O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

FARG’ONA FILIALI

“ KOMPYUTER INJINIRINGI ” fakulteti

“DASTURIY INJINIRING” kafedrasi

“C++ da Dasturlash”

fanidan

MARUZALAR MATNI

Farg’ona - 2015

Ushbu o’quv uslubiy ko’rsatma O’zbekiston Respublikasi Oliy va O’rta Maxsus ta’lim vazirligining Oliy O’quv yurtlari boshqarmasi 201__ yil _____da tasdiqlangan (ro’yxat t/r _____) namunaviy dasturi va “Dasturiy injiniringi” kafedrasida ishlab chiqilgan o’quv dasturi asosida tuzilgan va unga mos keladi

O’quv uslubiy ko’rsatma 5330500- Kompyuter injiniringi yo’nalishi talabalari uchun mo’ljallangan.

Tuzuvchilar: ass. Asrayev M

1-Ma’ruza. Dasturlashga kirish, dasturlasning asosiy tushunchalari

Reja:

1. C++ dasturlash tili
2. C++ tilida standart funksiyalarning yozilishi

Kalit so’zlar: *kommunikatsiya, dasturiy ta’minot, tashxis, teskari aloqa, loyihalash, foydalanuvchi interfeysi, foydalanuvchi, aniqlik, Stereotip, buyurtmachi, dasturchi, samaradorlik.*

C++ dasturlash tili

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/C++ algoritmik tilining alifbosi:

1. 26 ta lotin va 32 ta kirill harflari (katta va kichik);
2. 0 dan 9 gacha bo’lgan arab raqamlari;
3. Maxsus belgilar: - + * / : ; . , % ? ! = “” № <> { } [ ] ( ) $ # & ^ va h.k.

Dastur bajarilishi jarayonida o’z qiymatini o’zgartira oladigan kattaliklar o’zgaruvchilar deyiladi. O’zgaruvchilarning nomlari harfdan boshlanuvchi xarf va raqamlardan iborat bo’lishi mumkin. O’zguruvchilarni 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 tizimsida: ular faqat butun sondan iborat bo’ladilar. Masalan: 5; 76; -674 va h.k.
- Sakkizlik sanoq tizimsidagi sonlar: 0 (nol)dan boshlanib, 0dan 7 gacha bo’lgan raqamlardan tashkil topadi. Masalan: x=0453217; s=077;
- O’n oltilik sanoq tizimsidagi 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'ladilar. 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: $7{,}204*10^{12}$ yoki $3{,}567*10^{-11}$ kabi sonlar 7.204e+12 va 3.567e-11 ko'rinishda yoziladi.
- Simvolli konstantalar. Ular qatoriga dastur bajarilishi ' ' ichida qabul qilinadigan simvollar kiradi.

C/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 longdouble 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 chartoifasi 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: inta=5, b, s=-100; - a, b, 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++ tilida standart funksiyalarning yozilishi

Funksiya	Ifodalanishi	Funksiya	Ifodalanishi
Sin x	sin(x)	$\sqrt{x}$	sqrt(x); pow(x,1/2.)
Cos x	cos(x)	$\lvert x \rvert$	abs(x) yoki fabs(x)
Tg x	tan(x)	Arctan x	atan(x)
e^x	exp(x)	Arcsin x	asin(x) ?
Ln x	log(x)	Arccos x	acos(x) ?
Lg x	log10(x)	$\sqrt[3]{x^2}$	pow(x,2/3.)
x^a	pow(x,a)	Log_2x	log(x)/log(2)

Masalan: $\frac{-b+\sqrt{b^2-4ac}}{2a}$ → (-b+sqrt(b*b-4*a*c)/(2*a); yoki
(-b+pow(b*b-4*a*c,1/2.)/(2*a);
$e^{\sin x} + tg^2(x+3)$ → exp(sin(x)) + pow(tan(x+3),2);
k=(m*5)+((7 % n) / (9+x));
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> - S 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 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 aloxida INCLUDE deb nomlanadigan papkada saqlanadiar. Hozirda C++ kutubxonasini yangilandi va undagi fayllarning nomlaridan .h (head – bosh ma’nosida) kengaytmasi olib tashlandi va oldiga c harfi qo’shildi (C dan qolgan 18 tasiga). Bu fayllarda funksiya prototoifalari, toifalari, o’zgaruvchilar, o’zgarmaslar ta’riflari yozilgan bo’ladi.

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 o’z 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++ dagi 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 parametrlar 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 xam kelishi mumkin, ular /* ....*/ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

main ( ) // C++ tilining asosiy funksiyasi

Tilda 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 xaqiqiy 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; || (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 || s4<=20);

Yoki $6 \le x \le 10$ 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 → a = a + b;

- = : a-=b → a = a - b;

* = : a*=b → a = a * b;

/ = : a/=b → a = a / b;

% = : a%=b → 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: ceil (12.6) = 13.0; 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;

Muhokama savollari

1. Tilning alifbosi.
2. O'zgarmaslar
3. O'zgaruvchilarning toifalari.
4. Standart funksiyalarning ko'rinishi.

Nazorat savollari:

1. C/C++ tilida o’zgarmaslar.
2. C/C++ tilida o’zgaruvchilarning toifalari
3. Kompanovka bosqichlarini ayting.
4. Standart funksiyalarning qo’llanishi.
5. Ifodalar haqida tushuncha.
6. Dastur tuzilishi.
7. Preprosessor direktivalari
8. Identifikator, o’zgaruvchilar va o’zgarmaslar.
9. O’zgaruvchilarning oddiy toifalari.
10. Amallar va ifodalar.
11. Standart matematik funksiyalar.
12. Oddiy arifmetik ifodalar.

2-ma’ruza. C++ da dastlabki dasturni yozish

Ma’ruza rejasi:

2.1 Birinchi dasturni yozish
2.2 Obyektlar, qiymat va toifalar
2.3 Hisoblash
2.4 Xatoliklar

Kalit so’zlar: *toifalar*, *xatolik, sintaktik xatolik, dastur ishlashi davomidagi xatolik, testlash, kompilyatsiya paytidagi xatolik, talab, mantiqiy xato, turlar xatosi*.

