

L.N.XUDOYOROV, SH.R.DAVRONOV, B.N.NOSIROV,  
N.F.AXMEDOVA, SH.M.SAMANDAROVA

# DASTURLASH

DARSLIK

**O'ZBEKISTON RESPUBLIKASI**  
**OLIY TA'LIM, FAN VA INNOVATSIYALAR VAZIRLIGI**  
**MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT**  
**TEXNOLOGIYALARI UNIVERSITETI QARSHI FILIALI**

**L.N. XUDOYOROV, SH.R. DAVRONOV, B.N. NOSIROV**  
**N.F. AXMEDOVA, SH.M. SAMANDAROVA**

## **DASTURLASH**

Darslik 60310500-Raqamli iqtisodiyot, 60611100-Telekommunikatsiya  
texnologiyalari (Telekommunikatsiyalar), 60611300-Axborot-kommunikatsiya  
texnologiyalari sohasida kasb-ta'lifi, 60611400-Pochta aloqasi texnologiyasi  
yo'nalish talabalari uchun mo'ljallangan

Qarshi  
“BIG MAKRO WORLD” nashriyoti,  
2025

## UO‘T 004.6

L.N. Xudoyorov, Sh.R. Davronov, B.N. Nosirov, N.F. Axmedova, Sh.M. Samandarova

Dasturlash. Darslik. – Qarshi. «BIG MAKRO WORLD» nashriyoti, 2025. – 280 bet.

Mazkur darslik 60310500 – Raqamli iqtisodiyot, 60611100– Telekommunikatsiya texnologiyalari, 60611300– Axborot–kommunikatsiya texnologiyalari sohasida kasb–ta’limi, 60611400–Pochta aloqasi texnologiyasi yo‘nalishlarida tahsil olayotgan talabalarga mo‘ljallangan bo‘lib, dasturlashning nazariy va amaliy jihatlarini qamrab oladi. Unda chiziqli, tarmoqlanish va takrorlanish algoritmlarining asoslari, statik va dinamik massivlar, satrlar, fayllar bilan ishlash, obyektga yo‘naltirilgan dasturlash tamoyillari keng yoritilgan. Shuningdek, GUI muhitida dasturlash, foydalanuvchi interfeysi yaratish, grafik imkoniyatlar va muloqot oynalari bilan ishlash bo‘yicha amaliy tavsiyalar berilgan. Darslik maktab, kollej va oliy ta’lim muassasalari o‘quvchilari va talabalari uchun dasturlash ko‘nikmalarini shakllantirishga, algoritmik fikrlashni rivojlantirishga xizmat qiladi.

Ushbu darslikni yozish jarayonida K.B.Ablaqulovning “Ma’lumotlar tuzilmasi va algoritmlari” o‘quv qo‘llanmasidan hamda sun’iy intellekt va ma’lumotlarni qayta ishlashga oid maqolalaridan foydalanildi.

### **Mualliflar:**

Xudoyorov Laziz Niyozovich – iqtisod fanlari falsafa doktori

Davronov Shohjaxon Rizamat o‘g‘li – texnika fanlari falsafa doktori

Nosirov Baxtiyor Nusratovich – katta o‘qituvchi

Axmedova Nilufar Farxodovna – o‘qituvchi–stajyor

Samandarova Shaxnoza Mirjamilovna – o‘qituvchi–stajyori

### **Taqrizchilar:**

Uzakov Zair – Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Qarshi filiali “Axborot texnologiyalarining dasturiy ta’minoti” kafedrasi professor v.b., fizika-matematika fanlari nomzodi.

Yakubov Sobir Xolmurodovich – O‘zbekiston Respublikasi Harbiy aviatsiya instituti “Umumtexnika fanlari” kafedrasi professori, texnika fanlari doktori.

ISBN 978-9910-578-09-0

©N.L. Xudoyorov, R.Sh. Davronov, N.B. Nosirov,

F.A. Axmedova, M.Sh. Samandarova 2025-y.

© “BIG MAKRO WORLD” nashriyoti, 2025-y.

## MUNDARIJA

	KIRISH .....	4
I BOB	CHIZIQLI, TARMOQLANISH VA TAKRORLANISH ALGORITMLARI	
1.1-§	Algoritmlar va dasturlashning asosiy tushunchalari.....	6
1.2-§	Dasturlash tillarining tuzilmasi.....	17
1.3-§	Tarmoqlanish va tanlash operatorlari.....	28
1.4-§	Takrorlanish operatorlari.....	35
1.5-§	Funksiyalar.....	45
II BOB	STATIK VA DINAMIK MASSIVLAR. SATRLAR VA KENGAYTIRILGAN BELGILAR	
2.1-§	Bir o'lchovli massivlar.....	53
2.2-§	Ko'p o'lchovli massivlar.....	60
2.3-§	Ko'rsatkichlar va dinamik xotira bilan ishlash.....	63
2.4-§	Dinamik massivlar.....	69
2.5-§	Satrlar va kengaytirilgan belgilari (char toifasida).....	74
2.6-§	Satrlar va kengaytirilgan belgilari (string toifasida).....	83
III BOB	FAYLLAR BILAN ISHLASH, FOYDALANUVCHI TOIFASINI YARATISH. OBYEKTGA YO'NALTIRILGAN DASTURLASH ASOSLARI	
3.1-§	Fayllar bilan ishlash.....	88
3.2-§	Foydalanuvchi toifasini yaratish.....	96
3.3-§	Obyektga yo'naltirilgan dasturlash asoslari.....	102
IV BOB	INKAPSULYATSIYA, MEROSXO'RLIK VA POLIMORFIZM. KONTEYNER SINFLAR	
4.1-§	Shablonlar bilan ishlash.....	118
4.2-§	Konteynerlar (kolleksiyalar).....	127
V BOB	C++ DASTURLASH TILIDA GUI MUHITI VA FOYDALANUVCHI INTERFEYSINI YARATISH	
5.1-§	GUI muhitida dasturlash.....	149
5.2-§	Komponentalar bilan ishlash. Tarmoqlansh va tanlash uchun mo'ljallangan komponentalar. Massivlar bilan ishlash komponentalari.....	171
5.3-§	GUI muhitida grafik imkoniyatlar. To'g'ri chiziq va turli xil geometrik figuralarni chizish komponentalari.....	215
5.4-§	GUI muhitida grafik imkoniyatlar. GUI muhitida grafik holat, tasvirlarni va funksiya grafiklarini qurish.....	226
5.5-§	Muloqot oynalari bilan ishlash. GUI muhitida muloqot oynalari va ularni sozlash, boshqarish elementlari.....	247
	FOYDALANILGAN ADABIYOTLAR .....	277

## KIRISH

Axborot kommunikatsion texnonologiyalarini taraqqiy etishida bevosita dasturlash tillarining o‘rni beqiyos. Ayniqsa, hozirgi davrga kelib C++, Java, Python, Kotlin va boshqa dasturlash tillar yordamida shaxsiy kompyuterlar uchun amaliy dasturiy to‘plamlardan tashqari SmartPhone va Planshetlar uchun operatsion tizim va ilovalar yaratilmoqda.

Axborot texnologiyalarning yana bir muhim jihatlaridan biri shundaki, bu fan jadal sur’atlarda o‘sib, yil sayin yangidan-yangi yo‘nalishlarga, mutaxassisliklarga tarmoqlanib ketmoqda: algoritmik, mantiqiy, obyektga yo‘naltirilgan, vizual, parallel dasturlash texnologiyalari, animatsiya, multimediya, web, ma’lumotlar bazasini boshqarish tizimlari, ko‘p prosessorli, neyron arxitekturali kompyuterlar, sun’iy intellekt va hokazo. Ko‘rinib turibdiki, dasturlash meta fan darajasiga ko‘tarilib, uni bitta o‘quv kursi chegarasida to‘liq o‘zlashtirishning imkonini bo‘lmay qoldi.

Raqamli texnologiyalar sohasi bo‘yicha chet tillarida darslik va qo‘llanmalar juda ko‘p chop etilmoqda. Oxirgi yillarda o‘zbek tilidagi darslik va qo‘llanmalar ham ko‘payib qoldi.

Ushbu taklif etilayotgan “Dasturlash” fanidan darslik qo‘llanmasi asosan C++ dasturlash tilini o‘rganmoqchi bo‘lgan, darslarda olgan bilimlarini mustahkamlamoqchi bo‘lgan talabalar uchun mo‘ljallangan. Shu sababli darslikda C++ tiliga bog‘liq boshlang‘ich ma’lumotlardan boshlab foydalanuvchining grafikli interfeysi (GUI) muhitida dasturlash, Visual dasturlash muhitida kichik loyihalarni dasturlash kabi ma’lumotlar yoritilgan. Bu darslikdan C++ dasturlash tilini o‘rganuvchilar, dastur tuzishni o‘rganayotganlar hamda “Dasturlash” va boshqa dasturlashga oid fanlardan olingan nazariy bilimlarni mustahkamlash uchun foydalanishlari hisobga olingan. Ushbu darslikga kiritilgan ma’lumotlar dasturlashning bazaviy kursidagi deyarli barcha

bo‘limlarni, ya’ni skalyar turlar, boshqaruv operatorlari, ma’lumotlarning murakkab turlari, obyektga yo‘naltirilgan dasturlashdan tortib, visual dasturlash, GUI muhitida grafik holat, tasvirlarni va funksiya grafiklarini qurish, visual dasturlashda loyihalar yaratish kabilarni o‘z ichiga oladi.

Darslik bo‘yicha takliflar, undagi aniqlangan kamchiliklar bo‘yicha fikr-mulohazalaringizni mualliflardan birining, masalan nosirovbn@gmail.com elektron pochtasiga yuborishingizni so‘raymiz. Sizning bu bildirgan fikr-mulohazalaringiz darslikning keyingi nashrlarida uni yanada mazmunliroq va kamchiliklardan xoli tarzda chop etishimizga albatta asqotadi.

## **I BOB. CHIZIQLI, TARMOQLANISH VA TAKRORLANISH ALGORITMLARI**

### **1.1-§. ALGORITMLAR VA DASTURLASHNING ASOSIY TUSHUNCHALARI**

**Reja:**

- 1. Algoritmlarning xossalari va ifodalash usullari.**
- 2. Dasturlashga kirish.**
- 3. Kompilyator turlari.**
- 4. Identifikator va ularning turlari.**

Inson amaliy ish jarayonida juda ko‘p masalalarni hal etishiga to‘g‘ri keladi. Masalalarning ba’zilari oson, ba’zilari murakkab hisob-kitob bilan bog‘liq bo‘ladi. Ba’zi masalalarni hal etishda biror amallar guruhi esa minglab marta bajarilishiga to‘g‘ri kelishi mumkin. Shuning uchun beminnat va o’ta tez ishlaydigan yordamchimiz bo‘lgan kompyuter bu ishimizda yordam bera oladimi, agar yordam bera olsa, u holda masalalarni kompyuterda hal etish qanday tashkil etiladi degan savol tabiiydir.

Masala yechishning turli usullari mavjud bo‘lib, ulardan biri masala shartiga mos tenglama tuzish usulidir. Bu usul matematika kursidan ma’lum. Unda masala shartiga ko‘ra tenglama tuziladi va uning turiga qarab ma’lum bir usul tanlanadi hamda yechiladi.

**1. Masalaning qo‘yilishini aniqlash va matematik modelini ishlab chiqish.** Masalani yechishdan oldin uning qo‘yilishi oydinlashtiriladi, ya’ni bunda uning maqsadi va yechilish shartlari aniqlanadi, boshlang‘ich ma’lumotlar va natijalarning tarkibi asoslanadi. Bu ma’lumotlar asosida u matematik formulalar ko‘rinishida ifoda qilinadi.

**2. Masalani yechishning sonli usulini tanlash.** Qo‘yilgan matematik masalalar uchun uning sonli yechish usulini tanlash kerak bo‘ladi. Sonli usullar turli- tuman bo‘lganligidan ularning eng samarali va qulayini tanlash kerak. Bu

masala bilan matematikaning sonli usullar bo‘limi shug‘ullanadi. Yechish usulini tanlashda masalaga qo‘yilgan barcha talablarni va uni konkret kompyuterlarda hal qilish imkoniyatlarini hisobga olish kerak.

**3. Masalani yechish algoritmini ishlab chiqish.** Masalani yechish uchun tanlangan sonli usulning algoritmi ishlab chiqiladi, ya’ni masalani yechish uchun bajariladigan arifmetik va mantiqiy amallar ketma-ketligi yoritiladi. Masalani yechish algoritmlari ko‘rgazmaliroq bo‘lishi uchun, ular ko‘p hollarda bloksxema ko‘rinishida ifodalanadi.

**4. Kompyuter uchun dastur tuzish.** Kompyuter uchun dastur masalaning umumiy yechimidir. U algoritmning mashina buyruqlari ketma- ketligi shaklidagi yozuvidir. Buning uchun dasturlash tillari (Besk, Fortran, Paskal, C va boshqalar) dan biri tanlanadi va unga mos dastur tuziladi. Tuzilgan dasturni sifatli bo‘lishi va uni mashina xotirasidan kam joyni egallashi muhim ahamiyatga ega.

**5. Dasturni kompyuter xotirasiga kiritish, rostlash va tekshirish.** Tuzilgan dastur kompyuter klaviaturasi orqali uning xotirasiga kiritiladi. Kiritilgan dasturni rostlash va tekshirish amalga oshiriladi, ya’ni yo‘l qo‘yilgan xatoliklar tuzatiladi.

**6. Hisoblash natijalarini qayta ishlash va tahlil qilish.** Bu bosqichda tuzilgan dastur bo‘yicha hisoblash bajariladi va hosil bo‘lgan natija kompyutering display ekraniga chiqariladi yoki chop etish qurilmasi orqali qog‘ozga chop etiladi. Natijalarni jadvallar, grafiklar yoki diagrammalar ko‘rinishida hosil qilish mumkin. Hosil bo‘lgan natija esa foydalanuvchi tomonidan tahlil qilinadi.

Algoritm tushunchasi IX asrda yashab ijod etgan buyuk bobokalonimiz Muhammad al-Xorazmiy nomi bilan uzviy bog‘liqligini tushuntirish lozim. Algoritm so‘zi al-Xorazmiyning arifmetikaga bag‘ishlangan asarining dastlabki betidagi “Dixit Algoritmi” (“dediki al-Xorazmiy” ning lotincha ifodasi) degan jumlalardan kelib chiqqan. Shundan so‘ng al-Xorazmiyning sanoq sistemasini

takomillashtirishga qo'shgan hissasi, uning asarlari algoritm tushunchasining kiritilishiga sabab bo'lganligi ta'kidlab o'tiladi.

**Algoritm** asosiy tushuncha sifatida qabul qilingan, uning faqat tavsifi beriladi, ya'ni biror maqsadga erishishga yoki qandaydir masalani yechishga qaratilgan ko'rsatmalarning (buyruqlarning) aniq, tushunarli, chekli hamda to'liq tizimi tushuniladi.

Algoritmning asosiy xossalari.

**1-xossa.** Diskretlilik, ya'ni algoritmnini chekli sondagi oddiy ko'rsatmalar ketma-ketligi shaklida ifodalash mumkin.

**2-xossa.** Tushunarilik, ya'ni ijrochiga tavsiya etilayotgan ko'rsatmalar uning uchun tushunarli bo'lishi shart, aks holda ijrochi oddiy amalni ham bajara olmay qolishi mumkin. Har bir ijrochining bajara olishi mumkin bo'lgan ko'rsatmalar tizimi mavjud.

**3-xossa.** Aniqlik, ya'ni ijrochiga berilayotgan ko'rsatmalar aniq mazmunda bo'lishi lozim hamda faqat algoritmda ko'rsatilgan tartibda bajarilishi shart.

**4-xossa.** Ommaviylik, ya'ni har bir algoritm mazmuniga ko'ra bir turdag'i masalalarning barchasi uchun yaroqli bo'lishi lozim. Masalan, ikki oddiy kasr umumiy maxrajini topish algoritmi har qanday kasrlar umumiy maxrajini topish uchun ishlataladi.

**5-xossa.** Natijaviylik, ya'ni har bir algoritm chekli sondagi qadamlardan so'ng albatta natija berishi lozim. Bu xossalari mohiyatini o'rganish va konkret algoritmlar uchun qarab chiqish talabalarning xossalari mazmunini bilib olishlariga yordam beradi.

Algoritmning tasvirlash usullari:

**1. Algoritmning so'zlar orqali ifodalanishi.**

**2. Algoritmning formulalar yordamida berilishi.**

**3. Algoritmning jadval ko'rinishida berilishi,** masalan, turli matematik jadvallar, loteriya yutuqlari jadvali, funksiyalar qiymatlari jadvallari bunga misol bo'ladi.

**4. Algoritmnning dastur shaklida ifodalanishi**, ya’ni algoritm kompyuter ijrochisiga tushunarli bo‘lgan dastur shaklida beriladi.

**5. Algoritmnning algoritmik tilda tasvirlanishi**, ya’ni algoritm bir xil va aniq ifodalash, bajarish uchun qo‘llanadigan belgilash va qoidalar majmui algoritmik til orqali ifodalashdir. Ulardan o‘quv o‘rganish tili sifatida foydalanimoqda.

**6. Algoritmlarning grafik shaklda tasvirlanishi**. Masalan, grafiklar, sxemalar ya’ni blok - sxema bunga misol bo‘la oladi. Blok sxemaning asosiy elementlari quyidagilar: **oval** (ellips shakli)-algoritm boshlanishi va tugallanishi, **to‘g‘ri burchakli to‘rtburchak**-qiymat berish yoki tegishli ko‘rsatmalarni bajarish. **Romb** - shart tekshirishni belgilaydi. Uning yo‘naltiruvchilari tarmoqlar bo‘yicha biri ha ikkinchisi yo‘q yo‘nalishlarni beradi, **parallelogramma**’lumotlarni kiritish yoki chiqarish, yordamchi algoritmga murojaat - parallelogramm ikki tomoni chiziq, yo‘naltiruvchi chiziq - blok-sxemadagi harakat boshqaruvi, nuqta-to‘g‘ri chiziq (ikkita parallel) - qiymat berish.

Algoritmda bajarilishi tugallangan amallar ketma-ketligi algoritm qadami deb yuritiladi. Har bir alohida qadamni ijro etish uchun bajarilishi kerak bo‘lgan amallar haqidagi ko‘rsatma buyruq deb aytildi.

Algoritmlarni ko‘rgazmaliroq qilib tasvirlash uchun blok-sxema, ya’ni geometrik usul ko‘proq qo‘llaniladi. Algoritmnning blok-sxemasi algoritmnning asosiy tuzilishining yaqqol geometrik tasviri: algoritm bloklari, ya’ni geometrik shakllar ko‘rinishida, bloklar orasidagi aloqa esa yunaltirilgan chiziqlar bilan ko‘rsatiladi. Chiziqlarning yunalishi bir blokdan so‘ng qaysi blok bajarilishini bildiradi.

Har qanday algoritm mantiqiy tuzilishga, ya’ni bajarilish tartibiga qarab uch asosiy turga bo‘linadi: chiziqli (ergashish), tarmoqlamivchi va takrorlanuvchi.

**Chiziqli algoritmlar**. Barcha ko‘rsatmalar ketma-ket joylashish tartibida bajarib boriladigan algoritmlar chiziqli algoritmlar deyiladi. «Choy damlash»,

doira yuzini hisoblash algoritmlari chiziqli algoritmlarga misol bo‘ladi. Lekin hayotimizdagi juda ko‘p jarayonlar shartlar asosida boshqariladi.

**Tarmoqlanuvchi algoritmlar.** Shartga muvofiq bajariladigan ko‘rsatmalar ishtirok etgan algoritmlar tarmoqlanuvchi algoritmlar deb ataladi. Algoritmlarning bu turi hayotimizda har kuni va har qadamda uchraydi. Eshikdan chiqishimiz eshik ochiq yoki yopiqligiga, ovqatlanishimiz qornimiz och yoki to‘qligiga yoki taomning turiga, ko‘chaga kiyinib chiqishimiz ob-havoga, biror joyga borish uchun transport vositasini tanlashimiz to‘lash imkonimiz bo‘lgan pulga bog‘liqdir.

Demak, tarmoqlanuvchi algoritmlar chiziqli algoritmlardan tanlanish imkoniyati bilan farqlanar ekan. Kvadrat tenglamani yechish, ikki sonning EKUBini topish algoritmlari tarmoqlanuvchi algoritmlarga misol bo‘ladi.

**Takrorlanuvchi (siklik) algoritmlar.** Masalalarni tahlil etish jarayonida algoritmdagi ba’zi ko‘rsatmalar takroran bajarilishini kuzatish mumkin. Masalan, eng katta kvadratlar kesib olish masalasi, Evklid algoritmi. Hayotimizda ham juda ko‘p jarayonlar takrorlanadi. Shuningdek darslarning har hafta takrorlanishi, har kuni nonushta qilish yoki universitetga borish va hokazo. Ko‘rsatmalari takroriy bajariladigan algoritmlar takrorlanuvchi algoritmlar deb ataladi.

Takrorlanuvchi algoritmlar « $I := I + 1$ », « $S := S + I$ » yoki « $P := P * I$ » ko‘rinishidagi ko‘rsatmalarning ishtiroki bilan ajralib turadi (\* — ko‘paytirish amali). Bunday ko‘rsatmalarning mazmunini tushunish uchun takrorlanishning bir nechta qadamini ko‘rib chiqish lozim.

Odatda, yig‘indi uchun boshlang‘ich qiymat (inglizchadan SUM, ya’ni yig‘indi ma’noli so‘zning bosh harfi)  $S := 0$  va ko‘paytma uchun (inglizchadan PRODUCT, ya’ni ko‘paytma ma’noli so‘zning bosh harfi)  $P := 1$  deb olinadi, chunki bu qiymatlar, ya’ni 0 va 1 lar, mos ravishda, yig‘indi va ko‘paytmaning natijasiga ta’sir etmaydi:

1-qadam:  $I := 1$  bo‘lsin, u holda  $S := S + I = 0 + 1 = 1$ ,  $P := P * I = 1 * 1 = 1$ ;

2-qadam:  $I := I + 1 = 1 + 1 = 2$ ,  $S := S + I = 1 + 2 = 3$ ,  $P := P * I = 1 * 2 = 2$ ;

3-qadam:  $I := I + 1 = 2 + 1 = 3, S := S + I = 3 + 3 = 6, P := P * I = 2 * 3 = 6;$

4-qadam:  $I := I + 1 = 3 + 1 = 4, S := S + I = 6 + 4 = 10, P := P * I = 6 * 4 = 24.$

Algoritmnini ishlab chiqishda uni bir nechta xil usul bilan ifodalab bersa bo‘ladi. Shulardan uchtasi keng tarqalgan bular:

1. Algoritmnini oddiy tilda tavsiflash
2. Algoritmnini tizim ko‘rinishida ifodalash
3. Algoritmnini maxsus (algoritmik) tilda yozish.

C++ tili Byarn Straustrup tomonidan 1980 yil boshlarida ishlab chiqilgan.

Bu til asosan tizim sathida dasturlovchilar uchun yaratilgan.

**Til alifbosi.** C++ tili alifbosi quyidagilarni o‘z ichiga oladi:

- Katta va kichik lotin alifbosi harflari va ostki chiziq belgisi;
- 0 dan 9 gacha bo‘lgan arab raqamlari;
- Maxsus belgilar: " { } , [ ] ( ) + - % \* . \ : ? < = ! > & ~ ; ^
- bo‘sh joy belgisi, tabulyatsiya belgisi, yangi qatorga o‘tish belgisi;
- identifikatorlar;
- kalit so‘zlar;
- operatsiya belgilari;
- o‘zgarmaslar;
- ajratuvchi belgilar;

**Leksema.** Alifbo belgilaridan tilning leksemalari shakllanadi (mustaqil ma’noga ega bo‘lgan tilning minimal birligi).

Leksemalar turlari:

- identifikatorlar;
- kalit so‘zlar;
- operatsiyalar belgilari;
- o‘zgarmas;
- ajratuvchi (qavs, vergul, bo‘sh joy belgilari).

**Identifikator** – bu dastur ob'ektining nomi. Identifikatorlar lotin harflari, ostki chiziq belgisi va sonlar ketma-ketligidan iborat bo'ladi. Identifikator lotin harfidan yoki ostki chiziq belgisidan boshlanishi lozim. Identifikatordag'i birinchi belgi harf yoki pastki chiziq bo'lishi mumkin. Identifikator ichida bo'shliq bo'lmasligi kerak.

Misol uchun:

A2, \_MIN, nomber\_02, RIM, rim

Katta va kichik harflar farqlanadi, shuning uchun oxirgi ikki identifikator bir-biridan farq qiladi.

**Kalit so'zlar.** C++ tilida ayrim so'zlar oldindan zahiralanadi. Bunday so'zlar kalitli so'zlar deb aytildi. Bunday so'zlarni o'zgaruvchilarni nomlashda ishlatish mumkin emas. Ularga **if**, **while**, **for** va **main** kabi so'zlar kiradi. C++ tilidagi kalit so'zlar ro'yxati 1-jadvalda keltirilgan.

1.1-jadval. C++tilidagi kalit so'zlar ro'yxati

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typeid
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t

double	mutable	switch	while
dynamic_cast	namespace	template	

## **Ma'lumotlarning asosiy turlari. Asosiy ma'lumotlar turlari**

quyidagilardan iborat:

- int (butun son);
- char (belgi);
- wchar\_t (kengaytirilgan belgi);
- bool (mantiqiy);
- float (haqiqiy);
- double(ikkilangan aniqlikdagi haqiqiy).

Standart turlarning qiymatlar diapazonini aniqlash uchun to'rt turdagi spetsifikatorlar ishlatiladi:

- short(qisqa);
- long(uzun);
- signed (imzolangan);
- unsigned(imzo qo'yilmagan).

1.2-jadval. Turlarning o'lchamlari va qiymatlarining o'zgarish oralig'i

Tur	O'lchami (байт)	Qiymatning o'zgarish oralig'i
<b>bool</b>	1	true, false
<b>signed char</b>	1	-128 ... 127
<b>unsigned char</b>	1	0 ... 255
<b>signed short int</b>	2	-32768 ... 32767
<b>unsigned short int</b>	2	0 ... 65535
		-2 147 483 648 ...
<b>signed long int</b>	4	2 147 483 647
<b>unsigned long int</b>	4	0 ... 4 294 967 295
<b>float</b>	4	3.4e-38 ... 3.4e+38
<b>double</b>	8	1.7e-308 ... 1.7e+308
<b>long double</b>		3.4e-4932 ... 3.4e+4932

**Arifmetik ifoda va amallar.** C++ tilida ifodalar biror bir hisoblash natijasini qaytaruvchi boshqa ifodalar ketma-ketligini boshqaradi yoki hech nima qilmaydi (nol ifodalar).

C++ tilida barcha ifodalar nuqtali vergul bilan yakunlanadi. Ifodaga misol qilib o'zlashtirish amalini olish mumkin.

$$x=a+b;$$

Algebradan farqli ravishda bu ifoda  $x = a+b$  ga teng ekanligini anglatmaydi. Bu ifodani quyidagicha tushinish kerak:

**a** va **b** o'zgaruvchilarni qiymatlarini yig'ib natijasini **x** o'zgaruvchiga beramiz yoki **x** o'zgaruvchiga  $a+b$  qiymatni o'zlashtiramiz. Bu ifoda birdaniga ikkita amalni bajaradi, yig'indini hisoblaydi va natijani o'zlashtiradi. Ifodadan so'ng nuqtali vergul qo'yiladi. (=) operatori o'zidan chap tomondagи operandga o'ng tomondagи operandlar ustida bajarilgan amallar natijasini o'zlashtiradi.

Bajarilishi natijasida biror bir qiymat qaytaradigan barcha ifodalar C++ tilida amallar deyiladi. Amallar albatta biror bir qiymat qaytaradi. Masalan,  $4+6$  amali 10 qiymatni qaytaradi.

Inkrement va dekrement. Dasturlarda o‘zgaruvchiga 1 ni qo‘shish va ayirish amallari juda ko‘p hollarda uchraydi. C++ tilida qiymatni 1 ga oshirish inkrement, 1 ga kamaytirish esa dekrement deyiladi. Bu amallar uchun maxsus operatorlar mavjuddir.

Inkrement operatori (++) o‘zgaruvchi qiymatini 1 ga oshiradi, dekrement operatori (--) esa o‘zgaruvchi qiymatini 1 ga kamaytiradi. Masalan, a o‘zgaruvchisiga 1 qiymatni qo‘shmoqchi bo‘lsak quyidagi ifodani yozishimiz lozim.

`a++ //a o‘zgaruvchi qiymatini 1 ga oshirdik.`

Bu ifodani quyidagicha yozishimiz mumkin edi.

`a=a+1;`

Bu ifoda o‘z navbatida quyidagi ifodaga teng kuchli:

`a+=1;`

Dasturlash jarayonida dasturchi o‘z dasturini yaratish uchun xar hil turdag'i kodlar yozadi. Kodlarning qanday bo‘lishi qaysi dasturlash tilidan foydalanib, dastur tuzishga bog‘liq bo‘ladi. Dasturlash tillaridan C, C++, Java, ... . Yozilgan kodlarni kompyuter tushunmaydi, kompyuter tushunishi uchun uchun bu kodlarni kompyuter tushunadigan tilga o‘zgartirish lozim. Mana shu vaziyatda 2 ta termin kerak bo‘ladi(kompilyator yoki interpreter).

Kompyuter faqatgina raqamli kodlarni tushunadi, ya’ni **0** yoki **1**. Bu 2 son orqali dastur tuzish juda qiyin hisoblanadi. Shuning uchun, insonlar tushunadigan qilib dasturlash tillari yaratilgan. Dasturchi dasturlash tillari orqali kodlar yozadi va bu kodlar kompyuter tushunadigan 0 va 1 sonlariga almashtiriladi va dastur kompyuterda ishlaydi, bu jarayonni kompilyator yoki interpreter amalga oshirib beradi.

**Kompilyator** — murakkab dasturdir, dasturlash tilida yozilgan barcha kodlarni birdaniga **ob’yektli kodga** o‘zgartirib beradi. Ob’yektli kodni yana **ikkilik kod** yoki **mashina kodi** deb ham atashadi. Keyinchalik bu ob’yektli kod kompyuterda to‘g‘ridan to‘g‘ri ishlatilishi mumkin bo‘ladi. Dasturlash tillarida

yozilgan kodlar bu ob'yektli kodga ta'sir qilmaydi. Ob'yektli kodni o'zgartirish uchun esa, qaytadan kompilyatsiya qilinib ob'yektli kod o'zgartiriladi. Natija bajariladigan, **.exe** ko'rinishidagi fayl bo'ladi. Bu faylni bloknotda ochib o'zgartirib bo'lmaydi, ya'ni bu fayl tayyor dastur hisoblanadi. Kompilyatorning kamchiligi sifatida, dasturlash tilidagi ma'lum bir qatorlarni alohida tekshirish imkoniyati yo'qligidir, uning uchun ob'yektli kod yaratib, uni ishga tushurish lozim bo'ladi, ortiqcha ish bo'lib qoladi. Undan tashqari ba'zi kompilyatorlar bir dasturlash tilidan, ikkinchisiga ham o'zgartirib berishi mumkin. Kompilyator ishlatadigan dasturlash tillariga C, C++, Delphilarni misol qilib keltirish mumkin.

**Interpreter** — ham dastur ham jihoz ko'rinishida bo'lishi mumkin. Bu ham kompyuter tiliga o'zgartirib berish vazifasini bajaradi, faqatgina ishlash texnologiyasi boshqacharoqdir. Interpreter, dasturlash tilida yozilgan kodlarni ketma — ket o'qib, mashina tiliga o'zgartirib boradi. Xatolik paydo bo'lsa, o'sha zahoti dasturchiga ma'lum qiladi. Bu ketma — ketlikda o'zgartirish, kompilyatorga nisbatan sekinroq amalga oshiriladi(ba'zi hollarda kompilyatorga qaraganda 50 barobar sekin). Dastur natijasini ko'rish uchun, har safar kodlarni interpretatordan o'tkazish kerak bo'ladi(kompilyatorga o'xshab bir marotaba ob'yekt kod yaratib qo'yib, keyin har doim ishlatishning iloji yo'q). Bundan ko'rini turibdiki, interpreter asosan saytlar, umumiyl holda veb dasturlashda ishlatiladi. Biror saytning yuklanishi jarayoni uzunligi, interpretatorda o'zgartirish amalga oshirilishi bilan tushuntirilishi mumkin. Interpreter ishlatadigan dasturlash tillariga PHP, JavaScript, JScript, Basiclar misol bo'la oladi.

Bu ikki termin umumiyl holda **translyator** deyiladi, ya'ni o'zgartirgichlardir. Biror projekt amalga oshirilganda bu 2 o'zgartirgichlar birgalikda ham ishlatilishi mumkin.

## **Nazorat savollari**

1. Masalaning qo‘yilishini aniqlash va matematik modelini ishlab chiqishni tushuntirib bering.
2. Masalani yechishning sonli usulini tanlash qanday amalga oshiriladi?
3. Masalani yechish algoritmini ishlab chiqish deganda nimani tushunasiz?
4. Kompyuter uchun dastur tuzish qanday amalga oshiriladi?
5. Dasturni kompyuter xotirasiga kiritish, rostlash va tekshirish deganda nimani tushunasiz?
6. Hisoblash natijalarini qayta ishlash va tahlil qilish qanday amalga oshiriladi?
7. Algoritm va uning asosiy xossalari tushuntirib bering?
8. Algoritmning tasvirlash usullari deganda nimani tushunasiz?
9. Chiziqli algoritmlar deganda nimani tushunasiz?
10. Tarmoqlanuvchi algoritmlar deganda nimani tushuniladi?
11. Takrorlanuvchi (siklik) algoritmlar deganda nimani tushunasiz?
12. Leksema va identifikatorni tushuntirib bering?

## 1.2-§. DASTURLASH TILLARINING TUZILMASI.

**Reja:**

- 1. Chiziqli algoritmlarni tashkil qilish.**
- 2. Algebraik ifodalarni matematik kutubxona funksiyalari yordamida hisoblash.**

Dastur bajarilishi jarayonida o‘z qiymatini o‘zgartira oladigan kattaliklar o‘zgaruvchilar deyiladi. O‘zgaruvchilarning nomlari lotin harfdan boshlanuvchi harf va raqamlardan iborat bo‘lishi mumkin. O‘zgaruvchilarni belgilashda katta va kichik harflarning farqlari bor (A va a harflari 2 ta o‘zgaruvchini bildiradi). Har bir o‘zgaruvchi o‘z nomiga, toifasiga, xotiradan egallagan joyiga va son qiymatiga ega bo‘lishi kerak. O‘zgaruvchiga murojaat qilish uning ismi orqali bo‘ladi. O‘zgaruvchi uchun xotiradan ajratilgan joyning tartib raqami uning adresi hisoblanadi. O‘zgaruvchi ishlatilishidan oldin u aniqlangan bo‘lishi lozim.

O‘zgaruvchilarning son qiymatlari quyidagi ko‘rinishda yoziladi:

- Butun toifali o‘nlik sanoq tizimida: ular faqat butun sondan iborat bo‘ladi.

Masalan: 5; 76; -674 va h.k.

• Sakkizlik sanoq tizimidagi sonlar: 0 (nol)dan boshlanib, 0 dan 7 gacha bo‘lgan raqamlardan tashkil topadi. Masalan:  $x=0453217$ ;  $s=077$ ;

• O‘n otilik sanoq tizimidagi sonlar: 0 (nol) dan boshlanadi va undan keyin x yoki X harfi keladi, so‘ngra 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma-ketliklar bo‘ladi. Masalan: 10 s.s.dagi 22 soni 8 s.s. da 026, 16 s.s.da 0x16 shaklida bo‘ladi.

• Haqiqiy toifali sonlar: ular butun va kasr qismlardan iborat bo‘ladi. Masalan: 8,1; -12,59 va x.k. Haqiqiy toifali sonlarning bu ko‘rinishi oddiy ko‘rinish deyiladi. Juda katta yoki juda kichik haqiqiy toifali sonlarni darajali (eksponensional) formada yozish qulay. Masalan:  $7,204 \cdot 10^{12}$  yoki  $3,567 \cdot 10^{-11}$  kabi sonlar  $7.204e+12$  va  $3.567e-11$  ko‘rinishda yoziladi.

- Simvolli konstantalar. Ular qatoriga dastur bajarilishi“ ichida qabul qilinadigan simvollar kiradi.

C++ tilida har qanday o‘zgaruvchi ishlatilishidan oldin e’lon qilinishi kerak. E’lon qilish degani ularning toifalarini aniqlab qo‘yish demakdir.

C++ tilida quyidagi toifali o‘zgaruvchilar ishlatiladi:

- Butun toifali kichik sonlar yoki simvollar uchun: char uning o‘zgarish intervali -128 dan +127 gacha yoki apostrof ichidagi ixtiyoriy 1ta simvol. Xotiradan 1 bayt joy oladi. Simvollar ASCII kodlariga mos keladi. ( ASCII – American Standart Code for Information Interchange)

• Butun toifali o‘zgaruvchilar: int. Masalan: int a, i, j ; Bu yerda dasturda ishlatilayotgan a, i, j o‘zgaruvchilarining toifasi butun ekanligi ko‘rsatildi. Bu toifadagi o‘zgaruvchilar 2 bayt joy egallaydi. Ularning o‘zgarish intervali: -32768 dan +32767 gacha; (Hozirgi 32 razryadli kompyuterlarda 4 bayt joy oladi va oralig‘i 2 marta oshgan).

• Butun toifali katta (uzun) o‘zgaruvchilar: long. Masalan: long s, s2, aa34; Bu toifadagi o‘zgaruvchilar 4 bayt joy egallaydi. Ular -2147483648 dan +2147483647 oraliqdagi sonlarni qabul qilishi mumkin.

• Ishorasiz butun o‘zgaruvchilar: unsigned short – 2 bayt joy oladi, o‘zgarish intervali 0 dan 65535 gacha; unsigned long – 4 bayt joy oladi, o‘zgarish intervali: 0 dan 4294967295 gacha; unsigned char – 1 bayt joy oladi, o‘zgarish chegarasi 0 dan 255 gacha.

• Haqiqiy toifadagi o‘zgaruvchilar: float. Masalan: float a, b: Bu yerda dasturda ishlatilayotgan a, b o‘zgaruvchilarining toifasi haqiqiy ekanligi ko‘rsatilgan. Bu toifadagi o‘zgaruvchilar 4 bayt joy egallaydi va qabul qilish chegarasi 10-38 dan 10+38 gacha.

• Katta yoki kichik qiymatli o‘zgaruvchilarni ifoda etishda double toifasi ishlatiladi. Ular uchun 8 bayt joy ajratiladi va qabul qilish chegarasi 10-304 dan 10+304 gacha.

•Juda katta yoki juda kichik qiymatli o‘zgaruvchilar uchun long double toifasi ishlatiladi, u 10 bayt joy oladi va qabul qilish chegarasi  $3.4*10-4932$  dan  $1.1*10-4932$ gacha.

•Qator toifasidagi o‘zgaruvchilar uchun ham char toifasi belgilangan. Ular ham 1 bayt joy oladi va 0 dan 256 tagacha bo‘lgan simvollar ketma-ketligidan iborat bo‘lishi mumkin. Satr toifasidagi o‘zgaruvchilar qo‘shtirnoq (“”) ichida yoziladi.

C++ tilida o‘zgaruvchilarni inisializasiya qilish degan tushuncha ham mavjud. Inisializasiya qilish degani o‘zgaruvchini e’lon qilish barobarida unga boshlang‘ich qiymatini ham berish demakdir. Masalan: int a=5, s=-100; - a, s o‘zgaruvchilari butun toifali ekanligi ko‘rsatildi va a o‘zgaruvchisiga 5 (a=5), s o‘zgaruvchisiga esa -100 (s=-100) boshlang‘ich qiymatlar berildi.

Dastur bajarilishi jarayonida o‘z qiymatini o‘zgartira olmaydigan kattaliklar o‘zgarmaslar deyiladi. Masalan: x=1; bo‘lsa keyinchalik x=x+5 deb yozib bo‘lmaydi. O‘zgarmaslarni const so‘zi bilan ko‘rsatiladi. Maslan: const int x=95; float y=9.17; ( const lar simvol yoki nol (NULL) bo‘lishi xam mumkin.)

### **C++ tilidagi dastur quyidagi tarkibdan tashkil topadi:**

1. Direktivalar – # include <file.h> direktiva – instruksiya degan ma’noni beradi. C++ tilida dasturning tuzilishiga, ya’ni ehtiyojiga qarab, kerakli direktivalar ishlatiladi. Ular <> belgisi orasida keltiriladi. Umuman olganda quyidagi direktivalar mavjud (jami 32 ta):

```
#include <stdio.h> - C da oddiy kiritish/chiqarish dasturi uchun. Bu yerda std - standart, i – input, o - output degani.
```

•#include <iostream.h> - C++ da kiritish/chiqarish uchun, oddiy amallar bajarilsa.

•#include <math.h> - standart matematik funksiyalarni ishlatish uchun.

•#include <conio.h> - dasturning tashqi ko‘rinishini shakllantirish uchun.

- `#include <string.h>` - satr toifasidagi o‘zgaruvchilar ustida amallar bajarish uchun.

- `#include <stdlib.h>` - standart kutubxona fayllarini chaqirish uchun.

- `#include <time.h>` - kompyuter ichidagi soat qiymatlaridan foydalanish uchun.

- `#include <graphics.h>` - C++ tilining grafik imkoniyatlaridan foydalanish uchun.

Bu fayllar maxsus kutubxona e’lon fayllari hisoblanadilar va ular alohida INCLUDE deb nomlanadigan papkada saqlanadi. Hozirda C++ kutubxonasi yangilangan va undagi fayllarning nomlaridan .h (head – bosh ma’nosida) kengaytmasi olib tashlandi va oldiga c harfi qo’shildi (C dan qolgan 18 tasiga).

Direktivalar dasturni uni kompilyasiya qilinishidan oldin tekshirib chiqadi.

2. Makroslar - `# define` makro qiymati. Masalan:

```
#define y sin(x+25) - u = sin(x+25) qiymati berildi;
```

```
#define pi 3.1415 - pi = 3.1415
```

```
#define s(x) x*x - s(x) = x*x (; belgisi qo‘yilmaydi)
```

Global o‘zgaruvchilarni e’lon qilish. Asosiy funksiya ichida e’lon qilingan o‘zgaruvchilar lokal, funksiyadan tashqarida e’lon qilinganlari esa global o‘zgaruvchilar deyiladi. Global o‘zgaruvchilar dastur davomida ishlaydi va xotiradan ma’lum joyni egallaydi. O‘zgaruvchini bevosita ishlatishdan oldin e’lon qilsa ham bo‘ladi, u holda u lokal bo‘ladi. Global o‘zgaruvchilar nomi lokal o‘zgaruvchilar nomi bilan bir xil bo‘lishi ham mumkin. Bunday holatda lokal o‘zgaruvchining qiymati joriy funksiya ichidagini qiymatini o‘zgartiradi, funksiyadan chiqishi bilan global o‘zgaruvchilar ishlaydi.

Asosiy funksiya - `main ( )` hisoblanadi. Bu funksiya dasturda bo‘lishi shart. Umuman olganda C++ tilidagi dastur funksiyalardan iborat deb qaraladi. `main ( )` funksiyasi { boshlanadi va dastur oxirida berkitilishi shart } . `main` – asosiy degan ma’noni beradi. Bu funksiya oldida uning toifasi ko‘rsatiladi. Agar `main ( )`

funksiyasi beradigan (qaytaradigan) javob oddiy so‘z yoki gaplardan iborat bo‘lsa, hech qanday natija qaytarmasa, void so‘zi keltiriladi. main ( ) funksiyasi dastur tomonidan emas, balki OS tomonidan chaqiriladi. OSga qiymat qaytarish shart emas, chunki u bu qiymatdan foydalanmaydi. Shuning uchun main ( ) funksiyasining turini void deb ko‘rsatganimiz ma’qul. Har bir funksiyaning o‘z argumenti bo‘ladi, shuning uchun main funksiya ( ) lari ichiga uning parametri keltiriladi. Ba’zan u bo‘sh bo‘lishi ham mumkin. Bu funksiyadan chiqish uchun odatda return operatori ishlatiladi. 0 (nol) qiymatining qaytarilishi operasion tizimga ushbu dastur normal bajarilib turganini bildiradi. return orqali qaytadigan qiymat toifasi funksiya e’lonidagi qaytish toifasi bilan bir xil bo‘lishi kerak. Masalan int main () va 0 (nol) qiymat butun toifalidir. Bu funksiyadan so‘ng lokal o‘zgaruvchilar, qism dasturlar, ularning haqiqiy parametrlari e’lon qilinadi. So‘ngra dasturning asosiy operatorlari (kiritish/chiqarish, hisoblash va h.k.) yoziladi. Agar bu operatorlar murakkab toifali bo‘lsalar, ularni alohida {} qavslarga olinadi. C++ tilida dastur kichik harflarda yoziladi. Ba’zi operatorlar katta harflar bilan kelishi mumkin, bunday xollarda ular alohida aytib o‘tiladi. Operatorlar oxiriga ; belgisi qo‘yiladi. Operatorlar bir qatorga ketma-ket yozilishi mumkin. Dasturda izohlar ham kelishi mumkin, ular /\* .... \*/ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

### **main () // C++ tilining asosiy funksiyasi**

C++ tilida quyidagi amallardan foydalanish mumkin:

Arifmetik amallar: +, -, /, \*, %. Barcha amallar odatdagidek bajariladi, faqat bo‘lish amali butunga bo‘lish bajariladi, ya’ni agar butun sonlar ustida bajarilayotgan bo‘lsa, natija doim butun bo‘ladi, ya’ni kasr qism tashlab yuboriladi ( $9/5=1$ ; vaxolanki  $1,8$  bo‘lishi kerak). Shuning uchun surat yoki maxrajiga nuqta (.) qo‘yilsa, natija ham haqiqiy bo‘ladi ( $9./5=1.8$ ). % belgisi (modul operatori) esa butun sonni butun songa bo‘lgandan hosil bo‘ladigan qoldiqni bildiradi.

**Masalan: 9 % 5=4**

Taqqoslash amallari:  $=$  (tengmi?);  $!=$  (teng emas);  $<$ ;  $>$ ;  $>=$ ;  $<=$

Mantiqiy amallar:  $\&\&$  (and) mantiqiy ko‘paytirish;  $\parallel$  (or) mantiqiy qo‘shish;  $!$  (not) mantiqiy inkor. Mantiqiy amallarni ixtiyoriy sonlar ustida bajarish mumkin. Agar javob rost bo‘lsa, natija 1 bo‘ladi, agar javob yolg‘on bo‘lsa, natija 0 bo‘ladi. Umuman olganda 0 (nol)dan farqli javob rost deb qabul qilinadi.

**Masalan:  $i>50 \&\& j==24$  yoki  $s1 < s2 \&\& (s3>50 \parallel s4<=20)$ ;**

Yoki  $6 \leq x \leq 10$  yozuvini  $x>=6 \&\& x<=10$  deb yoziladi

Qiymat berish amallari:

$a=5$ ;  $b = 2*c$ ;  $x = y = z = 1$ ;  $a = (b = c)*d // 3=5$  deb yozib bo‘lmaydi

qabul qildim va almashtirdim deb nomalandigan amallar:

$+= : a+=b \rightarrow a = a + b;$

$- = : a-=b \rightarrow a = a - b;$

$* = : a*=b \rightarrow a = a * b;$

$/ = : a/=b \rightarrow a = a / b;$

$\% = : a\%=b \rightarrow a = a \% b;$

- inkrement operatsiyasi  $(++)$  ikki ma’noda ishlatiladi: o‘zgaruvchiga murojaat qilinganidan keyin uning qiymati 1 ga oshadi ( $a++$  postfiks ko‘rinishi) va o‘zgaruvchining qiymati uning murojaat qilishdan oldin 1 ga oshadi  $(++a$  prefix ko‘rinishi);

- dekrement operatsiyasi  $(--)$ , xuddi inkrement operatsiyasi kabi, faqat kamaytirish uchun ishlatiladi. Masalan:  $s = a + b++$  ( $a$  ga  $b$  ni qo‘shib keyin  $b$  ning qiymatini 1 ga oshiradi);  $s = a+(--b)$  ( $b$  ning qiymatini 1 ga kamaytirib, keyin  $a$  ga qo‘shadi).

Yuqoridagi standart funksiyalardan tashqari yana quyidagi funksiyalar ham ishlatiladi:

- ceil ( $x$ ) -  $x$  ni  $x$  dan katta yoki unga teng bo‘lgan eng kichik butun songacha yaxlitlash. Masalan:  $\text{ceil}(12.6) = 13.0$ ;  $\text{ceil}(-2.4) = -2.0$ ;

- floor (x) - x ni x dan kichik bo‘lgan eng katta butun songacha yaxlitlash.

Masalan: floor (4.8) = 4.0; floor (-15.9) = -16.0; floor(12.1) = 12; floor(-12.1)=-13;

- fmod (x,y) – x / y ning qoldig‘ini kasr son ko‘rinishida berish. Masalan: fmod(7.3, 1.7) = 0.5;

Dasturlar. Kompyuterni biron bir amalni bajarishga majburlash uchun, siz (yoki boshqalar) unga nima xoxlayotganingizni aniq, batafsil aytishingiz kerak.

Bu ma’ruzada biz juda oddiy dasturdan boshlanadigan kodni hamda uning bajarilishi uchun kerak bo‘ladigan bir qancha usullarni ko‘rib chiqamiz.

Birinchi klassik dasturlarni variantlarini keltiramiz. U ekranga Hello, World! xabarini chiqaradi.

```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     cout<<"Hello, World!\n";
6     return 0;
7 }
8 // izoh hisoblanadi

```

Hello, World!

// (ya’ni, ikkita egri chiziqdan keyin ) belgidan keyin yozilgan barchasi izoh hisoblanadi. U kompilyator tomonidan inkor qilinadi va dasturni o‘quvchi dasturchilar uchun mo‘ljallangan. Bu holatda biz satrning birinchi qismi nimani anglatishini sizga xabar qilish uchun izohdan foydalandik.

```
#include <iostream>
```

Bu #include direktivasini akslantiradi. U iostream faylida tasvirlanganlarni imkoniyatlarini «faollashtirish» uchun kompyuterni majburlaydi. Bu fayl C++ (C++ tilining standart kutubxonasi) tilining barcha amalga oshirishlarida ko‘zda tutilgan imkoniyatlaridan foydalanishni osonlashtiradi.

iostream faylning imkoniyatlari berilgan dastur uchun shunda hisoblanadiki, uning yordami bilan biz C++ tilining standart kiritish-chiqarish vositalaridan foydalanish imkoniyatiga ega bo‘lamiz. Bu yerda biz faqatgina cout

standart chiqarish potoki va << chiqarish operatoridan foydalanamiz. Sarlavha biz dasturimizda foydalanadigan cout kabi atamalarni aniqlashni o‘z ichiga oladi.

Dastur bajarilishi boshlanadigan nuqtani dastur qanday aniqlaydi? U main nomli funksiyani ko‘rib chiqadi va uni qo‘llanmalarini bajarishni boshlaydi. Bizning “Hello, World!” dasturimizda main funksiyasi quyidagicha ko‘rinadi:

```
int main() // C++ da dasturlash main funksiyasi yordamida amalga oshiriladi
```

```
{  
    cout <<“Hello, World!\n”; // «Hello, World!» chiqarish  
    return 0;  
}
```

Bajarishning boshlang‘ich nuqtasini aniqlash uchun, C++ tilidagi har bir dastur main nomli funksiyadan tashkil topgan bo‘lishi zarur. Bu funksiya to‘rtta qismdan iborat.

1. Qiymat qaytaruvchi toifa, bu funksiyada - int toifa (ya’ni, butun son), funksiya chaqirish nuqtasida qanday natija qaytarishini aniqlaydi (agar u qandaydir qiymat qaytarsa). int so‘zi C++ tilida zahiralangan hisoblanadi (kalit so‘z), shuning uchun uni boshqa bir narsaning nomi sifatida foydalanish mumkin emas.

2. Nom, main berilgan holatda.

3. Parametrlar ro‘yxati, aylana qavsda berilgan; berilgan holatda parametrlar ro‘yxati bo‘sh.

4. Funksiya tanasi, figurali qavsda berilgan va funksiya bajarishi kerak bo‘lgan faoliyatlar ro‘yxati.

Bundan kelib chiqadiki, C++ tilidagi kichik dastur quyidagicha ko‘rinadi:

```
int main(){ }
```

Bu dastur hech qanday amal bajarmaganligi uchun undan foyda kam. “Hello, World!” dasturining main funksiyasining tanasi ikkita qo‘llanmadan iborat:

```

cout<<“Hello,World!\n”;      //“Hello,World!” chiqish
return 0;

```

Birinchidan, u ekranga Hello, World! satrini chiqaradi, so‘ngra chaqirish nuqtasiga 0 (nol) qiymat qaytaradi. Qachonki main() funksiyasi tizim sifatida chaqirilsa, biz qiymat qaytarmasdan foydalanamiz. Lekin ayrim tizimlarda (Unix/Linux) bu qiymatdan dasturni muvaffaqiyatli bajarilishini tekshirish uchun foydalanish mumkin. main() funksiyasidagi qaytariluvchi Nol (0), dastur muvaffaqiyatli bajarilganini bildiradi.

Chiziqli algoritmlarni tashkil qilishga oid misol.

Misol. Ikkita musbat son berilgan, bu sonlarning o‘rta arifmetik va o‘rta geometrik qiymatlarini aniqlang.

### C++ da dasturi:

```

1 #include<iostream>
2 #include <math.h>
3 using namespace std;
4 int main()
5 {
6     float a,b,s,p;
7     cout<<"a=";cin>>a;
8     cout<<"b=";cin>>b;
9     s=(a+b)/2;
10    p=sqrt(a*b);
11    cout <<"s=" <<s<<endl;
12    cout<<"p=" <<p;
13 }

```

a=2  
b=3  
s=2.5  
p=2.44949

## Algebraik ifodalarni matematik kutubxona funksiyalari yordamida hisoblash.

C++ dasturlash tilida raqamlarda matematik vazifalarni bajarish imkoniyatini beradigan ko‘pgina funksiyalar mavjud.

### MIN va MAX

MAX funksiyasidan eng yuqori qiymatini topish uchun foydalanish mumkin x va y : max(x,y)

```
cout << max(13, 15);
```

Shuningdek min funksiya x va y ning eng past qiymatini topish uchun ishlatalishi mumkin :  $\min(x,y)$

```
cout << min(5, 10);
```

C ++ <math.h> Kutubxonasi.

$\sqrt{}$ (Kvadrat ildiz),  $\text{round}$ (sonni aylantiradi) va  $\log$  (natural logarifm) kabi boshqa funktsiyalarni <cmath> kutubxonasida topish mumkin :

```
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int main() {
5     cout << sqrt(64) << "\n";
6     cout << round(2.6) << "\n";
7     cout << log(2) << "\n";
8 }
```

```
8
3
0.693147
```

Boshqa matematik vazifalar.

Boshqa Math funktsiyalari ( <math.> kutubxonasida) ro‘yxatini quyidagi jadvalda topish mumkin.

1.3. Jadval. Math kutubxonasi funktsiyalari ro‘yxati

Funksiya	Vazifasi
$\text{abs}(x)$	<b>x soning modulini qaytaradi</b>
$\text{acos}(x)$	x ni $\arccos$ radian bilan qaytaradi
$\text{asin}(x)$	x ni $\arcsin$ radian bilan qaytaradi
$\text{atan}(x)$	x ni $\arctg$ radian bilan qaytaradi
$\text{cbrt}(x)$	x ning kub ildizini qaytaradi
$\text{ceil}(x)$	x qiymatini eng yaqin butun songa qadar qaytaradi
$\text{cos}(x)$	$\cos(x)$ ni hisoblaydi
$\text{sin}(x)$	$\sin(x)$ ni hisoblaydi
$\text{exp}(x)$	expotensial qiymatini qaytaradi
$\text{expm1}(x)$	$e^x - 1$ qiymatni qaytaradi
$\text{fabs}(x)$	haqiqiy son x ning modul qiymatini qaytaradi
$\text{fdim}(x,y)$	x va y o‘rtasidagi farqni qaytaradi
$\text{floor}(x)$	x qiymatini eng yaqin butun songa qaytaradi

Funksiya	Vazifasi
hypot(x,y)	$\sqrt{x^2+y^2}$ natijani qaytaradi
fmod(x,y)	$x/y$ ning haqiqiy qiymatining qoldiq qismini qaytaradi
pow(x,y)	$x$ ni $y$ darajaga ko‘taradi
tan(x)	$\tan(x)$ ni hisoblaydi. $\cot(x)$ ni hisoblash uchun $1/\tan(x)$ yoziladi
sqrt(x)	$x$ ni kvadrat ildizi
log(x)	$x$ ni natural logorifmi

### Nazorat savollari

1. C++ da o‘zgaruvchilar deganda nima tushuniladi?
2. C++ tilida qanday toifali o‘zgaruvchilar ishlatiladi?
3. Direktiva deganda nima tushuniladi?
4. Global o‘zgaruvchilar nima?
5. Lokal o‘zgaruvchilar nima?
6. C++ da izohlar qanday yoziladi?
7. math.h kutubxonasidan nima maqsadda foydalaniadi?
8.  $x$  ni kvadrat ildizi qanday chiqariladi?
9.  $x$  ni  $y$  darajaga ko‘tarish qanday amalga oshiriladi?

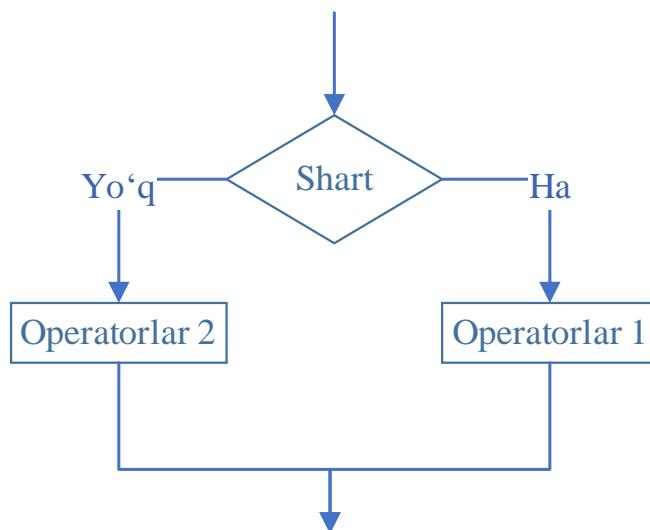
## 1.3-§. TARMOQLANISH VA TANLASH OPERATORLARI.

**Reja:**

- 1. Tarmoqlanuvchi operatorlar va ularni ishlash tartibi.**
- 2. Ternar operatori.**
- 3. Shartsiz o‘tish operatori.**

Agar algoritm qadamlari ketma-ket bajarilish jarayonida qandaydir shartga bo‘gлиq ravishda o‘zgarsa, bunday algoritm tarmoqlanuvchi algoritm deb nomlanadi. Shart bu mantiqiy ifoda bo‘lib, faqat rost yoki yolg‘on qiymatni qabul qiladi. Agar shart rost bo‘lsa Xa, yolg‘on bo‘lsa Yo‘q tarmog‘i bo‘yicha algoritm qadami davom etadi.

Tarmoqlanuvchi algoritm to‘liq tarmoqlanuvchi va to‘liqmas tarmoqlanuvchi turlariga bo‘linadi. To‘liq tarmoqlanuvchi algoritmda shart bajarilganda va bajarilmaganda ikkalasida ham amallar bajariladi.



**1.1-rasm. To‘liq tarmoqlanuvchi algoritm bloksxemasi**

Agar shart bajarilsa Operatorlar1 bajariladi, aks holda Operatorlar2 bajariladi.

Tarmoqlanish shart asosida bo‘ladi. Shart mantiq ifoda bo‘ladi. Mantiqiy ifoda mantiqiy o‘zgaruvchi, taqqoslash amallari yoki ularning inkor, konyuksiya,

dizyunksiya amallaridan iborat bo‘lishi mumkin. Shart operatori C++ da quyidagicha yoziladi:

```

1 if (shart) {
2   Operatorlar1;
3 }
4 else {
5   Operatorlar2;
6 }

```

1.4. Jadval. C++ da taqqoslash amallari

Nº	Matematika	C++
<b>1</b>	>	>
<b>2</b>	<	<
<b>3</b>	$\geq$	$\geq$
<b>4</b>	$\leq$	$\leq$
<b>5</b>	$=$	$=$
<b>6</b>	$\neq$	$\neq$

Misol:

$$y = \begin{cases} x^2, \text{agar } x \geq 0 \\ 2 * x, \text{agar } x < 0 \end{cases}$$

**Yechimi:** y ning qiymati x ga bog‘liq ravishda yoki  $x^2$  formula, yoki  $2x$  formula bo‘yicha hisoblanadi. Tekshirilishi kerak bo‘lgan shart  $x \geq 0$ .

```

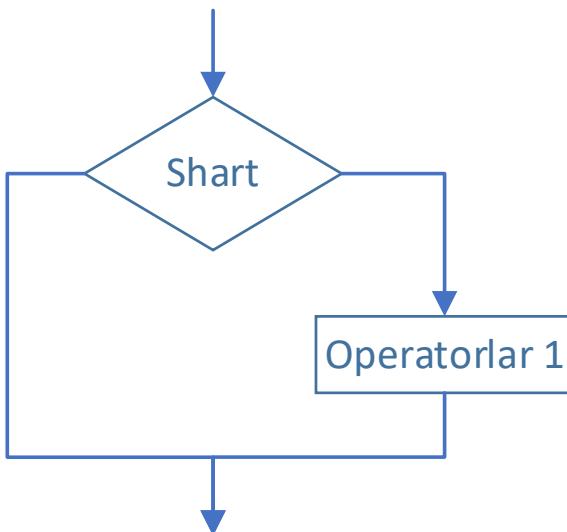
1 #include <iostream>
2 using namespace std;
3 int main() {
4   double x, y;
5   cout<<"x=";
6   cin>>x;
7   if (x >= 0) {
8     y = x * x;
9   }
10  else {
11    y = 2 * x;
12  }
13  cout<<"y="<<y;
14 }

```

x=5  
y=25

To‘liqmas tarmoqlanuvchi algoritmda shart bajarilganda bu shartga bog‘liq amallar bajariladi, bajarilmagan holatda hech qanday amal bajarish shart emas.

C++ da to‘liqmas tarmoqlanuvchida faqat **if** operatori ishlataladi, **else** ishlatilmaydi.



### 1.2-rasm. To‘liq bo‘limgan tarmoqlanuvchi algoritm bloksxemasi

**Misol 2.**  $a$  va  $b$  sonlari berilgan. Ulardan kattasini topuvchi dastur tuzing.

**Yechimi:** Dastavval  $a$  sonni maksimal deb tasavvur qilamiz. Agar  $b$  soni undan katta bo‘lsa u holda  $b$  soni maksimal bo‘ladi.

<pre> 1 #include &lt;iostream&gt; 2 using namespace std; 3 int main() { 4     double a, b; 5     cout&lt;&lt;"Birinchi sonni kirititing: "; 6     cin&gt;&gt;a; 7     cout&lt;&lt;"Ikkinchchi sonni kirititing: "; 8     cin&gt;&gt;b; 9     double max = a; 10    if (b &gt; max) 11        max = b; 12    cout&lt;&lt;a&lt;&lt;" va "&lt;&lt;b&lt;&lt;" sonlarining maksimali "&lt;&lt;max&lt;&lt;" ga teng"; 13 }   </pre>	<pre> Birinchi sonni kirititing: 3 Ikkinchchi sonni kirititing: 5 3 va 5 sonlarining maksimali 5 ga teng   </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------

**Ternar operatori.** Odatda ternar operatori shartni tekshirish natijasida bajarilishi kerak bo‘lgan shart va kod juda oddiy bo‘lgan hollarda qo‘llaniladi. Masalan, foydalanuvchidan dasturda ishlashni davom ettirishni yoki undan chiqishni xohlashini so‘rang. Sintaksis quyidagicha:

**shart ? birinchi buyruq : ikkinchi buyruq;**

Birinchi navbatda, biz kerakli shartni yozib undan so‘ng savol belgisini qo‘yishimiz kerak. Bundan tashqari, xuddi shu satrda, savol belgisidan so‘ng, agar shart haqiqatni qaytarsa (rost) bajariladigan birinchi oddiy buyruqni (kodni) yozamiz. Ushbu buyruqdan so‘ng : ni quyib ikkinchi buyruqni (kodni) yozamiz. Yo‘g‘on ichakdan keyin bu ikkinchi buyruq, agar shart noto‘g‘ri (noto‘g‘ri) qaytsa, bajariladi. Qisqacha qilib aytganda if va else operatorlarini qisqacha shaklda ? va : belgilari orqali yozish mumkin.

**Misol 3.** n natural soni berilgan. Agar u toq bo‘lsa "odd", juft bo‘lsa "even" so‘zini chiqaruvchi dastur tuzing.

**Yechimi:** n natural soni toq bo‘lishi uchun uni ikkiga bo‘lganda qoldiq 1 ga teng bo‘lishi kerak, aks holda juft bo‘ladi.

```

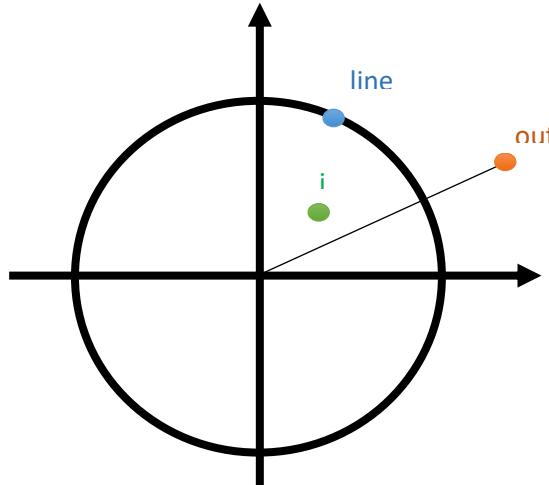
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int n;
5     cin >> n;
6     n % 2==1 ? cout<<"odd" : cout<<"even";
7 }
```

4  
even

### Murakkab tarmoqlanuvchi.

Agar biror shart asosida tarmoqlangandan so‘ng yana shart asosida tarmoqlansa (ya’ni **else if**), bunday tarmoqlanish murakkab tarmoqlanish deyiladi.

**Misol 4.** Markazi koordinatalar boshida va radiyusi R ga teng bo‘lgan aylana berilgan. Tekislikdagi (x,y) nuqta bu aylanaga tegishlilagini aniqlang. Agar aylana tashqarisida yotsa "out", chizig‘ida yotsa "line", ichida yotsa "in" so‘zini chiqaring.



**Yechimi:** Berilgan nuqatdan koordinata boshigacha masofani topamiz. Qaysi holat bo‘lishi bu masofaga bo‘g‘liq. **Masofa  $d$**  ga teng. Agar  $d > R$  bo‘lsa u holda nuqta aylanadan tashqarda, agar  $d < R$  bo‘lsa u holda nuqta aylanaga tegishli, aks holda aylana chizig‘ida yotadi.

```

1 #include <iostream>
2 using namespace std;
3 int main() {
4     int x, y, R;
5     cout<<"x=";
6     cin>>x;
7     cout<<"y=";
8     cin>>y;
9     cout<<"R=";
10    cin>>R;
11    if (x*x+y*y > R*R) {
12        cout<<"out";
13    }
14    else if (x*x+y*y==R*R) {
15        cout<<"line";
16    }
17    else {
18        cout<<"in";
19    }
20 }
```

x=2  
y=5  
R=70  
in

### C++ da murakkabroq shartlarni yozish.

Murakkab shart sodda shartlarning konyuksiya, dizyunksiya va inkorlaridan tashkil topadi.

Berilgan sonning  $[a, b]$  intervalga tegishli ekanligini aniqlash uchun,  $x \geq a$  va  $x \leq b$  shartlari bir vaqtning o‘zida o‘rinli bo‘lishi kerak. Shartlarning ikkalasi ham bajarilish shartini **&&** (va - and) amali orqali yozamiz:

```

If (x >= a && x <= b)
    cout<<"Tegishli";
```

```

else
```

```
    cout<<"Tegishli emas";
```

Berilgan sonning  $[a, b]$  intervalga tegishli emasligini aniqlash uchun,  $x < a$  yoki  $x > b$  shartlari istalgan biri bajarilishi kerak. Shartlarning istalgan biri bajarilishi yetarliligi shartini **||** (yoki - or) amali orqali yozamiz:

```
if (x < a || x > b)
```

```

cout<<"Tegishli emas";  

else  

cout<<"Tegishli";
```

**Tanlash operatori.** Tanlash operatori switch tanlanuvchi ifoda qiymatini bir nechta konstantalar bilan taqqoslab chiqadi. switch case ko‘plik tanlov operatori hisoblanadi. switch da ko‘rsatilgan ifoda qiymati case so‘zidan keyin yozilgan har bir qiymat bilan taqqoslab chiqiladi. Taqqoslanuvchi qiymat qaysidir qatordagi case operatoridan yozilgan qiymatga teng u holda uning davomida yozilgan amallar bajariladi.

