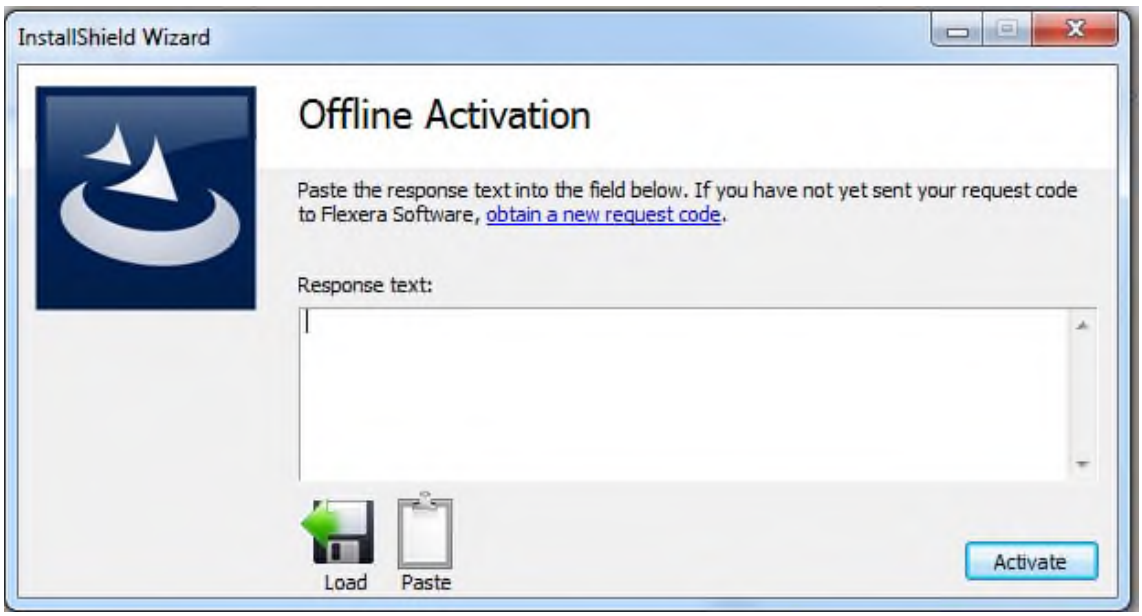
4.38-rasm. Yuklanuvchi paketlarni yaratishning 2- bosqichi

2-qadam.
Undan keyin anketa to‘ldiriladi.

4.39-rasm. Yuklanuvchi paketlarni yaratishning 3- bosqichi

3-qadam.

Anketa to‘ldirilgandan keyin “**Download Now**” tugmasini bosib, dasturni yuklab olasiz va tizimga yuklaysiz. Yaratilgan tayyor loyiha dasturini ochasiz. Dastur ochilishida yangi yuklangan dastur lisenziya kalitini talab qiladi:



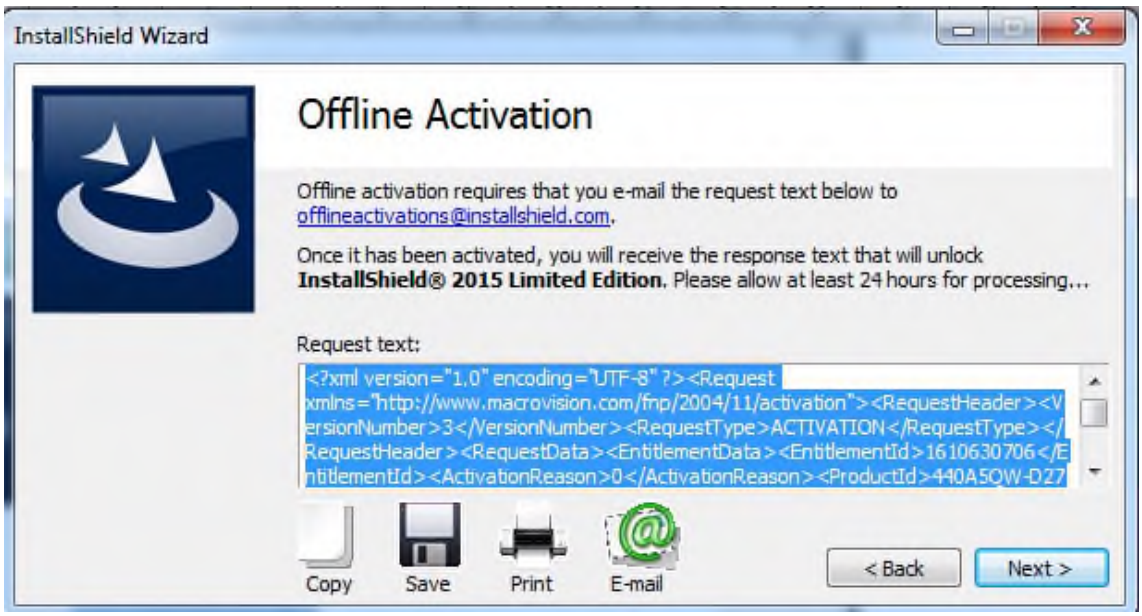
4.40-rasm. Yuklanuvchi paketlarni yaratishning 4- bosqichi

4-qadam.