2.1 Birinchi dasturni yozish

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

Bundan tashqari, biz o’zimiz bajarishimiz kerak bo’lgan vazifa tavsifini olamiz, masalan, "yaqin oradagi kinoteatrga qanday borish mumkin" yoki "to’lqinli pechda go’shtni qanday qovurish mumkin". bunday tavsiflar va dasturlar orasidagi farq aniqlik darajasida aniqlanadi: insonlar sog’lom aql bilan qo’llanmani noaniqligini aniqlashga harakat qiladilar, kompyuter bunday qila olmaydi. Masalan, "yo’lak bo’ylab o’nga, zinadan yuqoriga, so’ngra chapga" - yuqori qavatdagi yuvinish xonasini topish imkonini beruvchi aniq qo’llanma. Biroq, agar siz bunday sodda qo’llanmaga qarasangiz, u holda ular grammatik noaniqligi va to’liq emasligini ko’rish mumkin. Masalan, siz stol atrofida o’tiribsiz va yuvinish xonasiga qanday o’tishni so’radingiz. Sizga javob beruvchi, o’rningizdan turishingizni, uni aylanib o’tishingizni va boshqalarni aytishi shart emas. Yana sizga hyech kim sanchqini stolga qo’yishingiz, zanadan ko’tarilayotganda chiroqni yoqishingiz kerakligini, yuvinish xonasiga kirish uchun eshikni ochish kerakligini maslahat bermaydi.

Qarama-qarshi holatda bunga kompyuterning aqli yetmaydi. Unga barchasini aniq va batafsil tavsiflash kerak. Kompyuterga qo’llanmani batafsil tavsiflash uchun, o’ziga xos grammatikaga ega bo’lgan aniq belgilangan til hamda biz bajarishni xoxlayotgan faoliyatlarni barcha ko’rinishlari uchun yaxshi aniqlikdagi lug’at kerak bo’ladi. Bunday til dasturlash tili va ko’p qamrovli masalalarni yechish uchun ishlab chiqilgan - C++ dasturlash tili deb nomlanadi.

Kompyuterlar, dasturlar va dasturlash bo'yicha falsafiy qarashlar 1-maruzada kengroq yoritib berilgan. Bu yerda biz juda oddiy dasturdan boshlanadigan kodni hamda uning bajarilishi uchun kerak bo'ladigan bir qancha usullar va qurilmalarni ko'rib chiqamiz.

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

```
// Bu dastur ekranga "Hello, World! xabarini chiqaradi"
#include <iostream>
using namespace std;
int main() // C++ da dasturlar main funksiyasidan boshlanadi
{
cout<< "Hello, World!\n"; // "Hello, World!" ni chiqarish
return 0;
}
```

Kompyuter bajarishi lozim bo'lgan bu buyruqlar to'plami, pazandalik resepti yoki yangi o'yinchoqni yig'ish bo'yicha qo'llanmani eslatadi. Eng boshidan boshlab, dasturning har bir satrini ko'rib chiqamiz:

```
cout<<"Hello, World!\n";// "Hello, World!" chiqarish
```

Aynan mana shu satr xabarni ekranga chiqaradi. U yangi satrga o'tuvchi belgi bilan Hello, World! belgilarini chop etadi; aks holda, Hello, World! belgisini chiqargandan so'ng kursor yangi satrning boshiga joylashtiriladi. Kursor - bu keyingi belgi qayerda chiqishini ko'rsatib turuvchi katta bo'lmagan yonib o'chib turuvchi belgi yoki satr.

C++ tilida satrli literallar qo'shtirnoq (") bilan belgilanadi; ya'ni "Hello,Word!\n" — bu belgilar satri. \n belgi - yangi satrga o'tishni bildiruvchi maxsus belgi. cout standart chiqarish oqimiga tegishli. "cout oqimida chiquvchilar" << chiqarish operatori yordamida ekranda aks etadi. cout "see-out" kabi talaffuz qilinadi, lekin "character putstream" ("belgilarni chiqarish oqimi") qisqartmasi hisoblanadi. Dasturlashda qisqartmalar yetarlicha keng tarqalgan. Albatta, qisqartmalar birinchi marta eslab qolish uchun noqulay ko'rinishi mumkin, lekin o'rganib qolgach ulardan voz kecha olmaysiz, ya'ni ular qisqa va boshqarib bo'ladigan dasturlarni yaratishga imkon beradi.

Satr oxiri

```
//"Hello, World!" chiqarish
```

izoh hisoblanadi. // (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.

Izohlar insonlar uchun dastur matnida ifodalashning iloji bo’lmagan foydali ma’lumotlarni qamrab oladi va dastur manzilini tasvirlaydi. Umuman olganda, izoh o’zingiz uchun ham foydali bo’lishi mumkin, ya’ni yaratgan dasturingizga xafta yoki yillar o’tib murojat qilganingizda nima uchun yaratilganini tezda anglab olishga yordam beradi. O’zingizni dasturlaringizni yaxshi xujjatlashtirishga harakat qiling.

Dastur ikkita auditoriya uchun yoziladi. Albatta, biz ularni bajaruvchi kompyuterlar uchun dastur yozamiz. Biroq biz kodni o’qish va modifikasiyalashga uzoq yillar sarflaymiz. Shunday qilib, dastur uchun ikkinchi auditoriya bo’lib boshqa dasturchilar hisoblanadi. Shuning uchun dasturning yaratilishini insonlar o’rtasidagi muloqot shakli deb sanash mumkin. Albatta, insonlarni o’z dasturlarini birinchi o’quvchilari deb hisoblash maqsadga muvofiq: agar ular qiyinchilik bilan o’z yozganlarini tushunishsa, u holda dastur qachonlardir to’g’ri bo’lishi dargumon. Shu sababli, kod o’qish uchun mo’ljallanganligini esdan chiqarmaslik kerak - dastur onson o’qilishi uchun barcha imkoniyatlarni qilish kerak. Istalgan holatda izohlar faqat insonlar uchun kerak, kompyuter ularni inkor qiladi.

Dasturning birinchi satri - bu o’quvchilarga dastur nima qilishi kerakligi haqida xabar beradigan toifaviy izoh.

// Bu dastur ekranga "Hello, World!" xabarini chiqaradi.

Bu izohlarning foydali tomoni shundaki, bunda dastur kodi bo’yicha dastur nima qilayotganini tushunish mumkin, lekin biz aynan nima hohlayotganimizni aniqlash mumkin emas. Bundan tashqari, kodning o’zidan tashqari, biz izohlarda dasturning maqsadini qisqacha tushuntirishimiz mumkin. Ko’pincha bunday izohlar dasturning birinchi satrida joylashgan bo’ladi. Boshqa narsalar orasida, ular biz nima qilmoqchi ekanligimizni eslatib o’tadi.