**Misol 5.** Hafta kuni raqamda berilgan. Uni so‘zda chiqaruvchi dastur tuzing.

**Yechimi:**

<pre> 1 <b>#include</b> &lt;iostream&gt; 2 <b>using namespace</b> std; 3 <b>- int</b> main() { 4     <b>int</b> n; 5     cout&lt;&lt;"Hafta kunini raqamda kirititing: "; 6     cin&gt;&gt;n; 7     <b>switch</b> (n) { 8         <b>case</b> 1: cout&lt;&lt;"Dushanba"; <b>break</b>; 9         <b>case</b> 2: cout&lt;&lt;"Seshanba"; <b>break</b>; 10        <b>case</b> 3: cout&lt;&lt;"Chorshanba"; <b>break</b>; 11        <b>case</b> 4: cout&lt;&lt;"Payshanba"; <b>break</b>; 12        <b>case</b> 5: cout&lt;&lt;"Juma"; <b>break</b>; 13        <b>case</b> 6: cout&lt;&lt;"Shanba"; <b>break</b>; 14        <b>case</b> 7: cout&lt;&lt;"Yakshanba"; <b>break</b>; 15        <b>default</b>: cout&lt;&lt;"Hato kiritildi"; <b>break</b>; 16    } 17 }</pre>	Hafta kunini raqamda kirititing: 3 Chorshanba
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------

Agar har bir qatordan so‘ng **break** yozilmasa u holda qaysidir shart bajariladigan bo‘lsa keyingi break operatori kelgunga qadar barcha holatdagi amallar bajariladi.

Masalan quyidagi dasturda

```

switch (n) {  

case 1: cout<<"Dushanba";  

case 2: cout<<"Seshanba";  

case 3: cout<<"Chorshanba";  

case 4: cout<<"Payshanba";
```

```

case 5: cout<<"Juma"; break;
case 6: cout<<"Shanba"; break;
case 7: cout<<"Yakshanba"; break;
default: cout<<"Hato kiritildi"; break;
}
}

```

agar n=2 bo'lsa u holda ekranga SeshanbaChorshanbaPayshanbaJuma lar chiqadi.

### Shartsiz o'tish operatori.

Shartsiz o'tish operatorining umumiyligi ko'rinishi quyidagicha:

**Goto** <metka>;

**Goto** operatoridan keyin boshqarilish <metka> ga uzatiladi va dasturning bajarilishi shu yerdan davom etadi.

Metkani nishon ham deb aytildi bu davomida': qo'yilgan identifikator.

Misol uchun: metka: ;

Metka har qanday operator oldidan ishlatalishi mumkin, shuningdek shart operatori oldidan ham.

Misol 5. N natural sonini kiritishni taklif qiluvchi dastur tuzilsin. Agar natural bo'limgan son kiritilsa, qayta kiritish taklif qilinsin.

<pre> 1 <b>#include</b> &lt;iostream&gt; 2 <b>#include</b> &lt;math.h&gt; 3 <b>using namespace</b> std; 4 <b>int</b> main() 5 { 6 <b>float</b> n; 7 nishon: 8 cout &lt;&lt; " natural son kirititing: " &lt;&lt; endl; 9 cin &gt;&gt; n; 10 <b>if</b> ((ceil(n) != n)    (n &lt;= 0)) 11 <b>goto</b> nishon; 12 cout &lt;&lt; " natural son kiritildi!" &lt;&lt; endl; 13 <b>return</b> 0; 14 } </pre>	<pre> natural son kirititing: 12.2 natural son kirititing: 12 natural son kiritildi! </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------

Odatta shartsiz o'tish operatoridan foydalanmaslik tavsiya qilinadi. Chunki shartsiz o'tish operatori dasturning strukturasini buzishi mumkin. Shartsiz o'tish

operatori orqali qism dasturga, sikllarga o‘tib bo‘lmaydi va shu kabi holatlarda etiborli bo‘lish talab qilinadi.

### **Nazorat savollari**

1. Tarmoqlanish operatoridan qachon foydalaniladi?
2. Tanlash operatorining vazifasi nimadan iborat?
3. Ternar operator deganda nimani tushunasiz?
4. Shartsiz o‘tish operatorining ishlash tartibini tushuntiring.
5. Agar tanlash operatorining har bir qatordan so‘ng **break** yozilmasa qanday holat kuzatiladi?
6. C++ da murakkabroq shartlarni yozish deganda nimani tushunasiz
7. Murakkab tarmoqlanuvchi deganda nima tushuniladi?
8. Tarmoqlanuvchi algoritm qanday turlarga bo‘linadi?

## **1.4-§. TAKRORLANISH OPERATORLARI**

### **Reja:**

1. **Parametrli takrorlash operatori (for).**
2. **Old shartli va so‘ng shartli takrorlanuvchi operatorlari (while, do while).**

Dasturlashda qator masalalarni yechish uchun ko‘pincha bitta amalni bir necha marotaba bajarish talab qilinadi. Amaliyotda bu rekursiyalar va iterativ algoritmlar yordamida amalga oshiriladi. Iterativ jarayonlar – bu amallar ketma-ketligini zaruriy sonda takrorlanishidir. Takrorlanuvchi algoritmla dasturlarda aniq bir yoki bir necha amallar takror va takror bajarilish imkoniyati ko‘zda tutilgan bo‘ladi. Takrorlanishni amalga oshirilishi uchun dasturlash tilining takrorlash operatorlaridan foydalanish mumkin bo‘ladi. C++ dasturlash tilida takrorlash operatorlarining bir necha turi mavjud. Takrorlash operatorlari “takrorlash sharti” deb nomlanuvchi ifodaning rost qiymatida dasturning ma‘lum bir qismidagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajaradi.

Takrorlash o‘zining kirish va chiqish nuqtalariga ega, lekin chiqish nuqtasi bo‘lmasligi mumkin, bunday takrorlashlarga cheksiz takrorlash deyladi. Cheksiz takrorlash uchun takrorlashni davom ettirish sharti doimo rost bo‘ladi. Takrorlash shartini tekshirish takrorlash tanasidagi operatorlarni bajarishda oldin tekshirilishi mumkin (for, while operatolari) yoki tanasidagi operatorlar bir marta bajarilgandan keyin tekshirilishi mumkin (do-while operatori).

### **Takrorlanish jarayonlari.**

Takrorlanish – bu bir xil ketma-ketlikda bajariladigan ko‘p qirrali harakat.

Ma‘lum qadamlar sonidagi takrorlanish va Noma‘lum qadamlar sonidagi takrorlanish (shartli takrorlanish).

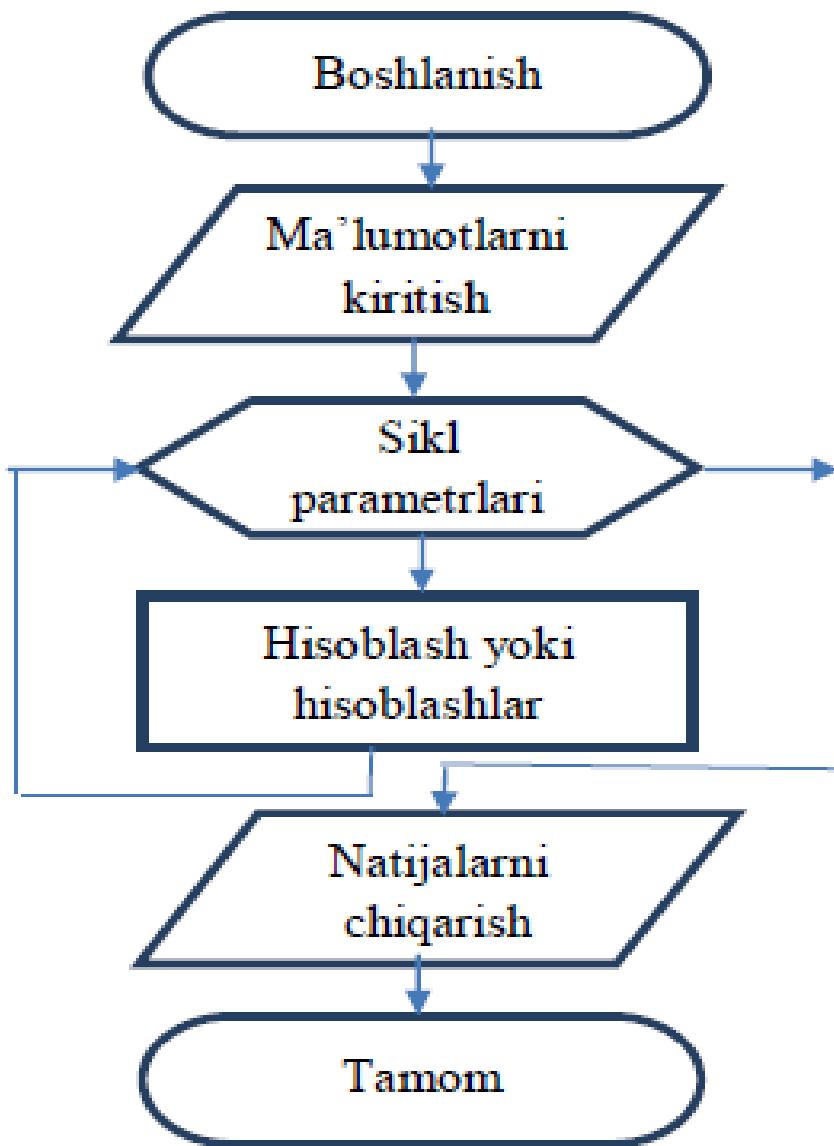
**Masala.** 1 dan 8 gacha (a dan b gacha) butun sonlarning kvadratlarini va kublarini ekranga chiqaring.

**Xossa:** bir xil harakatlar 8 marta bajariladi.

Takrorlanuvchi jarayonga misol: Avval berilgan ma’lumotlar kiritiladi. So‘ngra takrorlanuvch jarayonning, ya‘ni siklning parametrlari o‘rnatiladi. Buni matematikada takrorlanish opalig‘i deb ham yuritiladi.

Masalan: X soni  $[0;10]$  bo‘lsa, sikl parametrlari 0 dan 10 gacha hisoblanadi. Keyin hisoblash yoki bir nech hisoblashlar amalga oshiriladi. Natija 1 ta yoki bir nechta chiqishi mumkin, bu masalaning qo‘yilishiga bog‘liq bo‘ladi. Agar masalaning javobi bir nechta chiqadigan bo‘lsa, u holda chiqarish blogi ham sikl parametri ichida bo‘ladi.

### **for-parametrlı takrorlash operatori.**



### 1.3 – rasm. for takrorlash operatorining blok-sxemasi

For takrorlash operatorining sintaksisi quyidagi ko‘rinishga ega:

```

for (<ifoda1>; <ifoda2>;<ifoda3>)
    <operator yoki blok>;
  
```

Bu operator o‘z ishini <ifoda1> ifodasini bajarishdan boshlaydi. Keyin takrorlash qadamlari boshlanadi. Har bir qadamda <ifoda2> bajariladi, agar natija 0 qiymatidan farqli yoki true bo‘lsa, takrorlash tanasi - <operator yoki blok> bajariladi va oxirida <ifoda3> bajariladi. Agar <ifoda2> qiymati 0 (false) bo‘lsa, takrorlash jarayoni to‘xtaydi va boshqaruv takrorlash operatoridan keyingi operatorga o‘tadi. Shuni qayd etish kerakki, <ifoda2> ifodasi vergul bilan

ajratilgan bir nechta ifodalar birlashmasidan iborat bo‘lishi mumkin, bu holda oxirgi ifoda qiymati takrorlash sharti hisoblanadi. Takrorlash tanasi sifatida bitta operator, jumladan bo‘sh operator bo‘lishi yoki operatorlar bloki kelishi mumkin.

Misol uchun 10 dan 20 gacha bo‘lgan butun sonlar yig‘indisini hisoblash masalasini ko‘raylik.

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int Summa=0;
5     for ( int i= 10 ; i<= 20 ; i++ )
6         Summa+=i;
7     cout<< " Yig‘indi= " << Summa;
8 }
```

Yig‘indi= 165  
...Program finished with exit code 0  
Press ENTER to exit console.

Dasturdagi takrorlash operatori o‘z ishini,  $i$  takrorlash parametriga (takrorlash hisoblagichiga) boshlang‘ich qiymat - 10 sonini berishdan boshlaydi va har bir takrorlash qadamidan (iteratsiyadan) keyin qavs ichidagi uchinchi operator bajarilishi hisobiga uning qiymati bittaga oshadi. Har bir takrorlash qadamida takrorlash tanasidagi operator bajariladi, ya‘ni *Summa* o‘zgaruvchisiga  $i$  ning qiymati qo‘shiladi. Takrorlash sanagichi  $i$  ning qiymati 21 bo‘lganda “ $i<=20$ ” takrorlash sharti *false(0)*qiymatini qaytaradi va takrorlash tugaydi. Natijada boshqaruv takrorlash operatoridan keyingi *cout* operatoriga o‘tadi va ekranga yig‘indi chop etiladi. Yuqorida keltirilgan misolga qarab takrorlash operatorlarining qavs ichidagi ifodalariga izoh berish mumkin: <ifoda1> - takrorlash sanagichi vazifasini bajaruvchi o‘zgaruvchiga boshlang‘ich qiymat berishga xizmat qiladi va u takrorlash jarayoni boshida faqat bir marta hisoblanadi. Ifodada o‘zgaruvchi e‘loni uchrashi mumkin va bu o‘zgaruvchi takrorlash operatori tanasida amal qiladi va takrorlash operatoridan tashqarida «ko‘rinmaydi», <ifoda2> - takrorlashni bajarishni yoki bajarilmasligini aniqlab beruvchi mantiqiy ifoda, agar shart rost bo‘lsa, takrorlash davom etadi, aks holda yo‘q. Agar bu ifoda bo‘sh bo‘lsa, shart doimo rost deb hisoblanadi; <ifoda3> - odatda takrorlash sanagichining qiymatini oshirish (kamaytirish) uchun xizmat

qiladi yoki unda takrorlash shartiga ta'sir qiluvchi boshqa amallar bo'lishi mumkin.

Takrorlash operatori tanasi sifatida operatorlar bloki ishlatalishini faktorialni hisoblash misolida ko'rsatish mumkin:

**Misol.** Faktorialni hisoblash dasturi

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4 int n,i;
5 long long fact=1;
6 cout<<"n ni kirititing:";
7 cin>>n;
8 for(i=1; i<=n; i++)
9 fact*=i;
10 cout<<"Natija="<<fact;}
```

n ni kirititing:7  
Natija=5040

**Ichma-ich joylashgan for takrorlanish operatori.**

**Misol:** Takrorlash operatorining ichma-ich joylashuviga misol sifatida 20 gacha bo'lgan sonlarning tub son yoki murakkab son ekanligi haqidagi ma'lumotni chop qilish masalasini ko'rishimiz mumkin:

```
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int main(){
5 const int m=20;
6 int n[m];
7 int i,j,f;
8 for(i=0; i<=m; i++)
9 n[i]=1;
10 for(i=2; i<=m/2; i++) {
11 if (n[i]==1){
12 for(j=i+1; j<=m; j++)
13 if (n[j]==1)
14 if (j%i==0)
15 n[j]=0; } }
16 for(i=2; i<=m; i++) {
17 if (n[i]==1)
18 cout<<i<<"=Tub son"<<endl;
19 else
20 cout<<i<<"=Murakkab son"<<endl;
21 } }
```

2=Tub son  
3=Tub son  
4=Murakkab son  
5=Tub son  
6=Murakkab son  
7=Tub son  
8=Murakkab son  
9=Murakkab son  
10=Murakkab son  
11=Tub son  
12=Murakkab son  
13=Tub son  
14=Murakkab son  
15=Murakkab son  
16=Murakkab son  
17=Tub son  
18=Murakkab son  
19=Tub son  
20=Murakkab son

Agar takrorlash jarayonida bir nechta o'zgaruvchilarning qiymati sinxron ravishda o'zgarishi kerak bo'lsa, takrorlash ifodalarida zarur operatorlarni ',' bilan yozish orqali bunga erishish mumkin:

```
for(int i=10; j=2; i<=20; i++; j=i+10) {...};
```

Takrorlash operatorining har bir qadamida  $j$  va  $i$  o‘zgaruvchilarning qiymatlari mos ravishda o‘zgarib boradi.

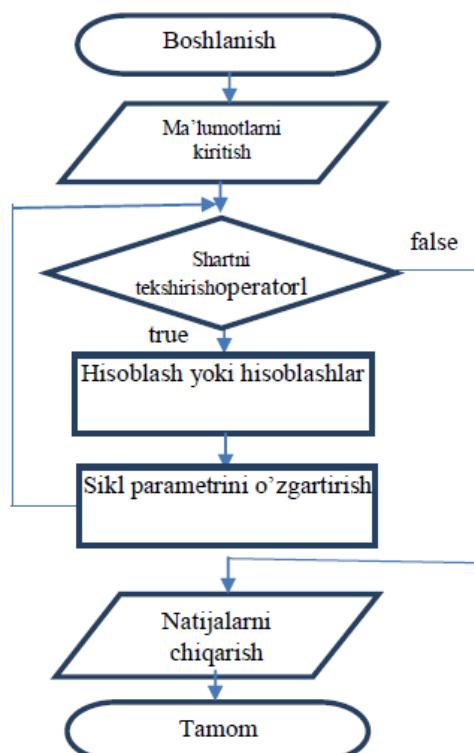
**Xossa:**

- Shart takrorlashning keyingi qadami boshlanishidan oldin tekshiriladi, agar u yolg‘on bo‘lsa takrorlash bajarilmaydi;
- o‘zgartirish (sarlavhaning uchinchi qismi) takrorlashning navbatdagi qadamina oxirida bajariladi;
- Agar shart yolg‘on bo‘lmasa takrorlash to‘xtovsiz ishlashi mumkin (sikl ichiga tushib qoladi).

**While takrorlash operatori.** **while** takrorlash operatori, operator yoki blokni takrorlash sharti yolg‘on (false yoki 0) bo‘lguncha takror bajaradi. U quyidagi sintaksisiga ega:

**while (<ifoda>) <operator yoki blok>;**

Agar **<ifoda>** rost qiymatli o‘zgarmas ifoda bo‘lsa, takrorlash cheksiz bo‘ladi. Xuddi shunday, **<ifoda>** takrorlash boshlanishida rost bo‘lib, uning qiymatiga takrorlash tanasidagi hisoblash ta‘sir etmasa, ya‘ni uning qiymati o‘zgarmasa, takrorlash cheksiz bo‘ladi.



1.4 – rasm. While takrorlash operatorining blok-sxemasi

**while** takrorlash shartini oldindan tekshiruvchi takrorlash operatori hisoblanadi. Agar takrorlash boshida <ifoda> yolg‘on bo‘lsa, **while** operatori tarkibidagi <operator yoki blok> qismi bajarilmasdan cheklab o‘tiladi.

Ayrim hollarda <ifoda> qiymat berish operatori ko‘rinishida kelishi mumkin. Bunda qiymat berish amali bajariladi va natija **0** bilan solishtiriladi. Natija noldan farqli bo‘lsa, takrorlash davom ettiriladi.

Agar rost ifodaning qiymati noldan farqli o‘zgarmas bo‘lsa, cheksiz takrorlash ro‘y beradi.

### Masalan:

**while(1); // cheksiz takrorlash**

Xuddi **for** operatoridek, ‘;’ yordamida <ifoda> da bir nechta amallar sinxron ravishda bajarish mumkin.

**Misol.** Son va uning kvadratlarini chop qilinadigan dasturda ushbu holat ko‘rsatilgan:

<pre>1 #include &lt;iostream&gt; 2 using namespace std; 3 int main(){ 4 int n,n2; 5 cout&lt;&lt;"Sonni kiritning (1..10)="; 6 cin&gt;&gt;n; 7 n++; 8 while(n--,n2=n*n,n&gt;0) 9 cout&lt;&lt; " n soni = "&lt;&lt;n&lt;&lt; " sonning kvadrati="&lt;&lt;n2&lt;&lt;endl; }</pre>	Sonni kiritning (1..10)=4 n soni = 4 sonning kvadrati=16 n soni = 3 sonning kvadrati=9 n soni = 2 sonning kvadrati=4 n soni = 1 sonning kvadrati=1
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

Dasturdagi takrorlash operatori bajarilishi n soni 1 gacha kamayib boradi. Har bir qadamda n va uning kvadrati chop qilinadi. Shunga e‘tibor berish kerakki, shart ifodasida operatorlarni yozilish ketma-ketligining ahamiyati bor, chunki eng oxirgi operator takrorlash sharti sifatida qaraladi va n qiymati 0 bo‘lganda takrorlash tugaydi.

**Misol:** Ixtiyoriy natural sonlar kiritiladi, qachonki char tipidagi biron belgi kiritilguncha va kiritilgan sonlar yig‘indisi hisoblanadi.

```
#include <iostream>
#include <string.h>
using namespace std;
```

```

int main()
{
    int a, ans;
    char s;
    cin >> a;
    ans = a;
    while(cin >> s >> a)
    {
        ans += a;
    }
    cout << ans;
}

```

while takrorlash operatori yordamida samarali dastur kodi yozishga yana bir misol bu - ikkita natural sonlarning eng katta umumiy buluvchisini (EKUB) Yevklid algoritmi bilan topish masalasini keltirishimiz mumkin:

```

1 #include <iostream>
2 #include <stdio.h>
3 using namespace std;
4 int main (){
5
6
7
8     int a, b;
9     cout<< " A va B natural sonlar EKUBini topish \n" ;
10    cout<< " A va B natural sonlarni kiriting : " ;
11    cin >> a >> b ;
12    while ( a != b ) a > b ? a -= b : b -= a ;
13    cout<< " Bu sonlar EKUBi= "<< a ;
14 }

```

```

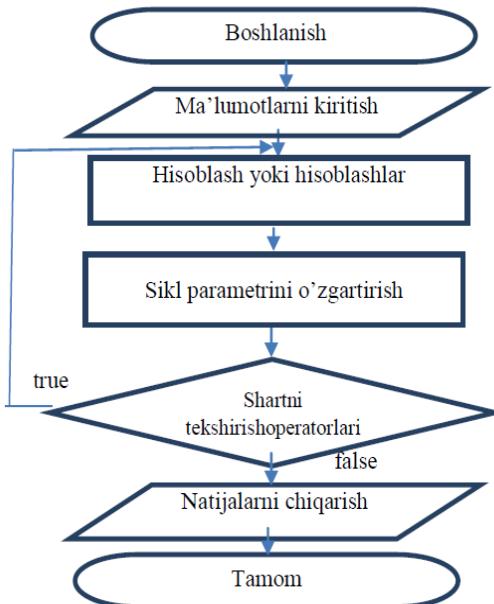
A va B natural sonlar EKUBini topish
A va B natural sonlarni kiriting : 2
8
Bu sonlar EKUBi= 2

```

**do <operator yoki blok>; while (<ifoda>);**

**do-while takrorlash operatori.** do-while takrorlash operatori while operatoridan farqli ravishda oldin operator yoki blokni bajaradi, keyin takrorlash shartini tekshiradi. Bu qurilma takrorlash tanasini kamida bir marta bajarilishini ta‘minlaydi.

Bunday takrorlash operatorining keng qo‘llaniladigan holatlari - bu takrorlashni boshlamasdan turib, takrorlash shartini tekshirishning iloji bo‘lmagan holatlar hisoblanadi.



1.5-rasm. do-while takrorlash operatorining blok-sxemasi.

Misol: **do-while** takrorlash operatoridan foydalanib x ning qiymatlarini ekranga chiqarish.

```

1 #include <iostream>
2 using namespace std;
3 int main() {
4     int x = 1;
5     do {
6         cout << "X=" << x << endl;
7         x = x + 1;
8     } while (x < 5);
9 }
```

X=1  
X=2  
X=3  
X=4

**O'tish operatorlari, break operatori.** Ba'zi hollarda takrorlash bajarilishini ixtiyoriy joyda to'xtatishga to'g'ri keladi. Bu vazifani **break** operatori bajarishga imkon beradi. Bu operator darhol sikl bajarilishini to'xtatadi va boshqaruvni sikldan keyingi operatorlarga uzatadi. **Break** operatorini takrorlash operatori tanasining ixtiyoriy (zarur) joylariga qo'yish orqali shu joylardan takrorlashdan chiqishni amalga oshirish mumkin.

**continue operatori.** Takrorlanish bajarilishiga ta'sir o'tkazishga imkon beradigan yana bir operator **continue** operatoridir. Bu operator takrorlanish qadamining bajarilishini to'xtatib, for va while da ko'rsatilgan shartli tekshirishga o'tkazadi.

Biz C++ dasturlash tilida 3 xil ko‘rinishdagi takrorlash operatorlarni borligini ko‘rib chiqdik. Bular for, while, do-while. Ular funksional nuqtaiy nazardan bir ish bajaradi, ya‘ni ularni funksiyasi ma‘lum bir amalni (yoki amallarni) ketma-ket bir necha marta takrorlashdan iborat. Ilmiy tilda siklik takrorlanishni «*iteratsiya*» deb ataladi.

Ammo bu operatorlar mazkur funksiyani (ya‘ni takrorlash jarayonini tasiflashni) turlicha amalga oshiradi. Bu operatorlarining asosiy farqi quydagilardan iborat:

- for operatori faqat iteratsiyalar (ya‘ni takrorlashlar) soni ma‘lum bo‘lgan holda, while operator esa iteratsiyalar soni noma‘lum bo‘lgan holda ham ishlaydi.
- do-while operatori ham xuddi while operatori kabi iteratsiyalar soni noma‘lum bo‘lgan holda ham ishlaydi. Ammo u bir iteratsiyadan so‘ng “shart”ni tekshiradi, while operatori esa avval “shart”ni tekshiradi va so‘ngra birinchi iteratsiyani amalga oshiradi.
- Parametrli takrorlash for operatorini parametrning boshlang‘ich va oxirgi qiymati hamda o‘zgarish qadami aniq bo‘lganda qo‘llash juda qulay.

### **Nazorat savollari**

1. C++ dasturlash tilida takrorlash operatorlari nima uchun kerak?
2. *while* operatorining umumiyo ko‘rinishi. *do while* operatori *while* operatoridan qanday farq qiladi?
3. *for* operatorining umumiyo ko‘rinishi. Ichki sikllar qanday tashkil etiladi?
4. For operatorining takrorlanishini tugashi uning qaysi qismiga bo‘liq?
5. While operatori qanday ishlaydi? Qahday holatda boshqaruv while opertoridan keyingi operatorga uzatiladi?
6. Qahday holatda while opertori tanasidagi amallar ketma-ketligi bajariladi va yana shart tekshirishga qaytiladi?
7. *for* bilan *while* va *do-while* operatorlarining qanday farqli tomonlari bor?
8. Takrorlanishlar algoritmlarda qanday ko‘rinishda bo‘ladi?
9. *while* operatorida qachon cheksiz takrorlash ro‘y beradi?
10. Takrorlanishlar ichma-ich bo‘lishi mumkinmi?
11. *do-while* operatori qanday ishlatiladi?

## 1.5-§. FUNKSIYALAR

**Reja:**

- 1. Funksiya tavsifi. Qiymatlarni qaytarish. Funksiya prototiplari.**
- 2. Parametrlarni qiymat va adresga ko‘ra jo‘natish.**
- 3. Qiymat berish parametrlari. Funksiyalar parametr sifatida.**
- 4. Rekursiv funksiyalar. Funksiyalarni qayta yuklash.**
- 5. Foydalanuvchi kutubxonasini tashkil etish.**

C++da funksiya faqat chaqirilgan vaqtida ishlaydigan kod blogi hisoblanadi. Funksiyaga parametrlar sifatida ma’lumotlarni uzatish mumkin. Funksiyalar muayyan bir vazifani bajarish uchun ishlatiladi. Dasturchi kod yozish jarayonida yaratgan funksiyadan bir yoki bir nechta marta ishlatish yoki umuman ishlatmaslik imkoniyatiga ega.

**Funksiya yaratish.** **main()** ham bir funksiya hisoblanadi. Dastur ishga tushgan vaqtida birinchi bo‘lib ushbu funksiya ishlaydi. Bu standart funksiya bo‘lib foydalanuvchi bu kabi funksiyalarini yaratishi mumkin.

Funksiya yaratish uchun birinchi funksiya qaytaradigan tip yoki funksiya turi keyin nomi va () ochiladi. Qavs ichida ma’lumotlar ya’ni parametrlar qabul qilinadi.

```
void myFunction() {  
    // Bu qiymat qaytarmaydigan funksiya  
}
```

Bu dastur kodida **myFunction()** funksiya nomi, **void** funksiya qiymat qaytarmaydigan funksiyaligidan dalolat beradi. {} bu funksiya tanasi bo‘lib. Bu yerda kerakli kodni kiritish mumkin. **main()** funksiyasida nima amal bajarilgan bo‘lsa bunda ham shunday amalni bajarish mumkin.

**Funksiya chaqirish.** Funksiya o‘zidan o‘zi ishlab ketmaydi. Funksiyani chaqirishdan so‘ng ishga tushadi. Funksiyadan qayta qayta foydalanish mumkin. Funksiyani chaqirish uchun funksiya nomi ikki qavs () va nuqtali vergul bilan tugallanadi.

```

#include <iostream>
using namespace std;
void myFunction() {
    cout << "I'm a student!";
}
int main() {
    myFunction();
}

```

Dasturda funksiya ishlatiladigan bo'lsa, uni albatta e'lon qilish shart. Funksiyani e'lon qilishda uning tipi, nomi va qaytaradigan parametrlari haqida xabar beriladi. Dasturda biror funksiyani oldindan e'lon qilmasdan turib uni chaqirish mumkin emas. Funksiyani asosiy funksiya main( ) dan oldin va keyin aniqlanishi mumkin. Agar funksiya asosiy funksiyadan oldin aniqlansa, u aniqlanishi bilan birga e'lon qilingan deb hisoblanadi va uni alohida main( ) ichida e'lon qilish shart bo'lmay qoladi. Agar funksiya asosiy funksiyadan keyin aniqlanayotgan bo'lsa, uni main( ) ichida albatta e'lon qilish shart bo'ladi. Funksiyani main( ) ichida e'lon qilinadigan bo'lsa, uning nomi bilan birga ishlatiladigan parametrlarining faqatgina tiplari ko'rsatilishi ham mumkin.

Funksiyalarga murojaat qilinganida uning uzatiladigan parametrlariga alohida e'tibor berish kerak. Parametrlarning uzatilishi quyidagi bosqichlardan iborat:

➤ Funksiyani tashkil etadigan rasmiy parametrlar uchun xotiradan joy ajratiladi. Agar parametrlar haqiqiy tipga ega bo'lsa, ular double tipga, agar char va short int tipli bo'lsalar, ular int tipi sifatida tashkil etiladilar. Agar parametrlar massiv shaklida bo'lsalar, massiv boshiga ko'rsatkich qo'yiladi va u funksiya tanasi ichida massiv parametr bo'lib xizmat qiladi.

➤ Funksiya chaqirilganida kerak bo'ladigan ifodalar yoki haqiqiy parametrlar aniqlanadi va ular rasmiy parametrlar uchun ajratilgan joyga yoziladi;

➤ Funksiya chaqiriladi va aniqlangan haqiqiy parametrlar yordamida hisoblanadi. Bu yerda ham agar parametrlar haqiqiy tipga ega bo'lsa, ular double tipga, agar char va short int tipi bo'lsalar, ular int tipi sifatida tashkil etiladilar.

➤ Natija funksiya chaqirilgan joyga qaytariladi.

➤ Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo'shatiladi.

**Funksiya** - bu dasturning istalgan joyida cheksiz ko'p marta foydalanish mumkin bo'lgan kod blokidir.

Masalan, quyidagi dasturda biz 2 qatorni (funktsiyalardan foydalanmasdan) chop etamiz:

```
#include <iostream>
using namespace std;
int main() {
    cout << "Funktsiya dasturlashda juda yaxshi vositadir";
    cout << "Siz undan dasturlash darajangizni yaxshilash uchun
foydalanishingiz mumkin";
}
```

Funksiyadan foydalanib xuddi shunday natijani quyidagi dastur kodi yordamida olishimiz mumkin:

```
#include <iostream>
using namespace std;
void func () { // funksiya
    cout << "Funktsiya dasturlashda juda yaxshi vositadir";
    cout << "Siz undan dasturlash darajangizni yaxshilash uchun
foydalanishingiz mumkin";
}
int main() {
    func(); // funksiyani chaqirish
}
```

Pechat qilinishi kerak bo‘lgan ikki qatorni 5 marta chop etishda birinchi dasturdagi satrlar sonining ko‘payishiga diqqat bilan qarashimiz kerak.

Bu dasturdan ko‘rinib turibdiki funktsiyalardan to‘g‘ri foydalanilsa, dastur kodini bir necha marta qisqartirishingiz mumkin. Ammo esda tutishingiz kerak - hech qanday sababsiz funktsiyalardan foydalanishning ma’nosi yo‘q (masalan, funktsiya ichidagi mantiq juda aniq bo‘lsa).

Funktsiya argumentlari. Funktsiyaga argumentlar qavslar ichida (funktsiya nomidan keyin) ko‘rsatilishi mumkin. Funksiya argumenti funktsiya chaqirilganda unga uzatiladigan qiymatdir. Agar funktsiya argumenti bir nechta bo‘lsa ular vergul bilan ajratilishi kerak.

```
int sum(int b, int c) { // bu kodda b и c funktsiya argumenti.
```

Agar funktsiyaga argumentlar bo‘lmasa, u holda qavs ichida void turini yozish mumkin. Odatda voidni yozmasa ham hato xisoblanmaydi.

```
void stroka(void) {  
    cout << "Oddiy qator";  
}
```

Kod bloki. Figurali qavsdan keyin funktsiya chaqirilganda ishlaydigan dastur kodi yoziladi. Funktsiyadan biron bir qiymatni qaytarish uchun yoki uning ishlashini butunlay to‘xtatish uchun return buyrug‘idan foydalanish kerak bo‘ladi.

```
int sum(int b, int c) { // bu kodda b и c funktsiya argumenti.  
    return a + b; // a + b quymat qaytariladi  
}
```

C++ da funktsiyani e’lon qilmasdan turib uni chaqirib bo‘lmaydi. Buning sababi, kompilyator funktsiyaning to‘liq nomini (funktsiya nomi, argumentlar soni, argument turlari) bilmaydi. Shunday qilib, quyidagi misolda kompilyator bizga xato haqida xabar beradi:

```
int main() {  
    Sum_numbers(1, 2);  
    return 0;
```

```

    }

int Sum_numbers(int a, int b) {
    cout << a + b;
}

```

Funksiya prototipi bu funksiya bo‘lib unda kod bloki mavjud bo‘maydi.

Funksiya prototipi quyidagilardan iborat bo‘ladi:

- Funksiyaning to‘liq nomi;
- Funksiya qiymatini qaytarish tipi.

```

#include<iostream>
using namespace std;
int Max(int a, int b); // Funksiya prototipi

int main()
{
    int x,y;
    cin >> x >> y ;
    cout<<"\n max=" << Max(x,y) << endl ;
    return 0 ;
}

int Max(int a, int b) // Yordamchi
{ // Funksiya
    if (a > b )  b = a ;
    return b ;
}

```

Rekursiv funksiyalar bu o‘zini chaqiradigan funksiyalardir. Rekursiyali funksiyalarga q o‘yiladigan asosiy talab, qandaydir qiymatda rekursiya 0- yolg‘on yoki 1-rost qiymat qabul qilishi kerak. Shundagina chaqirilgan funksiyalar qaytadi. Aks holda funksiya o‘z – o‘zini davomli ravishda chaqiradi.

Misol:  $n!$  faktorialni rekursiv funksiya orqali hisoblochi dastur tuzilsin.

```

n!=1*2*...*(n- 1)*n
#include <iostream>
using namespace std;
int factorial(int);
int main() {
    int n, result;

```

```

cout << "Faktorial sonini kiriting: ";
cin >> n;
result = factorial(n);
cout << "Faktorial " << n << " = " << result;
return 0; }

int factorial(int n)
{ if (n > 1)
{ return n * factorial(n - 1); }
else { return 1; } }

```

Misol: Fibonachchi ketma ketligining n – hadini rekursiya qism dastur orqali hisoblovchi dastur

```

#include <iostream>

int fib(int);
int main()
{
int n;
cout << "n="; cin >> n;
cout << fib(n) << endl;
}

int fib(int k)
{
if (k == 0 || k == 1) return 1;
else return fib(k - 1) + fib(k - 2);
}

```

C++ da foydalanuvchi kutubxonasini tashkil etish uchun include katalogida dasturlash.h nomli sarlavha fayli yaratiladi. Yaratilgan dasturlash.h faylga kerakli funksiyalar kiritiladi.

Berilgan son raqamlari yig‘indisini hisoblovchi funksiya kodi quyidagicha:

```

int raqam_summa (int n)
{
    int s=0;
    while (n!=0)
    {
        s=s+n%10;
        n=n/10;
    }
    return s;
}

```

Dastur kodini dasturlash.h faylining ichiga yozamiz va saqlaymiz.

```

#include<iostream>

#include<dasturlash.h> // dasturlash.h fayli (kutubxonasiga) murojaat
using namespace std;

int main(){
    int ,son2;
    cin>>son1>>son2;
    cout<<son1<<" raqamlari yig'indisi-";
    cout<<raqam_summa(son1)<<endl;
    cout<<son2<<" raqamlari yig'indisi-";
    cout<<raqam_summa(son2)<<endl;
}

```

Natija quyidagicha buladi:

235
12
235 raqamlari yig'indisi-10
12 raqamlari yig'indisi-3

### **Nazorat savollari**

1. C++ da funksiya deganda nima tushuniladi?
2. Funksiya prototipi deganda nima tushuniladi?
3. Funksiyaning qiymat qaytarishini tushuntirib bering.
4. Rekursiv funksiya nima?
5. Foydalanuvchi kutubxonasi qanday tashkil etiladi?
6. C++ da foydalanuvchi kutubxonasini tashkil etish uchun qaysi kutubxonadan foydalaniadi?
7. Yordamchi funksiya deganda nima tushuniladi?
8. Funksiya argumenti nima?

## II-BOB. STATIK VA DINAMIK MASSIVLAR. SATRLAR VA KENGAYTIRILGAN BELGILAR

### 2.1-§ BIR O'LCHOVLI MASSIVLAR

**Reja:**

- **Statik massivlar.**
- **Massiv elementlarini saralash va qidirish usullari.**
- **Massivlar ustida turli xil amallar bajarish usullari.**

**Massiv** - bu bir tipli nomerlangan ma`lumotlar jamlanmasidir. Massiv indeksli o`zgaruvchi tushunchasiga mos keladi.

Massiv ta`riflanganda :

<tipi><massiv\_nomi>[elementlar\_soni]={boshlang`ich qiymat};

Masalan:

- *long int a[5];*
- *char w[200];*
- *double f[4][5][7];*
- *char[7][200].*

Massiv indekslar har doim 0 dan boshlanadi. C ++ tili standarti bo`yicha indekslar soni 31 tagacha bo`lishi mumkin, lekin amalda bir o'lchovli va ikki o'lchovli massivlar qo'llaniladi. Bir o'lchovli massivlarga matematikada vektor tushunchasi mos keladi.

long int a[5];					
Massiv elementlari	a[0]	a[1]	a[2]	a[3]	a[4]
qiymati	25	41	20	5	8

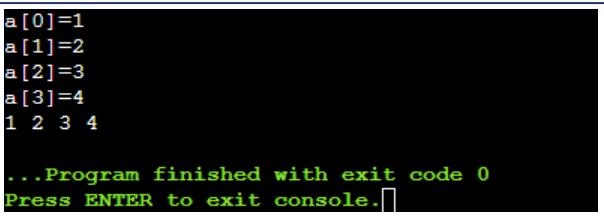
Massivning int z[3] shakldagi ta`rifi, int tipiga tegishli z[0],z[1],z[2] elementlardan iborat massivni aniqlaydi.

Massivlar ta`riflanganda initsializatsiya qilinishi, ya`ni boshlang`ich qiymatlarlari ko`rsatilishi mumkin. Masalan, *float C[]={1,-1,2,10,-12.5};* Bu

misolda massiv chegarasi avtomatik aniqlanadi. Agar massiv initsializatsiya qilinganda elementlar chegarasi ko‘rsatilgan bo‘lsa , ro‘yxatdagi elementlar soni bu chegaradan kam bo‘lishi mumkin, lekin ortiq bo‘lishi mumkin emas. Masalan,  $\text{int } A[5]=\{2,-2\}$ . Bu holda  $a[0]$  va  $a[1]$  qiymatlari aniqlangan bo‘lib, mos holda 2 va -2 ga teng.

**Masala:** Elementlari butun sonlardan iborat bo‘lgan, n elementdan tashkil topgan massiv elementlarini kirituvchi va ekranga chiqaruvchi dastur tuzing.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int n;
6     int a[n];
7     cout << "n="; cin >>n;
8     for (int i=0; i<n; i++)
9     {
10         cout << "a["<<i<<"]=";
11         cin >> a[i];
12     }
13     for (int i=0; i<n; i++){
14         cout << a[i]<< " ";
15     }
16     return 0;
17 }
```



```
a[0]=1
a[1]=2
a[2]=3
a[3]=4
1 2 3 4
...Program finished with exit code 0
Press ENTER to exit console.
```

**Masala:** n ta elementdan tashkil topgan massiv berilgan . Shu massiv elementlar yig‘indisini chiqaruvchi dastur tuzing.

```

1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     int n,sum=0;
7     cout <<"n="; cin >>n;
8     int a[n];
9     for (int i=0;i<n;i++)
10    {
11        cout <<"a["<<i<<"]=";
12        cin >>a[i];
13        sum+=a[i];
14    }
15    cout <<"Yig`indi:"<<sum<<endl;
16    return EXIT_SUCCESS;
17 }
18
19

```

**Masala:** n ta natural soni berilgan. Dastlabki n ta toq sondan tashkil topgan massivni hosil qiling va elementlarini chiqaring.

```

1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     int n;
7     cout <<"n="; cin >>n;
8     int a[n];
9     int b = 1;
10    for (int i=0; i < n; i++)
11    {
12        a[i] = b;
13        b = b + 2;
14    }
15    for(int i=0 ;i< n;i++)
16    {
17        cout <<"a["<<i<<"]=" << a[i]<< " " << endl;
18    }
19    system("PAUSE");
20    return EXIT_SUCCESS;
21 }
22

```

**Saralash**— bu massiv elementlarini tartiblash ( o'sish, kamayish, oxirgi raqami, bo'luvchilari bo'yicha, ...).

**Tanlash usuli.** Ushbu usul bilan saralashda ma'lumotlarning tartibga solingan ketma-ketligi xotiraning dastlabki ketma-ketlik joylashgan uchastkasining o'zida tashkil etiladi. Birinchi o'tish davomida eng kichik element izlanadi. Bu element topilganidan so'ng uni dastlabki ketma-ketlikdagi birinchi element bilan joyi almashtiriladi, natijada eng kichik element tuzilayotgan tartibga solingan ketma-ketlikda birinchi holatni egallaydi. So'ngra qolgan elementlar ichidan keyingi eng kichik element izlanadi. Topilgan bu element ham dastlabki ketma-ketlikning ikkinchi elementi bilan joyi almashtiriladi. Ikkinchi o'tishdan so'ng ikki elementdan iborat bo'lgan ketma-ketlik tuzilgan bo'ladi, ulardan birinchisi ikkinchisidan kichik bo'ladi. Kalitining qiymati eng kichik bo'lgan keyingi elementni izlash va uni dastlabki ketma-ketlikning tegishli pozisiyalariga joylashtirish barcha elementlar oshib boruvchi tartibda saralanib bo'lingunga qadar davom etadi.

2.1-jadval . Tanlash usulida saralash

i	1	2	3	4	5	6
A[i]	<b>10</b>	<b>4</b>	<b>11</b>	<b>9</b>	<b>7</b>	<b>2</b>
1-o'tish	<b>2</b>	<b>4</b>	<b>11</b>	<b>9</b>	<b>7</b>	<b>10</b>
2-o'tish	<b>2</b>	<b>4</b>	<b>11</b>	<b>9</b>	<b>7</b>	<b>10</b>
3-o'tish	<b>2</b>	<b>4</b>	<b>7</b>	<b>9</b>	<b>11</b>	<b>10</b>
4-o'tish	<b>2</b>	<b>4</b>	<b>7</b>	<b>9</b>	<b>11</b>	<b>10</b>
5-o'tish	<b>2</b>	<b>4</b>	<b>7</b>	<b>9</b>	<b>10</b>	<b>11</b>

**Masala:** Massiv elementlarini o'sish tartibida saralansin

```

1 #include <iostream>
2 using namespace std;
3 int main(){
4     long n, i, j;
5     cin >> n;
6     long a[n];
7     for (i = 1; i <= n; i++)
8         cin >> a[i];
9     for (i = 1; i < n; i++)
10    for (j = i + 1; j <= n; j++)
11    {
12        if (a[i] < a[j])
13        {
14            a[0] = a[i];
15            a[i] = a[j];
16            a[j] = a[0];
17        }
18    }
19    for (i = n; i >= 1; i--)
20        cout << a[i] << " ";
21        system("PAUSE");
22    return 0;
23 }

```

**Almashtirish usuli.** Bu usul bilan saralashda tartibga solinadigan ketma-ketlik xotiraning dastlabki ketma-ketlik joylashgan erida tashkil etiladi. Saralash jarayonida qo'shni elementlar juftlab solishtiriladi. Agar solishtirilayotgan elementlar o'rtasidagi tartib buzilgan bo'lsa ularning joylari almashtiriladi. Bu almashtirish usuli ko'pincha pufakcha usuli deb ham ataladi, chunki eng kichik elementlar har bir o'tishda xuddi pufakchalarga o'xshab ketma-ketlikning birinchi pozisiyasi yo'nalishida «qalqib» chiqadi. 2-rasmida pufakcha usulida saralash namunasi keltirilgan. Birinchi o'tish davomida A1 va A2 elementlari solishtiriladi. Agar A2 < A1 bo'lsa, elementlarning joylari almashtiriladi, bunda A2 element birinchi pozisiyani, A1 element esa ikkinchi pozisiyani egallaydi. Bu jarayon A2 va A3, A3 va A4 element juftlari uchun takrorlanadi va hokazo. Birinchi o'tishdan so'ng eng katta element N pozisiyani egallaydi, eng kichik element esa bitta pozisiyaga yuqoriga ko'tariladi («qalqib chiqadi»). Har bir keyingi o'tishda navbatdagi eng katta elementlar tegishlicha N - 1, N - 2 va hokazo pozisiyalarni egallaydi, natijada tartibga solingan massiv tuziladi.

Har bir o‘tishdan so‘ng ushbu o‘tish davomida joy almashtirishlar bo‘lgan-bo‘lmanligini tekshirib qo‘yish mumkin. Agar joy almashtirishlar bo‘lman bo‘lsa, bu ketma-ketlik tartibga solinganligi va keyingi o‘tishlar talab etilmasligini bildiradi. O‘tishlar davomida almashtirishda ishtirok etadigan oxirgi element (2-rasmda bu elementlar qo‘sh chiziq bilan chizilgan) qayd etiladi. Navbatdagi o‘tishda tagiga chizilgan element va barcha undan keyingi elementlar solishtirishda ishtirok etmaydi, chunki shu pozisiyadan boshlab ketma-ketlik tartibga solingan bo‘ladi.

2.2-jadval. Almashtirish usulida saralash

i	A[i]	1-o‘tish	2-o‘tish	3-o‘tish	4-o‘tish	5-o‘tish
1	10	4	4	4	4	2
2	4	10	9	7	2	4
3	11	9	7	2	7	7
4	9	7	2	9	9	9
5	7	2	10	10	10	10
6	2	11	11	11	11	11

Massivlar ustida bir nechta ammallar bajarish mumkin. Shulardan bir nechtasini ko‘rib o‘tamiz.

**Masala:** n ta elementdan tashkil topgan massiv berilgan. Shu massiv elementlarining eng katta va eng kichik elementlarining yig‘indisini topuvchi dastur tuzing.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {int n;
5     cout << "n="; cin >>n;
6     int a[n];
7     for (int i=0;i<n;i++)
8     {cout << "a[" << i << "]=";
9         cin >>a[i];
10    }
11    int max=a[0];
12    int min=a[0];
13    for (int i=0;i<n;i++)
14    {
15        if (max>a[i])
16        {
17            max=a[i];
18        }
19        if (min<a[i])
20        {
21            min=a[i];
22        }
23        cout << "yig`indi=" << max+min;
24    return EXIT_SUCCESS;}
```

n=5  
a[0]=12  
a[1]=89  
a[2]=120  
a[3]=5  
a[4]=12  
yig`indi=125  
...Program finished with exit code 0  
Press ENTER to exit console.[]

### Nazorat savollari :

1. Statik massiv deb nimaga aytildi.
2. Bir o‘lchovli massivni e’lon qilish tartibini aytинг.
3. Saralash bu.
4. Tanlash usulini aytib bering.
5. Almashtirish usulini aytib bering.
6. Bir o‘lchovli massivga doir masala ishlang.

## 2.2-§ KO‘P O‘LCHOVLI MASSIVLAR

**Reja:**

- **Statik massivlar.**
- **Massiv elementlarini saralash va qidirish usullari.**
- **Massivlar ustida turli xil amallar bajarish usullari.**

C++ dasturlash tilida ko‘p o‘lchovli massivlar bilan ham ishlash mumkin.

Bir o‘lchovli massivlar uchun ishlatiladigan o‘zgaruvchilar, bir xil jinsdagi berilganlarni xotirada saqlash uchun foydalaniladi ikki o‘lchovli massiv satrlari soni ( $m$ ), ustunlari soni  $n$ , hamda elementlari soni larni e’tiborga olish lozim. Agar masalalarda satrlar va ustunlar soni aniq ko‘rsatilmagan bo‘lsa, ularni 2 dan 10 gacha bo‘lgan oraliqda o‘zgartirish tavsiya etiladi. Matritsaning boshlang‘ich qiymati 1- va 2-indekslari 1 bo‘lgan elementida joylashadi. Matritsaga kiritish va chiqarish satrlar bo‘yicha amalga oshiriladi.  $m$  o‘lchamli kvadrat matritsa 2 o‘lchovli massiv hisoblanadi. Matritsalarni tashkil eish va ularning elementlarini chiqarish. Matritsalarni tashkil etish masalalaridagi natijaviy matritsa o‘lchami  $10 \times 10$  dan oshmaydi.

Ikki o‘lchamli statik massivlarning e’lon qilinishida, bir o‘lchamlidan farqi, massiv nomidan keyin qirrali qavs ichida ikkita qiymat yozilganligidadir. Bulardan birinchisi, satrlar sonini, ikkinchisi esa ustunlar sonini bildiradi. Ya’ni ikki o‘lchamli massiv elementiga ikkita indeks orqali murojaat qilinadi. Ikki o‘lchamli massivlar matematika kursidan ma’lum bo‘lgan matritsalarni eslatadi.

Ikki o‘lchamli massiv e’loniga misol:

int a[3][3], b[2][4];

A matritsa B matritsa

$a_{00} \ a_{01} \ a_{02} \ b_{00} \ b_{01} \ b_{02} \ b_{03}$

$a_{10} \ a_{11} \ a_{12} \ b_{10} \ b_{11} \ b_{12} \ b_{13}$

$a_{20} \ a_{21} \ a_{22}$

**Masala :** a(mxn) matritsa berilgan. Shu matritsa elementlarini kirituvchi va ekranga jadval ko‘rinishida chiqaruvchi dastur tuzilsin.

```

1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     int m,n;
7     cout << "m="; cin >> m;
8     cout << "n="; cin >> n;
9     int a[m][n];
10    for (int i=0;i<m;i++){
11        for (int j=0;j<n;j++){
12            cout << "a[" << i << "][" << j << "]=";
13            cin >> a[i][j];
14        }
15    }
16    for (int i=0;i<m;i++){
17        for (int j=0;j<n;j++){
18            cout << a[i][j] << " ";
19        }
20        cout << endl;
21    }
22    system("PAUSE");
23    return EXIT_SUCCESS;
24 }
```

**Masala:** (mxm) o‘lchamli kvadrat matritsa berilgan. Matritsaning asosiy dioganali elementlari “X”, qolgan elementlari “0” qiymat qaytaruvchi dastur tuzilsin.

```

1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4 int main(int argc, char *argv[])
5 {
6     int m,n;
7     cout << "m="; cin >> m;
8     cout << "n="; cin >> n;
9     int a[m][n];
10    for (int i=0;i<m;i++)
11    {
12        for (int j=0;j<n;j++)
13        {
14            if(i==j)
15            {
16                cout << "X" << " ";
17            }
18            else
19            {
20                cout << "0" << " ";
21            }
22            cout << "\n" << endl;
23        }
24    }
25    return EXIT_SUCCESS;
26 }
```

**Masala :** a(mxnx) matritsa berilgan. Shu matritsaning eng katta va eng kichik elementlarining o‘rtalari arifmetikini hisoblovchi dastur tuzilsin.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 { int m,n;
5   cout <<"m="; cin >>m;
6   cout <<"n="; cin >>n;
7   int a[m][n];
8   for (int i=0;i<m;i++)
9   {for (int j=0;j<n;j++)
10    {cout <<"a["<<i<<"]["<<j<<"]=";
11     cin >>a[i][j];}
12    }
13   int max=a[0][0];      m=2
14   int min=a[0][0];      n=2
15   for (int i=0;i<m;i++) a[0][0]=45
16   {for (int j=0;j<n;j++) a[0][1]=78
17    { if (max>a[i][j]) a[1][0]=89
18     max=a[i][j]; a[1][1]=12
19     if (min<a[i][j]) o`rtalari arifmetiki:50.5
20     min=a[i][j];
21    }
22   float ar=(max+min)/2.;
23   cout <<"O`rtalari arifmetiki:"<<ar;
24   return EXIT_SUCCESS;}
```

## Nazorat savollari

1. Statik ko‘p o‘lchovli massiv deb nimaga aytiladi.
2. Ko‘p o‘lchovli massivni e’lon qilish tartibini ayting.
3. Bir o‘lchovli massivdan ikki o‘lchovli massivning farqi.
4. Ikki o‘lchovli massivda satr va ustunni farqlang.
5. Ko‘p o‘lchovli massivga doir masala ishlang.

## § 2.3. KO'RSATKICHLAR VA DINAMIK XOTIRA BILAN ISHLASH.

**Reja:**

1. **Ko'rsatkichlar bilan ishslash.**
2. **Xotirani taqsimlovchi funksiyalar.**

Ko'rsatkich. O'zining qiymati sifatida xotira manziliini ko'rsatuvchi (saqlovchi) o'zgaruvchilarga - ko'rsatkich o'zgaruvchilar deyiladi.

Masalan : Ko'rsatkichning qiymati

- 0x22ff40
- 0x22ff33

va xakazo kabi xotiraning aniq qismi bo'lishi mumkin.

Boshqa o'zgaruvchilar kabi, ko'rsatkichlardan foydalanish uchun ularni e'lon qilish, toifasini aniqlash shart.

➤ **int \*countPtr, count;**

bu yerda countPtr - int toifasidagi ob'ektga ko'rsatkich, count esa oddiy butun (int) toifasidagi o'zgaruvchi. Ko'rsatkichlarni e'lon qilishda har bir o'zgaruvchi oldiga \* qo'yilishi shart.

Ko'rsatkichga doir misol:

**Masala:**

```
include <cstdlib>
include <iostream>
sing namespace std;
nt main( )

    int n=5;
    int*nptr;
    nptr = &n;
    cout << "n="<<n<<endl;
    *nptr = 15;
    cout <<"n="<<n<<endl;
    cout << "\nKorsatkich qiymati,\n";
    cout << "ya'ni ko'rsatkich ko'rsatayotgan adres=0x7ffdd7f84ecc
    Ko'rsatkich ko'rsatayotgan adres qiymati=15
```

## Masala:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
    double n=5;
    double*kptr;
    kptr = &n;
    cout << "O`zgaruvchi qiymati" << endl;
    cout << "n=" << n << endl;
    cout << "kptr=" << *kptr << endl;
    cout << "\nxotira adresi" << endl;
    cout << "n-o`zgaruvchisi joylashgan adres.&n=" << &n << endl;
    cout << "Ko`rsatkich ko`rsatayotgan adres. kptr=" << kptr << endl;
    cout << "Ko`rsatkich - joylashgan adres. &kptr=" << &kptr << endl;
    cout << "\no`zgaruvchilarni xotirada egallagan xajmi" << endl;
    cout << "n=" << sizeof(n) << endl;
    cout << "*kptr=" << sizeof(kptr) << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Masalaning yechimi quyidagicha:

```
n=5
kptr=5

xotira adresi
n-o`zgaruvchisi joylashgan adres.&n=0x7fffb9f2f068
Ko`rsatkich ko`rsatayotgan adres. kptr=0x7fffb9f2f068
Ko`rsatkich - joylashgan adres. &kptr=0x7fffb9f2f070

o`zgaruvchilarni xotirada egallagan xajmi
n=8
*kptr=8
sh: 1: PAUSE: not found
```

Dastur kompyutering asosiy xotirasida axborotni ikkita asosiy usulda saqlashi mumkin. Birinchisi global va mahalliy o`zgaruvchilardan, jumladan massivlar, tuzilmalar va sinflardan foydalanadi. Global va statik mahalliy o`zgaruvchilar bo`lsa, ma'lumotni saqlash joyi dasturni bajarish muddati uchun belgilanadi. Mahalliy o`zgaruvchilar bo`lsa, xotira stekga ajratiladi. Garchi Borland C++ bu o`zgaruvchilar bilan juda samarali ishlasa ham, ularidan foydalanish dasturchidan dasturni bajarish jarayonida kerak bo`ladigan xotira hajmini oldindan bilishni talab qiladi.

Axborotni saqlashning ikkinchi usuli Borland C++ dinamik xotira ajratish tizimidan foydalanishdir. Ushbu usulda ma'lumotni saqlash uchun xotira bo'sh xotira maydonidan kerak bo'lganda ajratiladi va orqaga qaytariladi, ya'ni. kerak

bo‘lma ganda chiqariladi. Bo‘sh xotira maydoni dastur joylashgan xotira maydoni va stek o‘rtasida joylashgan. Bu maydon to‘p deb ataladi va dinamik xotirani ajratish so‘rovlari uchun ishlatiladi.

Dinamik xotiradan foydalanishning afzalligi shundaki, dasturni bajarish jarayonida bir xil xotira turli xil axborotlarni saqlash uchun ishlatilishi mumkin. Xotira ma’lum bir maqsad uchun ajratilganligi va undan foydalanish tugagandan so‘ng bo‘shatilganligi sababli, dasturning boshqa qismida bir xil xotirani boshqa bir vaqtning o‘zida boshqa maqsadlarda ishlatish mumkin. Dinamik xotirani taqsimlashning yana bir afzalligi - u bilan bog‘langan ro‘yxatlar, ikkilik daraxtlar va boshqa dinamik ma’lumotlar tuzilmalarini yaratish qobiliyati.

C ning dinamik xotira taqsimotining yadrosi standart kutubxonaning bir qismi bo‘lgan malloc() va free() funksiyalaridir. Har safar malloc() funksiyasi tomonidan xotira ajratish so‘rovi amalga oshirilganda, mavjud bo‘sh xotiraning bir qismi ajratiladi. Har safar bu xotira free() funksiyasi yordamida bo‘shatilganda, bu xotira tizimga qaytariladi.

C++ tili ikkita dinamik xotira ajratish operatorlarini belgilaydi, yangi va o‘chirish.

ANSI C standarti faqat to‘rtta dinamik xotira ajratish funksiyasini belgilaydi: calloc(), malloc(), free() va realloc(). Biroq, Borland C++ bir qancha boshqa dinamik xotira ajratish funksiyalarini o‘z ichiga oladi. Zamonaviy 32-bitli xotira modeli uchun kodni kompilyatsiya qilishda xotira tekis bo‘ladi va odatda faqat to‘rtta standart xotira ajratish funksiyasidan foydalaniladi.

ANSI C standarti dinamik xotirani ajratish uchun zarur bo‘lgan sarlavha ma’lumotlari stdlib.h faylida mavjudligini belgilaydi. Biroq, Borland C++ stdlib.h yoki alloc.h sarlavha fayllaridan foydalanish imkonini beradi. Biz bu yerda stdlib.h sarlavha faylidan foydalanamiz, chunki u portativlikni ta’minlaydi. Ba’zi boshqa dinamik xotira ajratish funksiyalari alloc.h, malloc.h yoki dos.h sarlavha fayllarini talab qiladi. Har bir funksiyadan foydalanish uchun qaysi sarlavha fayli kerakligiga alohida e’tibor bering.

FROM dinamik tanlash Xotirada biz massiv hajmini oldindan belgilashimiz shart emas, ayniqsa massiv qanday o'lchamga ega bo'lishi har doim ham ma'lum emas. Keyinchalik, xotirani qanday ajratish mumkinligini ko'rib chiqamiz.

C da xotirani ajratish (malloc funksiysi)

malloc() funksiysi ichida belgilangan sarlavha fayli stdlib.h , u kerakli hajmdagi xotira bilan ko'rsatgichlarni ishga tushirish uchun ishlataladi. Xotira sektordan ajratilgan tasodifiy kirish xotirasi ushbu mashinada ishlaydigan har qanday dastur uchun mavjud. Argument ajratiladigan xotira baytlari soni bo'lib, funksiya xotiradagi ajratilgan blokga ko'rsatgichni qaytaradi. malloc() funksiysi boshqa funksiyalar kabi ishlaydi, yangilik yo'q.

Chunki turli xil turlari ma'lumotlar turli xil xotira talablariga ega, biz qandaydir tarzda ma'lumotlar uchun baytdagi hajmni qanday olishni o'rganishimiz kerak har xil turdag'i. Misol uchun, bizga int tipidagi qiymatlar massivi uchun xotira bo'lagi kerak - bu xotiraning bir o'lchamidir va agar biz bir xil o'lchamdag'i, lekin allaqachon char tipidagi massiv uchun xotirani ajratishimiz kerak bo'lsa - bu boshqa o'lcham. Shuning uchun siz qandaydir tarzda xotira hajmini hisoblappingiz kerak. Buni ifodani qabul qiladigan va uning hajmini qaytaradigan sizeof() operatsiyasi yordamida amalga oshirish mumkin. Masalan, sizeof(int) int qiymatini saqlash uchun zarur bo'lgan baytlar sonini qaytaradi. Bir misolni ko'rib chiqing:

```
#o'z ichiga oladi int *ptrVar = malloc(sizeof(int));
```

Ushbu misolda, in 3-qator ptrVar ko'rsatkichiga xotira qismining manzili beriladi, uning hajmi int ma'lumotlar turiga mos keladi. Avtomatik ravishda ushbu xotira maydoni boshqa dasturlar uchun mavjud bo'lmaydi. Va bu ajratilgan xotira keraksiz holga kelgandan so'ng, uni aniq bo'shatish kerak degan ma'noni anglatadi. Agar xotira aniq bo'shatilmasa, dastur oxirida xotira bo'shatilmaydi. operatsion tizim, bu xotira sizintisi deb ataladi. Shuningdek, siz ajratilishi kerak

bo‘lgan ajratilgan xotira hajmini null ko‘rsatkichni o‘tkazish orqali aniqlashingiz mumkin, bu erda bir misol:

```
int *ptrVar = malloc(sizeof(*ptrVar));
```

Bu yerda nima bo‘lyapti? `Sizeof(*ptrVar)` operatsiyasi ko‘rsatgich tomonidan ko‘rsatilgan xotira maydoni hajmini taxmin qiladi. `ptrVar` int tipidagi xotira qismiga ko‘rsatgich bo‘lgani uchun `sizeof()` butun son hajmini qaytaradi. Ya’ni, aslida, ko‘rsatgich ta’rifining birinchi qismiga ko‘ra, ikkinchi qism uchun o‘lcham hisoblanadi. Xo‘sh, nega bizga kerak? Agar to‘satdan ko‘rsatgich ta’rifini o‘zgartirish kerak bo‘lsa, bu zarur bo‘lishi mumkin, int, masalan, float va keyin, biz ko‘rsatgich ta’rifining ikki qismida ma’lumotlar turini o‘zgartirishimiz shart emas. Birinchi qismni o‘zgartirish kifoya qiladi:

```
Float *ptrVar = malloc(sizeof(*ptrVar));
```

Ko‘rib turganingizdek, bunday rekordda bittasi bor forte, biz `malloc()` funktsiyasini `sizeof(float)` yordamida chaqirmasligimiz kerak. Buning o‘rniga, biz float turiga ko‘rsatkichni `malloc()` ga uzatdik, bu holda ajratilgan xotira hajmi avtomatik ravishda o‘zi tomonidan aniqlanadi!

Bu, ayniqsa, xotirani ko‘rsatkich ta’rifidan uzoqroqqa ajratish kerak bo‘lsa foydalidir:

```
Float *ptrVar; /* ... yuz satr kod */... ptrVar = malloc(sizeof(*ptrVar));
```

Agar siz `sizeof()` operatsiyasi bilan xotirani ajratish konstruksiyasidan foydalansangiz, koddagi ko‘rsatgich ta’rifini topishingiz, uning ma’lumotlar turiga qarashingiz kerak bo‘ladi va shundan keyingina xotirani to‘g‘ri taqsimlay olasiz.

Xotirani bo‘shatish `free()` funktsiyasi bilan amalga oshiriladi. Mana bir misol:

```
Bepul (ptrVar);
```

Xotirani bo‘shatgandan so‘ng, ko‘rsatkichni nolga qaytarish yaxshi amaliyotdir, ya’ni `*ptrVar = 0` ni o‘rnating. Agar siz ko‘rsatkichga 0 ni belgilasangiz, ko‘rsatkich null bo‘ladi, boshqacha aytganda, u endi hech qanday

joyga ishora qilmaydi. Har doim xotirani bo'shatgandan so'ng, ko'rsatkichga 0 qo'ying, aks holda xotira bo'shatilgandan keyin ham ko'rsatgich unga ishora qiladi, ya'ni siz tasodifan ushbu xotiradan foydalanishi mumkin bo'lgan boshqa dasturlarga zarar etkazishingiz mumkin, lekin siz bu haqda hech narsa bilmaysiz. bilib olasiz va dastur to'g'ri ishlaydi deb o'ylaysiz.

### **Nazorat savollari**

1. Ko'rsatkich deb nimaga aytildi.
2. Ko'rsatkichni e'lon qilish tartibi.
3. calloc(), malloc(), free() va realloc() funksiyalari bu.
4. sizeof() operatsiyasining vazifasi.
5. Xotirani taqsimlovchi funksiyalar.

## 2.4. § DINAMIK MASSIVLAR.

**Reja:**

**1. Dinamik massivlar.**

**2. Dinamik massivlarni funksiya parametri sifatida qo'llanilishi.**

Massiv *statik* yoki *dinamik* turda bo'lishi mumkin. *Statik* massivning uzunligi oldindan ma'lum bo'ladi va uning elementlari xotirada aniq bir adresdan boshlab ketma-ket joylashadi. *Dinamik* massivning uzunligi dastur bajariishi jarayonida aniqlanadi va uning elementlari dinamik xotirada ayni paytda bo'sh bo'lgan adreslarga joylashadi.

*Masalan, int a[5];* ko'rinishida e'lon qilingan bir o'lchovli massiv elementlari xotirada quyidagicha joylashadi:

Massivning *i*-elementiga *a[i]* yoki *\*(a+i)* – vositali murojaat qilish mumkin. Massiv uzunligi *sizeof(a)* amali orqali aniqlanadi.

Massiv e'lonida uning elementlariga boshlang'ich qiymatlar berish (initsializatsiyalash) mumkin va uning bir nechta variantlari mavjud.

1) O'lchami ko'rsatilgan massiv elementlarini to'liq initsializatsiyalash:

*int b[5]={8,5,-11,41,39};*

Bunda 5 ta elementdan iborat bo'lgan b nomli bir o'lchovli massiv e'lon qilingan va uning barcha elementlariga boshlang'ich qiymatlar berilgan. Bu e'lon quyidagi e'lon bilan ekvivalent:

*int b[5];*

*b[0]=8; b[1]=5; b[2]=-11; b[3]=41; b[4]=39;*

2) O'lchami ko'rsatilgan massiv elementlarini to'liqmas initsializatsiyalash:

*int b[5]={-11,5,29};*

Bu erda faqat massiv boshidagi uchta elementga boshlang'ich qiymatlar berilgan. Shuni aytib o'tish kerakki, massivni initsializatsiyalashda uning boshidagi elementlariga boshlan'ich qiymatlar bermasdan turib, oxiridagi

elementlariga boshlang‘ich qiymatlar berish mumkin emas. Agarda massiv elementlariga boshlang‘ich qiymat berilmasa, unda kelishuv bo‘yicha *static* va *extern* modifikatori bilan e’lon qilingan massiv uchun elementlarning boshlang‘ich qiymati *O(nol)* soniga teng bo‘ladi. Automatik massiv elementlarining boshlang‘ich qiymatlari esa noma’lum hisoblanadi.

3) O‘lchami ko‘rsatilmagan massiv elementlarini to‘liq initsializatsiyalash:

```
int d[]={-15,7,15,24};
```

Bu misolda massivning barcha elementlariga qiymatlar berilgan hisoblanadi, massiv uzunligi esa kompilyator tomonidan boshlanib qiyatlar soniga qarab aniqlanadi.

Shuni ta’kidlash kerakki, massivning uzunligi berilmasa, unga boshlang‘ich qiymatlar berilishi shart.

Dinamik massivlarga xotiradan joy ajratish uchun malloc(), calloc() funksiyalaridan yoki new operatoridan foydalilanadi. Dinamik massivga ajratilgan xotirani bo‘shatish uchun free() funksiyasi yoki delete operatori ishlatiladi.

Yuqorida qayd qilingan funksiyalar alloc.h kutubxonasida joylashgan.

malloc() funksiyasining sintaksisi

```
➤ void * malloc(size_t ulcham) ;
```

ko‘rinishida bo‘lib, u xotiraning uyum qismidan’ulcham’ bayt o‘lchamidagi uzluksiz sohani ajratadi. Agar xotira ajratish muvaffaqiyatli bo‘lsa, malloc() funksiyasi shu soha boshlanishining adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo‘lsa, funksiya NULL qiymatini qaytaradi.

Sintaksidan ko‘rinib turibdiki, funksiya void turidagi qiymat qaytaradi. Amalda esa aniq bir turdagiligi massiv ob’ekti uchun xotiradan joy ajratish zarur bo‘ladi. Shu bois void turini aniq bir turga keltirish texnologiyasidan foydalilanadi. Masalan, butun turdagiligi 3 ga teng massivga joy ajratishni quyidagicha amalga oshirish mumkin:

```
➤ int * d_mas=(int*)malloc(3*sizeof(int));
```

malloc() funksiyasidan farqli ravishda calloc() funksiyasi massiv uchun joy ajratishdan tashqari massiv elementlarini 0(nol) qiymati bilan initsializatsiyalaydi. Bu funksiya sintaksisi quyidagi:

➤ void \* calloc (size\_t miqdor, size\_t ulcham) ;

ko‘rinishda bo‘lib,’miqdor’ parametri ajratilgan sohada nechta element borligini,’ulcham’ esa element o‘lchamini bildiradi.

free() dinamik xotirani bo‘shatish funksiyasi bo‘lib, ko‘rsatilgan dinamik massiv egallab turgan xotira qismini bo‘shatadi:

➤ void free (void \* blok);

free() funksiyasi parametrining void turida bo‘lishi, ixtiyoriy turdag'i xotira bo‘lagini o‘chirish imkonini beradi.

Endi new va delete operatorlari bilan tanishamiz. new operatori yordamida massivga xotiradan joy ajratish uchun massiv turidan keyin kvadrat qavs ichida massiv elementlari soni ko‘rsatiladi. Masalan, butun turdag'i 10 ta sondan iborat massivga joy ajratish quyidagi

```
d_mas=new int[10];
```

ko‘rinishda bo‘ladi. Bu usulda massivga ajratilgan xotirani bo‘shatish uchun

```
delete [] d_mas;
```

buyrug‘ini berish kerak bo‘ladi.

Dinamik massivlar bilan ishlash.

## Masala:

```
1 #include <cstdlib>
2 #include <iostream>
3 #include <alloc.h>
4 using namespace std;
5 int main()
6 {
7     int n,s=0;
8     int*a;
9     cout <<"n=";
10    cin>>n;
11    a=(int*) malloc(n*sizeof(int));
12    if (a==NULL)
13    {
14        cout <<"Xotira yetarli emas";
15        return 1;
16    }
17    for (int i=0;i<n;i++)
18    {
19        cout <<"a["<<i<<"]=";
20        cin >>a[i];
21        s+=a[i];
22    }
23    free (a);
24    cout <<s<<endl;
25    system("PAUSE");
26    return EXIT_SUCCESS;
```

Funksiya parametri sifatida massivni jo‘natish va funksiya natijasi sifatida massivni olish ham mumkin. Funksiyaga matritsani uzatishda matritsani uzatishda matritsa nomi bilan uning satrlar va ustunlar sonini ham jo‘natish kerak bo‘ladi. Funksiyada massivdan foydalanishni bir necha xil usuli bor, shularning ba`zilari bilan tanishamiz.

Funksiyaga matritsani uzatish

## Masala:

```
1 #include <iostream>
2 using namespace std;
3 void matrix_print(int a[10][10],int m,int n)
4 {
5     for (int i=0;i<m;i++)
6     {
7         for (int j=0;j<n;j++)
8             cout <<a[i][j]<<"\t";
9         cout <<"\n"; }
10    void matrix_input (int a[10][10],int m,int n)
11    {
12        cout <<"Massiv elementlarini kriting\n";
13        for (int i=0;i<m;i++)
14        {
15            for (int j=0;j<n;j++)
16                cin >>a[i][j];
17        }
18        int m,n,a[10][10];
19        cout <<"Satrlar sonini kriting\nm=";
20        cin >>m;
21        cout <<"Ustunlar sonini kriting\nn=";
22        cin >>n;
23        matrix_input(a,m,n);
24        cout <<"Kiritilgan matritsa\n";
25        matrix_print(a,m,n);
26        return EXIT_SUCCESS;
```

Masalaning natijasi quyidagicha:

```
3atrlar sonini kiriting
n=2
Ustunlar sonini kiriting
n=2
Massiv elementlarini kiriting
45
78
63
96
Kiritilgan matritsa
45      78
63      96
```

### Nazorat savollari

1. Dinamik massiv deb nimaga aytildi.
2. Dinamik massivlarga xotiradan joy ajratish uchun qanday funksiyalardan foydalilanildi.
3. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo'lsa, funksiya nima qiymatini qaytaradi.
4. malloc(), calloc() funksiyalarining vazifalari.
5. new va delete operatorlarining vazifalari.
6. Dinamik xotirani bo'shatish funksiyasi bu.

## 2.5-§ SATRLAR VA KENGAYTIRILGAN BELGILAR (CHAR TOIFASIDA).

**Reja:**

1. **Satr standart funksiyalari.**
2. **Satr standart funksiyalari yordamida satrlarga ishlov berish.**

**Satrlar.** C da belgili ma'lumotlar uchun char turi qabul qilingan. Belgili axborotni taqdim etishda belgilar, simvolli o'zgaruvchilar va matnli konstantalar qabul qilingan.

➤ **const char c = 'c';**

➤ **char a,b;**

C dagi satr - bu nul-belgi - \0 (nul-terminator)- bilan tugallanuvchi belgilar massivi. Nul-terminatorning holatiga qarab satrning amaldagi uzunligi aniqlanadi. Bunday massivdagi elementlar soni, satr tasviriga qaraganda, bittaga ko'p. Simvolli massivlar quyidagicha inisializasiya qilinadi:

➤ **char shaxar[] = "QASHQADARYO";**

Bu holda avtomatik ravishda massiv elementlari soni aniqlanadi va massiv oxiriga satr ko'chirish '\n' simvoli ko'shiladi. Yukoridagi initsializatsiyani quyidagicha amalga oshirish mumkin:

➤ **char shaxar[] = {'Q','A','S','H','Q','A','D','A','R','Y','O','\n'};**

Bu holda so'z oxirida '\n' simvoli aniq ko'rsatilishi shart.

Misol uchun palindrom so'zni toppish masalasini ko'rib chikamiz. Palindrom deb oldidan ham oxiridan ham bir xil o'qiladigan so'zlarga aytildi.

Standart C++ tili ikki xildagi belgilar majmuasini qo'llab - quvvatlaydi. Birinchi toifaga, an'anaviy, "tor" belgilar deb nomlanuvchi 8-bitli belgilar majmuasi kiradi, ikkinchisiga 16-bitli "keng" belgilar kiradi. Til kutubxonasida har bir guruh belgilari uchun maxsus funksiyalar to'plami aniqlangan.

C++ tilida satr uchun maxsus tur aniqlanmagan. Satr **char** turidagi belgilar massivi sifatida qaraladi va bu belgilar ketma - ketligi satr terminatori deb

nomlanuvchi nol kodli belgi bilan tugaydi ('\0'). Odatda, nol - terminator bilan tugaydigan satrlarni ASCIIZ –satrlar deyiladi. Sart konstanta deb qo'shtirnoqlar ichiga olingan belgilar ketma–ketligiga aytildi: “*Ushbu belgilar ketma–ketligiga satr deyiladi.*”

Quyidagi jadvalda C++ tilida belgi sifatida ishlatilishi mumkin bo'lgan konstantalar to'plami keltirilgan.

Satr massivi e'lon qilinishida satr oxiriga terminator qo'yilishini va natijada satrga qo'shimcha bitta bayt qo'shilishi inobatga olinishi kerak:

**char satr[10];**

Ushbu e'londa **satr** satri uchun jami 10 bayt ajratiladi, 9 satr hosil qiluvchi belgilar uchun va 1 bayt terminator uchun.

Satr o'zgaruvchilari e'lon qilinishida boshlang'ich qiymatlar qabul qilishi mumkin. Bu holda kompilyator avtomatik ravishda satr uzunligini hisoblaydi va satr oxiriga nol terminatorni qo'shib qo'yadi:

**char Hafta\_kuni[]="Juma";**

Ushbu e'lon quyidagi e'lon bilan ekvivalent:

**char Hafta\_kuni[]={‘J’,’u’,’m’,’a’,’\0’};**

Satr qiymatini o'qishda oqimli o'qish operatori “>>” o'rniga **getline()** funksiyasini ishltagan ma'qul hisoblanadi, chunki oqimli o'qishda probellar inkor qilinadi (garchi ular satr belgisi hisoblansa ham) va o'qilayotgan belgilar ketma - ketligi satrdan “oshib” ketganda ham belgilarni kiritish davom etishi mumkin. Natijada satr o'ziga ajratilgan o'lchamdan ortiq belgilarni qabul qilishi mumkin. Shu sababli, **getline()** funksiyasi ikkita parametrga ega bo'lib, birinchi parametr o'qish amalga oshirilayotgan satrga ko'rsatgich, ikkinchi parametrda esa kiritilishi kerak bo'lgan belgilar soni ko'rsatiladi.

## Satrni getline() funksiyasi orqali o‘qishga misol ko‘raylik:

```
1 #include <iostream>
2 using namespace std;
3 int main(){
4     char satr[6];
5     cout<<"Satrni kiritning:"<<"\n";
6     cin.getline(satr,6);
7     cout<<"Siz kiritgan satr:"<<satr;
8     return 0;}
```

satrni kiritning:  
Salom talabalar  
Siz kiritgan satr:Salom

Dasturda **satr** satri 5 ta belgini qabul qilishi mumkin, ortiqchalari tashlab yuboriladi. **getline()** funksiyasiga murojaatda ikkinchi parametr qiymati o‘qilayotgan satr uzunligidan katta bo‘lmasligi kerak.

Satr bilan ishlaydigan funksiyalarning aksariyati **string.h** kutubxonasida jamlangan. Nisbatan ko‘p ishlatiladigan funksiyalarning tavsifini keltiramiz.

## Satr uzunligini aniqlash funksiyalari

Satrlar bilan ishlashda, aksariyat hollarda satr uzunligini bilish zarur bo‘ladi. Buning uchun **string.h** kutubxonasida **strlen()** funksiyasi aniqlangan bo‘lib, uning sintaksisi quyidagicha bo‘ladi:

### ➤ **size\_t strlen (const char\* string)**

Bu funksiya uzunligi hisoblanishi kerak bo‘lgan satr boshiga ko‘rsatgich bo‘lgan yagona parametrga ega va u ishlash natijasi sifatida ishorasiz butun sonni qaytaradi. **strlen()** funksiyasi satrning real uzunligidan bitta kam qiymat qaytaradi, ya’ni nol-terminator o‘rni hisobga olinmaydi.

Xuddi shu maqsadda **sizeof()** funksiyasidan ham foydalanish mumkin va u **strlen()** funksiyasidan farqli ravishda satrning real uzunligini qaytaradi. Quyida keltirilgan misolda satr uzunligini hisoblashning har ikkita varianti keltirilgan:

## Misol:

```
1 #include <iostream>           strlen(Str)=10
2 #include <cstring>           sizeof(Str)=11
3 using namespace std;
4 int main(){
5     char Str[]="1234567890";
6     cout << "strlen(Str)="\<<strlen(Str)<<endl;
7     cout<< "sizeof(Str)="\<<sizeof(Str)<<endl;
8     return 0; }
```

Odatda sizeof() funksiyasidan getline() funksiyasining ikkinchi argumenti sifati ishlatiladi va satr uzunligini yaqqol ko'rsatmaslik imkonini beradi:

**cin.getline(Satr, sizeof(Satr));**

## Satrlarni nusxalash

Satr qiymatini biridan ikkinchisiga nusxalash mumkin. Buning uchun bir qator standart funksiyalar aniqlangan bo'lib, ularning tavsiflari quyida keltiramiz.

### ➤ strcpy() funksiyasi prototipi

**char\* strcpy(char\* str1, const char\* str2)**

ko'rnishga ega va bu funksiya str2 ko'rsatib turgan satrdagi belgilarni str1 ko'rsatib turgan satrga baytma-bayt nusxalaydi. Nusxalash str2 ko'rsatib turgan satrdagi nol-terminal uchraguncha davom etadi. Shu sababli, str2 satr uzunligi str1 satr uzunligidan katta emasligiga ishonch hosil qilish kerak, aks holda berilgan sohasida (segmentida) str1 satrdan keyin joylashgan berilganlar "ustiga" str2 satrning "ortiqcha" qismi yozilishi mumkin.

Navbatdagi dastur qismi "Satrni nusxalash!" satrini Str satrga nusxalaydi:

### ➤ **char Str[20];**

**strcpy(Str, "Satrni nusxalash!");**

Zarur bo'lganda satrning qaysidir joyidan boshlab, oxirigacha nusxadash mumkin. Masalan, "Satrni nusxalash!" satrini 8 belgisidan boshlab nusxa olish zarur bo'lsa, uni quyidagicha yechish mumkin.

### Misol:

```
1 #include <iostream>           nusxalash!
2 #include <string.h>
3 using namespace std;
4 int main(){
5
6 char Str1[20] = "Satrni_nusxalash!";
7 char Str2[20];
8 char* kursatgich = Str1;
9 kursatgich += 7;
10 strcpy(Str2, kursatgich);
11 cout << Str2 << endl;
12 return 0; }
```

strncpy() funksiyasining strcpy() funksiyasidan farqli joyi shundaki, unda bir satrdan ikkinchisiga nusxalanadigan belgilar soni ko'rsatiladi. Uning sintaksisi quyidagi ko'rinishga ega:

char\* strncpy(char\* str1, const char\* str2, size\_t num)

Agar str1 satr uzunligi str2 satr uzunligidan kichik bo'lsa, ortiqcha belgilar "kesib" tashlanadi. strncpy() funksiyasi ishlatilishiga misol ko'raylik:

### Misol:

```
1 #include <iostream>           Uzun_str=01234567890123456789
2 #include <string.h>           Qisqa_str=0123EF
3 using namespace std;
4 int main(){
5 char Uzun_str[] = "01234567890123456789";
6 char Qisqa_str[] = "ABCDEF";
7 strncpy(Qisqa_str, Uzun_str, 4);
8 cout << "Uzun_str=" << Uzun_str << endl;
9 cout << "Qisqa_str=" << Qisqa_str << endl;
10 return 0; }
```

Dasturda Uzun\_str satri boshidan 4 belgi Qisqa\_str satriga oldingi qiymatlar ustiga nusxalanadi va natijada ekranga

01234567890123456789

0123EF

xabarlari chop etiladi.

strup() funksiyasiga yagona parametr sifatida satr–manbaga ko‘rsatgich uzatiladi. Funksiya, satrga mos xotiradan joy ajratadi, unga satrni nusxalaydi va yuzaga kelgan satr-nusxa adresini qaytaradi. strup() funksiya sintaksisi:

**char\* strup(const char\* source)**

Quyidagi dastur bo‘lagida satr1 satrining nusxasi xotiraning satr2 ko‘rsatgan joyida paydo bo‘ladi:

➤ **char\* satr1=”Satr nusxasini olish.”;**

**char\* satr2;**

**satr2=strup(satr1);**

### **Satrlarni ulash**

Satrlarni ulash (konkatenatsiya) amali yangi satrlarni hosil qilishda keng qo‘llaniladi. Bu maqsadda string.h kutubxonasida strcat() va strncat() funksiyalari aniqlangan.

➤ **strcat( ) funksiyasi sintaksisi quyidagi ko‘rinishga ega:**

**char\* strcat(char\* str1, const char\* str2)**

Funksiya ishlashi natijasida str2 ko‘rsatayotgan satr, funksiya qaytaruvchi satr - str1 ko‘rsatayotgan satr oxiriga ulanadi. Funksiyani chaqirishdan oldin str1 satr uzunligi, unga str2 satr ulanishi uchun yetarli bo‘lishi hisobga olingan bo‘lishi kerak.

Quyida keltirilgan amallar ketma-ketligi bajarilishi natijasida satr satriga qo‘shimcha satr ostilari ulanishi ko‘rsatilgan:

**char satr[80];**

**strcpy(satr,”Bu satrga “);**

**strcat(satr,”satr osti ulandi.”);**

Amallar ketma-ketligini bajarilishi natijasida satr satri “Bu satrga satr osti ulandi.” qiymatiga ega bo‘ladi.

**strncat( )** funksiyasi strcat( ) funksiyadan farqli ravishda str1 satrga str2 satrning ko‘rsatilgan uzunligidagi satr ostini ulaydi. Ulanadigan satr osti uzunligi

funksiyaning uchinchi parametri sifatida beriladi. Funksiya sintaksisi

**char\* strncat(char\* str1, const char\* str2, size\_t num)**

Pastda keltirilgan dastur bo‘lagida str1 satrga str2 satrning boshlang‘ich 10 ta belgidan iborat satr ostini ulaydi:

➤ **char satr1[80] = "Programmalash tillariga misol bu";**  
**char satr2[80] = "C++, Pascal, Basic";**  
**strncpy(satr1, satr2, 10);**  
**cout << satr1;**

Amallar bajarilishi natijasida ekranga “Programmalash tillariga misol bu-C++, Pascal” satri chop etiladi.

### **Satrlarni solishtirish**

Satrlarni solishtirish ularning mos o‘rindagi belgilarini solishtirish (katta yoki qichikligi) bilan aniqlanadi. Buning uchun string.h kutubxonasida standart funksiyalar mavjud.

strcmp( ) funksiyasi sintaksisi

**int strcmp(const char\* str1, const char\* str2)**

ko‘rinishiga ega bo‘ltb, funksiya str1 va str2 solishtirish natijasi sifatida son qiymatlarni qaytaradi va ular quyidagicha izohlanadi:

➤ **<0 – agar str1 satri str2 satridan kichik bo‘lsa;**  
**=0 – agar str1 satri str2 satriga teng bo‘lsa;**

**>0 – agar str1 satri str2 satridan katta bo‘lsa.**

Funksiya harflarning bosh va kichikligini farqlaydi. Buni misolda ko‘rishimiz mumkin:

➤ **char satr1[80] = "Programmalash tillariga bu- C++,pascal, Basic.;"**  
**char satr2[80] = "Programmalash tillariga bu- C++,Pascal, Basic.;"**  
**int i;**  
**i = strcmp(satr1, satr2);**

Natijada i o‘zgaruvchisi musbat qiymat qabul qiladi, chunki solishtirilayotgan satrlardagi “pascal” va “Pascal” satr ostilarida birinchi harflar farq qiladi. Keltirilgan misolda i qiymati 32 bo‘ladi – farqlanuvchi harflar satrning 32 elementi hisoblanadi. Agar funksiyaga

**i=strcmp(satr2,satr1);**

ko‘rinishida murojaat qilinsa i qiymati –32 bo‘ladi.

Agar satrlardagi bosh yoki kichik harflarni farqlamasdan solishtirish amalini bajarish zarur bo‘lsa, buning uchun strcmp() funksiyasidan foydalanish mumkin. Yuqorida keltirilgan misoldagi satrlar uchun

**i=strcmp(satr2,satr1);**

amali bajarilganda i qiymati 0 bo‘ladi.

**strncmp( ) funksiyasi sintaksisi**

**int strncmp(const char\* str1, const char\* str2, size\_t num)**

ko‘rinishida bo‘lib, str1 str2 satrlarni boshlang‘ich num sonidagi belgilarini solishtiradi. Funksiya harflar registrini inobatga oladi. Yuqorida misolda aniqlangan satr1 va satr2 satrlar uchun

**i=strncmp(satr1,satr2,31);**

amali bajarilishida i qiymati 0 bo‘ladi, chunki satrlar boshidagi 31 belgilar bir xil. strncmp( ) funksiyasi strncmp( ) funksiyasidek amal qiladi, farqli tomoni shundaki, solishtirishda harflarning registrini hisobga olinmaydi. Xuddi shu satrlar uchun

**i=strncmp(satr1,satr2,32);** amali bajarilishi natijasida i o‘zgaruvchi qiymati 0 bo‘ladi.

**Satrda harflar registrini almashtirish**

Berilgan satrdagi kichik harflarni bosh harflarga yoki teskari almashtirishga mos ravishda \_strupr( ) va \_strlwr( ) funksiyalar yordamida amalga oshirish mumkin. Kompilyatorlarning ayrim variantlarida funksiyalar nomidagi tagchiziq (‘\_’) bo‘lmasligi mumkin.

**\_strlwr( ) funksiyasi sintaksisi**

### **char\* \_strlwr(char\* str)**

ko‘rinishida bo‘lib, argument sifatida berilgan satrdagi bosh harflarni kichik harflarga almashtiradi va hosil bo‘lgan satr adresini funksiya natijasida qaytaradi. Quyidagi dastur bo‘lagi `_strlwr()` funksiyasidan foydalanishga misol bo‘ladi.

```
char str[]="10 TA KATTA HARFLAR";  
_strlwr(str);  
cout<<str;
```

Natijada ekranga “10 ta katta harflar” satri chop etiladi.

`_strupr()` funksiyasi xuddi `_strlwr()` funksiyasidek amal qiladi, lekin satrdagi kichik harflarni bosh harflarga almashtiradi:

```
char str[]="10 ta katta harflar";  
_strupr(str);  
cout<<str;
```

Natijada ekranga ”10 TA KATTA HARFLAR” satri chop etiladi.

Dasturlash amaliyotida belgilarni qaysidir oraliqqa tegishli ekanligini bilish zarur bo‘ladi. Buni ctype.h sarlavha faylida e’lon qilingan funksiyalar yordamida bilsa bo‘ladi. Quyida ularning bir qismining tavsifi keltirilgan:

- `isalnum()` – belgi raqam yoki harf (true) yoki yo‘qligini (false) aniqlaydi;
- `isalpha()` – belgini harf (true) yoki yo‘qligini (false) aniqlaydi;
- `isascii()` – belgini kodi 0..127 oralig‘ida (true) yoki yo‘qligini (false) aniqlaydi;
- `isdigit()` – belgini raqamlar diapazoniga tegishli (true) yoki yo‘qligini (false) aniqlaydi.

### **Nazorat savollari :**

1. Satr deb nimaga aytildi.
2. Satr uzunligini aniqlash funksiyalari.
3. Satrlarni nusxalash
4. Satrlarni ularash
5. Satrdagi harflar registrini almashtirish
6. Char toifasidagi satrda ishlatiladigan funksiyalar.

## § 2.6. SATRLAR VA KENGAYTIRILGAN BELGILAR (STRING TOIFASIDA)

**Reja:**

- 1. Satrlar bilan ishslash.**
- 2. String turidagi satrni e'lon qilish va qiymat berish**
- 3. String turidagi satr funksiyalar (metodlar).**

C++ tilida standart satr turiga qo'shimcha sifatida string turi kiritilgan va u string sinfi ko'rinishida amalga oshirilgan. Bu turdag'i satr uchun '\0' belgisi tugash belgisi hisoblanmaydi va u oddiygina belgilar massivi sifatida qaraladi. string turida satrlar uzunligining bajariladigan amallar natijasida dinamik ravishda o'zgarib turishi, uning tarkibida bir qator funksiyalar aniqlanganligi bu - tur bilan ishslashda ma'lum bir qulayliklar yaratadi.

### **String turidagi satrni e'lon qilish va qiymat berish**

string turidagi o'zgaruvchilar quyidagicha e'lon qilinishi mumkin:

**string s1,s2,s3;**

Bu turdag'i satrlar uchun maxsus amallar va funksiyalar aniqlangan. string satrga boshlang'ich qiymatlar har xil usullar orqali berish mumkin:

```
string s1="birinchi usul";
string s2("ikkinchi usul");
string s3(s2);
string s4=s2;
```

Xuddi shunday, string turidagi o'zgaruvchilar ustida qiymat berish amallari ham har xil:

```
➤ string s1,s2,s3 ;
char *str="misol";
//satrli o'zgarmas qiymati berish
s1="Qiymat berish 1-usul";
s2=str;           // char turidagi satr yuklanmoqda
```

```
s3='A';           // bitta belgi qiymat sifatida berish
s3=s3+s1+s2+"0123abc"; //qiymat sifatida satr ifoda
```

### Satr qismini boshqa satrga nusxalash funksiyasi

Bir satr qismini boshqa satrga yuklash uchun kuyidagi funksiya-larni ishlatish mumkin, ularni prototipi kuyidagicha:

```
> assign(const string &str);
assign(const string &str,unsigned int pos, unsigned int n);
assign(const char *str, int n);

string s1,s2;
s1="birinchi satr";
s2.assign(s1); // s2=s1 amalga ekvivalent
```

Ikkinchi funksiya chaqiruvchi satrga argumentdagi str satrning pos o‘rnidan n ta belgidan iborat bo‘lgan satr qismini nusxalaydi. Agarda pos qiymati str satr uzunligidan katta bo‘lsa, xatolik haqida ogohlantiriladi, agar pos + n ifoda qiymati str satr uzunligidan katta bo‘lsa, str satrining pos o‘rnidan boshlab satr oxirigacha bo‘lgan belgilar nusxalanadi. Bu qoida barcha funksiyalar uchun tegishlidir.

Satr qismini boshqa satrga qo‘shish funksiyalari quyidagicha:

```
append(const string &str);
append(const string & str,unsigned int pos, unsigned int n);
append(const char *str, int n);
```

```
1 #include <cstdlib>           TATU Qarshi filiali
2 #include <iostream>
3 #include <string.h>
4 using namespace std;
5 int main()
6 {
7     string a="TATU ";
8     string b="Qarshi filiali";
9     string c=a.append(b);
10    cout <<c<<endl;
11 }
```

## *Satr qismini boshqa satr ichiga joylashtirish funksiyasi*

Bir satrga ikkinchi satr qismini joylashtirish uchun quyidagi funksiyalar ishlatiladi:

**insert(unsigned int pos1,const string &str);**

**insert(unsigned int pos1,const string & str, unsigned int pos2,unsigned int n);**

**insert(unsigned int pos1,const char \*str, int n);**

```
1 #include <cstdlib>          Salom guruxda kim yo`q talabalar
2 #include <iostream>
3 #include <string.h>
4 using namespace std;
5 int main()
6 {
7     string a="Salom talabalar";
8     string b="guruxda kim yo`q ";
9     a.insert(6,b);
10    cout <<a<<endl;
11 }
```

## *Satr qismini o‘chirish funksiyasi*

Satr qismini o‘chirish uchun quyidagi funksiyani ishlatish mumkin:

➤ **erase(unsigned int pos=0,unsigned int n=npos);**

**Misol:**

```
1 #include <cstdlib>          talabalar
2 #include <iostream>
3 #include <string.h>
4 using namespace std;
5 int main()
6 {
7     string a="salom talabalar";
8     string b="guruha dars";
9     a.erase(0,6);
10    cout <<a<<endl;
11 }
```

Bu funksiya, uni chaqiruvchi satrning pos o‘rnidan boshlab n ta belgini o‘chiradi. Agarda pos ko‘rsatilmasa, satr boshidan boshlab o‘chiriladi. Agar n ko‘rsatilmasa, satrni oxirigacha bo‘lgan belgilar o‘chiriladi:

**void clear() funksiyasi, uni chaqiruvchi satrni to‘liq tozalaydi.**

Masalan:

**s1.clear(); //satr bo‘sh hisoblanadi (s1="")**

```
1 #include <cstdlib>
2 #include <iostream>
3 using namespace std;
4 int main()
5 {
6     string a;
7     getline (cin,a);
8     a.clear();
9     cout <<a<<endl;}
```

Darsimiz dasturlash

### *Satr qismini almashtrish funksiyasi*

Bir satr qismining o‘rniga boshqa satr qismini qo‘yish uchun quyidagi funksiyalardan foydalanish mumkin:

➤ **replace(unsigned int pos1,unsigned int n1, const string & str);**  
**replace(unsigned int pos1,unsigned int n1, const string & str,unsigned int pos2, unsigned int n2);**

➤ **replace(unsigned int pos1,unsigned int n1, const char \*str, int n);**

Bu funksiya insert kabi ishlaydi, undan farqli ravishda amal chaqiruvchi satrning ko‘rsatilgan o‘rnidan (pos1) n1 belgilar o‘rniga satrni yoki uning (pos2) o‘rnidan

### Misol:

```
1 #include <cstdlib>
2 #include <iostream>
3 #include <string.h>
4 using namespace std;
5 int main()
6 {
7     string a;
8     cout << "Satrni kirititing:" ;
9     getline (cin,a);
10    A: int k=a.find("a");
11    if (k!=-1)
12    {
13        a.replace(k,1,"A");
14        goto A;
15    }
16    cout << a << endl;
17 }
```

```
Satrni kirititing:Bugun darsimiz dasturlash
Bugun dArsimiz dAsturlAsh
```

Ushbu dasturda matndan kichik a harfning o‘rniga katta A harfiga o‘zgaruvchi dastur kodi kiritilgan.

### Nazorat savollari :

1. Satr deb nimaga aytildi.
2. Satrni e`lon qilish tartibi.
3. Satr qismini almashtirish funksiyasi
4. Satr qismini o‘chirish funksiyasi
5. Satrlarni nusxalash
6. String toifasidagi satrda ishlatiladigan funksiyalar.

### III BOB. FAYLLAR BILAN ISHLASH, FOYDALANUVCHI TOIFASINI YARATISH. OBYEKTGA YO'NALTIRILGAN DASTURLASH ASOSLARI. INKAPSULYASIYA VA MEROSXO'RLIK.

#### 3.1-§ FAYLLAR BILAN ISHLASH

**Reja:**

1. **Fayllar va oqimlar, matnli fayllar.**
2. **Fayllar bilan ishlovchi maxsus funksiyalar**
3. **Binar fayllar**
4. **Binar fayllar bilan ishlovchi maxsus funksiyalar**

C++ dasturlash tili nafaqat boshqa dasturlash tillarida ham fayllar bilan ishlash juda katta ahamiyatga ega hisoblanadi. C++ dasturlash tilida ***fstream*** standart kutubxonadan foydalilanadi. ***fstream***dan foydalinish uchun ***<iostream>*** va ***<fstream>*** standart kutubxonalardan foydalilanadi.

```
#include <iostream>
```

```
#include <fstream>
```

***fstream*** standart kutubxonasi ichida 3 ta obyekt mavjud.

#### **Fayl yaratish va unga yozish**

Fayl yaratish uchun, ***ofstream*** yoki ***fstream*** obyektdan foydalaning va fayl nomini ko'rsating. Faylga yozish uchun kiritish operatoridan (<<) foydalaning.

```
#include <iostream>
using namespace std;
int main() {
    // Faylni yaratadi yoki ochadi.
    ofstream MyFile("filename.txt");
    // Faylga yozadi.
    MyFile << "Hello World. MasterSherkulov.Uz";
    #include <fstream>
```

```
// Faylni yopadi.  
MyFile.close();  
}
```

## Fayldan o‘qish

Fayldan o‘qish uchun, ***ifstream*** yoki ***fstream*** obyektdan va fayl nomidan foydalaning. E’tibor bering, biz funksiyani (obyektga tegishli) funksiya while bilan bir qatorda fayl satrini o‘qish va fayl tarkibini chop etish uchun ishlatalamiz.

```
#include <iostream>  
#include <fstream>  
#include <string>  
using namespace std;  
int main () {  
    // Faylni yaratish  
    ofstream MyWriteFile("filename.txt");  
    // Faylga yozish  
    MyWriteFile << "Hello World";  
    // Faylni yopish  
    MyWriteFile.close()  
    // String tipiga tegishli o‘zgaruvchi yaratish  
    string myText  
    // Text faylni o‘qish  
    ifstream MyReadFile("filename.txt");  
    // getline() funksiyasidan foydalanib faylni o‘qish  
    while (getline (MyReadFile, myText)) {  
        // O‘qilgan faylni qora ekranga chiqarish
```

```
cout << myText;}  
// Faylni yopish  
MyReadFile.close();}
```

*Hello World*

### Fayllar bilan ishlash. Binar fayllar

Programma ishlashi natijasida olingan ma'lumotlarni saqlab qo'yish uchun, CD,DVD disklar, qattiq disklar va boshqa har xil tashqi qurilmalardan foydalaniadi.

Ma'lumotlarni saqlab qo'yish uchun tashqi qurilmalardan foydalanish qulay va ishonchlidir.

Ma'lumotlarni saqlab qo'yish uchun, tashqi xotiraning nomlangan qismiga *fayl* deyiladi. Bunday fayllar *fizik fayllar* deyiladi.

**Mantiqiy fayllar.** Fizik fayllar bilan ishlash uchun, programmalashtirish tillarida maxsus strukturalashgan, toifalangan fayllar kiritilgan. Bunday fayllar mantiqiy (logicheskiy) fayllar deyiladi. Mantiqiy fayllar, hech qanday fizik xotirani band qilmasdan ma'lumotlarlarning mantiqiy modelini o'zida saqlaydi.

**Fizik va mantiqiy fayllar** bir - biri bilan fopen funksiyasi orqali bog'lanadi.

Fayl bir nechta elementdan tashkil topgan bo'lganligi uchun, faqat fayl ko'rsatkichi ko'rsatayotgan elementga murojaat qilish mumkin.

Fayldan o'qish yoki yozish mumkin bo'lgan o'rinni ko'rsatuvhi elementga *fayl ko'rsatkichi* deyiladi.

Fayldan ma'lumot o'qiganda yoki yozganda fayl ko'rsatkichi avtomat ravishda o'qilgan yoki yozilgan bayt miqdoricha siljiydi. Fayl ko'rsatkichini magnitafon galovkasiga o'xshatish mumkin.

**Binar fayl** - har xil obyektlarni ifodalovchi baytlar ketma-ketligidir. Obyektlar faylda qanday ketma-ketlikda joylashganini programmaning o'zi

aniqlashi lozim.

Fayllar bilan ishlovchi funksiyalardan foydalanish uchun `<stdio.h>` sarlavha faylini programmaga qo'shish kerak bo'ladi. Fayldan ma'lumotlarni o'qish yoki yozish uchun ochish **fopen** funksiyasi orqali amalga oshiriladi.

`FILE * fopen (const char * filename, const char * mode);`

filename - o'zgaruvchisi char toifasidagi satr bo'lib, faylning to'liq nomini ko'rsatishi lozim:

`(filename="D:\Qudrat_c++\Namuna\file\file.txt").`

Agar faylning faqat nomi ko'rsatilgan bo'lsa, fayl joriy katalogdan qidiriladi (filename = "file.txt").

**Mode** - o'zgaruvchisi ham char toifasidagi satr bo'lib, faylni qaysi xolatda ochish lozimligini bildiradi.

<b>Mode qiymati</b>	<b>Faylning ochilish xolati</b>
"w"	Faylni yozish uchun ochish. filename o'zgaruvchisida ko'rsatilgan fayl hosil qilinadi va unga ma'lumot yozish mumkin bo'ladi. Agar fayl oldindan bor bo'lsa (ya'ni oldin hosil qilingan bo'lsa), faylning ma'lumotlari o'chiriladi va yangi bo'sh fayl faqat yozish uchun ochiq holda bo'ladi.
"r"	Fayl o'qish uchun ochiladi. Agar fayl oldindan mavjud bo'lmasa, xatolik sodir bo'ladi. Ya'ni ochilishi lozim bo'lgan fayl oldindan hosil qilingan bo'lishi shart.
"a"	Faylga yangi ma'lumotlar qo'shish - kiritish uchun ochiladi. Yangi kiritilgan ma'lumotlar fayl oxiriga qo'shiladi. Agar fayl oldindan mavjud bo'lmasa, yangi fayl hosil qilinadi.
"w+"	Yozish va o'qish uchun faylni ochish. Agar fayl oldindan bor bo'lsa (ya'ni oldin hosil qilingan bo'lsa), faylning ma'lumotlari

Mode qiymati	Faylning ochilish xolati
	o‘chiriladi va yangi bo‘sh fayl yozish va o‘qish uchun ochiqholda bo‘ladi.
"r+"	Oldindan mavjud bo‘lgan faylni o‘qish va yozish uchun ochish.
"a+"	Fayl ma’lumotlarni o‘qish va yangi ma’lumot qo‘shish uchun ochiladi. fseek, rewind

Faylni ochishda xatolik sodir bo‘lsa, **fopen** funksiyasi NULL qiymat qaytaradi. Ochilgan faylni yopish uchun **fclose** funksiyasi ishlatiladi.

*int fclose ( FILE \* stream );*

Faylni yopishda xato sodir bo‘lmasa, fclose funksiyasi nol qiymat qaytaradi. Xato sodir bo‘lsa, EOF - fayl oxiri qaytariladi.

### **Faylga ma’lumot yozish va o‘qish**

*size\_t fread ( void \*ptr, size\_t size, size\_t n, FILE \* stream );*

**fread** funksiyasi, fayldan **ptr** ko‘rsatkichi adresiga size xajmdagi ma’lumotdan n tani o‘qishni amalga oshiradi. Agar o‘qish muvoffaqiyatli amalga oshsa fread funksiyasi o‘qilgan bloklar soni n ni qaytaradi. Aks holda nol qaytariladi.

*size\_t fwrite ( const void \* ptr, size\_t size, size\_t n, FILE \* stream );*

**fwrite** funksiyasi, faylga ptr ko‘rsatkichi adresidan boshlab size xajmdagi ma’lumotdan n tani yozishni amalga oshiradi.

Yuqoridagi misolda satrni yozish va o‘qish uchun quyidagicha kod ishlatildi:

*fwrite(s, sizeof(char), strlen(s) + 1, f); fread (s, sizeof(char), strlen(s) + 1, f);*

Buning kamchiligi s satridagi har bir belgi alohida - alohida faylga yozildi va o‘qildi. Bu masalani quyidagicha hal qilish mumkin edi:

*fwrite(s, sizeof(s), 1, f);*

*fread (s, sizeof(s), 1, f);*

Lekin bu usulning ham kamchiligi bor. Ya’ni s satri belgilari soni massiv o‘lchamidan kam bo‘lgan holda keraksiz ma’lumotlarni saqlash va o‘qish sodir bo‘ladi.

### **Fayl ko‘rsatkichi bilan ishlovchi funksiyalar**

Fayldan ma’lumot o‘qiganda yoki yozganda fayl ko‘rsatkichi avtomat ravishda o‘qilgan yoki yozilgan bayt miqdoricha siljiydi. Fayl ko‘rsatkichining kelgan joyini aniqlash uchun ftell funksiyasi ishlariladi.

*long int ftell ( FILE \* stream );*

Fayl ko‘rsatkichini siljitish uchun fseek funksiyasi ishlatiladi.

*int fseek ( FILE \* stream, long int offset, int whence);*

Bu funksiya **offset** da ko‘ratilgan bayt miqdoricha siljishni amalga oshiradi. whence o‘zgaruvchisi quyidagi qiymatlarni qabul qilishi mumkin:

O‘zgarmas	whence	Izoh
<b>SEEK_SET</b>	0	Fayl boshiga nisbatan siljitish
<b>SEEK_CUR</b>	1	Fayl ko‘rsatkichining joriy xolatiga nisbatan siljitish
<b>SEEK_END</b>	2	Fayl oxiriga nisbatan siljitish

Agar whence = 1 bo‘lsa (SEEK\_CUR), offset musbat (o‘ngga siljish) yoki manfiy (chapga siljish) bo‘lishi mumkin.

Fayl ko‘rsatkichini faylning boshiga o‘rnatish uchun **rewind** funksiyasi ishlatiladi.

*void rewind ( FILE \* stream );*

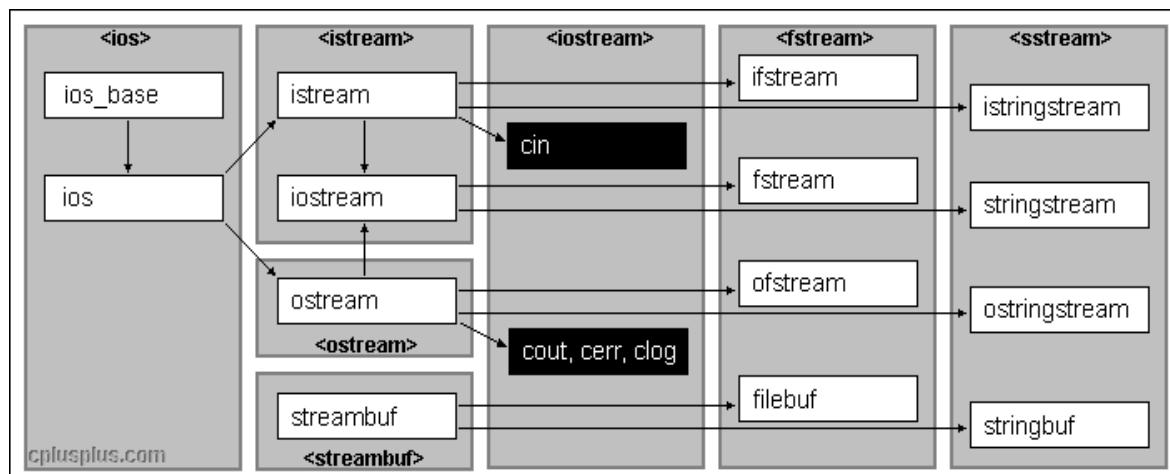
Bu amalni fayl ko‘rsatkichini siljitish orqali ham amalga oshirish mumkin.

## Matnli fayllar.

Matnli fayllar bilan ishlash binar fayllar bilan ishlashdan bir oz farq qiladi. Matnli fayllarda ma'lumotlar satrlarda saqlanadi. Matnli fayl elementilari har xil uzunlikdagi satrlardir. Bu satrlar bir biridan satr oxiri belgisi bilan ajratiladi. Matnli fayl elementlari indekslanmagan bo'lganligi uchun, faylning istalgan elementiga bevosita murojaat qilib bo'lmaydi.

C++ da matnli yoki binar fayllar bilan ishlash uchun keng imkoniyatlar berilgan. Matnli fayllar bilan ishlashda oddiy C ning funksiyalaridan ham foydalanish mumkin. Masalan, formatli o'qish va yozish funksiyalari yoki oldingi mavzudagi funksiyalardan foydalanishimiz mumkin. Matnli fayllar bilan ishlashning bunday usuli kitoblarda keng yoritilgan. Ularni mustaqil o'qib - o'rganishingiz mumkin.

Bu mavzu fayllar bilan ishlovchi oqimlarni qisqacha o'rganamiz va buni matnli fayl misolida ko'ramiz.



Standart kiritish / chiqarish kutubxonasi sinflari quyidagicha ko'rinishga ega:

Fayllar bilan ishlash uchun quyidagi sifnlar obyektlari hosil qilinadi:

- ofstream - faylga ma'lumot yozish uchun
- ifstream - fayldan ma'lumot o'qish uchun
- fstream - fayldan ma'lumot o'qish uchun va yozish uchun

Bu sinflarni dasturda ishlatish uchun **<fstream>** sarlavha faylini qo'shish kerak bo'ladi. Bundan keyin programmada aniq fayllar oqimini aniqlash mumkin. Masalan:

```
ofstream yozish; // faylga yozish oqimini e'lon qilish
ifstream oqish; // fayldan o'qish oqimini e'lon qilish
fstream yoz_oqi; // faylga yozish va o'qish oqimini e'lon qilish
```

Keyin faylni ochish kerak bo'ladi. Faylni ochish deganda, uning ustida nima amal qilinishi haqida amaliyot tizimiga xabar berish tushuniladi.

```
void open (const char * filename, ios_base::openmode mode = ios_base::out
);
```

mode parametri quyidagicha qiymatlarni qabul qilishi mumkin:

ios::in	faqat ma'lumot o'qish uchun
ios::out	faqat ma'lumot yozish uchun
ios::ate	faylni ochishda fayl ko'rsatkichini fayl oxiriga qo'yish
ios::app	fayl oxiriga ma'lumotlarni yozish uchun
ios::trunc	bor bo'lgan faylning ustidan yangi faylni yozish
ios::binary	binar holda ma'lumotlarni almashish uchun

Har bir sinf uchun mode parametrining odatiy qiymatlari mavjud:

class	default mode parameter
ofstream	ios::out
ifstream	ios::in
fstream	ios::in   ios::out

Fayl ustida o'qish yoki yozish amalini bajarib bo'lgandan song, faylni yopish kerak bo'ladi. Faylni yopish uchun **close** funksiyadi ishlatiladi:

istream sinfi funksiyalari istream& seekg ( streampos pos );

istream& seekg ( streamoff off, ios\_base::seekdir );;

oqish oqimi ko'rsatkichini o'rnatish (siljitim). pos - oqim buferining yangi pozitsiyasi.

### 3.2-§ FOYDALANUVCHI TOIFASINI YARATISH

**Reja:**

- 1. Struktura (Tuzilma)**
- 2. Union (Birlashma)**

Ma'lumotlarning ixtiyoriy toifasi qiyamatlar sohasi va ular ustida bajarilishi mumkin bo'lgan amallar orqali tavsiflanadi.

#### **Butun toifa – INT**

Mazkur toifa butun sonlar to‘plamini qandaydir qism to‘plami bo‘lib, uning o‘lchami mashina, ya’ni EHM konfiguratsiyasiga bog‘liq ravishda o‘zgarib turadi. Agar butun sonni mashinada tasvirlash uchun p ta razryaddan foydalanilsa (bunda qo‘sishimcha koddan foydalanilganda), u holda x butun sonning qiyamat qabul qilish oralig‘i quyidagicha bo‘lishi zarur, ya’ni quyidagi shartni qanoatlantirishi lozim:

$$-2n-1 \leq x < 2n-1.$$

Butun toifadagi ma'lumotlar ustida bajariladigan barcha amallar to‘g‘ri amalga oshiriladi deb hisoblanib, ushbu amallar arifmetikada qabul qilgan qoidalariga bo‘ysunadi. Agar ushbu toifada amallar bajarilganda natija ruxsat etilgan oraliqdan chiqib ketsa, u holda hisoblash to‘xtatiladi. Bunday hol to‘lib ketish deb ataladi.

Mazkur toifaga kiruvchi sonlar ikkitaga bo‘linadi: ishorali va ishorasiz. Ularning har bir uchun mos ravishda qiyamat qabul qilish oralig‘i mavjud:

- a) ishorasiz sonlar uchun (0..2n-1);
- b) ishoralilar uchun (-2N-1.. 2N-1-1).



Sonlar mashinada qayta ishlanayotganda ularning ishorali ko‘rinishidan foydalaniladi. Agar mashina so‘zi yozuv, komandarani qayta ishlash va

ko‘rsatkichlar uchun foydalanilayotgan bo‘lsa, u holda sonning ishorasiz ko‘rinishidan foydalaniladi.

Butun sonlar ustida – qo‘shish, ayrish, ko‘paytirish, butunsonli bo‘lish (qoldiqni tashlab yuborish orqali), berilgan modul bo‘yicha hisoblash (bo‘lishda qolgan qoldiqni hisoblash), berilgan sonlar to‘plamining eng katta va eng kichik elementini aniqlash, butun darajaga oshirish, sonning qiymatiga qarab o‘zidan oldingi yoki keyingi sonni aniqlash. Bu operatsiyalarning natijalari ham butun sonlar bo‘ladi.

Butun sonlar ustida ==, !=, <, <=, >, >= operatorlar bilan taqqoslash amallarni ham bajarish mumkin. Ammo bu operatsiyalarning natijalari INT toifasiga kirmaydi, ular BOOL toifasiga kiradi.

### **Haqiqiy toifa**

Haqiqiy toifaga kasr qismlari bor chekli sonlar to‘plami kiradi. To‘plamni chekli bo‘lish sharti EXMda sonlarni ifodalash chegaralanganligi bilan bog‘liq. Haqiqiy sonlar ustida quyidagi amallarni bajarish mumkin: qo‘shish, ayrish, bo‘lish, ko‘paytirish, trigonometrik funksiyalarini xisoblash, darajaga oshirish, kvadrat ildiz chiqarish, logarifmlash, minimum va maksimum elementlarni topish va boshqalar. Bularning natijalari ham haqiqiy toifaga kiradi. Bu yerda ham binar amallarga nisbatan masalaning yechimlari mantiqiy toifaga tegishli bo‘ladi.

EHM xotirasida haqiqiy sonlar asosan qo‘zg‘aluvchan nuqta formatida saqlanadi. Bu formatda x haqiqiy son quyidagi ko‘rinishda ifodalanadi:  $x = +/- M * q(+/-P)$  – soning yarimlogarifmik shakldagi ifodalanishi quyidagi chizmada keltirilgan.

$$937,56 = 93756 * 10^{-2} = 0,93756 * 10^3$$

0	1	9 10	11	15
Мантисса ишораси	Мантисса	Тартиб ишораси	Тартиб	

## Mantiqiy toifa

Mazkur toifa mantiqiy mulohazalarni to‘g‘riligini aniqlash uchun, turli hil dasturlash tillarida turlicha ifodalaniladigan ifodalarni 2 ta true(1), false(0) ko‘rinishda aniqlaydi. Mantiqiy ma’lumotlar ustida quyidagi mantiqiy operatsiyalarni bajarish mumkin: konyunksiya (va), dizyunksiya (yoki) va inkor (yo‘q), hamda qiyinroq bo‘lgan ekvivalentlik, implikatsiya, chiqarib tashlash, yoki va boshqa operatsiyalar. Yuqorida keltirilgan ixtiyoriy operatsiyaning natijasi – mantiqiy qiymatga ega bo‘ladi. Mantiqiy qiymatni xotirada saqlash uchun bitta bit yetarli.

Asosiy mantiqiy funksiyalarning chinlik jadvali

A	B	not A	A or B	A and B
1	1	0	1	1
1	0	0	1	0
0	1	1	1	0
0	0	1	0	0

## Belgili toifa

Belgili toifaga belgilarning chekli to‘plami yoki liter, ularga lotin alifbosidagi xarflar va unda yo‘q kirill xarflar, o‘nlik raqamlar, matematik va maxsus belgilar kiradi. Belgili ma’lumotlar hisoblash texnikasi bilan inson o‘rtasidagi aloqani o‘rnatishda katta ahamiyatga ega. Ko‘pincha, dasturlashning har bir tizimida belgilar to‘plami fiksirlangan bo‘lib, ular turli tizimlarda turli hil bo‘lishi mumkin. Bundan tashqari ular tartiblangan bo‘lib, har bir uning elementiga aniq bir sonli kod mos qo‘yilib, u to‘plamdagagi tartib raqamini aniqlaydi. Belgini sonli kodiga o‘tib, relyatsion operatorlardan foydalanib, simvollarni taqqoslash mumkin. Bunday taqqoslashlarning natijalari BOOL toifasiga kiradi.

C++ tilida belgili toifadan tashqari belgilar massividan tashkil topgan satrli toifalar bilan ham ishlash mumkin, ya’ni char []. Shu o‘rinda aytib o‘tish kerakki, satrlar bilan ishlashda belgilar massividan tashqari satrlar bilan ishlashga mo‘ljallangan maxsus kutubxona mavjud bo‘lib, u “ String “ deb nomlanadi. Satr (qator, String) – bu qandaydir belgilar ketma-ketligi. Satr bitta, bo‘sh yoki bir nechta belgilar birlashmasidan iborat bo‘lishi mumkin. C++ tilida satr 0 dan to 255 tagacha uzunlikka ega bo‘lishi mumkin. Agar o‘zgaruvchi satr toifasiga tegishli bo‘lsa, u holda o‘zgaruvchi toifasi yozilayotganda 2 xil ko‘rinishda char [] yoki String deb aniqlanadi.

Belgili toifadagi amallar:

- a) O‘zlashtirish;
- b) Taqqoslash;

### **Ko‘rsatkichli toifa(Pointer)**

Ko‘rsatkichli toifa ma’lumotlarni ko‘rsatkichlari yoki manzillari (adres) to‘plamini namoyon qiladi, ya’ni ko‘rsatkichlar ma’lumotlarni emas balki bu ma’lumotlar joylashgan xotiradagi manzilni o‘z ichiga oladi. Ko‘rsatkichlar xotirada bori yo‘g‘i 4 bayt joyni egallab, u ko‘rsatayotgan ma’lumotlar ancha katta joyni egallagan bo‘lishi mumkin. Pointer toifasi ma’lumoti ixtiyoriy boshqa biror ma’lumot yoki ma’lumotlar guruhiga yo‘naltirilgan bo‘ladi. Ko‘rsatkichga mumkin bo‘lgan u yoki bu qiymatni o‘zlashtirib, ushbu ko‘rsatkich orqali kerakli ma’lumotga murojatni amalga oshirish mumkin. Pointer toifasidagi ma’lumotlarni qiymatlar to‘plamida bitta maxsus qiymat bo‘lib, uni o‘zlashtirish hech qayerga yo‘naltirilmaganligini ko‘rsatadi, ya’ni nol yoki bo‘sh ko‘rsatkich xisoblanadi. Masalan, C++ tilida bunday qiymat sifatida NULLdan foydalaniлади. Ko‘rsatkichlar ustida amallar quyidagicha bo‘lishi mumkin: biror bir ko‘rsatkichga boshqa ko‘rsatkich qiymatini o‘zlashtirish mumkin yoki boshqa ma’lumot egallab turgan xotira sohasi adresini o‘zlashtirish mumkin. Ko‘rsatkichlar o‘zaro bog‘langan ma’lumotlar tuzilmasini yaratishda va qayta ishlashda katta ahamiyatga ega. Xotirada ko‘rsatkichlarni ifodalash uchun asosan

dasturlash tizimiga mos ravishda manzilni maksimal uzunligicha joy ajratiladi. Ko‘rsatkichlarni qiymati nomanfiy butun sonlar sifatida sohada bitlarni ketma-ketligi ko‘rinishida saqlanadi.C++ tilida ko‘rsatkichli o‘zgaruvchilarni e’lon qilish uchun ularning toifasini aniqlash kerak. Buning uchun ko‘rsatkich xotirada qanaqa toifadagi ma’lumotlarni ko‘rsatayotgan bo‘lsa, ko‘rsatkichli o‘zgaruvchiga ham xuddi shunday toifa beriladi.

```
int a=9;  
int *p=&a;  
float f=4.6; float *d=&f;  
FILE*f=fopen(“talaba.txt”,’r’);
```

### **Foydalanuvchi tomonidan aniqlanadigan toifalar**

#### **Sanaladigan toifalar**

Qiymatlarning o‘zgaruvchan toifalari standartlardan farqliroq yangi toifalarni yaratishga imkon beradi. Bu guruhga sanaladigan va chegaralangan toifalar kiradi.

Qiymatlarning sanaladigan toifalarning bunday atalishiga sabab, ular qat’iy aniqlangan tartibda sanaladigan ko‘rinishda beriladi va xamma qiymatlarning soni qat’iy chegaralangan xamda ko‘rilayotgan toifadagi qiymatlarni qabul qilishi mumkin. Sanaladigan toifa yechilayotgan masalaga qarab foydalanuvchi tomonidan berilishi mumkin.

Sanaladigan toifa konstantalar ro‘yxatidan tashkil topadi. Bu toifadagi o‘zgaruvchilar ro‘yxatidagi ixtiyoriy qiymatni qabul qilishi mumkin. Sanaladigan toifaning umumiy yozilish shakli quyidagicha:

```
enum toifaning nomi {konstantalar ro‘yxati};  
toifaning nomi o‘zgaruvchi nomi;
```

Bu yerda konstanta tushunchasi foydalanuvchi tomonidan berilagan maxsus konstanta ko‘rinishi tushuniladi. Konstantalar ro‘yxati bir-biridan vergul bilan ajratiladi va ular oddiy qavslar ichiga olinadi.

Masalan:

*enum Ranglar{oq, qora, qizil, yashil};*

*Ranglar rang;*

Bu yerda Ranglar – sanaladigan toifaning nomi;oq, qora, qizil, yashil-konstantalar. Rang - o‘zgaruvchi nomi bo‘lib u yuqoridagi konstantalardan ixтиyoriysini qabul qilishi mumkin.

Har bir konstanta tartib raqamiga ega bo‘lib, xisobdan boshlanadi, ya’ni  $oq=0$ ,  $qora=1$ ,  $qizil=2$ ,  $yashil=3$ raqamlarigaega. Konstantalar tartiblangani uchun ularga solishtirish amallari  $<$ ,  $<=$ ,  $==$ ,  $!=$ ,  $>=$ ,  $>$  shuningdek standart funksiyalarni qo‘llash mumkin.

### **Strukturalar**

Strukturalar turli toifadagi maydonlardan tashkil topgan yozuv xisoblanadi. Strukturalarni e’lon qilish uchun struct kalit so‘zi ishlataladi. Undan keyin toifaga nom beriladi va {} qavs ichida maydonlar toifalari va nomlari e’lon qilinadi.

*Struct G{*

*charch;*

*} talaba, talabalar[10];*

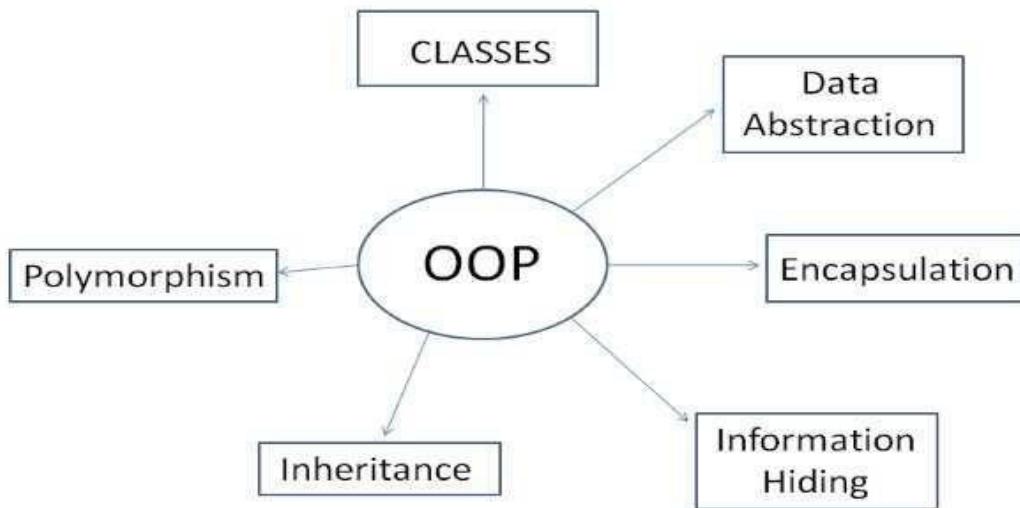
Ushbu toifadagi o‘zgaruvchiyoki massiv elementi maydonlariga murojaat:

- *Jadval\_elementi[indeks].maydon\_nomi=qiymat;*
- Ya’ni, *talabalar[i].ch=‘a’;*

### **Nazorat savollari**

1. Matnli fayllarni binary fayllardan farqi?
2. Ma’lumotlarning qanday toifalarini bilasiz?
3. Butun toifadagi ma’lumotlar ustida qanday amallarni bajarish mumkin?
4. Ma’lumotlarning bul toifasida qanday amallar mavjud?
5. CHAR toifasining tuzilmasi qanday? Belgili toifadan qanday amallarni bajarish mumkin?
6. Ko‘rsatkichli toifa ma’lumoti yordamida nimani hisoblash mumkin?
7. Ma’lumotlarning sanaladigan toifasi degani nima?

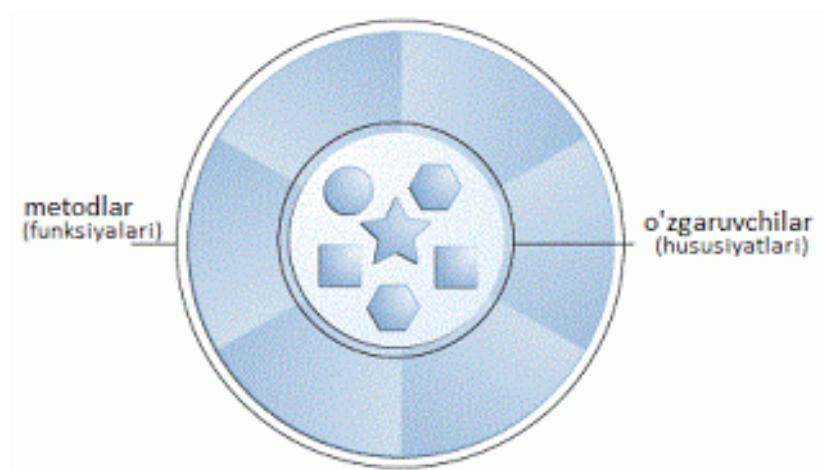
### 3.3-§ OBYEKTGA YO‘NALTIRILGAN DASTURLASH ASOSLARI



Obyekt - bu obyektga yo‘naltirilgan dasturlash (OYD) texnologiyasining eng asosiy kalit tushunchasidir. Atrofga qarang, haqiqiy hayotdagi bir necha obyektlarni ko‘rishingiz mumkin: stol, uy, it, mushuk, televizor va h.k.

Ularning barchasining albatta xususiyatlari va bajaradigan vazifalari (funksiya-lari) bor. Masalan, Mushuk xususiyatlari: rangi, qorni to‘qligi, yoshi, jinsi; funksiyalari: ovqat yeyishi, myovlashi, yurishi, sichqon tutishi. Mashina, xususiyatlari: tezligi, rangi, nomi, narxi; funksiyalari: yurishi, to‘xtashi, oyna artgichlarining ishlashi, eshiklarning ochilib yopilishi v.h.k. Bu kabi hayotiy misollarning xususiyatlari va funksiyalarini aniqlash OYD nuqtai nazaridan fikrlashning eng zo‘r ko‘rinishidir.

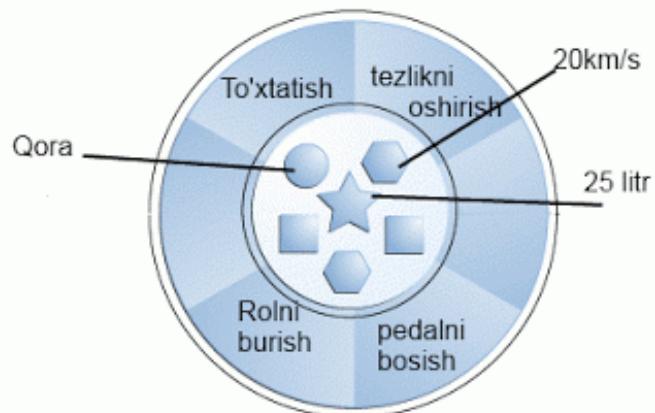
Bir daqiqaga to‘xtang va hozirda atrofingizdagи biror narsalarni analiz qiling. Har bir obyekt uchun o‘zingizdan so‘rang: "Bu obyektning qanday xususiyatlari bor?", "Qanday vazifalarni bajaradi?" kabi. Va kuzatish natijalaringizni yozib oling, sezgan bo‘lsangiz tuziladigan ro‘yxat obyektning murakkabligiga qarab ko‘payib boradi. Kompyuter indikatorining 2 ta xususiyati bor o‘chiq va yoniq; funksiyalari esa yonish va o‘chish. Bu barcha kuzatishlar OYD dunyosiga o‘tkazish mumkin.



### **Dasturlashdagi obyekt.**

Dasturlashdagi obyekt (bundan keyin oddiygina obyekt deb ketiladi) ham haqiqiy hayotdagi obyektlarga o'xshash: Ular ham qandaydir xususiyatlari va bajaradigan funksiyalardan iborat bo'ladi. Obyektning xususiyatlari har xil dasturiy o'zgaruvchilardan iborat bo'ladi va ularning o'zgartirish uchun qandaydir funksiyalar bajariladi. Bunday funksiyalar bilan o'zgaruvchilarning holatini berkitish mumkin ya'ni aynan o'sha o'zgaruvchini tashqaridan o'zgartirish uchun albatta maxsus funksiyadan foydalanish kerak bo'ladi. Bu jarayon "Inkapsulatsiya" deb atalib, OYDning eng muxim tushunchalaradian biridir. Hech e'tibor berganmisiz dorilarda ham shu termin ishlataladi ya'ni kapsula (ustidan maxsus modda bilan o'ralgan dorilar), buni misolni Enkapsulatsiya jarayoni esda yaxshi qolishi va tushunarli bo'lishi uchun keltirdim.

Tasavvur qiling, mashinani dasturlash obyekti sifatida modellashtiramiz:



Uning o‘zgaruvchilari (hozirgi tezligi, qolgan benzini, va h.k) va uning funksiyalari (to‘xtatish, tezlikni oshirish, rulni burish, va h.k.). Bu yerda uning bakidagi benzini yurishi tufayli kamayib boradi demak uning qiymatining o‘zgarishi 0dan bakning sig‘imigacha bo‘ladi, yoki uning tezligi ham shu kabi aynan qaysidir funksiyalarning amalga oshirilishi orqali u ham 0 dan maksimal tezligigacha o‘zgarishi mumkin. Bulardan tashqari mashinaning ba’zi xususiyatlari borki ular o‘zgarmasligi mumkin, masalan, rangi. Demak, ko‘rinib turiptiki mashina ham o‘z navbatida bir necha mayda obyeklardan iborat bo‘ladi. Va albatta ularni kodda yozganda ham alohida obyekt sifatida ifodalash kerak bu orqali nimalarga erishish mumkin:

1. **Qismlilik:** Har bir obyektga tegishli bo‘lgan kodlar alohida-alohida, boshqa obyeklarga bog‘liq bo‘lmagan holda boshqarish imkoniyatiga ega bo‘lamiz. Bu hammasi emas, tasavvur qiling mashina obyektini ifodalovchi kodni bo‘lmasdan faqat bitta faylda ifodaladik; bu esa murakkabligiga qarab yuzlab hatto minglab qatorli kod bo‘lishi mumkin. Undan biror narsani topib-o‘zgartrish ancha mashaqqat bo‘ladi.

2. **Qayta foydalanish:** Yana boshqa plyus tarafi biz bo‘laklagan mashinaning detallarini boshqa obyektlarda ham ishlatishimiz mumkin. Masalan, 2 xil mashina ularning shunday qismlari borki aynan bir xil, ana o‘shalar uchun ikki marta alohida kod yozmasdan, bitta yozganimizni qayta ishlatishimiz mumkin.

3. **Uzib-ulanuvchanligi:** buni tushunish uchun yuqoridagi misoldan foydlanamiz, aytaylik, mashinaning biror qismi ishlamayapti, xo‘sh nima qilinadi? yoki ishlab turgan boshqasiga almashtiramiz yoki tuzatamiz. Mashinaning biror bolti buzilsa uni boshqa ishlab turgani bilan almashtirasiz yoki tuzatamiz lekin mashinani almashtirmaymiz.

Bu yuqoridagi ma’lumotlar yaxshi tushunarli bo‘lmagan bo‘lsa. Keyingi maqolalarda bularni kodlar bilan yozib tushuntirib boriladi.

**C ++ OYD nima? (Object Oriented Programming = OOP)**

OOP (Object Oriented Programming - Obyektga yo‘naltirilgan dasturlash)

- bu biron bir maqsadga yo‘naltirilgan dasturlash degan ma’noni anglatadi.

Protseduraviy dasturlash - bu ma’lumotlarga ishlov beradigan protseduralar yoki funksiyalarni yozish, obyektga yo‘naltirilgan dasturlash esa ma’lumot va funksiyalarni o‘z ichiga olgan obyektlarni yaratish haqida.

Obyektga yo‘naltirilgan dasturlash protsessual dasturlashdan bir qator afzalliklarga ega:

- OYDtezroq va bajarilishi osonroq

- OYD dasturlarning aniq tuzilishini ta’minlaydi

- OYD C ++ kodini DRY "Don’t Repeat Yourself" saqlashga yordam beradi va kodni saqlash, o‘zgartirish va disk raskadrovka qilishni osonlashtiradi.

- OYD kodni kam va ishlab chiqarish vaqtini qisqartirgan holda to‘liq qayta ishlatiladigan ilovalarni yaratishga imkon beradi.

Maslahat: "O‘zingizni takrorlamang" (DRY) printsipi kodning takrorlanishini kamaytirishga qaratilgan. Dastur uchun odatiy bo‘lgan kodlarni chiqarib, ularni bitta joyga joylashtiring va takrorlash o‘rniga ularni qayta ishlatishingiz kerak.

### ***C ++ Sinflar va obyektlar nima?***

Sinflar va obyektlar obyektga yo‘naltirilgan dasturlashning ikkita asosiy jihat. Sinf va obyektlar o‘rtasidagi farqni ko‘rish uchun quyidagi rasmga qarang:

Boshqa misol:

Shaxsiy obyektlar yaratilganda, ular barcha o‘zgaruvchilar va funksiyalarni sinfdan meros qilib oladilar.

C ++ - bu obyektga yo‘naltirilgan dasturlash tili. C++ dasturlash tilida hamma narsa uning xususiyatlari va usullari bilan bir qatorda sinflar va obyektlar bilan bog‘liq.

Sinf - bu bizning dasturimizda foydalanishimiz mumkin bo‘lgan foydalanuvchi belgilaydigan ma’lumot turi va u obyekt tuzuvchisi yoki obyektlarni yaratish uchun "reja" sifatida ishlaydi.

## Sinf yaratish.

Sinf yaratish uchun class kalit so‘zdan foydalanib, "MyClass" nomli sinf yaratamiz.

```
class MyClass {    // class
    public:        // ochiqlik siyosati
    int myNum;    // Attribute (int tipiga tegishli)
    string myString; // Attribute (string tipiga tegishli)
};
```

## Misolni tushuntirish.

- **class** Kalit so‘z **MyClass** deb atalgan bir sinf yaratish uchun ishlataladi.
  - **Public** Kalit so‘z bir bo‘lib **kirish belgisi** ifodalarydi. Bu degani class dan tashqarida ham attributlardan foydalanish mumkin.
  - Sinf ichida butun son **myNum** va satr o‘zgaruvchisi mavjud **myString**. O‘zgaruvchilar sinf ichida e’lon qilinganida, ular **atributlar** deb nomlanadi.
  - Nihoyat, sinf ta’rifini nuqta-vergul bilan tugatiladi.

## Obyektni yaratish.

C++ dasturlash tilida biz sinf yaratdik, **MyClass** nomli class yaratdik, shuning uchun bundan foydalanib obyekt yaratamiz. Obyektni yaratish uchun **MyClass** sinf nomini, so‘ngra obyekt nomini ko‘rsating.