4.41-rasm. Seriya raqamini kiritish oynasi

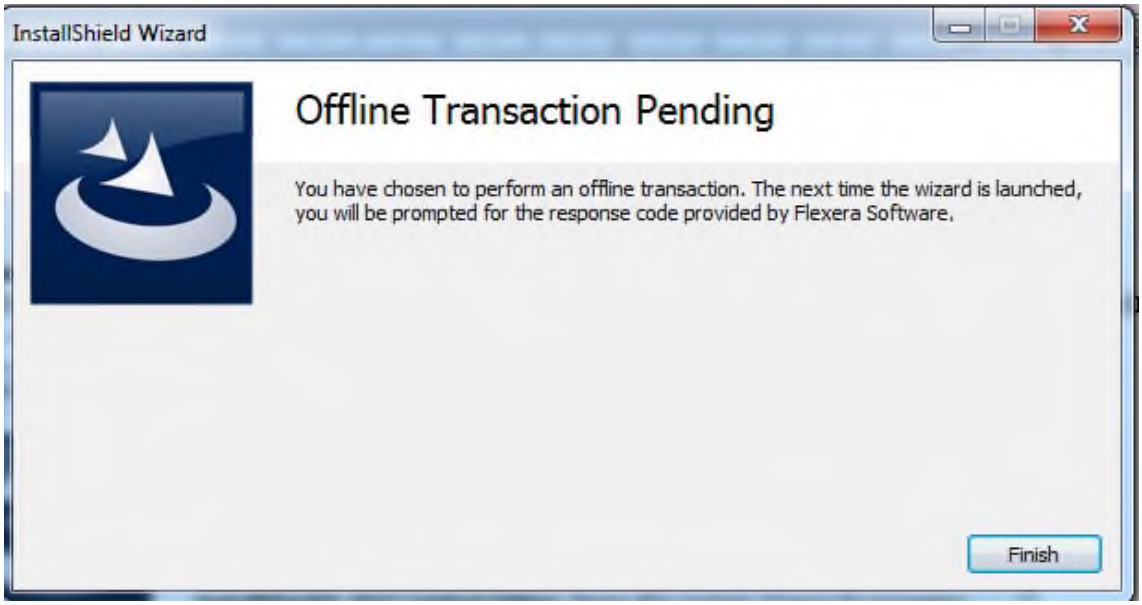
5-qadam.

Lisenziya kalitini anketangizda ko‘rsatilgan pochta manzilidan olinadi, so‘ralgan joyga yoziladi va “**Activate**” tugmachasi bosiladi.



4.42-rasm. Dasturni offline aktivlashtirish oynasi

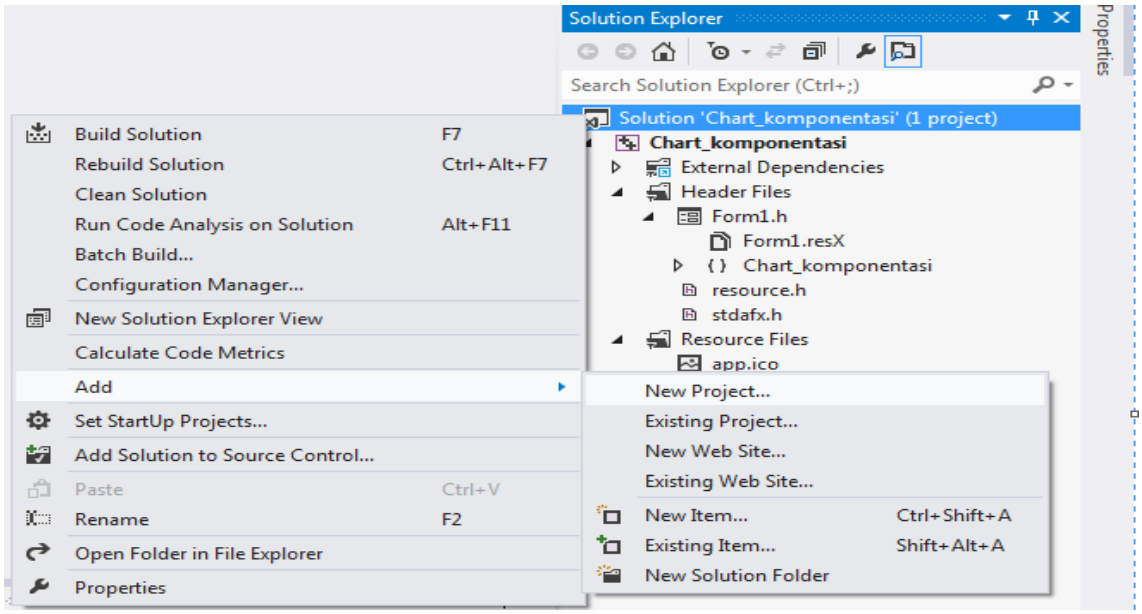
6- qadam.



4.43-rasm. Offline tranzaksiyani kutish oynasi

7-qadam.