Satr

#include "std_lib_facilities.h"

o’zi bilan #include direktivasini akslantiradi. U std_lib_facilities.h faylida tasvirlanganlarni imkoniyatlarini "faollashtirish" uchun kompyuterni majburlaydi. Bu fayl C++ (C++ tilining standart kutubxonasi) tilining barcha amalga oshirishlarida ko’zda tutilgan imkoniyatlaridan foydalanishni osonlashtiradi.

std_lib_facilities.h 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. #include direktivasi yordamida dasturga kiruvchi fayl odatda .h kengaytmasiga ega bo'ladi va sarlavha (header) yoki sarlavha fayli (header file) deb nomlanadi. 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:

```
intmain(){}
```

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
```

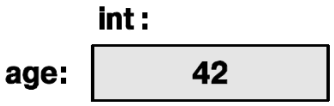
return0;

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.

2.2 Obyektlar, qiymat va toifalar

"Hello world" dasturi faqatgina ekranga yozuv chiqaradi holos. U hechnima o'qimaydi, ya'ni, foydalanuvchi hechnima kiritmaydi. Bu juda zerikarli. Haqiqiy dasturlar odatda biz unga kiritgan ma'lumotlarga qarab qandaydir hisob kitoblar o'tkazadi.

Ma'lumotlarni o'qib olish uchun bu ma'lumotlarni saqlashga joy bo'lishi kerak, boshqacha qilib aytganda, o'qib olingan ma'lumotni yozish uchun bizga kompyuter hotirasidan joy kerak. Bu joyni biz Obyekt deb ataymiz. Obyekt – bu ma'lum bir tur (bu joyda saqlash mumkin bo'lgan ma'lumot turi) ga ega bo'lgan hotiradagi joy. Nom berilgan obyekt esa o'zgaruvchi deyiladi. Masalan, satrlar *string* turiga ega bo'lgan o'zgaruvchilarga saqlanadi, butun sonlar esa – *int* turiga ega bo'lgan o'zgaruvchilarga saqlanadi. Obyektni uning ichiga ma'lum bir turdagi ma'lumotni saqlash mumkin bo'lgan "quti" deb tasavvur qilishimiz mumkin.



Masalan, rasmda *int* turiga mansub, nomga ega bo'lgan va 42 butun soni o'zida saqlagan obyekt tasvirlangan. Satrni turli o'zgaruchilarni kiritish qurilmasidan o'qib olib quyidagicha ekranga chop etishimiz mumkin:

```
// ismni o'qish va yozish
#include "std_lib_facilities.h"
int main()
{
cout << "Iltimos, ismingizni kiriting (keyin 'enter' tugmasini
bosing):\n";
string first_name; // first_name — bu string turli o'zgaruvchi
cin >> first_name; // first_name o'zgaruvchiga belgilarni o'qib olamiz
```

cout << "Hello, " << first_name << "!\n";

}

#include direktivasi bizning barcha dasturlarimizda ishlatilgani uchun, chalkashishlardan qochish maqsadida biz buni o'rganishni ortga suramiz. Shu kabi ba'zida biz *main()* yoki boshqa bir funksiya ichiga yozilganda ishlaydigan kodlarni keltirib o'tamiz.

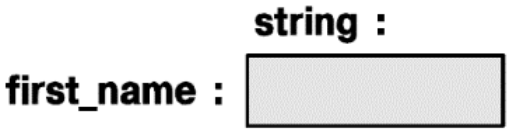
cout << "Iltimos, ismingizni kiriting (keyin 'enter' tugmasini bosing):\n";

Biz sizni bu kodni testlash uchun to'liq dasturga qanday qo'shishni bilasiz deb hisoblaymiz.

main() funksiyasining birinchi qatori foydalanuvchiga ismini kiritishni taklif qiluvchi habar chiqaradi. Keyingi qatorlarda string turli*first_name* o'zgaruvchi e'lon qilindi, ekrandan shu o'zgaruvchiga ma'lumot o'qib olindi va ekranga *Hello* so'zidan so'ng *first_name*o'zgaruvchining qiymati chiqarildi. Shu qatorlarni birin ketin ko'rib chiqamiz.

string first_name; // first_name — bu string turga ega bo'lgan o'zgaruvchi

Bu qatorda kompyuter hotirasidan belgilarni saqlash uchun joy ajratilmoqda va unga *first_name*nomi berilmoqda.

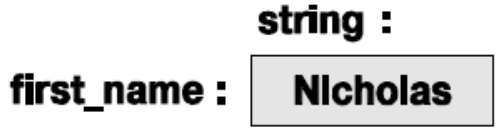


Xotiradan joy ajratish va unga nom berish jarayoni *e'lon qilish* deyiladi.

Keyingi qatorda kiritish qurilmasidan (klaviaturadan) o'zgaruvchiga ma'lumot o'qib olinmoqda:

cin >> first_name; // first_name o'zgaruvchisiga belgilarni o'qib olamiz

cin ("si-in" singari o'qiladi, ingilizcha character input so'zlari qisqartmasidir) nomi standart kutibhonada e'lon qilingan standart oqim kirituviga tegishli. Keyinigi >> (kirituv) operatorining operandi kirituv natijasi saqlanadigan joyni ko'rsatadi. Demak, agar biz Nicholas ismini kiritib yangi qatorga o'tganimizda (ya'ni enter tugmasini bosganimizda) *"Nicholas"* satri *first_name* o'zgaruvchisining qiymatiga aylanadi.



Yangi qatorga o’tish kompyuterning e’tiborini jalb yetish uchun muhim. Yangi qatorga o’tilmagunga qadar (enter tugmasi bosilmagunga qadar) kompyuter shunchaki belgilarni to’plab boradi. Bu kutish bizga ba’zi belgilarni o’chirish yoki boshqa belgilar bilan almashtirish imkonini beradi.

first_name o’zgaruvchisiga qiymatni kiritganimizdan so’ng uni kelgusida ishlatishimiz mumkin bo’ladi.

```
cout << "Hello, " << first_name << "!\n";
```

Bu qator Hello, so’zi va undan so’ng Nicholas ismini (*first_name* o’zgaruvchisining qiymati), ohirida undov belgisi (!) va yangi qatorga o’tish belgisi (‘*n*’)ni chiqaradi

Hello, Nicholas!

Agar biz takrorlanish va ko’p matn yozishni yoqtirganimizda yuqoridagi kodni quyidagicha yozishimiz mumkin:

```
cout << "Hello, ";
cout << first_name;
cout << "!\n";
```

E’tibor bering, Hello, so’zini bir qo’shtirnoqda chiqardik *first_name* o’zgaruvchisini bo’lsa qo’shtirnoqlarsiz yozdik. Qo’shtirnoqlar literal satrlar bilan ishlash uchun zarur, agar satr qo’shtirnoqsiz yozilgan bo’lsa demak u biror bir nomga (o’zgaruvchiga) murojaat qilgan bo’lamiz.

