O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

JIZZAX POLITEXNIKA INSTITUTI
“INFORMATSION TEXNOLOGIYALAR”

kafedrasi

“TEXNIK TIZIMLARDA AXBOROT TEXNOLOGIYALARI”

FANIDAN

MARUZALAR MATNI

Jizzax-2020

Tuzuvchilar:

Baratov J.R. JizPI, “Informatsion texnologiyalar” kafedrasi assistenti

Hasanov U.J. JizPI, “Informatsion texnologiyalar” kafedrasi assistenti

MUNDARIJA

MA’RUZALAR MATNI

1-Mavzu: Texnik tizimlarda zamonaviy dasturlash texnologiyalari

Reja:

1. **Zamonaviy ilovalar.**
2. **Dasturlashning rivojlanish bosqichlari.**
3. **Zamonaviy dasturlash muhitlari.**

1. Zamonaviy ilovalar. Ma’lumki, zamonaviy kompyuterlar operatsion tizimga ega. Operatsion tizim ilovalari ular yordamida amaliy vazifalarni bajarishga xizmat qiladi. Ilovalarga misol sifatida ofis ilovalari: MicroSoft Word (qisqacha – MS Word), – MS Excel, – MS Access, – MS Power Pointlarni keltirish mumkin. Bu ilovalarning o‘ziga xos jihati ularning interfaolligi bo‘lib, ular yordamida yuzlab va minglab amallarni bajarish mumkin. Bunday murakkab ilovalarning o‘zi qanday yaratiladi? Ilovalarni yaratishning dasturiy vositalari bugungi kunga kelib juda katta imkoniyatlarga ega. Ular bilan tanishishni dasturlash tarixiga nazar tashlashdan boshlaymiz.

2. Dasturlashning rivojlanish bosqichlari. Ilovalar (amaliy dasturlar) yaratish vositalarining rivojlanishini quyidagi bosqichlarga ajratish mumkin:

Dastlab yaratilgan kompyuterlarda dastur bevosita mikroprotsessorning buyruqlari (mashina kodi) ketma-ketligi ko‘rinishida yozilgan. Bu esa dasturlash uchun juda katta kuch va vaqt talab qilgan, dasturdagi xatolarni topish mushkul bo‘lgan. Bu ishni bir oz bo‘lsada osonlashtirish uchun mikroprotsessor buyruqlari uchun qisqa nomlar kiritilgan va maxsus dastur bunday nomlarni mashina kodi (mikroprotsessor buyruqlari)ga o‘girib bergan. Hosil bo‘lgan dastur kodi bevosita kompyuterda bajarilgan. Bunday dasturlash Assembler tilida dasturlash deb atalgan. Kompyuterlar ommaviy ishlab

chiqarila boshlangach (uchinchi avlod kompyuterlari), ularda operatsion tizim vujudga keldi. Bunday kompyuterlarda dasturlash uchun yuqori darajadagi dasturlash tillari yaratildi. Dasturlash endi kompyuterning mashina kodiga bevosita bog‘liq bo‘lmay qoldi. Dasturlash tili odamlar orasidagi muloqot tiliga ko‘proq o‘xshab bora boshladi. Masalan, bu tillarda agar x>0 bo‘lsa, u holda y=ln(x) kabi jumlalardan foydalanish mumkin bo‘lgan. Dastur matnini kompyuter tushunadigan mashina kodiga o‘girishni maxsus ishlab chiqilgan va translyator deb ataluvchi dastur bajargan. Natijada dasturlash ancha osonlashib, kompyuter yordamida yechiladigan masalalar ko‘lami kengaydi. Navbatdagi bosqichda kompyuterda yechiladigan masala bir necha kichikroq va osonroq masalalarga ajratilgan. Zarurat bo‘lganda, ular ham o‘z navbatida yanada kichikroq masalalarga bo‘lib chiqilgan. Bu esa bitta masala bo‘yicha bir nechta, ba’zan o‘nlab dasturchilar birgalikda shug‘ullanishlariga imkon berdi. Yaratilayotgan dasturiy vositalarning, ya’ni ilovalarning sifati yanada oshdi, dasturlar yanada murakkablashdi, dasturlar ko‘proq imkoniyatlarga ega bo‘ldi. Bunday dasturlash tuzilmaviy dasturlash deb atalgan. Biz bilgan dasturlash tili Turbo Pascal shunday tillar jumlasiga kiradi. Masalani kichikroq masalalarga ajratish hamda dasturlash protseduralar va funksiyalar yordamida amalga oshirilgan. Dasturlashning navbatdagi bosqichi obyektlarga yondashgan dasturlash deb ataladi. Bu birinchi navbatda shaxsiy kompyuterlarning keng tarqalishi va ularda ishlashni yanada oson va qulay qilish maqsadida yaratilgan grafik operatsion tizimlar (ularga hozirgi paytda keng tarqalgan Windows ham kiradi) bilan bog‘liq. Biz bilamizki, axborotni qayta ishlash usulini ma’lum bo‘lgan ma’lumot deb atash mumkin. Har qanday axborotni obyekt deb qarashimiz mumkin. Turbo Pascalda ma’lumotlar o‘zgaruvchi va o‘zgarmaslarga bo‘linadi. Ularni qayta ishlash usullari (qayta ishlash algoritmlari) alohida protseduralar va funksiyalar ko‘rinishida bo‘ladi. Obyektlarga mo‘ljallangan dasturlashda avvallari birlashtirish mumkin bo‘lmagan bu ikki unsurni birlashtirish imkoni paydo bo‘ldi. Ular orasidagi farq yo‘qola boshladi. Natijada axborot bilan ishlash yanada qulay va yanada osonroq bo‘lib qoldi. Bu esa bitta dastur ustida o‘nlab, yuzlab va hatto minglab dasturchilar birgalikda ishlashlariga imkon berdi. Yaratilgan amaliy dasturiy

vositalarning imkoniyatlari keskin oshib ketdi. Yaratilayotgan dasturiy vositaning o‘zini ham axborot deb qarash mumkin. Demak, dastur kodini yaratishda ham obyektlardan foydalanish mumkin. Masalan, har bir dasturning o‘z oynasi bo‘ladi. Dastur oynasining bo‘yi va eni kabi xossalari (oyna obyekti xossalari) bor, oynani yaratish, yopish, joyini va o‘lchovlarini o‘zgartirish kabi qayta ishlash usullari (oyna obyekti usullari) yordamida dastur oynasi bilan ishlashni osonlashtirish mumkin. Agar oyna degan obyekt yaratilgan bo‘lsa, dastur oynasi bilan ishlash bu obyektning xossalarini kerakli tarzda o‘rnatish va obyekt usullaridan kerakli joyda foydalanish ko‘rinishida juda oson kechadi. Natijada dasturlash yanada osonlashdi, yaratilayotgan dasturlarning sifati yanada oshdi.

Endi ilova yaratish uchun boshqalar tomonidan yaratilgan tayyor obyektlarni dasturga kiritish va ularning xossalarini kerak bo‘lgandek qilib o‘rnatish yetarli bo‘lib qoldi. Bunday dasturlash vizual dasturlash deb ataladi va dasturlash ko‘proq Lego yordamida o‘yinchoq yasashga o‘xshab qoldi. Hozirgi paytda umumlashgan dasturlash usuli ommaviylashib bormoqda. Uning ma’nosini quyidagicha tushuntirish mumkin. Dasturda uchburchak, to‘rtburchak, ko‘pburchak, aylana kabi obyektlar va ularning yuzasi degan xossalari bo‘lsin. Ularning har biridan foydalanish o‘rniga geometrik shakl yuzasi degan xossadan foydalanishimiz mumkin. Dasturning o‘zi qanday geometrik shakl to‘g‘risida gap ketayotganini aniqlab, kerakli obyektning kerakli xossasidan foydalanib javobni topadi. Natijada dasturlash yanada osonlashdi va arzonlashdi. Endi yaratilayotgan dasturni bir paytning o‘zida hoxlagan operatsion tizim uchun va uning interfeysini kerakli tilda yaratish mumkin. Ularni o‘zgartirish ko‘p vaqt va kuch talab qilmaydi.

3. Zamonaviy dasturlash muhitlari. Hozirgi paytda dunyoda o‘n milliondan ortiq dasturchilar bo‘lib, ulardan ikki millioni professional, qolganlari esa havaskor

dasturchilardir. Albatta, ularning ish qurollari, ya’ni ular foydalanadigan dasturlash muhitlari ham bir-biridan farqlanadi. Bugungi kunda keng tarqalgan dasturlash muhitlarida asosan uchta dasturlash tili: Si, BASIC, Pascal dan keng foydalaniladi. Professional va tajribali havaskor dasturchilar asosan C (Si) va uning keyingi variantlari C++ va C# dan foydalanishsa, faqat o‘zlarining masalalarini yechish uchun dasturlashda foydalanadiganlarning ko‘pchiligi Pascal dan foydalanishadi. BASIC (to‘g‘risi, Visual BASIC) Microsoft kompaniyasi mahsulotlari ofis ilovalari va boshqa kompaniyalarning bir qator mahsulotlari, grafik muharrirlarda ishlarni avtomatlashtirish uchun qo‘llaniladi. Kuchli raqobat va foydalanuvchilarni jalb qilish maqsadida hozirgi paytda ommaviy dasturlash muhitlari bir paytda bir nechta dasturlash tillaridan foydalanish imkonini beradi. Shuningdek, so‘nggi paytda yaratilayotgan dasturlash tillarining ko‘pchiligi yuqoridagi tillardan biridan foydalanayotganlar uchun mo‘ljallab yaratilgan. Masalan, keng tarqalgan veb dasturlash tillari Java, Java Script, ASP, PHP, Python lar C++ va C# ga, keng tarqalgan kompyuter matematikasi paketlari MatLab, MathCAD, Maplelarning dasturlash tillari Pascalga, MacroMedia Flash nomli animatsion grafika yaratish ilovasining dasturlash tili Action Script esa Visual BASICga o‘xshab ketadi.

SAVOL VA TOPSHIRIQLAR

1. Dasturlashning rivojlanish bosqichlarini sanab bering.
2. Zamonaviy dasturlash muhitlari haqida nimalar bilasiz?

2-Mavzu: Dasturlashga kirish, dasturlashning asosiy tushunchalari

Reja:

1. **C++ algoritmik tilining asosiy tushunchalari**
2. **Butun va haqiqiy sonlar**

C++ algoritmik tilining asosiy tushunchalari

C++ tili Byarn Straustrup tomonidan 1980 yil boshlarida ishlab chiqilgan. C++ tilida yaxshi dastur tuzish uchun “aql, farosat va sabr” kerak bo’ladi. Bu til asosan tizim sathida dasturlovchilar uchun yaratilgan.

C++ algoritmik tilining alifbosi quyidagilardan iborat:

- katta va kichik lotin harflari;
- 0 dan 9 gacha raqamlari;
- maxsus belgilar (+,-,*,/,=,>,<,{,},[,],') ni o'z ichiga oladi.

C++ tilida **so'z** deb bir nechta belgilar ketma – ketligi tushuniladi. Xizmatchi so'z deb C++ tilidagi standart nom tushuniladi. Bu nom maxsus ma'noni anglatadi va uni ma'lumotlarga berib bo'lmaydi. Masalan: **int, float, for, while** va hokazo. C++ tilida ma'lumotlarning elementlari bo'lib o'zgaruvchilar, o'zgarmaslar, izohlar xizmat qiladi.

O'zgaruvchi. Xotiraning nomlangan qismi bolib, o'zida ma'lum bir toifadagi qiymatlarni saqlaydi. O'zgaruvchining nomi va qiymati bo'ladi. O'zgaruvchining nomi orqali qiymat saqlanayotgan xotira qismiga murojaat qilinadi. Programma ishlashi jarayonida o'zgaruvchining qiymatini o'zgartirish mumkin. Har qanday o'zgaruvchini ishlatishdan oldin, uni e'lon qilish lozim. Quyida butun sonlardan foydalanish uchun b, haqiqiy sonlardan foydalanish uchun h o'zgaruvchisi e'lon qilingan:

```
int b;
float h;
```

O'zgarmaslar (**const**) Hisoblash jarayonida qiymatini o'zgartirmaydigan kattaliklarga aytiladi.

```
float const pi = 3.14;
```

Izohlar. Programmaning ma'lum qismini tavsiflash uchun ishlatiladi va bu qatorda hech qanday amal bajarilmaydi, ya'ni programmaning biror qismini yaxshiroq tushuntirish uchun xizmat qiladi. Izoh "/*" va "*/" simvollari orasida beriladi.

/ Bu yerga izoh yoziladi. */*

Bundan tashqari bir satrli izohlardan ham foydalanish mumkin. Buning uchun izoh boshiga "//" belgisi qo'yiladi.

Operator. Tilning yakunlangan jumlasi hisoblanadi va ma'lumolar taxlilining tugallangan bosqichini ifodalaydi. Operatorlar nuqtali vergul “;” bilan ajratiladi. Ya'ni “;” operatorning tugallanganligini bildiradi. C++ da operatorlar progammada keltirilgan ketma - ketlikda bajariladi.

Identifikator. Programmist tomonidan programma elementlari (funksiya, o'zgaruvchilar, o'zgarmaslar …) uchun ixtiyoriy tanlangan nom.