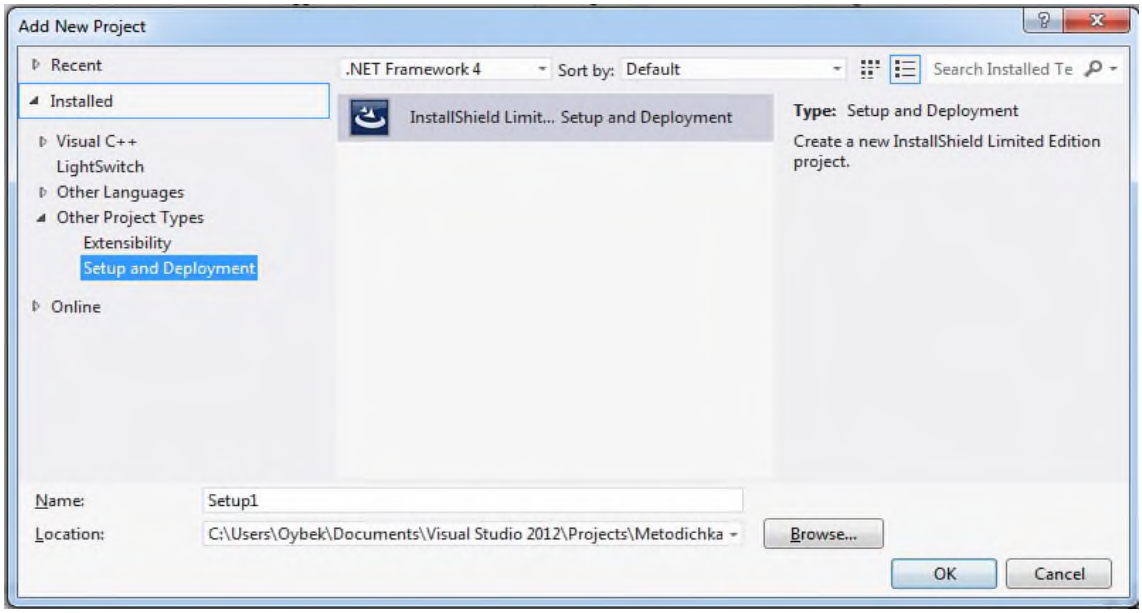
Finish tugmasi bosilgandan keyin. Yuklanuvchi paketini yaratish kerak bo‘lgan dasturga yangi loyiha qo‘shiladi: (**Solution Explorer->** loyiha tanlanadi va sichqonchani o‘ng tugmasini bosib, kontekst menyudan->**Add->New Project..)**



4.44-rasm. Yuklanuvchi dasturga qo‘shiladigan yangi loyihani tanlash oynasi

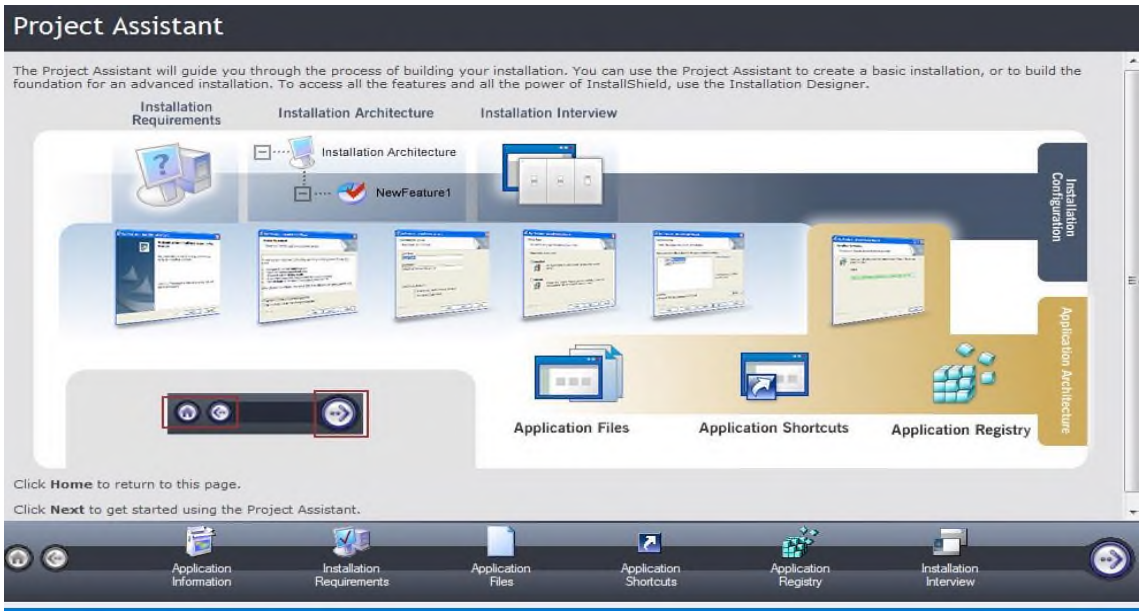
8-qadam.

Yangi loyiha qo‘shish oynasida **Other Project Types** bo‘limidan **Setup and Deployment** qismiga o‘tiladi va OK tugmasi bosiladi:



4.45-rasm. Yangi loyiha qo‘shish oynasi

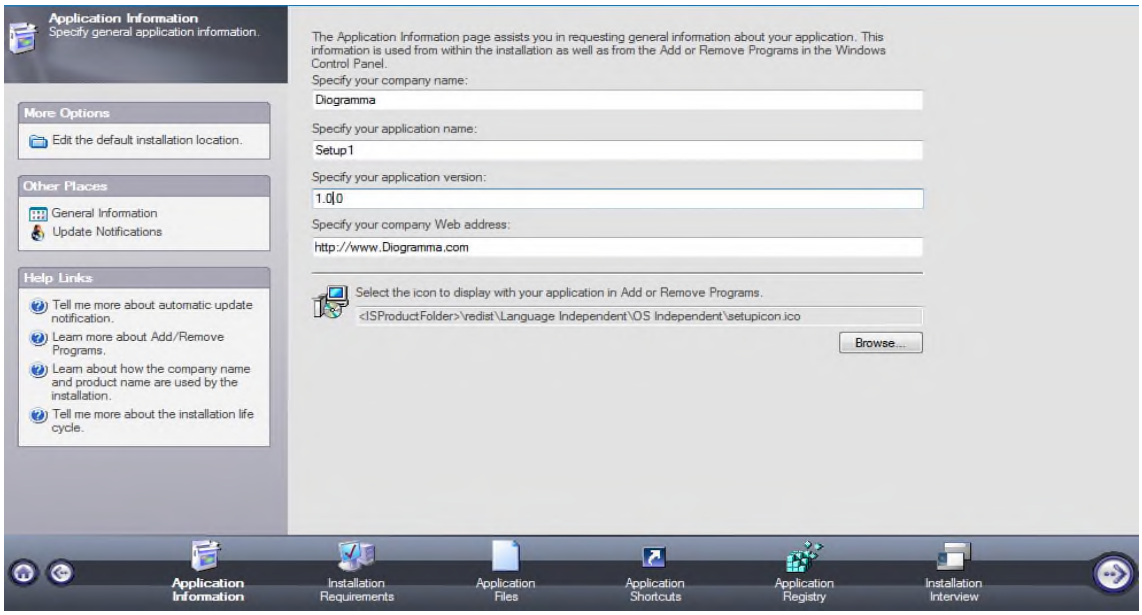
9-qadam. Ushbu jarayondan keyin quyidagi jarayon bajariladi:



4.46-rasm. Jarayon yordamchisi oynasi

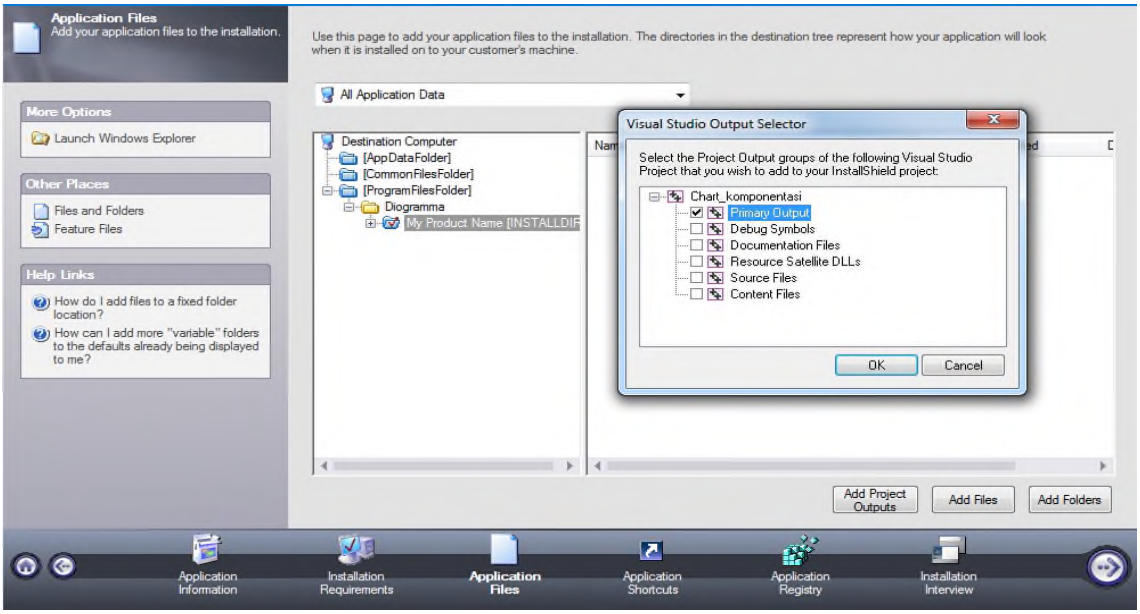
10-qadam.

“Далее” tugmasi bosiladi.



4.47-rasm. Loyiha to‘g‘risida ma`lumot oynasi

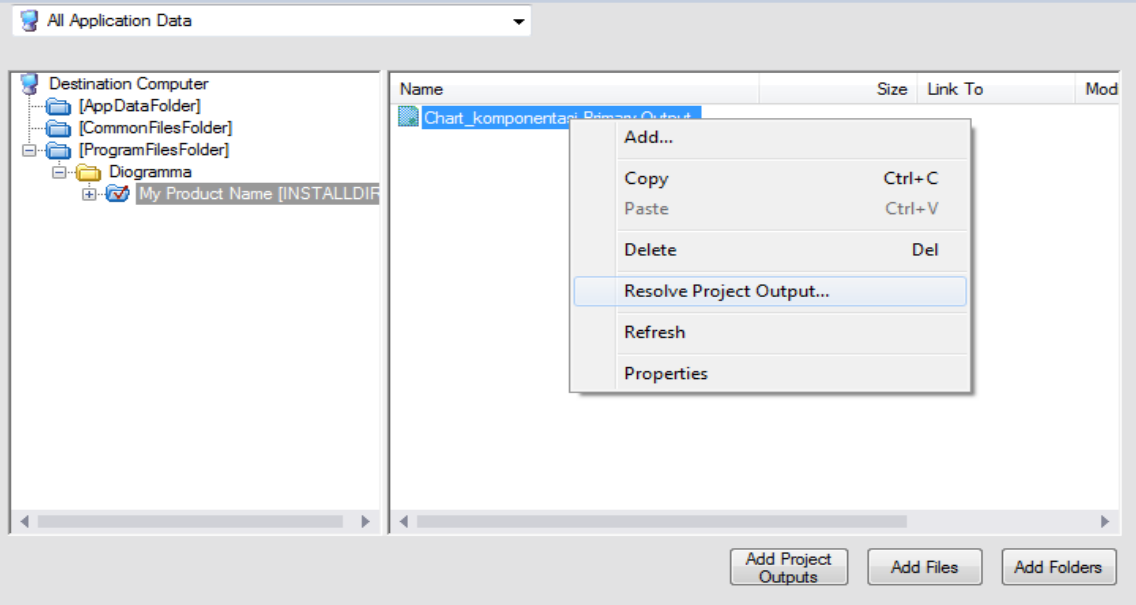
11-qadam. “**Далее**” tugmasi bosiladi.