```
cout << "Ism" << " — " << first_name;
```

Bu yerda “Ism” uchta belgidan iborat satrni tashkil qiladi, *first_name* bo’lsa ekranga *first_name* o’zgaruvchisining qiymatini (bizning holatda Nicholas) chiqaradi. Demak natija quyidagicha ko’rinishga ega:

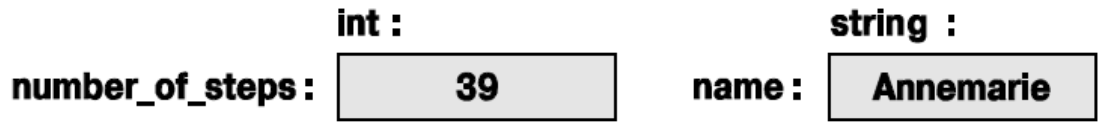
Ism — Nicolas

O’zgaruvchilar. Yuqorida keltirilgan misolda ko’rsatilganidek kompyuter hotirasida ma’lumot saqlash imkinoyitisiz kompyuterda hech qanday qiziqarli ish qilib bo’lmaydi. Ma’lumotlar saqlanadigan joyini obyekt deb ataymiz. Obyektga murojaat qilish uchun obyekt nomini bilish kerak. Nomlangan obyekt o’zgaruvchi deyiladi va unda qanday ma’lumotni saqlash mumkinligini aniqlovchi (masalan *int* turidagi obyektda butun sonlarni saqlash mumkin *string* turidagi obyektga ‘*Hello*

world' singari satrlarni saqlash mumkin) aniq bir turga ega bo'ladi. (masalan, *int* yoki *string*), bundan tashqari ular ustida amallar bajarish mumkin (masalan *int* turidagi obyekt ustida * operatori orqali ko'paytirish amalini bajarish mumkin, *string* turidagi obyektlarni esa<= operatori orqali taqqoslash mumkin). O'zgaruvchilarga yozilgan ma'lumotlar qiymat deyiladi. O'zgaruvchini aniqlash instruktsiyasi e'lon qilish deyiladi va o'zgaruvchi e'lon qilinayotgan paytda unga boshlang'ich qiymat berib ketsa bo'ladi. Quyidagi misolni ko'rib chiqamiz:

string name = "Annemarie";
int number_of_steps = 39;

Bu o'zgaruvchilarni quyidagicha tasavvur qilish mumkin:



Biz o'zgaruvchiga to'g'ri kelmaydigan turdagi ma'lumotni yozaolmaymiz.

string name2 = 39; // xato: 39 — satr emas
int number_of_steps = "Annemarie"; // xato: "Annemarie" —
// butun son emas

Kompilyator har bir o'zgaruvchining turini saqlab qoladi va o'zgaruvchini uning turiga mos ravishda ishlatishingizga yo'l qo'yadi.

C++ tilida juda ko'plab turlar tanlovi mavjud. Lekin ularning atiga 5 tasidan foydalangan holda foydali dasturlar tuzish mumkin.

int number_of_steps = 39; // int — butun sonlar uchun
double flying_time = 3.5; // double — haqiqiy sonlar uchun
char decimal_point = '.'; // char — belgilar uchun
string name = "Annemarie"; // string — satrlar uchun
bool tap_on = true; // bool — mantiqiy o'zgaruvchilar uchun

E'tibor bering, barcha o'zgaruvchilar o'zining yozilish uslubiga ega:

39 // int: butun son
3.5 // double: xaqiqiy son
'.' // char: bittalik tirnoqcha ichida yagona belgi
"Annemarie" // string: qo'shtirnoq ichida yozilgan belgilar to'plami
true // bool: yoki rost (true), yoki yolg'on (false) qiymatga ega

Boshqacha qilib aytganda, raqamlar ketma ketligi (masalan 1234, 2 yoki 973) butun sonni bildiradi, bittalik tirnoqcha ichidagi belgi (masalan, ‘1’, ‘@’ yoki ‘x’) belgini bildiradi, xaqiqiy sonlar esa (masalan, 123.123, 0.12, .98) haqiqiy sonlarni bildiradi, qo’shtirnoq ichidagi belgilar ketma ketligi esa (masalan, “123141”, “Howdy!” yoki “Annemarie”), satrni bildiradi. Batafsil ma’lumot ilova qilingan adabiyotlarda berilgan.

Kiritish va tur. Kiritish operatori ma’lumotlar turiga juda ta’sirchan, ya'ni u kirituv amalga oshayotgan o'zgaruvhi turiga mos ravishda ma'lumotlarni o'qiydi. Quyidagi misolga e’tibor bering:

```
// ism va yoshni kiritish
int main()
{
cout << "Iltimos ismingiz va yoshingizni kiriting\n";
string first_name; // string turdagi o’zgaruvchi
int age;       // integer turidagi o’zgaruvchi
cin >> first_name; // string turdagi ma’lumotni o’qib olamiz
cin >> age;  // integer turidagi ma’lumotni o’qib olamiz
cout << "Hello, " << first_name << " (age " << age << ")\n";
}
```

Demak siz klaviaturada Carlos 22 ni tersangiz kiritish operatori >> first_name o’zgaruvchisiga Carlosni 22 sonini esa age o’zgaruvchisiga o’qib oladi va quyidagi natijani ekranga chop etadi:

Hello, Carlos (age 22)

Nega Carlos 22 satri butunlayicha *first_name* o’zgaruvchisiga yozilmaganining sababi satrlarni o’qish ajratish belgisi (white space) ya’ni probel yoki tabulyatsiya belgisi uchrashi bilan yakunlanadi. Bunday holatda ajratish belgisi kiritish operatori >> tomonidan tashlab ketiladi va sonni o’qishga o’tiladi.

Agar siz klaviaturada 22 Carlos ni terib ko’rsangiz kutilmagan natijaga guvoh bo’lasiz. 22 soni first_name o’zgaruvchisiga yoziladi, chunki 22 ham belgilar ketma ketligi hisoblanadi. Boshqa tomondan esa Carlos butun son emas va u o’qilmasdan tashlab ketiladi. Natijada esa ekranga 22 soni va daviomda “(age” literali va ihtiyoriy son masalan -9842 yoki 0 chop etiladi. Nega? Chunki siz age o’zgaruchisining boshlang’ich qiymatini kiritmadingiz va hech nima kiritmadingiz, natijada unda musor qiymat qolib ketdi. Hozir esa shunchaki age o’zgaruchisiga boshlang’ich qiymat berib qo’yamiz.