```
#include <iostream>
#include <string>
using namespace std;
class MyClass {
public:
    int myNum;
    string myString;
```

```

};

int main() {
    MyClass myObj;
    myObj.myNum = 15;
    myObj.myString = "Matn";
    cout << myObj.myNum << "\n";
    cout << myObj.myString;
    return 0;
}

```

15

Matn

### **Bir nechta obyektlar.**

Siz bitta sinfning bir nechta obyektlarini yaratishingiz mumkin:

```

#include <iostream>
#include <string>
using namespace std;
class Car {
public:
    string brand;
    string model;
    int year;
};
int main() {
    Car carObj1;
    carObj1.brand = "BMW";
    carObj1.model = "X5";
    carObj1.year = 1999;
    Car carObj2;
}

```

```

carObj2.brand = "Ford";
carObj2.model = "Mustang";
carObj2.year = 1969;
cout << carObj1.brand << " " << carObj1.model << " "
<< carObj1.year << "\n";
cout << carObj2.brand << " " << carObj2.model << " "
<< carObj2.year << "\n";
return 0;
}

```

*BMW X5 1999*

*Ford Mustang 1969*

Hozirda, siz **public**bizning barcha sinf misollarimizda paydo bo‘lgan kalit so‘z bilan juda yaxshi tanishsiz:

```

#include <iostream>
using namespace std;
class MyClass {
public:
    int x;
};
int main() {
    MyClass myObj;
    myObj.x = 15;
    cout << myObj.x;
    return 0;
}

```

## **Public**

Kalit so‘z bir bo‘lib **kirish ma’lumoti**. Kirish spetsifikatorlari sinf a’zolariga (atributlar va usullarga) qanday kirish mumkinligini belgilaydi. Yuqoridagi misolda, a’zolar **public**- bu kodni tashqaridan kirish va o‘zgartirish mumkinligini anglatadi.

Dasturlash xavfsizlik birinchi o‘rinda turadi. Biz aynan tashqi kirishni to‘xtatish kerak bo‘lsa nima qilamiz? buning javobi quyidagi ma’lumot turlari bilan izohlanadi.

C ++ da uchta kirish spetsifikatorlari mavjud:

- **public** - A’zolarga sinfdan tashqari kirish mumkin.
- **private** - A’zolarga sinfdan tashqarida kirish (yoki ko‘rish) mumkin emas.

• **protected**- A’zolarga sinfdan tashqariga kirish mumkin emas, ammo ularga meros qolgan sinflarda kirish mumkin. Keyinchalik **merosxo‘rlik** haqida ko‘proq bilib olasiz. Quyidagi misolda biz **public** va **private** a’zolar o‘rtasidagi farqni ko‘rsatamiz :

```
#include <iostream>
using namespace std;
class MyClass {
public:
    int x;
private:
    int y;
};
int main() {
    MyClass myObj;
    myObj.x = 25;
```

```

myObj.y = 50;
return 0;
}

In function 'int main()':
Line 8: error: 'int MyClass::y' is private
Line 14: error: within this context
error: y is private

```

**private** Agar kirish spetsifikatsiyasini ko'rsatmasangiz, avtomatik tarizda **private** bo'ladi.

```

s MyClass {
    int x; // Private attribute
    int y; // Private attribute

```

Assalomu alaykum bugungi darsimiz ma'lumotlar xavfsizligini ta'minlash haqida gaplashamiz. Ko'pchilik ma'lumotlarni foydalanuvchidan yashirish kerak bo'lib qoladi. Bunda bizda C++ dasturlash tilida bizga **Enkapsulatsiya** yordam beradi. Bu atama bo'yicha quyidagicha tushunchaga ega bo'lishiningiz mumkin.

**Encapsulation** (Inkapsulatsiya)ning ma'nosi, "sezgir" ma'lumotlar foydalanuvchilardan yashirilganligiga ishonch hosil qilishdir. Bunga erishish uchun siz sizinf o'zgaruvchilari atributlarini e'lon qilishingiz kerak **private** (sinf tashqarisidan kirish mumkin emas).

**Inkapsulatsiya Uchun Maxsus.**

```

#include <iostream>
using namespace std;
class Employee {
private:
    int salary;
public:
    void setSalary(int s) {
        salary = s;
    }
    int getSalary() {
        return salary;
    }
};
int main() {
    Employee myObj;
    myObj.setSalary(50000);
    cout << myObj.getSalary();
    return 0;
}

```

50000

### Yuqoridagi misolni tushuntirish.

Kirish cheklangan **salary** o‘zgaruvchi hisoblanadi chunki **private** deb e’lon qilingan.

Umumiylar **setSalary()** usul (s) parametrni oladi va uni atributga **salary** (ish haqi = s) tayinlaydi.

Umumiylar **getSalary()** usul xususiy **salary** atributning qiymatini qaytaradi.

Ichkarida **main()** biz Employeesinf obyektini yaratamiz. Endi biz **setSalary()** xususiy usulning qiymatini belgilash uchun usuldan foydalanishimiz

mumkin . Keyin biz **getSalary()** obyektni qiymatini qaytarish uchun usuli ya’ni funksiya hisoblanadi.

### **Nima uchun inkapsulatsiya?**

- Xavfsizlik uchun juda muhim. Yashirin ma’lumotlarni saqlash va undan foydalanuvchilarni cheklash.
- **Class** ni ichida xavfsizlikga ega bo‘lgan attributga biz murojaat qila olamiz.

### **§ 3.5. C++ Merosxurlik (Sinf/Class)**

Assalomu alaykum bugun siz bilan merosxurlik haqida gaplashamiz. Class ichida va undan tashqari foydalanish va uning qulayliklari haqida gaplashib olamiz.

#### **Meros olish**

- **derived class** (bola) - *boshqa sinfdan meros qolgan sinf.*
- **base class** (ota-on) - *meros bo‘lib o‘tgan sinf.*

Sinfdan meros olish uchun: belidan foydaniladi. Quyidagi misolda, **Car** sinf (bola) atributlar va usullarni **Vehicle** sinfdan (ota-onadan) meros qilib oladi:

```
#include <iostream>
#include <string>
using namespace std;
// Base class
class Vehicle {
public:
    string brand = "Ford";
    void honk() {
        cout << "Tuut, tuut! \n";
    }
};
```

```

// Derived class

class Car: public Vehicle {
public:
    string model = "Mustang";
}
int main() {
    Car myCar;
    myCar.honk();
    cout << myCar.brand + " " + myCar.model;
    return 0;
}

```

Tuut, tuut!

Ford Mustang

**“Meros”** - bu odatda bitta class dan qayta qayta foydalanish imkoniyatini beradi. Yangi sinf yaratishda sinf attributlaridan foydalanish imkoniyati.

### *Ko‘p darajali meros*

```

#include <iostream>
using namespace std;
class MyClass {
public:
    void myFunction() {
        cout << "Hello World.Nilufar." ;
    }
};
class MyChild: public MyClass {

```

```

};

class MyGrandChild: public MyChild {

};

int main() {

    MyGrandChild myObj;

    myObj.myFunction();

    return 0;

}

```

*Hello World.Nilufar.*

### ***Ko‘p meros.***

Sinf shuningdek, vergul bilan ajratilgan ro‘yxat yordamida bir nechta asosiy sinflardan olinishi mumkin:

```

#include <iostream>

using namespace std;

class MyClass {

public:

    void myFunction() {

        cout << "Hello World. Nilufar.\n" ; }

};

class MyOtherClass {

public:

    void myOtherFunction() {

        cout << "Hello World.Nilufar.\n" ; }

};

class MyChildClass: public MyClass, public MyOtherClass {

```

```

};

int main() {
    MyChildClass myObj;
    myObj.myFunction();
    myObj.myOtherFunction();
    return 0;
}

```

*Hello World.Nilufar.*

*Hello World.Nilufar.*

## **Meros Huquqi.**

Siz **Kirish** xususiyatlarini aniqlash bo‘limidan C++ da uchta aniqlovchi mavjudligini bilib oldingiz . Hozirgacha biz faqat foydalanganmiz **public** (sinf a’zolariga sinf tashqarisidan kirish mumkin) va **private** (a’zolarga faqat sinf ichida kirish mumkin). Uchinchi spetsifikatorga **protected** o‘xshash **private**, ammo unga **meros qilib olingan** klassda kirish mumkin:

```

#include <iostream>
using namespace std;
class Employee {
protected: // Protected access specifier
    int salary;
};
class Programmer: public Employee {
public:

```

```
int bonus;  
void setSalary(int s) {  
    salary = s;  
}  
int getSalary() {  
    return salary;  
}  
};  
int main() {  
  
    Programmer myObj;  
    myObj.setSalary(50000);  
    myObj.bonus = 15000;  
    cout << "Salary: " << myObj.getSalary() << "\n";  
    cout << "Bonus: " << myObj.bonus << "\n";  
    return 0;  
}
```

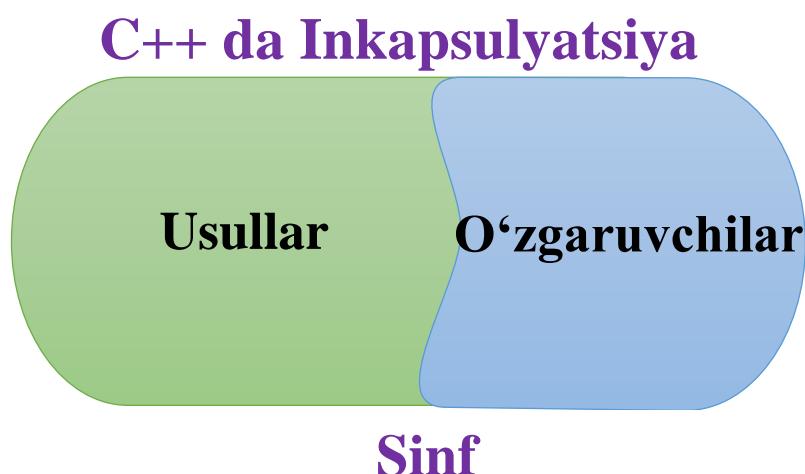
## IV BOB. INKAPSULYATSIYA, MEROSXO‘RLIK VA POLIMORFIZM. KONTEYNER SINFLAR.

### 4.1-§ INKAPSULYATSIYA VA MEROSXO‘RLIK. ASOSIY SINF A’ZOLARIGA MUROJAATNI BOSHQARISH.

**Reja:**

- 1. Inkapsulyatsiya**
- 2. Asosiy sinf a’zolariga murojaatni boshqarish**
- 3. Ruxsat funksiyalari**

Assalomu alaykum aziz tinglovchilar. Bugun biz ob’ektga yo‘naltirilgan dasturlashning (OYD) asosiy tamoyillari – inkapsulyatsiya va merosxo‘rlikni va ularning C++ tilida qandav amalga oshirilishini ko‘rib chiqamiz.



*4.1 – rasm. C++ da Inkapsulyatsiya*

**Inkapsulyatsiya: ta’rifi va tamoyillari.**

Inkapsulyatsiya - bu sinf ichida ushbu ma’lumotlarda ishlaydigan ma’lumotlar va usullarni birlashtirgan OYD tushunchasi. U ichki amalga oshirish tafsilotlarini yashirishni ta’minlaydi va ob’ekt bilan o‘zaro aloqa qilish uchun boshqariladigan interfeysni ko‘rsatadi.

## **C++ da inkapsulyatsiya tamoyillari:**

Maxfiylik: Shaxsiy deb e'lon qilingan sinf a'zolariga faqat sinfning o'zida kirish mumkin. Bu tashqaridan to'g'ridan-to'g'ri kirishni oldini oladi va ma'lumotlar xavfsizligini ta'minlaydi.

Yozishdan himoyalangan (faqat o'qish uchun): Ma'lumotlarga kirishni ta'minlaydigan usullar const deb e'lon qilinishi mumkin, ya'ni ular ob'ekt holatini o'zgartirmaydi.

Obyektga yo'naltirilgan dasturlashda inkapsulyatsiya (yoki ma'lumotni yashirish) ob'ektni amalga oshirish tafsilotlarini yashirish jarayonidir. Foydalanuvchilar ob'ektga umumiy interfeys orqali kirishadi.

C++ da inkapsulyatsiya kirish spetsifikatorlari orqali amalga oshiriladi. Odatda, sinfning barcha a'zo o'zgaruvchilari shaxsiydir (amalga oshirish tafsilotlarini yashirish) va ko'pchilik usullar ochiqdir (foydalanuvchiga interfeysni ko'rsatish). Foydalanuvchilardan umumiy interfeysdan foydalanishni talab qilish oddiygina a'zo o'zgaruvchilarni oshkor qilishdan ko'ra og'irroq tuyulishi mumkin bo'lsa-da, u aslida kodning qayta ishlatilishi va barqarorligini yaxshilaydigan ko'plab foydali imtiyozlarni beradi.

## **Inkapsulyatsiya xususiyatlari**

Inkapsulyatsiyaning quyidagi xususiyatlari mavjud:

1. Biz to'g'ridan-to'g'ri sinfdan biron bir funktsiyaga kira olmaymiz. Ushbu funktsiyaga kirish uchun bizga ushbu sinfning a'zo o'zgaruvchilaridan foydalanadigan ob'ekt kerak.
2. Biz sinf ichida yaratgan funksiya faqat a'zo o'zgaruvchilardan foydalanishi kerak, shundan keyingina u inkapsulyatsiya deb ataladi.
3. Agar sinfning a'zo o'zgaruvchisidan foydalanadigan sinf ichida funksiya yaratmasak, u holda biz uni inkapsulyatsiya deb atamaymiz.
4. Inkapsulyatsiya ma'lumotlar va usullarni bиргаликда guruhashlар орқали о'qилиши, barqarорлиги ва xavfsizligini yaxshilaydi.

5. Bu bizning ma'lumotlar elementlarini o'zgartirishni boshqarishga yordam beradi.

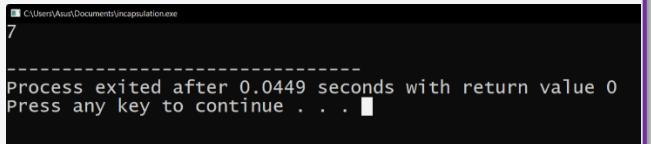
### **Inkapsullangan sinflardan foydalanish osonroq va dasturlaringizning murakkabligini kamaytiradi.**

To'liq inkapsullangan sınıf bilan siz faqat qaysi usullardan foydalanish mumkinligini, ular qanday argumentlarni olishini va ular qanday qiymatlarni qaytarishini bilishingiz kerak. Sinf ichki tarzda qanday amalga oshirilishini bilishingiz shart emas. Misol uchun, nomlar ro'yxatini o'z ichiga olgan sınıf dinamik massiv, C uslubidagi satrlar, std::array, std::vector, std::map, std::list yoki boshqa ma'lumotlar strukturasi yordamida amalga oshirilishi mumkin. Ushbu sınıfdan foydalanish uchun uni amalga oshirish tafsilotlarini bilishingiz shart emas. Bu sizning dasturlaringizning murakkabligini sezilarli darajada kamaytiradi va shuningdek, mumkin bo'lgan xatolar sonini kamaytiradi. Bu inkapsulyatsiyaning asosiy afzalligi.

#### **Inkapsullangan sinflarni o'zgartirish osonroq.**

Quyidagi oddiy misolni ko'rib chiqamiz:

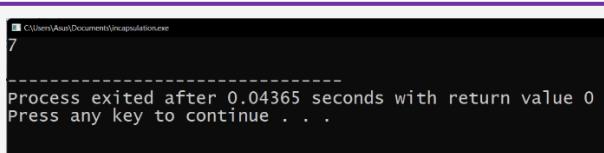
```
1 #include <iostream>
2 class Values
3 {
4 public:
5     int m_number1;
6     int m_number2;
7     int m_number3;
8 };
9 int main()
10 {
11     Values value;
12     value.m_number1 = 7;
13     std::cout << value.m_number1 << '\n';
14 }
```



Ushbu dastur yaxshi ishlayotgan bo'lsa-da, m\_number1 nomini o'zgartirishga yoki ushbu o'zgaruvchining turini o'zgartirishga qaror qilsak nima bo'ladi? Biz nafaqat ushbu dasturni, balki Values sınıfidan foydalanadigan ko'pgina dasturlarni ham buzamiz!

Inkapsulyatsiya sinflarni ishlataligani barcha dasturlarni buzmasdan amalga oshirish usullarini o‘zgartirish imkoniyatini beradi. Mana yuqoridagi sinfning inkapsullangan versiyasi, lekin u m\_number1 ga kirish usullaridan foydalanadi:

```
1 #include <iostream>
2 class Values
3 {
4 private:
5     int m_number1;
6     int m_number2;
7     int m_number3;
8 public:
9     void setNumber1(int number) { m_number1 = number; }
10    int getNumber1() { return m_number1; }
11 };
12 int main()
13 {
14     Values value;
15     value.setNumber1(7);
16     std::cout << value.getNumber1() << '\n';
17 }
```



## C++ da kirish spetsifikatorlari

### Xususiy (private):

```
1 class MyClass {
2 private:
3     // sinf a'zolari};
```

### Ximoyalangan (protected):

```
1 class MyClass {
2 protected:
3     // sinf a'zolari };
```

### Ochiq (public):

```
1 class MyClass {
2 public:
3     // sinf a'zolari };
```

## Ruxsat funksiyalari (getter va setterlar)

Sinfning qanaqaligiga qarab, sinfning xususiy a'zo o'zgaruvchilari qiymatlarini olish/o'rnatish imkoniyatiga ega bo'lish (sinf bajaradigan kontekstda) mos bo'lishi mumkin.

Ruxsat olish funksiyasi qisqa umumiy funksiya bo'lib, uning vazifasi xususiy sinf a'zosi o'zgaruvchisining qiymatini olish yoki o'zgartirishdan iborat.

Masalan:

```
1 class MyString
2 {
3     private:
4         char *m_string; // Satrni dinamik tarzda belgilaymiz
5         int m_length; // satr uzunligini kuzatish uchun o'zgaruvchidan foydalanamiz
6
7     public:
8         int getLength() { return m_length; } // m_length qiymatini olish uchun
9             ruxsat funksiyasi
9 };
```

Bu yerda getLength() bor yo'g'i m\_length qiymatini qaytaruvchi yordamchi funksiyadir.

Kirish funksiyalari odatda ikki xil bo'ladi:

**Getterlar** – xususiy sinf a'zo-o'zgaruvchilari qiymatlarini qaytaradigan funksiyalardir;

**Setterlar** – xususiy sinf a'zo-o'zgaruvchilarga qiymatlarni belgilash imkonini beruvchi funksiyalardir.

Quyida o'zining yopiq o'zgaruvchi-a'zolaridan foydalanish uchun getter va setterlardan foydalanadigan sinf misoli:

```

1  class Date
2  {
3      private:
4          int m_day;
5          int m_month;
6          int m_year;
7  public:
8      int getDay() { return m_day; } // геттер для day
9      void setDay(int day) { m_day = day; } // сеттер для day
10
11     int getMonth() { return m_month; } // геттер для month
12     void setMonth(int month) { m_month = month; } // сеттер для month
13     int getYear() { return m_year; } // геттер для year
14     void setYear(int year) { m_year = year; } // сеттер для year
15 }

```

## C++ da merosxo‘rlik

Sinfning boshqa sinfdan xossa va xususiyatlarni olish qobiliyati merosxo‘rlik deb ataladi. Vorislik obyektga yo‘naltirilgan dasturlashning eng muhim xususiyatlaridan biridir.

Merosxo‘rlik - bu mavjud sinflardan yangi sinflar yaratiladigan funksiya yoki jarayon. Yaratilgan yangi sinf "hosil sinf" yoki "bola sinf" deb nomlanadi va mavjud sinf "asosiy sinf" yoki "ota sinf" deb nomlanadi. Endi olingan sinf asosiy sinfdan meros bo‘lib hisoblanadi.

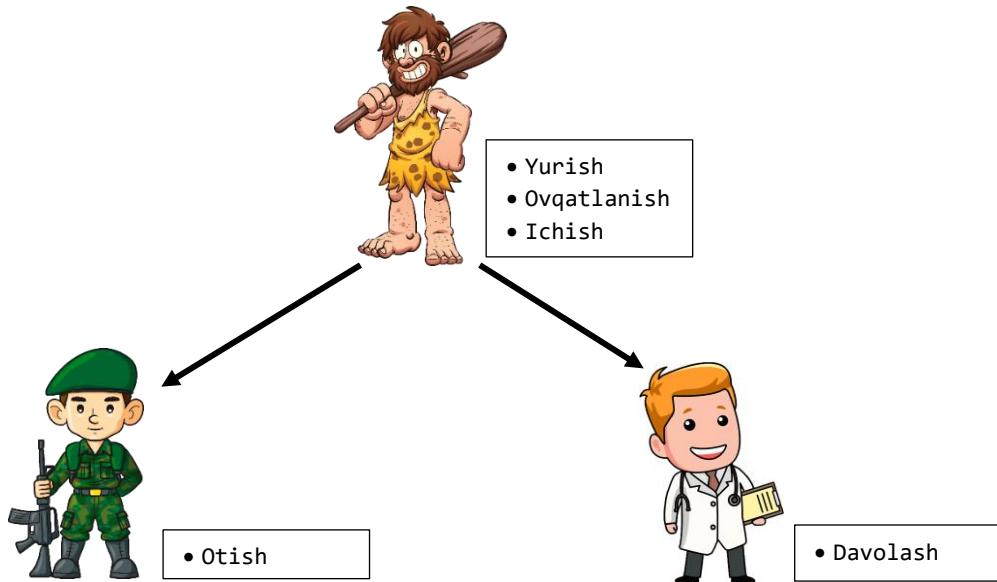
Olingan sinf asosiy sinfdan meros bo‘ladi deganda, bola sinf asosiy sinfning xususiyatlarini o‘zgartirmasdan, asosiy sinfning barcha xususiyatlarini meros qilib oladi va o‘ziga yangi xususiyatlar qo‘shishi mumkinligini anglatadi. Olingan sinfdagi bu yangi xususiyatlar asosiy sinfga ta’sir qilmaydi. Bola sinf - bu ota sinf uchun maxsus sinf hisoblanadi.

**Subsinf:** Boshqa sinfdan xususiyatlarni meros qilib olgan sinf subsinf yoki hosila sinf deb ataladi.

**Supersinf:** Xususiyatlari kichik sinf tomonidan meros bo‘lib qolgan sinf asosiy sinf yoki supersinf deb ataladi.

Meros mavzusi quyidagi kichik mavzularga bo‘linadi:

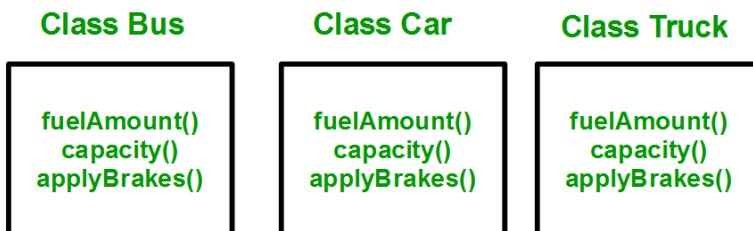
- Merosdan nima uchun va qachon foydalanish kerak?
- Meros olish usullari
- Merosxo‘rlik turlari



#### 4.2 – rasm. Merosxo‘rlikga misol

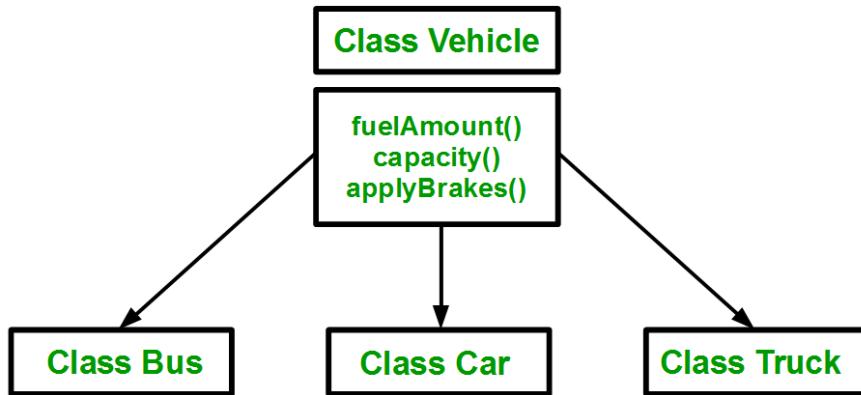
##### Nega va qachon merosxo‘rlikdan foydalanish kerak?

Bir guruh transport vositalarini ko‘rib chiqing. Bus, Car va Truck uchun darslar yaratishingiz kerak. `fuelAmount()`, `capacity()`, `applyBrakes()` usullari barcha uch sinf uchun bir xil bo‘ladi. Agar biz ushbu sinflarni merosdan qochib yaratadigan bo‘lsak, unda biz quyidagi rasmda ko‘rsatilganidek, ushbu barcha funktsiyalarni uchta sinfning har biriga yozishimiz kerak bo‘ladi:



#### 4.3 – rasm. Bir xil funktsiyalarga ega sinflarni e’lon qilish

Yuqoridagi jarayon natijasida bir xil kod 3 marta takrorlanishini aniq ko‘rishingiz mumkin. Bu xato va ma’lumotlarning ortiqcha bo‘lish ehtimolini oshiradi. Bunday holatlarning oldini olish uchun merosdan foydalaniladi. Agar biz `Vehicle` sinfini yaratsak va unga ushbu uchta funktsiyani yozsak va qolgan sinflarni avtomobil sinfidan meros qilib olsak, biz shunchaki ma’lumotlarning takrorlanishini oldini olishimiz va qayta foydalanishni yaxshilashimiz mumkin. Quyidagi diagrammaga qarang, bu yerda uchta sinf avtomobil sinfidan meros bo‘lib o‘tadi:



#### 4.4 – rasm. Kerakli funksiyalar va kichik sinflarga ega asosiy sinf deklaratsiyasi

Merosxo‘rlikdan foydalaniib, biz funksiyalarni uch marta emas, faqat bir marta yozishimiz kerak, chunki qolgan uchta sinfni asosiy sinfdan (Vehicle) meros qilib oldik.

C++ da merosxo‘rlikni amalga oshirish: Asosiy sinfdan meros bo‘ladigan kichik sinf yaratish uchun biz quyidagi sintaksisga amal qilishimiz kerak.

Olingan sinflar: Olingan sinf asosiy sinfdan olingan sinf sifatida aniqlanadi.

Sintaksis:

```

1 class <derived_class_name> : <access-specifier> <base_class_name>
2 {
3     /body
4 }
  
```

Yuqorida,

class - bu yangi sinf yaratish uchun kalit so‘z,

derived\_class\_name - asosiy sinfga kirish spetsifikatsiyasini meros qilib oladigan yangi sinf nomi.

access-specifier - shaxsiy, ommaviy yoki xavfsiz. Agar hech biri belgilanmagan bo‘lsa, odatta PRIVATE

base-class-name - asosiy sinf nomi

**Eslatma:**

Agar asosiy sinf olingan sinf tomonidan xususiy meros qilib olingan bo‘lsa, asosiy sinfning umumiyligi a’zolari olingan sinfning shaxsiy a’zolariga aylanadi va shuning uchun asosiy sinfning umumiyligi a’zolariga faqat olingan sinfning a’zo funksiyalari orqali kirish mumkin. Ular olingan sinf ob’ektlari uchun mavjud emas.

Boshqa tomondan, asosiy sinf hosila sinf tomonidan ommaviy meros bo‘lib o‘tganda, asosiy sinfning ochiq a’zolari ham hosila sinfning umumiy a’zolariga aylanadi. Shuning uchun, asosiy sinfning umumiy a’zolaridan olingan sinf ob’ektlari, shuningdek, hosila sinfning a’zo funktsiyalari uchun foydalanish mumkin.

```

1 #include <iostream>
2 using namespace std;
3
4+ class Person {
5     int id;
6     char name[100];
7
8 public:
9+     void set_p() {
10         cout << "Enter the Id: ";
11         cin >> id;
12         cout << "Enter the Name: ";
13         cin >> name; }
14
15+     void display_p() {
16         cout << endl << "Id: " << id << "\nName: " << name << endl; } };
17+ class Student : private Person {
18     char course[50];
19     int fee;
20 public:
21+     void set_s() {
22         set_p();
23         cout << "Enter the Course Name: ";
24         cin >> course;
25         cout << "Enter the Course Fee: ";
26         cin >> fee; }
27
28+     void display_s() {
29         display_p();
30         cout << "Course: " << course << "\nFee: " << fee << endl; }};
31+ int main() {
32     Student s;
33     s.set_s();
34     s.display_s();
35     return 0; }
```

Enter the Id: 101  
Enter the Name: Dev  
Enter the Course Name: GCS  
Enter the Course Fee:70000  
  
Id: 101  
Name: Dev  
Course: GCS  
Fee: 70000

## Nazorat savollari

1. Inkapsulyatsiya nima?
2. Modifikatorlar nima va qayerda qo‘llaniladi?
3. Gettarning vazifasi nimadan iborat?
4. Merosxo‘rlikning ustunlik jihatlarini keltiring.
5. Kodning protected qismiga tashqi sinfdan murojaat qilish mumkinmi?

## 4.2-§ POLIMORFIZM

**Reja:**

- 1. Polimorfizm va uning turlari**
- 2. Virtual funksiya**
- 3. Abstrakt sinf**

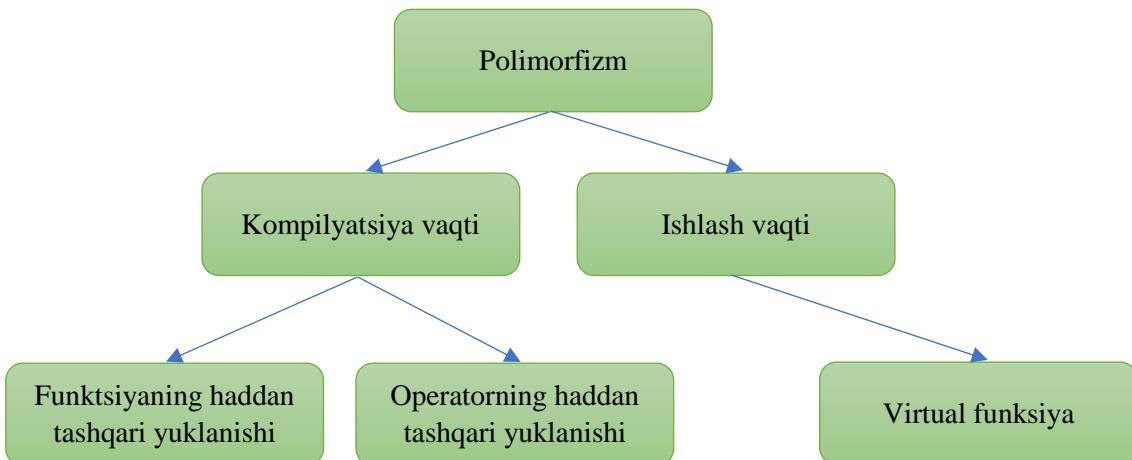
“Polimorfizm” so‘zi ko‘p shakllarning mavjudligini anglatadi. Oddiy so‘zlar bilan aytganda, biz polimorfizmni xabarning bir nechta shaklda paydo bo‘lish qobiliyati deb ta’riflashimiz mumkin. Polimorfizmning haqiqiy namunasi - bir vaqtning o‘zida turli xil xususiyatlarga ega bo‘lishi mumkin bo‘lgan shaxs. Erkak bir vaqtning o‘zida ota, er va xodimdir. Shunday qilib, bir odam turli vaziyatlarda turli xil xatti-harakatlarni namoyon qiladi. Bunga polimorfizm deyiladi. Polimorfizm ob’ektga yo‘naltirilgan dasturlashning muhim xususiyatlaridan biri hisoblanadi.

Obyektga abstrakt darajada qarash xususiyati. Masalan, turli xil eshiklar mavjud: xonaga kirish eshigi, avtomobil eshigi, muzlatgich eshigi, shkaf eshigi. Bularning barchasi bir biridan ishlatilish sohasi, tuzulishi, shakli bilan farq qiladi. Lekin barchasini umumiy qilib eshik deb qarash mumkin. Polimorfizm turli xil obyektlar bilan bir xil uniformada ishlash imkoniyatini beradi.

Polimorfizm orqali bir jarayonni turli yo‘llar bilan tashkillashtirishimiz mumkin. Polimorfizm so‘zi yunoncha ikki so‘zning birikmasidan tashkil topgan «poly» — Ko‘p va «morphs» — formalar. Polimorfizm ham ko‘p formalar degan ma’noni anglatadi.

### **Polimorfizm turlari**

- Kompilyatsiya vaqtida polimorfizmi
- Bajarilish muhiti polimorfizmi



#### **4.2 – rasm. Polimorfizm ierarxiyası**

#### **Funksiyani haddan tashqari yuklash**

Agar bir xil nomga ega, ammo parametrlari har xil bo‘lgan bir nechta funksiyalar mavjud bo‘lsa, u holda funksiyalar haddan tashqari yuklangan deb ataladi, shuning uchun u funksiyani haddan tashqari yuklash deb ataladi. Argumentlar sonini o‘zgartirish yoki/va argumentlar turini o‘zgartirish orqali funksiyalarni ortiqcha yuklash mumkin. Oddiy so‘z bilan aytganda, bu bir xil nomdagi bir nechta vazifalarni bir xil funksiya nomi ostida sanab o‘tilganda, lekin turli parametrlarga ega bo‘lgan bir nechta funksiyalarni ta’minlaydigan obyektga yo‘naltirilgan dasturlashning xususiyati. Funksiyani haddan tashqari yuklashda amal qilish kerak bo‘lgan funksiyani ortiqcha yuklash qoidalari mavjud.

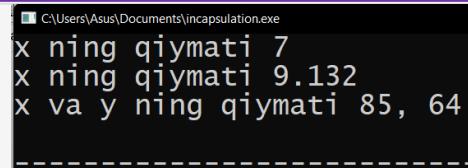
#### **Haddan tashqari yuklanishning ta’rifi**

Kompilyatsiya qilingan vaqtdagi polymorfizm "haddan tashqari yuklanish" deb nomlanadi. Haddan tashqari yuklanish polymorfizm tushunchasidan hosil bo‘lganligi sababli, "ko‘p usullar uchun umumiylar" interfeysi ni ta’minlaydi. Bu shuni anglatadiki, agar funksiya haddan tashqari yuklangan bo‘lsa, u qayta belgilanadigan bo‘lsa, u xuddi shu funksiya nomini o‘z ichiga oladi.

Haddan tashqari yuklangan funksiyalar, har xil "parametr (lar) ning soni yoki turi" jihatidan farq qiladi, bu ortiqcha yuklangan funksiyani boshqasidan farq qiladi. Shu tarzda, kompilyator qaysi ortiqcha yuklangan funksiya chaqirilishini

tan oladi. Ko‘pincha ortiqcha yuklangan funksiyalar "konstrukturlar" dir. "Nusxalashtiruvchi konstruktor" - bu "konstruktoring ortiqcha yuklanishi".

Quyida kompilyatsiya vaqtida funksiyaning haddan tashqari yuklanishi yoki polimorfizmni ko‘rsatadigan C++ dasturi keltirilgan:



```
1 // Ko'rsatish uchun C++ dasturi
2 // funksiyani haddan tashqari yuklash yoki
3 // Kompilyatsiya vaqtiga polimorfizmi
4 #include <bits/stdc++.h>
5
6 using namespace std;
7 class Polymf {
8 public:
9     // 1 int parametrli funksiya
10    void func(int x) {
11        cout << "x ning qiymati " << x << endl; }
12    // Xuddi shu nomdagi funksiya lekin
13    // 1 double parametr
14    void func(double x) {
15        cout << "x ning qiymati " << x << endl; }
16    // Xuddi shu nomdagi funksiya va
17    // 2 int parametr
18    void func(int x, int y) {
19        cout << "x va y ning qiymati " << x << ", " << y
20        << endl; }
21    };
22 int main(){
23     Polymf obj1;
24     // Chaqirilayotgan funksiya uzatilgan parametrlarga bog'liq
25     // func() int qiymati bilan chaqiriladi
26     obj1.func(7);
27
28     // func() double qiymati bilan chaqiriladi
29     obj1.func(9.132);
30
31     // func() 2 ta int qiymati bilan chaqiriladi
32     obj1.func(85, 64);
33     return 0;
34 }
```

**Izoh:** Yuqoridagi misolda func() nomli funksiya polimorfizm xossasi bo‘lgan uch xil holatda turlicha ishlaydi.

### Operatorning haddan tashqari yuklanishi

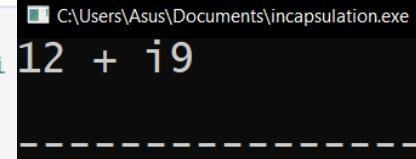
C++ operatorlarga ma'lumotlar turi uchun o‘ziga xos ma’no berish qobiliyatiga ega, bu qobiliyat operatorning ortiqcha yuklanishi deb nomlanadi.

Masalan, ikkita satrni birlashtirish uchun string klassi uchun qo'shish operatoridan (+) foydalanishimiz mumkin. Bizga ma'lumki, bu operatorning maqsadi ikkita operand qo'shishdir. Shunday qilib, butun operandlar orasiga joylashtirilgan bitta "+" operatori ularni qo'shadi va satr operandlari orasiga joylashtirilgan bitta "+" operatori ularni birlashtiradi.

Quyida operatorning haddan tashqari yuklanishini ko'rsatish uchun C++ dasturi keltirilgan:

```

1 // Ko'rsatish uchun C++ dasturi
2 // Operatorning haddan tashqari yuklanishi yoki
3 // Kompilyatsiya vaqtiga polimorfizmi
4 #include <iostream>
5 using namespace std;
6 class Complex {
7 private:
8     int real, imag;
9 public:
10 Complex(int r = 0, int i = 0){
11     real = r;
12     imag = i; }
13 // Ikkita Complex obyekt o'rtaida "+" ishlatilsa,
14 // bu avtomatik ravishda chaqiriladi
15 Complex operator+(Complex const& obj){
16     Complex res;
17     res.real = real + obj.real;
18     res.imag = imag + obj.imag;
19     return res; }
20 void print() { cout << real << " + " << imag << endl; }
21
22 int main(){
23     Complex c1(10, 5), c2(2, 4);
24     // Misol uchun "operator+" chaqiruvi
25     Complex c3 = c1 + c2;
26     c3.print();}
```

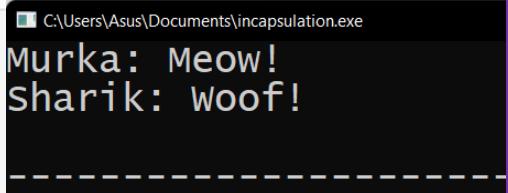


**Izoh:** Yuqoridagi misolda "+" operatori haddan tashqari yuklangan. Odatda bu operator ikkita sonni (butun yoki suzuvchi nuqta) qo'shish uchun ishlatiladi, lekin bu erda operator ikkita xayoliy yoki murakkab sonlarni qo'shish uchun mo'ljallangan.

Yuqorida bir nechta dastur kodlari misol sifatida keltirildi, polimorfizm yanada tushunarliroq bo‘lishi uchun nisbatan soddaroq dastur kodida tushuntiramiz. Odatta polimorfizm haqida hayvonlar misolida tushuntirishadi, biz ham shu an’anadan uzoqlashmagan holda misol keltiramiz. Aytaylik qandaydir bir o‘yinni ishlab chiqish uchun bizga mushuk va it sinflarini yaratish lozim. Sinflar hayvonlar ismlarini saqlash va ovoz funksiyalariga ega bo‘lishi kerak.

```

1 #include <string>
2 #include <iostream>
3 class Mushuk {
4 private:
5     std::string name;
6 public:
7     Mushuk(const std::string& n): name(n) {
8     }
9     const std::string& GetName() const {
10         return name;
11     }
12     std::string Voice() const {
13         return "Meow!";
14     }
15 class It {
16 private:
17     std::string name;
18 public:
19     It(const std::string& n): name(n) {
20     }
21     const std::string& GetName() const {
22         return name;
23     }
24     std::string Voice() const {
25         return "Woof!";
26     }
27 void Process(const Mushuk& creature) {
28     std::cout << creature.GetName() << ": " << creature.Voice() << "\n";
29 void Process(const It& creature) {
30     std::cout << creature.GetName() << ": " << creature.Voice() << "\n";
31 int main() {
32     Mushuk c("Murka");
33     It d("Sharik");
34     Process(c); // Murka: Meow!
35     Process(d); // Sharik: Woof!
36 }
```



Murka: Meow!  
Sharik: Woof!

Ushbu kodda darhol sezilarli takrorlanish mavjud. Keling, uni shablonlardan foydalanib olib tashlaymiz:

```

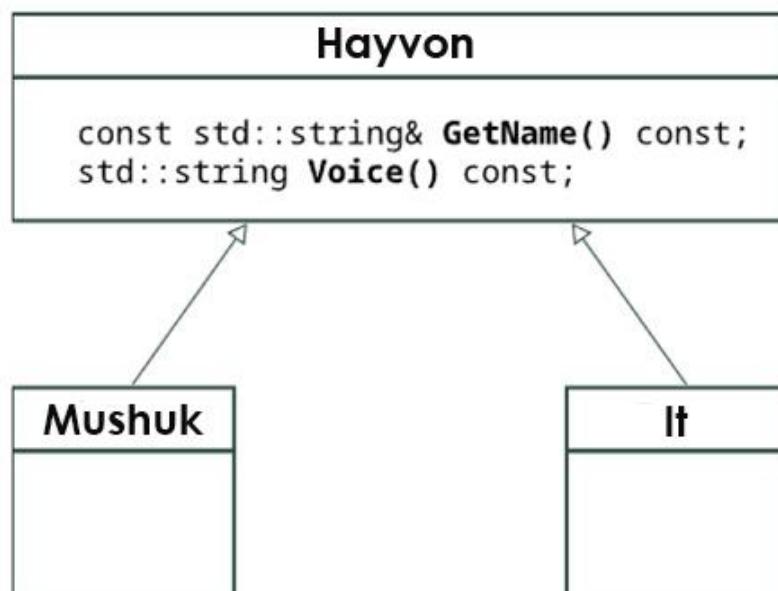
1 template <typename Creature>
2 void Process(const Creature& creature) {
3     std::cout << creature.GetName() << ": " << creature.Voice() << "\n";
4 }

```

Asosiy funksiya kodi o‘zgarmaydi. Qaysidir ma’noda, bu allaqachon polimorfizm: bizda bir xil interfeysga ega turli sinflar, shuningdek, ularni qayta ishslash uchun bir xil nomdagi ortiqcha yuklangan funksiyalar to‘plami mavjud. Bunday funksiyani chaqirish sintaktik jihatdan qaysi turdagiga hayvon ishlatilishiga bog‘liq emas, garchi funksiyalar aslida boshqacha bo‘lsa ham. Istalgan funksiyani tanlash kompilyatsiya paytida sodir bo‘ladi. Ushbu yondashuv parametrik polimorfizm deb ataladi.

### **Merosxo‘rlik orqali polimorfizm**

Sinflarimizni **Hayvon** deb nomlangan asosiy sinf bilan merosxo‘rlik ierarxiyasiga joylashtiramiz. Bu sizga kodning name maydoni va GetName funksiyasining takrorlanishidan xalos bo‘lish imkonini beradi, shuningdek, Voice funksiyasini o‘zingizga xos tarzda aniqlash imkonini beradi.



### **4.3 – rasm. Merosxo‘rlik orqali polimorfizm**

Har bir merosxo‘r sinf uchun o‘z konstruktoringizni yozishingizga to‘g‘ri keladi. U faqat bazaviy sinf konstruktornini chaqiradi.

```

1 #include <string>
2 class Animal {
3 private:
4     std::string name;
5 public:
6     Animal(const std::string& n): name(n) {
7     }
8     const std::string& GetName() const {
9         return name;
10    }
11    std::string Voice() const {
12        return "Umumiy hayvon ovozi";
13    };
14 class Mushuk: public Animal {
15 public:
16     Mushuk(const std::string& n): Animal(n) {
17     }
18     std::string Voice() const {
19         return "Meow!";
20     };
21 class It: public Animal {
22 public:
23     It(const std::string& n): Animal(n) {
24     }
25     std::string Voice() const {
26         return "Woof!";
27     };
28 #include <iostream>
29 template <typename Creature>
30 void Process(const Creature& creature) {
31     std::cout << creature.GetName() << ": " << creature.Voice() << "\n";
32 }
33 int main() {
34     Mushuk c("Murka");
35     It d("Sharik");
36     Process(c); // Murka: Meow!
37     Process(d); // Sharik: Woof!
38 }

```

C:\Users\Asus\Documents\incapsulation.exe

Murka: Meow!  
Sharik: Woof!

Biz bilamizki, merosxo‘r sinflar avtomatik tarzda bazaviy sinf turiga keltirilishi mumkin. Animal bazaviy sinfi argumentlarini unga jo‘natish uchun, Process funksiyasida shablonni olib tashlashga urinamiz

```

1 #include <iostream>
2
3 void Process(const Animal& creature) {
4     std::cout << creature.GetName() << ": " << creature.Voice() << "\n";
5 }
6
7 int main() {
8     Mushuk c("Murka");
9     It d("Sharik");
10    Process(c); // Tom: Umumiy hayvon ovozi
11    Process(d); // Sharik: Umumiy hayvon ovozi
12 }

```

Bizning dasturimiz tuziladi, lekin hayvonlar birdan ovozini yo‘qotadilar. Voice funksiyasining bekor qilingan versiyalari o‘rniga asosiy sinfdan Animal::Voice versiyasi chaqiriladi. Haqiqatdan ham Process funksiyasi Animal tipidagi parametr bilan ishlaydi va bu sinfda funksiyalarning o‘z versiyalariga ega avlodlari bo‘lishi mumkinligi haqida hech narsa bilmaydi. Siz bu muammoni Hayvonlarning virtual sinfidagi Ovoz funksiyasini e’lon qilish orqali hal qilishingiz mumkin.

### **Virtual funksiyalar**

Oldingi misolda Voice funksiyasi dasturni kompilyatsiya qilish bosqichida tanlangan. Creature argumentining formal turidan kelib chiqib kompilyator uni tanladi - funksiya Ichida u const Animal& edi. Bu C++ uchun standart yondashuv bo‘lib erta bog‘lanish deb ataladi. Bu shuni anglatadiki, dastur ishga tushirilgunga qadar ham xotirada tanlangan Animal::Voice funksiyasining manzili kodning shu joyiga oldindan bog‘langan.

Virtual sinf funksiyalari kech ulanishga imkon beradi. Bunday holda, mos funktsiyani tanlash dasturni bajarish jarayonida sodir bo‘ladi.

Funksiya asosiy sinfda virtual deb e'lon qilinishi kerak:

```
1 #include <string>
2
3+ class Animal {
4 public:
5     // ...
6
7+     virtual std::string Voice() const {
8         return "Umumiy hayvon ovozi";
9     }
10};
```

Merosxo'r sinflarda u qo'shimcha override belgisi bilan qayta belgilanishi kerak:

```
1+ class Mushuk: public Animal {
2 public:
3     // ...
4     std::string Voice() const override {
5         return "Meow!";
6     }
7+ class It: public Animal {
8 public:
9     std::string Voice() const override {
10        return "Woof!";
11    }
12};
```

## Polimorfizm va konteynerlar

Keling, hayvonot bog'ini tashkil qilaylik va hayvonlarimizni konteynerga yig'aylik. Standart konteynerlar element turi aniq ko'rsatilishi kerak. Keling, Animal asosiy sinfini ushbu tur sifatida belgilashga harakat qilaylik:

```
1 #include <vector>
2
3+ int main() {
4     std::vector<Animal> zoo;
5
6     zoo.push_back(Mushuk("Murka"));
7     zoo.push_back(It("Sharik"));
8
9     // Biz bu yerda sifl yozishimiz mumkin,
10    // lekin biz aniqlik uchun ikkita chaqiruv yozamiz
11    Process(zoo[0]); // Murka: umumiy hayvon ovozi
12    Process(zoo[1]); // Sharik: umumiy hayvon ovozi
13}
```

To'satdan virtual xatti-harakatlar buzildi: biz yana Animal::Voice asosiy funksiyasining chiqishini ko'ramiz. Bu vektor push\_back ga o'tkazilgan ob'ektlarning nusxalarini saqlaganligi sababli sodir bo'ldi va ushbu nusxalar Animal turni qabul qiladilar. Agar Process funksiyasi o'z parametrini doimiy havola orqali emas, balki qiymat bo'yicha qabul qilsa, xuddi shu narsa sodir bo'lardi.

Virtual funksiyalarning ishlashi uchun siz vektorning ichiga Animalga ko'rsatkichlar qo'yishingiz mumkin:

```
1 #include <vector>
2
3 int main() {
4     std::vector<Animal*> zoo;
5
6     // Hozircha stekda obyektlar yaratamiz
7     Mushuk c("Murka");
8     It d("Sharik");
9
10    // Ushbu ob'ektlarning manzillarini vektorga qo'yamiz
11    zoo.push_back(&c);
12    zoo.push_back(&d);
13
14    // Aloqani bekor qilish uchun yulduzcha kerakchka
15    Process(*zoo[0]); // Murka: Meow!
16    Process(*zoo[1]); // Sharik: Woof!
17 }
```

Bu misol ishlaydi, lekin bu juda qiziq emas. Biz stekdagi avtomatik ob'ektlarning manzillarini vektorga joylashtiramiz. Ushbu ob'ektlar blokdan chiqayotganda yo'q qilinadi. Shuning uchun, bunday vektorni funktsiyadan qaytarish yomon bo'ladi - undagi barcha ko'rsatkichlar darhol yaroqsiz bo'ladi.

## Nazorat savollari

1. Polimorfizm nima va uni oddiy so'zlar bilan qanday ta'riflash mumkin?
2. Polimorfizm OOP (Ob'ektga yo'naltirilgan dasturlash) da qanday rol o'ynaydi?
3. Polimorfizmning turlari qanday va ularning har birini qisqacha tushuntiring.
4. Funksiyani haddan tashqari yuklash nima va qanday amalga oshiriladi?
5. Operatorning haddan tashqari yuklanishi nima va qanday misol keltirilgan?
6. Merosxo'rlik orqali polimorfizmni tushuntirib bering va misol keltiring.
7. Virtual funksiyalar nima va ular qanday ishlaydi?
8. Polimorfizm va konteynerlar bilan ishlashda qanday muammolar yuzaga keladi va qanday hal qilinadi?

### 4.3-§ SHABLONLAR BILAN ISHLASH

**Reja:**

- 1. Shablon tushunchasi va ularning qo‘llanilishi;**
- 2. Funksiya shablonlarini, sınıf shablonlarini yaratish usullari va ularning qo‘llanilishi.**

Funksiyalar va sinflar samarali dasturlash uchun kuchli va moslashuvchan vositalar bo‘lsa-da, ular ba’zi hollarda C++ da ishlatiladigan barcha parametrlarning turlarini belgilash talabi bilan cheklangan. Masalan, ikkita sonning eng kattasini hisoblash uchun funksiya yozishimiz kerak deylik:

```
1 int max(int a, int b)
2 {
3     return (a > b) ? a : b;
4 }
```

Agar biz butun sonlar bilan ishlasak, hamma narsa yaxshi. Agar biz double turiga tegishli qiymatlar bilan ishlashimiz kerak bo‘lsa-chi? double tur bilan ishlash uchun max() funksiyasini ortiqcha yuklashga qaror qilasiz:

```
1 double max(double a, double b)
2 {
3     return (a > b) ? a : b;
4 }
```

Endi bizda bir xil funksiyaning ikkita versiyasi mavjud bo‘lib, int, double turlari va  $>$  operatorini ortiqcha yuklasak, ular hatto sinflar bilan ishlaydi! Biroq, C++ bizdan o‘zgaruvchilarning turlarini ko‘rsatishni talab qilganligi sababli, biz bir xil funksiyaning faqat parametrlar turi o‘zgargani uchungina bir nechta versiyasini yozishimizga to‘g‘ri keladi.

Bu esa o‘z navbatida, dasturchilar uchun ortiqcha muammolar keltirib chiqaradi, chunki bunday kodni saqlash kuch va vaqt jihatidan oson emas. Eng muhimi, bu samarali dasturlash tushunchalaridan birini buzadi - kodlarning takrorlanishini minimal darajada qisqartirish. **Har qanday** turdagи parametrlar bilan ishlaydigan max() funksiyasining bitta versiyasini yozish yaxshi emasmi?

Buning imkoni bor. Shablonlar dunyosiga xush kelibsiz!

Agar biz lug‘atdagi "shablon" so‘zining ta’rifiga qarasak, biz quyidagilarni ko‘ramiz: "Shablon - shunga o‘xhash mahsulotlar tayyorlanadigan namunadir." Masalan, shablon trafaret - buyum (masalan, plastinka), unda chizma/naqsh/ramz kesiladi. Agar biz trafaretini boshqa obyektga qo‘llasak va bo‘yoq purkasak, biz minimal kuch sarflab, tez bir xil dizaynga ega bo‘lamiz va eng muhimi, biz o‘nlab bunday dizaynlarni turli rangda yasay olamiz! Bunday holda, bizga faqat bitta trafaret kerak va biz dizaynning rangini oldindan aniqlashimiz shart emas (trafaretni ishlatishdan oldin).

C++ tilida **funksiya shablonlari** - boshqa shunga o‘xhash funksiyalarni yaratish uchun shablon bo‘lib xizmat qiladigan funksiyalardir. Asosiy g‘oya - ba’zi yoki barcha o‘zgaruvchilarning aniq turini (turlarini) ko‘rsatmasdan Funksiyalarni yaratish. Buning uchun har qanday ma’lumotlar turi o‘rniga qo‘llaniladigan shablon parametrining turini ko‘rsatish orqali Funksiyani aniqlaymiz. Shablon parametr turiga ega Funksiyani yaratganimizdan so‘ng, biz umuman olganda "Funksiya trafaretini" yaratgan hisoblanamiz.

Funksiya shablonini chaqirganda, kompilyator funksiya shablonlari sifatida "trafaret" dan foydalanadi va shablon parametri turini Funksiyaga uzatilgan o‘zgaruvchilarning haqiqiy turi bilan almashtiradi! Shunday qilib, biz faqat bitta shablon yordamida 50 ta funksiya ko‘rinishini yaratishimiz mumkin!

Sinflar, funksiyalar va (C++14 dan beri) o‘zgaruvchilar shablonlanishi mumkin. Shablon ba’zi bepul parametrlarga ega bo‘lgan kod bo‘lagi bo‘lib, barcha parametrlar aniqlanganda aniq sinf, funksiya yoki o‘zgaruvchiga aylanadi. Parametrlar turlar, qiymatlar yoki shablonlarning o‘zlari bo‘lishi mumkin. Mashhur shablon std::vector bo‘lib, u element turi ko‘rsatilganda aniq konteyner turiga aylanadi, masalan, std::vector<int>

### **Funksiya shablonlarini yaratish**

Hozircha siz C++ da funksiya shablonlari qanday yaratilishiga qiziqayotgandirsiz, bu unchalik qiyin ish emas. max() funksiyasining butun sonli versiyasini yana bir bor ko‘rib chiqamiz:

```
1 int max(int a, int b)
2 {
3     return (a > b) ? a : b;
4 }
```

Bu yerda ma'lumotlar turini uch marta ko'rsatamiz: a, b parametrlarida va Funksiyaning qaytish turida. Ushbu funksiya shablonini yaratish uchun biz int turini funksiya shablon parametr turi bilan almashtirishimiz kerak. Bu holatda faqat bitta ma'lumot turi (int) mavjud bo'lgani uchun biz shablon parametrining faqat bitta turini ko'rsatishimiz kerak.

Biz bu turni har qanday narsa deb atashimiz mumkin, agar u zaxiralangan/kalit so'z bo'lmasa. C++ tilida bosh T harfi bilan shablon parametrlari turlariga murojaat qilish odat tusiga kiradi ("Type" ning qisqartmasi).

Mana bizning o'zgartirilgan max() funksiyamiz:

```
1 T max(T a, T b)
2 {
3     return (a > b) ? a : b;
4 }
```

Lekin bu hammasi emas. **Dastur ishlamaydi**, chunki kompilyator T nima ekanligini bilmaydi!

Buni amalga oshirish uchun biz kompilyatorga ikkita narsani aytishimiz kerak:

- Funksiya shablonini aniqlash.
- T funksiya shablon parametrining turi ekanligini bildiradi.

Biz buni bitta kod qatorida **shablonni e'lon qilish** orqali amalga oshirishimiz mumkin (aniqrog'i, **shablon parametrlarini e'lon qilish**):

```
1 template <typename T> // funksiya shablonining parametrini e'lon qilish
2 T max(T a, T b)
3 {
4     return (a > b) ? a : b;
5 }
```

Endi dastur kodimiz ishlaydi.

Keling, shablon parametrlari deklaratsiyasini batafsil ko'rib chiqaylik:

Avval biz template kalit so‘zini yozamiz, bu esa kompilyatorga keyin shablon parametrlarini e’lon qilishimizni aytadi.

Funksiya shablonining parametrlari burchakli qavslarda ( $\langle \rangle$ ) ko‘rsatiladi.

Shablon parametr turlarini yaratish uchun typename va class kalit so‘zlaridan foydalaniladi. Funksiya shablonlarini ishlashning asosiy holatlarida typename va class o‘rtasida farq yo‘q, shuning uchun istalgan ikkalasidan biridan foydalanishingiz mumkin. Agar siz class kalit so‘zidan foydalansangiz, u holda parametrlarning haqiqiy turi sinf bo‘lishi shart emas (bu asosiy ma’lumotlar turi o‘zgaruvchisi, ko‘rsatgich yoki boshqa narsa bo‘lishi mumkin).

Keyin shablon parametrining turini nomlaymiz (odatda T).

Agar bir necha turdagи shablon parametrlari kerak bo‘lsa, ular vergul bilan ajratiladi:

```
1 template <typename T1, typename T2>
2 // Funksiya shabloni
```

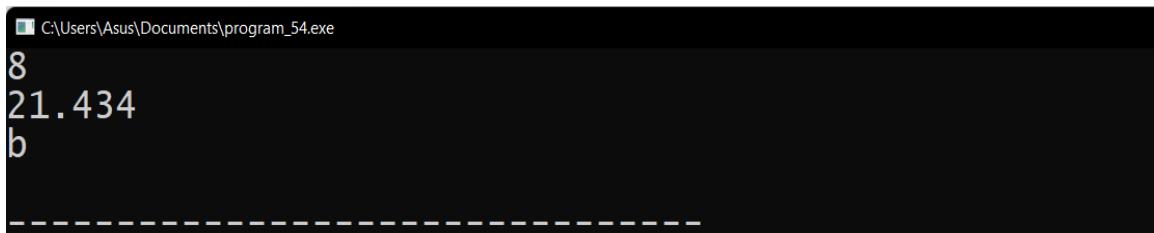
Agar bir nechta parametrlar mavjud bo‘lsa, ular odatda T1, T2 yoki boshqa harflar bilan belgilanadi: T, S.

### Funksiya shablonlaridan foydalanish

Funksiya shablonlaridan foydalanish oddiy funksiyalardan foydalanishga o‘xshaydi:

```
1 #include <iostream>
2
3 template <typename T>
4 const T& max(const T& a, const T& b)
5 {
6     return (a > b) ? a : b;
7 }
8
9 int main()
10 {
11     int i = max(4, 8);
12     std::cout << i << '\n';
13
14     double d = max(7.56, 21.434);
15     std::cout << d << '\n';
16
17     char ch = max('b', '9');
18     std::cout << ch << '\n';
19
20     return 0;
21 }
```

Natija:



```
C:\Users\Asus\Documents\program_54.exe
8
21.434
b
```

E'tibor bering, `max()` ga uchta chaqiruv ham har xil turdag'i parametrleriga ega! Biz `max()` funksiyasini uch xil turdag'i parametrler bilan chaqirganimiz sababli, kompilyator `max()` funksiyasining uch xil versiyasini yaratish uchun funksiya shablonidan foydalanadi:

- int tipidagi parametrleriga ega versiya (`max<int>`).
- double tipidagi parametrleriga ega versiya (`max<double>`).
- char tipidagi parametrleriga ega versiya (`max<char>`).

O'tkazilayotgan qiymatlar turini (`max<int>`ning `<int>` qismi) aniq ko'rsatishning hojati yo'q, kompilyator buni o'zi aniqlaydi.

Funksiya shablonlari ko'p vaqtini tejaydi, chunki biz shablonni faqat bir marta yozamiz va uni turli xil ma'lumotlar turlari bilan ishlatishimiz mumkin. Funksiya shablonlarini yozishga odatlanganingizdan so'ng, oddiy funksiyani (oddiy funksiyaning bitta versiyasi) yozish uchun ko'proq vaqt talab qilinmasligini aniqlaysiz. Funksiya shablonlari kodni keyinchalik saqlashni ancha osonlashtiradi va ular xavfsizroq, chunki kodni nusxalash va faqat yangi ma'lumotlar turini qo'llab-quvvatlash kerak bo'lganda ma'lumotlar turlarini o'zgartirish orqali funksiyani haddan tashqari yuklashingiz shart emas.

Funksiya shablonlari bir nechta kamchiliklarga ega va agar biz ular haqida gapirmasak, e'tibordan chetda qoldirgan hisoblanamiz:

- Birinchidan, ba'zi eski kompilyatorlar funksiya shablonlarini qo'llab-quvvatlamasligi yoki ularni qo'llab-quvvatlashi mumkin, lekin cheklovlar bilan. Biroq, bu endi avvalgidek muammo emas.

- Ikkinchidan, funksiya shablonlari ko‘pincha aqlbovar qilmaydigan xatolik xabarlarini chiqaradi, ularni tushunish oddiy funksiyalar xatoliklaridan ko‘ra qiyinroq.
- Uchinchidan, funksiya shablonlari kompilyatsiya vaqtini va kod hajmini oshirishi mumkin, chunki bitta shablonni bir nechta fayllarda “amalga oshirish” va qayta kompilyatsiya qilinishi mumkin.

Funksiya shablonlarining kuchi va moslashuvchanligi bilan solishtirganda, bu kamchiliklar unchalik qo‘rinchli emas.

### **Nazorat Savollari**

1. Shablonlarning asosiy maqsadi nima?
2. Funksiya shablonlari qanday aniqlanadi va foydalaniladi?
3. Qanday qilib kompilyator shablon parametrlarini haqiqiy turlar bilan almashtiradi?
4. Shablon parametrlarining turlarini ko‘rsatish uchun qanday kalit so‘zlardan foydalaniladi va ularning farqi nimada?
5. Shablon funksiyalaridan foydalanishda qaysi holatlarda turli parametr turlari bo‘lishi mumkin?
6. Funksiya shablonlarining asosiy afzalliklari va kamchiliklari nimada?
7. C++ tilida qanday hollarda shablonlar ishlatilishi mumkin?

## 4.4-§ KONTEYNERLAR (KOLLEKSIYALAR)

**Reja:**

- 1. STL kutubxonaları**
- 2. Konteyner sinfları**

Standart shablonlar kutubxonasi (STL) C++ standart kutubxonasining bir qismi bo‘lib, konteyner sinfi shablonlari (std::vektor va std::massiv kabi), algoritmlar va iteratorlarni o‘z ichiga oladi. Bu dastlab uchinchi tomon ishlanmasi edi, lekin keyinchalik C++ standart kutubxonasiga kiritildi. Agar sizga umumiy sinf yoki algoritm kerak bo‘lsa, standart shablonlar kutubxonasida u allaqachon mavjud. Yaxshi tomoni shundaki, siz ushbu sinflarni o‘zingiz boshidan yozmasdan (yoki ular qanday amalga oshirilayotganini tushunmasdan) foydalanishingiz mumkin. Bundan tashqari, siz ushbu sinflarning juda samarali (va ko‘p marta sinovdan o‘tgan) versiyalarini olasiz. Salbiy tomoni shundaki, standart shablonlar kutubxonasining funksionalligi bilan hamma narsa unchalik oddiy/ravshan emas va bu yangi o‘rganuvchilar uchun biroz chalkash bo‘lishi mumkin, chunki aksariyat sinflar aslida sinf shablonlaridir.

Albatta, STL kutubxonasining eng ko‘p qo‘llaniladigan funksionalligi konteyner sinflari (yoki ularni "konteynerlar" deb ham atashadi). STL turli vaziyatlarda ishlatilishi mumkin bo‘lgan turli xil konteyner sinflarini o‘z ichiga oladi. Umuman olganda, STL konteynerlari uchta asosiy toifaga bo‘linadi:

- ketma-ket;
- assotsiativ;
- adapterlar.

### **Ketma-ket konteynerlar**

Ketma-ket konteynerlar – bu elementlari ketma-ketlikda joylashgan konteyner sinflari. Ularning o‘ziga xos xususiyati shundaki, siz elementingizni konteynerning istalgan joyiga qo‘shishingiz mumkin. Ketma-ket konteynerning

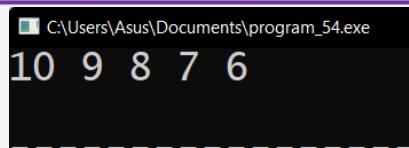
eng keng tarqalgan misoli massivdir: massivga 4 ta element qo'shsangiz, u elementlar (massivda) aynan siz qo'shgan tartibda paydo bo'ladi.

C++11 dan boshlab, STL 6 ta ketma-ketlik konteynerlarini o'z ichiga oladi:

- std::vector;
- std::deque;
- std::array;
- std::list;
- std::forward\_list;
- std::basic\_string.

**Vektor sinfi** (yoki oddiygina "vektor") dinamik massiv bo'lib, uning barcha elementlarini o'z ichiga olishi uchun kerak bo'lganda o'sishi mumkin. Vektor klassi indekslash operatori [] orqali o'z elementlariga tasodifiy kirishni ta'minlaydi, shuningdek elementlarni qo'shish va olib tashlashni qo'llab-quvvatlaydi.

Quyidagi dasturda biz vektorga 5 ta butun son qo'shamiz va keyinchalik chiqarish uchun ularga kirish uchun haddan tashqari yuklangan indekslash operatoridan [] foydalanamiz:



```
1 #include <iostream>
2 #include <vector>
3
4 int main()
5 {
6     std::vector<int> vect;
7     for (int count=0; count < 5; ++count)
8         vect.push_back(10 - count); // raqamlarni massiv oxiriga qo'shamiz
9
10    for (int index=0; index < vect.size(); ++index)
11        std::cout << vect[index] << ' ';
12    std::cout << '\n';
13 }
```

**Deque sinfi** (yoki oddiygina "deq") – har ikki uchida ham o'sishi mumkin bo'lgan dinamik massiv sifatida amalga oshiriladigan ikki tomonlama navbatdir.

Masalan:

```

1 #include <iostream>
2 #include <deque>
3 int main()
4 {
5     std::deque<int> deq;
6     for (int count=0; count < 4; ++count)
7     {
8         deq.push_back(count); // sonlarni massiv oxiriga qo'shamiz
9         deq.push_front(10 - count); // sonlarni massiv boshiga qo'shamiz
10    }
11
12    for (int index=0; index < deq.size(); ++index)
13        std::cout << deq[index] << ' ';
14
15    std::cout << '\n';
16 }

```

C:\Users\Asus\Documents\program\_54.exe  
7 8 9 10 0 1 2 3  
-----

**List** (yoki oddiygina “ro‘yxat”) – ikki tomonlama bog‘langan ro‘yxat bo‘lib, uning har bir elementi 2 ta ko‘rsatkichdan iborat: biri ro‘yxatning keyingi elementiga, ikkinchisi esa ro‘yxatning oldingi elementiga ishora qiladi. List faqat ro‘yxatning boshi va oxiriga kirishni ta’minlaydi - tasodifiy kirishga ruxsat berilmaydi. Agar siz o‘rtada biron bir qiymatni topmoqchi bo‘lsangiz, unda siz bir tomondan boshlashingiz va izlayotgan narsangizni topmaguningizcha ro‘yxatning har bir elementini ko‘rib chiqishingiz kerak. Ikki bog‘amali ro‘yxatning afzalligi shundaki, elementlarni qo‘sishni juda tez amalgalash oshiriladi, faqat elementni qayerga qo‘sishni bilsangizgina. Odatda, iteratorlar ikki bog‘lamali ro‘yxat elementlarini o‘tib chiqish uchun foydalaniladi.

Garchi **string** (va **wstring**) klassi odatda ketma-ket konteyner sifatida ko‘rilmasa ham, u mohiyatan shunday, chunki uni **char** (yoki **wchar**) elementlarning vektori sifatida ko‘rish mumkin.

### Assotsiativ konteynerlar

Assotsiativ konteynerlar – bu o‘zining barcha elementlarini avtomatik tarzda saralaydigan konteyner sinflaridir. Odatda, assotsiativ konteynerlar elementlarni < taqqoslash operatori yordamida tartiblaydi.

- **set** – faqat noyob elementlarni saqlaydigan konteyner bo‘lib, hech qanday takrorlashga yo‘l qo‘yilmaydi. Elementlar qiymatlari bo‘yicha saralanadi.
- **multiset** – bu to‘plam, lekin unda takroriy elementlarga ruxsat beriladi.

- **map** (yoki "assotsiativ massiv") – har bir element kalit-qiyamat juftligi bo‘lgan to‘plamdir. "Kalit" ma’lumotlarni saralash va indekslash uchun ishlatiladi va noyob bo‘lishi kerak, "qiymat" esa haqiqiy ma’lumotdir.
- **multimap** (yoki "lug‘at") – bu ikki nusxadagi kalitlarga ruxsat beruvchi **map**. Barcha kalitlar o‘sish tartibida tartiblangan va siz qiymatni kalit bo‘yicha ko‘rishingiz mumkin.

### **Tartibsiz assotsiativ konteynerlar**

**Tartiblanmagan assotsiativ konteynerlar** tezda qidirilishi mumkin bo‘lgan saralanmagan (xeshlangan) ma’lumotlar tuzilmalarini amalga oshiradi ( $O(1)$  amortizatsiya,  $O(n)$  eng yomon murakkablik).

- **unordered\_set**: Noyob kalitlar to‘plami, kalit tomonidan xeshlangan (sinf shabloni)
- **unordered\_map**: kalit-qiyamat juftliklari to‘plami, kalitlar tomonidan xeshlangan, kalitlar noyobdir. (sinf shabloni)
- **unordered\_multiset**: Kalitlar to‘plami, kalitlar tomonidan xeshlangan (sinf shabloni)
- **unordered\_multimap**: Kalit-qiyamat juftliklari to‘plami, kalitlar tomonidan xeshlangan (sinf shabloni).

### **Adapterlar**

**Adapterlar** - bu aniq vazifalarni bajarish uchun moslashtirilgan, oldindan belgilangan maxsus konteyner sinflari. E’tiborlisi shundaki, siz adapter qaysi ketma-ket konteynerdan foydalanishi kerakligini tanlashingiz mumkin.

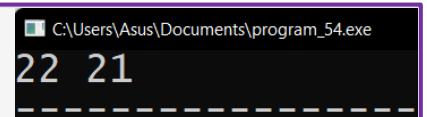
- **stack** (stek) – bu konteyner sinfi bo‘lib, uning elementlari LIFO (Last In, First Out) tamoyiliga muvofiq ishlaydi, ya’ni elementlar idishning oxiriga qo‘shiladi (kiritiladi) va bir joydan (idishning oxiridan) chiqariladi. Odatda steklar ketma-ket konteyner sifatida deque dan foydalanadi, lekin siz **vector** yoki **list** dan ham foydalanishingiz mumkin.

- **List** (navbat) – bu konteyner sinfi, uning elementlari FIFO tamoyiliga muvofiq ishlaydi (inglizcha "Birinchi kirdi, birinchi chiqadi" = "birinchi kirdi, birinchi chiqadi"), ya’ni elementlar idishning oxiriga qo‘shiladi, lekin idishning boshidan olib tashlanadi. Odatiy bo‘lib, navbat ketma-ket konteyner sifatida deque dan foydalanadi, lekin **list** ham ishlatilishi mumkin.
- **priority\_queue** – bu (<taqqoslash operatori yordamida) barcha elementlar tartiblangan navbat turi hisoblanadi. Navbatga element qo‘shilganda, u avtomatik tarzda tartiblanadi. Eng yuqori ustuvorga ega element **priority\_queue** ning boshida bo‘ladi. Hamda elementlarni o‘chirish ham navbatning boshidan amalga oshiriladi.

Quyida stek konteyner adapteriga misol keltirilgan:

```

1 #include <iostream>
2 #include <stack>
3 using namespace std;
4 int main() {
5     stack<int> stack;
6     stack.push(21);
7     stack.push(22);
8     stack.push(24);
9     stack.push(25);
10    int num=0;
11    stack.push(num);
12    stack.pop();
13    stack.pop();
14    stack.pop();
15
16    while (!stack.empty()) {
17        cout << stack.top() << " ";
18        stack.pop();
19    }
20 }
```



C++ standart shablonlar kutubxonasining (STL) kamchiliklari:

1. **O‘rganish qiyinligi:** STLni o‘rganish qiyin bo‘lishi mumkin, ayniqsa yangi boshlanuvchilar uchun, uning murakkab sintaksi si va iteratorlar va funksiya ob’ektlari kabi ilg‘or xususiyatlardan foydalanish tufayli.

2. **Nazoratning yetishmasligi:** STL dan foydalanganda siz kutubxona tomonidan taqdim etilgan dasturga tayanishingiz kerak, bu sizning kodingizning ayrim jihatlari ustidan nazoratingizni cheklashi mumkin.
3. **Unumdorligi:** Ba’zi hollarda, STL-dan foydalanish, maxsus kod bilan solishtirganda, ayniqsa kichik hajmdagi ma’lumotlar bilan ishlashda sekinroq bajarilish vaqtlariga olib kelishi mumkin.

### **Nazorat savollari**

1. STL (Standart Shablonlar Kutubxonasi) nima va u nima uchun ishlatiladi?
2. STL konteyner sinflari qanday toifalarga bo‘linadi?
3. Ketma-ket konteynerlar nima va qanday misollar keltirilgan?
4. Assotsiativ konteynerlar qanday ishlaydi va qanday turlari mavjud?
5. Tartiblanmagan assotsiativ konteynerlar nima, ularga misollar keltiring?
6. Adapterlar nima va misollar keltiring?
7. STLning kamchiliklari nimada?

## V BOB. C++ DASTURLASH TILIDA GUI MUHITI VA FOYDALANUVCHI INTERFEYSINI YARATISH

### 5.1.1-§ GUI MUHITIDA DASTURLASH

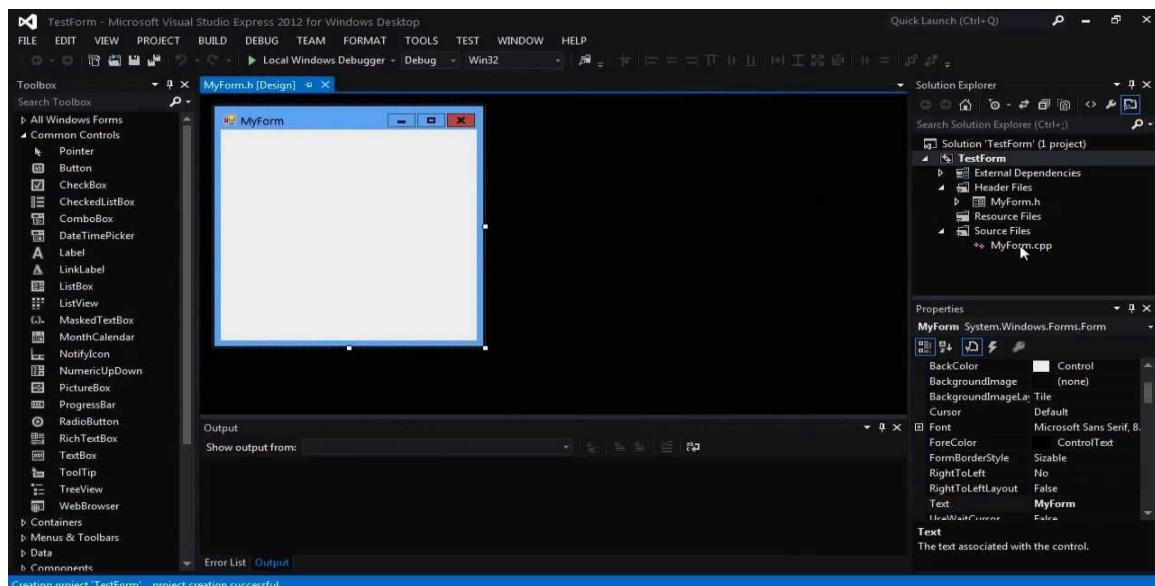
**Reja:**

- 1. GUI muhitida dasturlash**
- 2. GUI muhitida menyular va uskunalar paneli**

**Visual C++** - bu dasturni ishlab chiqish muhiti va komponentlar kutubxonasidan foydalangan holda C++ tilida ilovalar yaratish imkonini beruvchi dasturlarni tez ishlab chiqish vositasi. Ushbu o‘quv qo‘llanmada Visual C++ dasturlash muhiti va foydalanuvchi interfeysini loyihalashda qo‘llaniladigan asosiy texnikalarni qamrab oladi.

#### **Visual C++ ishlab chiqish muhiti**

Visual C++ SDI ilovasini taqdim etadi, uning asosiy oynasida sozlanadigan uskunalar paneli (yuqorida) va komponentlar palitrasи (o‘ngda) mavjud. Bundan tashqari, standart bo‘yicha, Visual C++ dasturini ishga tushirganingizda, Object Inspector oynasi (pastki chap) va Yangi ilova shakli (yuqori chap) paydo bo‘ladi. Ilova shakli oynasi ostida kod muharriri oynasi joylashgan.

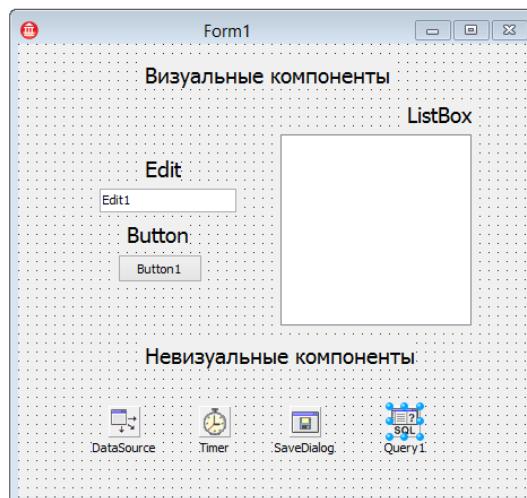


5.1 - rasm. Visual C++ ishlab chiqish muhiti

Shakllar Visual C++ dasturlarining asosiy qismidir. Ilovaning foydalanuvchi interfeysi yaratish shakl oynasiga komponentlar deb ataladigan Visual C++ obyekt elementlarini qo'shishni o'z ichiga oladi. Visual C++ komponentlari komponentlar palitrasida joylashgan bo'lib, ko'p sahifali bloknot sifatida yaratilgan. Visual C++ ning muhim xususiyati shundaki, u o'z komponentlarini yaratish va Komponentlar palitrasini moslashtirish, shuningdek, turli loyihalar uchun Komponentlar palitrasining turli versiyalarini yaratish imkonini beradi.

### **Visual C++ komponentlari**

Komponentlar ko'rindigan (vizual) va ko'rinnmas (vizual bo'lmagan) ga bo'linadi. Vizual komponentlar loyihalash vaqtida bo'lgani kabi ish vaqtida ham paydo bo'ladi. Masalan, tugmalar va tahrirlanadigan maydonlar. Vizual bo'lmagan komponentlar dizayn paytida shakldagi piktogramma sifatida paydo bo'ladi. Ular hech qachon ish vaqtida ko'rinnmaydi, lekin ma'lum funksiyalarga ega (masalan, ular ma'lumotlarga kirishni ta'minlaydi, standart dialoglarni chaqiradi va hokazo).



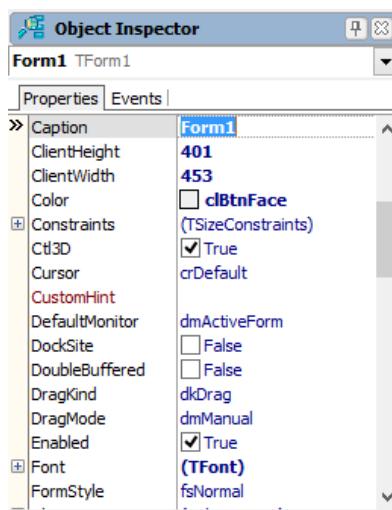
5.2 - rasm. Ko'rindigan (vizual) va ko'rinnmaydigan (vizual bo'lmagan) komponentlar

Formaga komponent qo'shish uchun sichqoncha yordamida palitradan kerakli komponentni tanlab, loyihalashtirilgan formaning kerakli joyida

sichqonchaning chap tugmasini bosish mumkin. Komponent shaklda paydo bo‘ladi, keyin siz uni ko‘chirishingiz, hajmini va boshqa xususiyatlarini o‘zgartirishingiz mumkin.

Har bir Visual C++ komponenti uch xil xususiyatga ega: xususiyatlar, hodisalar va usullar.

Agar siz palitradan komponentni tanlab, uni formaga qo‘shsangiz, Obyekt inspektori avtomatik ravishda ushbu komponent bilan ishlatalishi mumkin bo‘lgan xususiyatlar va hodisalarini ko‘rsatadi. Obyekt inspektorining yuqori qismida shaklda mavjud bo‘lganlardan kerakli obyektni tanlash imkonini beruvchi ochiladigan ro‘yxat mavjud.



5.3 - rasm. Obyektlar inspektori

### Komponent xususiyatlari

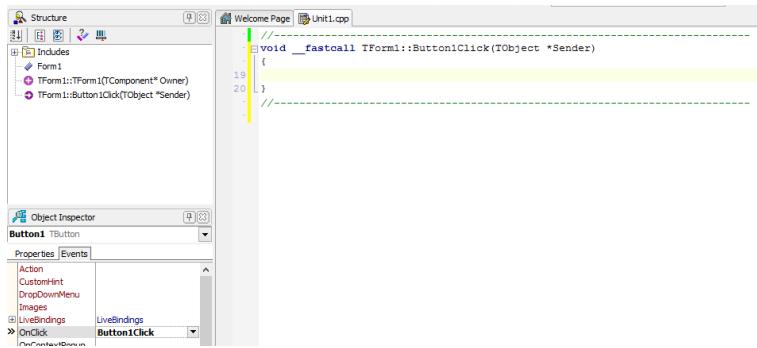
Xususiyatlari - komponentning tashqi ko‘rinishi va xatti-harakatini belgilaydigan atributlari. Xususiyatlari ustunidagi ko‘plab komponentlar xossalari standart qiymatga ega (masalan, tugma balandligi). Komponentning xususiyatlari xususiyatlar sahifasida ko‘rsatiladi. Obyekt inspektori komponentlarning e’lon qilingan xususiyatlarini ko‘rsatadi. Chop etilgan xususiyatlarga qo‘shimcha ravishda, komponentlar faqat dastur ishlayotgan paytda mavjud bo‘lgan ommaviy, nashr etilgan xususiyatlarga ega bo‘lishi mumkin va ko‘pincha mavjud. Obyekt inspektori dizayn vaqtida xususiyatlarni o‘rnatish uchun ishlataladi.

Xususiyatlar ro‘yxati Obyekt inspektorining xususiyatlar sahifasida joylashgan. Siz dizayn vaqtida xususiyatlarni belgilashingiz yoki ish vaqtida komponent xususiyatlarini o‘zgartirish uchun kod yozishingiz mumkin va hokazo.).

## **Voqealar**

Obyekt inspektorining Voqealar sahifasi komponent tomonidan tan olingan hodisalar ro‘yxatini ko‘rsatadi (grafik foydalanuvchi interfeysi bo‘lgan operatsion tizimlar uchun dasturlash, xususan Windows uchun, ilovaning ma’lum hodisalarga munosabatini tavsiflashni o‘z ichiga oladi va operatsion tizimning o‘zi doimiy ravishda so‘rov o‘tkazadi. har qanday voqealr sodir bo‘lishini aniqlash uchun kompyuter). Har bir komponentning o‘ziga xos hodisa ishlovchilar to‘plami mavjud. Visual C++ da siz hodisalarni ishlov beruvchilar deb ataladigan funksiyalarni yozasiz va hodisalarni ushbu funksiyalarga bog‘laysiz. Voqealr uchun ishlov beruvchini yaratish orqali siz ushbu hodisa ro‘y bersa, dasturga yozma funktsiyani bajarishni buyurasiz.

Hodisa ishlov beruvchisini qo‘shish uchun sichqoncha yordamida formada hodisa ishlovchisi kerak bo‘lgan komponentni tanlashingiz kerak, so‘ngra Obyekt inspektorining Voqealar sahifasini oching va hodisa yonidagi qiymat ustunidagi sichqonchaning chap tugmchasini ikki marta bosing. Visual C++ prototipini voqealr ishlov beruvchisini yaratishga majburlash va uni kod muharririda ko‘rsatish. Bunday holda, bo‘sh funksiya matni avtomatik ravishda hosil bo‘ladi va tahrirlovchi kodni kiritish kerak bo‘lgan joyda ochiladi. Kursor operator qavslar ichida joylashgan { ... }. Keyinchalik, voqealr sodir bo‘lganda bajarilishi kerak bo‘lgan kodni kiritishingiz kerak. Hodisa ishlov beruvchisi qavs ichidagi funksiya nomidan keyin ko‘rsatilgan parametrlarga ega bo‘lishi mumkin.



5.4 - rasm. Voqealarni qayta ishslash prototipi.

## Usullari

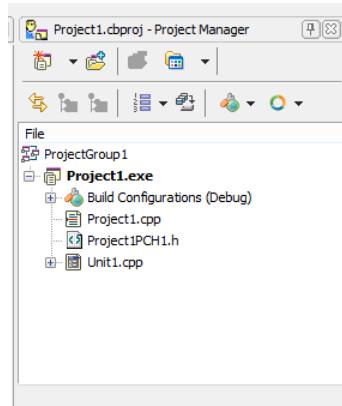
Usul - komponent bilan bog‘langan va obyektning bir qismi sifatida e’lon qilingan funksiya. Hodisa ishlov beruvchilarini yaratishda siz quyidagi belgilar yordamida usullarni chaqirishingiz mumkin: ->, masalan:

Edit1->Show();

Esda tutingki, forma yaratishda \*.h kengaytmasi bilan bog‘langan modul va sarlavha fayli albatta yaratiladi, yangi modul yaratishda esa u shakl bilan bog‘lanishi shart emas (masalan, agar u hisoblash protseduralarini o‘z ichiga olgan bo‘lsa). . Shakl va modul nomlari o‘zgartirilishi mumkin va buni yaratilgandan so‘ng darhol boshqa shakl va modullarda ularga ko‘plab havolalar paydo bo‘lishidan oldin qilish tavsiya etiladi.

## Loyihalar menejeri

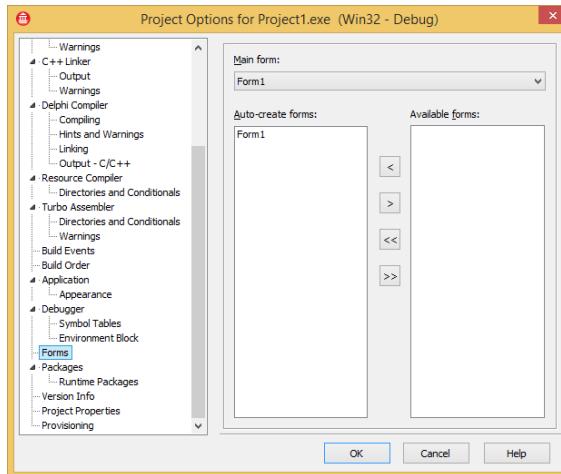
Ilovani tashkil etuvchi fayllar - shakllar va modullar loyihada to‘planadi. Loyiha menejeri fayllar va dastur modullari ro‘yxatini ko‘rsatadi va ular orasida harakat qilish imkonini beradi (2.5-rasm). Ko‘rish/Loyiha menejeri menu bandini tanlash orqali loyiha menejerini ochishingiz mumkin. Odatiy bo‘lib, yangi yaratilgan loyiha Project1.cpp deb nomlanadi.



### 5.5 - rasm. Loyihalar bo‘yicha menejer

Odatiy bo‘lib, loyiha dastlab bitta shakl va bitta modulning manba kodi uchun fayllarni o‘z ichiga oladi. Biroq, aksariyat loyihalar bir nechta shakl va modullarni o‘z ichiga oladi. Loyihaga modul yoki shakl qo‘shish uchun sichqonchaning o‘ng tugmchasini bosish va kontekst menyusidan “Yangi forma”ni tanlash kerak. Loyiha menejerining kontekst menyusidagi Qo‘shish tugmasidan foydalanib va qo‘shmoqchi bo‘lgan modul yoki shaklini tanlash orqali loyihaga mavjud shakl va modullarni ham qo‘shishingiz mumkin. Shakllar va modullar loyihani ishlab chiqish paytida istalgan vaqtda o‘chirilishi mumkin. Biroq, forma har doim modul bilan bog‘langanligi sababli, modul forma bilan bog‘lanmagan bo‘lsa, birini o‘chirmasdan ikkinchisini o‘chirish mumkin emas. Loyiha menejerining O‘chirish tugmasi yordamida loyihadan modulni olib tashlashingiz mumkin.

Agar siz loyiha menejerida Variantlar tugmchasini tanlasangiz, loyiha variantlari dialog oynasi ochiladi, unda siz ilovaning asosiy shaklini tanlashingiz, qaysi shakllar dinamik ravishda yaratilishini, modulni kompilyatsiya qilish va joylashtirish parametrlari qanday ekanligini aniqlashingiz mumkin (2-rasm). 2.6).



5.6 - rasm. Loyerha parametrlarini sozlash

Visual C++ dasturlash muhitining muhim elementi sichqonchaning o‘ng tugmachasini bosganingizda paydo bo‘ladigan va eng ko‘p ishlatiladigan buyruqlarga tezkor kirishni taklif qiluvchi kontekst menyusidir.

Albatta, Visual C++ har qanday interfeys elementi uchun mayjud bo‘lgan va Visual C++ haqida to‘liq ma’lumot manbasi bo‘lgan o‘rnatilgan kontekstga sezgir yordam tizimiga ega.

### **Visual C++ da ilovalar yaratish**

Visual C++ dasturini ishlab chiqishda birinchi qadam loyerha yaratishdir. Loyerha fayllari avtomatik ravishda yaratilgan manba kodini o‘z ichiga oladi, u kompilyatsiya qilingan va ishga tayyor bo‘lganda dasturning bir qismiga aylanadi. Yangi loyerha yaratish uchun menu bandini tanlash kerak File/New VCL Forms Application - C++ Builder.

Visual C++ standart nomli Project1.cpp va Project1.cbproj standart nomli makefile bilan loyerha faylini yaratadi. Loyihaga o‘zgartirishlar kiritganingizda, masalan, yangi shakl qo‘shsangiz, Visual C++ loyerha faylini yangilaydi.

```
1 //\n\n2 #include <vcl.h>\n3 #pragma hdrstop\n4\n5 #include "Unit1.h"\n6 //\n7 #pragma package(smart_init)\n8 #pragma resource "*.dfm"\n9\n10 TForm1 *Form1;\n11 //\n12\n13 __fastcall TForm1::TForm1(TComponent* Owner)\n14 : TForm(Owner)\n15 {\n16\n17 }
```

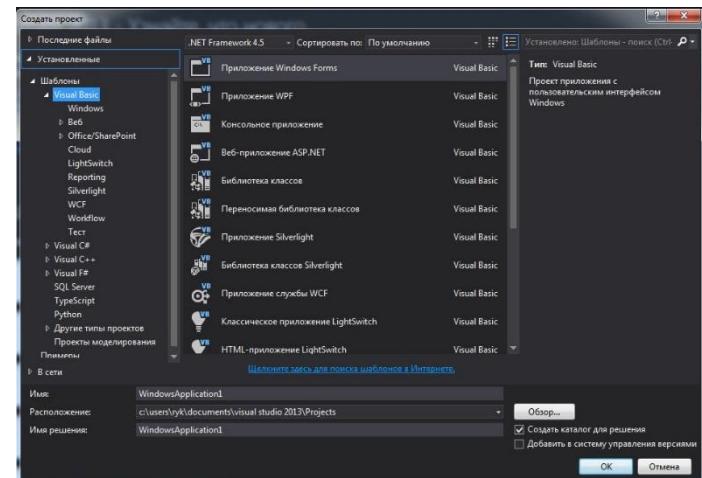
## 5.7 – rasm. Loyiha fayli

Loyiha yoki dastur odatda bir nechta shakllarni oladi. Loyihaga forma qo'shish quyidagi qo'shimcha fayllarni yaratadi:

- .DFM kengaytmali forma fayli, formani yaratish uchun oyna resurslari haqidagi ma'lumotlarni o'z ichiga oladi;
  - C++ kodini o'z ichiga olgan .CPP kengaytmali modul fayli;
  - Shakl sinfining tavsifini o'z ichiga olgan .H kengaytmali sarlavha fayli.

Yangi shakl qo'shsangiz, loyiha fayli avtomatik ravishda yangilanadi.

Loyihaga bir yoki bir nechta shakl qo'shish uchun menuy bandini tanlang File/New VCL Form - C++ Builder. Bo'sh shakl paydo bo'ladi va loyihaga qo'shiladi. Menyuning File/New bandidan foydalanishingiz mumkin, Boshqa sahifani tanlashingiz va obyektlar omboridan mos shablonni tanlashingiz mumkin (2.8-rasm).



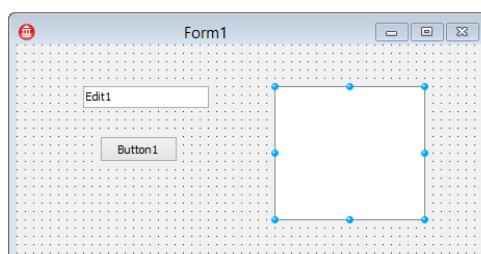
5.8 – rasm. Obyektlar shablonlari bo‘limi

Loyihani kompilyatsiya qilish va joriy loyiha uchun bajariladigan faylni yaratish uchun Run menyusidan Run menuy bandini tanlash kerak. Loyerha qurilishi bosqichma-bosqich (faqat o‘zgartirilgan modullar qayta kompilyatsiya qilinadi).

Agar dasturni ishga tushirish vaqtida ish vaqtida xatolik yuzaga kelsa, Visual C++ dasturni to‘xtatib turadi va xatoga sabab bo‘lgan bayonotda joylashgan kursov bilan kod muharririni ko‘rsatadi. Kerakli tuzatishni amalgalashdan oldin, kontekst menyusidan yoki "Ishga tushirish" menyusidan "Ishga tushirish" bandini tanlab, dasturni qayta ishga tushirishingiz kerak, dasturni yoping va shundan keyingina loyihaga o‘zgartirishlar kriting. Bu Windows resurslarini isrof qilish ehtimolini kamaytiradi.

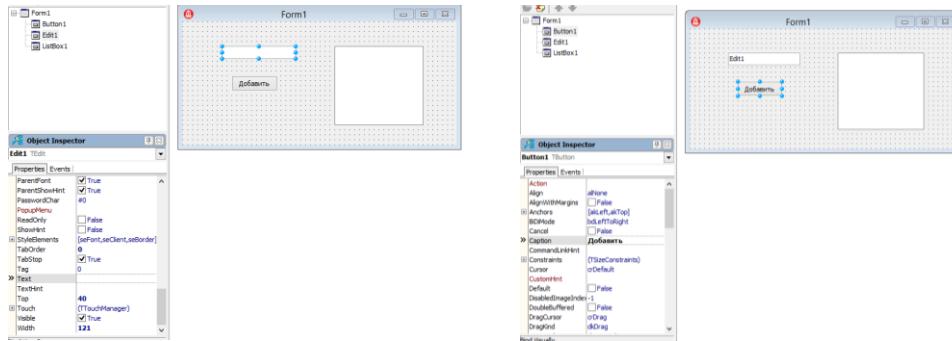
### **Oddiy dastur yaratishga misol**

Endi tahrir qilinadigan maydonga matn kiritish va tugmani bosganingizda ushbu matnni ro‘yxatga qo‘shish imkonini beruvchi oddiy dastur yaratishga harakat qilaylik. Loyerha yaratish va uning asosiy shaklini lab1.cpp nomi ostida, loyihaning o‘zini esa lab.cbproj nomi ostida saqlash uchun File/New VCL Forms Application - C++ Builder menuy bandini tanlang. Formaga komponentlar palitrasining Standard sahifasidan Button, Edit va ListBox komponentalarini joylashtiramiz.



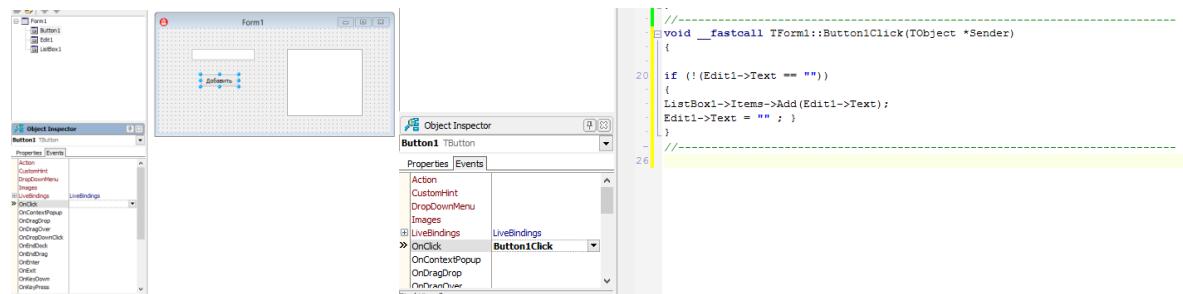
5.9 – rasm. Komponentlarni formaga joylashtirish

Shundan so‘ng, formadagi Edit komponentini tanlang va Text xususiyatining joriy qiymatini o‘chiring. Keyinchalik, Button1 ning Caption xususiyatini "Qo‘shish" ga o‘rnating.



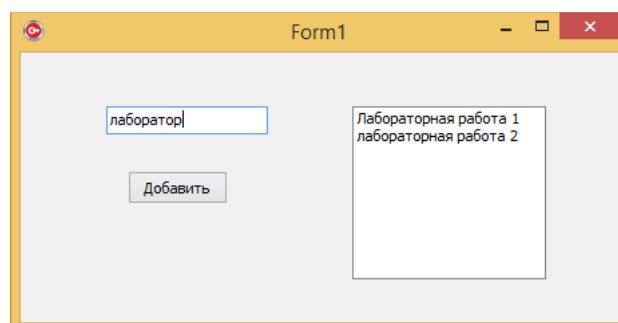
5.10 – rasm. Edit1 uchun Matn xususiyatlarini va Button1 uchun Caption xususiyatlarini o‘zgartirish

Qo‘shish tugmasi uchun OnClick hodisasi ishlov beruvchisini qo‘shish uchun formadagi tugmani tanlang, obyektlar inspektoridagi Voqealar sahifasini oching va OnClick hodisasining o‘ng tomonidagi ustunni ikki marta bosing. Funktsiya nomi mos keladigan kiritish qatorida paydo bo‘ladi. Visual C++ prototip hodisa ishlovchisini yaratadi va uni kod muharririda ko‘rsatadi. Shundan so‘ng, funktsiya tanasining { ... } operator qavslariga quyidagi kodni kiritishingiz kerak:



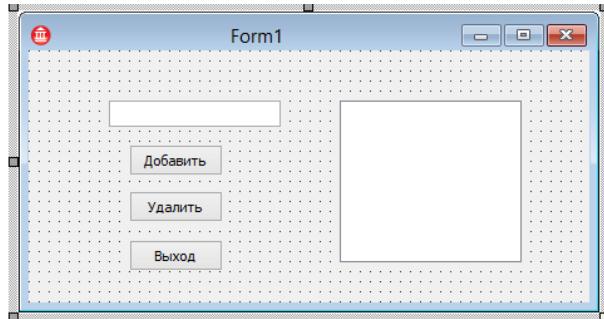
5.11 - rasm. OnClick voqealari ishlovchisini o‘zgartirish

Ilovani kompilyatsiya qilish uchun Run menyusidan Run-ni tanlang. Endi siz tahrirlanadigan maydonga biror narsa kiritishingiz mumkin, Qo‘shish tugmasini bosing va kiritilgan qatorlar ro‘yxatga qo‘shilganligiga ishonch hosil qiling.



5.12 - rasm. Yaratilgan dastur.

Endi Delete va Exit tugmalarini qo'shish orqali dasturni o'zgartiramiz. Buning uchun keling, yana ikkita tugma qo'shamiz, ularning Caption xususiyatini o'zgartiramiz va ushbu tugmalarni bosish bilan bog'liq hodisa ishlov beruvchilarini yaratamiz:



5.13 - rasm. Takomillashtirilgan ilova

**O'chirish** tugmasi uchun:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    if (!(ListBox1->ItemIndex == -1))
        ListBox1->Items->Delete(ListBox1->ItemIndex);
}
```

**Chiqish** tugmasi uchun:

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    Close();
}
```

Keling, dasturni saqlaymiz va kompilyatsiya qilamiz va keyin uni sinab ko'ramiz.

Shunday qilib, biz Visual C++ dasturlash muhiti bilan tanishdik va oddiy dastur yaratdik. Quyidagi laboratoriyalarda formadagi komponentlarni manipulyatsiya qilish usullari tasvirlanadi va ilovadagi turli komponentlarning xatti-harakatlari batafsilroq ko'rib chiqiladi.

**Nazorat savollari:**

1. Visual C++ dasturlash muhiti nima?
2. Visual C++ qanday komponentlardan iborat?
3. Komponentning xossalari qanday?
4. Obyekt inspektorining Voqealar sahifasi nima uchun ishlatiladi?
5. Visual C++ da usul?
6. Visual C++ loyihalar menejeri haqida gapirib bering

## **5.1.2-§ KOMPONENTA TUSHUNCHASI KOMPONENTALAR BILAN ISHLASH.**

**Reja:**

- 1. Komponenta tushunchasi.**
- 2. Komponenta xususiyatlari.**
- 3. Formalar bilan ishslash.**

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 ishslashini 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 ishslash, ya’ni oldindan hosil qilingan shablonlar asosida hosil qilinadigan holatlar bilan tanishamiz.

Konsol oynasi - bu grafik interfeysga ega bo‘limgan, 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 ishslash 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 astur 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 obyektni joylashtirish, xotirani bo‘shatish, obyekt bilan ishlash vaqtি kabi parametrlarni bajarar edik. Boshqa tomondan, CLR rejimi yoqilganda, bu ilova boshqa ilovalarga nisbatan tubdan farq qiladi, bunda ilovaga ularish System muhiti orqali amalga oshiriladi, bunda obyektlarning xotiraga joylashishi avtomatik boshqariladi.

Regulyatsiyalanuvchi ko‘rsatkich - bu shunday ko‘rsatkich turiki, bunda havola orqali obyektning “to‘plam” xotiradagi o‘rni ko‘rsatiladi, bu holat sinov paytida amalga oshiriladi. Bunday ko‘rsatkichlar uchun “\*” o‘rniga “A” 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 malumotlarni biz keyingi boblarimiz davomida ko‘rib chiqamiz.

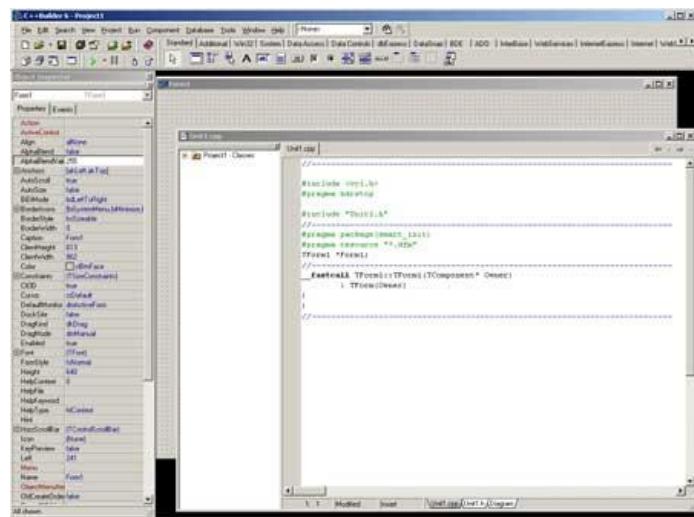
Agar ishchi stolimizning strukturasiga e'tibor bersak, har bir oynaning sarlavhasi o'z tarkibida bo'lgan ko'plab funksional masalalarga mos ravishda tanlangan. Bu oynalarni sichqonchaning chap tugmasi yordamida istalgan joyga joylashtirish mumkin. Oynalarni o'zaro birlashtirib, bir nechta vkladkalar ko'rinishida ham joylashtirsa bo'ladi. Siz agar biror bir oynani tanlab, sichqoncha yordamida kerakli joyga o'rnatmoqchi bo'lsangiz, darhol sizga kerakli obyektni 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 kata 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 **Вид** menyusi orqali ham amalga oshirish mumkin.

Ilovalar interfeysi yaratish uchun Delphi vizual komponentalarining ulkan (keng) to'plamini taqdim qiladi. Ularning asosiyлари komponentalar palitrasining Standart, Additional va Win32 varaqlarida joylashgan. Biz ushbu componentalardan foydalanishni turli dasturlash muhitlarida kurishimiz mumkin,



10.1-rasm Vizual komponenta interfeysi.

Standart varaqda interfeys komponentalarining aksariyati Windows boshlang‘ich versiyalirida ishlatilgan interfeys komponentalaridan iborat:

Frames - Freymlar

MainMenu - Asosiy Menu

PopurMenu - Paydo bo‘luvchi menu

Label - Yozuv

Edit - Bir satrli taxrir

Memo - Ko‘pqatorli taxrir

Button - Standart tugma

CheckBox - Bog‘liqmas (pereklyuchatel)

RadioButton - (pereklyuchatel)

ListBox - Ro‘yxat

ComboBox - Ro‘yxatli Maydon

ScrollBar - Harakatlantirish yulagi

GroupBox - Guruh

RadioGroup - O‘zaro bog‘liq (pereklyuchateli) guruhi

Panel - Panel

ActionList - Amallar ro‘yhati

## Standart Varog‘i



Rasmdagi komponentalar, ular sanab utilishiga mos keladi. Birinchi pikrogramma komponenta hisoblanmaydi va u varoqda tanlangan komponentadan voz kechish uchun xizmat qiladi. Additional varog‘ida komponentalar quyidagicha joylashgan:

BitBtn - Rasmli tugma

SpeedButton - Tezkor murojaat tugmasi

MaskEdit - Qolip buyicha berilganlar kiritiluvchi bir qatorli taxrir

StringGrid - Satrlar jadvali

DrawGrid - Jadval

Image - Grafik shakl

Shape - Geometrik figura

Bevel - Faska

ScrollBox - Harakatlantirishlar soxasi

CheckListBox - Pereklyuchatellar ro‘yhati

Splitter - Ajratuvchi

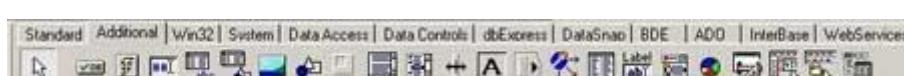
StaticText - Statik (turg‘un) matn

ControlBar - Vositalar (instrumentlar) paneli uchun konteyner

ApplicationEvents - Ilovaning hodisalari

Chart - Diagramma

## Additional varog‘i



Win 32 varog‘ida 32 razryadli Windows interfeysiiga ta’luqli komponentalar joylashgan.

TabControl - Zakladka

PageControl - Bloknot

ImageList - Grafik shakllar ro‘yhati  
RichEdit - To‘liq imkoniyatli matn rejimi  
TrackBar - Yugurdak (Begunok)  
ProgressBar - Ish bajarish indikatori  
UpDown - Xisoblagich  
HotKey - Qaynoq tugmalarkombinatsiyalri tahriri  
Animate - Videokliplarni tomosha qilish  
DateTimePicker - Sanani kiritish satri  
MonthCalendar - Kalendar  
TreeView - Obektlar daraxti  
ListView - Ro‘yhat  
HeaderControl - Ajratuvchi  
StatusBar - Holat satri  
ToolBar - Instrumentlar (asboblar) paneli  
CoolBar - "Epchil" instrumentlar paneli  
PageScroller - Tasvirda harakatlanish

### **Win32 varog‘i**



### **Vizual komponentalarning umumiy xususiyatlari**

Barcha vizual komponentalar uchun TControl sinfi asos hisoblanadi va u elementning o‘lchami va joylashuvi, uning sarlohasi ,rangi va shunga uxshash parametrlaridan iborat asosiy funksional atributlarni ta’minlaydi . TControl sinfi vizual komponentalar uchun umumiy bo‘lgan xossalari , hodisalar va metodlarni o‘z ichiga oladi . Vizual komponentalarni ikkita katta guruhga ajratish mumkin :

1. To‘g‘ri to‘rtburchakli boshqaruv elementlari
2. To‘g‘ri to‘rtburchakli bo‘lmagan boshqaruv elementlari

To‘g‘ri to‘rtburchakli boshqaruv elementi o‘zida ma’lum bir maqsad uchun aniqlangan maxsus to‘g‘ri to‘rtburchakni ifodalaydi . Bu elementlarga misol

tariqasida boshqaruv tugmalarini , taxrir maydonlari ,xarakatlanish yo‘laklarini ko‘rsatishimiz mumkin. Ular uchun asos sinf TWinControl hisoblanadi. To‘g‘ri to‘rburchak elementlari qiymar kiritish fokuslarini ilishi mumkin . Elementni fokus olganligi ikki xil usulda kursatiladi :

1. Taxrir kursori yordamida
2. To‘g‘ri to‘rburchak orqali

Matn taxrirlari bo‘lgan Edit va Memo elementlari o‘z sohasida tahrir kursori (matn kursori) paydo bo‘lishi orqali fokus olganligini bildiradi. Ma’lumotlarni tahrirlash bilan bog‘liq bo‘lmagan komponentalarda qora punktir chiziqli to‘g‘ri to‘rburchak paydo bo‘lishi uning fokus olganligini anglatadi . Masalan , Button tugmasi fokus olganda sarloha atrofida to‘g‘ri to‘rburchak paydo bo‘ladi , ListBox da esa ro‘yxatdagi ayni paytda tanlangan satrni ajratilgan holda (aksariyat hollarda ko‘k fonda) ko‘rsatadi. Bularidan tashqari , to‘g‘ri to‘rburchak boshqaruv elementlari konteyner sifatida o‘z ichida boshqa boshqaruv elementlarini olishi mumkin . Bu holda boshqaruv elementi o‘z ichidagilarga ota hisoblanadi. To‘g‘ri to‘rburchak bo‘lmagan boshqaruv elementlari TGraphicControl sinfining avlodlari hisoblanadi . Bu guruh elementi qiymat kiritish fokusini ololmaydi va interfeys elementlari uchun ota bo‘la olmaydi .

### **Nazorat savollari**

1. Ilova – bu
2. Standart Varog‘i komponentalarini keltiring.
3. Additional varog‘i komponentalarini keltiring.
4. Win32 varog‘i komponentalarini keltiring.
5. Vizual komponentalarning umumiy xususiyatlari.

### 5.1.3-§ KOMPONENTALAR BILAN ISHLASH.

**Reja:**

1. **Ma'lumotlarni kiritish komponentalari.**
2. **Ma'lumotlarni chiqarish komponentalari.**

**Label.**

**A** Shaklda tahrir qilib bo'lmaydigan statik matnning to'rtburchak sohasini aks ettiradi. Odatda matn boshqa komponenta nomidan iborat bo'ladi.

Nom matni Text xususiyatining qiymatidir. **Alignment** xususiyati matnni tekislash usulini aniqlaydi. Shrift o'lchami avtomatik tarzda sohaning maksimal to'ldirilishiga mos kelishi uchun, **AutoSize** xususiyatining **true** qiymatini o'rnating. Kalta soha ichida matnning hammasini ko'rish imkoniga ega bo'lish uchun, **WordWrap** xususiyatiga **true** qiymatini bering. **Transparent** xususiyatiga **true** qiymatini o'rnatsangiz, boshqa komponentaning bir qismini to'g'ri uning ustida joylashtirilgan nom orasidan ko'rinish turadigan qilishingiz mumkin.

**TextBox**  **TextBox**

Axborot yakka satrining tahrir qilinayotgan kiritishidagi to'rtburchak sohani shaklda aks ettiradi. Tahrir sohasining ichidagi boshlang'ich narsalarni **Text** xususiyatining qiymati bo'lan satr aniqlaydi.

**TextBox** komponentasi TtextBox sinfining to'g'ridan-to'g'ri hosilasi bo'lib, uning barcha xususiyatlari, metodlari va voqealariga vorislik qiladi.

**Button**

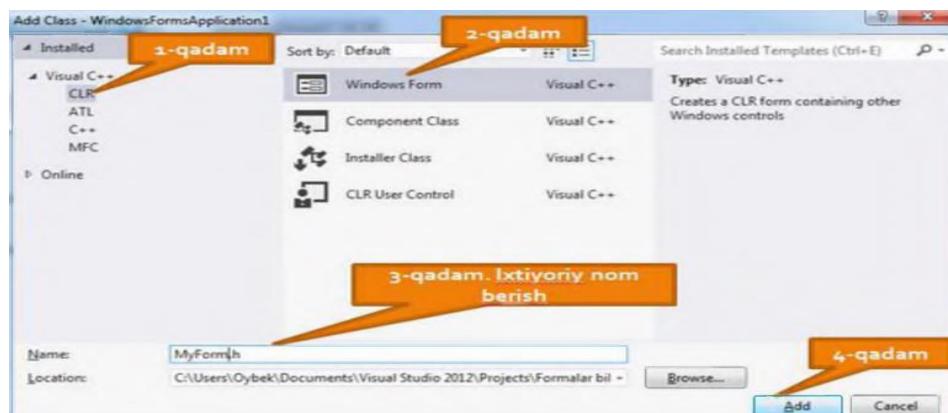


Yozuvli to'rtburchak tugmani yaratadi. Tugmacha bosilganda, dasturda biror-bir hatti-harakat nomlanadi (initsiallashtiriladi). Tugmachalar ko'proq dialogli darchalarda qo'lanadi. Default xususiyatining **true** qiymati tomonidan

tanlab olingan yashirin tugmacha, dialog darchasida har gal enter klavishasi bosilganda **OnClick** voqea qayta ishlatgichini ishga tushiradi. Cancel xususiyatining true qiymati tanlab olgan bekor qilish tugmachasi, dialog darchasida har gal **Escape** klavishasi bosilganda, **OnClick** voqea qayta ishlatgichini ishga tushiradi.

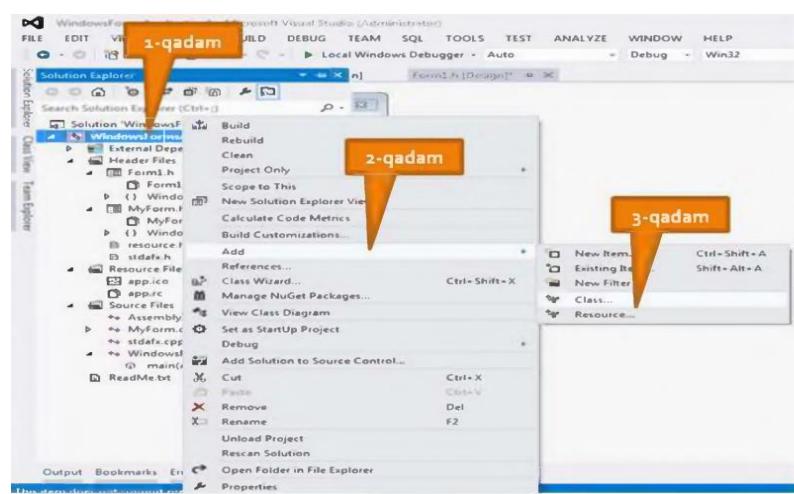
**Masalan:** Visual C++ dasturida Windows Application muhitida 1-formada  $a[N][M]$  massiv elementlarini  $[N; M]$  oraliqdagi tasodifiy sonlar bilan to‘ldiring.  $N$  va  $M$  ni TextBox komponetasi yordamida kriting. Massiv elementlaridan qiymati just bo‘lganlarining yig‘indisini 2-formaning label komponentasida chiqaring.

**1-qadam.** Yangi forma ochiladi.



11.1 - rasm. Yangi forma ochish oynasi

**2-qadam.** Loyihaga yangi forma qo‘shiladi.



11.2 - rasm. Yangi forma qo‘shish oynasi

**3-qadam.** 1-formaga 2 ta TextBox komponentalari (misoldagi N va M uchun) joylashtiriladi:

**4-qadam.** 1-formaga 2 ta dataGridView komponentasi (misoldagi N va M o'lchovli massiv elementlarini chiqarish uchun) joylashtiriladi:

**5-qadam.** 1-formaga button1 tugmachasi joylashtiriladi va button1tugmachaSini 2 marta bosib, kodlar oynasida quyidagi kodlar teriladi:

```
1. #include "iostream"  
2. #include "conio.h"  
3. #include "stdlib.h"  
4. #include "form2.h"  
5. ...  
6. ...  
7. int s;  
8. #pragma endregion  
9. private: System::Void button1_Click(System::ObjectA sender,  
System::EventArgsA e) {  
10. int a[10][10];  
11. s=0;  
12. int n=Convert::ToInt32(textBox1->Text);  
13. int m=Convert::ToInt32(textBox2->Text);  
14. dataGridView1->ColumnCount = n;  
15. dataGridView1->RowCount = n;  
16. for (int i=0;i<n;i++){  
17. for (int j=0;j<m;j++){  
18. a[i][j]=rand()%50+1;  
19. dataGridView1->Columns[j]->HeaderText=(j+1).ToString()+" - ustun";  
20. dataGridView1->Rows[i]->Cells[j]->Value =a[i][j].ToString();  
21. if (a[i][j]%2==0){s+=a[i][j];}  
22. } }
```

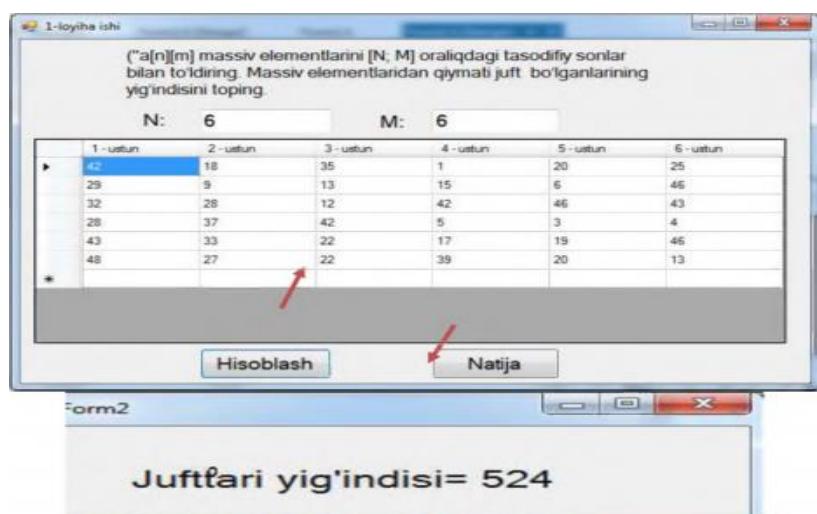
23. }

**6-qadam.** 2- formaga label komponentasi joylashtiriladi va uning Modifiers xossasi Public ga o‘zgartiriladi. Eslatma: Agar xossasi **Public** qilinmasa 1-formadan turib 2-forma komponentasini boshqarishga ruxsat berilmaydi.

**7-qadam.** 1- formaga **button2** tugmachaşini joylashtiriladi va uni sichqonchaning chap tugmasi bilan 2 marta bosib, kodlar oynasida quyidagi kodlar teriladi:

```
1. private: System::Void button2_Click(System::ObjectA sender,  
System::EventArgsA e) {  
2. Form2A open=gcnew Form2();  
3. th i s ->Hide();  
4. open->Show();  
5. open->label1->Text="Juf tari y ig findisi= " + s.ToString();}
```

Dastur natijasi quyidagi ko‘rinishda bo‘ladi:



11.3 - rasm. Dastur oynasi.

## **5.2-§ KOMPONENTALAR BILAN ISHLASH. TARMOQLANSH VA TANLASH UCHUN MO'LJALLANGAN KOMPONENTALAR. MASSIVLAR BILAN ISHLASH KOMPONENTALARI.**

**Reja:**

- 1. Komponentalar bilan ishash.**
- 2. Tarmoqlanish va tanlash uchun mo'ljallangan komponentalar.**
- 3. Massivlar bilan ishlash komponentalari.**

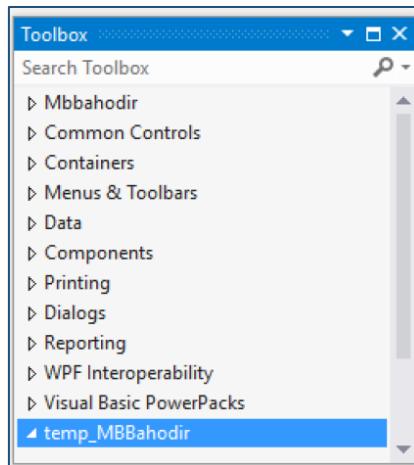
### **1. Komponentalar bilan ishash.**

MS Visual Studioda ishlash tamoylining asosiy qulayliklari bu komponentalaridir. Foydaluvchining interfeysini tahlil qilib qaralsa, unda juda ko'plab komponentalar joylashtirilgan. Bu juda osondir. MS Visual Studio Visual C++ daturli vazifalarni amalga oshirish komponentalari muvjud. Ularni asosan **Toolbox** deb ham yuritiladi. Toolbox dagi har bir komponentaning nomlar fazosi va sinflari mavjud. Ulardan foydalanganda shu sinflardan merosxo'r olib yaratiladi va bir oynada bir xil tipdagi komponentadan bir nechtaidan foydalanish mumkin.

**Komponenta tushunchasi va xususiyatlari.** Komponenta bu MS Visual Studio ning eng asosiy qurolidir. Dasturchi bular orqali tez dasturlash va IDE muhitning imkoniyatlaridan foydalaniladi. Komponentalar to'plamini **Toolbox** deb yuritiladi.

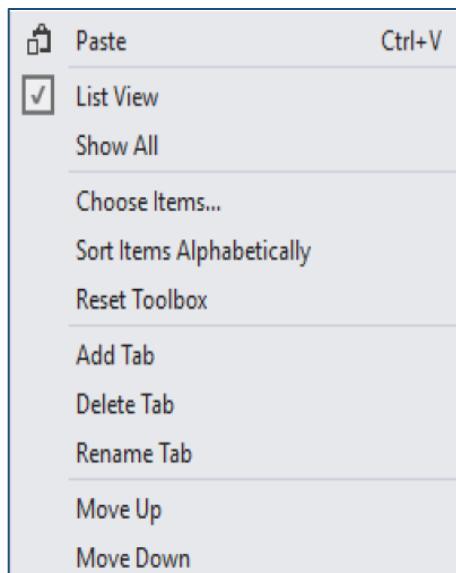
Toolbox interaktiv oynasi muhitning ixtiyoriy joyida joylashgan bo'lshi mumkin. Loyihalovchi uni o'ziga mos qilib joylashtirishi mumkin. Agar muhitning oynasida Toolbox oynasini ko'rmayotgan bo'lsangiz, menyudan foydalanib, [view → Toolbox] buyruqlarini bajaring. [Ctrl + W, X] tugmachalar majmuasini ham bosish orqali Toolbox oynasiga o'tish/chaqirish mumkin. Muhit komponentalar soni nechta degan savolga javob berishdan boshlaymiz. Muhit integrallashgan muhitligini bilamiz, shuning uchun undagi komponentalar soni oldindan xech kim aytaolmaydi. Yangi kutubxonalarini joriy qilish yangi

komponentalarni olib kiradi. Standart foydalanuvchilar uchun Toolbox oynasi 5.1-rasmda tasvirlangan.

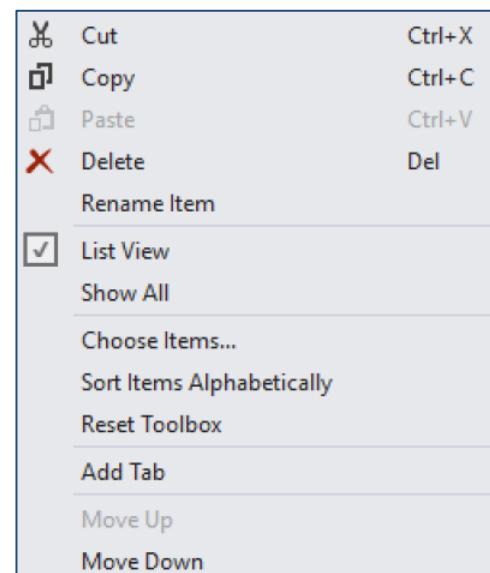


5.1-rasm. Toolbox oynasi

Toolbox oynasini tahrirlash mumkin. Buning uchun sichqonchaning o‘ng tugmasini Tab ga, ya‘ni komponentalar guruhining nomiga bosib, 5.2-rasmdagi kontekst menyuni, Tab ning ichiga kirib komponentani ustiga bossak 5.3-rasmdagi kontekst menyuni chaqiramiz.



5.2-rasm. Komponentalar guruhni uchun kontekst menyu.



5.3-rasm. Komponenta uchun kontekst menyu.

Rasmlarga qarasak, ularda bir xil nomli buyruqlar uchraydi, bular butun Toolbox oynasi uchun xizmat qiladi va 5 ta bo‘limdan iborat. Har bo‘limga va uning buyruqlariga to‘xtalib o‘tamiz:

**1– Bo‘lim.** Komponentalar ustida amal bajarish uchun mo‘ljallangan bo‘lib, [Cut]– komponentani buferga olish va joyidan o‘chirish uchun, [Copy]– komponentaning nusxasini olishsh uchun, [Paste]– komponentaning olingan nusxasini joylashtirish, buni komponenta guruhi qo‘llash mumkin, [Delete]– komponentani guruhdan o‘chirib tashlash, [Rename item]– komponenta nomini o‘zgartirish uchun ishlidi.

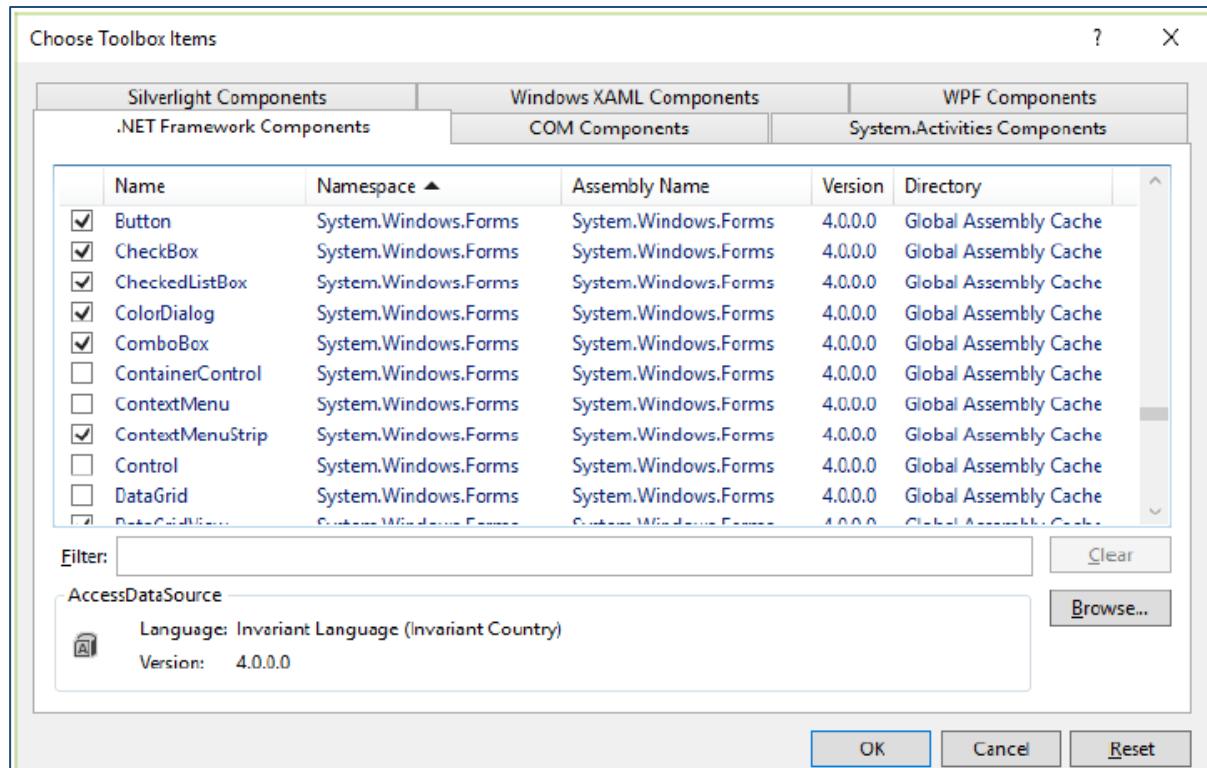
**2– Bo‘lim.** Toolbox oynasiga xizmat qiladi. [List View]– tanlagan guruh komponentalarini ro‘yxat qilib chiqaradi, o‘chirilgan bo‘lsa, uskunalar to‘plami sifatida ko‘rsatadi, [Show All]– muhitdagi barcha faollashtirilgan komponentalarni chiqarib beradi. Komponentlar juda ko‘pligi uchun bu ishlatish o‘ta mushkul ish deb hisoblanadi.

**3– Bo‘lim.** Toolbox oynasiga xizmat qiladi va komponentalar hosil qilish uchun ishlatiladi. [Choose Items..]– yangi komponentalarni qo‘sish uchun ishlatiladi, [Sort Items Alphabetically]– komponentalarni alfavit bo‘yicha saralash, [Reset Toolbox]– Toolbox komponentalarni sozlamalarini standart shaklga keltirish.

**4– Bo‘lim.** Toolbox oynasiga Tab larni, ya‘ni komponenta guruhlari ustida ishlash imkoniyatini beradi. [Add Tab]– yangi komponenta guruhi qo‘sish, [Delete Tab]– komponenta guruhini o‘chirish, [Rename Tab]– komponenta guruhini qayta nomlash, standart nomlaganlarini ham bu amallar bilan o‘zgartirish mumkin.

**5– Bo‘lim.** Komponenta guruhi va komponentalarning o‘rinlarini almashtirish uchun ishlatiladi. [Move Up]– yuqoriga ko‘chirish, [Move Down]– pastga ko‘chirish. Bularning barchasi muhitning dasturchiga moslashuvchanligini bildiradi.

Yangi komponentalarni qo'shishni ko'rib chiqaylik, bu buyruq bosilishi bilan 5.4-rasmda tasvirlangan muloqot oynasi chiqadi.



5.4-rasm. Yangi komponentalarni qo'shish.

Bu rasmdan komponentalarning turlarini ko'rish mumkin, har bir bo'limi 1000 dan ortiq sinflarga ega. Har bir sinf esa, bitta komponenta hisoblanadi. Agar tizimda mavjud bo'limgan komponentalarni ham shu oyna orqali qo'shish mumkin, masalan, MySql, oqimlar bilan ishlash, OpenCV, OpenMP, Fuzzy ToolBox kabi komponentalarni qo'shish mumkin.

**Komponentaning xususiyatlari.** Har bir komponentaning shunday xususiyatlari bor. Eslab ko'ring OYD da sinfni yaratish vaqtida nimalar yaratilar edi. Xuddi shuningdek, har bir komponeta sinf, uning xususiyatlari (properties), usullari (methods) va hodisalari (events) mavjud.

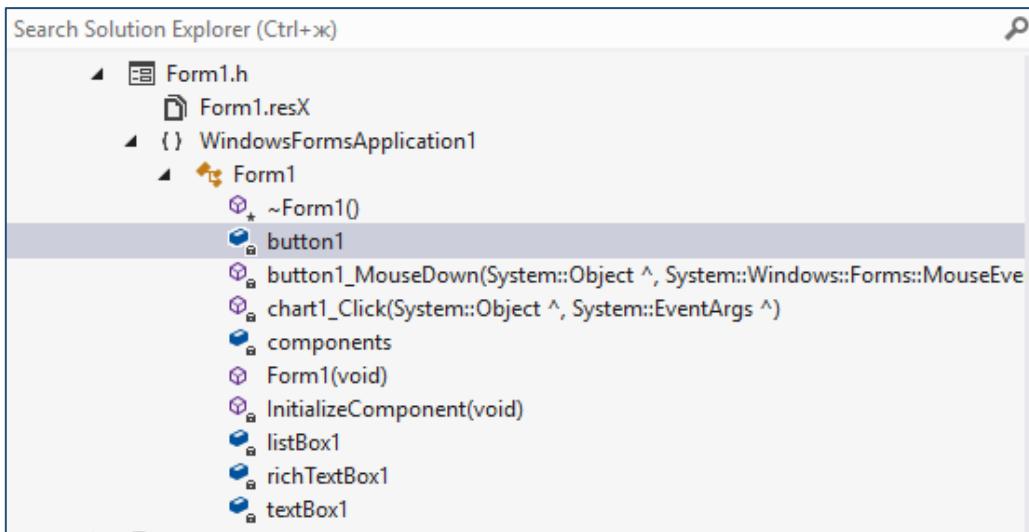
Komponentaning xususiyatlari quyidagi guruhlarga bo‘linadi (5.1-jadvalga qarang)

5.1-jadval. Komponentaning xususiyatlari

1	<b>Accessibility guruhi</b>	Bu guruhdagi xususiyatlar bir komponentada o‘zgarishlarni aniqlash va o‘rnatish uchun ishlatiladi.
2	<b>Appearance guruhi</b>	Komponentaning tashqi ko‘rinishi uchun xususiyatlarni o‘rnatishga mo‘ljallangan.
3	<b>Behavior guruhi</b>	Komponentaning rejim xususiyatlarini o‘rnatishga mo‘ljallangan.
4	<b>Data guruhi</b>	Komponentani ichki va tashqi ma’lumotlar bilan bog‘lash xususiyayalari uchun ishlatiladi.
5	<b>Design guruhi</b>	Komponentani loyihalash xususiyatlari uchun ishlatiladi
6	<b>Focus guruhi</b>	Komponentada fokuslarni boshqarish xususiyatlari uchun ishlatiladi
7	<b>Layout guruhi</b>	Komponentani tartiblash xususiyatlari uchun ishlatiladi
8	<b>Misc guruhi</b>	Boshqa xususiyatlarni o‘rnatish, komponentani turiga qarab o‘zgarib turadi.

Bu xususiyat guruhlari komponenta tipiga moslashgan holda doimiy o‘zgarib turadi va tipga mos xususiyatlarni aniqlab beradi. Komponenta xususiyatlarga qiymatlar maxsus, to‘plamdan tanlash, sonli, matnli, mantiqiy kabi ma’lum tiplar qiymatini oladi. Agar murakkab tip bo‘lsa, albatta muloqot oynasi orqali tanlash mumkin.

Komponenta xususiyatlariga qiymatlarni berish vizual amalga oshiriladi va asosiy formaga uning dastur fragmentlari yozib ketiladi. Buni loyiha boshqaruv oynasi orqali ko‘rish mumkin (5.5-rasmga qarang)



5.5-rasm. Komponentalarning qo'shilganligi.

Komponentaning **usullari (methods)** va **hodisalari (events)** quyidagi guruhlarga bo'linadi (5.2-jadvalga qarang)

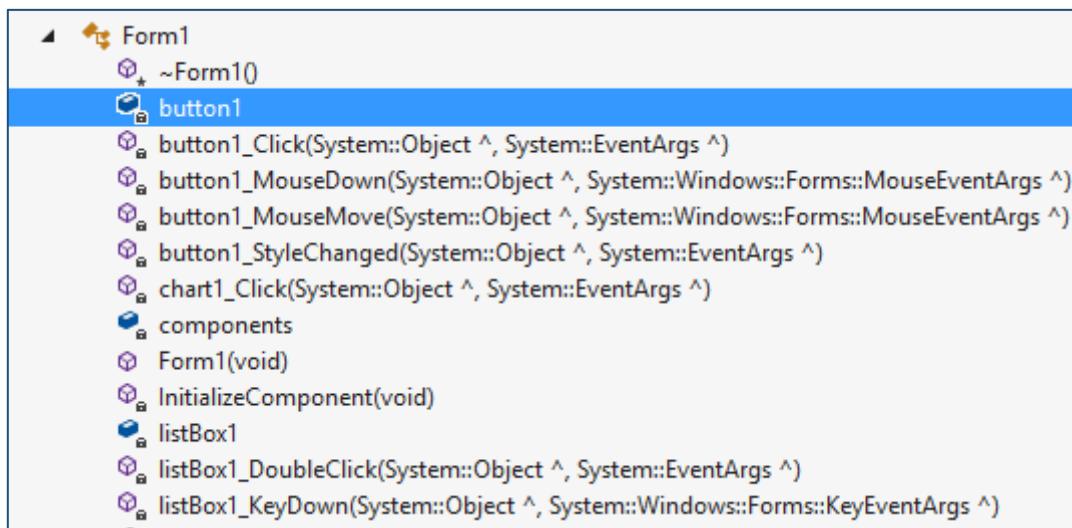
5.2-jadval. Komponentaning usullari (methods) va hodisalari (events)

1	<b>Action guruhi</b>	Komponenta uchun global hodisalar faollashtirish va ma'lum bir algoritm yozish
2	<b>Appearance guruhi</b>	Komponentaning ichki hodisalari uchun ishlatiladi va ma'lum bir algoritm yozish
3	<b>Behavior guruhi</b>	Komponenta holatlari uchun hodisalarni faollashtirish
4	<b>Data guruhi</b>	Ichki va tashqi ma'lumotlar bilan ishlash hodisalarini faollashtirish
5	<b>Drag drop guruhi</b>	Komponentaning harakatlanish hodisalarini faollashtirish
6	<b>Focus guruhi</b>	Komponentada fokuslarni boshqarish hodisalarini faollashtirish
7	<b>Key guruhi</b>	Komponentaning mos tugmalar bilan ishlash hodisalarini faollashtirish
8	<b>Layout guruhi</b>	Komponentada atrofidagilar bilan ishlash tartibini nazorat qilish hodisalarini faollashtirish

9	<b>Mouse guruhi</b>	Komponentada sichqoncha bilan bo‘ladigan hodisalarini faollashtirish
10	<b>Property Changed guruhi</b>	Komponentaning xususiyatlari o‘zgarganda bajariladigan hodisalarini faollashtirish

Bu usullar va hodisalarining odatda hammasi ham vizual ishlaish imkonini yo‘q. Shuning uchun agar komponentaning biror funksiya **On** bilan boshlansa bilingki bu usul, **ed**, **ing** bilan tugagan funksiyalari bo‘lsa, bular ko‘proq hodisalar hisoblanadi. Guruhrar komponenta tipiga moslashgan holda doimiy o‘zgarib turadi va tipga mos hodisalarini aniqlab beradi.

Komponenta hodisalarini funksiya sifatida yaratiladi va dasturchi unga kerakli o‘zining algoritmini yozadi. Komponenta hodisalarini qo‘shish vizual amalga oshiriladi va asosiy formaga uning dastur fragmentlari yozib ketiladi. Ammo dasturchi o‘z funksiyasining algoritmini yozishi kerak. Buni loyiha boshqaruv oynasi orqali ko‘rish mumkin (12.6-rasmga qarang)



5.6-rasm. Komponentalarning hodisalari.

Shuningdek, ba‘zi komponentalar faol va faol bo‘lmasdan holatlarda bo‘lishi mumkin. Bu huddi windows oynalaridagi buyruqlarning foydalanishiga o‘xshash ishlatiladi.

Komponentalarni o‘z joyida ishlatishni ham bilish kerak va ularni qachon ishlatish kerakligini bilash ham muhim.

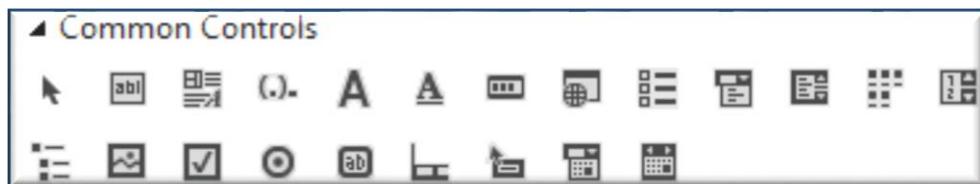
**Standart komponentasining xususiyatlari va hodisalari.** Standart komponentalarga Toolbox oynasidagi **[Common Controls]** Tab dagi komponentalar kiradi. Ularni tarkibini o‘zgartirish mumkin, ya‘ni yangilarini qo‘sish va keraksizlarini o‘chirib tashlash mumkin. Shuningdek, agar maxsus ishlab chiqilgan yangi komponentalar guruhi bo‘lsa ulardan ham foydalanish mumkin. Standart komponentalarni quyidagi guruhlarga bo‘linadi va shu asosida ularning hususiyatlari va hodisalari 95% bir xil bo‘ladi.

5.3-jadval. Standart komponentalarni guruhlari va vazifalari

Komponenta-ning guruhi	Komponenta-ning nomi	Vazifasi va izoh
Mantlarni tahrirlash	<b>TextBox</b>	Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtida foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	<b>RichTextBox</b>	Oddiy va RTF formatida Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtida foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	<b>Masked-TextBox</b>	Foydalanuvchi tomonidan kiritilayotgan matnlarni maxsus belgi bilan himoyalaydi
Ma’lumotlarni ko‘rsatish (faqat o‘qish uchun)	<b>Label</b>	Foydalanuvchi tomonidan to‘g‘ridan-to‘g‘ri tahrir qilinmaydigan matnni ko‘rsatadi.
	<b>LinkLabel</b>	Veb-link sifatida matnni ko‘rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havolani faollashtiradi.
	<b>ProgressBar</b>	Foydalanuvchi uchun joriy amal bajarilish jarayonini ko‘rsatadi.
Veb-sahifani ko‘rsatish	<b>WebBrowser</b>	Foydalanuvchi oynasida veb-sahifalar orqali harakat qilish imkonini beradi.
Ro‘yxatdan tanlash	<b>Checked-ListBox</b>	Har biri katak bilan birga bo‘lgan elementlarning takrorlanadigan ro‘yxatini ko‘rsatadi.
	<b>ComboBox</b>	Qalqib chiquvchi elementlar ro‘yxatini ko‘rsatadi

Komponenta-ning guruhi	Komponenta-ning nomi	Vazifasi va izoh
Grafiklarni tasvirlash parametrlari qiymat	<b>ListBox</b>	Matnlar va grafik elementlar ro'yxatini ko'rsatadi
	<b>ListView</b>	Elementlarni to'rt xil ko'rinishdan birida ko'rsatadi. Ko'rinishlarga faqat matn, kichik piktogrammali matn, katta piktogrammali matn va batafsil ko'rinish kiradi.
	<b>Numeric-UpDown</b>	Foydalanuvchilar yuqoriga va pastga tugmalari yordamida o'tiish mumkin bo'lgan raqamlar ro'yxatini ko'rsatadi.
	<b>TreeView</b>	Qo'shimcha kataklar yoki piktogrammalar bilan matndan iborat bo'lishi mumkin bo'lgan tugun ob'ektlarining iyerarxik to'plamini ko'rsatadi.
Buyruqlar	<b>PictureBox</b>	Bitmaps va piktogramma kabi tasvir fayllarini ko'rsatadi.
	<b>CheckBox</b>	Matn uchun katakcha va yorliqni ko'rsatadi. Odatta parametrlarni belgilash uchun ishlataladi.
	<b>Checked-ListBox</b>	Har biri katak bilan birga bo'lgan elementlarning qaytariladigan ro'yxatini ko'rsatadi.
	<b>RadioButton</b>	Yoqilgan yoki o'chirilishi mumkin bo'lgan tugmani ko'rsatadi.
	<b>TrackBar</b>	Foydalanuvchilarga shkala bo'ylab "Thumb" ni harakatlantirib shkala qiymatlarini o'rnatish imkonini beradi.
Foydalanuvchi uchun yordam	<b>Button</b>	Boshlanadi, to'xtaydi yoki jarayonni bekor qiladi.
	<b>LinkLabel</b>	Veb-link sifatida matnni ko'rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havola faollashtiradi.
	<b>NotifyIcon</b>	Fonda ishlaydigan dasturni ifodalovchi vazifalarni paneli holati xabarnomasidagi belgini ko'rsatadi.
Vaqt	<b>HelpProvider</b>	Elementlari uchun pop-up yordam oynasini yoki tezkor yordam oynasini beradi.
	<b>ToolTip</b>	Foydalanuvchi nazorati bilan hovers nazorat maqsadi bo'yicha qisqacha tavsifi ko'rsatadi va bir pop-up oyna beradi.
	<b>Date-TimePicker</b>	Foydalanuvchilarga sana yoki vaqtni tanlash imkonini beradigan grafik taqvim ko'rsatadi.
	<b>Month-Calendar</b>	Foydalanuvchilarga sana qatorini tanlash imkonini beradigan grafik taqvim ko'rsatadi.

Standart komponentalarni guruhlashning asosiy maqsadi, guruhlarga mos xususiyatlar va hodisalar bir xil bo‘lishini ta’minlash. Bu komponentalar Toolbox oynasida quyidagicha ko‘rinishga ega (5.7-rasmga qarang)



5.7-rasm. Standartkomponentalarko‘rinishi.

Bu komponentalarni oynaga joylashtirish uchun sichqonchaning chap tugmasi bitta bosiladi va oynaga keltirib chap tomoni yana bir marta bosiladi va oynada tanlangan komponentaning nusxasi paydo bo‘ladi. Bu nusxani ob’ekt deb aytamiz. Chunki eslab ko‘ring, sinfning nusxasi bu ob’ekt deb aytildi. Ob’ektning ustiga sichqonchaning o‘ng tugmasini bossangiz, u bilan bajarish mumkin bo‘lgan amallarni ko‘rasiz. Agar ob’ektdan nusxa olsangiz uning super sinfining, ya‘ni shu ob’ekt tipidagi boshqa bir ob’ekt yaratiladi. Ikkita bir xil tipdagi ob’ektlarning xususiyatlari va hodisalariga alohida – alohida ishlov beriladi.

Standart komponentalarni xususiyat va hodisalarini ko‘rib chiqish uchun arifmetik amallarni bajaruvchi dastur yaratish masalasini olamiz.

Mantiqiy jihatdan sonlarni kiritish uchun 2 ta NumericUpDown, matnli yozuvlar uchun 4 ta **Label**, hisoblash uchun bitta **Button**, amallar uchun bitta **CheckBox** komponentlarini olamiz va oynaga quyidagi 5.8-rasmdagidek qilib joylashtiramiz.



5.8-rasm. Standart komponentalarni xususiyat va hodisalaridan foydalanish.

Bu masalani loyiha sifatida amalga oshirish uchun quyidagi qadamlar bajariladi:

**1-qadam.** Forma xususiyatlarini o‘rnatish:

- BackColor xususiyatiga oq rang;
- font xususiyatiga 14 o‘lchamli yozuv;
- Icon ga maxsus ikonka;
- MaximizeBox xususiyatiga false qiymat;
- MinimizeBox xususiyatiga false qiymat;
- Size xususiyatiga yangi o‘lchamlar;
- StartPosition–CenterScreen holati;
- Text xususiyatiga “Easy Calculator” matnni yoziladi.

Buni qanday ajratish mumkin, ya‘ni o‘zgargan xususiyatlarni, qiymatining matni yog‘on qora ranga kirganlari o‘zgartirilgan hisoblanadi. Bu o‘zgarishlarni dastur fragmenti ko‘rinishida keltirish ham mumkin. Bu dastur fragmentini qo‘lda yozish kerak emas, ammo tushunarli bo‘lishi uchun keltiramiz. Buni **InitializeComponent** funksiyasi ichida kerakli izohlar bilan keltirilgan bo‘ladi.