4.48-rasm. Loyihaning fayllarini qo‘shish oynasi

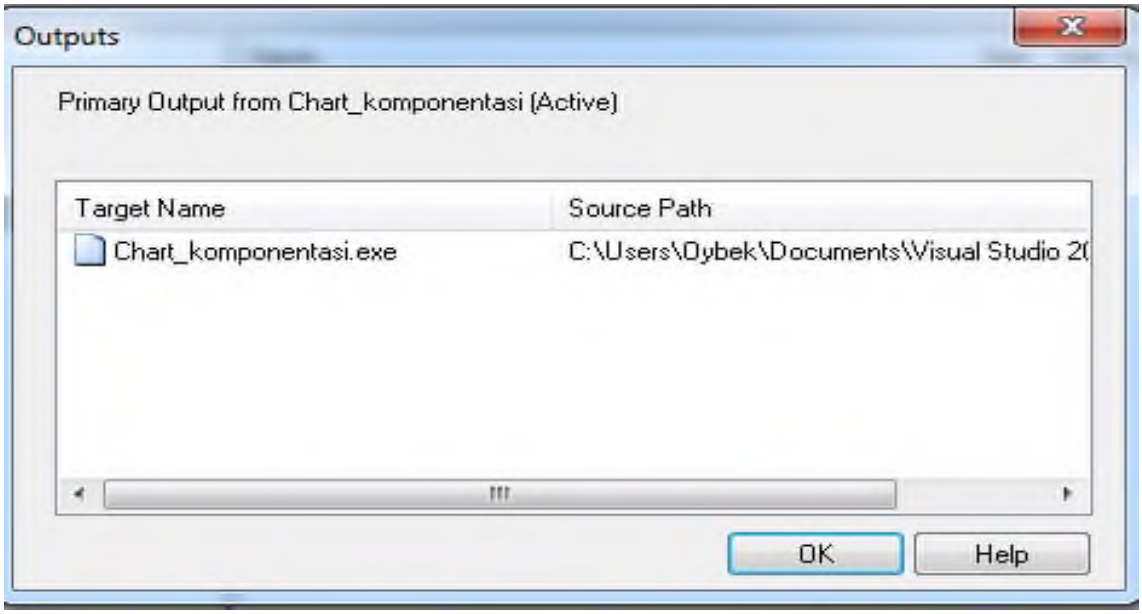
12-qadam.

“**OK**” va “**Далее**” tugmasi bosiladi.



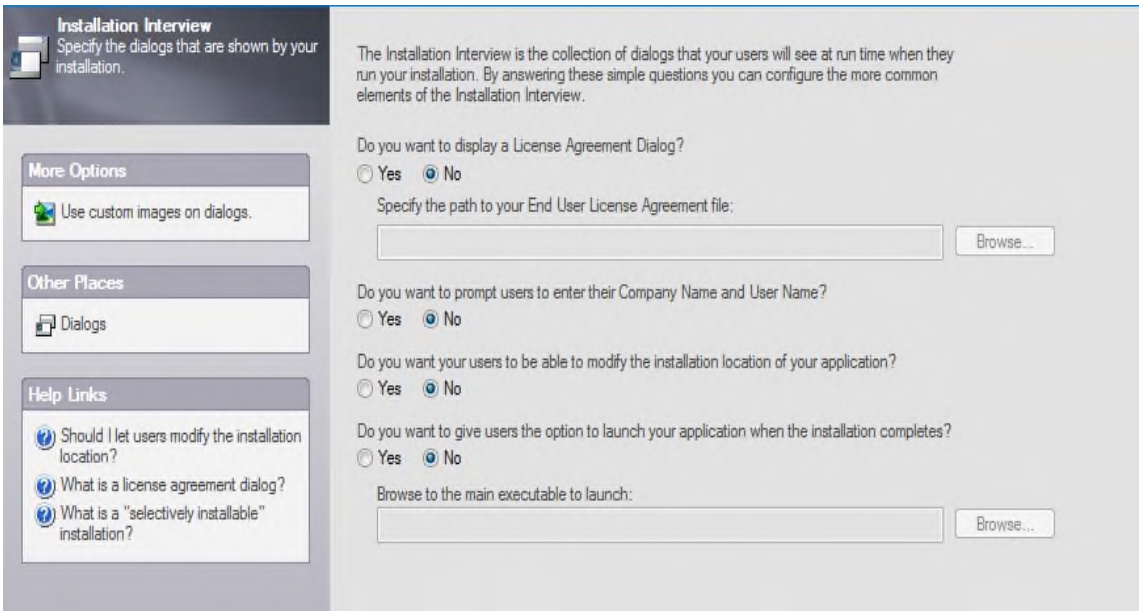
4.49-rasm. Yuklanuvchi paketlarni yaratishning 12- bosqichi

13-qadam. “**OK**” va “**Далее**” tugmasi bosiladi.