```
// ism va yoshni kiritish (2- usul)
int main()
{
cout << "Iltimos ismingiz va yoshingizni kiriting\n";
string first_name = "???";// string turidagi o’zgaruvchi
// ("???" ism kiritilmaganligini bildiradi")
int age = –1;        // int turidagi o'zgaruvchi (–1 “yosh
aniqlanmaganligini bildiradi”)
cin >> first_name >> age; // satr undan so’ng butun sonni o’qiymiz
cout << "Hello, " << first_name << " (age " << age << ")\n";
}
```

Endi 22 Carlos satrini kiritish quyidagi natijaga olib keladi:

Hello, 22 (age –1)

E’tibor bering, biz kiritish operatori orqali bir nechta qiymatlarni kiritishimiz mumkin, bitta chiqarish operatori bilan ularni chop etishimiz mumkin. Bundan tashqari chiqarish operatori << ham kiritish operatori >> singari turlarga sezuvchandir, shuning uchun string turidagi o’zgaruvchi va bir qator satrlar bilan birgalikda butun son (int) turdagi o’zgaruvchini chop etishimiz mumkin.

string turidagi obyektni kiritish operatori >> orqali kiritish ajratish belgisi uchraganda to’xtatiladi boshqacha qilib aytganda kiritish operatori alohida so’zlarni o’qiydi. Ba’zida bizga bir nechta so’zlarni o’qish kerak bo’ladi. Buning ko’plab usuli bor, masalan ikkita so’zdan iborat ismni o’qib olish mumkin:

```
int main()
{
cout << "Iltimos ism, familiyangizni kiriting\n";
string first;
string second;
cin >> first >> second; // ikkita satr o’qib olamiz
cout << "Hello, " << first << ' ' << second << '\n';
}
```

Bu yerda biz kiritish opratorini >> ikki marta ishlatdik. Agar bu so’zlarni ekranga chiqarish kerak bo’lsa ular orasidan probel qo’yish zarur.

Amallar va operatorlar. O’zgaruvchilarning turlari ularda qanday ma’lumot saqlanishidan tashqari ular bilan qanday amallar bajarish mumkinligini ham ko’rsatadi:

```
int count;
cin >> count; // kiritish operatori >> butun sonni count obyektiga yozadi
string name;
cin >> name;        // kiritish operatori kiritilgan satrni name o'zgaruvchisiga yozadi
int c2 = count+2; // + operatori ikki sonni qo'shadi
string s2 = name + " Jr. "; // + operator belgilar bilan to'ldiradi
int c3 = count–2;  // - ikki sonni ayiradi
string s3 = name – "Jr. ";        // xato: - operatori satrlar uchun aniqlanmagan.
```

Xato o'rnida biz kompiltatorning dasturni kompilyatsiya qilmasligini nazarda tutyapmiz. Kompilyator har bir o'zgaruvchiga qanday amallarni bajarish mumkinligini biladi va xato qilsihga yo'l qo'ymaydi. Lekin kompilyator qaysi o'zgaruvchilarga qanday amallarni bajarish mumkinligi haqida bilmaydi va quyidagidek be'mani xatolarga yo'l qo'yib beradi:

```
int age = –100;
```

Ko'rinib turibdiki, inson manfiy yoshga ega bo'la oilmaydi, lekin hech kim bu haqida kompilyatorga aytmadi, shuning uchun bunday hollarda kompilyator hech qanday xatolik haqida habar bermaydi. Quyida eng ko'p tarqalgan turlar uchun amallar ro'yxati keltirilgan:

	bool	**char**	**int**	**double**	**string**
Tenglash (qiymat	**=**	**=**	**=**	**=**	**=**
Qo'shish			**+**	**+**	
Ulash					**+**
Ayirish			**–**	**–**	
Ko'paytirish			*****	*****	
Bo'lish			**/**	**/**	
Qoldiq			**%**		
Birga inkrement			**++**	**++**	
1ga			**—**	**—**	
n ga inkrement			**+=n**	**+=n**	
Ohiriga qo'shish					**+=**
n ga qo'shish			**-=n**	**-=n**	
Ko'paytirib tenglash			***=**	***=**	
Bo'lib tenglash			**/=**	**/=**	
Qoldiq olish va			**%=**		
s fayldan x ga o'qish	**s>>x**	**s>>x**	**s>>x**	**s>>x**	**s>>x**
x ni s faylga yozish	**s<<x**	**s<<x**	**s<<x**	**s<<x**	**s<<x**
Teng	**==**	**==**	**==**	**==**	**==**

Teng emas	!=	!=	!=	!=	!=
Katta	>	>	>	>	>
Katta yoki teng	>=	>=	>=	>=	>=
Kichig	<	<	<	<	<
Kichig yoki teng	<=	<=	<=	<=	<=

Bo'sh kataklar amal bu turlarga to'g'ridan to'g'ri qo'llanib bo'lmasligini ko'rsatadi.Vaqti bilan barcha opreatorlarni tushuntirib o'tamiz.

Xaqiqiy sonlarga doir quidagi na'munani ko'rib chiqamiz:

```
// operatorlar ishini ko'rsatuvchi sodda dastur
int main()
{
cout << "Iltimos haqiqiy son kiriting: ";
double n;
cin >> n;
cout << "n == " << n
<< "\nn+1 == " << n+1
<< "\ntri raza po n == " << 3*n
<< "\ndva raza po n == " << n+n
<< "\nn kvadrati == " << n*n
<< "\nyarim n == " << n/2
<< "\nn ning kvadrat ildizi == " << sqrt(n)
<< endl; // yangi qatorga o'tishning sinonimi ("end of line")
}
```

Ko'rinib turibdiki, oddiy arifmetik amallar odatiy ko'rinishga ega va ularning vazifasi bizga maktab dasturidan ma'lum. Albatta haqiqiy sonlar ustidagi barcha amallar ham operatorlar ko'rinishida tasvirlanmagan, masalan, kavadrat ildiz funksiya ko'rinishida tasvirlangan. Ko'plab amallar funksiyalar sifatida tasvirlangan. Bu holda kvadrat ildizini toppish uchun standart kutubhonalardagi sqrt(n) funksiyasi ishlatilmoqda.

string turi uchun kam operatorlar ko'zda tutilgan lekin bu tur ustida ko'plab amallar funksiyalar sifatida ifodalangan. Bundan tashqari ularga operatorlarni ham qo'llash mumkin:

```
// ism familyani kiritish
int main()
{
cout << "Iltimos, ism va familiyangizni kiriting\n";
```

```
string first;
string second;
cin >> first >> second; // ikki satrni o’qib olamiz
string name = first + ' ' + second; // satrlarni birlashtiramiz
cout << "Hello, " << name << '\n';
}
```

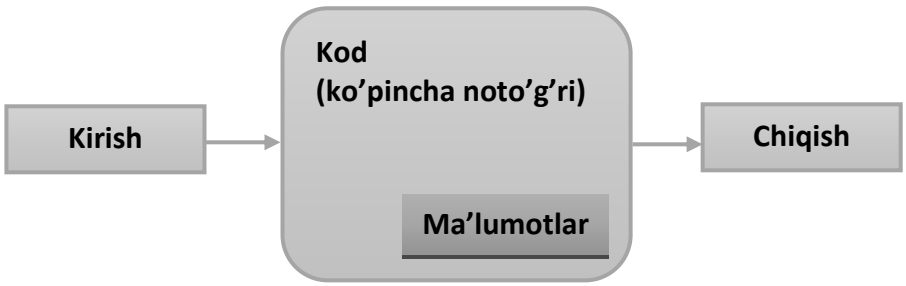
Satrlar uchun + operatori birlashtirishni bildiradi; boshqacha qilib aytganda, agar s1 va s2 o’zgaruvchilari string turiga ega bo’lsa, unda s1 + s2 ham s1 satrdan keyin s2 satr belgilari kelgan satr hisoblanadi. Masalan s1 “Hello” qiymatiga, s2 bo’lsa “World” qiymatiga ega bo’lsa, unda s1 + s2 “HelloWorld” qiymatiga ega bo’ladi. Satrlar ustidagi amallarning asosiylaridan biri bu taqqoslashdir.