Identifikator tanlaganda quyidagilarga ahamiyat berish kerak:

- Identifikator lotin harflaridan boshlanishi shart;
- Ikkinchi simvoldan boshlab raqamlardan foydalanish mumkin;
- C++ da katta kichik harflar farq qiladi. Ya'ni quyidagilarning har biri alohida identifikator hisoblanadi: KATTA, katta, KaTTa, kAttA, Katta, KattA, ...
- Probel C++ da so'zlarni ajratish uchun ishlatiladi. Shuning uchun identifikatorda probeldan foydalanib bo'lmaydi;
- Xizmatchi (**int, float, for, while** kabi) so'zlardan identifikator sifatida foydalanib bo'lmaydi;

C++ tilining kalit so'zlariga quyidagilar kiradi:

asm, auto, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, explicit, extern, float, for, friend, goto, if, inline, int, long, mutable, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, swith, template, this, throw, try, typedef, typename, union, unsigned, virtual, void, volatile, while.

Protsessor registrlarini belgilash uchun quyidagi so'zlar ishlariladi: _AH, _AL, _AX, _EAX, _BH, _BL, _BX, _EBX, _CL, _CH, _CX, _ECX, _DH,

_DL, _DX, _EDX, _CS, _ESP, _EBP, _FS, _GS, _DI, _EDI, _SI, _ESI, _BP, _SP, _DS, _ES, _SS, _FLAGS.

Eslatma. Identifikator tanlashda birinchi belgi sifatida "_" belgisidan foydalanmaslik
tavsiya etiladi.

C++ da programma funksiya yoki funksiyalardan tashkil topadi. Agar programma bir nechta funksiyadan iborat bo'lsa, bir funksiyaning nomi **main** bo'lishi shart.

Programma aynan main funksiyasining birinchi operatoridan boshlab bajariladi.

Funksiyaning aniqlashishi quyidagicha bo'ladi:

qaytariluvchi_qiymat_toifasi funksiya_nomi ([parametrlar])
{ funksiya tanasini tashkil qiluvchi operatorlar }

Qoida bo'yicha funksiya qandaydir bir qiymatni hisoblash uchun ishlatiladi. Shuning uchun funksiya nomi oldidan, funksiya qaytaradigan qiymat toifasi yoziladi. Agar funksiya hech qanday qiymat qaytarmaydigan bo'lsa, **void** toifasi yoziladi. Agar funksiya qaytaradigan qiymat toifasi yozilmagan bo'lsa, **int** (butun) toifali qiymat qaytariladi deb qabul qilinadi.

C++da oddiy matnni ekranga chiqaruvchi programmani ko'rib chiqamiz

```
1 // Muallif: Jasur Baratov
2 // Sana: 23 fevral 2011 yil
3 // Maqsad: Matnni ekranga chiqaruvchi programma
4
5 #include <iostream> // ekranga ma'lumot chiqarish uchun
6
7 int main()
8 {
9 std::cout << "Assalomu alaykum bo'lajak programmist!\n";
10
```

11 **return** 0;

12 }

Har bir satrni o'rganib chiqamiz:

1, 2, 3 - satrlar izoh hisoblanadi. Malakali programmistlar har qanday programma muallif, programmaning tuzilish sanasi va maqsadini ifodalovchi izoh bilan boshlanishini maslahat berishadi.

4, 6, 10 - satrlar bo'sh satrlar hisoblanadi. Bosh satrlar programma qismlarini bir -biridan ajratib qo'yish uchun ishlatiladi. Programma qismlarining bir - biridan ajralib turishi, programma o'qilishini osonlashtiradi.

5 - satrda, klaviaturadan ma'lumotlarni kiritish va ekranga chiqarish uchun **<iostream>** sarlavha fayli programmaga qo'shilyapti. Bu satr klaviatura orqali ma'lumot kirituvchi va ekranga nimadir chiqaruvchi har qanday programmada bo'lishi shart. Aks xolda xato sodir bo'ladi.

Agar sizning kompilyatoringiz eski bo'lsa, unda **<iostream.h>** yozishingiz lozim bo'ladi. *"// ekranga ma'lumot chiqarish uchun"* yozuvi bir satrli izoh hisoblanadi.

7 - satrda butun toifadagi qiymat qaytaruvchi **main** funksiyasi berilgan. **int** xizmatchi so'zi butun toifadagi ma'lumotlarni e'lon qilishi uchun ishlatiladi.

8 - satrdagi ochuvchi figirali { funksiya tanasining boshlanganini bildiradi.

12 - satrdagi yopuvchi figirali } funksiya tanasining tugaganini bildiradi.

9 - satrda **std::cout** << orqali ma'lumotlar ekranga chiqariladi. Qo'shtirnoq ("_") orasida yozilgan ma'lumotlar satr deyiladi. Qo'shtirnoq orasida nima yozilsa, hech qanday o'zgarishsiz ekranga chiqariladi.

9 - satr oxiridagi nuqtali vergul (;) **std::cout** operatori tugallanganligini bildiradi. ; operatorlarni bir - biridan ajratish uchun xizmat qiladi. Ya'ni operator tugallanganligini bildiradi. 5 - satrdagi kabi reprotsessor amalidan keyin ; qo'yilmaydi.

11 - satrdagi **return** xizmatchi so'zi orqali funksiya 0 qiymat qaytaradi va programma muvoffaqiyatli yakunlanadi.

O'zgaruvchilarni e'lon qilish. Programmada ishlatilgan barcha o'zgaruvchilarni qaysi toifaga tegishli ekanligini e'lon qilish kerak. Ma'lulotlarni e'lon qilishning umumiy ko'rinishi quyidagicha:

toifa_nomi o'zgaruvchi;

Agar bir nechta o'zgaruvchi bir toifaga mansub bo'lsa, ularni vergul bilan ajratib berish mumkin. Butun sonlarni ifodalash uchun **int** a haqiqiy sonlarni ifodalash uchun **float** xizmatchi so'zlaridan foydalaniladi. Bu ma'ruzada shu 2 tasini bilish bizga kifoya qiladi. Keyingi mavzuda butun va haqiqiy sonlar haqida batafsil gaplashamiz. int x,y; *// butun toifadagi o'zgaruvchilarni e'lon qilish* float a,b,c; *// haqiaiy toifadagi o'zgaruvchilar e'lon qilish*

Kiritish va chiqarish operatorlari. Programmada klaviatura orqali ma'lumot kiritish va ekranga chiqarish uchun preprotsessor direktivasini, ya'ni **#include <iostream>** ni programmaga qo'shish shart. Ma'lumotlarni kiritish **std::cin** >>, ma'lumotlarni chiqarish **std::cout** << operatori orqali amalga oshiriladi.

std::cin >> a;

Bu operator bajarilganda ekranda kursor paydo bo'ladi. Kerakli ma'lumot klaviatura orqali kiritilgandan so'ng **Enter** tugmasi bosiladi. cout orqali ekranga ixtiyoriy ma'lumotni chiqarish mumkin. Satrli ma'lumotlarni ekranga chiqarish uchun, ularni qo'shtirnoq orasida yozish kerak. Quyida a va b sonlarining yig'indisini chiqaruvchi programma berilgan:

```
#include <iostream>
// standart nomlar fazosidan foydalanishni e'lon qilish
using namespace std;
int main()
{
int a, b, c;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
c = a + b;
cout << c << endl;
```

return 0;

}

Butun va haqiqiy sonlar

Programmistlar doim programma ishlashi jarayonida xotiradan kamroq joy talab qilishligi haqida bosh qotirishadi. Bu muammolar programmadagi o'zgaruvchilar sonini kamaytirish, yoki o'zgaruvchilar saqlanadigan yacheyka hajmini kamaytirish orqali erishiladi.

Biz butun va haqiqiy sonlarni e'lon qilishni bilamiz. Bulardan tashqari C++ da butun va haqiqiy sonlarni e'lon qilish uchun bir nechta toifalar mavjud. Ular bir – biridan kompyuter xotirasida qancha hajm egallashi va qabul qiluvchi qiymatlar oralig'i bilan farq qiladi.

Butun sonlar

Toifa ko'rinishi	Qabul qiladigan qiymatlar oralig'i	Kompyuter xotirasida egallagan hajmi
unsigned short int	0..65535	2 bayt
short int	–32768..32767	2 bayt
unsigned long int	0..42949667295	4 bayt
long int	–2147483648..2147483647	4 bayt
Int	(16 razryadli) –32768..32767	2 bayt
Int	(32 razryadli) –2147483648..2147483647	4 bayt
unsigned int	(16 razryadli) 0..65535	2 bayt
unsigned int	(32 razryadli) 0..42949667295	4 bayt

Haqiqiy sonlar

Toifa ko'rinishi	Qabul qiladigan qiymatlar oralig'i	Kompyuter xotirasida egallagan hajmi

float	1.2E–38..3.4E38	4 bayt
double	2.2E–308..1.8E308	8 bayt
long double (32 razryadli)	3.4e-4932..-3.4e4932	10 bayt

Boshqa toifalar

Toifa ko'rinishi	Qabul qiladigan qiymatlar oralig'i	Kompyuter xotirasida egallagan hajmi
bool	**true** yoki **false**	1 bayt
char	0..255	1 bayt
void	2 yoki 4	

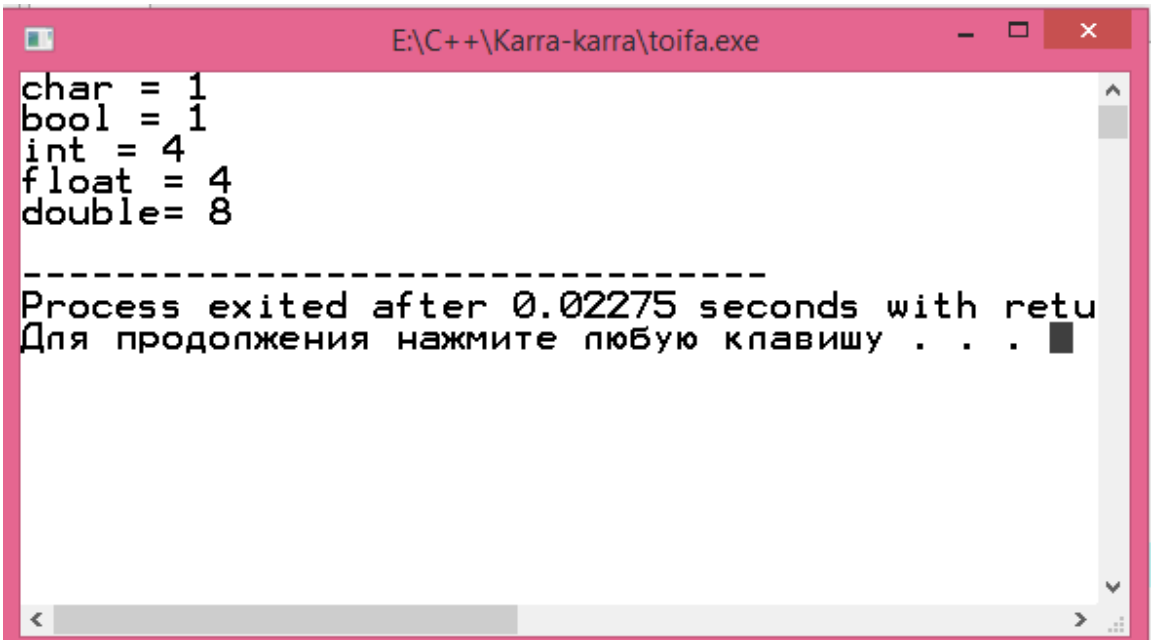
Har xil toifadagi o'zgaruvchilar kompyuter xotirasida turli xajmdagi baytlarni egallaydi. Xattoki bir toifadagi o'zgaruvchilar ham qaysi kompyuterda va qaysi operatsion sistemada ishlashiga qarab turli o'lchamdagi xotirani egallashi mumkin.C++ da ixtiyoriy toifadagi o'zgaruvchilarning o'lchamini sizeof funksiyasi orqali aniqlash mumkin. Bu funksiyani o'zgarmasga, biror toifaga va o'zaruvchiga qo'llash mumkin

Toifalarni kompyuter xotirasida egallagam xajmini aniqlash

```
// Muallif : Jasur Baratov
// Sana : 11 sentyabr 2019 yil
// Maqsad : Toifalarni kompyuter xotirasida egallagan xajmini aniqlash
#include <iostream>
using namespace std;
int main()
{
cout << "char = " << sizeof(char) << endl;
cout << "bool = " << sizeof(bool) << endl;
```

```
cout << "int = " << sizeof(int) << endl;
cout << "float = " << sizeof(float) << endl;
cout << "double= " << sizeof(double)<< endl;
return 0;
}
```

Ekranga quyidagicha natija chiqariladi



(1.1-rasm)

Matemetik funksiyalardan programmada foydalanish uchun **math.h** sarlavha faylini progarmmaga qo'shish kerak. **#include <math.h>**

Funksiyaning **C++** da ifodalanishi	Funksiyaning matematik ifodalanishi
1. abs(x) - butun sonlar uchun 2. fabs(x) - haqiqiy sonlar uchun 3. labs(x) - uzun butun son uchun	/x/
pow(x, y)	X^y
pow10(x)	10x
sqrt(X)	$\sqrt{x}$
ceil(x)	haqiqiy toifadagi x o'zgaruvchisi qiymatini unga eng yaqin katta butun songa aylantiradi.

floor(x)	haqiqiy toifadagi x o'zgaruvchisi qiymatini unga eng yaqin kichik butun songa aylantiradi
cos(x)	cosx
cin(x)	sinx
tan(x) yoki sin(x)/cos(x)	tgx
1/tan(x) yoki cos(x)/sin(x)	ctgx
log10(x)	lgx
log(x)	lnx
log(a)/log(b)	$\log_b a$
eps(x)	e^x
pow(x, b/a)	$\sqrt[a]{x^b}$
pow(sin(pow(x,a)) ,b)	$\sin^b x^a$

Eslatma: Barcha trigonometrik funksiyalar radian o`lchovida beriladi

1 - Misol: n va m natural sonlari berilgan. n sonini m soniga bo'lib, qoldiqni aniqlovchi programma tuzilsin.

```
// Muallif: Jasur Baratov
// Sana: 24 fevral 2011 yil
// Maqsad: n sonini m soniga bo'lib, qoldiqni aniqlash
#include <iostream>
int main()
{
int n, m, qoldiq;
cout << "n="; cin >> n;
cout << "m="; cin >> m;
// % qoldiqni olishni bildiradi
qoldiq = n % m;
cout << "Qoldiq=" << qoldiq << endl;
return 0;
}
```

Programmalash san'ati

1. Har bir programma, muallif, sana, programma maqsadini anglatuvchi izoh bilan boshlanishi kerak.
2. Programma yozayotganda joy tashlashlarni kelishilgan, aniq bir qoida asosida olib borgan maqul.