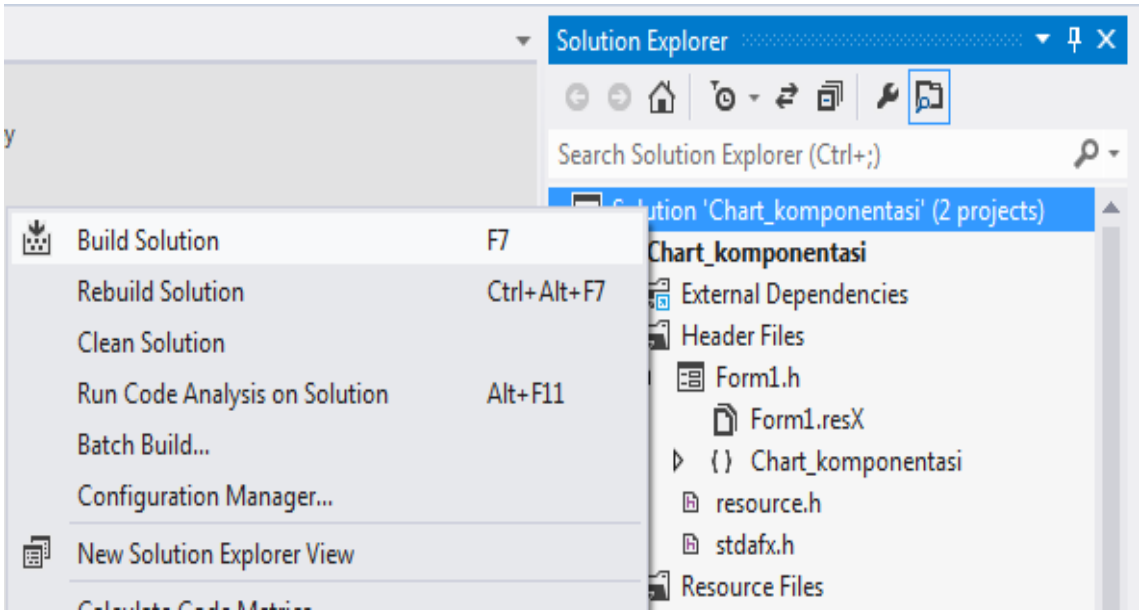
4.50-rasm. Yuklanuvchi paketlarni yaratishning 13- bosqichi

14-qadam. “**OK**” va “**Далее**” tugmasi bosiladi.



4.51-rasm. Yuklanuvchi paketlarni yaratishning 14- bosqichi

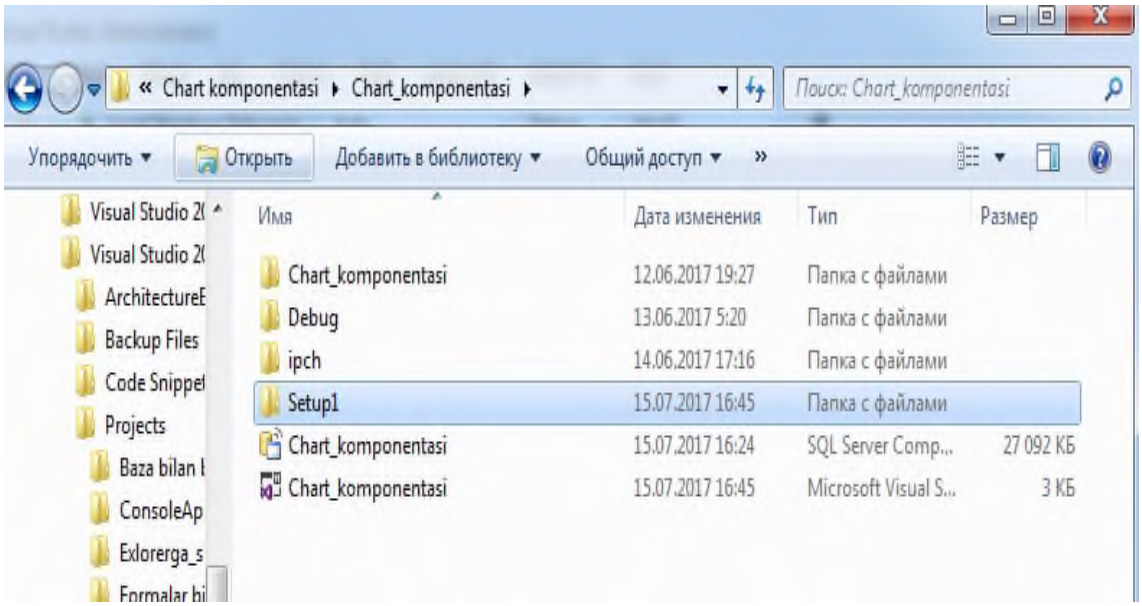
15-qadam. Kontekst menyudan “**Build Solution”** tanlanadi.