```
// ismni kiritish va taqqoslash
int main()
{
cout << "Iltimos ikkita ism kiriting\n";
string first;
string second;
cin >> first >> second; // ikkita satr o’qib olamiz
if (first == second) cout << "ismlar bir hil\n";
if (first < second)
cout << first << " alifbo bo’yicha birinchi keladi " << second <<'\n';
if (first > second)
cout << first << " alifbo bo’yicha keyin keladi " << second <<'\n';
}
```

Bu yerda harakatlarni tanlash uchun if operatori ishlatilgan. Bu operator haqida ilovada keltirilgan adabiyotlardan o’qishingiz mumkin.

2.3. Hisoblash

Barcha dasturlar nimanidir hisoblaydi; boshqacha aytganda ular kirishda qandaydir ma’lumotlarni oladi va qandaydir natijalarni chiqaradi. Bundan tashqari, dasturlar bajariladigan qurilmaning o’zi kompyuter deb nomlanadi (ingliz tilidan tarjima qilganda computer - hisoblagich). Bu nuqtai nazar biz kiritish va chiqarish muomalasiga keng rioya qilishimizda to’g’ri va oqilona hisoblanadi.

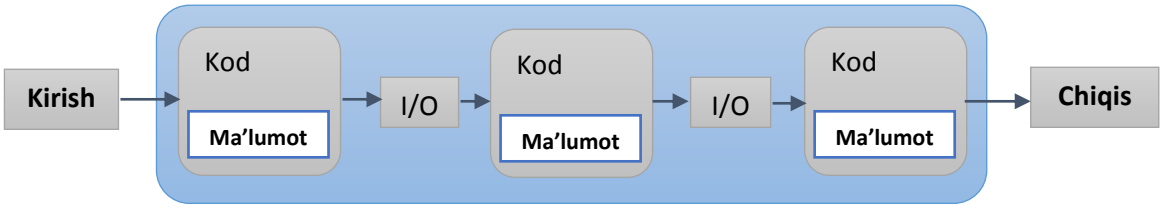


Kiruvchi ma’lumotlar klaviaturadan, sichqonchadan, sensor ekrandan, fayllardan, kiritishning boshqa qurilmalaridan va dasturning boshqa qismlaridan kiritilishi mumkin. "Kiritishning boshqa qurilmalari" kategoriyasi qiziqarli ma’lumotlar manbaini o’z ichiga oladi: musiqiy klavishli pultlar, videoyozuv qurilmalari, harorat hisoblagichlari, raqamli videokameralar sensori va shu kabilar. Bu qurilmalarning turlari chegarasi yo’q.

Dasturga kiruvchi ma’lumotlarni qayta ishlash uchun odatda ma’lumotlar tuzilmasi (data structures) yoki ularning tarkibi (states) deb nomlanuvchi maxsus ma’lumotalrdan foydalaniladi. Masalan, kalendarni aks ettiruvchi dasturda turli mamlakatlardagi bayram kunlari ro’yxati va ish yuzasidan uchrashuvlaringiz keltirilgan bo’lishi mumkin. Ma’umotlarning ayrimlari avval boshidan dastur qismi hisoblanadi, boshqalari esa, qachonki dastur ma’lumotlarni o’qiganda va ularda foydali ma’lumotalrni uchratganda sodir bo’ladi. Masalan, kalendarni aks ettiruvchi dastur, ish bo’yicha uchrashuvlaringizni kiritishni boshlaganingizda ro’yhatni yaratishi mumkin. Bu holatda, asosiy kiruvchi ma’lumot uchrashuv oy va kuniga so’rov yuborishda va ish bo’yicha uchrashuvlar ma’lumotlarini kiritishda hisoblanadi.

Kiruvchi ma’lumot turli xil manbalardan kirishi mumkin. Shunga o’xshash, natijalar ham turli xil qurilmalarda chop etilishi mumkin: boshqa dasturlar yoki dastur qismlari ekranida. Chiquvchi qurilmaga tarmoq interfeyslari, musiqiy sintezatorlar, elektr motorlar, quvvat generatorlari, isitgichlar va boshqalar kiradi.

Dasturchi nuqtai nazaridan kiritish-chiqarishda eng muxim va qiziqarli kategoriyalar “boshqa dasturga” va “dasturning boshqa qismlariga” hisoblanadi. Qanday qilib dasturni o’zaro ta’sir etuvchi qismlar ko’rinishida tasavvur qilish va ma’lumotlarga o’zaro kirish va ma’lumotlar almashishni ta’minlash mumkinligi. Bu dasturlashning kalit savollari. Ularni grafik ko’rinishda tasvirlaymiz.



I/O qisqartmasi kiritish-chiqarishni bildiradi. Bunday holatda dasturning bir qismidan chiqish keyingi qismga kirish hisoblanadi. Dasturning bu qismlari doimiy xotira qurilmasida ma'lumotlarni saqlash yoki tarmoq aloqalari orqali uzatiladigan asosiy xotirada saqlanadigan malumotlarga kirishga ega bo'ladi.

Dastur qismida kiruvchi ma'lumotlar, odatda argument, dastur qismidan chiquvchi ma'lumotlar esa natija deb ataladi.

Hisoblash deb, aniq kiruvchi ma'lumotlar asosida va aniq natijalarni hosil qiluvchi qandaydir ta'sirlarga aytiladi.Eslatib o'tamizki, 1950 yilgacha AQSh da kompyuter deb, hisoblashlarni amalga oshiruvchi insonlar atalgan, masalan, hisobchilar, navigator, fizik. Hozirgi kunda oddiygina ko'plab hisoblashlarni kompyuyeterga topshirdik, bularning orasida kalkulyator eng soddasi hisoblanadi.

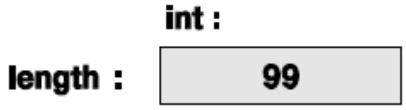
Ifoda. Dasturning asosiy konstruktor qurilmasi ifoda hisoblanadi. Ifoda aniqlangan operandlar soni asosida qandaydir qiymatlarni hisoblaydi. Sodda ifodalar o'zida oddiy literal konstantalarni ifodalaydi, masalan, 'a', 3.14 yoki "Norah".

O'zgaruvchilar nomi ham ifoda hisoblanadi. O'zgaruvchi – bu nomga ega bo'lgan obyekt. Misol ko'ramiz.