   Masalan, tabulyatsiyani 4 ta probel deb qabul qilish mumkin. Ammo bu har kimning tasavvuriga bog'liq, maqsad shuki, programma sodda q'ishli va ko'rinishli bo'lsin.
3. Har bir verguldan keyin probel tashlang, programma oson o'qilsin.
4. O'zgaruvchilarni e'lon qilishni boshqa operatorlardan bo'sh satr bilan ajratib qo'ying.
5. (+, -, *, /) kabi amallarni har ikkala tomonidan probel qo'ying. Bu programma o'qilishini qulaylashtiradi.

2 - Misol: n va m natural sonlari berilgan. n sonini m soniga bo'lib, butun qismini aniqlovchi programma tuzilsin.

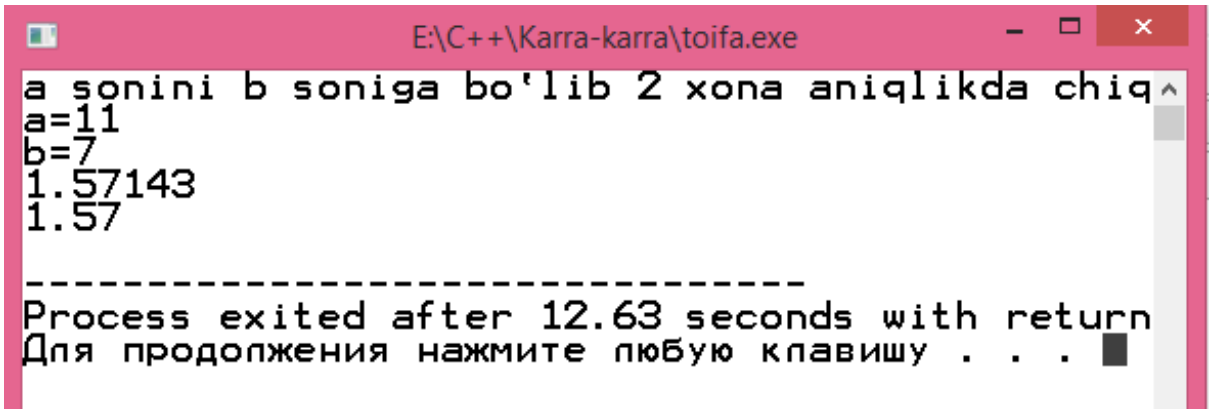
```
// Muallif: Jasur Baratov
// Sana: 24 fevral 2011 yil
// Maqsad: n sonini m soniga bo'lib, butun qismini aniqlash
#include <iostream>
int main()
{
int n, m, b;
cout << "n="; cin >> n;
cout << "m="; cin >> m;
b = n / m;
cout << "Butun qismi=" << b << endl;
return 0;
}
```

3 - misol. a sonini b soniga bo`lib 2 xona aniqlikda chiqarish.

```
// Muallif: Jasur Baratov
```

```
// Sana : 8 senyabr 2019 yil
// Maqsad: Haqiqiy sonni 2 xona aniqlikda chiqarish
#include <iostream>
#include <iomanip>
// <iomanip> sarlavha faylini qo'shamiz
using namespace std;
int main()
{
float a, b;
cout << "a sonini b soniga bo`lib 2 xona aniqlikda chiqarish"<<endl;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
a = a / b;
cout << a << endl;
cout << setprecision(2) << fixed << a << endl; return 0; }
```

Programma ishga tushganda ekranda quyidagicha natija chiqariladi:



(1.2-rasm)

4 - misol. Bir toifadan boshqasiga o'tish c++ da bir toifadan boshqasiga o'tishning oshkor va oshkormas usullari mavjud. Oshkor ravishda toifaga keltirish uchun qavs ichida boshqa toifa nomi yoziladi.

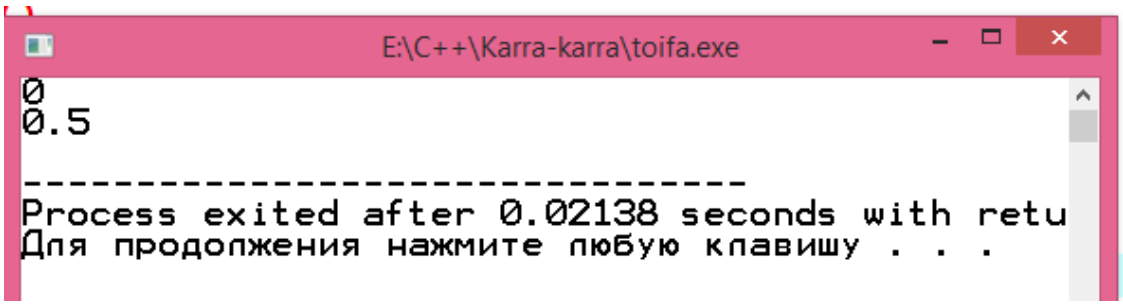
```
#include <iostream>
using namespace std;
```

```
int main()
{
float haqiqiy = 5.57;
int oshkor, oshkormas;
// oshkormas ravishda butun toifaga o'tish oshkormas = haqiqiy; oshkor = (int)
haqiqiy; // oshkor holda butun toifaga o'tish
cout << "haqiqiy = " << haqiqiy << endl;
cout << "oshkor = " << oshkor << endl;
cout << "oshkormas = " << oshkormas << endl;
return 0;
}
```

5 - misol. Butun sonni bo'lish

```
#include <iostream>
using namespace std;
int main()
{
int bir = 1;
int ikki = 2;
cout << bir / ikki << endl;
cout << ((float)bir) / ((float)ikki) << endl;
return 0;
}
```

Programma ishga tushganda ekranda quyidagicha natija chiqariladi:

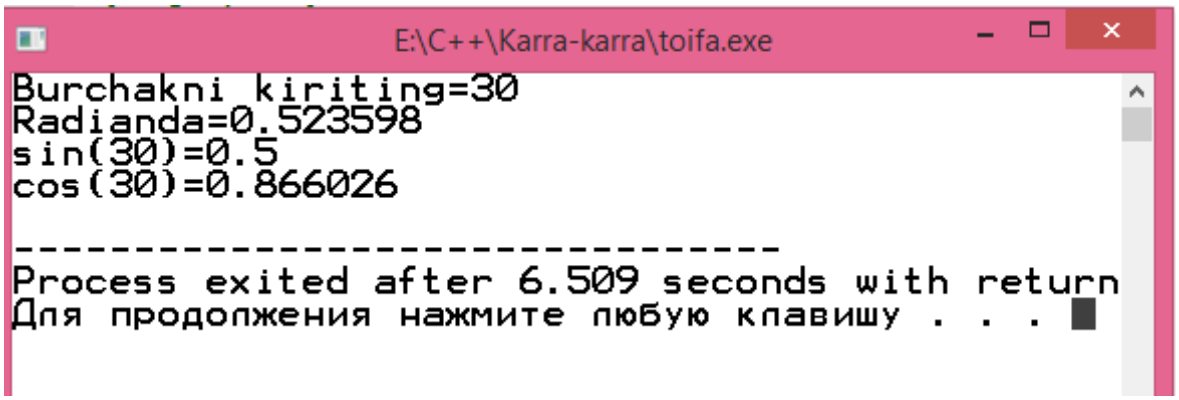


(1.3-rasm)

6 - misol. Trigonometrik funksiyalar bilan ishlash

```
// Muallif: Jasur Baratov
// Sana : 05.11.2011
// Maqsad: Kiritilgan burchak sin va cosinusini topish
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
 float const pi = 3.14159;
float burchak, burchak_radian;
cout << "Burchakni kiriting=";
cin >> burchak;
burchak_radian = burchak * pi / 180;
cout << "Radianda=" << burchak_radian << endl;
cout << "sin(" << burchak << ")=" << sin(burchak_radian) << endl;
cout << "cos(" << burchak << ")=" << cos(burchak_radian) << endl;
return 0;
}
```

Programma ishga tushganda ekranda quyidagicha natija chiqariladi:



(1.4-rasm)

Nazorat savollari:

1. C++ algoritmik tilining alifbosi nimalardan iborat?

2. O'zgaruvchi, o'zgarmaslarga ta’rif bering.
3. Operator nima?
4. Identifikator nima?
5. Ma'lumotlar qanday e'lon qilinadi?
6. Butun va haqiqiy sonlar qanday e'lon qilinadi?
7. Butun sonlar toifalarini sanab bering. Ular nimasi bilan farq qiladi?
8. Haqiqiy sonlar toifalarini sanab bering. Ular nimasi bilan farq qiladi?
9. Matematik funksiyalarni sanab bering.

3-Mavzu: C++ dasturlash tilining asosiy konstruksiyalari, ulardan foydalanish xususiyatlari.

Reja:

1. C++ tilida ifodalar
2. Ma`lumotlarning mantiqiy toifalari

C++ tilida o'zgaruvchi qiymatini birga oshirish va kamaytirishning samarali usullari mavjud. Ular inkrement (++) va dekrement (--) unar amallaridir.

Inkrement va dekrement amallarining prefiks va postfiks ko'rinishlari mavjud.

```
x = y++; // postfiks
x = --y; // prefix
sanagich++; // unar amal, "++sanagich;" bilan ekvivalent
a--; // unar amal, "--a;" bilan ekvivalent
```

Quyida keltirilgan amallar bir xil vazifani bajaradi:

i++	i = i + 1
i--	i = i - 1
a += b	a = a + b
a -= b	a = a - b
a *= b - c	a = a * (b - c)
++i	i++
--c	c--

C++ da ifodalar quyidagi tartibda hisoblanadi:

1. Qavs ichidagi ifodalar hisoblanadi
2. Funksiyalar qiymati hisoblanadi. (sin(x), cos(x), sqrt(x) va xakazo)
3. Inkor amali (! - not)
4. Bo'lish, ko'paytirish kabi amallar (/,*, %, ...)
5. Qo'shish kabi amallar (+, -, or, xor)
6. Munosabat amallari (=, <>, <, >, <=, >=)

Ma`lumotlarning mantiqiy toifalari

Mantiqiy toifa **bool** ikki hil qiymat qabul qilishi mumkin: true (rost, 1) va false (yolg`on, 0). Mantiqiy ma`lumotlarni e`lon qilish uchun **bool** xizmatchi so'zidan foydalaniladi.

bool a, b;

Mantiqiy toifadagi o`zgaruvchilarga qiymat berish quyidagicha amalga oshiriladi:

a = **true**; // rost

b = **0**; // yolg'on, false

Mantiqiy amallar:

! (inkor qilish) - mantiqiy operatori mantiqiy ifodalar yoki o`zgaruvchilar oldidan qo`yiladi. Mantiqiy ifoda yoki o`zgaruvchining qiymatini teskarisiga o`zgartiradi.

&& (Mantiqiy ko`paytirish) - mantiqiy operatori ikkita mantiqiy o`zgaruvchini birlashtiradi. Agar ikkala o`zgaruvchi ham rost qiymatga ega bo`lsa natija rost, aks holda yolg`on natija beradi.

|| (mantiqiy qo`shish) - mantiqiy operatori ikkita mantiqiy o`zgaruvchini birlashtiradi.

Agar o`zgaruvchilardan kamida bittasi rost qiymatga ega bo`lsa natija rost, aks holda yolg`on natija beradi.

! - mantiqiy inkor operatori jadvali

X	**!**X
false	true
true	false

&&, || manqiqiy operatorlai jadvali

X	Y	X && Y	X || Y
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Munosabat amallari

== - teng

<= - kichik yoki teng

!= - teng emas

>= - katta yoki teng

< - kichik

> - kata

Nazorat savollari:

1. Mantiqiy toifalar qanday e’lon qilinadi?
2. Mantiqiy amallarni tushuntirib bering.
3. Munosabat amallarini tushuntiring.
4. Mantiqiy amallar jadvalini tuzib bering.