4.52-rasm. Yuklanuvchi paketlarni yaratishning 15- bosqichi

16-qadam.

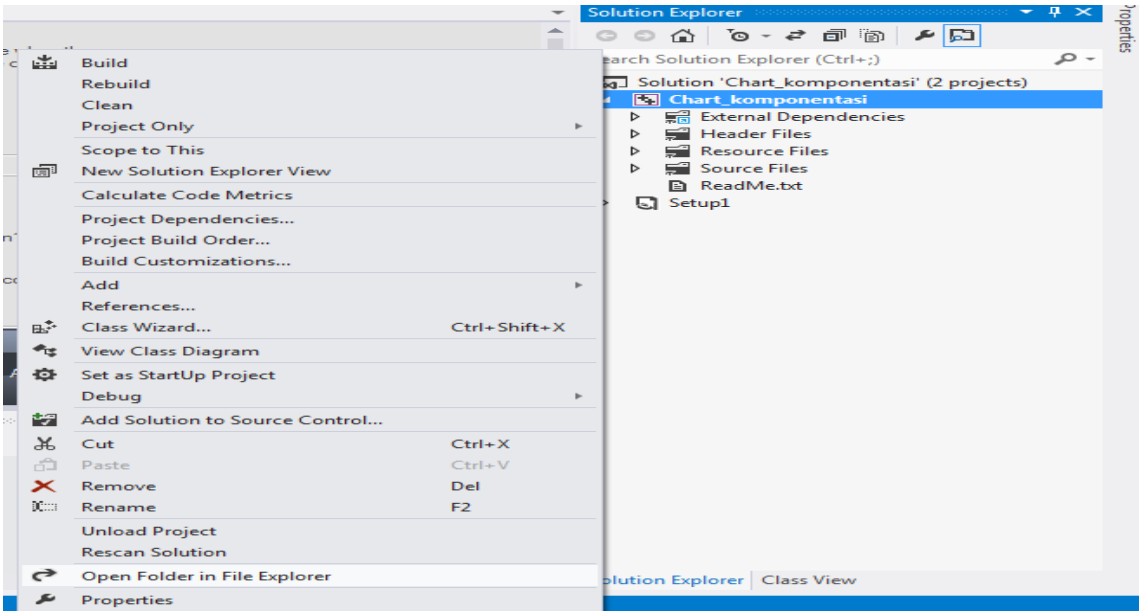
Bu jarayondan so‘ng yuklanuvchi paket saqlangan katalog ichida **setup1** nomli katalog paydo bo‘ladi.



4.53-rasm. Yuklanuvchi paket saqlangan joy

17-qadam.

Ushbu katalogga quyidagicha o‘tsa ham bo‘ladi:



4.54-rasm. Yuklanuvchi paket saqlangan joyga o‘tish

18-qadam.

Setup1\Express\DVD-5\DiskImages\DISK1 ushbu manzilda yuklanuvchi fayl joylashgan. Natijada ushbu loyihaning yuklanuvchi paketini tarqatish imkoniyati vujudga keldi.

Bob xulosasi

Ushbu bobda **MS Access 2010** dasturida talabalarni baholash reytinglari haqidagi ma`lumotlar bazasini loyihalashtirish uchun jadvallar, so‘rovlar, formalar, hisobotlar va makroslar yaratish usullari keltirilgan. **SQL(structured query language)** so‘rovlarini ishlatilish qoidalari yuzasidan ma`lumotlar keltirildi hamda ushbu ma`lumotlar asosida amaliy mashg‘ulotlar tashkillashtirildi va amaliy dasturlar tuzildi.

Nazariy savollar va amaliy topshiriqlar

1. MS Access dasturida kitoblarning avtomatlashgan reytingini formallashgan jadvallarini shlab chiqish.
2. MS Access dasturida kitoblarning avtomatlashgan reytingini ma`lumotlar bazasini ishlab chiqish. MB da jadvallar, jadvallarning bog‘lanishini, so‘rovlarni va makroslarni tashkil etish.
3. SQL dasturida SQL so‘rovlari yordamida jadvallarni, jadvallar asosida ma`lumotlarni izlash usullarini ishlab chiqish.
4. MS Access dasturida SQL so‘rovlari yordamida kitoblarni qidirish usullari ishlab chiqish.
5. Visual Studio/Embarcadero dasturida komponentalar, formalar, kutubxonalarni tashkil etish. Boshqa dasturlash tillarida tuzilgan funksiyalardan foydalanish.
6. Visual Studio/Embarcadero dasturida yordamchi maxsus dasturlar yordamida loyiha interfeysini ishlab chiqish.
7. Dasturiy loyiha uchun ishlab chilgan algoritm funksiyalarni ob`ektga yo‘naltirilgan dasturlash shartlari asosida tashkil etish.
8. Ma`lumotlarning universal strukturasi va classlarini tashkil etish. Ular asosida DLL kutubxona fayllarini tashkil etish.
9. **Visual C++** da qanday dastur yuklanuvchi dastur paketini yaratadi?
10. **Visual C++** da yuklanuvchi dastur paketini yaratishning qanday turlarini bilasiz?