```
this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::Font;
this->ClientSize = System::Drawing::Size(351, 225);
this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->HelpButton = true;
this->Icon = (cli::safe_cast<System::Drawing::Icon^>((resources-
>GetObject(L"$this.Icon"))));
this->MaximizeBox = false;
this->MinimizeBox = false;
this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
this->Text = L"Easy Calculator";
```

Label1

**2-qadam.** Ob'ektning xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->label1->Text = L"Birinchi son:";*

Label2

**3-qadam.** Ob'ektning xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->label2->Text = L"Ikkinchi son:";*

Label3

**4-qadam.** Ob'ektining xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->label3->Text = L"Natija:";*

*numericUpDown1*

**5-qadam.** Ob'ektning xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->numericUpDown1->TextAlign =*

*System::Windows::Forms::HorizontalAlignment::Right;*

*numericUpDown2*

**6-qadam.** Ob'ektning xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->numericUpDown2->TextAlign=System::Windows::Forms::*

*HorizontalAlignment::Right;*

**7-qadam.** ComboBox1 ob'ektining xususiyatlarini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->comboBox1->Items->AddRange(gcnew cli::array < System::Object^>(5){L"["+] (qo 'shish)", L"[-] (ayirish)", L"[\*] (ko 'paytirish)", L"[/] (butun bo 'lish)", L"[%] (qoldiq bo 'lish)"});*

*this->comboBox1->Text = L"tanlang...";*

**8-qadam.** label4 ob'ektning xususiyatini o'rnatish dastur fragmenti sifatida keltiramiz.

*this->label4->Text = L"Arifmetik amal";*

**9-qadam.** button1 ob'ektning xususiyatini va hodisani o'rnatish dastur fragmenti sifatida keltiramiz.

*this->button1->Text = L"Hisoblash";*

```

this->button1->Click += gcnew
System::EventHandler(this, &Form1::button1_Click);
button1_Click funksiyasiga quyidagia algoritmi yozamiz.

int one =
Convert::ToInt16(Math::Round(numericUpDown1->Value,0));
int two = Convert::ToInt16(numericUpDown2->Value);
int result = 0;
label3->Text = "Natija:";

switch(comboBox1->SelectedIndex){
case 0: result = one + two; break;
case 1: result = one - two; break;
case 2: result = one * two; break;
case 3: result = one / two; break;
case 4: result = one % two; break;
default: result = 0;
break;
}
label3->Text += result.ToString();

```

Algoritmga izoh berish shart emas, chunki ko‘p kamchiliklari mavjud, masalan, bo‘lish arifmetik amalida. Ammo ba‘zi xulosalarni chiqarish mumkin. [NumericUpDown1->Value] ob’ektning tanlangan haqiqiy tipdagi qiymatini qaytaruvchi xususiyat, [Convert::ToInt16] – ixtiyoriy tipni butun tipga almashtirib beradi, [Label3->Text] – ob’ektning matn xususiyatini o‘zgartirishi mumkin, [ComboBox1->SelectedIndex] – tanlagichning tanlangan indeksini qaytaradi, [Label3->Text += result.ToString();] – ob’ektning matniga matn qo‘shish va tipni satrli tipga o‘tkazish ko‘rsatilgan.

Dastur fragmentida 3 tabelga [::] – sinfning usullariga murojaat qilish, [->] – ob’ektning xususiyatiga murojaat qilish, [.] – ob’ektning qiymati ustida bajarilishi kerak bo‘lgan amal.

Visual C++ da juda ko‘p usul va funksiyalarni ishlatish tamoyllari bor, ularni faqat kompilyatorning imkoniyatiga qarab ajratish mumkin.

Standart komponentalarning ba‘zi xususiyatlari va hodisalari, ular uchun ba‘zi usullarni ishlatini ko‘rdik. Bularni chuqurroq o‘rganish uchun ko‘proq amaliy ishlarni bajarish kerak.

**Additional komponentasining xususiyatlari va hodisalari.** Bu standart komponentalar bilan ishlash uchun qo‘shimcha komponentlardir. Bular asosan boshqaruv elementlarni guruhlash uchun ishlatiladi. Quyidagi jadvalda qo‘shimacha komponentalarni va vazifalarini keltiramiz.

5.4-jadval. Additional komponentalarining guruhlari va vazifalari

Komponenta-ning guruhi	Komponenta-ning nomi	Vazifasi va izoh
Boshqaruvlar elementlarni guruhlash	<b>Panel</b>	Elementlarni boshqarish uchun to‘plamini guruhlaydi.
	<b>GroupBox</b>	Nomlangan elementlarni boshqarish to‘plamini guruhlaydi.
	<b>TabControl</b>	Guruhangan ob’ektlarni samarali tashkil etish va kiritish uchun sohali sahifani taqdim etadi.
	<b>SplitContainer</b>	Bir biridan ajratilgan ikki panelni beradi. SplitContainer – Splitter o‘rniga mo‘ljallangan.
	<b>Table-LayoutPanel</b>	Kontent dinamik satr va ustunlardan iborat panelni ifodalaydi.
	<b>Flow-LayoutPanel</b>	Dinamik ravishda kontentni gorizontal yoki vertikal joylashtiradigan panelni ifodalaydi.

Bu komponentalar **Containers** nomli tabda joylashgan. Ularning ko‘rinishi 5.9-rasmda keltirilgan.



5.9-rasm. Containers komponentalari.

Bu komponentlarning xususiyat va hodisalariga alohida to‘xtalib o‘tamiz. Vazifalari elementlar guruhini boshqarishdan iborat, shu maqsadda ishlatiladigan xususiyat va hodisalarini keltiramiz.

**Panel komponentasi.** **Panel**— bu oddiy to‘rt tomonidan chegaralangan hududga o‘xshaydi. Uni butun sohani egallashi uchun o‘ng tomonidagi [Panel Task] dagi [dock in parent container] ni bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Ob’ektni harakatlantirish uchun chap tomonidagi 4 tatomonga yo‘naltirilgan tugmachadan foydalilanadi.

**AutoSize**— xususiyati true/false qiymatlarni qabul qiladi. Odatda chin qiymat ko‘proq ishlatiladi, chunki ob’ektga joylashtirilgan boshqaruv elementlarininng o‘lchamlariga mos holda o‘lchamni sozlaydi. Joriy holatda yolg‘on qiymatga ega bo‘ladi.

**Cursor**— xususiyati sichqonchaning kursor ko‘rinishini o‘rnatish imkonini beradi.

**BorderStyle**— ob‘oektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

**AutoScoroll**— ob’ektida birlashtirilgan elementlar to‘liq ko‘rinmasa, harakatlaguvchi tugmalarini o‘rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

**AutoScorollMargin**— bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o‘rnatida, Wight va Height ichki xususiyatlari bor.

**AutoScorollMinSize**—bu xususiyat harakatlanuvchi tugmachalar eng kichik o‘lchamlarini o‘rnatadi. Wight va Height ichki xususiyatlari bor.

**Dock**— ob’ektni oynaning qaysi qismida joylashtirish kerakligini o‘rnatadi.

**Margin**— ob’ekt chet qismlarida qoldirilishi kerak bo‘lgan masofalarini o‘rnatadi. Uning 5 ta xususiyati bor, barchasi, chap, o‘ng, yuqori va past.

**groupBox komponentasi.** **groupBox**– nomlagan elementlar guruhini birlashtirish uchun ishlatiladi. Ob'ektni harakatlantirish uchun chap tomonagi 4 tatomonga yo'naltirilgan tugmachadan foydalaniladi.

**Text**– bu xususiyat elementlar guruhini nomini oynaga chiqaradi.

**AutoSize**– xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o'lchamlariga mos ob'ektga o'zlcham ajratadi.

**BackColor**– ob'ektning fon xususiyati rangini o'rnatadi.

**BackgroundImage**– ob'ektning fon xususiyati Image ob'ektlarni o'rnatadi.

**Dock**– ob'ektni oynaning qaysi qismida joylashtirish kerakligini o'rnatadi.

**FlatStyle**– ob'ektning stil xususiyatini o'rnatadi.

**Font** – bu xususiyat ob'ektning yozuvlari uchun aniq bir standart yozuvni o'rnatish uchun ishlatiladi.

**MaximumSize**– ob'ektning eng katta o'lchamini belgilaydi.

**MinimumSize**– ob'ektning eng kichik o'lchamini belgilaydi.

**Padding**– ob'ektgacha bo'lgan masofalarni o'rnatish uchun ishlatiladi.

Uning 5 ta xususiyati bor, barchasi, chap, o'ng, yuqori va past.

**RightToLeft**– ob'ektning matnini o'ng tomonidan o'rnatish.

**TabIndex**– ob'ektning tab tugmasi bosilganda, nechinchi navbatda bo'lishini belgilashni o'rnatadi.

**tabControl komponentasi.** **tabControl** bu oddiy to'rt tomonidan chegaralangan hududga o'xshaydi. Unga yangi sahifa qo'shish yoki sahifani o'chirish uchun o'ng tomonidagi [tabControl Task] dagi [Add Tab] va [Remove Tab] ni bosish yetarli. Ob'ektni harakatlantirish uchun chap tomonagi 4 ta tomonaga yo'naltirilgan tugmachadan foydalaniladi.

**Anchor**– ob'ektni aftor yoki oyna bo'yiga tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to'rt ta qiymati o'ng, chap, yuqori va past.

**Appearance**– bu xususiyat ob'ektning tugmachalarining ko'rinishini o'rnatish uchun ishlatiladi. Uning qiymatlari [Normal], [Buttons], [Flat Buttons] qiymatlarni qabul qiladi.

**DrawMode**— ob’ektning xususiyatini chizilgan formasini o‘zgartrish uchun ishlatiladi.

**TabPages**— bu xususiyat to‘plam qiymatlarni qabul qiladi va uning nechta sahifa bo‘lishini ta‘minlaydi. U bilan ishlaganda har bir sahifaning alohida xususiyatlariga ishlov berish mumkin.

**Tabpage1**— bu ob’ektning sahifa xususiyatlarni o‘rnatish uchun ishlatiladigan ob’ekt hisoblanadi va panel ob’ektidek xususiyatlarga ega.

**tabStop**— ob’ekt bo‘yicha tab tugmasining harakatlanishini o‘rnatish va o‘chirish qiymatini o‘rnatadi.

**splitContainer komponentasi.** **splitContainer** ikkiga ajratilgan elementlarni boshqarish uchun to‘plamni hosil qiladi. Uni butun sohani egallashi uchun o‘ng tomonidagi [splitContainer Task] dagi [dock in parent container] ni bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Shuningdek, [Horizontal Splitter Orientation] – ob’ekt ramkalarini gorizontal o‘rnatadi va [Vertical Splitter Orientation] esa vertikal o‘rnatadi. Ob’ektni harakatlantirish uchun chap tomonidagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

**Anchor**— ob’ektni aftor yoki oyna bo‘yicha tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati (o‘ng, chap, yuqori, past) bor.

**Backcolor**— ob’ektning fon xususiyati rangini o‘rnatadi.

**BorderStyle**— ob’ektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

**FixedPanel**— ob’ekt uchun asosiy panelni o‘rnatish uchun, ya‘ni ikkita paneldan biri asosiy bo‘lishi kerak.

**isSplitterFixed**— ob’ekt uchun asosiy panelni o‘rnatish amal qilishi yoki qilmasligini o‘rnatadi, xususiyat true/false qiymatlarni qabul qiladi.

**Orientation**— ob’ektning panellarini gorizontal yoki vertikal o‘rnatish uchun ishlatiladi. Ikkita Horizontal va Vertical qiymat qabul qiladi.

**splitContainer1.Panel1**— ob'ektning panel xususiyatlariga ishlov berish uchun panel xususiyatlarini ochib beradi va u uchun xususiyatlarni o'rnatish mumkin.

**Panel1Collapsed**— ob'ektda panelni to'liq qilib o'rnatadi. Bunday xususiyat barcha panellar uchun o'rinlidir.

**Panel1MinSize**— ob'ektdagi panelning eng kichik o'lchamini o'rnatishdan iborat.

**RightToLeft**— ob'ekt panellarini o'ngdan o'rnatish xususiyati.

**tableLayoutPanel komponentasi.** **tableLayoutPanel**— bu jadval kabi tartiblangan panellar to'plami orqali elementlarni birlashtirish ob'ektidir.

**AutoScoroll**— ob'ektda birlashtirilgan elementlar to'liq ko'rinsasa, harakatlanuvchi tugmalarni o'rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

**AutoScorollMargin**— bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o'rnatida, Wight va Height ichki xususiyatlari bor.

**AutoScorollMinSize**— bu xususiyat harakatlanuvchi tugmachalar eng kichik o'lchamlarini o'rnatadi. Wight va Height ichki xususiyatlari bor.

**CellBorderStyle**— ob'ektning yacheykalarining chet chegaralarining stilini belgilaydi.

**ColumnCount**— ob'ektdagi ustunlar sonini o'rnatish.

**Columns**— ob'ekt ustunlari uchun xususiyatlarni o'rnatish uchun alohida muloqot oynasi ochiladi.

**GrowStyle**— ob'ektga ustun yoki qator yoki ikkalasini qo'shish xususiyatini o'rnatish.

**RowCount**— qatorlar soni

**Rows**— ob'ekt qatorlari uchun xususiyatlarni o'rnatish uchun alohida muloqot oynasi ochiladi.

**flowLayoutPanel komponentasi.** **flowLayoutPanel**— ketma ketlik kabi tartiblangan panellar to'plami orqali elementlarni birlashtirish ob'ektidir.

**Anchor**—ob’ektni aftor yoki oyna bo‘yiga tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati (o‘ng, chap, yuqori, past) bor va bularning o‘zaro aralashuvidan foydalaniladi.

**Backcolor**— ob’ektning fon xususiyati rangini o‘rnatadi.

**AutoSize**— xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o‘lchamlariga mos ob’ektga o‘lcham ajratadi.

**WrapContents**— ob’ektda son tentlarni ketma ketligini ustuvor qilish yoki qilmaslikni o‘rnatadi, xususiyat true/false qiymatlarni qabul qiladi.

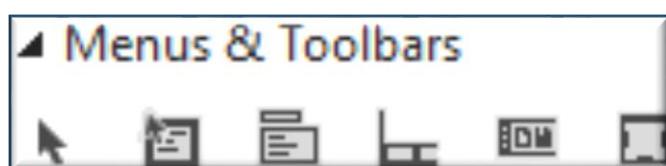
Bu qo‘shimacha komponentalarning xususiyatlari haqida ma’lumotlar berildi. Ularning hodisalari bir biriga o‘xshash bo‘lib, standart komponentalardan farqi juda kam. Shuning uchun hodisalarga to‘xtalaib o‘tirmaymiz. Shuningdek, qo‘shimacha komponentalar juda ko‘p, buni ko‘rish uchun Tollbox oynasiga sichqonchaning o‘ng tugmasin ibosing va [Shown All] buyrug‘ini tanlang.

**System komponentasining xususiyatlari va hodisalari.** Bu komponentalarga asosan operatsion tizim bilan ishlaydigan komponentalar kiradi. Ularni xususiyatlari murakab hisoblanadi. Chunki birgina menu yaratish uchun standart va qo‘shimcha komponentalardan ham foydalaniladi. Shuning uchun umumiyl holda bu komponentalarning vazifalarini keltirib o‘tamiz.

5.5-jadval. System komponentalarining guruhlari va vazifalari

Komponenta-ning guruhi	Komponenta-ning nomi	Vazifasi va izoh
Bo‘yruqlar	<b>ToolStrip</b>	Microsoft Windows, Microsoft Office, Microsoft Internet Explorer bo‘lishi mumkin uskunalar majmuasi yoki foydalanuvchi interfeysi yaratadi. ToolStrip control Asboblar paneli boshqaruvini almashtirishga mo‘ljallangan.
	<b>Tool-StripContainer</b>	ToolStrip elementlarini ajralmas to‘plamini beradi (birikib, sohasida gorizontal yoki vertikal almashish)

Komponenta-ning guruhi	Komponenta-ning nomi	Vazifasi va izoh
<b>Ma'lumotni aksettirish (faqat o'qish elementlari uchun)</b>	<b>StatusStrip</b>	Windowsning holat satrini beradi, Statusstrip – Statusbar o'rmini bosadi. Statusstrip maxsus xususiyatlari tartibini o'z ichiga oladi.
	<b>MenuStrip</b>	Forma uchun menu tizimini beradi. Menustrip – Mainmenu o'rmini bosadigan yuqori darajali konteyner hisoblanadi. Bundan tashqari, asosiy ishlov berish va hujjat interfeysi imkoniyatlarini beradi. Funksionaligi ToolStripitem olingan bo'lsada, ToolStrip Dropdownitem va ToolStripmenuItem bilan birga ishlatiladi.
<b>Menyuni boshqarish elementlari</b>	<b>Context-MenuStrip</b>	Kontekst menyuni ifodalaydi. Contextmenustrip bu Contextmenu o'rmini bosadi. Contextmenustrip bilan birga ishlatiladi va sichqonchaning o'ng tugmasini bosishda avtomatik ravishda kontekst menyusi namoyon bo'ladi.



5.10-rasm. Menyu va uskunalar panellarini yaratish uchun komponentalarning ko'rinishi

Menyu va uskunalar panellarini yaratish uchun komponentalardan tashqari tizim bilan ishlaydigan komponentalarga quyidagi 5.6-jadvaldagi komponentalar ham qiradi, ular **Components tab** ga kiritilgan.

5.6-jadval. Components tab ning komponentalari.

<b>Komponentaning nomi</b>	<b>Vazifasi va izoh</b>
<b>Background-Worker</b>	Alovida ajratilgan oqim uchun amal bajara oladigan Background-Worker komponentaning nusxasini yaratadi
<b>DirectoryEntry</b>	ActiveDirectory xizmati bilan o‘zaro bog‘lanib ishlashi va uning iearxiyasidan ob’ekt yoki tugunlarni inkapsulyatsiya oladigan DirectoryEntry komponentaning nusxasini yaratadi
<b>Directory-Searcher</b>	ActiveDirectoryda so‘rovlarni bajarish imkonini beradigan DirectorySearcher komponentaning nusxasini yaratadi
<b>ErrorProvider</b>	Formada elementni boshqarishdagi xato foydalanuvchiga ko‘rsatish uchun ErrorProvider komponentaning nusxasini yaratadi
<b>EventLog</b>	Tizim va foydalanuvchining hodisalar jurnaliga bog‘lanish imkonini beradigan EventLog komponentaning nusxasini yaratadi
<b>FileSystem-Watcher</b>	Ruxsat berilgan papka va fayllarga o‘zgartirishlar kiritish imkonini beradigan FileSystemWatcher komponentaning nusxasini yaratadi
<b>HelpProvider</b>	Tezkor va foydalanuvchining yordam oynasini chaqirish imkonini beruvchi HelpProvider komponentaning nusxasini yaratadi
<b>ImageList</b>	Image ob’ektlarining to‘plamini boshqarish usullarini ta‘minlovchi ImageList komponentaning nusxasini yaratadi
<b>MessageQueue</b>	Navbatdagi xabarlarni o‘qish imkoniyatini beruvchi Message-Queue komponentaning nusxasini yaratadi

Komponentaning nomi	Vazifasi va izoh
<b>Performance-Counter</b>	Windowsning hisoblagichlari bilan ishlash uchun foydalilanadi, yangi toifalar va misollar yaratish, hisoblagichlardan qiymatlarni o‘qish va hisoblagich ma’lumotlariga asoslangan hisob-kitoblarni bajarish uchun PerformanceCounter komponentaning nusxasini yaratadi
<b>Process</b>	Tizimdagi jarayonlar bilan bog‘liq ma’lumotlarni to‘xtatish, ishga tushirish va o‘zgartirish uchun Process, komponentaning nusxasini yaratadi
<b>SerialPort</b>	Sinxron va voqeal imkoniyatlar uchun SerialPort komponentaning nusxasini yaratadi
<b>Service-Controller</b>	Mavjud xizmatlarni, shu jumladan, boshlang‘ich va to‘xtatish xizmatlarini boshqarish va ularga buyruqlar berish uchun ServiceController komponentaning nusxasini yaratadi
<b>Timer</b>	Windows ilovalar uchun vaqtga asoslangan vazifalarni kiritish uchun Timer komponentaning nusxasini yaratadi



5.11-rasm. Components tab ga kiritilgan komponentalarning ko‘rinishi.

Bu komponentalardan kontkst menyuni yaratish va boshqa ob’ektlarga o‘rnatishni ko‘rib chiqamiz.

**Masala.** Oynada ikkita kontekst menu yaratish kerak. Birinchisi oyna uchun, ikkinchisi **RichTextBox** ob’ekti uchun bo‘lsin. Birinchi kontekstda oyna

yopish, kattalashtirish, kichiklashtirish, yig‘ishtirish, tugmachalarni yoqish va o‘chirish bo‘yruqlari bo‘lsin. Ikkinchi kontekstda matnni tahrirlash uchun buferga nusxalash, buferga nusxalash va o‘chirish, nusxa qo‘shish, ob‘ektni tozalash, gorozantal va vertikal harakatlanish tugmachalarni qurish va o‘chirish amallari bo‘lsin.

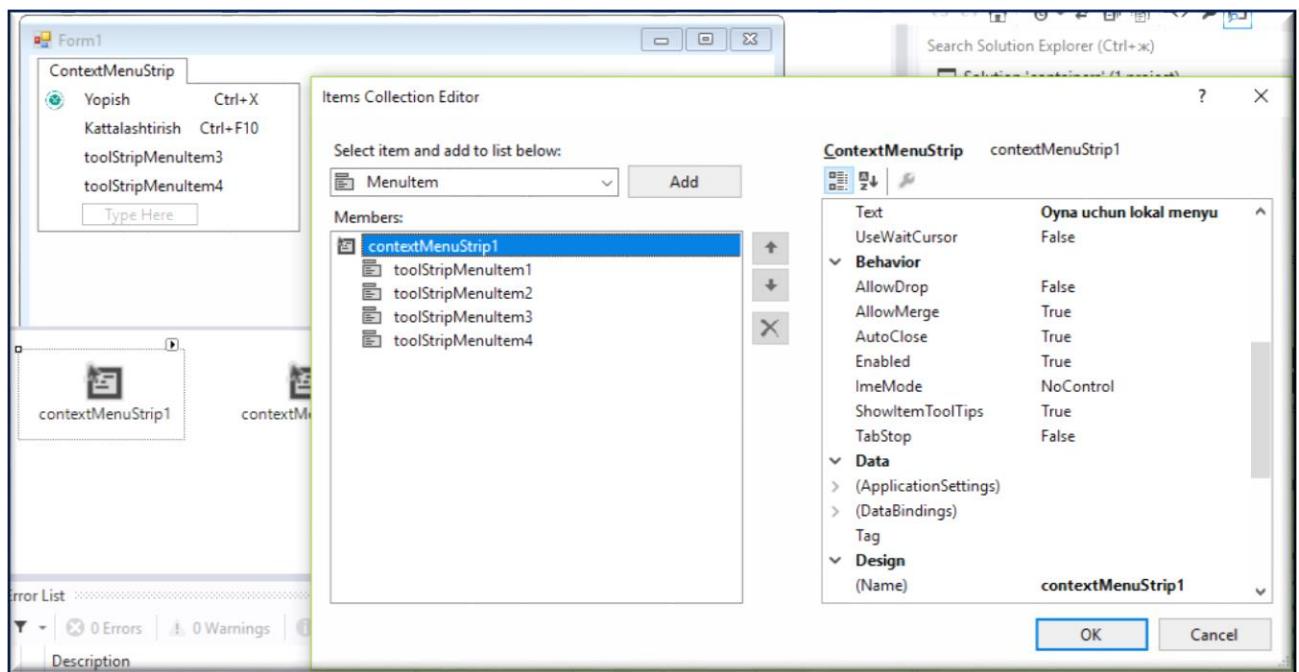
Yangi loyiha yaratamiz va bir oyna bo‘ladi. Oynaning bir chetiga **RichTextBox** ni o‘rnatamiz. Menyu va uskunalar tabidan **contextMenuStrip** ob‘ektidan ikkita o‘rnatamiz. Ular formada ko‘rinmasligi mumkin. Ular loyihaning ishchi maydoni pastki qismida joylashadi (5.12 – rasmga qarang).



5.12-rasm. contextMenuStrip ob‘ekti o‘rnatilgan ko‘rinishi.

Bularni kerakli ob‘ektlari bilan bog‘laymiz. Oynaga birinchi kontekst menyuni [contextMenuStrip1], [RichTextBox1] ob‘ektga ikkinchi kontekst menyuni [contextMenuStrip2] o‘rnatamiz. Buning uchun oynaning [ContextMenuStrip] xususiyatiga [contextMenuStrip1]ni va RichTextBox1] ob‘ektning [ContextMenuStrip] xususiyatiga [contextMenuStrip2] ni o‘rnatamiz. Agar bitta kontekstni ikkita ob‘ektga o‘rnatilsa, kontekstning buyruqlarga joriy qilingan algoritmlarni bajaraveradi.

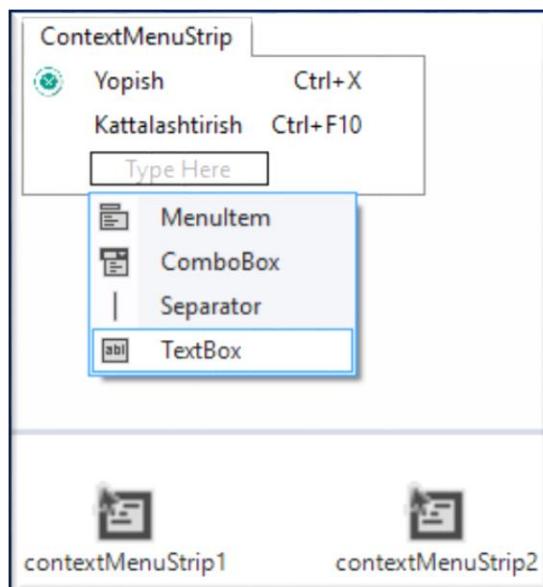
O‘rnatilgan kontekst menu ob‘ekti [ContextMenuStrip] xususiyati uchun ichki xususiyatlarni, ya‘ni [ContextMenuStrip] xususiyatlarni o‘rnatish uchun ochib beradi. Bu xususiyatlarni hammasiga to‘xtalib o‘tmaymiz. Faqat Items xususiyatiga to‘xtalamiz. Shu xususiyat tanlanganda quyidagicha muloqot oynasi chiqadi (5.13-rasmga qarang).



5.13-rasm. Kontekst menyu buyruqlarini o‘rnatish.

E‘tibor qaratsangiz, muloqot oynasida qilingan o‘rnatishlar muloqot oynasida emas balki asosiy oynada ko‘rinadi. Agar bu muloqot oynasida ishlash noqulay bo‘lsa, asosiy ishchi maydonda ham ishlash mumkin.

Buning kerakli kontekst menyu ob’ekti ustiga sichqonchani bir marta bosilsa, oynada kontekst menyu hosil bo‘ladi (5.14-rasmga qarang).



5.14-rasm. Oynada kontekst menyuga ishlov berish.

[Type Here] tugmasi orqali joriy [MenuItem], [ComboBox], [Separator], [TextBox] – 4 ta elementlarni qo’shish mumkin. Qo’shilgan har bir elementning o’ng tugmasini bosib, u bilan ma’lum bajarilishi mumkin bo’lgan amallarni bajarish mumkin. Shuningdek, [Properties] degan xususiyatlarga olib kiradi. [Text] – xususiyatiga kirib buyruq nomini berish mumkin. Agar harf bilan murojjat qilishni xoxlasangiz, kerakli harf oldiga [&] belgisini qo’yish kerak. Interaktiv tugma o’rnatish uchun esa [ShortcutKeys] xususiyatiga kerakli tugmalar kombinatsiyasini o’rnatish mumkin. Ikonka o’rnatish uchun [Image] xususiyatga kerakli rasmni o’rnatish mumkin. Shunday qilib kerakli barcha buyruqlarni amalga oshirish mumkin. Buyruqlarni guruhlash uchun [Separator] komponentasini tanlash kerak. Buyruqlarga algortim yozish uchun ularni ustiga sichqonchani ikkimarta bosish yetarli yoki hodisalariga kirib hodisasiga yozish mumkin.

Agar yuqorida kontekst menyuga ishlov berish tushunarli bo’lgan bo’lsa, masalani o’zingiz tugatib qo’ying. Agar ketma ketlik asosida bajarsangiz albatta masalani hech bo’lmasa, algoritmlarsiz hal qilishingiz lozim.

## **2. Tarmoqlanish va tanlash uchun mo’ljallangan komponentalar.**

Ko’pgina masalalarni yechishda ba’zi bir jarayonlar ma’lum shart yoki shartlarning qo’yilishiga nisbatan bajariladi. Bunday jarayonlar ***tarmoqlanuvchi jarayonlar*** deb yuritiladi. Tarmoqlanuvchi hisoblash jarayonlari oddiy va murakkab bo’lishi mumkin. Bu esa jarayondagi tarmoqlar soniga bog’liq. Ma’lum bir tarmoqlanuvchi jarayon tarkibida yana tarmoqlanishlar bo’lishi mumkin. Bunday tarmoqlanishlari bor bo’lgan hisoblash jarayonlari murakkab tarmoqlanuvchi hisoblash jarayonlari deb ataladi. C++ tilida tarmoqlanuvchi jarayonlarni dasturlash uchun shartsiz, shartli o’tish va tanlash operatorlaridan foydalilanadi: ***goto, if else, switch case***.

Aksariyat masalalar yuzaga keladigan turli holatlarga bog’liq ravishda mos qaror qabul qilishni (yechimni) talab etadi. C++ tilida dasturning alohida bo’laklarini bajarilish tartibini boshqarishga imkon beruvchi qurilmalarning

yeterlicha katta majmuasiga ega. Masalan, dastur bajarilishining birorta qadamida qandaydir shartni tekshirish natijasiga ko‘ra dasturning u yoki bu bo‘lagiga boshqaruvni uzatish mumkin (tarmoqlanuvchi algoritm). Tarmoqlanishni amalga oshirish uchun tarmoqlanuvchi operatorlardan foydalaniladi. Ular quyidagilar:

**To‘liqsiz tarmoqlanish if operatori.** if operatori qandaydir shartni rostlikka tekshirish natijasiga ko‘ra dasturda tarmoqlanishni amalga oshiradi:

*if (<tekshiriladigan shart>)<operator>1;*

Bu yerda <tekshiriladigan shart> har qanday ifoda bo‘lishi mumkin, odatda u taqqoslash operatori bo‘ladi.

Agar tekshiriladigan shart rost (true) bo‘lsa, <operator>1 bajariladi, aks holda (false) dastur keyingi operatorlarni bajarishga o‘tadi.

C++ tilida bir nechta amallarni blok (guruuh) larga birlashtirish mumkin. Blok ‘{‘ va ‘}‘ belgi oralig‘iga olingan amallar ketma-ketligi bo‘lib, u kompilyator tomonidan yaxlit bir operator deb qabul qilinadi.

**To‘liq tarmoqlanish.** if – else operatori:

Shart operatorining if – else ko‘rinishi quyidagicha:

*if (<shart-ifoda>)<operator> 1; else<operator> 2;*

Bu yerda <shart-ifoda> rost (true) bo‘lsa, <operator> 1 bajariladi, aks holda <operator> 2 bajariladi. if – else shart operatori mazmuniga ko‘ra algoritmnинг tarmoqlanuvchi blokini ifodalaydi: <shart-ifoda> – shart bloki (romb) va <operator> 1 blokning “ha” shoxiga, <operator> 2 esa blokning “yo‘q” shoxiga mos keluvchi amallar bloklari deb qarash mumkin.

**Tanlash operatori. switch operatori.** Shart operatorining yana bir ko‘rinishi switch tarmoqlanish operatori bo‘lib, uning sintaksisi quyidagicha:

```
switch (<ifoda>)
{ case <konstanta ifoda> :
<operatorlar guruhi>;
break;
case <konstanta ifoda> :
```

```
<operatorlar guruhi>;
```

```
break;
```

```
...
```

```
default :
```

```
<operatorlar guruhi>; }
```

Bu operator quyidagicha ishlaydi: birinchi navbatda `<ifoda>` qiymati hisoblanadi, keyin bu qiymat **case** kalit so‘zi bilan ajratilgan `<konstanta ifoda>` bilan solishtiriladi. Agar ular ustma-ust tushsa, ‘:: belgisidan keyingi **break** kalit so‘zigacha bo‘lgan `<operatorlar guruhi>` bajariladi va boshqaruv tarmoqlanuvchi operatordan keyingi operatorga o‘tadi. Agar `<ifoda>` birorta ham `<konstanta ifoda>` ifoda bilan mos kelmasa, operatorning **default** kalit so‘zidan keyingi operatorlar guruhi bajariladi.

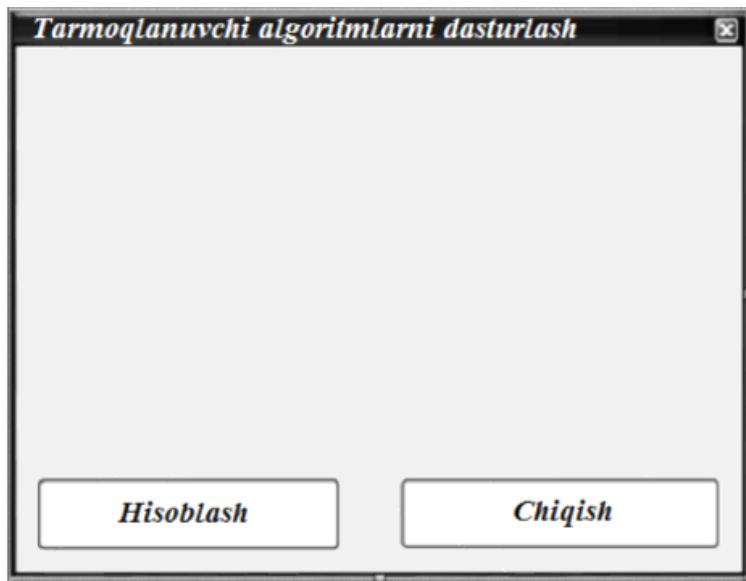
Yuqorida keltirilgan ma’lumotlarni qo‘llanilishini quyidagi misolda ko‘rib chiqamiz

**Misol.** Quyidagi funksiya qiymatini hisoblash uchun windows ilovasi yaratish talab etilgan bo‘lsin:

$$y = \begin{cases} \sin(x^2 + ax), & \text{agar } x \leq 0 \text{ bo'lsa} \\ 1 - \frac{1 + \sqrt{(x^2 + ax)}}{e^{\sin(x)}(1 + x)}, & \text{agar } 0 < x \leq a \text{ bo'lsa} \\ \frac{\cos(x^2 - a^2)}{\sqrt{1 - \sin(a - x)}} - \frac{1 - \sin(a - x)}{e^{\sin(x)}}, & \text{agar } x > a \text{ bo'lsa} \end{cases}$$

1. Visual Studio tizimini ishga tushuramiz
2. **Создать Проект** oynasida Visual C++ bo‘limini ochamiz, **CLR** bandiga o‘tib markaziy paneldan **Windows Form Applicationni** tanlaymiz.
3. Keyin muharrirning **Имя** maydoniga loyiha nomini kiritamiz, masalan **Visual\_funk1**. **Расположение** maydonida siz loyihaning joylashuv yo‘lini belgilashingiz yoki **Обзор** tugmasi yordamida loyihaning joylashuv yo‘lini tanlashingiz mumkin. (masalan, E:\VC\С++\Amaliy2).

4. Forma uchun quyidagi ma'lumotlarni kiritish orqali **Text** xususiyatining qiymatini o'zgartiramiz masalan: "*Tarmoqlanuvchi algoritmlarni dasturlash*"
5. Formaning **FormBorderStyle** xossasini (oyna ramkasi uslubi) **FixedToolWindow**ga o'zgartiramiz. Bu qiymat oynani dialog oynasi sifatida belgilaydi, uning hajmini dasturni ishga tushirish bosqichida (ilova ishlayotgan vaqtida) o'zgartirib bo'lmaydi.
6. Formaga **Button1** komponentini (buyruq tugmasi) joylashtiramiz va unga "*Hisoblash*" deb yozamiz (ya'ni bu matnni **Text** xossasiga yozamiz).
7. Xuddi shunday, **Button2** tugmачасини yaratamiz va unga "*Chiqish*" deb yozamiz. **Shrift** xossasidan foydalanib, tugmachadagi shrift hajmini va matn yozishni o'zgartirishingiz mumkin. **ForeColor** xususiyatidan foydalanib, siz shrift rangini o'zgartirishingiz mumkin (5.2.1-rasmga qarang).



5.2.1-rasm – Dasturni yaratish bosqichidagi forma oynasi

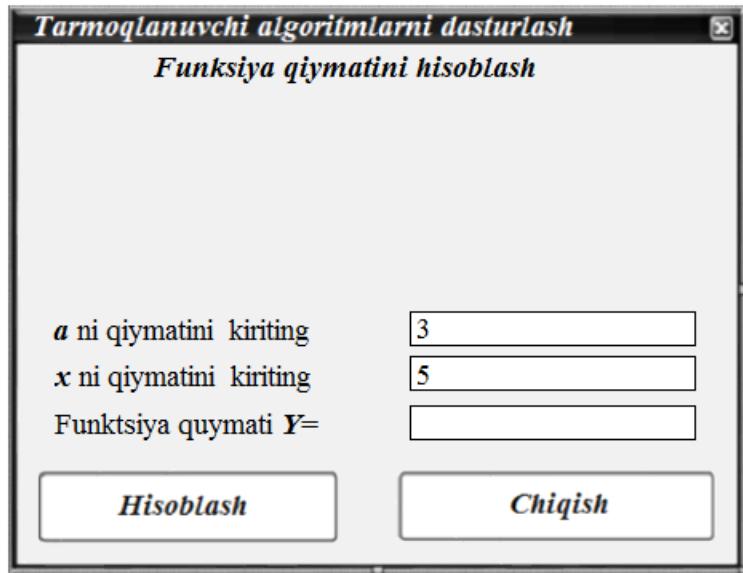
8. **Label** komponentasini formaning eng yuqori qismiga joylashtiramiz. "**Text**" xususiyatiga "**Funksiya qiymatini hisoblash**" matnnini kiritamiz (bu yerda ham shrift parametrlarini o'zgartirishimiz mumkin).

Hisob-kitoblarni amalga oshirish uchun dastlabki ma'lumotlarni kiritish kerak: Ya'ni *a* parametrining qiymatini va *x* o'zgaruvchining qiymatini kiritish kerak bo'ladi. Ushbu ma'lumotlar **TextBox** komponenti yordamida klaviaturadan kiritiladi.

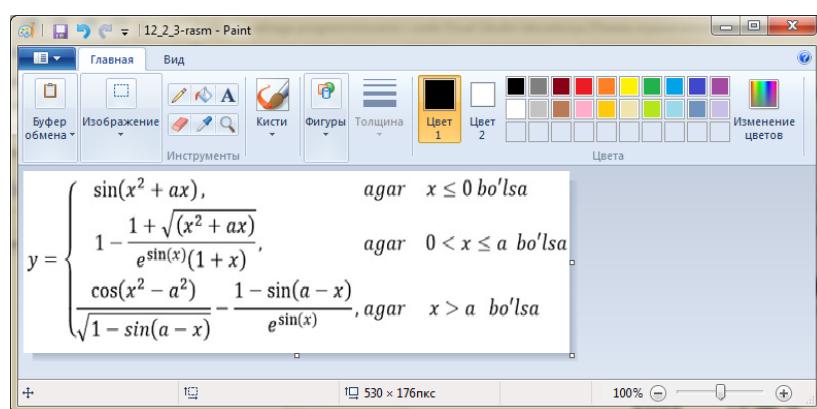
Kirish natijasi joylashtirilgan ushbu komponentning xossasi odatda matn tipida bo'ladi, ya'ni har qanday belgilarni kiritish imkonini beradi. Ma'lumotlarni kiritish jarayonida foydalanuvchi xato (yaroqsiz – raqamli bo'limgan) belgilarni kiritishi mumkin. Bunday holatga yo'l qo'yib bo'lmaydi, chunki agar biz noto'g'ri belgilar kiritsak, noto'g'ri ma'lumotlarni haqiqiy turga o'tkazishda dastur ishdan chiqadi, ya'ni dasturda xatolik yuz beradi (dastur ishlamaydi). Shuning uchun foydalanuvchi ushbu maydonlarga ma'lumotlarni diqqat bilan kiritishi kerak. Buning uchun uchta **TextBox** elementini tugmachalarning ustiga qo'yamiz (bitta **TextBox** elementini joylashtirishib olib, qolgan ikkitasini birinchisidan nusxalashimiz mumkin). Birinchi ikkitasi mos ravishda *a* parametrining qiymatini va *x* o'zgaruvchining qiymatini kiritishga mo'ljallangan. Ushbu komponentlarning **Text** xususiyatlariga bir nechta standart qiymatlarni kiritishimiz mumkin. Ushbu qiymatlar ilova ishga tushirilganda ko'rsatiladi. Illova ishga tushganda, kiritilgan qiymatlarni boshqalari bilan almashtirilishi mumkin. Funktsiyani hisoblash natijasini ko'rsatish uchun **TextBox3** komponentidan foydalilanadi, shuning uchun unga foydalanuvchi ma'lumot kiritishini taqilash kerak bo'ladi. Buning uchun faqat **ReadOnly** xossasini *true* xolatiga o'rnatib quyamiz, bu esa foydalanuvchi komponentga biron bir ma'lumot kiritishiga yo'l qo'yaydi.

9. Formadagi birinchi **TextBox1** elementi ro'parasiga **Label2** elementini qo'yamiz va ushbu elementning **Text** xossasiga “*a* ni qiymatini kriting” matnini kiritamiz, **TextBox2** elementi ro'parasiga **Label3** elementini quyamiz va **Text** xossasiga “*x* ni qiymatini kriting” matnini kiritamiz. Oxirgi **TextBox3** elementi ro'parasiga esa **Label4** elementini joylashtiramiz

va **Text** xususiyatiga “Funktsiya quymati  $Y=$ ” matnini kiritamiz. Natijada quyidagicha forma ko‘rinishiga ega bo‘lamiz. (5.2.2-rasmga qarang).



- 5.2.2-rasm – Loyihalashtirilgan formaning taxminiy ko‘rinishi
10. **Microsoft Word** matn protsessorini ishga tushiramiz va undagi **formula** muharriridan foydalanib, word sahifasiga berilgan vazifaga muvofiq matematik formulani kiritamiz. Yozilgan formulani belgilab olib uni vaqtinchalik xotira (буфер обмена)ga nusxasini olamiz.
11. Biror grafik muharririni, masalan oddiy **Paint** grafik muharririni ishga tushiramiz (Пуск □ Программы □ Стандартные □ Paint) va xotiraga nusxalangan (buferdan) terilgan formulani grafik muharriga joylashtiramiz (5.2.3-rasm).



5.2.3-rasm – Paint grafik muharriri oynasi

12. Yaratilgan grafik faylni dastur loyihasi (*Amaliy2*) bilan bir papkada (ingliz tilida nom berib) saqlaymiz.
13. Formaga **PictureBox1** komponentini joylashtiramiz va **Image** xususiyatidan foydalanib, unga yaratilgan grafikli fayl funksiyani aks ettiruvchi rasmni yuklaymiz. Rasm standart dialog oynasi yordamida yuklanadi. **SizeMode** xossasidan foydalanib, rasmimizni **PictureBox1** blokiga **StretchImage** bandini tanlab joylashtiramiz.
14. Yuqorida amalga oshirilgan operatsiyalar natijasida biz taxminan 5.2.4-rasmda ko‘rsatilganidek bir xil formaga ega bo‘lishimiz kerak.

**Tarmoqlanuvchi algoritmlarni dasturlash**

**Funksiya qiymatini hisoblash**

$$y = \begin{cases} \sin(x^2 + ax), & \text{agar } x \leq 0 \text{ bo'lsa} \\ 1 - \frac{1 + \sqrt{(x^2 + ax)}}{e^{\sin(x)}(1 + x)}, & \text{agar } 0 < x \leq a \text{ bo'lsa} \\ \frac{\cos(x^2 - a^2)}{\sqrt{1 - \sin(a - x)}} - \frac{1 - \sin(a - x)}{e^{\sin(x)}}, & \text{agar } x > a \text{ bo'lsa} \end{cases}$$

**a** ni qiymatini kriting   
**x** ni qiymatini kriting   
 Funktsiya quymati **Y=**

**Hisoblash** **Chiqish**

5.2.4-rasm – Ilova formasining taxminiy ko‘rinishi

15. **Button1** tugmasi uchun “**Hisoblash**” deb nomlangan “**Click**” hodisasini yarataylik. Buning uchun **Button1** komponentida sichqonchaning chap tugmasi bilan ikki marta bosamiz. Natijada quyidagi dastur shablonini olamiz:

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

16. Ikki fugurali qavs orasiga quyidagi dastur kodining fragmentini kiritamiz:

```
double y, x, a;
//x va a uchun ma'lumotlar kiritilganligini tekshirish
```

```

if((textBox1->Text!="")&&(textBox2->Text!=""))
{
// x va a larni matndan xaqiqiy tipli soniga aylantirish
a=Convert::ToDouble(textBox1->Text);
x=Convert::ToDouble(textBox2->Text);
if(x<=0)y=sin(x*x+a*x);
else
if(x>0&&x<=a){y=1-1+sqrt(x*x+a*x))/(exp(sin(x))*(1+x));}
else
{y=(cos(x*x-a*a)/(sqrt(1-sin(a-x)))-(1-sin(a-x))///(exp(sin(x)));}
// TextBox3 komponentida matnga aylantirilgan Y qiymatini chiqarish
textBox3->Text=Convert::ToString (y);}
// Komponentlarga hech qanday ma'lumot kiritilmagan bo'lsa, xabar oynasini
ko'rsatadi

```

### **TextBox1** va **TextBox2**

```

{MessageBox::Show( "Marhamat, a va x qiymatlarini kriting", "Kiritilgan
ma'lumotlar xato", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation); }

```

17. "**Chiqish**" yozushi (yorlig'i-tugmasi) **Button2** uchun Click hodisasini yarataylik. Buning uchun **Button2** komponentida sichqonchaning chap tugmasi bilan ikki marta bosamiz. Natijada quyidagi dastur shablonini olamiz:

```

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
}

```

18. Ikki fugurali qavs orasiga quyidagi dastur kodining fragmentini kiritamiz:

```

Application ::Exit();

```

19. Dasturning yuqori qismida bir marta #pragma so'zidan keyin matematik funktsiyalar bilan ishlash uchun kutubxonani bog'laydigan qator qo'shamiz.

```

#include <math.h>;

```

Xuddi shuningdek satrli funksiyalar bilan ishlaydigan kutubxonani bog‘laydigan qator qo‘shamiz.

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

20. Ilova dasturlarini yaratishda ishlab chiquvchilar (dasturchilar) interfeysi iloji boricha tushunishni osonlashtiradigan va dasturning o‘zini foydalanuvchi harakatlariga imkon qadar "do‘stona" qilishga harakat qilishadi. Yaratilayotgan ilovamizda foydalanuvchi dastlabki ma’lumotlarni (*a* va *x* raqamli qiymatlari) kiritishda xatoga yo‘l qo‘yishi va raqam o‘rniga matn kiritib quyishi mumkin. **TextBox1** va **TextBox2** komponentlari uchun **LostFocus** hodisasini boshqarish protsedurasini yaratish kerak (bu hodisa kursor joriy bir qatorli matn maydonidan boshqa ob’ektga o‘tganda sodir bo‘ladi). Yaratilayotgan formada **TextBox1** obyektini tanlaymiz va **Xususiyatlar** oynasiga  **Voqealar** yorlig‘iga o‘tamiz, so‘ng bizni qiziqtirayotgan **Leave** hodisasini topamiz va kichik dastur shablonini yaratish uchun hodisa nomining o‘ng tomonidagi (bo‘sh oq maydonda) sichqonchaning chap tugmachasini ikki marta bosamiz va dastur fragmentini kiritamiz:

```
private: System::Void textBox1_Leave(System::Object^ sender,  
System::EventArgs^ e)  
{  
}
```

21. Ikki fugurali qavs orasiga quyidagi dastur kodining fragmentini kiritamiz:

```
int l, t, k; bool a=true; String ^str; str=textBox1->Text;  
l=str->Length;  
// biz ishlayotgan belgi (simvol)ning indeksi  
t=0;  
// satrdagi vergullar soni (variantda quyidagicha yozilsa 0,2384,1254,1251 - bu  
raqam emas)
```

```

k=0;
// agar raqam manfiy bo'lsa, indeks satrni siljitalmiz
if(str[t]== '-' ) t++;
// raqam vergul bilan boshlanmaydi
if(str[t]== ',') a=false; while(t<l)
{ if(str[t]== ',')
// agar vergul oxirgi belgi bo'lsa yoki vergul allaqachon topilgan bo'lsa
{ if(t==l-1 // k>0) a=false;k++;
}
// agar t-belgi '0' dan '9' gacha bo'lgan oraliqda yotmasa
else if(str[t]<'0'||str[t]>'9') a=false;t++;
}
if(a==false)
{ MessageBox::Show("a parametri raqam emas ","Kiritilgan ma'lumotlar
xato",MessageBoxButtons: :OK, MessageBoxIcon:
:Exclamation);
}

```

// fokusni (diqqatni) matn maydoniga qaytarish  
*this->textBox1->Focus();*  
}

22. Xuddi shunday, **TextBox2** maydonidagi raqamli ma'lumotlar kiritilishiga muvofiqligini tekshiradigan protsedura (qism dastur) tuzamiz. Biz yana **Leave** hodisasini quyidagi kod bilan boshqaradigan qismdastur yaratamiz:

```

int l, t, k; bool a=true; String ^str;
str=textBox2->Text;
l=str->Length;
// biz ishlayotgan belgi (simvol)ning indeksi
t=0;
// satrdagi vergullar soni (variantda quyidagicha yozilsa 0,2384,1254,1251 - bu
raqam emas)

```

```

k=0;
// agar raqam manfiy bo'lsa, indeks satrni siljitalimiz
if(str[t]=='-') t++;
// raqam vergul bilan boshlanmaydi
if(str[t]==',') a=false; while(t<l)
{ if(str[t]==',')
// agar vergul oxirgi belgi bo'lsa yoki vergul allaqachon topilgan bo'lsa
{ if(t==l-1 // k>0) a=false;k++;
}
// agar t-belgi '0' dan '9' gacha bo'lgan oraliqda yotmasa
else if(str[t]<'0' || str[t]>'9') a=false;t++;
}
if(a==false)
{ MessageBox::Show("x parametri raqam emas", "Kiritilgan ma'lumotlar
xato", MessageBoxButtons::OK, MessageBoxIcon:
:Exclamation);
// fokusni (diqqatni) matn maydoniga qaytarish
this->textBox2->Focus();
}

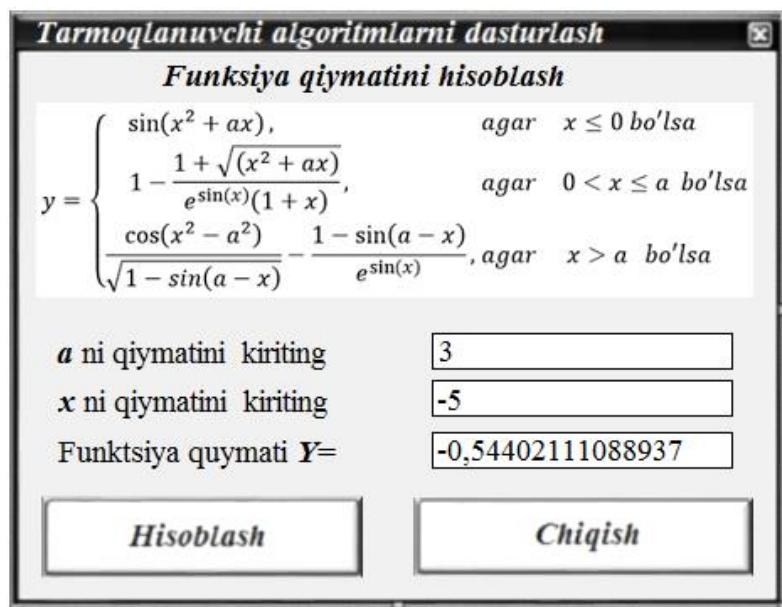
```

23. Dasturning yana bir "korekt emasligi – noto'g'riligi" bu foydalanuvchi *x* ning bitta qiymatiga javob olgandan so'ng, boshqa javob olish uchun argument qiymatini o'zgartirganda sodir bo'ladi. Foydalanuvchi "**Hisoblash**" tugmasini bosmasdan oldin, forma yangi argument qiymatini va eski (oldingi natija) funktsiya qiymatini ko'rsatadi (bu esa "xato"). Kursorni **TextBox1** yoki **TextBox2** matn maydonlariga joylashtirishda **TextBox3** matn maydonini eski yozuvdan tozalash kerak. Bu **GetFocus** (Enter) hodisasini boshqarish uchun qismdastur (podprogramma) yaratish orqali amalga oshiriladi. Yaratilayotgan formada **TextBox1** obyektini tanlaymiz va **Xususiyatlar** oynasigagi **Xodisalar** bandiga o'tamiz. Bizni

qiziqtirgan **Enter** hodisasini topamiz va qism dastur shablonini yaratish uchun hodisa nomining o‘ng tomonidagi (bo‘sh oq maydonda) sichqonchaning chap tugmachasini ikki marta bosamiz:

```
private:Void textBox1_Enter(System::Object^ sender,
System::EventArgs^e)
{
}
```

24. Ikki fugurali qavs orasiga quyidagi dastur kodining fragmentini kiritamiz:  
textBox3->Text = "";
25. Xuddi shunday, formadagi **TextBox3** matn maydonini tozalash uchun **TextBox2** matn maydoni *focus (Enter)*ga qism dasturini yaratamiz, unga ham xuddi shu yuqoridagi buyruqni joylashtiramiz.
26. Shunday qilib, shartli *if* operatori (buyrug‘i) yordamida funktsiya qiymatini hisoblash uchun to‘llaqonli dasturga ega bo‘lamiz.
27. Yaratilgan dasturni ishga tushuramiz (F5 funksional klavishini bosib) va quyidagi ilovamizning oyna ko‘rinishini olamiz (5.2.5-rasm).



5.2.5-rasm – Ilova formasining ishchi ko‘rinishi

### 3. Massivlar bilan ishlash komponentalari.

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

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

Bir o‘lchovli massivda uning har bir elementi o‘zining joylashgan o‘rin nomeri bilan aniqlanadi va nomeri qavs ichida indeks bilan yoziladi.

Agar struktura bir xil kattalikdagi tiplardan tuzilgan bo‘lsa, uning nomini **massiv** deb aytamiz. Massivlar dasturlashda eng ko‘p qo‘llaniladigan ma‘lumotlar tiplaridir. Bundan tashqari strukturalar bir necha farqli tipdagi o‘zgaruvchilardan tashkil topgan bo‘lishi mumkin, buni **klass** deymiz.

**Massiv** deb – bir nom bilan ataluvchi, bir turga mansub bo‘lgan, tartiblashgan kattaliklar ketma-ketligiga aytildi.

Xotirada ketma-ket (regulyar) joylashgan bir xil turdagilari qiyatlarga **massiv** deyiladi.

Agar massiv elementiga bir indeks orqali murojaat qilish mumkin bo‘lsa u holda bunday massiv **bir o‘lchovli massiv** deyiladi.

{2, 5, -9, 12, 0, 7, 36, 4} – butun sonlar massivi

{3.14, 8.76, -5.11, 0.25, -6.47} – haqiqiy sonlar massivi

{‘t’, ‘A’, ‘#’, ‘f’, ‘\$’, ‘\*’} – belgilar massivi

{true, false, true, true, false} – mantiqiy tipdagi massiv

{"toyota", "lexus", "hyundai", "porsche"} – satrlar massivi

`int a[10];`



**int - massiv tipi**



**a - massiv nomi**



**10 - uzunligi**

*elementlar*

0	1	2	3	4	5	6	7	8	9

*indeks*

Bir o‘lchovli massivdan farqli, ikki o‘lchovli massiv e’lon qilishda massivning satrlari va ustunlari soni quyidagicha ko‘rsatiladi:

`int a[3][4];`



**int - matritsa tipi**



**a - nomi**



**3 - satr uzunligi**



**4 - ustun uzunligi**

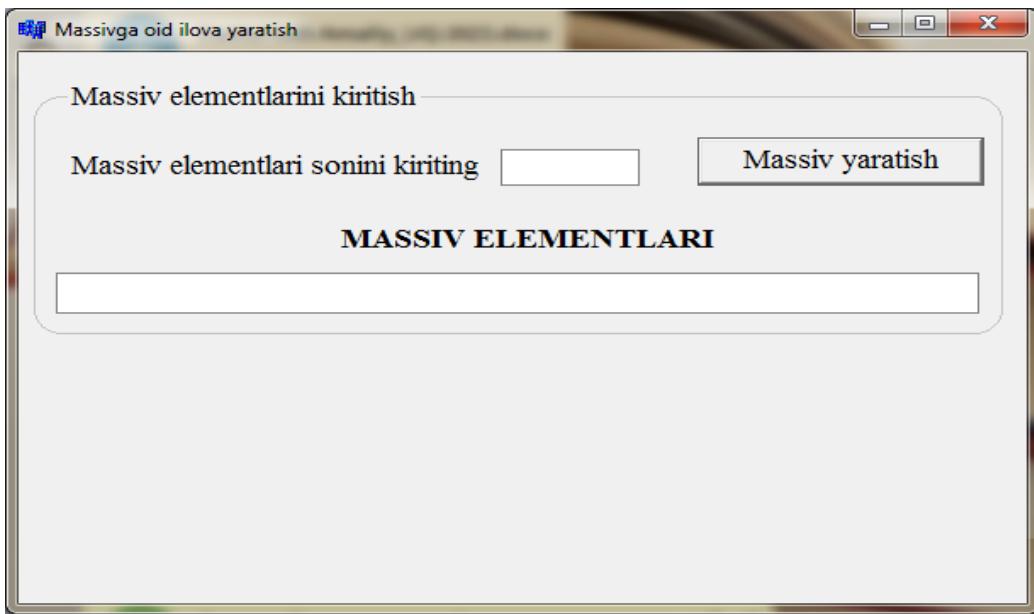
Yuqorida berilgan ma’lumotlardan foydalangan xolda misol keltirib o‘tamiz.

### **Misol.**

Windows ilovasini yaratish, Bu ilova foydalanuvchilarga chiziqli massiv o‘lchovini belgilashni taklif qiladigan, massivni avtomatik ravishda –10 dan 10 gacha bo‘lgan oraliqdagi tasodifiy butun sonlar bilan to‘ldiradigan va massiv elementlarini ko‘rsatadigan, so‘ngra foydalanuvchining xohishiga ko‘ra massivning juft elementlari hamda massivning musbat elementlari soni va ularning yig‘indisini hisoblaydigan ilova bo‘lsin.

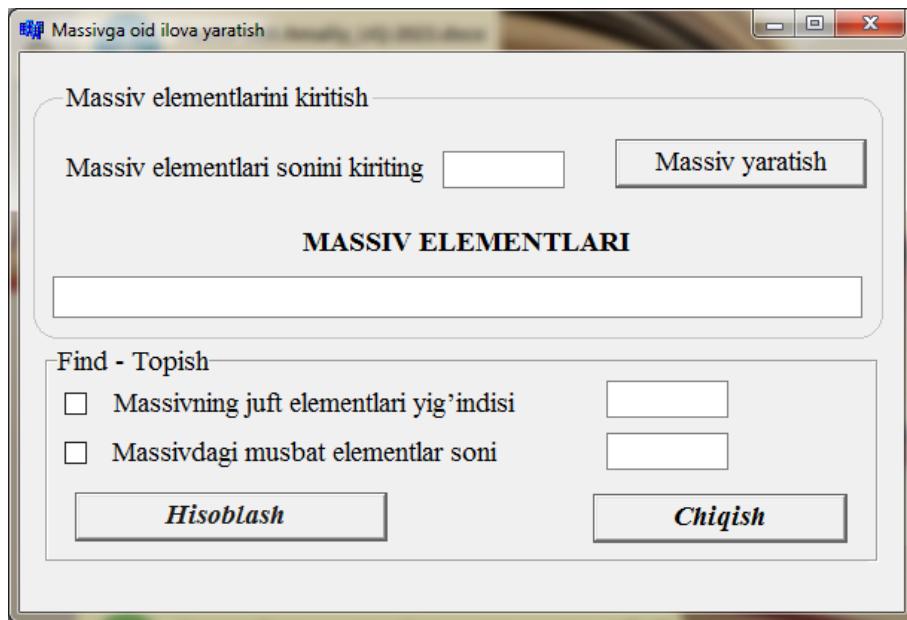
1. Visual Studio tizimini ishga tushuramiz.
2. **Создать Проект** oynasida Visual C++ bo‘limini ochamiz, **CLR** bandiga o‘tib markaziy paneldan **Windows Form Applicationni** tanlaymiz.

3. Keyin muharrirning **Имя** maydoniga loyiha nomini kiritamiz, masalan **Visual\_amaliy3. Расположение** maydonida siz loyihaning joylashuv yo‘lini belgilashingiz yoki **Обзор** tugmasi yordamida loyihaning joylashuv yo‘lini tanlashingiz mumkin. (masalan, E:\\VC\\C++\\Amaliy3).
4. Forma uchun quyidagi ma’lumotlarni kiritish orqali **Text** xususiyatining qiymatini o‘zgartiramiz, masalan: "**Massivga oid ilova yaratish**"
5. Formaning **FormBorderStyle** xossasini (oyna ramkasi uslubi) **FixedToolWindow**ga o‘zgartiramiz. Bu qiymat oynani dialog oynasi sifatida belgilaydi, uning hajmini dasturni ishga tushirish bosqichida (ilova ishlayotgan vaqtida) o‘zgartirib bo‘lmaydi.
6. **GroupBox1** konteynerini oynaning yuqori qismiga joylashtiramiz va **Text** xususiyatiga “**Massiv elementlarini kiritish**” matnini kiritamiz.
7. Shu **GroupBox1** konteneriga yana beshta komponenta joylashtiramiz: **Label1**, **Label2**, **TextBox1**, **TextBox2** u **Button1**. **Label1** birinchi komponenti uchun **Text** xususiyatiga “**Massiv elementlari sonini kirititing**” matnini kiritamiz. **Label2** ikkinchi komponenti uchun “**Massiv elementlari**” matnnini **Text** xususiyatiga kiritamiz. **TextBox2** komponenti uchun **ReadOnly** xossasini rostga o‘rnatamiz, ya’ni **true** ni tanlaymiz, bu foydalanuvchi komponentga har qanday ma’lumotlarni kiritishni taqiqlaydi. **TextBox1** komponenti uchun ma’lumotlar **TextBox** komponenti yordamida klaviaturadan kiritiladi. Ushbu komponentning **Text** xususiyatiga ixtiyoriy matni kiritish mumkin. Ushbu qiymat dastur ishga tushirilganda ko‘rsatiladi. Ilova ishga tushganda uni boshqasi bilan almashtirish mumkin.
8. Formaga **Button1** komponentini quyamiz va **Text** xossasiga “**Massiv yaratish**” matnnini kiritamiz. Ushbu tugmani bosganingizda tasodifiy sonlar generatori yordamida massiv avtomatik ravishda yaratiladi, uning elementlari belgilangan [-10; 10] diapazonda **Text-Box2** komponentida ko‘rsatiladi. Natijada quyidagi formani (shaklni) xosil qilamiz (5.3.1-rasm).



5.3.1-rasm – Forma ilovasining ko‘rinishi

9. **GroupBox2** konteynerini oynaning pastki qismiga joylashtiramiz va "**Text**" xususiyatiga "**Find - Topish**" matnini kiritamiz.
10. Shu **GroupBox2** konteyneriga oltita komponentni joylashtiramiz: **TextBox3**, **TextBox4**, **checkBox1**, **checkBox2**, **Button2** va **Button3**.
11. **CheckBox1** komponentini belgilaymiz va **Text** xususiyatiga “**Massivning juft elementlari yig‘indisi**” matnini kiritamiz. Xuddi shunday, **checkBox2** komponenti uchun **Text** xususiyatiga “**Massivdagi musbat elementlari soni**” matnini kiritamiz.
12. **Button2** komponentini tanlaymiz va **Text** xususiyatiga “**Hisoblash**” matnini kiritamiz. Xuddi shunday **Button3** komponentining **Text** xususiyatiga “**Chiqish**” matnini kiritamiz.
13. **TextBox3** va **TextBox4** komponentlari topshiriqlarni qidirish natijalarini ko‘rsatish uchun ishlataladi, shu sababli foydalanuvchi ularga ma’lumotlarni kiritishiga yo‘l qo‘ymaslik kerak. Buning uchun **ReadOnly** (faqat o‘qish) xossalida **true** (rost) o‘rnataladi, bu foydalanuvchi komponentga har qanday ma’lumotlarni kiritishni taqiqlaydi.
14. Amalga oshirilgan operatsiyalar natijasida biz quyida 5.3.2-rasmda ko‘rsatilganidek shaklga ega bo‘lamiz.



5.3.2-rasm – Dastur forma oynasining ko‘rinishi

15. Dastur kodiga o‘tamiz va

*#pragma once* qatordan keyin

matematik funktsiyalardan foydalanish uchun kutubxonalarni, o‘zgaruvchilar tavsifi, massiv o‘lchamini kiritishimiz kerak, buning uchun quyidagi qatorlarni kiritamiz:

```
#include<cmath>
```

```
#include <stdlib.h>
```

```
#define m 15
```

```
int i,n;
```

```
int A[m];
```

16. "Massiv yaratish" matni yozilgan **Button1** tugmasi uchun **Click** hodisasini yaratamiz. Buning uchun **Button1** komponentida sichqonchaning chap tugmasini ikki marta tez bosamiz. Qismdastur (Subprogramma) kodiga quyidagi o‘zgartirishlar kiritamiz:

```
//textBox1 bo‘sh emasligini tekshirish
```

```
if(textBox1->Text!="")
```

```
{n=Convert::ToInt32(textBox1->Text); }else
```

```

{MessageBox::Show( "Marhamat ma'lumotlarni kriting", "Xotolik:
ma'lumot noto'g'ri kiritildi", MessageBoxButtons::OK, MessageBoxIcon::Exclamation);}

//textBox2 komponentasini tozalash
{textBox2->Text="";

//Massiv yaratish va textBox2 komponentini [-10;10] diapazondagi tasodifiy
sonlar bilan to'ldirish jarayoni
for (i = 0; i < n; ++i)
{
    A[i] = rand() % 21-10;
    this->textBox2->AppendText(A[i] + " ");
}
}

```

17. "Hisoblash" deb nomlangan **Button2** tugmasi uchun **Click** hodisasini yaratamiz. Buni amalga oshirish uchun **Button2** komponentida sichqonchaning chap tugmasi bilan ikki marta bosamiz va natijada olingan kichik dastur shabloniga quyidagi hisob-kitoblar kodini kiritamiz:

```

//kol va sum o'zgaruvchilarini nolga tenglashtirish
int kol=0; int sum=0;
for (i=0; i<n; ++i)
{
//Shartni tekshirish va yig'indini topish
if (A[i]%2==0){sum=sum+A[i];}
//Shartni tekshirish, musbat elementlar sonini topish
if (A[i]>0){kol=kol+1;}
}
//Agar checkBox komponentlari aktiv bo'lsa, topshiriq natijalarini textBox
komponentlarida ko'rsatish
if (checkBox1->Checked==true){this->textBox3->Text= Convert::ToString

```

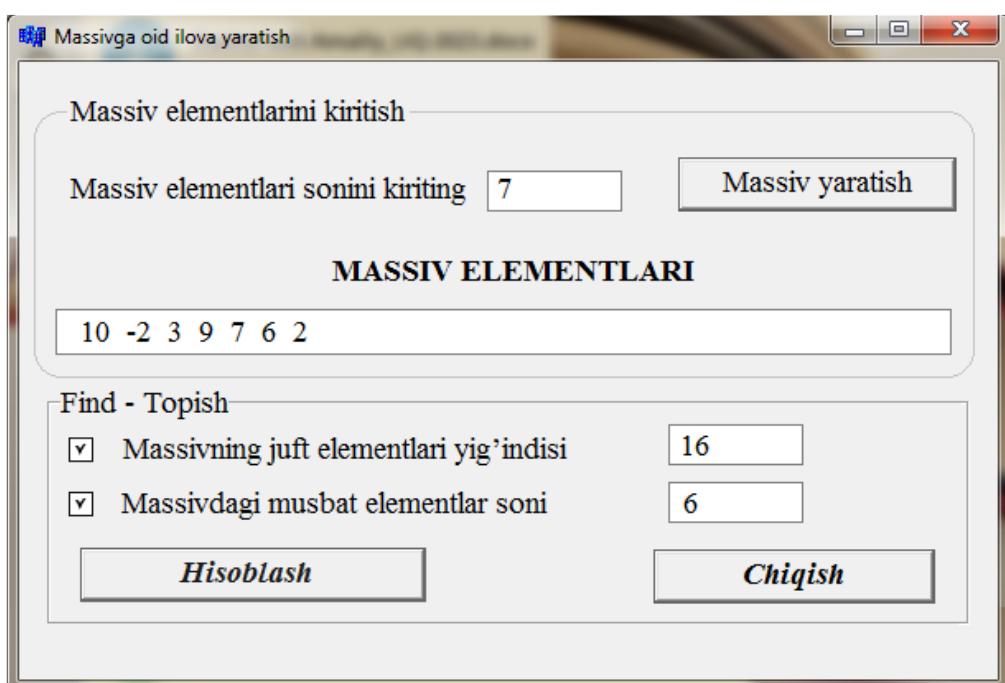
```

        (sum);}

if (checkBox2->Checked==true){this->textBox4->Text= Convert::ToString
(kol);}

```

18. **Button3** tugmasi uchun “**Chiqish**” yorliqli **Click** hodisasini yaratamiz. Buning uchun **Button3** komponentida sichqonchaning chap tugmasini ikki marta bosamiz va hosil bo‘lgan kichik dastur shabloniga dasturdan chiqish kodini kiritamiz (oldingi ma’ruzalarda buni bajarganmiz).
19. F5 funksiyanal tugmachasini bosib dasturni ishga tushiramiz. Natijada quyidagi forma oynasi ko‘rinishini olamiz. 5.3.3 – rasm.



5.3.3-rasm – natijaviy ilova oynasining ko‘rinishi

### Nazorat savollari

1. Windows Forms loyihasi qanday yaratiladi va qanday saqlanadi?
2. Ikkinchi Forma (shakl) qanday yaratiladi va uni birinchi formadan qanday chaqirish kerak?
3. Form komponenti va uning asosiy xossalari.
4. Memo komponenti va uning asosiy xossalari.
5. Label komponentasi va uning asosiy xossalari.

6. TextBox komponentasi va uning asosiy xossalari.
7. Button komponenti va uning asosiy xossalari.
8. ListView komponenti va uning asosiy xossalari.
9. RadioButton komponenti va uning asosiy xossalari.
10. CheckBox komponentasi va uning asosiy xossalari.
11. PictureBox komponentasi va uning asosiy xossalari.
12. If operatorining to‘liq va qisqa shakli qanday?
13. If shartli operatorda bir nechta amallar qanday tashkil qilinadi?
14. Formada matematik funktsiyaning rasmini qanday qilib ko‘rsatish mumkin?
15. C++ dasturlash tilida matematik funktsiyalar qanday yoziladi?
16. C++ tilida bir o‘lchovli raqamli massiv tushunchasi?
17. C++ da raqamli massivlarni indekslash qanday tashkil etilgan?
18. Massiv elementlarini qo‘lda kiritish uchun qanday vizual komponentlardan foydalanish mumkin?

### **5.3. § GUI MUHITIDA GRAFIK IMKONIYATLAR. TO‘G‘RI CHIZIQ VA TURLI XIL GEOMETRIK FIGURALARNI CHIZISH KOMPONENTALARI.**

**Reja:**

- 1. GUI muhitida grafik imkoniyatlar.**
- 2. To‘g‘ri chiziq va ellips chizish komponentalari.**
- 3. Turli xil geometrik figuralarni chizish komponentalari.**
  - 1. GUI muhitida grafik imkoniyatlar.**

**GUI- graphical user interface - Графический интерфейс пользователя**

**- Grafikli foydalanuvchi interfeysi.**

Visual C++ **SDI (Serial Digital Interface - Последовательный цифровой интерфейс - Сериали рабочий интерфейс)** ilovasini taqdim etadi, uning asosiy oynasida sozlanadigan uskunalar paneli (yuqorida) va komponentlar palitrasи (o‘ngda) mavjud. Bundan tashqari, standart bo‘yicha, Visual C++ dasturini ishga tushirganingizda, Object Inspector oynasi (pastki chap) va Yangi ilova shakli (yuqori chap) paydo bo‘ladi. Illova shakli oynasi ostida kod muharriri oynasi joylashgan.

Visual C++ o‘zi grafik tuzilmali muhit bo‘lib hisoblanadi. Grafik deganda ixtiyoriy narsani kompyuterda piksellarda hosil qilish tushuniladi. Pikselda ikkita argumentlari bor, ya‘ni koorlinatalari  $A(x,u)$  ko‘rinishida. Har qanday IDE muhitlarda grafika bilash ishlash imkonи bor. Ular turlicha nomlanishi mumkin. Ammo, ularning ruchkasi (qalami) va mo‘yqalami bo‘ladi. GUI ko‘rinishda ishlash to‘liq grafika bilan bog‘liq bo‘lib, texnikaning grafik rejimda ishlashini ta‘minlaydi. Hozirda deyarli barcha foydalanuvchilar grafik imkoniyatlaridan foydalanib ishlaydilar.

Grafik imkoniyatga ega bo‘lgan tizimlarda asosan, nuqta, chiziq, to‘rt burchak, aylana, ko‘p burchak kabi shakllarni qamrab oladi. GUI asosidagi barcha elementlarga diqqat bilan qarasangiz shu grafik tuzilmalardan iborat bo‘ladi.

*Visual C++ning imkoniyatlarini ko‘rish uchun* Graphics sinfiga murojaat qilamiz. Bu sinfning nomlar fazosi System.Drawing va kutubxonasi System.Drawing. Common.dll bo‘lib hisoblanadi.

Chizish uchun GDI + modulni inkapsulyatsiya qiladi va bu sinfdan merosxo‘r olish mumkin emas.

Bu sinf MarshalByRefObject, IDisposable, System::Drawing::IdeviceContextga asoslangan bo‘lib, Object → MarshalByRefObject → Graphics sinfining merosxo‘ri hisoblanadi.

Grafik ob‘yekt yordamida ko‘p turli shakl va chiziqlar chizish mumkin. Chiziqlar va shakllar chizish uchun maxsus **DrawGraphicalElement** usullarni o‘rganish lozim. Bu usullar **DrawLine**, **DrawArc**, **DrawClosedCurve**, **DrawPolygon** va **DrawRectangle** o‘z ichiga oladi. Chiziqlar va shakllar chizish uchun qalam yordamida va shakllarni to‘ldirish uchun mo‘yqalam yordamida amalga oshiriladi.

Grafika sinfi imkoniyatlarini uning xususiyatlari va usulari orqali ko‘rsatib o‘tamiz.

#### 5.7-jadval. Grafika sinfi xususiyatlari.

Xususiyait nomi	Vazifasi
Clip	Grafikaning chizilgan chegarasini cheklaydigan chegarani oladi yoki o‘rnatadi.
ClipBounds	Grafikaning kesish chegarasini chegaralovchi RectangleF tuzilishini oladi.
CompositingMode	Kompozit tasvirlar chizish holatini oladi yoki o‘rnatadi.
CompositingQuality	Grafikaga chizilgan kompozitsion tasvirlarning ko‘rsatish sifatini o‘rnatadi.
DpiX	Ushbu grafikaning gorizontal o‘lchamini oladi.
DpiY	Ushbu grafikaning vertikal o‘lchamini oladi.

Xususiyait nomi	Vazifasi
InterpolationMode	Ushbu grafikalar bilan bog‘liq interpolyatsiya rejimini oladi yoki o‘rnatadi.
IsClipEmpty	Bu grafika kesish sohasini bo‘sh yoki yo‘qligini ko‘rsatib, bir qiymat oladi.
IsVisibleClipEmpty	Bu grafika aniq kesish sohasini bo‘sh yoki yo‘qligini ko‘rsatib, bir qiymat oladi.
PageScale	Bu grafika uchun sahifa moduli va birlik moduli o‘rtasida chegarasini sozlash.
PageUnit	Bu grafika sahifa koordinatalarini uchun ishlataladigan o‘lchov birligi sozlash.
PixelOffsetMode	Bu grafika ko‘rsatish paytida Piksel ofset qanday ko‘rsatilgan qiymat sozlash.
RenderingOrigin	Bu grafika ko‘rsatish rejimini o‘rnatadi.
SmoothingMode	Grafikalar uchun ko‘rsatish sifatini oladi yoki o‘rnatadi.
TextContrast	Matn ko‘rsatish uchun gamma qiymatini belgilash.
TextRenderingHint	Bu grafika bilan bog‘liq matn uchun ko‘rsatish rejimini o‘rnatadi.
Transform	Grafikalar uchun geometrik o‘zgarishining nusxasini oladi yoki o‘rnatadi.
VisibleClipBounds	Grafikning aniq kesish sohasini tekslash uchun to‘rtburchak oladi.

Grafika sinfi usulari quyidagilardan iborat:

1. **AddMetafileComment(Byte[])** - Rasm metafayliga izoh qo‘shadi. Bunda Byte[] belgili massiv bo‘lib, <System::Byte> ^ data tipida aniqlanadi va masalan, array<Byte>^metaCom={(Byte)'T',(Byte)'e',(Byte)'s',(Byte)'t'}; kabi aniqlanishi mumkin.

2. ***BeginContainer()*** - Grafikaning hozirgi holati bilan grafik konteynerni saqlaydi va yangi grafik konteynerni ochadi va ishlatadi.

***BeginContainer(Rectangle, Rectangle, GraphicsUnit)*** - Grafik joriy holati bilan bir grafik konteyner saqlaydi, belgilangan parametrli o‘zgartirish bilan yangi grafik konteynerdan foydalanadi va ochadi. Bunda Rectangle to‘rt burchak bo‘lib, ***Rectangle(0,0,200,200)*** kabi aniqlanadi. Birinchi to‘rt burchak konteyner uchun shkalani va ikkinchisi konteyner uchun soha o‘zgarishi aniqlaydi.

***GraphicsUnit*** – konteyner uchun o‘lchov birligini aniqlash uchun ishlatiladi.

***BeginContainer(RectangleF, RectangleF, GraphicsUnit)*** - grafik joriy holati bilan bir grafik konteyner saqlaydi, belgilangan sohali o‘zgartirish bilan yangi grafik konteynerdan foydalanadi va ochadi.

3. ***Clear(Color)*** - butun chizma yuzasini tozalaydi va belgilangan fon rangi bilan to‘ldiradi. Color tuzilma bo‘lib uning qiymatlariga kengaytirish amali [::] bilan murojaat qiladilar.

4. ***CopyFromScreen(Int32, Int32, Int32, Int32, Size)*** - Pikseli to‘rtburchakka mos rangli ma‘lumotlarni ekrandan grafikaning chizma yuzasiga bit-blokli uzatishni amalga oshiradi.

*CopyFromScreen(int sourceX, int sourceY, int destinationX, int destinationY, System::Drawing::Size blockRegionSize)* – bunda x, y lar to‘rtburchakning mos koordinatalari va Size sohaning o‘lchamidir.

***CopyFromScreen(Point, Point, Size)*** - Ekrandan grafikaning chizma yuzasiga pikseli to‘rtburchakka mos rangli ma‘lumotlarni bit-blokli uzatishni amalga oshiradi. Point – bu nuqta yaratuvchi tuzilma bo‘lib, o‘zining 3ta Point(Int32), Point(Int32, Int32), Point(Size) konstruktorlari mavjud.

5. ***CreateObjRef(Type)*** - ob‘yekt bilan muloqot qilish uchun ishlatiladigan proksi ishlab chiqarish uchun zarur bo‘lgan barcha tegishli ma‘lumotlarni o‘z ichiga olgan ob‘yekt yaratadi.

Type - requestedType tipidagi abstrakt tip, array<Type^>{int::typeid, int::typeid}); kabi aniqlanadi.

**6. *Dispose()*** - Bu grafika tomonidan ishlatiladigan barcha resurslarni null qiladi.

**7. *DrawArc*.** Bu funksiya uch o‘lchovli sohani ifodalovchi yoyni chizadi.

**DrawArc** funksiyasida **Pen** bu qalam bo‘lib, uning uchun **Pen** sinfi ishlatiladi. **Pen** ning 4 ta konstruktori mavjud bo‘lib, **Pen(Brush)**, **Pen(Brush, Single)**, **Pen(Color)**, **Pen(Color, Single)** konstruktorlardan foydalaniлади, xususiyat va usullarilarni o‘zgartirish orqali o‘rnatish mumkin.

**Brush** bu mo‘yqalam bo‘lib, **Brush** sinfi ishlatiladi. Brush ning **Brush()** konstruktori mavjud bo‘lib, xususiyat va usullarilarni o‘zgartirish orqali o‘rnatish mumkin. Funksiyaning qolgan parametrlari barchasi yuqorida keltirilgan parametrlar kabi amalga oshiriladi.

**8. *DrawBezier*.** Bu funksiya berilgan 4 ta nuqtali soha uchun splayin chizadi.

**9. *DrawBeziers*.** Bu funksiya berilgan nuchtalar sohasi uchun splayin chizadi.

**10. *DrawClosedCurve*.** Bu funksiya yopiq egri chiziq chizishga mo‘ljallangan.

**11. *DrawCurve*.** Bu funksiya berilgan parametrlar asosida spalyn (egri chiziq) chizadi.

**12. *DrawEllipse*.** Bu funksiya berilgan parametrlar asosida ellipsis chizadi.

**13. *DrawIcon*.** Bu funksiya berilgan parametrlar asosida belgilangan belgi bo‘yicha chizadi.

**14. *DrawImage*.** Bu funksiya berilgan parametrlar asosida belgilangan tasvirni chizadi.

**15. *DrawImageUnscaled*.** Bu funksiya berilgan parametrlar asosida tasvirni belgilangan joyda asl kattaligidan foydalanib chizadi.

**16. *DrawLine*.** Bu funksiya berilgan parametrlar asosida chiziq chizadi.

**17. *DrawPath*.** Bu funksiya berilgan parametrlar asosida bog‘langan chiziqlar va egri chiziqlar qatorini chizadi.

**18. *DrawPie*.** Bu funksiya berilgan parametrlar asosida Koordinata jufti, kengligi, balandligi va ikkita radial chiziq bilan belgilangan ellips bilan belgilangan nok shaklini chizadi.

**19. *DrawPolygon*.** Bu funksiya berilgan parametrlar asosida ko‘p shaklini chizadi.

**20. *DrawRectangle*.** Bu funksiya berilgan parametrlar asosida to‘rtburchak shaklini chizadi.

**21. *DrawRectangles*.** Bu funksiya berilgan parametrlar asosida ulanadigan to‘rtburchak shaklini chizadi.

**22. *DrawString*.** Bu funksiya berilgan parametrlar asosida shrift ob‘yektlar bilan belgilangan joyda belgilangan matn chizadi (joylashtiradi).

**23. *EndContainer*.** Bu funksiya berilgan parametrlar asosida joriy grafik konteynerni yopadi va BeginContainer() usuli uchun ko‘rsatkich orqali saqlangan bu grafik holatini chizadi.

**24. *EnumerateMetafile*.** Bu funksiya berilgan parametrlar asosida matnni metafaylga uzatadi va belgilangan joyda chizadi (joylashtiradi).

**25. *Equals*.** Belgilangan ob‘yekt joriy ob‘yektga teng yoki yo‘qligini aniqlaydi. Equals(Object) - Belgilangan ob‘yekt joriy ob‘yektga teng yoki yo‘qligini aniqlaydi.

**26. *ExcludeClip*.** Bu funksiya berilgan parametrlar asosida ko‘rsatilgan maydonni istisno qilish uchun ushbu grafikaning klip hududini yangilaydi.

**27. *FillClosedCurve*.** Bu funksiya berilgan parametrlar asosida yopiq egri chiziqli splaynni bo‘yylgan holda chizadi.

**28. *FillEllipse*.** Bu funksiya berilgan parametrlar asosida ellipsisni bo‘yylgan holda chizadi.

**29. *FillPath*.** Bu funksiya berilgan parametrlar asosida bog‘langan va egri chiziqlar bo‘yagan holda chizadi.

**30. *FillPie*.** Bu funksiya berilgan parametrlar asosida bir juft koordinatalar, kenglik, balandlik va ikkita radial chiziq bilan belgilangan ellipsni bo‘yagan holda chizadi.

**31. *FillPolygon*.** Bu funksiya berilgan parametrlar asosida maydonni

**32. *FillRectangle*.** Bu funksiya berilgan parametrlar asosida to‘rtburchakni bo‘yagan holda chizadi.

**33. *FillRectangles*.** Bu funksiya berilgan parametrlar asosida ulanadigan to‘rtburchakni bo‘yagan holda chizadi.

**34. *FillRegion*.** Bu funksiya berilgan parametrlar asosida sohani bo‘yagan holda chizadi.

### **To‘g‘ri chiziq va ellips chizish komponentalari.**

Tasvirlarni qurish uchun **PictureBox** komponentasidan foydalanamiz.

Chiziq chizish algoritmi. Bunda komponentaning berilgan joyida, rangli chiziq chizishni ko‘ramiz.

Buning uchun 1 ta komponenta, 1 ta tugma va 4 ta **textBox** komponentalarini oynaga qulay qilib joylashtiramiz.

**1-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "DrawLine - chiziq chizish";
```

```
button1->Text = "CHIZISH";
```

**2- qadam.** Komponenta va textBox xususiyatlarini sozlash amallarni bajarish mumkin.

**3- qadam.** Chiziq chizish uchun nuqtalarga atab int myPoint[4]; o‘zgaruvchisi olamiz.

**4- qadam.** Komponentaning Paint degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
e->Graphics->DrawLine(System::Drawing::Pens::Red,  
myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
```

**5- qadam.** Tugmaning button1\_Click hodisasida quyidagi algoritmni joylashtiramiz.

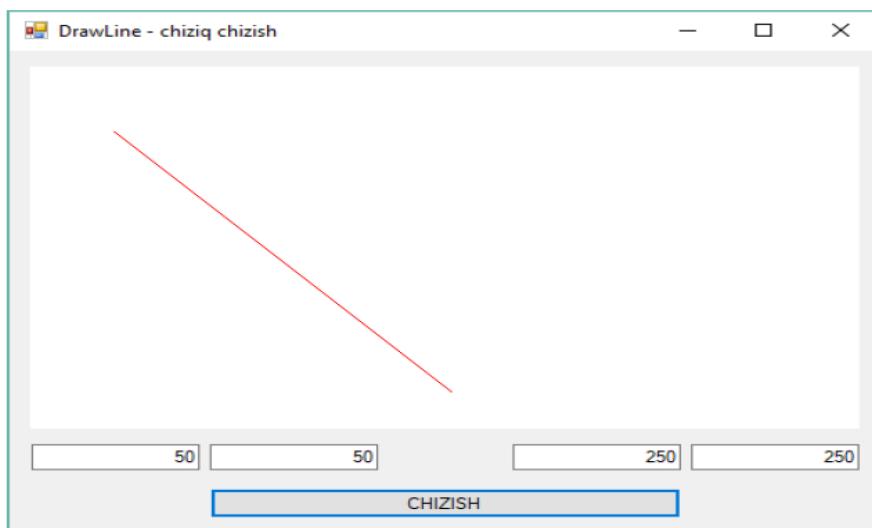
```
myPoint[1] = Convert::ToInt32(textBox1->Text);
myPoint[2] = Convert::ToInt32(textBox2->Text);
myPoint[3] = Convert::ToInt32(textBox3->Text);
myPoint[4] = Convert::ToInt32(textBox4->Text);
pictureBox1->Refresh();
```

Bunda matndan son tipiga o‘tish amalidan foydalanilgan va har bir tugma bosilganda komponentaning chizmasi o‘zgarib turadi.

Agar loyihani ishga tushursangiz, **textBox**larga kerakli sonlarni kiritib, chiziq chizish mumkin.

Komponentaning o‘lchamidan katta chiziqlarni chizish mumkin ammo ko‘rinmaydi. Uni boshqarish lozim, ya‘ni kerakli kattalikdan oshib ketganda foydalanuvchiga muloqot oynasi bilan xabar berish mumkin, buni **textBox**larning fokusi o‘zgarganda amalga oshirish maqsadga muvofiq.

Chiziq chizishning natijasi 5.3.1-rasmda keltirilgan.



5.3.1.-rasm. Chiziq chizish natijasi.

### 3. Turli xil geometrik figuralarni chizish komponentalari.

**Uchburchak chizish.** Bunda komponentaning berilgan joyida, rangli chiziqlar orqali uchbursak chizishni ko‘ramiz. Buning uchun 1 ta komponenta, 1 ta tugma va 6 ta **textBox** komponentalarini oynaga qulay qilib joylashtiramiz. Chuniki 3 ta chiziqning tutashtirsak, uchburchak hosil bo‘ladi.

**1-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "DrawLine - uchburchak chizish";
```

```
button1->Text = "CHIZISH";
```

**2- qadam.** Komponenta va **textBox** xususiyatlarini sozlash amallarni bajarish mumkin.

**3- qadam.** Chiziq chizish uchun nuqtalarga atab int **myPoint[6]**; o‘zgaruvchisi olamiz.

**4- qadam.** Komponentaning **Paint** degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
e->Graphics->DrawLine(System::Drawing::Pens::Red,
```

```
myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
```

```
e->Graphics->DrawLine(System::Drawing::Pens::Black,
```

```
myPoint[3], myPoint[4], myPoint[5], myPoint[6]);
```

```
e->Graphics->DrawLine(System::Drawing::Pens::Blue,
```

**5- qadam.** Tugmaning button1\_Click hodisasida quyidagi algoritmni joylashtiramiz.

```
myPoint[1]= Convert::ToInt32(textBox1->Text);
```

```
myPoint[2]= Convert::ToInt32(textBox2->Text);
```

```
myPoint[3]= Convert::ToInt32(textBox3->Text);
```

```
myPoint[4]= Convert::ToInt32(textBox4->Text);
```

```
myPoint[5]= Convert::ToInt32(textBox3->Text);
```

```
myPoint[6]= Convert::ToInt32(textBox4->Text);
```

```
pictureBox1->Refresh();
```

Agar loyihani ishga tushursangiz, textBoxlarga kerakli sonlarni kiritib, uchburchakni chizish mumkin.

**Ellipsis chizish.** Bunda komponentaning berilgan joyida, rangli ellipsis chizishni ko‘ramiz.

Buning uchun 1 ta komponenta, 1 ta tugma va 4 ta *textBox* komponentalarini oynaga qulay qilib joylashtiramiz.

**1-qadam.** Oynaning *Form1\_Load* hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "Ellipse - chizish";
```

```
button1->Text = "CHIZISH";
```

**2- qadam.** Komponenta va *textBox* xususiyatlarini sozlash amallarni bajarish mumkin.

**3- qadam.** Chiziq chizish uchun nuqtalarga atab *int myPoint[4]*; o‘zgaruvchisi olamiz.

**4- qadam.** Komponentaning *Paint* degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
Pen^pen = gcnew Pen(Color::Black);
```

```
e->Graphics->DrawEllipse(pen,
```

```
myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
```

**5- qadam.** Tugmaning *button1\_Click* hodisasida quyidagi algoritmni joylashtiramiz.

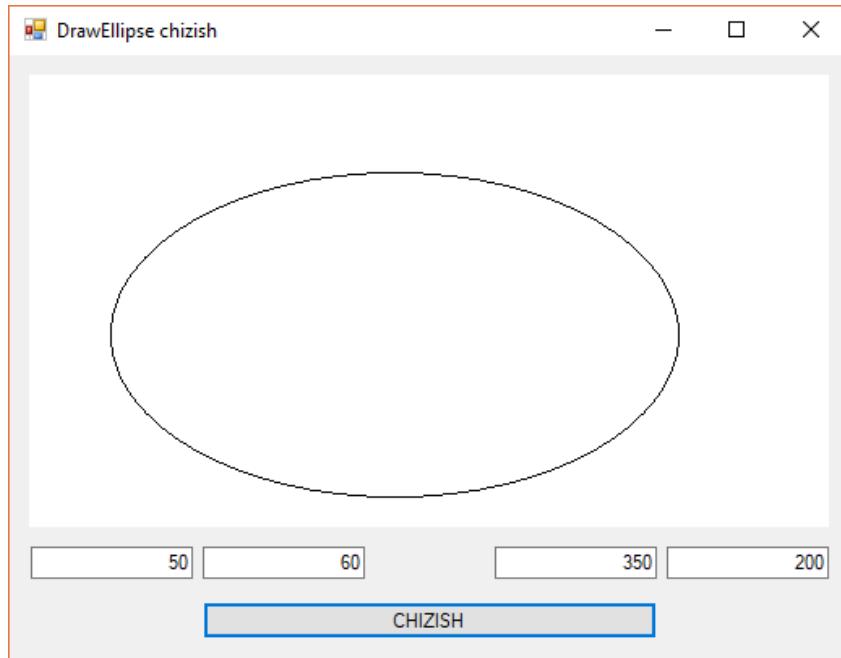
```
myPoint[1] = Convert::ToInt32(textBox1->Text);
```

```
myPoint[2] = Convert::ToInt32(textBox2->Text);
```

```
myPoint[3] = Convert::ToInt32(textBox3->Text);
```

```
myPoint[4] = Convert::ToInt32(textBox4->Text);
```

```
pictureBox1->Refresh();
```



5.3.2.-rasm. *DrawEllipse chizish* natijasi.

Yuqorida keltirilgan chizishlar orqali boshqa ixtiyoriy shakllarni ham chizish mumkin.

### **Nazorat savollari**

1. Grafika sahifa koordinatalarini uchun ishlataladigan o‘lchov birligi sozlash xususiyatini ayting.
2. GraphicsUnit qiymatlari sanab bering.
3. Butun chizma yuzasini tozalaydigan va belgilangan fon rangi bilan to‘ldiradigan usulning nomini ayting.
4. Bu funksiya berilgan parametrlar asosida tasvirni belgilangan joyda asl kattaligidan foydalanib chizadi. Funksiya nomini ayting.
5. Chiziq chizishni qanday amalga oshiriladi?
6. Uchburchak chizishni qanday amalga oshiriladi?
7. Ellipsis chizishni qanday amalga oshiriladi?

## **5.4-§ GUI MUHITIDA GRAFIK IMKONIYATLAR. GUI MUHITIDA GRAFIK HOLAT, TASVIRLARNI VA FUNKSIYA GRAFIKLARINI QURISH.**

**Reja:**

- 1. GUI muhitida grafik holat**
- 2. Tasvirlarni va funksiya grafiklarini qurish**
- 3. Grafik holatda mathlar bilan ishlash**

### **1. GUI muhitida grafik holat**

Visual C++ va C++ Builder muhitlarida, grafikaga aloqasi bor ob'ektlarning 3 xil turi mavjud:

- 1) **Kanva-grafik** chiqishi uchun ishlatalishi mumkin bo'lgan, dastur oynasi ustining bitli kartasini, komponentlar, printerlar va h.k larni taqdim etadi. Kanva mustaqil ob'ekt emas, u doim boshqa bir grafik ob'ektning xossasi bo'ladi.
- 2) **Grafika**—biror bir fayl yoki resursning (bitli obrazni, piktogrammani yoki metafaylni) rastrli tasvirini tashkil qiladi.
- 3) C++Builder boshlang'ich **TGraphic** sinfidan quyidagi ob'ektli sinflarning o'zgaruvchilarini aniqlaydi.:

- TBitmap, – TIcon,
- TMetafile.

Tasvir (**TPicture**) grafik ob'ektlarning istalgan sinfini o'z ichiga olish imkoniga ega, grafiklar uchun konteynerni tashkil qiladi. Natijada TPicture, foydalanuvchi tomonidan belgilangan, bitli obraz, piktogramma, metafayl yoki boshqa bir grafik tipni o'z ichiga olishi mumkin, dastur esa TPicture ob'ekti orqali konteynerning barcha ob'ektlariga murojat qilishi mumkin.

Tbitmap sinfi, Borland C++ Builder da **Graphic::TBitmap** deb belgilangan, u grafik tasvirlarni yaratish va ularning xususiyatlarini boshqarish, xotirani o'qish va diskda saqlash xususiyat va metodlarga ega. Graphic::Tbitmap piksellar massivi ko'rinishidagi rastrli grafik tasvirlarni qo'llagani kabi, bmp. formatidagi

tasvirlarni xam qo'llaydi. Sinfning asosiy xususiyati – *Canvas*. TGraphic sinfi TBitmap grafika bilan ishlash uchun minimal standart interfeysni tashkil etadi.

TIcon va TMetafile sinflari ham Tgraphic sinfining erkin tashkil etuvchilaridir, ular ham grafika bilan ishlash uchun xossa va metodlariga ega, lekin TBitmap dan farqli tomoni shundaki, Canvas xossasiga ega emas. Buni o‘z navbatida, \*.ico \*. va wmf (\*.emf) formatidagi tasvirlardan foydalanish va o‘zgacha qurilishi ko‘rsatib beradi.

Tasvir grafika uchun konteynerdir, ya’ni u grafik ob’ektlarning barcha sinflarni o‘z ichiga olishi mumkin. Konteynerli TPicture sinf bitli obrazga, piktogrammaga, metafayl va boshqa, foydalanuvchi tomonidan belgilangan grafik tipga ega bo‘lishi mumkin, dastur esa “tasvir” ob’ekti orqali, konteynerdagi barcha ob’ektlarga standartlashgan holatda murojat qilishi mumkin.

Xozirgi vaqtda, zamonaviy kompyuterlarda ma’lumot chiqarishning grafik rejimi qo’llaniladi, Windows operatsion tizimi esa simvolli rejimni tanimaydi. DOS operatsion tizimdan reallashtirish bilan, C++ dasturlash tilida, grafik funtsiyalar **graphics.lib** kutubxonasida saqlanadi, bu funktsiyalarning protiplari (xabarlar) esa **graphics.h**. faylida joylashgan bo‘ladi. Bu keltirilgan ma’lumotlar C++ consolli rejimi uchun mo‘ljallangan. Maqsad esa GUI muhitidagi grafik imkoniyatlarni o‘rganish.

Istalgan tasvir, chizma yoki sxemaga grafik primitivlar (nuqtalar, chiziqlar, aylanalar, yoqlar va boshqalar) yig‘indisi sifatida qarash mumkin. Demak ekranda kerakli tasvir paydo bo‘lishi uchun, dastur, grafik elementlarni (shu tasvirni tashkil etuvchi primitivlarni) chizishni (chiqarishni) ta’minlashi lozim.

**Turli shakllarni bo‘yyash.** Bunda komponentaning berilgan joyida, bo‘yyalgan shaklarni chizishni ko‘ramiz. Buning uchun **Brush** – mo‘yqalamdan foydalanib, 1 ta komponenta, 1 ta label va 1 ta **Combox** komponentalarini oynaga qulay qilib joylashtiramiz.

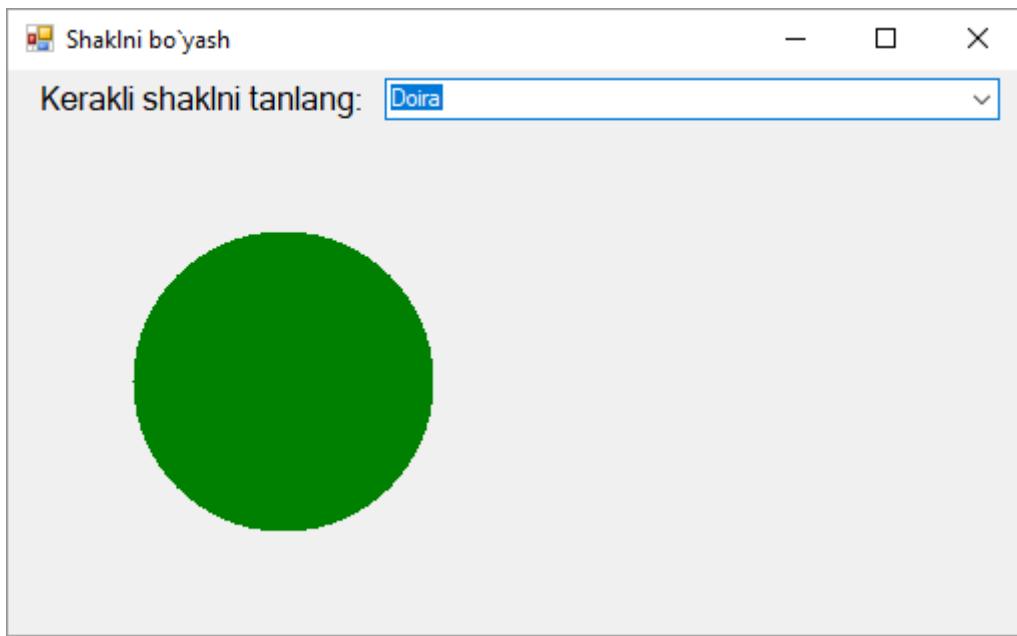
**1-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "Shaklni bo'yash";  
label1->Text = "Kerakli shaklni tanlang:";  
comboBox1->Text = "Shakllar";comboBox1->Items->Add("To'rtburchak");  
comboBox1->Items->Add("Ellipsis");  
comboBox1->Items->Add("Qirqilgan doira ");
```

**2- qadam.** Komponenta va **ComboBox** xususiyatlarini sozlash amallarni bajarish mumkin.

**3-qadam.** **ComboBox** komponentasining **comboBox1\_SelectedIndexChanged** hodisasida quyidagi algoritmni joylashtiramiz.

```
Graphics ^grp = pictureBox1->CreateGraphics();  
Brush ^brsh = gcnew SolidBrush(Color::Green);  
grp->Clear(SystemColors::Control);  
switch (comboBox1->SelectedIndex){  
case 0:  
    grp->FillRectangle(brsh,50,50,150,150); break;  
case 1:  
    grp->FillEllipse(brsh,50,50,300,150); break;  
case 2:  
    grp->FillEllipse(brsh,50,50,150,150); break;  
case 3:  
    grp->FillPie(brsh,50,50,150,150,150,100); break;  
default:
```



5.4.1-rasm. Turli shakllarni bo'yash natijasi.

**Ko'p burchaklarni chizish.** Bunda komponentaning berilgan joyida, bo'yyalgan foydalanuvchi shaklarni chizishni ko'ramiz. Buning uchun **Brush** – mo'yqalamdan foydalanib, 1 ta komponenta, 1 ta **button** komponentasini oynaga qulay qilib joylashtiramiz.

**1-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi dastur kodini joylashtiramiz.

```
this->Text = "Ko'p burchakni chizish";
```

```
button1->Text = "Chizish";
```

**2- qadam.** Komponenta va button xususiyatlarini sozlash amallarni bajarish mumkin.

**3- qadam.** button komponentasining **button1\_Click** hodisasida quyidagi dastur kodini joylashtiramiz.

```
Graphics^graf= pictureBox1->CreateGraphics();
```

```
Pen^gPen= gcnew Pen( Color::Black,1);
```

```
HatchBrush^HBrush= gcnew HatchBrush(Hatchtype::Sphere, Color::Green,  
Color::Black);
```

```
Point point1= Point(50,50);
```

```
Point point2= Point(50,200);
```

```

Point point3= Point(100,250);
Point point4= Point(150,150);
Point point5= Point(240,250);
Point point6= Point(280,25);
Point point7= Point(300,50);
array<Point>^Points= {point1,point2,point3,point4, point5,point6,point7};
graf->DrawPolygon(gPen,Points);
graf->FillClosedCurve(HBrush,Points);

```

Bu dasturdagi **HatchBrush** sinfi ishlashi uchun **using namespace System::Drawing::Drawing2D;** nomlar fazosini qo'shib qo'yish kerak. Algoritmda 7 ta nuqtalarni birlashtirish orqali shakl chiziladi.



5.4.2.-rasm. Turli shakllarni bo'yyash natijasi.

## 2. Tasvirlarni va funksiya grafiklarini qurish.

**Hodisalar orqali shakl chizish.** Hamma "Paint" kabi dasturlarni biladi.

Uning eng ajoyib xususiyatlaridan biri sichqoncha bilan chiziqlar chizishdir. Buni amalga oshirish uchun komponentning sichqoncha bilash ishlash "**"MouseDown"**", "**"MouseUp"** va "**"MouseMove"**" hodisalaridan foydalanish mumkin.

Dasturning algoritmi g'oyasi quyidagicha: foydalanuvchi sichqonchaning chap tugmasini bosganda cursor orqasiga juda ko'p kichik kvadratlar chizila

boshlaydi. Bu kvadratlar hajmi kodda ko‘rsatilgan. Bundan tashqari, "**Button**" tugmachasini rasmga tushiradigan shaklga ko‘chirishingiz kerak.

Bunda komponentaning berilgan joyida, sichqoncha hodisalari orqali shaklarni chizishni ko‘ramiz. Buning uchun **Brush** – mo‘yqalamdan foydalanib, 1 ta komponenta, 1 ta **button** komponentasini oynaga qulay qilib joylashtiramiz.

**1-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi algoritmni joylashtiramiz.

**2- qadam.** Komponenta va **button** xususiyatlarini sozlash amallarni bajarish mumkin.

```
button1->Text = "Tozalash";
```

```
this->Text = "Shakl chizish";
```

**3- qadam.** button komponentasining **button1\_Click** hodisasida quyidagi algoritmni joylashtiramiz. Bu algoritm komponentaning tozalash uchun ishlatiladi.

```
Graphics ^ grp = pictureBox1->CreateGraphics();
```

```
grp->Clear(SystemColors::Window);
```

**4- qadam.** Komponentaning sichqoncha hodisalarini qayta ishlash uchun mantiqiy bir o‘zgaruvchi olinadi, uning qiymati [0] bo‘lsin.

```
Bool Drow = false;
```

**5- qadam.** Sichqoncha komponentaning ustiga kelganda mantiqiy o‘zgaruvchining qiymati 1 ga o‘zgaradi. Chunki chizishni boshlash uchun. Buni amalga oshirishsh uchun sichqonchaning **pictureBox1\_MouseDown** hodisasiga **Drow = true;** ni yozib qo‘yamiz.

**6- qadam.** Sichqoncha komponentaning ustidan ketganda mantiqiy o‘zgaruvchining qiymati 0 ga o‘zgaradi. Chunki chizishni tugatish uchun. Buni amalga oshirishsh uchun sichqonchaning **pictureBox1\_MouseUp** hodisasiga **Drow = false;** ni yozib qo‘yamiz.

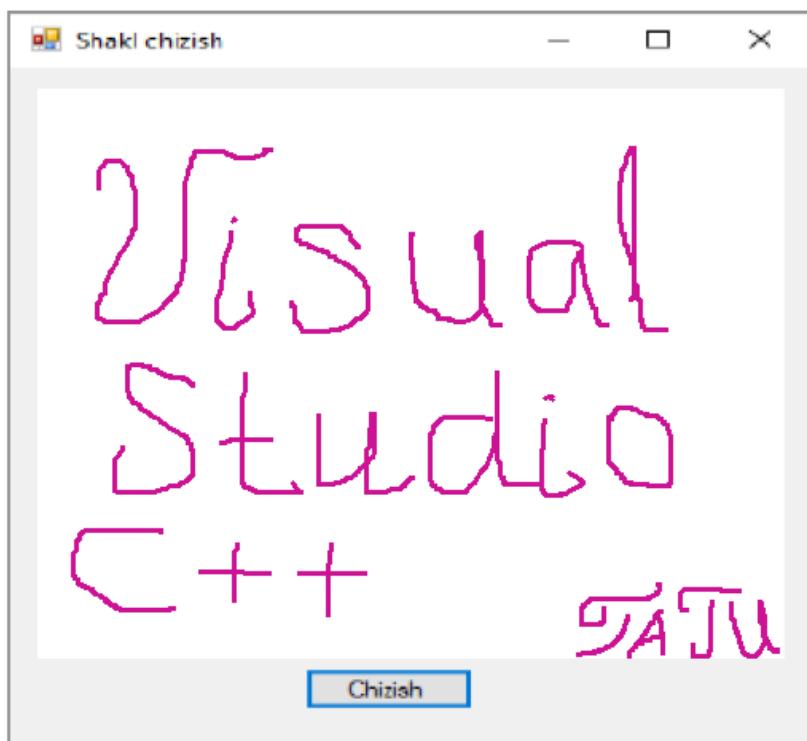
**7- qadam.** Sichqoncha komponentaning ustida kelganda chizishi uchun quyidagi algoritmni kiritamiz.

```

Graphics^ graf = pictureBox1->CreateGraphics();
if (Draw == true) {graf->FillEllipse(Brushes::Violet, e->X, e->Y, 3,3);
// mo 'y qalam qalinligi
}

```

Algoritmda shakllarni hosil qilish uchun juda kichik ellipischalardan foydalanamiz.



5.4.3.-rasm. Sichqoncha bilan turli shakllarni chizish.

**Chart komponenta xususiyati va hodisalari.** Bu komponenta **Data tab** bo‘limiga joylashgan bo‘lib, asosan ma‘lumotlarni infografiklarni yaratish uchun ishlataladi. Infografik uchun ma‘lumotlar to‘plami kerak. Bu komponentaning xususiyati va hodisalari boshqa komponentalarniki kabi bo‘lib, xuddi o‘shalar kabi bo‘limlarga bo‘lingan. Ularning maxsuslarini keltirib o‘tamiz.

1. **BorderSkin** xususiyatlar gruppasi bo‘lib, unda komponentaning yangi niqobga solish mumkin. Niqob deganda, uning yangi ko‘rinishi inobatga olingan. Rang (*color*), rasm (*image*), stil (*style*), kengligi (*width*) kabi xususiyatlari mavjud. Bu xususiyatlarni o‘rnatish muammo keltirib chiqarmaydi. Oldingi

o‘rganganlaringizda bunday xususiyatlardan foydalangansiz. Shuningdek asosiy niqob bu *SkinStyle* bo‘lib, komponentaning asosiy ko‘rinishini o‘zgartrish uchun xizmat qiladi. Uning mos qiymatlar ro‘yxati mavjud, shundan keraklisini tanlab olish mumkin.

2. ***Palette (palitra)*** – xususiyati yordamida komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin. Uning mos qiymatlar ro‘yxati mavjud.

3. ***PaletteCustomColors*** – bunda ham komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin. Ammo foydalanuvchi o‘zining rangi tanlashimi mumkin. Bunda maxsus muloqot oynasi aosida palitraga turli ranglarni qo‘shish mumkin.

4. ***Annotations*** – bunda komponentaga izohlarni yozish mumkin. Buning o‘zining xususiyatlari maxsus muloqot oynasi yordamida o‘rnataladi.

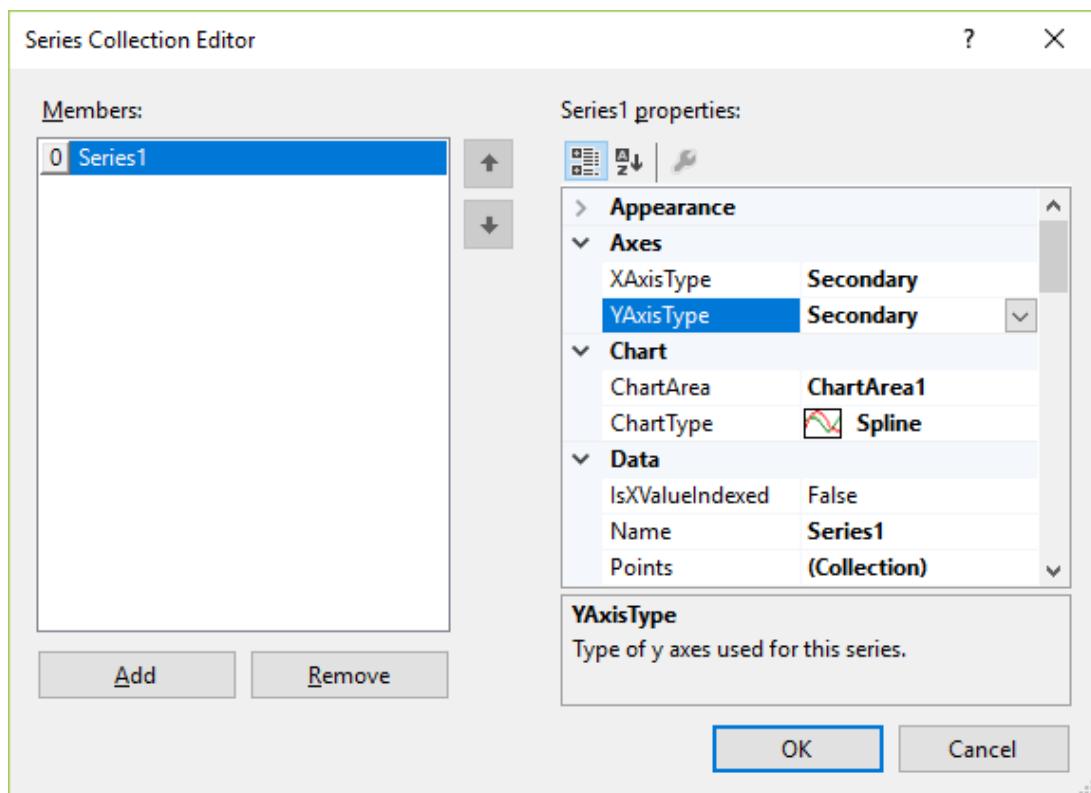
5. ***ChartAreas*** – bu xususiyat orqali komponentaga bir nechata chart infografika joylashtirish mumkin. Buning ham o‘zi mos xususiyatlarini mos muloqot oynasi bilan o‘rnatish lozim.

6. ***Legends*** – infografikaga keltirilgan qiymatlarining joylashish maydoni. Buni ham maxsus muloqot oynasi asosida tahrir qilish mumkin

7. ***Series*** – bu xususiyat asosiy bo‘lib, infografikaning qiymatlarini belgilovchi, har bir qiymat tegishliligini bildiradi. Buning uchun maxsus mulovot oynasi mavjud. Bunga to‘liqroq to‘xtalib o‘tamiz.

8. ***Titles*** – komponentalarga joylashtirilgan infografikalarga sarlavha qo‘yish uchun ishlatiladi. Uning maxsus muloqot oynasi orqali ishlov berish orqali o‘rnatish mumkin.

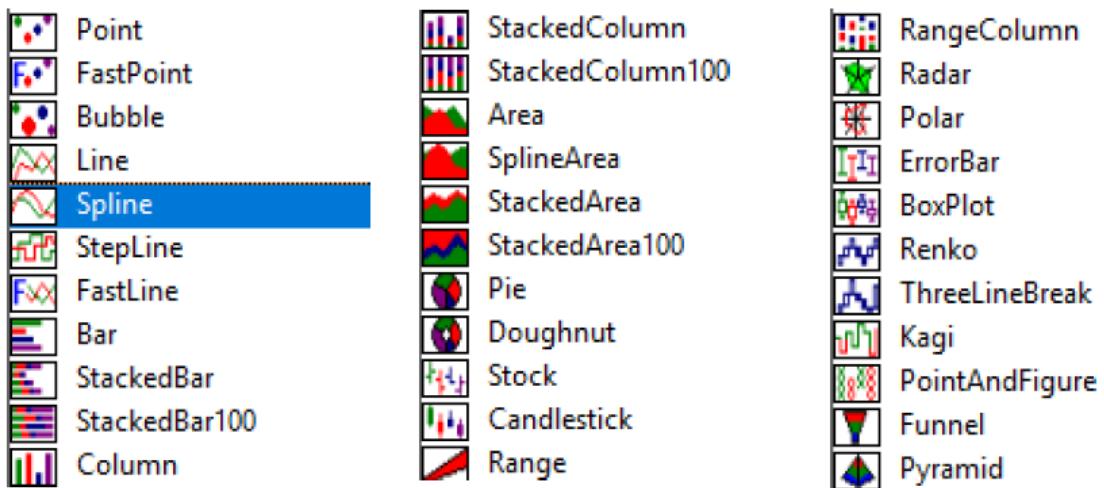
*Series* – bu komponentaga qiymatlarni qo‘shish va uni tasvirlash uchun xizmat qiladi. Komponentaning ichiga joylashtirilgan sohani tahrirlash uchun ishlatiladi. Unga bosganda quyidagi muloqot oynasi chiqadi.



5.4.4.-rasm. Tahrirlash oynasi.

Bu oynda yangi qiymatlar maydoni qo'shish uchun **[Add]** tugmasi va uni o'chirish uchun **[Remove]** tugmasi ishlataladi. Infografikaning asosiy xususiyatlarini boshqarish uchun o'ng tomondagi xususiyatlar panelidan foydalanish mumkin. Unda infografikani taxhrirlash uchun zarur bo'lgan barcha xususiyatlar bor. Ulardan biri bu **ChartType** bo'lib, infografikaning turlarini belgilash uchun xizmat qiladi. Uning turlari quyidagi 5.10-rasmda keltirilgan.

Shuningdek, infografikaning ma'lumotlari, yozuvlari, qiymatlar, maydoni, qiymat turlari, chegaralari bilan ishslash xususiyatlari xam muvjud. Odatda bu xususiyatlarni dasturlash orqali dastur fragmentlarida o'rnatish fa foydalanish dasturchiga qulay hisoblanadi. Ammo vizual dusturlashning imkoniyatidan foydalanish uchun buni ham ishlatischni o'rganish lozim.



#### 5.4.5.-rasm. Infografikaning turlari

**Funksiyalarni grafiklarini qurish.** Infografika komponentasiga mos ravishda funksiyalarni grafikgini chizish usullari ko‘rib chiqamiz.

Matematik funksiyalarni grafiklarini chizish uchun avval shu funksiyalarni bir sinfga yaratib olamiz.

*private value class MyFunction*

{

*private:*

*double \_value;*

*public:*

*double getValuePow(double x){return Math::Pow(x, 2);}*

*double getValueX4(double x){return x\*x\*x\*x;}*

*double getValuekxa(double x, int k, int a){return k\*x+a; }*

Bu ko‘rinishda matematikaning barcha funksiyalarini yaratib olish yoki to‘g‘ridan to‘g‘ri foydalanish mumkin.

**Chart** komponentasiga grafikni chizish uchun **Form1\_Load** hodisasiga quyidagicha algoritmni yozamiz.

*chart1->Series->Clear();*

*Series^ series1 = gcnew Series(L"X^2 grafigi");*

*// rangni tanlash*

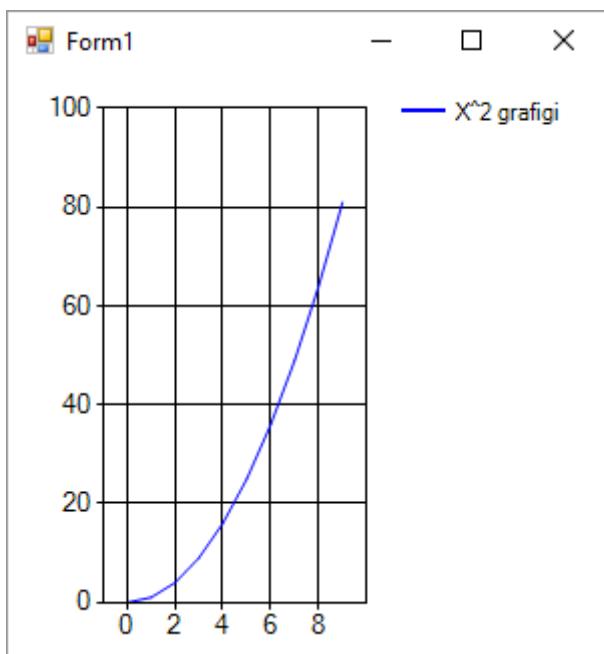
*series1->Color = Color::Blue;*

```

series1->IsVisibleInLegend = true;
series1->IsXValueIndexed = true;
// infografikani turini tanlash
series1->ChartType = SeriesChartType::Line;
// qiymatlar qatlamini qo'shish
chart1->Series->Add(series1);
// qiymatlarni
MyFunction^ func = gcnew MyFunction();
for (double i = 0; i < 10; i++) {
    series1->Points->AddXY(i, func->getValuePow(i));
}

```

Dasturni ishga tushursak, dastur bitta  $x^2$  funksiyaning grafigini chizish imkoniyatini beradi.



5.4.6.-rasm.  $x^2$  funksiyaning grafigi.

Bir vaqtning o'zida bir nechta funksiyaning grafiklarini chizish uchun yuqorida aniqlangan sinfdan foydalanib, **Form1\_Load** hodisasiga quyidagicha algoritmi yozamiz.

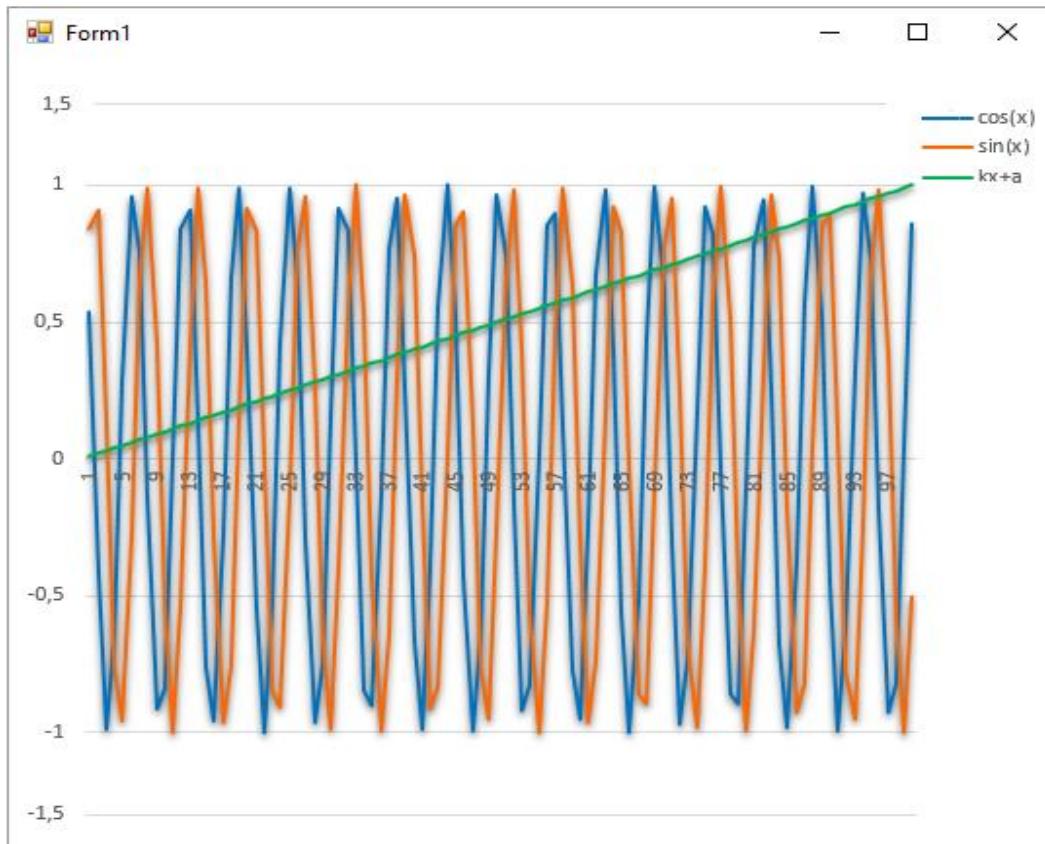
```
chart1->Series->Clear();
```

```

Series^ series1 = gcnew Series(L"Cos(x)");
Series^ series2 = gcnew Series(L"Sin(x)");
Series^ series3 = gcnew Series(L"kx+a");
// rangni tanlash
series1->Color = Color::Blue;
series2->Color = Color::Red;
series3->Color = Color::Green;
// infografikani turini tanlash
series1->ChartType = SeriesChartType::Spline;
series2->ChartType = SeriesChartType::Spline;
series3->ChartType = SeriesChartType::Spline;
// qiymatlar qatlamini qo'shish
chart1->Series->Add(series1);
chart1->Series->Add(series2);
chart1->Series->Add(series3);
MyFunction^ func = gcnew MyFunction();
for (double i = 0; i <= 100; i++) {
    series3->Points->AddXY(i, func->getValuekxa(i*0.01,1,0));
    series1->Points->AddXY(i, Math::Cos(i));
    series2->Points->AddXY(i, Math::Sin(i));
}

```

Dasturni ishga tushursak, dastur funksiyalarning grafigini chizish imkoniyatini beradi.



5.4.7.-rasm. Funksiyalarning grafigi.

Gistogramma grafiklarini chizish uchun massivlardan yoki ixtiyoriy to‘plamlardan foydalanish mumkin. Buning uchun 3 ta massiv olamiz, **Form1\_Load** hodisasiga quyidagicha dasturiy kodni yozamiz

```

Chart1->Series->Clear();
Title^title = gcnew Title("Sotilgan texnikalar soni");
chart1->Titles->Add(title);
Series^ series1 = gcnew Series(L"Telefon");
Series^ series2 = gcnew Series(L"Kompyuter texnikasi");
Series^ series3 = gcnew Series(L"Avtomashina");
// rangni tanlash
series1->Color = Color::Blue;

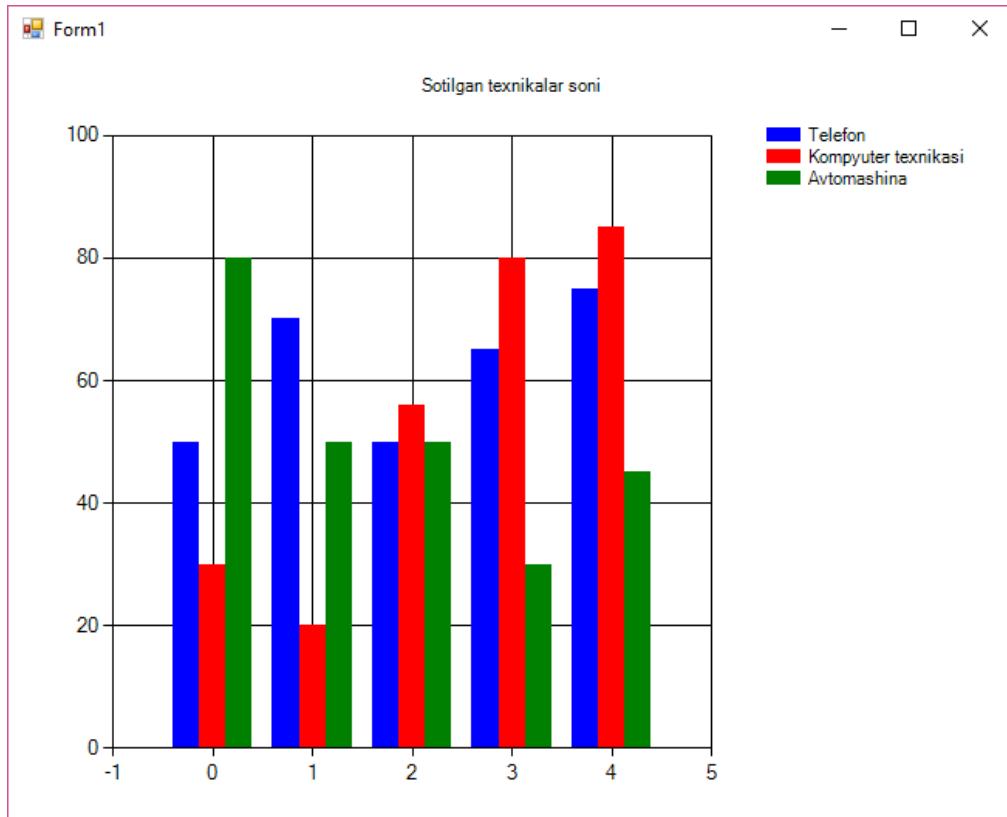
```

```

series2->Color = Color::Red;
series3->Color = Color::Green;
// infografikani turini tanlash
series1->ChartType = SeriesChartType::Column;
series2->ChartType = SeriesChartType::Column;
series3->ChartType = SeriesChartType::Column;
// qiymatlar qatlamini qo'shish
chart1->Series->Add(series1);
chart1->Series->Add(series2);
chart1->Series->Add(series3);
// qiymatlarni
array<int>^ arr1 = {30,20,56,80,85};
array<int>^ arr2 = {50,70,50,65,75};
array<int>^ arr3 = {80,50,50,30,45};
for (int i = 0; i < arr1->Length; i++) {
    series3->Points->AddXY(i, arr3[i]);
    series1->Points->AddXY(i, arr2->GetValue(i));
    series2->Points->AddXY(i, arr1->GetValue(i));
}

```

Dasturni ishga tushursak, Gistogramma grafigini chizish imkoniyatini beradi.



5.4.8.-rasm. Gistogramma grafiklarini chizish

### 3. Grafik holatda mathlar bilan ishlash

**Matnlarni tasvir kabi joylashtirish.** Keling komponentaning belgilangan joyida, aniq o'lchamli va rangli matnni joylashtirishni ko'ramiz. Buning uchun komponentaga 3 ta **textbox** va 1ta **button** komponentlarini qulay qilib joylashtiramiz. Birinchisi matn yozish uchun, qolgan ikkitasi matnni joylashtirish nuqtasi uchun.

**1- qadam.** *Textbox* ni rejimini **multiline** rejimiga va **scrollBars** xususiyatiga **Vertical** qiymati o'rnatiladi.

**2- qadam.** Oynaning **Form1\_Load** hodisasida quyidagi dasturiy kodni joylashtiramiz.

```
Font = gcnew System::Drawing::Font("Times New Roman", 12,  
FontStyle::Bold);
```

```
button1->Text = "Chizish";
```

```
this->Text = "Matn chizish";
```

**3- qadam.** Komponenta va **Button** xususiyatlarini sozlash amallarni bajarish mumkin.

**4- qadam.** **Button** komponentasining **button1\_Click** hodisasida quyidagi dasturiy kodni joylashtiramiz. Bu dastur komponentaning tozalash uchun ishlatiladi.

```
String^ Text = String::Format("{0}", textBox1->Text);
```

```
Brush^ brsh = gcnew SolidBrush(Color::LimeBlue);
```

```
Graphics^ G = pictureBox1->CreateGraphics();
```

```
G->TextRenderingHint = System::Drawing::Text::
```

```
TextRenderingHint::AntiAlias;
```

```
int xP = Convert::ToInt16(textBox2->Text);
```

```
int yP = Convert::ToInt32(textBox3->Text);
```

Dasturni ishga tushuramiz va quyidagi natijani olamiz



5.4.9.-rasm. Matnlarni joylashtirish.

**Matnli rasmlarni yaratish.** Matnli rasmlarni yaratish uchun 1 ta komponenta, 4 ta **label**, 4 ta **button**, 4 ta **textbox** larni qulay qilib joylashtiriladi.

**1-qadam.** 2 ta **Textbox** ni rejimini **multiline** rejimiga va **scrollBars** xususiyatiga **Vertical** qiymati o‘rnataladi.

**2-qadam.** Oynaning **Form1\_Load** hodisasida quyidagi algoritmni joylashtiramiz.

```
this->label1->Text = "/ Yuqoridagi matn";
```

```
label2->Text = "/";
```

```
label3->Text = "/ Pastdagi matn";
```

```
label4->Text = "/";
```

```
button1->Text = "Tozalsh";
```

```
button2->Text = "Chizish";
```

```
button3->Text = "Saqlash";
```

```
button4->Text = "Rasm yuklash";
```

**3- qadam.** Komponenta va **button** xususiyatlarini sozlash amallarni bajarish mumkin.

**4- qadam.** **TextBox3** va **TextBox4** ni mos ravishda **TextBox1** va **TextBox2** lar orqasiga tashlaymiz.

**5- qadam.** Formaning **formBorderStyle** xususiyatiga **SizableToolWindow** ni o‘rnatamiz. Formani o‘lchamlarini o‘zgartirmaslik uchun, buni boshqacha ham amalga oshirish mumkin.