4-Mavzu: Tarmoqlanuvchi jarayonlarni dasturlash.
Reja:

1. **Shart operatori**
2. **Tanlash operatori**

Shart operatori

Programma tuzish mobaynida o`zgaruvchilar qiymatiga qarab u yoki bu natijani qabul qilishga to`g`ri keladi. Bu o`z navbatida programmani tarmoqlanishga olib keladi. Tarmoqlarning qaysi qismi bajarilishi ayrim shartlarga qarab aniqlanadi.

Shart operatori: Shart operatori boshqarishni qaysi tarmoqqa uzatishni ta’minlaydi.

Shart operatorining ikki xil ko`rinishi mavjud. Operatorning umumiy ko`rinishi va qisqa ko`rinishi.

Shart operatorining umumiy ko`rinishi:

if (<shart>)

<operator1>;

else

<operator2>**;**

if agar, else aks holda ma`nolarini anglatadi. Shart operatorining qisqa ko`rinishi:

if (<shart>) <operator1>;

<**shart**> tekshirilishi lozim bo`lgan mantiqiy ifoda

<**operator** 1> Agar shart rost (**true**) qiymatga ega bo`lsa bajarilishi lozim bo`lgan operator.

<**operator** 2> Agar shart yolg`on (**false**) qiymatga ega bo`lsa bajarilishi lozim bo`lgan operator.

Shart operatori tarkibida **ixtiyoriy operatordan** foydalanish mumkin. Shu o`rinda Shart operatoridan ham.

Misol: Berilgan a sonini juft yoki toqligini aniqlovchi programma tuzilsin.

Agar a sonini 2 ga bo'lganda qoldiq 0 ga teng bo'lsa, bu son juft, aks xolda toq.

```
#include <iostream>
using namespace std;
int main()
{
int a;
cin >> a;
if (a % 2 == 0)
cout << "juft";
else
cout << "toq";
return 0;
}
```

C++ tili operatorlarni blok ko’rinishida bo’lishiga imkon beradi. Blok ‘{‘ va ‘}’ belgi oralig'iga olingan operatorlar ketma-ketligi bo’lib, u kompilyator tomonidan yaxlit bir operator deb qabul qilinadi. Blok ichida yangi o'zgaruvchilarni ham e'lon qilish mumkin. Bu o’zgaruvchilar faqat blok ichida ko’rinadi, undan tashqarida ko’rinmaydi, ya'ni blokdan tashqarida bu o'zgaruvchilarni ishlatib bo'lmaydi.

Blokdan keyin nuqtali vergul qo’yilmaydi, lekin blok ichida har bir operator nuqtali vergul bilan yakunlanishi shart.

Shart operatorida bir nechta operatordan foydalanish uchun bu operatorlarni blok ichiga yozish lozim bo'ladi. Yuqoridagi masalani blok orqali ifodalash quyidagicha bo'ladi.

Misol: Berilgan a sonini juft yoki toqligini aniqlovchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
int a;
```

```
cin >> a;
if (a % 2 == 0)
{
cout << "juft";
}
else
{
cout << "toq";
}
return 0;
}
```

Programmalashning yaxshi usuli:

Shart operatorida doimiy ravishda bloklardan foydalanish yo'l qo'yilishi mumkin bo'lgan xatoliklarni oldini oladi. Ba'zi programmistlar oldin ochuvchi va yopuvchi qavslarni {, } yozish, undan keyin blok ichidagi operatorlarni yozish lozimligini takidlashadi.

? : shart amali

Agar tekshirilayotgan shart nisbatan sodda bo’lsa, shart amalini «?: » ko’rinishini ishlatish mumkin. Bu operator quyidagi ko’rinishga ega:

<shart ifoda> ? <ifoda1> : <ifoda2>;

if shart operatoriga o’xshash holda bu shart amali quyidagicha ishlaydi: agar <shart ifoda> rost (true) bo’lsa <ifoda1> bajariladi, aks holda <ifoda2>. Odatda ifodalar qiymatlari birorta o’zgaruvchiga o’zlashtiriladi. Misol: 2 ta sondan kattasini topuvchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
int a, b, max;
```

```
cout << "a="; cin >> a;
cout << "b="; cin >> b;
max = ( a > b ) ? a : b;
cout << max << endl;
return 0;
}
```

Agar a>b shart bajarilsa max o'zgaruvchisi a ni, aks xolda b ni o'zlashtiradi.

Tanlash operatori

Boshqarishni uzatish operatorlaridan yana biri tanlash operatoridir. **Tanlash operatori** asosan bir nechta qiymatdan, o'zgaruvchiga mos qiymatni tanlashda va qiymatlarga mos ravishda boshqarishni uzatishda ishlatiladi.

Tanlash operatorining umumiy ko'rinishi:

switch (<o'zgaruvchi>)

{

case <o'zgarmas ifoda1> : <operator 1>; break;

case <o'zgarmas ifoda2> : <operator 2>; break;

. . .

case <o'zgarmas ifodaN> : <operator N>; break;

[default : operator N + 1];

}

Tanlash operatorida boshqarilish o`zgaruvchiga mos ravishda qiymatlarga uzatiladi va mos operator ishga tushadi. **default** operatori birorta ham qiymat o`zgaruvchiga to`g`ri kelmasa ishlatiladi. **default** operatorini ishlatmasdan tashlab ketish ham mumkin.

Eslatma: Dasturlashga doir kitoblarni o`qiganingizda, biror operatorning umumiy ko'rinishining to'rtburchak qavs [ ] belgisi oralig'ida yozilgan qismini ishlatmasdan tashlab ketish mumkin. Operatorning bu qismidan foydalanish ixtiyoriy bo`ladi.

Misol: Kiritilgan songa mos ravishda hafta kunini chiqaruvchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
int n;
cout << "Hafta kunini kiriting" << endl;
cin >> n;
switch (n)
{
case 1: cout << "Dushanba"; break;
case 2: cout << "Seshanba"; break;
case 3: cout << "Chorshanba"; break;
case 4: cout << "Payshanba"; break;
case 5: cout << "Juma"; break;
case 6: cout << "Shanba"; break;
case 7: cout << "Yakshanba"; break;
default: cout << "Bunday hafta kuni yo'q";
}
return 0;
}
```

Tanlash operatorida bir nechta qiymatga bir hil operator ishlatishi quyidagicha bo'ladi.

```
#include <iostream>
using namespace std;
int main()
{
int n;
cout << "1..10 oraliqdan son kiriting" << endl;
cin >> n;
```

```
switch (n)
{
case 1:
case 3:
case 5:
case 7:
case 9: cout << "Toq son kiritildi"; break;
case 2:
case 4:
case 6:
case 8:
case 10: cout << "Juft son kiritildi"; break; default: cout << "1 dan kichik
yoki 10 dan katta son kiritildi";
}
return 0;
}
```

Nazorat savollari

1. Shart operatorining qanday ko`rinishlarini bilasiz?
2. Shart operatori ichida shart operatoridan foydalanish mumkinmi?
3. Shart operatorida bir nechta operatordan foydalanish uchun nima qilinadi?
4. Tanlash operatori nima uchun ishlatiladi?
5. Tanlash operatori umumiy ko'rinishidagi [ ] ichiga yozilgan qismi nima ma'noni bildiradi?

5-Mavzu: Takrorlanuvchi jarayonlarni dasturlash va ularni muhandislik masalalarida qo’llash.

Reja:

1. **for sikl operatori**
2. **do - while sikl operatori**
3. **while sikl operatori**

Bir hil hisoblash jarayonlarini bir necha bor takrorlanishi **Sikl** deyiladi. C++ programmalashtirish tilida sikl operatorining bir necha xil turi mavjud.

• for sikl operatori

• do .. while sikl operatori

• while sikl operatori

Yechilayotgan masalaga qarab, programmist o`zi uchun qulay bo`lgan sikl operatoridan foydalanishi mumkin.

for takrorlash operatorining sintaksisi quyidagicha:

for (<ifoda1>; <ifoda2>; <ifoda3>)

<operator yoki blok>;

Bu operator amal qilishni <ifoda1> bajarishdan boshlaydi. Keyin takrorlash qadamlari boshlanadi. Har bir qadamda <ifoda2> bajariladi, agar natija 0 dan farqli yoki rost (true) bo’lsa, sikl tanasi - <operator yoki blok> bajariladi va oxirida <ifoda3> bajariladi, aks holda boshqaruv takrorlash operatoridan keyingi operatorga o’tiladi. Sikl tanasi – <operator yoki blok> sifatida bitta operator, shu jumladan bo’sh operator, yoki operatorlar bloki kelishi mumkin.

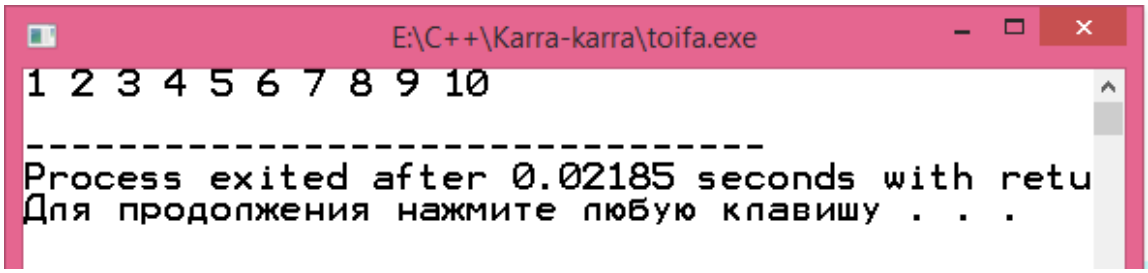
Sikl takrorlanishi davomida bajarilishi lozim bo'lgan operatorlar majmuasi sikl tanasi deyiladi. Sikl tanasi sifatida bir yoki bir nechta operatordan foydalanish mumkin. Agar sikl tanasida bir nechta operatordan foydalanmoqchi bo`lsak bu operatorlarni blok **{ }** orasiga olishimiz kerak.

1 dan 10 gacha bo'lgan sonlarni chiqaruvchi dastur:

```
#include <iostream>
using namespace std;
int main()
{
```

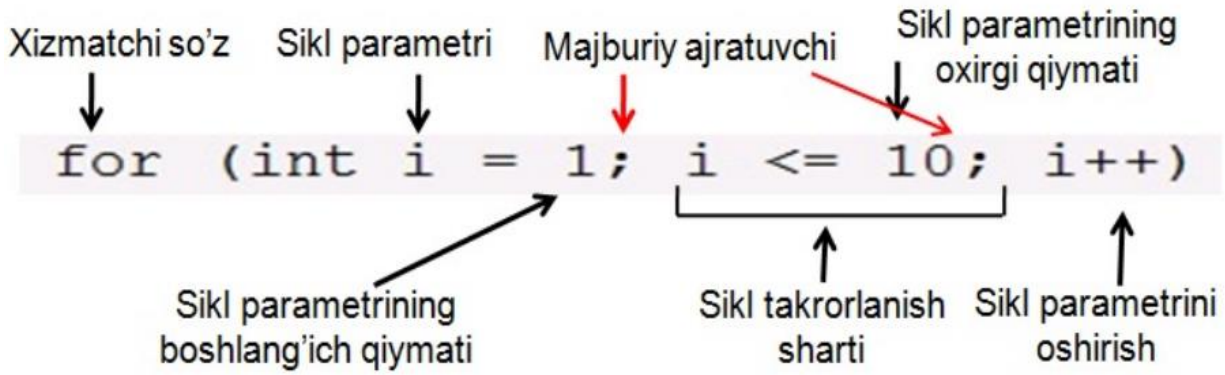
```
for (int i = 1; i <= 10; i++)
cout << i << " ";
cout << endl;
return 0;
}
```

Dastur natijasi



(5.1-rasm)

Quyidagi rasmda for sikl operatori batafsil berilgan.



(5.2-rasm)

for sikl operatorini umumiy ko'rinishda quyidagicha ifodalash mumkin:

for (sikl_o'zgaruvchisining_boshlang'ich_qiymati**;** takrorlanish_sharti**;** sikl_o'zgaruvchisini_oshirish) sikl_tanasi**;**

sikl_o'zgaruvchisini_oshirish o'rnida kamaytirish ham bo'lishi mumkin.

Agar sikl tanasida bir nechta operatordan foydalanmoqchi bo`lsak bu operatorlarni blok { } orasiga olishimiz kerak.

Programma taxlili:

Yuqoridagi 1 dan 10 gacha bo'lgan sonlarni chiqaruvchi programmani taxlil qilib chiqamiz.

for (int i = 1; i <= 10; i++)

cout << i << " ";

1. Sikl parametri (i) boshlang'ich qiymat 1 ni o'zlashtiradi. Ya'ni i = 1;
2. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 1 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "1 " chiqariladi.
3. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 2 ga teng bo'ladi.
4. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 2 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "2 " chiqariladi.
5. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 3 ga teng bo'ladi.
6. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 3 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "3 " chiqariladi.
7. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 4 ga teng bo'ladi.
8. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 4 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "4 " chiqariladi.
9. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 5 ga teng bo'ladi.
10. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 5 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "5 " chiqariladi.
11. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 6 ga teng bo'ladi.
12. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 6 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "6 " chiqariladi.
13. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 7 ga teng bo'ladi.
14. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 7 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "7 " chiqariladi.
15. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 8 ga teng bo'ladi.
16. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 8 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "8 " chiqariladi.
17. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 9 ga teng bo'ladi.
18. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 9 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "9 " chiqariladi.
19. Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 10 ga teng bo'ladi.

20.Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 10 <= 10 shart rost bo'lgani uchun sikl tanasi (cout << i << " ";) bajariladi. Ekranga "10 " chiqariladi.

21.Sikl parametrini oshirish (i++) bajariladi. i ning qiymati 11 ga teng bo'ladi.

22. Sikl takrorlanish sharti tekshiriladi. (i <= 10;). 11 <= 10 shart rost bo'lmagani uchun sikl tugatiladi va boshqarilish sikl operatoridan keyingi operatorga uzatiladi.

10 dan 1 gacha bo'lgan sonlarni chiqaruvchi dastur:

```
#include <iostream>
using namespace std;
int main()
{
for (int i = 10; i >= 1; i--)
cout << i << " ";
cout << endl;
return 0;
}
```

break – *funksiyasini har qanday sikl operatoriga qo`llash mumkin. Bu funksiya sikl tugatilishini ta'minlaydi. Ya`ni boshqarilishni sikl operatoridan keyingi operatorga uzatadi.*

continue – *funksiyasini har qanday sikl operatoriga qo`llash mumkin. Bu funksiya sikl parametrining keyingi qiymatni qabul qilishini taminlaydi.*

Boshqacha so`z bilan aytganda sikl tanasi tugatiladi. Bunda siklning o`zi tugatilmaydi.

Misol. n natural soni berilgan. Birdan n gacha bo`lgan sonlar yig`indisini hisoblovchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
```

```
int n, s = 0;
cout << "n="; cin >> n;
for (int i = 1; i <= n; i++)
s += i;
cout << s << endl;
return 0;
}
```

for sikl operatorining boshqa imkoniyatlari for sikl operatorida qavs ichidagi ifodalar bo'lmasligi mumkin, lekin ";" bo'lishi shart.

Eng sodda doimiy takrorlanuvchi sikl operatori quyidagicha:

```
for ( ; ; )
cout << "doimiy takrorlanish";
```

Agar takrorlash jarayonida bir nechta o'zgaruvchi bir vaqtda sinxron o'zgarishi lozim bo'lsa, ularni <ifoda1> va <ifoda3> da zarur bo'lgan o'rinda vergul bilan ajratib yozish mumkin.

```
#include <iostream>
using namespace std;
int main()
{
int n;
cin >> n;
for (int i = 1, j = 1; i <= n; i++, j += i)
cout << i << " " << j << endl;
return 0;
}
```

do – while sikl operatori

do - while operatorining umumiy ko'rinishi :

do *{operator;}*

while *(shart);*

Bu yerda **do** va **while** xizmatchi so`zlar. ( **shart )** sikl tanasi bajarilgandan so`ng, sikldan chiqish uchun tekshiriladigan shart. (mantiqiy ifoda).

do - while operatorning ishlash tartibi:

do xizmatchi so`zidan keyingi operatorlar bajariladi, keyin **while** xizmatchi so'zidan keyingi shart tekshiriladi. Agar shart rost (true) natija bersa **do** xizmatchi so`zidan keyingi operatorlar qayta bajariladi. Shart qayta tekshiriladi, bu jarayon shart yolg'on (false) natija berguncha takrorlanadi. Qachon while xizmatchi so'zidan keyingi shart yolg'on (false) qiymatga ega bo'lsa, boshqarilish **do - while** operatoridan keying operatorga uzatiladi.

do - while sikl operatorida sikllanib qolish DIQQAT: do - while sikl operatoridan, qachon **while** xizmatchi so'zidan keyingi **(shart)** false (yolg'on) qiymat qabul qilsa chiqiladi. Ya'ni boshqarilish **do – while** operatoridan keyingi operatorga uzatiladi. Agar **(shart)** false qiymat qabul qilmasa, do - while sikl operatoridan chiqib ketilmaydi va bu jarayon sikllanib qolish eyiladi.

1 dan 10 gacha bo'lgan sonlarni chiqaruvchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
int i = 1;
do {
cout << i << endl;
i++;
} while ( i <= 10);
return 0;
}
```

Misol. Quyidagi yig`indini hisoblovchi programma tuzilsin.

Bu programma parametrli sikl operatoridan foydalangan holda oldingi darsda tuzilgan edi. Endi do - while sikl operatori orqali programma tuzamiz va

sikl operatorlarini farqini ko`rib olamiz.

```
#include <iostream>
using namespace std;
int main()
{
float i = 1; // i - sikl uchun
float s = 0; // s - yig'indi
do {
s += 1 / i;
i++;
} while ( i <= 50);
cout << "yig`indi = " << s << endl;
return 0;
}
```

while sikl operatori

do - while operatorida siklning tanasi kamida bir marta takrorlanadi. Shu bir marta hisoblash ham yechilayotgan masalani mohiyatini buzib yuborishi mumkin. Bunday hollarda **while** sikl operatoridan foydalangan maqsadga muvofiq.

while operatorining umumiy ko'rinishi:

while (shart) {

sikl_tanasi**;**

}

while operatori sikl tanasida qanday operatorlar bo`lishi mumkin? sikl_tanasi ixtiyoriy operator yoki operatorlar majmuidan iborat bo'lishi mumkin..

while sikl operatorning ishlash tartibi:

Agar (shart) rost (**true**) qiymatga ega bo`lsa, **sikl_tanasi** bajariladi.

Qachonki shart yolg`on (**false**) qiymatga teng bo`lsa sikl tugatiladi.

Agar (shart) true qiymatga ega bo`lmasa sikl tanasi biror marta ham bajarilmaydi.

while sikl operatoridan qanday chiqiladi?

while sikl operatoridan, qachon **(shart)** false (yolg'on) qiymat qabul qilsa chiqiladi.

Ya'ni boshqarilish **while** operatoridan keyingi operatorga uzatiladi. Agar **(shart)** false qiymat qabul qilmasa, while sikl operatoridan chiqib ketilmaydi va bu jarayon sikllanib qolish deyiladi. Programmalash san'ati . **do - while** va **while** sikl operatorlarida sikl tanasi sifatida faqat bitta operator ishlatiladiga bo'lsa, bu operatorni blok orasiga { } olmasdan ham yozish mumkin. Lekin professional programmistlar har qanay xolda sikl tanasini blokka { } olib yozishni tavsiya qilishadi. Bu esa sodir bo'lishi mumkin bo'lgan mantiqiy xatoliklarni oldini oladi.

1 dan 10 gacha bo'lgan sonlarni chiqaruvchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
int i = 1;
while ( i <= 10 ) {
cout << i << endl;
i++;
}
return 0;
}
```

Misol. Quyidagi yig`indini hisoblovchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
float i = 1; // sanagich
```

```
float s = 0; // yig'indi
while ( i <= 50 ) {
s += 1 / i;
i++;
}
cout << s << endl;
return 0;
}
```

Kompyuter o'ylagan sonni topish dasturi

```
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
int x, y = 0, u = 0;
srand(time(NULL));
x = rand() % 1000 + 1;
cout << "Kompyuter o'ylagan sonni toping" << endl;
while (x != y)
{
cin >> y;
u++; if (x > y) cout << "Kompyuter o'ylagan son katta" << endl;
else if (x < y) cout << "Kompyuter o'ylagan son kichik" << endl;
}
cout << "Qoyil topdingiz!!!" << endl;
cout << "Urinishlar soni=" << u << endl;
return 0;}
```

Nazorat savollari:

1. for sikl operatorining sintaksisi qanday?
2. for sikl operatorida sikl tanasi deb nimaga aytiladi?

3. break va continue funksiyalari qo`llanilishini tushuntiring?
4. for sikl operatorida sikl parametrlari qanday toifali bo'lishi mumkin?
5. Siklning takrorlanishlar soni qanday aniqlanadi?
6. do - while operatori sikl tanasida qanday operatorlar bo`lishi mumkin?
7. do - while operatorining umumiy ko'rinishi qanday?
8. do - while operatoridan qanday chiqiladi?
9. do - while operatori ishlash tartibini tushintirib bering.
10. while operatori sikl tanasida qanday operatorlar bo`lishi mumkin?
11. while operatoridan qanday chiqiladi?
12. while operatori ishlashini tushintirib bering.

6-Mavzu: C++ da funksiya va ko’rsatkichlar.

Reja:

1. **Ko'rsatkichlar bilan ishlash**
2. **Funksiyalar bilan ishlash**

Ko'rsatkichlar bilan ishlash

Ko'rsatkich. O'zining qiymati sifatida xotira adresini ko'rsatuvchi (saqlovchi) o'zgaruvchilarga - ko'rsatkich o'zgaruvchilar deyiladi. Masalan : Ko'rsatkichning qiymati

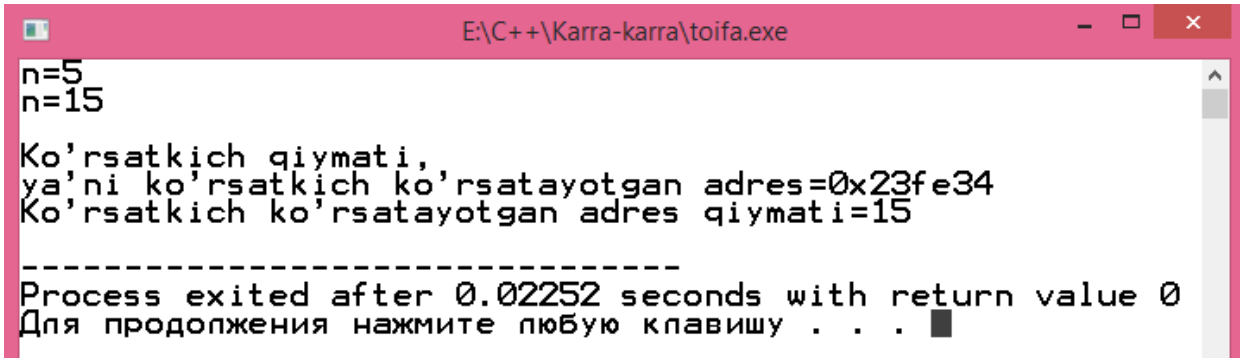
1) 0x22ff40
2) 0x22ff33
3) va xakazo kabi xotiraning aniq qismi bo'lishi mumkin.

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 oldigan * qo'yilishi shart.

Ko'rsatkichga doir oddiy misol

```
#include <iostream>
using namespace std;
int main()
{
int n = 5;
int * nPtr;
// & adresni olish amali
nPtr = &n;
cout << "n=" << n << endl;
*nPtr = 15;
cout << "n=" << n << endl;
cout << "\nKo'rsatkich qiymati,\n";
cout << "ya'ni ko'rsatkich ko'rsatayotgan adres=" << nPtr<<endl;
cout << "Ko'rsatkich ko'rsatayotgan adres qiymati=" <<*nPtr<<endl;
return 0;
}
```

Ekranda quyidagicha natija chiqariladi:

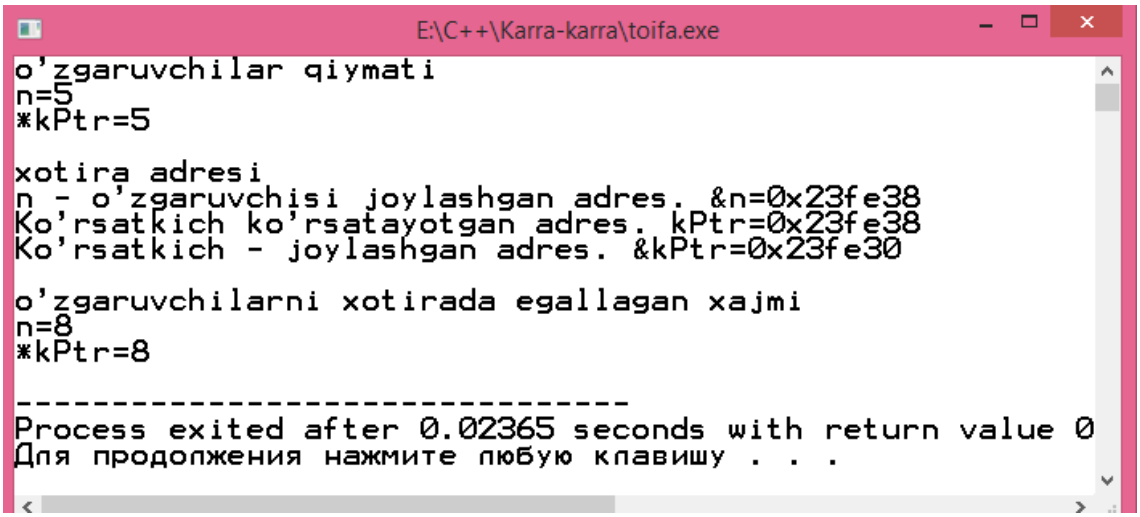


(6.1-rasm)

Ko'rsatkich bilan ishlash