11. **Visual C++** da yaratilgan loyihani boshqa kompyuterlarda ishlatish usullarini bilasizmi?
12. **.net framework** nima?
13. **Visual C++** da **.net framework** nimaga kerak?
14. Mavjud loyihalarni birlashtirish qanday amalga oshiriladi?
15. Loyihaga serverni ulash qanday amalga oshiriladi?
16. Loyihaga MB ni ulash qanday amalga oshiriladi?
17. Tizimga yuklangan yangi loyiha dasturini sozlash mumkinmi?
18. **Visual C++** da loyiha dasturining xavfsizligi qanday taminlanadi?

ADABIYOTLAR

1. O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha harakatlar strategiyasi to‘g‘risida. O‘zbekiston Respublikasi Prezidentining PF-4947- son farmoni . Toshkent, 2017 yil 7 fevral.
2. Mirziyoev SH.M. Tanqidiy tahlil, qat'iy tartib-intizom va shaxsiy javobgarlik - har bir rahbar faoliyatining kundalik qoidasi bo‘lishi kerak. Mamlakatimizni 2016 yilda ijtimoiy-iqtisodiy rivojlantirishning asosiy yakunlari va 2017 yilga mo‘ljallangan iqtisodiy dasturning eng muhim ustuvor yo‘nalishlariga bag’ishlangan Vazirlar Mahkamasining kengaytirilgan majlisidagi ma'ruza, 2017 yil 14 yanvar. - Toshkent: «O‘zbekiston», 2017.-104 b.
3. Mirziyoev SH.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. 2017.
4. Mirziyoev SH.M. Qonun ustuvorligi va inson manfaatlarini ta`minlash – yurt taraqqiyoti va xalq farovonligining garovi. 2017.
5. Nazirov SH.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. S va C++ tili. “Voris- nashriyot” MCHJ, Toshkent 2013, 488 b.
6. Maxarov Q.T., Maxarov T.A. Visual Studio muhitida dasturlash asoslari(uslubiy qo`llanma). Toshkent – 2017.
7. Maxarov Q.T., Nigmanova D.B., Navruzov E.R., Hayitqulov B.H. VC++ muhitida obyektlar bilan ishlash. Toshkent – 2017
8. Horstmann, Cay S. C++ for everyone/Cay S. Horstmann. Printed in the United States of America - 2nd ed. 2010. – P. 562.
9. Zubarov V.V. MS Visual C++ 2010 v srede .NET (2012) Piter.
10. Boris Paxomov. C/C++ i Visual C++ 2010 dlya nachinayuhix. - SBP: BXV-Peterburg. 2011.
11. J.Axmadaliev, R.Xoldorboev C++ dasturlash tilini o‘rganish bo‘yicha uslubiy qo‘llanma(2015).
12. Horton I.-Beginning Visual C++ 2012/ I.Horton. Published simultaneously in Canada.–2012. –P. 988.
13. Давыдов, В. Visual C++. Разработка Windows-приложений с помощью MFC и API-функций / В. Давыдов. - М.: БХВ-Петербург, 2014. - 576 с.
14. Довбуш, Галина Visual C++ на примерах / Галина Довбуш , Анатолий Хомоненко. - М.: БХВ-Петербург, 2012. - 528 с.

15. Зиборов, В. MS Visual C++ 2010 в среде .NET / В. Зиборов. - М.: Питер, 2012. - 320 с.
16. Мешков, А. Visual C++ и MFC / А. Мешков, Ю. Тихомиров. - М.: БХВ-Петербург, 2013. - 546 с.
17. Панюкова, Т. А. Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++. Учебное пособие / Т.А. Панюкова, А.В. Панюков. - Москва: Мир, 2015. - 144 с.
18. Пахомов, Б. C/C++ и MS Visual C++ 2010 для начинающих / Б. Пахомов. - М.: БХВ-Петербург, 2011. - 736 с.
19. Пахомов, Борис C/C++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - Москва: СИНТЕГ, 2015. - 518 с.
20. Пахомов, Борис C/C++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - М.: "БХВ-Петербург", 2013. - 502 с.
21. Понамарев, В. Программирование на C++/C# в Visual Studio .NET 2003 / В. Понамарев. - М.: БХВ-Петербург, 2015. - 917 с.
22. Сидорина, Татьяна Самоучитель Microsoft Visual Studio C++ и MFC / Татьяна Сидорина. - М.: "БХВ-Петербург", 2014. - 848 с.
23. http://labs.org.ru/visual-c/
24. https://msdn.microsoft.com/ru-ru/library/60k1461a.aspx
25. https://professorweb.ru/my/programs/visual-studio/level1/

MUNDARIJA

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

VISUAL C++ DA KICHIK LOYIHALAR YARATISH

(O ‘quv qo‘llanma).

Toshkent – «Aloqachi» – 2019

Muharrir: Q.Matqurbonov
Tex. muharrir: A.Tog‘ayev
Musavvir: B.Esanov
Musahhiha: F.Tog‘ayeva
Kompyuterda sahifalovchi: Sh.To‘xtamurodov

Nashr.lits. AI №176. 11.06.11.
Bosishga ruxsat etildi: 28.06.2019. Bichimi 60x841 /16.
Shartli bosma tabog‘i 14,5. Nashr bosma tabog‘i 14,0.
Adadi 100. Buyurtma №

«Nihol print» Ok da chop etildi.
Toshkent sh., M. Ashrafiy ko‘chasi, 99/101.