**6- qadam.** **TextBox3** va **TextBox4** ni shrift o‘lchamlarini 12, **TextBox1** va **TextBox2** lar boshqa kattaroq o‘lchamni o‘rnatamiz. Chunki dastur ishga tushgada asosiy ko‘rinadigann matnlar ko‘rinmaydigan matnlarga ko‘chiriladi va undan komponentaga ko‘chiriladi.

**7- qadam.** Zaruriy o‘zgaruvchilarni aniqlashmiz.

```
private: Bitmap^ bmp_for_draw;
```

```
private: String^ full_name_of_image;
```

**8- qadam.** Oynaning **Form1\_Load** hodisasi davomidan quyidagi dasturiy kodni joylashtiramiz.

```
Text = "Matnni rasmga joylashtirish";
Font = gcnew System::Drawing::Font("Times New Roman", 32,
FontStyle::Bold);
this->pictureBox1->SizeMode = PictureBoxSizeMode::StretchImage;
```

**9- qadam.** Komponentaga kerakli rasmni yuklash uchun **button4\_Click** hodisasiga quyidagi dasturiy kodni yoziladi

```
open_dialog->Filter = "Image Files (*.BMP; *.JPG; *.GIF;
*.PNG)/*.*; *.BMP; *.JPG; *.GIF; *.PNG /All files (*.*)/*.*";
OpenFileDialog^ open_dialog = gcnew OpenFileDialog();
if (open_dialog->ShowDialog() == System::Windows::
Forms::DialogResult::OK) {
try { full_name_of_image = open_dialog->FileName;
bmp_for_draw = gcnew Bitmap(open_dialog->FileName);
pictureBox1->Image = bmp_for_draw;
pictureBox1->Invalidate();
} catch (Exception^ e) {
System::Windows::Forms::DialogResult result = MessageBox::Show("Fayl
xato tanlandi" + e->ToString(), "Warning", MessageBoxButtons::OK,
MessageBoxIcon::Error);
```

**10- qadam.** Komponentadagi rasmni saqlash uchun **button3\_Click** hodisasiga quyidagi algoritmni yoziladi

```
if (pictureBox1->Image != nullptr){
String^ format = full_name_of_image->Substring(full_name_of_image-
>Length - 4, 4);
SaveFileDialog^ savedialog = gcnew SaveFileDialog();
savedialog->OverwritePrompt = true;
// Agar shu nomli fayl bo'lsa
```

```

savedialog->CheckPathExists = true;
// Agar noto 'g 'ri nom kiritilsa
savedialog->ShowHelp = true;
savedialog->Filter = "Image Files (*.BMP)/*.BMP/Image
Files (*.JPG)/*.JPG/Image
Files (*.GIF)/*.GIF/Image Files (*.PNG)/*.PNG/All files (*.*)/*.*";
if (savedialog->ShowDialog() == System::Windows::
Forms::DialogResult::OK){
try { Bitmap^ MBB = gcnew Bitmap(pictureBox1->Image);
MBB->Save(savedialog->FileName, System::Drawing::
Imaging::ImageFormat::Jpeg);}
catch(Exception^ e){ MessageBox::Show("Rasmni saqlashda xatolik", "FATAL
ERROR");}
}

```

**11- qadam.** Komponentadagi rasmni tozalash uchun **button1\_Click** hodisasiga quyidagi dasturni yoziladi

```

Bitmap^ BM = gcnew Bitmap(pictureBox1->Image);
Graphics^ C = System::Drawing::Graphics::FromImage(BM);
//Graphics C = CreateGraphics();
C->Clear(pictureBox1->BackColor);
pictureBox1->Image = BM;

```

**12- qadam.** Komponentaga matnlarni joylashtirish uchun uchun **button2\_Click** hodisasiga quyidagi algoritmni yoziladi

```

System::Drawing::Font^ FontTemp = gcnew System::Drawing::Font("Times
New Roman", 32, FontStyle::Bold);
textBox4->Text = textBox1->Text;
textBox3->Text = textBox2->Text;
SaveFileDialog^ savedialog = gcnew SaveFileDialog();
String^ Text = String::Format("{0}", textBox4->Text);
String^ Txt = String::Format("{0}", textBox3->Text);

```

```

Brush^ brsh = gcnew SolidBrush(Color::White);
Bitmap^ MBB = gcnew Bitmap(pictureBox1->Image);
Graphics^ G = System::Drawing::Graphics::FromImage(MBB);
Graphics^ Q = G;
G->TextRenderingHint =
System::Drawing::Text::TextRenderingHint::AntiAlias;
G->DrawString(Text, FontTemp, brsh, 75, 2);
Q->TextRenderingHint =
System::Drawing::Text::TextRenderingHint::AntiAlias;
Q->DrawString(Txt, FontTemp, brsh, 5, 225);
pictureBox1->Image = MBB;

```

Dasturni ishga tushiramiz va quyidagi natijani olamiz



5.4.10-rasm. Matnlarni rasmga joylashtirish.

Visual C++ da grafika sinfi, uning usullari va turli shakllarni chizish usullari, **Chart** va **Shape** sinflarining xususiyatlari va usullarini ko‘rib chiqdik. Bularni o‘ziga xos ma’lumotlarda ishlatishni tavsiya qilamiz.

## **Nazorat savollari**

1. Hodisalar orqali shakl chizishni qanday amalga oshiriladi?
2. Rasmlarni o‘zgartirishni qanday amalga oshiriladi?
3. Matnlarni tasvir kabi joylashtirishni qanday amalga oshiriladi?
4. Matnli rasmlarni yaratishni qanday amalga oshiriladi?
5. Chart komponenta xususiyati va hodisalari haqida gapirib bering?
6. Chart komponentasi yordamida foydlanuvchining ixtiyoriy ma‘lumotlarini infografikasini yaratish mumkinmi?
7. Komponentaning chegara rang qalinligini o‘rnatadi va qiymatlari sonlardan iborat bo‘lgan xususiyat nomini ayting?
8. Shakllarning FillColor, FillGradientColor, FillGradientStyle, FillStyle xususiyatlaridan foydalanib, nima qilish mumkin?
9. Tugmaning ko‘rinishini o‘zgartirish uchun nima amallar bajarilishi kerak?
10. Rasmlarni harakatlantirishda oqimlardan qanday foydalanamiz?

## **5.5-§ MULOQOT OYNALARI BILAN ISHLASH. GUI MUHITIDA MULOQOT OYNALARI VA ULARNI SOZLASH, BOSHQARISH ELEMENTLARI.**

**Reja:**

- 1. Visual C++ muhitida muloqot oynalari**
- 2. Muloqot oynalarini sozlash**
- 3. Muloqot oynalarini boshqarish elementlari**
- 4. Muloqot oynalarini yaratish**

### **1. Visual C++ muhitida muloqot oynalari.**

Foydalanuvchilar bilan tizimning muloqotini interaktiv amalga oshiri uchun muloqot oynalari kerak. Muloqot oynalari *3 ta katta guruh*larga bo‘linadi. Tizimli, ya‘ni OT bilan ishlashga mo‘ljallangan, interaktiv xabarlarni berish va aniq javoblarni olish uchun mo‘ljallangan, dasturchining yoki foydalanuvchining tashabbusi bilan yaratiladigan muloqot oynalari bor.

Muloqot oynalari yaratish ularni maqsadlaridan kelib chiqqan holda amalga oshiriladi. Bu oynalarning o‘zining talablari mavjudki, shu talablar bajarilsa, u *muloqot oynasi* bo‘la oladi.

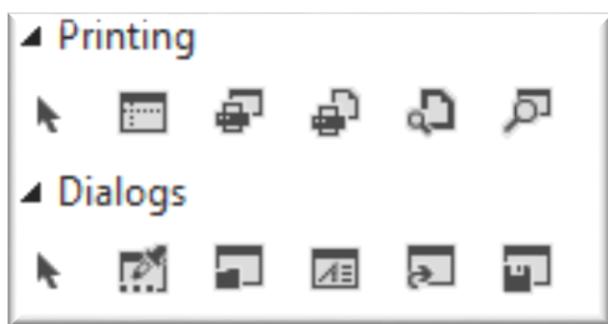
Bu talablarga quyidagilar kiradi:

1. Muloqot oynasining sarlavhasi bo‘lishi va unda faqat oynani yopish tugmasining bo‘lishi lozim, tizimli menu va boshqa tugmalar bo‘lishi mumkin emas. Istechno tariqasida ba‘zi hollarda, yordam tugmasini joylashtiish mumkin.
2. Muloqot oynasi teskari aloqaga mo‘ljallanganligi uchun, uni shartini bajarmasdan tizimning boshqa oynasiga o‘tish mumkin emas.
3. Teskari aloqaning bir nesta turlarini amalga oshiruvchi tugmalar bo‘lishi kerak.
4. Muloqot oynalar asosiy oynadan har doim kichik bo‘lshi shart.
5. Muloqot oynanining asosiy maqsadi yoki turi aniq kelitrilishi kerak, nima munosabat uchun muloqot oynasi chiqqanligi.

6. Muloqot oynasidan boshqa muloqot oynasiga o‘tish mumkin emas, asosiy oynaga o‘tish lozim.
7. Muloqot oyna murojaat qilinganda yaratilishi va teskari aloqa qabul qilingandan so‘ng xotiradan o‘chirib tashlanishi lozim.
8. Muloqot oynaga dinamik xotiralar bo‘lishi mumkin emas.

Ushbu talablarni bajargan har qanday oyna ***muloqot oynasi*** hisoblanadi.

Visual C++da OT bilan muloqot qilishga mo‘ljallangan muloqot oynalariga [Dialogs] tab dagi va [Printing] tab dagi barcha komponentalar kiradi (5.18 - rasmga qarang).



#### *5.5.1.-rasm. Tizimli muloqot oynalari yaratish komponentalari.*

Visual C++ning hujjatlariga qarasangiz tizimli muloqot oynalari uchun yagona muloqot oynalari ro‘yxati tuzilgan. Bu ro‘xatga quyidagi jadvalga keltirilgan komponentalar kiradi.

Visual C++da OT bilan muloqot qilishga mo‘ljallangan muloqot oynalari yaratishga mo‘ljallangan komponentalarning vazifalari.

5.8-jadval.

<b>Nº</b>	<b>Komponenta nomi</b>	<b>Vazifasi</b>
1	ColorDialog	Foydalanuvchilar interfeys elementi rangini o‘rnatish imkonini beruvchi ranglar palitrasini uchun muloqot oynasini ko‘rsatadi.

2	FontDialog	Foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi.
3	OpenFileDialog	Foydalanuvchilar uchun faylni tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.
4	PrintDialog	Foydalanuvchilarga printerni tanlash va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi.
5	PrintPreviewDialog	Foydalanuvchilar uchun chop qilishda PrintDocument boshqaruv elementining ko‘rinishining ko‘rsatish imkonini beradigan muloqot oynasini ko‘rsatadi.
6	FolderBrowserDialog	Foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.
7	SaveFileDialog	Foydalanuvchilar uchun faylni saqlash imkonini beradigan muloqot oynasini ko‘rsatadi.

Bu komponentalar maxsus xususiyatga asoslagan koponentalar bilash ishlatiladi.

Interaktiv xabarlarni berish va aniq javoblarni olish Visual C++ da **MessageBox** sinfi mavjud. Bu sinf bilan barcha ixtiyoriy turdagি muloqot oynalari yaratish mumkin. Sinfning nomlar fazosi **System.Windows.Forms** hisoblanadi va kutubxonasi **System.Windows.Forms.dll** hisoblanadi.

Bu muloqot oynasi forma sinfining merosxo‘ri hisoblanadi. Unda 21 ta turli kombinatsiyali **show** funksiyachi bor. Uning quyidagi parametrlari bor.

5.9-jadval. MessageBox sinfnining show funksiyasi parametrlari

Paramert nomi	Tipi	Vazifasi
Text	String	Muloqot oynasining xabari
Caption	String	Muloqot oynasining sarlavhasi
Buttons	MessageBoxButtons	Teskari aloqani ta'minalash tugmalari turlarini aniqlash
Icon	MessageBoxIcon	Muloqot oynalarining ikonkalari turlarini aniqlash
Defaultbutton	MessageBoxDefaultButton	Teskari aloqani ta'minalash uchun joriy tugmalari turlarini aniqlash
Options	MessageBoxOptions	Muloqot oynalarining amallari turlarini aniqlash
Helpfilepath	HelpNavigator	HelpNavigator ob'yektining qiymatlari uchun foydalilaniladi
Param	Object	Yordam tugmasi bosilganda ID formatni aniqlash imkonini beruvchi parametr
Returns	DialogResult	Muloqot oynalarining qiymatlari qaytarish uchun foydalilaniladigan DialogResult tipidagi turlarini aniqlash
Exceptions	InvalidOperationException	MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton ob'yektlarini birini qabul qiluvchi kengaytirilgan ob'yekt tipi
Owner	IWin32Window	Muloqot oynasini egasi Iwin32window amalga oshirish

Bu paramertlarning o‘ziga mos qiymatlari oldindan aniqlab berilgan bo‘lib, oddiy sodda ko‘rinishda muloqot oynasini yaratish imkoni beradi. Keyinroq bu sinf paramertlari qiymatlari va ularga ishlov berishni ko‘rib chiqamiz.

Foydalanuvchi tomonidan yaratiladigan muloqot oynasi forma kabi yaratiladi va ularni loyhalash dasturchining yoki foydalanuvchining xoxishiga qarab amalga oshiriladi. Interaktiv muloqot oynalari kabi interaktiv tugmalarni yaratish va ularni boqarish, kerakli ma‘lumotlarni olish uchun ishlatiladi. Shuni ham inboatga olish keraki yaratiladigan muloqot oynasi talablarga mos kelishi kerak. Bu talablarni amalga oshirish uchun formaning xususiyatlariga ishlov berish, lozim bo‘lsa, asosiy oynda kerakli xususiyatlarni o‘rnatish mumkin. Bunda ham formaning *show* usuli mavjud bo‘lib. Shu orqali forma chaqiriladi.

Sinfning nomlar fazosi *System.Windows* bo‘lib hisoblanadi va kutubxonasi *PresentationFramework.dll* hisoblanadi. Bu muloqot oynasi forma sinfining meros xo‘ri hisoblanadi.

## 2. Muloqot oynalarini sozlash.

Bu muloqot oynalarini sozlash uchun tizimli muloqot oynalaridan foydalanish va ularga ishlov berish nazarda tutilgan. Yuqorida keltirilgan **7 ta muloqot oynalaridan** foydalanishlar, xususiyatlarini va hodisalarini boshqarish to‘g‘risida to‘xtalamiz.

**1. ColorDialog muloqot oynasi.** Bu oyna - foydalanuvchilar interfeys elementi rangini o‘rnatish imkonini beruvchi ranglar palitrasи uchun muloqot oynasini ko‘rsatadi. Bu komponentani formaga o‘rnatilganda hech qanday ko‘rinish hosil bo‘lmaydi, ammo formaning ichki tuzilmasiga qo‘yshiladi. Fomraning ishchi holatidagi formasining pastki qismida uning ob‘yekti yaratiladi.

U **ColorDialog1** ob‘yektini yaratish orqali boshqariladi. Uning xususiyatlari va hodisalari ham mavjud va loyiha oynasida foydalanuvchi xususiyatlar oynasiga chiqadi. U yerdan kerakli ixtiyoriy aynan shu ob‘yektga mos xususiyat va hodisalarni o‘rnatish mumkin.

**ColorDialog** muloqot oynasidan foydalanish uchun **ColorDialog()** konstruktorini ishga tushirish lozim.

Bu sinfning xususiyailari, usullari va hodisalari mavjud.

#### 5.10-jadval. ColorDialog muloqot oynasining xususiyatlari

Xususiyatlari	Vazifasi
AllowFullOpen	Maxsus ranglar aniqlash uchun muloqot oynasini foydalanish mumkin yoki yo‘qligini o‘rnatish
AnyColor	Muloqot oynasida asosiy ranglar majmuining barcha mavjud ranglar ko‘rsatish yoki yo‘qligini o‘rnatish
CanRaiseEvents	Komponentaga bir hodisa o‘rnatish mumkinligini aniqlash
ColorContainer	Foydalanuvchi tomonidan tanlangan rangni o‘rnatish
Container	Komponentini o‘z ichiga olgan Icontaineni o‘rnatish
CustomColors	Muloqot oynasida ko‘rsatilgan maxsus ranglar to‘plamini oladi yoki o‘rnatadi
DesignMode	Komponenta joriy dizayn rejimida ekanligini ko‘rsatadigan qiymatni oladi
Events	Komponentaga ilova qilinadigan hodisalar ro‘yxatini oladi
FullOpen	Muloqot oynasi ochilganda maxsus ranglarni yaratish uchun ishlatiladigan boshqaruv elementlari ko‘rinib turishini ko‘rsatuvchi qiymatni oladi yoki o‘rnatadi
Options	Colordialog boshlash uchun xususiyatlarni oladi
ShowHelp	Yordam tugmasi rang muloqot oynasidagi paydo yoki yo‘qligini bo‘lishini o‘rnatish
Site	Komponentning ISiteni oladi yoki o‘rnatadi
SolidColorOnly	Muloqot oynasi qattiq ranglar tanlash uchun foydalanuvchilar cheklash yoki yo‘qligini o‘rnatish

Xususiyatlari	Vazifasi
Tag	Nazorat haqida ma'lumotlarni o'z ichiga olgan ob'yekt sozlash

5.11-jadval. ColorDialog muloqot oynasining usulari

Usular	Vazifasi
CreateObjRef(Type)	Ob'yekt bilan muloqot qilish uchun ishlatiladigan proksi ishlab chiqarish uchun zarur bo'lgan barcha tegishli ma'lumotlarni o'z ichiga olgan ob'yekt yaratadi
Dispose()	Komponent tomonidan ishlatiladigan barcha resurslarni chiqaradi
Dispose(Boolean)	Tarkibiy qism tomonidan ishlatiladigan boshqarilmaydigan resurslarni chiqaradi va ixtiyoriy ravishda boshqariladigan resurslarni chiqaradi
Equals(Object)	Belgilangan ob'yekt joriy ob'yektga teng yoki yo'qligini aniqlaydi
GetHashCode()	Standart hesh funksiyasi sifatida xizmat qiladi
GetService(Type)	Komponenta yoki uning konteyneri tomonidan taqdim etilgan xizmatni ifodalovchi ob'yektni qaytaradi
GetType()	Joriy ob'yekt turini oladi
HookProc (IntPtr, Int32, IntPtr, IntPtr)	Umumiy muloqot oynasiga xos funksiyalarni kiritish uchun bekor qilingan umumiy muloqot oynasi protsedurani belgilaydi
MemberwiseClone()	Joriy ob'yektning oddiy nusxasini yaratadi
MemberwiseClone (Boolean)	Joriy Marshalbyrefobject ob'yekt oddiy nusxasini yaratadi
OwnerWndProc(IntPtr, Int32, IntPtr, IntPtr)	Umumiy muloqot oynasiga maxsus funksiyalarni qo'shish uchun bekor qilinadigan oyna tartibini belgilaydi

Usular	Vazifasi
Reset()	Ularning standart xususiyatlarga barcha imkoniyatlari ishga soladi, oxirgi tanlangan qora rang va ularning standart xususiyatlarga maxsus ranglar moslashtiriladi
RunDialog(IntPtr)	Umumiy muloqot oynasini belgilaydi
ShowDialog()	Umumiy muloqot oynasini ishlatadi
ShowDialog(IWin32Window)	Umumiy muloqot oynasini ishlatadi
ToString()	Muloqot oyna qiymatini satrga o'tkazish

5.12-jadval. ColorDialog muloqot oynasining hodisalari

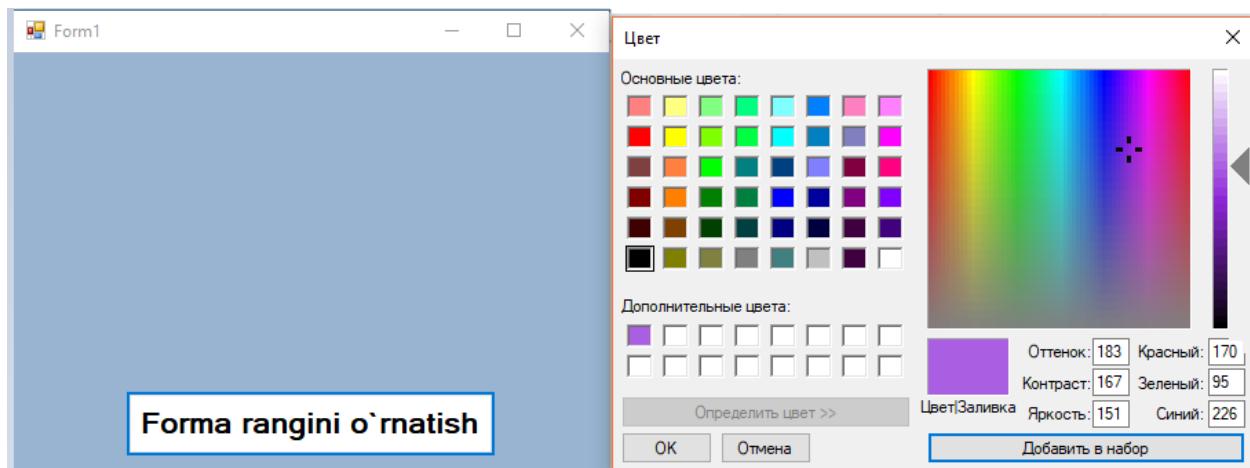
Disposed	Komponentni Dispose() metodiga chaqiriq orqali dispozitsiya qilinganda yuzaga keladi
HelpRequest	Foydalanuvchi umumiy muloqot oynasidagi yordam tugmasini bosganda sodir bo'ladi

*ColorDialog* muloqot oynasidan foydalanish uchun formaga bir tugma o'rnatamiz va uning **Click** hodisasi yordamida chaqiramiz. Tanlangan rang esa formaning va tugmaning fonini o'zgartirsin.

Tugmaning **Click** hodisasida quyidagicha dastur fragmentini o'rnatamiz.

```
if(colorDialog1->ShowDialog() == ::System::Windows::  
Forms::DialogResult::OK)  
Form1::BackColor = colorDialog1->Color;  
button1->BackColor = colorDialog1->Color;
```

Dastur fragmentida muloqot oynasi chaqirilganda va teskari aloqasi OK ob'yektini qaytarsa forma va tugmaning mos xususiyatlar ranglarini o'zgartirish algoritmi yozilgan.



5.5.2.-rasm. *ColorDialog* muloqot oynasidan foydalanish.

**ColorDialog** muloqot oynasining xususiyatlari, usullari va hodipsalarini masalaning ahamiyatiga qarab ishlatish mumkin.

**2. FontDialog muloqot oynasi.** Bu oyna – foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi. Bu komponentdani formaga o‘rnatilganda hech qanday ko‘rinish hosil bo‘lmaydi, ammo formaning ichki tuzilmasiga qo‘yshiladi. Fomraning ishchi holatidagi formasining pastki qismida uning ob‘yekti yaratiladi.

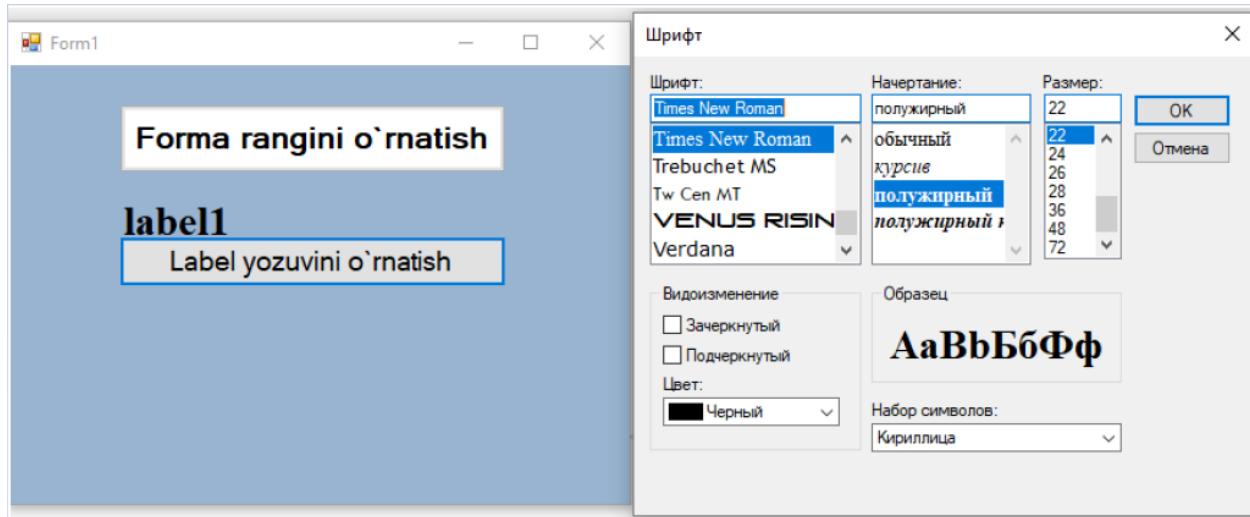
Bu muloqot oynasini ishlatish uchun formaga bir **Label** va **button** ob‘yektlarini o‘rnatamiz. Tugma bosilganda Label ob‘yektining matnini yozuv xususiyatlarini o‘rnatishni ko‘rib chiqamiz. Buning uchun tunmaning bosilganda xususiyatiga quyidagi dastur fragmentni yozish yetarli.

```

fontDialog1->ShowColor = true;
fontDialog1->Font = label1->Font;
fontDialog1->Color = label1->ForeColor;
Color color = label1->ForeColor;
System::Drawing::Font^ font = label1->Font;
System::Windows::Forms::DialogResult result = fontDialog1->ShowDialog();
if(result == ::System::Windows::Forms::DialogResult::OK) {
    label1->Font = fontDialog1->Font;
    label1->ForeColor = fontDialog1->Color;
}

```

Dasturda dastlab **Label** ob‘yektning rang va yozuvlarini saqlab olinadi, chunki yozuvni formatlash muloqot oynasi chaqirilganda joriy holatni olish uchun. Dasturda rang va yozuv qiymatlarini saqlash uchun o‘zgaruvchilarni aniqlab olish ham keltirilgan. Muloqot oynasining [OK] hodisasi bajarilganda yozuv va rangni o‘zgartirish ko‘rsatilgan.



5.5.3.-rasm. *FontDialog* muloqot oynasidan foydalanish.

Bu sinfning xususiyailari, usullari va hodisalari mavjud. Ular amaliy vazifalarni bajarishda foydalanish mumkin va mustaqil o‘rganish lozim. Chunki bir vazifani amalga oshirish uchun turli xil algoritmlardan foydalanish mumkin.

**3. OpenFileDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun faylni tanlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bu muloqot oynasi ham yuqoridagidek foydalaniladi. Shuningdek, sinfning mos xususiyatlari, usullari va hodisalari mavjud. Farqli xususiyatlarni ko‘rsatish uchun bir misol olamiz. Unda tugma bosilganda **label** matniga faylning to‘liq yo‘lini olish va o‘rnatish uchun quyidagi dastur fragmentini yoziladi.

```
openFileDialog1->InitialDirectory = "c:\\";
openFileDialog1->Filter = "txt files (*.txt)/*txt/doc files
(*.doc)/*.doc, *.docx/All files
(*:*)/*:*";
openFileDialog1->FilterIndex = 2;
```

```

openFileDialog1->RestoreDirectory = true;
openFileDialog1->Title = "Fayl nomini olish";
if( openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK ){
if(openFileDialog1->OpenFile() != nullptr ){
label1->Text = openFileDialog1->FileName;
}
}

```

Dasturning birinchi satri joriy katalog o‘rnatiladi. Shuningdek, joriy foydalanuvchining kerakli kataloglarini ham o‘rnatish qiymatlari bor. So‘ng, fayllarni tipi bo‘yicha filrlash o‘rantiladi. Hozirda 3 ta format o‘rnatilgan. Keyingi satrda fayl kengaytmalaridan qaysi biri joriy bo‘lib chiqishini belgilash amalga oshirilgan. Ko‘rsatilgan katalogni faollashtirish bajarilgan va muloqot oynasining sarlavhasida kerakli matn joylashtirilgan. Muloqot oynaning teskari aloqasi tekshirilgan va shu asosida amal bajarilgan.

Odatda bu muloqot oynasi ma‘lum tizimda fayllarni matnini joylashtirish uchun ishlataladi. Buni qanday amalga oshirish mumkin. Buning uchun kichik bo‘lsa ham matn muharriri yaratish lozim va unga OTdan oddiy manli fayllarni yuklab olish mumkin.

**4. PrintDialog muloqot oynasi.** Bu oyna - foydalanuvchilarga printerni tanlash va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi. Oynaning juda ko‘p xususiyatlari, usullari va hodisalari bor. Bularni mustaqil ishlarni bajarish vaqtida amalga oshirish mumkin. Shuningdek, bu muloqot oynani ishlatish usullari ham juda ko‘p. Odatda dasturchining loyiha holatidan kelib chiqqan holda foydalaniladi.

```

printDialog1->AllowSomePages =
true; printDialog1->ShowHelp = true; if( printDialog1== nullptr )
System::Windows::Forms::MessageBox::Show("pnull");
System::Windows::Forms::DialogResult result=printDialog1-> ShowDialog();

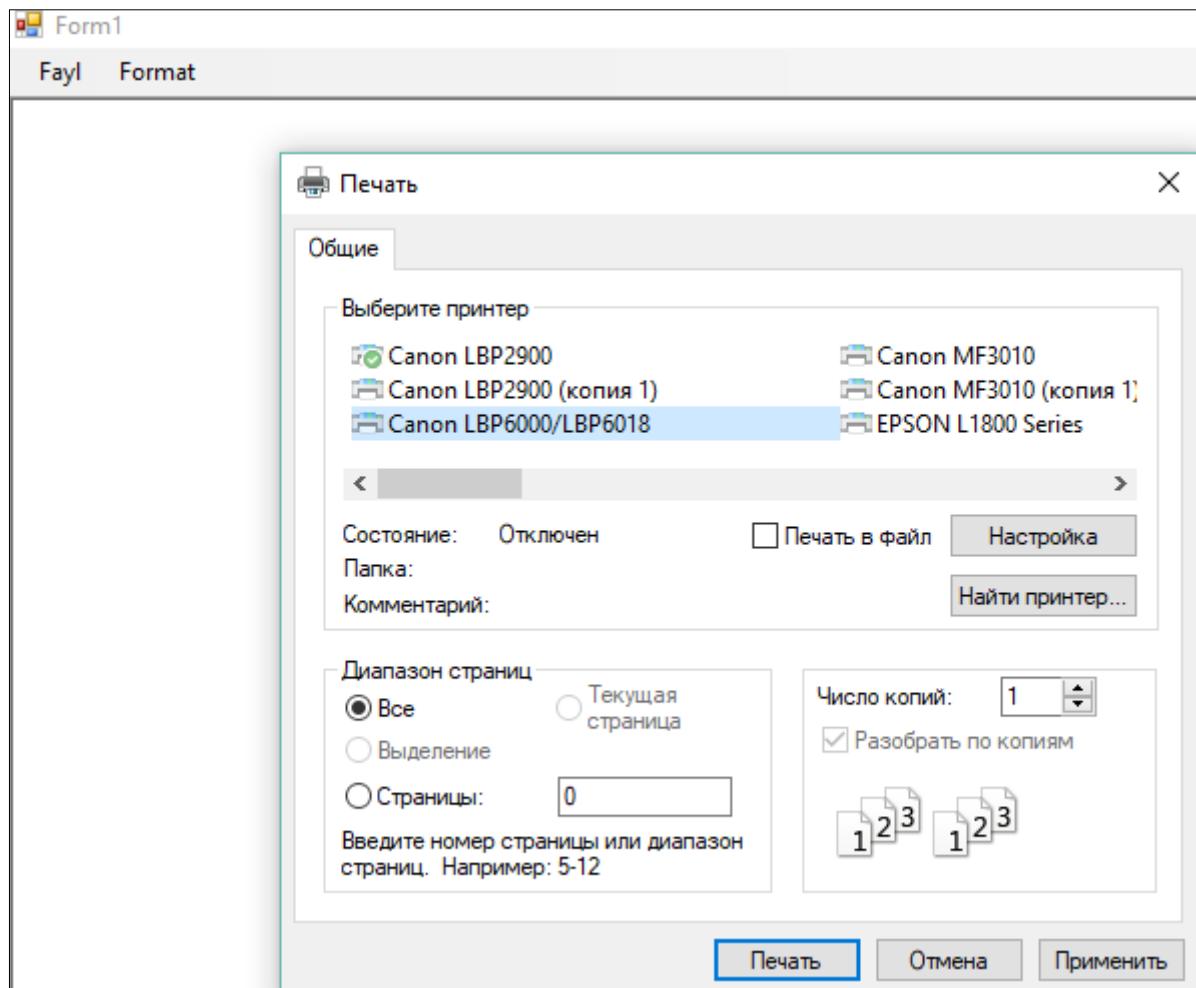
```

```

System:::Windows:::Forms:::MessageBox:::
Show(result.ToString());
if( result == System:::Windows:::Forms:::DialogResult:::OK )
{
// docToPrint->Print(); System:::Windows:::MessageBox:::Show("Chop qilish
boshlanadi");
}

```

Dastur fragmentida ma'lumotlarni chop qilish uchun avval uni ma'lum bir chiquvchi oqimga yozish va oqimni esa, hujjat formatiga joylashtirish kerak. Hujjatni esa, sahifalarni sozlab **docToPrint** ob'yekt yaratilishi kerak. Ammo chop qillishning turli parametrlarini ishlatish uchun muloqot oynasini chiqarib beradi. Kerakli xususiyatlarni o'rnatgandan so'ng amalni bajarish mumkin.

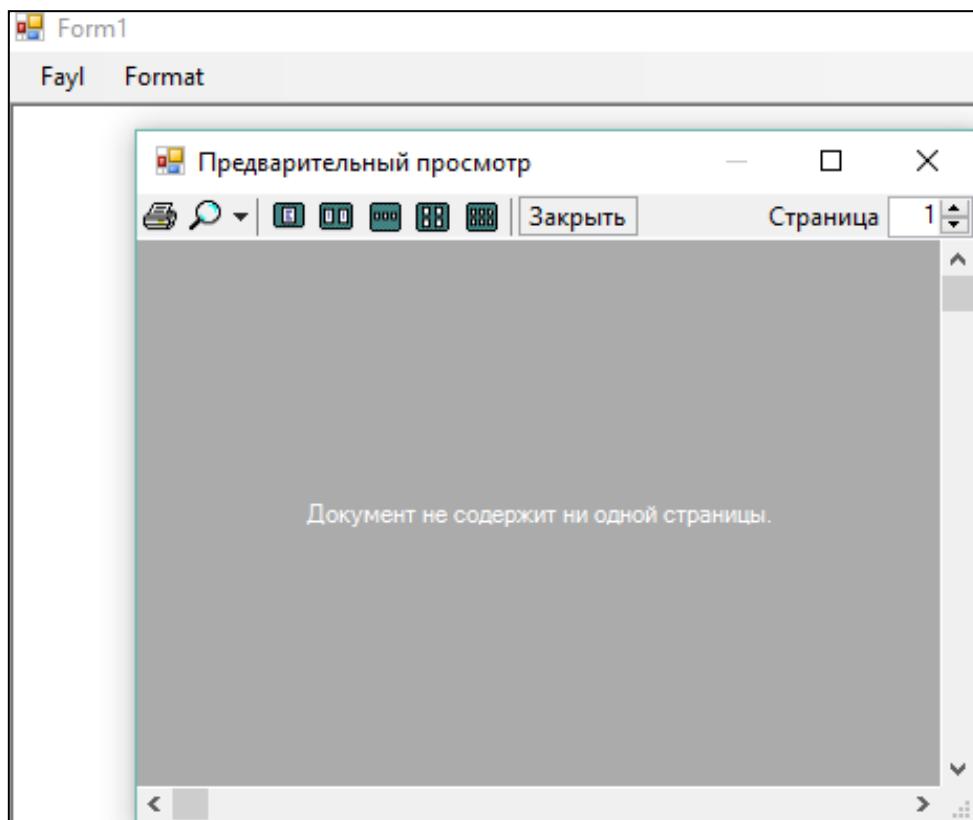


5.5.4.-rasm. PrintDialog muloqot oynasidan foydalanish.

**5. PrintPreviewDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun chop qilishda **PrintDocument** boshqaruv elementining ko‘rinishining ko‘rsatish imkonini beradigan muloqot oynasini ko‘rsatadi.

```
printPreviewDialog1->MinimumSize = System::Drawing::Size(375,250);  
printPreviewDialog1->UseAntiAlias=true;  
printPreviewDialog1->Document = document;  
printPreviewDialog1->ShowDialog();
```

Bu muloqot oynasi ham yuqoridagi muloqot oynasi kabi sozlashlarni bajarshgandan so‘ng foydalanish mumkin. Dastur fragmentida qarasangiz document ob‘yektni yaratish lozim. Chop qilishninig ob‘yektidan farq qilgan xolda oqimdagи ma‘lumotni formatlash va uni chiqishini A4 shaklga keltrish orqali amlga oshiriladi.



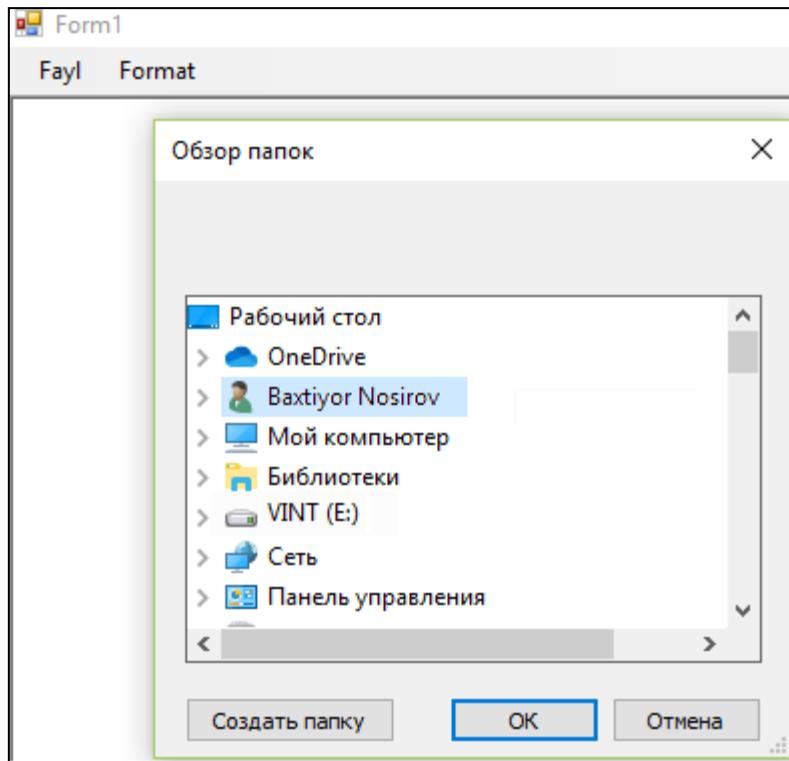
5.5.5.-rasm. *PrintPreviewDialog* muloqot oynasidan foydalanish.

**6. FolderBrowserDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bundan ma‘lumotlarni ko‘chirishda, papkalarni taqqoslashda ishlatalish mumkin. Muloqot oynasining xususiyatlari, usullari va hodisalar mavjud. Ularning ba‘zilarini quyidagi dastur fragmentiga keltirib o‘tamiz.

```
System::IO::Stream^myStream;  
  
System::Windows::Forms::DialogResult result=folderBrowserDialog1->  
ShowDialog();  
  
if(result==System::Windows::Forms::DialogResult::OK)  
{System::String^folderName=folderBrowserDialog1->SelectedPath;  
openFileDialog1->InitialDirectory=folderName;  
openFileDialog1->FileName=String::Concat(folderName,"\\1.cpp");  
if((myStream=openFileDialog1->OpenFile())!=nullptr)  
{System::IO::StreamReader ^ sr=gcnew System::IO::  
StreamReader(openFileDialog1->FileName);  
richTextBox1->Text=sr->ReadToEnd(); sr->Close();  
}  
}
```

Dastur fragmentida papka uchun muloqot oynasi chaqirilgan va ko‘rsatilgan papkadan **1.cpp** faylini yuklab kelgan. Bunday holat fayl menejerlar uchun papkadagi fayllarning ro‘yhatini ham olish mumkin.

Dastur fragmentida bir **StreamReader** oqim yaratilgan. Oqim faylning ma‘lumotlarini o‘qish uchun yaratilgan. Oqimning konstruktori asosida **sr** oqim ob‘yekti yaratiladi. Bu oqimning **ReadToEnd** funksiyasi orqali **richTextBox** ga ma‘lumotlar joylashtiriladi. **FolderBrowserDialog** muloqot oynasining ko‘rinishi quyidagicha:



5.5.6.-rasm. *FolderBrowserDialog* muloqot oynasidan foydalanish.

**7. SaveFileDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun faylni saqlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bu ham asosan matnli va maxsus tuzilmalar ma‘lumotlarini saqlash uchun ishlatiladi. Ma‘lumot qanaqa tuzilmada yozilsa, shunday tuzilmada o‘qiladi.

*System::IO::Stream^ myStream;*

```
MemoryStream^ userInput = gcnew MemoryStream();
richTextBox1->SaveFile( userInput, RichTextBoxStreamType::PlainText );
userInput->WriteByte( 32 );
// saveFileDialog1->CreatePrompt = true;
// saveFileDialog1->OverwritePrompt = true;
// saveFileDialog1->FileName = "myText";
// saveFileDialog1->DefaultExt = "txt";
saveFileDialog1->Filter = "txt files (*.txt)/*.*";
saveFileDialog1->FilterIndex = 2;
saveFileDialog1->RestoreDirectory = true;
```

```

if( saveFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK ){
myStream = saveFileDialog1->OpenFile();
userInput->Position = 0;
userInput->WriteTo( myStream );
myStream->Close();
}

```

Dastur fragmentida ikki xil oqim yaratilgan. **myStream** - birinchisi an‘anaviy oqim va MemoryStream -xotirali oqim yaratilgan. Biri ma‘lumotlarni olish va ikkinchisi ma‘lumotlarni xotiraga yozish, berilgan nom bilan nomlashgan qaratilgan. **Position** bu poizitsiyani boshlash, ammo olib tashlasa ham ishlaydi. Bu muloqot oynasi nafaqat matnli ma‘lumotlarni balki, foydlanuvchining xoxlagan ma‘lumotni matn yoki binar kodlash orqali saqlash mumkin.

Ko‘rib chiqilgan muloqot oynalari barchasi OT bilan ishlashga mo‘ljallanganligi ko‘rinib turibdi. OT qanday sozlangan bo‘lsa, bu muloqot oynalari ham shu rejimda ishlaydi. Oynalarga deyarli o‘zgartirish kiritish shart emas, xuddiki barchasi kutilgandek yaratilganga o‘xshaydi. Faqat eng katta muammosi lokalizatsiya qilish.

### **3. Muloqot oynalarini boshqarish elementlari.**

Muloqot oynalarini boshqarish elementlari deganda interaktiv muloqot qilish oynalari tushiniladi. Ularni yaratish va boshqarish dasturchining xoxishiga qarab amalga oshiriladi. Yuqorida aytib o‘tganimizdek, Visual C++ da **MessageBox** sinfi haqida aytib o‘tgan edik. Unda 21 ta turli kombinatsiyali **show** funksiyasi borligini ham ko‘rib chiqdik. Shuning uchun bu funksiyalar va ularning parametrlari hamda parametrlarining qiymatlari to‘g‘risi ma‘lumotlarni va muloqot oynalarini yaratishga va ishlov berishga e‘tiborni qaratamiz.

**1. Show(String) funksiysi.** Bu bir argumentli funksiya bo‘lib, belgilangan matn bilan xabar ko‘rsatadigan muloqot oynasini yaratish uchun ishlatiladi. Uning

kiruvchi parametri *System::String* tipida bo‘lib,

*System::Windows::Forms::DialogResult* tipidagi qiymat qaytaradi.

```
if(MessageBox::Show("Bu oddiy show") == System::Windows::
```

```
Forms::DialogResult::OK)
```

```
this->Close();
```

**2. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String)** funksiyasi. Bu ko‘p paramerli bo‘lib, belgilangan parametlar asosida interaktiv muloqot oynasini yaratish uchun ishlatiladi. Parametlari quyidagi qiymatlari qabal qiladi.

*IWin32Window Interface* - *System.Windows.Forms* nomlar fazosi va *System.Windows.Forms.dll* kutubxonasidan foydalanib, *Win32 HWND* ni joriy qilish interfeysini beradi. *IWin32Window* sinf interfeysidan merosxo‘r oladi. *System.Windows.Forms.Control* va *System.Windows.Forms.NativeWindow* umumiy ruxsat sinflari interfeyslarini ishlatadi, qiymatlar sifatida *ComVisibleAttribute*, *GuidAttribute*, *InterfaceTypeAttribute* tiplarini ishlatadi.

**MessageBoxButtons** - *Enum* tipidagi parametrdir. *System.Windows.Forms* nomlar fazosi va *System.Windows.Forms.dll* kutubxonasidan foydalanadi. Muloqot oynalarida ko‘rsatilishi kerak bo‘lgan tugmalarni aniqlaydi. Qiymatlari *public enum class MessageBoxButtons* ga ta‘luqlidir.

#### 5.12-jadval.MessageBoxButtons qabul qiluvchi qiymatlar

Nº	Qiymat nomi	Vazifasi
1	AbortRetryIgnore	Muloqot oynasida Abort, Retry Ignore tugmalarini o‘rnatish
2	OK	Muloqot oynasida OK tugmasini o‘rnatish
3	OKCancel	Muloqot oynasida OK va Cancel tugmalarini o‘rnatish

№	Qiymat nomi	Vazifasi
4	RetryCancel	Muloqot oynasida Retry va Cancel tugmalarini o‘rnatish
5	YesNo	Muloqot oynasida Yes va No tugmalarini o‘rnatish
6	YesNoCancel	Muloqot oynasida Yes, No va Cancel tugmalarini o‘rnatish

Muloqot oynasiga **MessageBoxButtons** tugmalarini o‘rnatish dastur fragmenti:

```
if ((MessageBox::Show("Joriy oynani yopishni xoxlaysizmi?", "Xabar",
MessageBoxButtons::YesNo)==
System::Windows::Forms::DialogResult::Yes)){
this->Close();
```

**MessageBoxIcon** – **Enum** tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida ko‘rsatish uchun muloqot oynalarining turlarini belgilovchi konstantalarni aniqlaydi.

5.13-jadval. MessageBoxIcon qabul qiluvchi qiymatlar

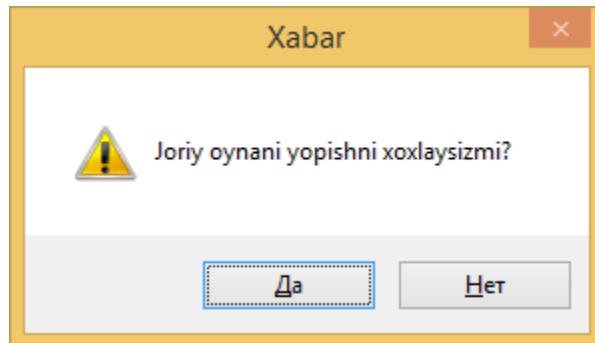
№	Qiymat nomi	Belgisi	Vazifasi
1	Asterisk		Qandaydir hodisa haqida faqat xabar beruvchi muloqot oynasi uchun ikonka
2	Error		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka
3	Exclamation		Qandaydir hodisa haqida faqat ogohlantirish haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka

Nº	Qiymat nomi	Belgisi	Vazifasi
4	Hand		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma'lumot beruvchi muloqot oynasi uchun ikonka
5	Information		Qandaydir hodisa haqida faqat xabar beruvchi muloqot oynasi uchun ikonka
6	None		Ikonkasiz muloqot oynasi uchun
7	Question		Qandaydir hodisa haqida faqat savolga javob oluvchi muloqot oynasi uchun ikonka
8	Stop		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma'lumot beruvchi muloqot oynasi uchun ikonka
9	Warning		Qandaydir hodisa haqida faqat ogohlantirish haqida ma'lumot beruvchi muloqot oynasi uchun ikonka

**MessageBoxIcon** ni ishlatish uchun joriy formani yopish uchun muloqot oynani ishlatish uchun dastur fragmentini keltiramiz. Buning uchun formaning FormClosing hodisasiga quyidagi dastur fragmentini yozamiz:

```
System::Void Form1_FormClosing(System::Object^sender,
System::Windows::Forms::FormClosingEventArgs^e)
{if ((MessageBox::Show("Joriy oynani yopishni xoxlaysizmi?", "Xabar",
MessageBoxButtons::YesNo,
MessageBoxIcon::Exclamation)==System::Windows::Forms::
DialogResult::No))
{e->Cancel=true;
}
```

Dastur fragmentida hodisaning to‘liq yozilishi keltirilgaganligining sababi unda hodisaga ishlov berilgan.



5.5.7.-rasm. *MessageBoxIcon* bilan muloqot oynasini yaratish.

Mazkur oynada yo‘q tugmasi bosilsa, forma yopilmaydi va aksincha ha tugmasi bosilsa, forma yopiladi.

**MessageBoxDefaultButton** - *Enum* tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida standart tugmalarni o‘rnatuvchi o‘zgarmaslarni belgilaydi.

5.14-jadval. MessageBoxDefaultButton qabul qiluvchi qiymatlar

<b>№</b>	<b>Qiymat nomi</b>	<b>Vazifasi</b>
1	Button1	Muloqot oynasida starndart birinchi tugmani joriy tugma sifatida o‘rnatish
2	Button2	Muloqot oynasida starndart ikkinchi tugmani joriy tugma sifatida o‘rnatish
3	Button3	Muloqot oynasida starndart uchinchi tugmani joriy tugma sifatida o‘rnatish

Tugma bosilganda formani yopishni so‘rash muloqot oynasini yaratish dastur fragmentini keltiramiz.

```
String^message="Joriy oynani yopishni xoxlaysizmi?";
```

```
String^caption="Xabar";
```

```
MessageBoxButtons buttons=MessageBoxButtons::YesNo;
```

```

System::Windows::Forms::DialogResult result;
System::Windows::Forms::DialogResult mayli=
System::Windows::Forms::DialogResult::Yes;
result=MessageBox::Show(this, message, caption, buttons,
MessageBoxIcon::Question, MessageBoxDefaultButton::Button2);
if(result==mayli)

```

Odatda tugmalarning farqini aniqlash ancha murakkab, agar OT maska bo‘lsa, farqlash mumkin.

**MessageBoxOptions - Enum** tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida standart xususiyatlarni o‘rnatish uchun foydalaniladi.

#### 5.15-Jadval . MessageBoxOptions

<b>№</b>	<b>Qiymat nomi</b>	<b>Vazifasi</b>
1	DefaultDesktopOnly	Oddiy muloqot oynalariga o‘xshaydi ammo faol ishchi maydonda ko‘rsatiladi
2	RightAlign	Muloqot onynasining ma‘lumotini to‘g‘ri joylashtirish
3	RtlReading	Muloqot oynasida ma‘lumotni o‘ngdan chapga qarab o‘qish uchun moslab joylashtirish
4	ServiceNotification	Oddiy muloqot oynalariga o‘xshaydi ammo faol ishchi maydonda ko‘rsatiladi va javob bermasdan o‘tib ketish mumkin xususiyati mavjud

Tugma bosilganda formani yopishni so‘rash muloqot oynasini yaratish dastur fragmentini keltiramiz.

```
String^message = "Joriy oynani yopishni xoxlaysizmi?";
```

```
String^caption = "Xabar";
```

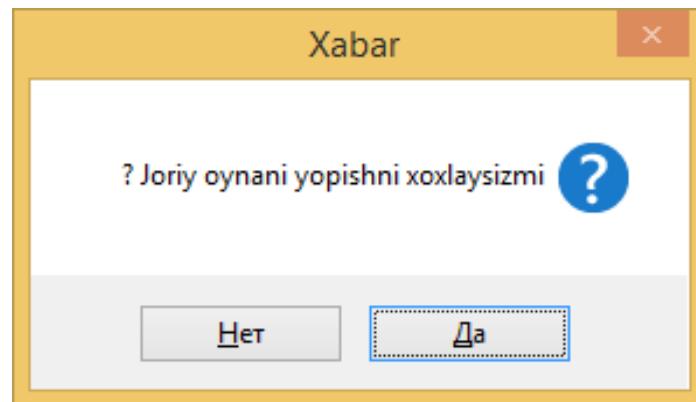
```
MessageBoxButtons buttons = MessageBoxButtons::YesNo;
```

```

System::Windows::Forms::DialogResult result;
System::Windows::Forms::DialogResult mayli=
System::Windows::Forms::DialogResult::Yes;
result = MessageBox::Show( this, message, caption, buttons,
MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
MessageBoxOptions::RtlReading);
if ( result == mayli )
{
this->Close();

```

Dastur fragmentini ishlatib ko‘rib, natijasini tahlil qilish va nimalarni o‘ngdan chapga qarab o‘qish o‘zgarganini ko‘rish mumkin.



5.5.8.-rasm. *MessageBoxOptions* bilan muloqot oynasini yaratish.

Barcha muloqot oynalari kabi yuqorida keltirilgan dastur fragmentlaridan ko‘rinib turibdiki, **DialogResult** tipini qaytaradi.

**3. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String) funksiyasi.** Bu funksiyaga yordam fayli va ichki indeks fayllarga ko‘rsatkich yaratadi.

```
String^ message = "Joriy oynani yopishni xoxlaysizmi?";
```

```
String^ caption = "Xabar";
```

```
MessageBoxButtons buttons = MessageBoxButtons::YesNo;
```

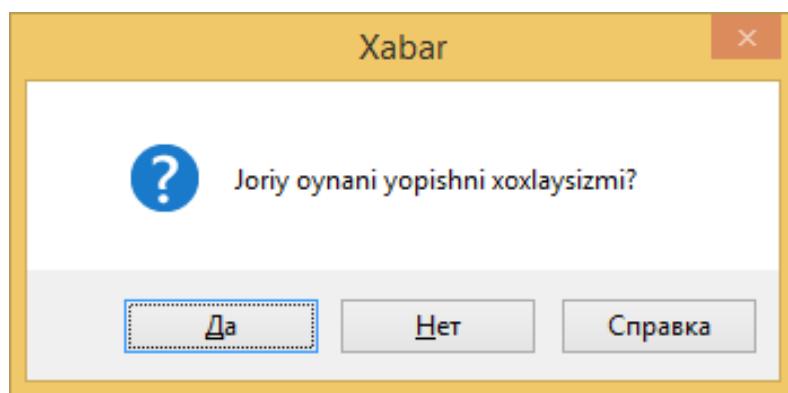
```
System::Windows::Forms::DialogResult result;
```

```

System::Windows::Forms::DialogResult mayli =
System::Windows::Forms::DialogResult::Yes;
result = MessageBox::Show( this, message, caption, buttons,
MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
MessageBoxOptions::RightAlign, "d:\tut.chm",
HelpNavigator::KeywordIndex, "iv");
if (result==mayli)

```

Dastur fragmentini ishlatib ko‘rib, natijasini tahlil qilish va nimalarni o‘zgarganini ko‘rish mumkin.



5.5.9.-rasm. Yordam tugmali muloqot oynasini yaratish.

Dastur fragmentida e‘tibor bilan qarasangiz, MessageBoxButtons::YesNo tshlatilgan, ammo, muloqot oynasida 3 ta tugma chiqqan, demak yordam tugmasining faol holatga kelganining ko‘rish mumkin.

Show usulining boshqa yangi parametrlarni qabul qilmaganligi uchun ularni yozilishi va dastur fragmentlaridan keltiramiz.

**4. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String)**  
`result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1, (MessageBoxOptions)0, "d:\tut.chm", HelpNavigator::KeywordIndex, "iv");`

**5. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator)**

*System::Windows::Forms::DialogResult r3 = MessageBox::Show( message, caption, MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1, (MessageBoxOptions)0, "mspaint.chm", HelpNavigator::Index );*

**6. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String)**

*System::Windows::Forms::DialogResult r7 = MessageBox::Show(message, caption,*

*MessageBoxButtons::OK, MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1,*

*(MessageBoxOptions)0, "mspaint.chm", "mspaint.chm::/paint\_brush.htm" );*

**7. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions)**

*result = MessageBox::Show( this, message, caption, buttons,*

*MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );*

**8. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String)**

*System::Windows::Forms::DialogResult r1 = MessageBox::Show(" message, caption,*

*MessageBoxButtons::OK, MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1,*

*(MessageBoxOptions)0, "mspaint.chm" );*

**9. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, Boolean)**

*System::Windows::Forms::DialogResult MessageBox::Show(message, caption,*

*MessageBoxButtons::OK, MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1,*

*(MessageBoxOptions)0, true );*

Show(IWin32Window,     String,     String,     MessageBoxButtons,  
    MessageBoxIcon,    MessageBoxDefaultButton,    MessageBoxOptions,   String,  
    HelpNavigator)

*System::Windows::Forms::DialogResult r4 = MessageBox::Show( this,  
message, caption,*

*MessageBoxButtons::OK, MessageBoxIcon::Question,  
MessageBoxDefaultButton::Button1,  
(MessageBoxOptions)0, "mspaint.chm", HelpNavigator::Index );*

**10. Show(IWin32Window, String, String, MessageBoxButtons,  
MessageBoxIcon, MessageBoxDefaultButton)**

*result = MessageBox::Show( this, message, caption, buttons,  
MessageBoxIcon::Question,  
MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );*

**11. Show(IWin32Window, String, String, MessageBoxButtons,  
MessageBoxIcon)**

*result = MessageBox::Show( this, message, caption, buttons,  
MessageBoxIcon::Question );*

**12. Show(String, String, MessageBoxButtons, MessageBoxIcon,  
MessageBoxDefaultButton)**

*result = MessageBox::Show( this, message, caption, buttons,  
MessageBoxIcon::Question,  
MessageBoxDefaultButton::Button1 );*

**13. Show(IWin32Window, String, String, MessageBoxButtons)**  
*result = MessageBox::Show( this, message, caption, buttons);*

**14. Show(String, String, MessageBoxButtons, MessageBoxIcon)**  
*result = MessageBox::Show(message, caption, buttons,  
MessageBoxIcon::Question);*

**15. Show(IWin32Window, String, String)**  
*result = MessageBox::Show( this, message, caption);*

## **16. Show(IWin32Window, String)**

*result = MessageBox::Show( this, message);*

## **17. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions)**

*result = MessageBox::Show( this, message, caption, buttons,*

*MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );*

## **18. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator, Object)**

*System::Windows::Forms::DialogResult r6 = MessageBox::Show(this,*

*message, caption,*

*MessageBoxButtons::OK, MessageBoxIcon::Question,*

*MessageBoxDefaultButton::Button1,*

*(MessageBoxOptions)0, "mspaint.chm", HelpNavigator::KeywordIndex, "ovals"*

*);*

Bu usullarni .NET 5 Preview 1, .NET Core 3.1 3.0 va .NET Framework 4.8, 4.7, 4.6, 4.5, 4.0, 3.5, 3.0, 2.0, 1.1 kabi varintlarda qo'llab quvatlanadi.

## **4. Muloqot oynalarini yaratish.**

Foydalanuvchi tomonidan muloqot oynalarini yaratish oyna formasiga ishlov berish asosida amalga oshiriladi. Forma oynasiga muloqot oynasini o'rnatish uchun quyidagi qadamlari bajarish lozim.

**1- qadam.** Menyudan foydlanib, [menu] → [project] → [addClass] → [CLR] → [Windows Form] buyruqlar ketma ketligi asosida yangi forma qo'shiladi.

Yoki, [menu] → [project] → [add New item] → [UI] → [Windows Form] bilan ham bajarsa bo'ladi.

**2- qadam.** Yaratilgan yangi forma oynasigi o'tib, formaga quyidagicha ishlov beriladi. [Autosize] xususiyatning qiymatini [true] ga, [StartPosition]

xususiyatining qiymatini [CenterParent]ga, [FormBorderStyle] xususiyati qiymatiga [none], [FixedDialog], [FixedToolWindows], [SizerableToolWindows]larning birini o‘rnatish mumkin.

**3- qadam.** Forma oynasini o‘zinigz xoxlagandek loyihalashingiz mumkin. Masalan, bir **Label**, **InputBox** va bir **Button** joylashtiramiz, ularni ham kerakli xususiyatlari o‘rnatamiz.

**4- qadam.** Asosiy formaga o‘tib, unga **#include "MyForm.h"** sarlavha faylni qo‘shamiz. Bu muloqot oynasi uchun yaratilgan forma oynasi bilan ishlash uchun kerak.

**5- Qadam.** Asosiy formaga bir **Label** va bir **Button** joylashtiramiz. Tugmaning **Click** hodisasida muloqot oynasini chaqirish dastur fragmentini yozamiz.

```
// MyForm myForm;  
// myForm.Show();  
// myForm.ShowDialog();  
MyForm^myFormWith=gcnew MyForm();  
// myFormWith->Show();  
myFormWith->ShowDialog();
```

Dastur fragmentida izohga olib qo‘yilagan yordami ham muloqot oynalarni yaratish va chaqirish mumkin. Ammo ularni farqlari mavjud. Shuning uchun ularni hammasini shu dastur fragmentida keltirdik. Birinchi qatorda formani **MyForm** oddiy ob‘yekt sifatida yaratilgan. Uning birinchi **Show( )** usuli hisoblanib, bu usul bilan muloqot oynasini chaqirish mantiqan xato, chuniki tizim oynani yaratadi va ustunlikni asosiy formaga beradi. Bu holda yaratilgan oyna bir lahzaga ko‘rinadi xolos.

Ikkinchi usuli bu **ShowDialog( )** usul yaxshi yondashuvlardan bo‘lib, muloqot formasiga o‘rnatilgan barcha xususiyatlarga rioya qiladi hamda foydalanish mumkin. Ikkinchi **MyForm()** konstruktordan foydalanib, **gcnew** operatori asosida yaratilgan, uning birinchi fuknsiya **Show( )** ham oldingisiga

o‘xhash bo‘lib, ammo ustunlikni foydalanuvchining o‘ziga qo‘yib beradi, ya‘ni foydalanuvchi muloqot oynaga javob bermasdan turib, asosiy forma oynasiga o‘tishi mumkin. Ikkinchisi esa bu **ShowDialog()** usul yaxshi yondashuvlardan bo‘lib, muloqot formasiga o‘rnatilgan barcha xususiyatlarga rioya qiladi hamda foydalanish mumkin.

**6- qadam.** Muloqot oynasini tugmasida quyidagicha algoritmni yozamiz. Muloqot oynasiga kiritilgan ma‘lumotni asosiy formaga olib o‘tish uchun avval **public: System::String^email;** kabi bir o‘zgaruvchi yaratib olamiz.

```
email = textBox1->Text;
```

```
this->Close();
```

**7- qadam.** Asosiy formaning tugmasining hodisasiga yozilgan dastur fragmenti davomidan quyidagini qo‘shib qshyamiz.

```
label1->Text = myFormWith->email;
```

**8- qadam.** Loyihani ishlatib yaratilgan, yaratilgan muloqot oynani ishlashini ko‘rish mumkin.

Muloqot oynani yuqori darajada yaratish uchun unga yangi konstruktor yozish ham mumkin. Uni quyidagicha amalga oshiriladi.

```
public: MyForm(System::String^title)
```

```
{
```

```
InitializeComponent();
```

```
_title = title;
```

```
}
```

```
public: System::String^ _title;
```

```
//...
```

```
private: System::Void MyForm_Load(System::Object^ sender,
```

```
System::EventArgs^ e) {
```

```
this->Text = _title;
```

```
}
```

```
// Asosiy formada esa
```

```
MyForm^ myFormWith = gcnew MyForm("Xabar");  
myFormWith->ShowDialog();  
label1->Text = myFormWith->email;
```

Bunday imkoniyat bilan foydalanuvchi uchun ixtiyoriy muloqot oynasini yaratish mumkin. Muloqot oynalaridan foydalanish dastur foydalanuvchilariga ko‘plab qo‘layliklar yaratib beradi. Har bir dasturda bir asosiy oynaga kamida funksional imkoniyatiga qarab 4-5 ta muloqot oynalari yaratish mumkin.

### **Nazorat savollari**

2. Muloqot oynada nimalar aniq ko‘rsatilishi kerak?
3. [Dialogs] tab oynasida joylashgan muloqot komponentalarini sanab bering.
4. ColorDialog komponentasining vazifasini ayting.
5. Qaysi komponenta foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi?
6. Interaktiv xabarlarni berish va aniq javoblarni olish Visual C++ da qanday sinfi mavjud?
7. MessageBox sinfnining show funksiyasi parametrlarini sanab bering.
8. MessageBoxOptions qanday ob‘yekt tipi va nima vazifani bajaradi?
9. Foydalanuvchi tomonidan yaratiladigan muloqot oynasi qanday yaratiladi?
10. ColorDialog muloqot oynasining asosiy vazifasi va ob‘yekti qanday yaratiladi?
11. ColorDialog muloqot oynasidan foydalanish uchun qanday konstruktorni ishga tushirish lozim?
12. Muloqot oynasi ochilganda maxsus ranglarni yaratish uchun ishlatiladigan boshqaruv elementlari ko‘rinib turishini ko‘rsatuvchi qiymatni oladigan yoki o‘rnatadigan xususiyat nomini ayting.
13. Dispose() bu qanday usul?

14. Foydalanuvchi umumiy muloqot oynasidagi yordam tugmasini bosganda sodir bo‘ladigan usul nomini ayting.
15. PrintPreviewDialog muloqot oynasi qaysi tabda joylashgan va vazifasini aniq aytib bering.
16. Win32Window Interface nima uchun ishlataladi?
17. Muloqot oynasida OK va Cancel tugmalarini o‘rnatish qayday amalga oshiriladi?
18. MessageBoxIcon muloqot oynalarida nima uchun ishlataladi?
19. Muloqot oynasida standart tugmalarni o‘rnatuvchi o‘zgarmaslarini belgilaydigan parametrni ayting.
20. Asosiy formaga yangi forma qanday qo‘shiladi?
21. ShowDialog() va Show() usullarining farqini tushuntirib bering.

## **FOYDALANILGAN ADABIYOTLAR RO‘YXATI:**

1. Mo‘minov B.B., Dasturlash 1 (Darslik).-T. “Nihol print” ok, 2021, 280 b.
2. Mo‘minov B.B., Dasturlash II (Darslik)-T. “Nihol print”, 2021, 604 b.
3. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. “C va C++ tili.” Voris-nashriyot MCHJ, Toshkent 2013, 488 b.
4. Nazirov Sh. C++ tilida dasturlash asoslari.-T.:2006.
5. Abdullayeva Z. Sh., Ishniyazov O.O. Dasturlash I va Dasturlash II fanidan o‘quv qo‘llanma. 2022.- 141 b.
6. J.Axmadaliyev, R.Xoldorboyev C++ dasturlash tilini o‘rganish bo‘yicha uslubiy qo‘llanma (2015).
7. K.B.Ablaqulov Ma’lumotlar tuzulmasi va algoritmlari. O‘quv qo‘llanma “Big marko world” nashriyoti. Qarshi. 258 bet.
8. Прохоренок Н.А. Программирование на C++ в Visual Studio 2012 Express(2018).
9. Макурин Ю.Д. Сивохин А.В. Проектирование и реализация баз данных и клиентских приложений в среде MS Visual Studio.NET (2016).
10. Xolmatov T.X., Taylaqov N.I. “Amaliy matematika, dasturlash va kompyuterning dasturiy ta’minoti” T. Mehnat -2000. 304 b.
11. Abdiqodirov A. “Turbo C++ tilida programmalash” 1999 y.
12. ZiyoNET.uz – O‘zbekistan Respublikasi axborot-ta’lim portali.
13. acm.tuit.uz – Dasturiy yechim to‘g‘riligini avtomatik testlovchi tizim.
14. <https://uzbekcoders.uz> – Bir million dasturchi loyihasi
15. <http://wwwcplusplus.com/>
16. lib.tuit.uz – TATU elektron kutubxonasi
17. acm.timus.ru – dasturlarni testlovchi tizim;
18. [www.dasturchi.uz](http://www.dasturchi.uz) - dasturlash tillarini o‘rgatuvchi veb sayt.
19. [moodle.tuitkf.uz](http://moodle.tuitkf.uz). – TATU Qarshi filiali vertuai ta’lim tizimi.
20. [hemis.tuitkf.uz](http://hemis.tuitkf.uz)– TATU Qarshi filiali masofaviy ta’lim boshqarish tizimi.

## O'QUV USLUBIY NASHR

**L.N. XUDOYOROV, SH.R. DAVRONOV, B.N. NOSIROV  
N.F. AXMEDOVA, SH.M. SAMANDAROVA**

# DASTURLASH

## Darslik

Nashriyot litsenziyasi 062153. 07.02.2023 yil.  
Terishga 09.06.2025 yilda berildi.  
Bosishga 30.06.2025 yilda ruxsat etildi.  
Bichimi 64x84 1/16. Shartli bosma tabog'i 17.5.  
Ofset usulida chop etildi. Times New Roman garniturasi.  
50 nusxada. Buyurtma 111-BMW/25. Hajmi 280 bet. Erkin narxda.

“BIG MAKRO WORLD” MCHJ nashriyoti, 180100.  
Qarshi shahri, Ko‘chabog‘ ko‘chasi, 17-uy

“BIG MAKRO WORLD” MCHJ bosmaxonasida bosildi.  
Qarshi shahri, Beglar MFY, Ko‘chabog‘ ko‘chasi, 2/49-uy  
Tel./faks: +998908680001