```
#include <iostream>
using namespace std;
int main()
{
double n = 5;
double *kPtr;
kPtr = &n;
cout << "o'zgaruvchilar 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;
return 0;
}
```

Ekranda quyidagicha natija chiqariladi:



(6.2-rasm)

Murojaatlar

Murojaatlar e’londa ko’rsatilgan nomning sinonimi sifatida ishlatiladi, yani bitta o’zgaruvchiga xar xil nom bilan murojaat qilish mumkin. Murojaatni doimiy qiymatga ega bo’lgan ko’rsatkich deb qarash mumkin xam bo’ladi. Murojaat quyidagicha e’lon qilinadi:

<toifa> & <nom>;

Bu yerda <toifa> – murojaat ko’rsatuvchi qiymatning toifasi, ‘&’ belgisi, undan keyin yozilgan <nom>- murojaat toifasidagi nom ekanligini bildiruvchi operator.

Boshqacha aytganda ‘&’ belgisiga adresni olish amali deyiladi.

Namuna:

```
int k;
int & p = k;
// p murojaati - k o’zgaruvchisining alternativ nomi
```

Murojaat asosan funktsiyalarda adres orqali uzatiluvchi parametrlar sifatida ishlatiladi. Murojaatning ko'rsatkichdan bir nechta farqi bor:

1) Murojaatni e'lon qilishda initsializatsiya qilish kerak
2) Murojaatning qiymatini o'zgartirib bo'lmaydi, ko'rsatkichning qiymatini, ko'rsatib turgan adresini o'zgartirish mumkin.

Funksiya va massivga ko'rsatkich orqali murojaatni keyingi mavzularda o'rganamiz.

Funksiyalar bilan ishlash

Dasturlash mobaynida bir xil ifodalarni, hisoblash jarayonlarini qayta – qayta hisoblashga to`g`ri keladi.

Dasturlash tillarida, kompyuter hotirasini va dasturchining vaqtini tejash maqsadida, bunday takkorlanuvchi jarayonlarni dasturda ajratib yozib, unga asosiy daturdan, boshqa funksiyalardan murojaat qilish imkoniyatlari keltirilgan.

Dasturning istalgan qismidan murojaat qilib, bir necha bor ishlatish mumkin bo`lgan operatorlar guruhiga funksiya deyiladi.

C++ funksiyalar tili deyiladi. Chunki dasturda kamida bitta main funksiyasi bo'ladi.

Asosiy dastur, asosiy funksiya deganda aynan manashu **main** funksiyasini tushunamiz.

Asosiy programmadan (yoki chaqiruvchi funksiyadan) xech qanday parametr qabul qilib olmaydigan funksiyalarga, parametrsiz funksiyalar deyiladi.

Parametrsiz funksiyaning o'zi ham 2 xil bo'lishi mumkin:
1) Asosiy programmaga (yoki chaqiruvchi funksiyaga) natijani qaytaruvchi.
2) void turidagi funksiya bo'lib, asosiy programmadan (yoki chaqiruvchi funksiyadan) xech qanday parametr qabul qilib olmaydi xam, asosiy programmaga xech qanday natija qaytarmaydi ham.

Parametrsiz funksiyaga murojaat qilishda dastur tanasida funksiya nomi yoziladi.

Dasturda funksiya nomi operatorlar kabi ishlatiladi. Parametrsiz funksiyada asosiy dasturning barcha global o`zgaruvchilaridan foydalanish mumkin.

Global o`zgaruvchilar

Ham asosiy programmada, ham funksiyada ishlatish mumkin bo'lgan o`zgaruvchilar global o'zgaruvchilar deyiladi. Global o'zgaruvchilar asosiy programmada e`lon qilishi kerak.

Lokal o`zgaruvchilar

Faqat funksiyada ishlatish mumkin bo'lgan o'zgaruvchilarga local o'zgaruvchilar deyiladi. Ular funksiyada e`lon qilinadi. Funksiyada yana bir nechta ichki funksiyalardan foydalanish mumkin.

Blok ichida e'lon qilingan o'zgaruvchilar, shu blok uchun lokal o'zgaruvchilar hisoblanadi. Bu o'zgaruvchilardan faqat blok ichida foydalanish mumkin.

Parametrli funksiyalar

Asosiy dasturdan (funksiyadan) chaqiriluvchi funksiyaga uzatilgan parametrlarni qabul qilib qayta ishlovchi funksiyalar parametrli funksiyalar deyiladi.

Qiymat parametrlar

Qiymat parametrlar – asosiy dasturdan funksiyaga uzatiladigan o'zgaruvchilar qiymatlarni qabul qilib oluvchi parametrlar. Funksiyaga murojaat qilinganida qiymat parametrlari uchun xotiradan joy ajratiladi. Funksiya tugaganida qiymat parametrlari uchun ajratilgan xotira bo'shatiladi.

Ko'rsatkich parametrlar

Ko'rsatkich parametrlar - asosiy dasturdan funksiyaga uzatiladigan o'zgaruvchilarning xotiradagi adresini qabul qilib oluvchi parametrlar. Ko'rsatkich parametrlari ustida bajarilgan har qanday o'zgarish, asosiy dasturdagi o'zgaruchilarning xotira adresida sodir bo'ladi. (Ya'ni asosiy dasturdagi o'zgaruvchi qiymati o'zgaradi)

Eslatma: Qiymat parametrlari va ko'rsatkich parametrlar toifasi, asosiy dasturdagi qiymati uzatilayotgan o`zgaruvchilar toifasi bilan bir xil bo`lishi lozim.

Funksiyadan chiqish

Ixtiyoriy funksiyadan chiqish uchun **return** xizmatchi so'zi ishlatiladi. **Misol:** To'g'ri burchakli uchburchakning katetlari berilgan. (3, 4), (6, 8), (12, 5) bo'lgan xollar uchun uchburchak gipotenuzasini hisoblovchi programma tuzilsin.

1) Parametrli funksiya

```
// Muallif: Jasur Baratov
// Sana : 1 noyabr 2011 yil
// Maqsad : Parametrli funksiyani o'rganish
#include <iostream>
#include <math.h>
using namespace std;
// funksiya prototipi
float hisobla(float , float );
int main()
{
float c;
c = hisobla(3, 4);
cout << c << endl;
```

```
c = hisobla(6, 8);
cout << c << endl;
c = hisobla(12, 5);
cout << c << endl;
return 0;
}
float hisobla(float a, float b)
{
//lokal o'zgaruvchi
float natija;
natija = sqrtf(a*a + b*b);
return natija;
}
```

2) void toifasidagi parametrli funksiya

```
// Muallif: Jasur Baratov
// Sana : 1 noyabr 2018 yil
// Maqsad : void toifasidagi parametrli funksiyani o'rganish
#include <iostream>
#include <math.h>
using namespace std;
// funksiya prototipi
void hisobla(float , float );
int main()
{
hisobla(3, 4);
hisobla(6, 8);
hisobla(12, 5);
return 0;
}
```

```
void hisobla(float a, float b)
{
float c;
c = sqrtf(a*a + b*b);
cout << c << endl;
}
```

Global va lokal o'zgaruvchilarga murojaatni o'rganish

```
// Muallif : Jasur Baratov
// Sana : 05.11.2011
// Maqsad : Global va lokal o'zgaruvchilarga murojaatni
// o'rganish
#include <iostream>
int x = 5; // global o'zgaruvchi
int main()
{
int x = 9; // lokal o'zgaruvchi
std::cout << "lokal x=" << x << std::endl;
std::cout << "global x=" << ::x << std::endl;
return 0;
}
```

Kiritilgan n sonini 3 - darajasini hisoblovchi funksiya tuzilsin

```
// Sana : 04.12.2018
// Maqsad : Funksiyaga ko'rsatkich parametrlari
// orqali murojaatni o'rganish
#include <iostream>
using namespace std;
void kub (int *);
```

```
int main()
{
int n;
cout << "n="; cin >> n;
kub (&n);
cout << "n ning qiymati =" << n << endl;
return 0;
}
void kub (int *nPtr)
{
*nPtr = *nPtr * *nPtr * *nPtr;
}
```

Ikkita son yig'indisini funksiya orqali hisoblovchi programma tuzilsin

```
// Muallif : Jasur Baratov
// Sana : 04.12.2011
// Maqsad : Funksiyaga qiymat va ko'rsatkich parametrlari
// orqali murojaatni o'rganish
#include <iostream>
using namespace std;
// funksiya prototipi
int sum(int , int);
void sum(int , int, int *);
int sum(int *, int *);
void sum(int *, int *, int *);
int main()
{
int a, b, c;
cout << "a="; cin >> a;
```

```
cout << "b="; cin >> b;
c = sum(a, b);
cout << "1-sul natijasi=" << c << endl;
sum(a, b, &c);
cout << "2-sul natijasi=" << c << endl;
c = sum(&a, &b);
cout << "3-usul natijasi=" << c << endl;
sum(&a, &b, &c);
cout << "4-usul natijasi=" << c << endl;
return 0;
}
// 1 - usul
int sum(int son1, int son2)
{
int natija;
natija = son1 + son2;
return natija;
}
// 2 - usul
void sum(int son1, int son2,int *natija)
{
*natija = son1 + son2;
}
// 3 - usul
int sum(int *son1, int *son2)
{
int natija;
natija = *son1 + *son2;
return natija;
}
```

```
// 4 - usul
void sum(int *son1, int *son2,int *natija)
{
*natija = *son1 + *son2;
}
```

Nazorat savollari:

1. Funksiya deb nimaga aytiladi?
2. Parametrsiz funksiyaga murojaat qanday amalgam oshiriladi?
3. Qanday o`zgaruvchilar global o`zgaruvchilar deyiladi?
4. Qanday o`zgaruvchilar lokal o`zgaruvchilar deyiladi?
5. Parametrli funksiya deb qanday funksiyalarga aytiladi?
6. Qanday parametrlar qiymat parametrlar deyiladi?
7. Qanday parametrlar ko'rsatkich parametrlar deyiladi?
8. Funksiyadan chiqish uchun qaysi operatordan foydalaniladi?

7-Mavzu: Rekursiv funksiyalar.
Reja:

1. **Rekursiv jarayon nima**
2. **Rekursiv funksilari**
3. **Rekursiv funksiya parametrlari**

Kalit so'zlar:*, ro'yxat, manzil, nolinchi ko'rchsatkich, tugun, adres olish &, bo'shatish, ko'rsatkich, virtual destruktor, xotira, xotira chiqishi, destruktor, toifani o'zlashtirish, resurslar chiqishi, a'zo destruktori.*

Joylashtiriladigan (inline) funksiyalar

Kompilyator ishlashi natijasida har bir funksiya mashina kodi ko‘rinishida bo‘ladi. Agar programmada funksiyani chaqirish ko‘rsatmasi bo‘lsa, shu joyda funksiyani adresi bo‘yicha chaqirishning mashina kodi shakllanadi. Odatda funksiyani chaqirish protsessor tomonidan qo‘shimcha vaqt va xotira resurslarini talab qiladi. SHu sababli, agar chaqiriladigan funksiya hajmi unchalik katta bo‘lmagan hollarda, kompilyatorga funksiyani chaqirish kodi o‘rniga funksiya tanasini o‘zini joylashtirishga ko‘rsatma berish mumkin. Bu ish funksiya prototipini inline kalit so‘zi bilan e’lon qilish orqali amalga oshiriladi. Natijada hajmi oshgan, lekin nisbatan tez bajariladigan programma kodi yuzaga keladi.

Funksiya kodi joylashtiriladigan programmaga misol.

```
#include <iostream.h>
inline int Summa(int,int);
int main()
{
 int a=2,b=6,c=3;
 char yangi_qator='\n';
 cout<<Summa(a,b)<<yangi_qator;
 cout<<Summa(a,c)<<yangi_qator;
 cout<<Summa(b,c)<<yangi_qator;
 return 0;
```

```
}
int Summa(int x,int y)
{
 return x+y;
}
```

Keltirilgan programma kodini hosil qilishda Summa() funksiyasi chaqirilgan joylarga uning tanasidagi buyruqlar joylashtiriladi.

Rekursiv funksiyalar

YUqorida qayd qilingandek *rekursiya* deb funksiya tanasida shu funksiyaning o‘zini chaqirishiga aytiladi. Rekursiya ikki xil bo‘ladi:

1) *oddiy* - agar funksiya o‘z tanasida o‘zini chaqirsa;

vositali - agar birinchi funksiya ikkinchi funksiyani chaqirsa, ikkinchisi esa o‘z navbatida birinchi funksiyani chaqirsa.

Odatda rekursiya matematikada keng qo‘llaniladi. CHunki aksariyat matematik formulalar rekursiv aniqlanadi. Misol tariqasida faktorialni hisoblash formulasini

$$n!=\begin{cases}1, & \text{arap } n=0;\\ n*(n-1)!, & \text{arap } n>0,\end{cases}$$

va sonning butun darajasini hisoblashni ko‘rishimiz mumkin:

$$x^n=\begin{cases}1, & \text{arap } n=0;\\ x*x^{n-1}, & \text{arap } n>0.\end{cases}$$

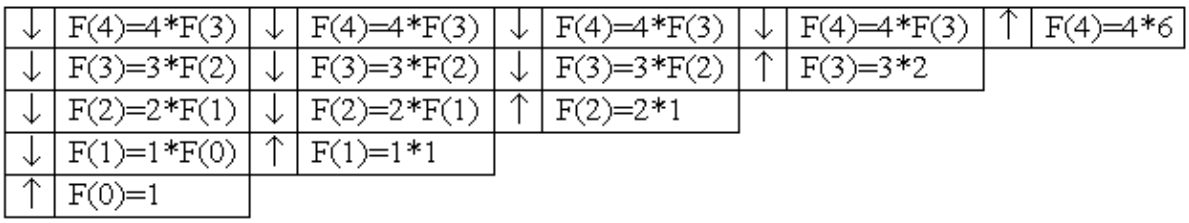
Ko‘rinib turibdiki, navbatdagi qiymatni hisoblash uchun funksiyaning «oldingi qiymati» ma’lum bo‘lishi kerak. S++ tilida rekursiya matematikadagi rekursiyaga o‘xshash. Buni yuqoridagi misollar uchun tuzilgan funksiyalarda ko‘rish mumkin. Faktorial uchun:

```
long F(int n)
{
 if(!n) return 1;
 else return n*F(n-1);
}
```

Berilgan haqiqiy x soning n- darajasini hisoblash funksiyasi:

```
double Butun_Daraja(double x, int n)
{
 if(!n) return 1;
 else return x*Butun_Daraja(x,n-1);
}
```

Agar faktorial funksiyasiga n>0 qiymat berilsa, quyidagi holat ro‘y beradi: shart operatorining else shoxidagi qiymati(n qiymati)stekdaeslab qolinadi. Hozircha qiymati noma’lum n-1 faktorialni hisoblash uchun shu funksiyaning o‘zi n-1 qiymati bilan bilan chaqiriladi. O‘z navbatida, bu qiymat ham eslab qolinadi (stekka joylanadi) va yana funksiya chaqiriladi va hakoza. Funksiya n=0 qiymat bilan chaqirilganda if operatorining sharti (!n) rost bo‘ladi va «return 1;» amali bajarilib, ayni shu chaqirish bo‘yicha 1 qiymati qaytariladi. SHundan keyin «teskari» jarayon boshlanadi - stekda saqlangan qiymatlar ketma-ket olinadi va ko‘paytiriladi: oxirgi qiymat - aniqlangandan keyin (1), u undan oldingi saqlangan qiymatga 1 qiymatiga ko‘paytirib F(1) qiymati hisoblanadi, bu qiymat 2 qiymatiga ko‘paytirish bilan F(2) topiladi va hakoza. Jarayon F(n) qiymatini hisoblashgacha «ko‘tarilib» boradi. Bu jarayonni, n=4 uchun faktorial hisoblash sxemasini 5.2-rasmda ko‘rish mumkin:



14.1-rasm. 4! hisoblash sxemasi

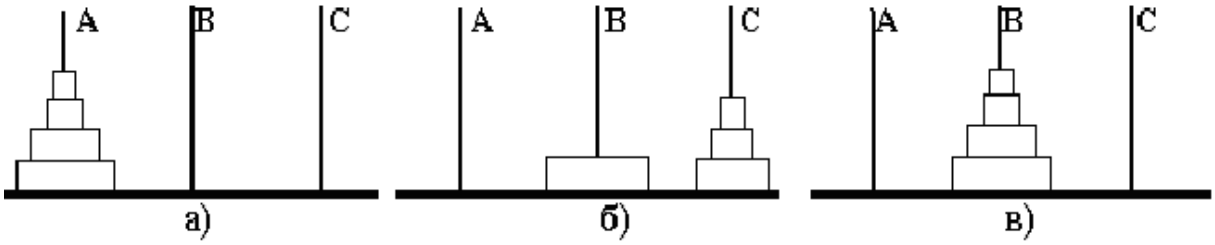
Rekursiv funksiyalarni to‘g‘ri amal qilishi uchun rekursiv chaqirishlarning to‘xtash sharti bo‘lishi kerak. Aks holda rekursiya to‘xtamasligi va o‘z navbatida funksiya ishi tugamasligi mumkin. Faktorial hisoblashida rekursiv tushishlarning to‘xtash sharti funksiya parametri n=0 bo‘lishidir (shart operatorining rost shoxi).

Har bir rekursiv murojaat qo‘shimcha xotira talab qiladi - funksiyalarning lokal ob’ektlari (o‘zgaruvchilari) uchun har bir murojaatda stekdan yangidan joy ajratiladi. Masalan, rekursiv funksiyaga 100 marta murojaat bo‘lsa, jami 100 lokal

ob’ektlarning majmuasi uchun joy ajratiladi. Ayrim hollarda, ya’ni rekursiyalar soni etarlicha katta bo‘lganda, stek o‘lchami cheklanganligi sababli (real rejimda 64Kb o‘lchamgacha) u to‘lib ketishi mumkin. Bu holatda programma o‘z ishini «Stek to‘lib ketdi» xabari bilan to‘xtadi.

Quyida, rekursiya bilan samarali echiladigan «Xanoy minorasi» masalasini ko‘raylik.

Masala. Uchta A, B, C qoziq va n-ta har xil o‘lchamli xalqalar mavjud. Xalqalarni o‘lchamlari o‘sish tartibida 1 dan n gacha tartib-langan. Boshda barcha xalqalar A qoziqqa 5.3a -rasmdagidek joylash-tirilgan. A qoziqdagi barcha xalqalarni B qoziqqa, yordamchi S qoziqdan foydalangan holda, quyidagi qoidalarga amal qilgan holda o‘tkazish talab etiladi: xalqalarni bittadan ko‘chirish kerak va katta o‘lchamli xalqani kichik o‘lchamli xalqa ustiga qo‘yish mumkin emas.



14.2-rasm. Xanoy minorasi masalasini echish jarayoni

Amallar ketma-ketligini chop etadigan («Xalqa q dan r ga o‘tkazilsin» ko‘rinishida, bunda q va r - 5.3-rasmdagi A,V yoki S xalqalar). Berilgan n ta xalqa uchun masala echilsin.

Ko‘rsatma: xalqalarni A dan B ga to‘g‘ri o‘tkazishda 5.3b –rasmlar-dagi holat yuzaga keladi, ya’ni n xalqani A dan B o‘tkazish masalasi n-1 xalqani A dan S ga o‘tkazish, hamda bitta xalqani A dan B o‘tkazish masalasiga keladi. Undan keyin S qoziqdagi n-1 xalqali A qoziq yordamida B qoziqqa o‘tkazish masalasi yuzaga keladi va hakoza.

```
#include <iostream.h>
void Hanoy(int n,char a='A',char b='B',char c='C')
{
 if(n)
```

```
 {
  Hanoy(n-1,a,s,b);
  cout<<”Xalqa”<< a<<” dan ”<<b<<” ga o’tkazilsin\n”;
  Hanoy(n-1,c,b,a);
  }
}
int main()
{unsigned int Xalqalar_Soni;
 cout<<”Hanoy minorasi masalasi”<<endl;
 cout<<”Xalqalar sonini kiriting: ”;
 cin>>Xalqalar_Soni;
 Hanoy(Xalqalar_Soni);
 return 0;
}
```

Xalqalar soni 3 bo‘lganda (Xalqalar_Soni=3) programma ekranga xalqalarni ko‘chirish bo‘yicha amallar ketma-ketligini chop etadi:

```
Xalqa A dan B ga o’tkazilsin
Xalqa A dan C ga o’tkazilsin
Xalqa B dan C ga o’tkazilsin
Xalqa A dan B ga o’tkazilsin
Xalqa C dan A ga o’tkazilsin
Xalqa C dan B ga o’tkazilsin
Xalqa A dan B ga o’tkazilsin
```

Rekursiya chiroyli, ixcham ko‘ringani bilan xotirani tejash va hisoblash vaqtini qisqartirish nuqtai-nazaridan uni imkon qadar iterativ hisoblash bilan almashtirilgani ma’qul. Masalan, x haqi-qiy sonining n-darajasini hisoblashning quyidagi echim varianti nisbatan kam resurs talab qiladi (n- butun ishorasiz son):

```
double Butun_Daraja(double x, int n)
{
 double p=1;
```

```
 for(int i=1; i<=n; i++)p*=x;
 return p;
}
```

Ikkinchi tomondan, shunday masalalar borki, ularni echishda rekursiya juda samarali, hattoki yagona usuldir. Xususan, grammatik tahlil masalalarida rekursiya juda ham o‘ng‘ay hisoblandi.

ch05/account.cpp

```
1 #include <iostream>
2
3using namespace std;
4 5 /**
6Withdraws the amount from the given balance, or withdraws
7a penalty if the balance is insufficient.
8@param balance the balance from which to make the withdrawal 9@param amount the amount to withdraw
10 */
11void withdraw(double& balance, double amount)
12 {
13constdouble PENALTY = 10;
14if (balance >= amount)
15 {
16balance = balance - amount;
17 }
18else
19 {
20balance = balance - PENALTY;
21 }
22 }
23
24int main()
```

```
25 {
26double harrys_account = 1000;
27double sallys_account = 500;
28 withdraw(harrys_account, 100);
29// Now harrys_account is 900
30withdraw(harrys_account, 1000); // Insufficient funds
31// Now harrys_account is 890
32 withdraw(sallys_account, 150);
33 cout <<"Harry's account: "<< harrys_account << endl; 34 cout <<"Sally's account:
"<< sallys_account << endl; 35
36return0;
37 }
```

program run

Harry's account: 890 Sally's account: 350

Nazorat savollari

1. C++da funksiya qanday ishlaydi?
2. funksiyaga kutubxona kerakmi?
3. Rekursiv funksiya nima?
4. Rekursiv funksiya parametrlari.
5. For operatori funksiyada qanday ishlatiladi?
6. Matematik funksiyalar qanday ishlaydi?
7. Funksiya parametrlar nima?

8-Mavzu: C++ tilida massivlar.
Reja:

1. **Bir o’lchamli massivlar**
2. **Ko’p o’lchamli massivlar**

Bir o’lchamli massivlar

Massiv - bu bir xil toifali, chekli qiymatlarning tartiblangan to`plamidir.

Massivlarga misol qilib matematika kursidan ma`lum bo`lgan vektorlar, matritsalarni ko`rsatish mumkin.

**Massiv bir o`lchamli deyiladi**, agar uning elementiga bir indeks orqali murojaat qilish mumkin bo`lsa.

Bir o`lchamli massivni e`lon qilish quyidagicha bo`ladi:

<toifa> <massiv_nomi> [elementlar_soni] = { boshlang'ich qiymatlar };

Quyida massivlarni e`lon qilishga bir necha misollar keltirilgan:

1) float a[5];

2) int m[6];

3) bool b[10];

1) a elementlari haqiqiy sonlardan iborat bo`lgan, 5 ta elementdan tashkil topgan massiv. Indekslari esa 0 dan 4 gacha bo`lgan sonlar

float a[5];					
Massiv elementilari	a[0]	a[1]	a[2]	a[3]	a[4]
qiymati	4	-7	5.5	6	8

Massiv elementlariga murojaat qilish oddiy o`zgaruvchilarga murojaat qilishdan biroz farq qiladi.

Massiv elementiga murojaat qilish uning indeksi orqali bo`ladi.

a[1] = 10; a massivining 1 – elementi 10 qiymat o’zlashtirsin;

cin >> a[2]; a massivining 2 – elementi kirtilsin;

cout << a[3]; a massivining 3 – elementi ekranga chiqarilsin;

Massivni e'lon qilishda uning elementlariga boshlang'ich qiymat berish mumkin va buning bir nechta usuli mavjud. 1) O'lchami ko'rsatilgan massivni to'liq initsializatsiyalash.

int k[5] = { 2, 3, 7, 8, 6};

Bu yerda 5 ta elementdan iborat bo'lgan k massivi e'lon qilingan va massivning barcha elemantlariga boshlang'ich qiymat berilgan.

2) O'lchami ko'rsatilgan massivni to'liqmas initsializatsiyalash.

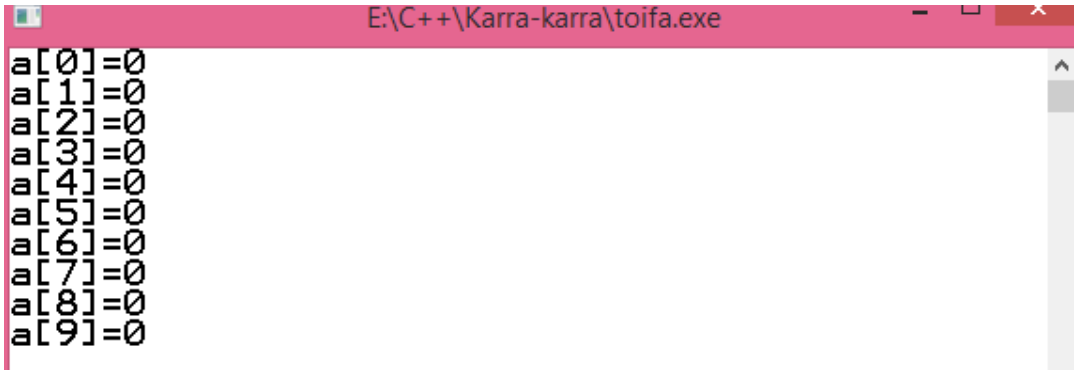
int k[5] = { 2, 3, 7 };

Bu yerda 5 ta elementdan iborat bo'lgan k massivi e'lon qilingan va massivning dastlabki 3 ta elemantlariga boshlang'ich qiymat berilgan.

O'lchami ko'rsatilgan massivning barcha elementlariga boshlang'ich qiymat 0 berish

```
#include <iostream>
using namespace std;
int main()
{
int a[10] = { 0 };
//massivning barcha elementlariga 0 qiymat berish
for (int i = 0; i < 10; i++)
cout << "a[" << i << "]=" << a[i] << endl;
return 0;
}
```

Ekranga quyidagicha natija chiqariladi:



(7.1-rasm)

Agar massiv elementlariga boshlang'ich qiymatlar berilmasa xatolik sodir bo'lishi mumkin.

n ta elementdan tashkil topgan massiv berilgan. Shu massiv elementlari yig'indisini chiqatuvchi programma tuzilsin. (n <= 10)

```
#include <iostream>
using namespace std;
int main()
{
int a[10] = { 0 }; // a massivini e'lon qilish
int n; // massiv elamentlari soni
int s = 0; // massiv elementlari yig'indisini hisoblash
uchun
cout << "n="; cin >> n;
for (int i = 0; i < n; i++)
{
cout << "a[" << i << "]=";
cin >> a[i];
s += a[i];
}
cout << "Massiv elementlari yig`indisi = " << s <<
endl;
return 0;
}
```

Ko’p o’lchamli massivlar

Assalomu alaykum bo`lajak dasturchi! Yangi mavzuni boshlashdan oldin, oldingi mavzuni qisqacha takrorlab olsak. Quyidagi savollarga og`izaki yoki yozma javob bering. Javob qanchalik to`g`riligini tekshirish uchun savolni bir marta bosing. Bir o`lchamli massivlar uchun ishlatilgan o`zgaruvchilar, bir xil jinsdagi berilganlarni xotirada saqlash uchun foydalaniladi. Ikki o`lchamli massivlarda esa,

satr va ustunlar orqali bir xil jinsdagi qiymatlarni ikki o`lchamli o`zgaruvchilar ichida saqlash uchun foydalaniladi.

Ikki o`lchamli statik massivlarni e`lon qilish.

toifa massiv_nomi [massiv_satrlari_soni][massiv_ustunlari_soni]; 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}				

(7.2-rasm)

A matritsa 3 ta satr, 3 ta ustunga ega;

B matritsa 2 ta satr, 4 ta ustunga ega;

Ikki o'lchamli massivlarda 1 - indeks satrni, 2 - indeks ustunni bildiradi. Birinchi satrning dastlabki elementi a10 – a biru nol element deb o`qiladi. a o`n deyilmaydi.

m ta satr va n ta ustunga ega bo`lgan massivga (mxn) o`lchamli massiv deyiladi.

Agar m=n (satrlar va ustunlar soni teng) bo'lsa kvadrat massiv deyiladi.

Ko'p o'lchamli massivlarni initsializatsiyalash misollar:

int a[2][2]={1,2,7,3};

int b[2][3]={ {0,1,2}, {3,4,5} };

Massivlarni qo`llanilishiga misol keltiradigan bo`lsak, satrlar talabalarni, ustunlar fanlardan olgan baholarini bildirsin. Ya`ni m ta talaba, n ta fan. n - ustunga

talabalarning o`rtacha baholari hisoblanib, shu asosida stipendiya bilan ta`minlansin. Va hakazo, bunga o`xshash ko`plab misollar keltirish mumkin. Bu masalalarga to`xtalishdan oldin bir ikkita oddiy masalar bilan tanishib chiqaylik.

1 - Masala. A(mxn) matritsa berilgan. Shu matritsa elementlarini kirituvchi va ekranga jadval ko`rinishida chiqaruvchi programma tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{ int m, n, a[10][10];
cout << "Satrlar sonini kiriting \nm=";
cin >> m; cout << "Ustunlar sonini kiriting \nn=";
cin >> n; cout <<"Massiv elementlarini kiriting \n";
for(int satr = 0; satr < m ; satr++)
for(int ustun = 0; ustun < n; ustun++)
{ cout << "a[" << satr << "][" << ustun << "]=";
cin >> a[satr][ustun];
 }
// matritsani jadval shaklida chiqarish for(int satr = 0; satr < m; satr++)
{ for(int ustun = 0; ustun < n; ustun++)
cout << a[satr][ustun] << "\t"; cout<<"\n"; }
return 0;
}
```

Funksiya parametri sifatida massivni jo'natish va funksiya natijasi sifatida massivni olish ham mumkin. Funksiyaga matritsani uzatishda matritsa nomi bilan uning satrlar va ustunlar sonini ham jo'natish kerak bo'ladi.

Nazorat savollari:

1. Massiv nima?
2. Bir o'lchamli massiv deb nimaga aytiladi?
3. Massiv elementi nima?

4. Massiv indeksi nima?

5. Bir o'lchamli massiv qanday e`lon qilinadi?

6. Massiv elementlari soni qanday aniqlanadi?

7. Ko`p o`lchamli massivlarni e`lon qilishning qanday usullarini bilasiz?

8.Qanday massivlarga kvadrat massivlar deyiladi?

9. Bir o'lchamli va ikki o'lchamli massivlar orasidagi farq nimada?

10. Statistik massivlar bilan ishlashning kamchiliklari nimada?

11. Ikki o'lchamli dinamik massivlar bilan ishlash qanday amalga oshiriladi?

9-Mavzu: C++ dasturlash tilining aralash toifasi.
Reja:

1. **Char toifasi**
2. **Satrlar bilan ishlash**

Nazariy qism. Belgini (simvolni) saqlash uchun mo'ljallangan o'zgaruvchilarga belgili o'zgaruvchilar deyiladi. C++ tilida bu o'zgaruvchilar uchun **char** toifasi keltirilgan. char toifasidagi o'zgaruvchi ASCII kodidagi 255 ta belgidan ixtiyoriy birisi bo'lishi mumkin.

ASCII kodi jadvali

Belgili o'zgaruvchilarni e'lon qilish quyidagicha bo'ladi:

char c, s;

belgili o'zgaruvchilar apostraf ichida yoziladi.

a='q'; c='*'; s='/';

char toifasini oshkor ravishda butun toifaga o'tkazish orqali, berilgan belgiga mos ASCII kodini aniqlash mumkin.

ASCII kodi jadvaliga e'tibor bering. Katta va kichik lotin harflari alifbo tartibida joylashtirilgan. Bu esa sikl orqali char toifasini tashkil qilish imkoniyatini beradi.

Sikl orqali lotin harflarini chiqarish.

```
#include <iostream>
using namespace std;
```

```
int main()
{ cout<<"KATTA LOTIN HARFLAR"<<endl;
for (int i=65; i<=90; i++)
        cout<<i<<"->"<<char(i)<<endl;
cout<<"KATTA LOTIN HARFLAR"<<endl;
for (int i=97; i<=122; i++)
        cout<<i<<"->"<<char(i)<<endl;
        return 0;
}
```

Yoki siklni quyidagicha ham tashkil qilish mumkin:

```
for  (char c =  'a'; c <='z'; c++)
cout <<   (int)  c << "  " << c << endl;
```

C++ da satr. C++ da satr deb - satr oxiri ('\0') belgisi bilan tugaydigan belgilar massiviga aytiladi. Demak C++ da satr - birinchi belgiga o'rnatilgan ko'rsatkich ekan. Chunki massiv ham, birinchi elementiga o'rnatilgan ko'rsatkichdir.

Satrni belgilar massivi ko'rinichida yoki char toifasidagi ko'rsatkich sifatida e'lon qilish mumkin.

```
char satr1[ ] = "dastur.uz"; char * satr2[ ] = "dastur.uz";
```

satrl massivi 10 ta elementdan, 'd', 'a', 's', 't', 'u', 'r', 'u', 'z', '\0' belgilaridan iborat.

Satrni kiritishda cin.getline funksiyasidan foydalanish mumkin.

```
cin.getline(satr, satruzunligi);
char satr[15]; cin.getline(satr, 15);
```

getline funksiyasining 2 - parametri sifatida sizeof funksiyasidan foydalanish tavsiya etiladi.

```
cin.getline(satr, sizeof(satr));
```

sizeof va strlen funksiyalarining farqi

```
#include <iostream>
using namespace std;
int main()
```

```
{
char s[20];
cout <<"Satr kiriting" <<endl; cin.getline(s,  sizeof(s));
cout << "sizeof(s)=" << sizeof(s)<<endl;
cout << "strlen(s)=" << strlen(s)<<endl;
return 0;
}
```

Belgilarni qayta ishlovchi funksiyalar

Quyidagi funksiyalardan foydalanish uchun ctype.h sarlavha faylini progarmmaga qo'shish kerak.

Funksiya prototipi	Funksiya tavsifi
int isdigit(int c)	Agar c raqam bo'lsa true, aks xolda false qiymat qaytaradi.
int isalpha(int c)	Agar c harf bo'lsa true, aks xolda false qiymat qaytaradi.
int isalnum(int c)	Agar c raqam yoki harf bo'lsa true, aks xolda false qiymat qaytaradi.
int islower(int c)	Agar c kichik harf bo'lsa true, aks xolda false qiymat qaytaradi.
int isupper(int c)	Agar c katta harf bo'lsa true, aks xolda false qiymat qaytaradi.
int tolower(int c)	Agar c katta harf bo'lsa kichik harf qaytariladi, aks xolda tolower argumentni o'zgarish qaytaradi.
int toupper(int c)	Agar c kichik harf bo'lsa katta harf qaytariladi, aks xolda toupper argumentni o'zgarish qaytaradi.

Toifalarni o'zgartirish funksiyalari

Quyidagi funksiyalardan foydalanish uchun stdlib.h sarlavha faylini progarmmaga qo'shish kerak.

Funksiya prototipi	Funksiya tavsifi
`double atof(const char *c)`	c satrini **double** toifasiga o'zgartiradi.
`int atoi(const char *c)`	c satrini **int** toifasiga o'zgartiradi.
`int atol(const char *c)`	c satrini **long int** toifasiga o'zgartiradi.
`double strtod(const char *c, char **endPtr)`	c satrini **double** toifasiga o'zgartiradi.
`char * itoa(int n, char *satr, int radix)`	n sonini radix sanoq sistemasida satr o'zgaruvchisiga o'zlashtiradi.

Standart kutubxonadagi string sinfi

C++ da satrlar bilan ishlashni qulaylashtirish uchun string sinfi kiritilgan. string sinfi satrlarida satr oxirini '\0' belgisi belgilamaydi.

String sinfidan foydalanish uchun qaysi sarlavha faylini dasturga qo'shish kerak?

Standart kutubxonadagi string sinfidan foydalanish uchun <string> sarlavha faylini dasturga qo'shish kerak.

Lekin ba'zi eski kompilyatorlarda <cstring.h> yoki <bstring.h> sarlavha faylini qo'shish kerak bo'ladi. Oddiy eski usuldagi satrlar bilan ishlash uchun esa, <string.h> sarlavha fayli qo'shiladi.

Eng afzali, o'zingiz ishlatayotgan kompilyator bilan yaxshilab tanishib chiqing. Satrlar bilan ishlovchi asosiy funksiyalar bilan tanishib chiqamiz.

```
Satr xususiyatlarini aniqlash uchun quyidagi funksiyalardan
foydalanish mumkin:
  unsigned int size() const;         // satr o'lchami
  unsigned int length() const;       // satr elementlar soni
  unsigned int max_size() const;     // satrning maksimal uzunligi
  unsigned int capacity() const;     // satr egallagan xotira hajmi
```

void clear(); - funksiyasi satrni tozalash (to'liq o'chirish) uchun ishlatiladi.

bool empty() const; - funksiyasi satrni bo'shligini tekshirish uchun ishlatiladi. Agar satr bo'sh bo'lsa, true qiymat qaytaradi.

Satrning biror qismidan nusxa olish

string& assign (const string &str);

Satrga str o'zgaruvchisidagi satrning to'liq nusxasini olish.

string& assign (const string& str, size_t pos, size_t n);

Satrga str o'zgaruvchisidagi satrning pos o'rindagi belgisidan boshlab n ta belgi nusxasini olish.

string& assign (const char* s, size_t n);

string toifasidagi satrga char toifasidagi satrning n ta belgisi nusxasini olish.

```
string si, s2, s3;
si = "dastur.uz";
s2.assign(si); // s2 = "dastur.uz"
s3.assign(si, 0, 6); // s3 = "dastur"
```

append funksiyasining assigndan farqi satrning davomiga satr qismining qo'shishidadir.

string& append (const string& str);

string& append (const string& str, size_t pos, size_t n);

string& append (const char* s, size_t n);

Satrdan nusxa olish

```
#include <iostream>
#include <string>
using namespace std;
int main() {
string s1, s2, s3;
s1= "GULBAHOR";
s2.assign(s1, 0, 3);
s3.assign(s1, 3,5);
cout << s1<< endl;
cout << s2 << endl;
cout << s3 << endl;
si = s3 + s2;
cout << si << endl;
return 0;
}
```