```
// maydonni hisoblash:
int length = 20; // literal butun qiymat
// (o'zgaruvchini inisializasiyalash uchun foydalaniladi)
int width = 40;
int area = length*width; // ko'paytirish
```

Bu yerda length so'zi, o'zlashtiruvchi operator chap operandalarini belgilovchi "length nomli obyektni" anglatadi, shuning uchun bu ifoda quyidagicha o'qiladi: "length nomli obyektga 99 raqamini yozish". O'zlashtiruvchi yoki inisializasiyalovchi operatorning chap qismida (u "length o'zgaruvchining lvalue" deb nomlanadi) va bu operatorlarning o'ng qismida (bu holatda u "length o'zgaruvchining rvalue", "length nomli obyektning qiymati" yoki oddiy "length qiymati") joylashgan length

o’zgaruvchini ajratib olish kerak. Bu kontekstda nomlangan quti ko’rinishida o’zgaruvchilarni tasvirlash foydali.



Boshqacha aytganda, length– bu 99 tagacha qiymatni o’z ichiga oluvchi inttoifali obyekt nomi. Ba’zida length nomi (lvalue sifatida) qutiga, ba’zida esa (rvalue sifatida) – mana shu qutida saqlanayotgan qiymatning o’ziga tegishli bo’ladi.

+ va * operatorlari yordamida ifodalarni birlashtirib, quyida ko’rsatilganidek murakkabroq ifodalarni yaratishimiz mumkin. Ifodalarni guruhlash kerak bo’lganda qavslardan foydalanish mumkin.

int perimeter = (length+width)*2; // qo’shish va ko’paytirish

Qavssiz bu ifodani quyidagi ko’rinishda yozish mumkin:

int perimeter = length*2+width*2;

juda qo’pol va xatoliklarni keltirib chiqaradi.

int perimeter = length+width*2; // length bilan width*2 qo’shish

Oxirgi xatolik mantiqiy hisoblanadi va kompilyator uni topa olmaydi. Kompilyatoraniq ifodada inisializasiyalangan perimeter nomili o’zgaruvchini ko’radi. Agar ifoda natijasi ma’noga ega bo’lmasa, bu sizning muammoyingiz. Siz perimetrning matematik aniqliklarini bilasiz, kompilyator esa yo’q.

Dasturda operatorlar bajarilish tartibini aniq belgilaydigan, oddiy matematik qoidalardan foydalaniladi, shuning uchun length+width*2, length+(width*2) ni anglatadi. Shunga o’xshash a*b+c/d, a*(b+c)/d emas (a*b)+(c/d) ni anglatadi.

Qavsdan foydalanishning birinchi qoidasi shundaki: “Agar ikkilanayotgan bo’lsang qavsdan foydalan”. Umuman olganda dasturchi a*b+c/d formula qiymatida ikkilanmaslik uchun ifodalarni to’g’ri shakllantirishni o’rganishi kerak. Operatorlardan keng foydalanish, masalan(a*b)+(c/d) dastur o’qilishini susaytiradi.

Nima uchun biz o’qilishiga to’xtalib o’tdik? Chunki sizning kodingizni faqatgina siz emas, balki boshqa dasturchilar ham o’qishi mumkin, chalkashtirilgan kod o’qishni sekinlashtiradi va uning tahliliga to’sqinlik qiladi. Qo’pol kodni faqatgina o’qish emas, balki uni to’g’irlash ham qiyin bo’ladi. Yomon yozilgan kod odatda mantiqiy xatoliklarni berkitib turadi. Uning o’qilishida qancha ko’p kuch ketgan bo’lsa,

o’zingizni va boshqalarni uning to’g’riligiga ishontirish shuncha qiyin bo’ladi. Juda ham qiyin bo’lgan ifodalarni yozmang va har doim ma’noli nomlarni tanlashga harakat qiling.

a*b+c/d*(e–f/g)/h+7 // juda qiyin

2.4 Xatoliklar

Dastur ishlab chiqishda xatolardan qochib bo’lmaydi, lekin dasturning ohirgi nusxasi iloji boricha xatolarsiz bo’lishi zarur.

Xatolarning ko’plab turi bor:

- *Kompilyatsiya paytidagi xato*. Kompilyator tomonidan aniqlangan xatolar.

Bularni dasturlash tilining qaysi qoidalarnini buzishiga qarab bir nechta turlarga bo’lishimiz mumkin.

- Sintaktik xatolar;
- Tiplar bilan yo’lga qo’yilgan xatolar.
- *Bog’lanishlar paytidagi xato.*Bu bog'lanishlar tahrirlovchisi tomonidan obyekt fayllarni bajarilish moduliga qo'shish paytida topilgan xatolar.
- *Bajarilish paytidagi xatolar*.Bu dastur bajarilish davomida vujudga kelgan xatolar. Bularni quyidagi turlarga bo’lish mumkin:
- Kompyuter tomonida aniqlangan xatolar (uskunaviy taminotva/yoki operatsion tizim tomonidan aniqlangan xatolar);
- Kutubhonalar tomonidan aniqlangan xatolar (masalan standart kutubhonalar tomonidan);
- Foydalanuvchining dasturi tomonidan aniqlangan xatolar.
- *Mantiqiy xatolar.*Dasturchi tomonidan noto'g'ri vaziyatlar yuzaga kelayotganda topilgan xatolar.

Oddiy qilib aytganda dasturchining vazifasi – barcha xatolarni bartaraf etish. Lekin ko’p xollarda bu vazifa amallab bo’lmas bo’ladi. Aslida haqiqiy dasturlar uchun “barcha xatolar” deganda nimani tushunish juda qiyin. Masalan, dastur ishlab turgan paytda biz kompyuterni elektor manbaidan uzib qo’ysak buni xato sifatida qarab unga qarshi chora ko’rish kerakmi? Ko’p hollarda rad javobini beramiz, lekin meditisina monitoringi dasturida yoki qo’ng’iroqlarni yo’naltirish dasturlarida bunday emas. Bunday xollarda foydalanuvchi sizning dasturingizdan ma’lum bir hisoblashlarni davom ettirishini talab qiladi. Asosiy savol shundan iborat: Dasturingiz o’zi xatolikni aniqlashi kerakmi yoki yo’qligida.

Agar bu aniq ko’rsatilmagan bo’lsa, biz sizning dasturingiz quyidagi shartlarni bajarishi kerak:

1. Istalgan kiruvchi ma’lumotlarda dasturdan talab qilingan hisoblash ishlarini bajarishi kerak.
2. Barcha to’g’irlab bo’lmas hollarda kerakli xabarni chiqarishi kerak.
3. Uskunaviy ta’minotning barcha xatoliklari xaqida ma’lumot berishi shart emas.
4. Dasturiy ta’minotning barcha xatosini chiqarishi shart ema.
5. Xatolik topilganda dastur ishini yakunlashi kerak.

3-5 ko’rsatilgan shartlarga javob bermaydigan dasturlarni ko’rib chiqmaymiz. Shu bilan birga 1 va 2 professional talablardan biri hisoblanadi, professionallik esa bizning maqsadimiz. 100% mukammalikka erisha olmasakda u qisman mavjud bo’lishi zarur.

Dastur tuzishda xatolar odatiy hol va ulardan qochib bo’lmaydi. Bizning fikrimizcha, jiddiy dasturlar yaratishda, xatolarni chtlab o’tish, topish va to’g’irlash 90% ko’proq vaqtni oladi. Xavfsizligi muhim bo’lgan dasturlarda esa bu vaqt undan xam ko’p bo’ladi. Kichik dasturlarda xatolardan osongina qochish mumkin, lekin katta dasturlarda bu xatoga yo’l qo’yish ehtimoli juda yuqori.

Biz dastur tuzishda quyidagi uch uslubni taklif qilamiz:

- Dasturiy ta’minotni xatoliklarini iloji boricha kamaytirib ishlab chiqish.
- Ko’plab xatolarni dstur ishlab chiqish va testlash jarayonida to’g’irlab ketish.
- Qolgan xatolarni jiddiy emasligiga ishonch hosil qilish.

Bu uslublarning birortasi ham o’z o’zidan xatolarni to’g’ri bo’lishini ta’minlamaydi. Shuning uchun ham biz barcha uch uslubni ham ishlatamiz.

Ishonchli dasturlarni ishlab chiqishda malaka katta ahamiyatga ega.

Xatolarning bir nechta manbalarini keltirib o’tamiz.

*Yomon tasvirlash.*Agar biz dasturning vazifasini tasavvur qila olmasak, uning “qora burchak” larini tekshirib chiqishimiz juda qiyin bo’ladi.

Chala dasturlar. Dasturni yaratish davomida biz e’tiborga olmay ketgan xatoliklar albatta yuzaga chiqadi. Bizning vazifamiz barcha holatlar to’g’ri ishlab chiqilganligiga ishonch hosil qilish.

Ko’zda tutilmagan argumentlar. Funksiyalar argumentlar qabul qiladi. Agar funksiya ko'zda tutilmagan argument qabul qilsa, muammo hosil bo'ladi, masalan standart kutubhonada mavjud sqrt() funksiyasiga manfiy ishoralik son berilsa. sqrt() funksiyasi faqatgina musbat xaqiqiy son qabul qilgani uchun u to’g’ri natija qaytara olmaydi. Bunday muammolar 5.5.3 bo’limda ko’rib o’tiladi.

Ko’zda tutilmagan kiruvchi ma’lumotlar. Odatda dasturlar malumotlarni o’qib olishadi (masalan, klaviaturadan, fayldan, lokal va global tarmoqlardan). Odatda dasturlar kiruvchi ma’lumotlar uchun ko’plab shart qo’yadi, masalan foydalanuvchi son kiritishi kerakligi. Agar foydalanuvchi kutilayotgan son o’rniga “Bas qil!” degan satrni kiritsachi? Muammoning bunday turi 5.6.3 va 10.6 bo’limlarida ko’rib chiqiladi.

Kutilmagan holat. Ko'plab dasturlar ko'plab ma'lumotlarni (holatlarni) saqlashga mo'ljallangan, masalan tizimnig qismlari. Ular safiga manzillar ro’yxati, telefon katalogi va havo harorati haqidagi ma’lumotlar vector turidagi obyektga yozilgan bo’lsin. Bu ma’lumotlar to’liq bo’lmasa yoki noto’g’ri bo’lsa nima bo’ladi? Bunday holatda dasturning turli qismlari boshqaruvni saqlab qolishi zarur.

Mantiqiy xatolar. Bunday xatolar dasturdan talab etilgan vazifani bajarmaslikka olib keladi. Bunday xatolarni topib bartaraf etishimiz zarur.

Sintaktik xatolar.

area() funksiyasini quyidagicha chaqirsak nima sodir bo’ladi:

```
int s1 = area(7;     // xato: qavs tushirib qoldirilgan )
int s2 = area(7)     // xato: nuqta vergul tushirib qoldirilgan ;
Int s3 = area(7);    // xato: Int — tur emas
int s4 = area('7); // xato: tirnoqcha tushirib qoldirilgan '
```

Xar bir qator sintaktik xatoga ega, boshqacha qilib aytganda ular C++ grammatikasiga to’g’ri kelmaydi. Afsuski barcha hollarda ham xatolarni dasturchi tushinishiga oson qilib ifodalash qiyin. Natijada eng oddiy sintaktik xatolar xam tushunarsiz ifodalanadi, bundan tashqari xatolikka ko’rsatayotgan qator ham bir oz uzoqroqda joylashgan bo’ladi. Shuning uchun kompilyator ko’rsatayotgan qatorda hech qanday xatolik ko’rmayotgan bo’lsangiz biroz yuqoriroq qatorlarni tekshirib chiqing.

Turlar bilan bog’liq xatoliklar.

Sintaktik xatolarni to’g’irlaganingizdan so’ng kompilyator turlarni e’lon qilishdagi xatoliklar xaqida ma’lumot bera boshlaydi. Masalan e’lon qilinmagan o’zgaruvchi, yoki unga berilayotgan qiymatning to’g’ri kelmasligi to’g’risidagi xatoliklar:

int x0 = arena(7); // xato: e’lon qilinmagan funksiya

int x1 = area(7); // xato: argumentlar soni noto’g’ri

int x2 = area("seven",2); // xato: birinchi argument noto’g’ri tipga ega

Xato emas.

Kompilyator bilan ishlash davomida qaysidir payt u sizning xatolaringizni oldindan bilishini xoxlaysiz va ularni xato sifatida qaramasligini hohlaysiz. Lekin malakangiz oshgani sari siz kompilyatorni iloji boricha ko’roq xatolar topishini hohlab qolasiz. Quyidagi misolni ko’rib chiqamiz:

int x4 = area(10,–7); // OK: lekin tomoni -7 ga teng bo’lgan to’g’ri to’rtburchakni qanday tasavvur qilish kerak?

int x5 = area(10.7,9.3); // OK: lekin aslida area(10,9) chaqirilyapti

char x6 = area(100, 9999); // OK: lekin natija kesib tashlanadi

Kompilyator x4 o’zgaruvchisi xaqida xech qanday xabar chiqarmaydi. Uning qarashi bo’yicha area(10, -7) to’g’ri hisoblanadi: area() funksiyasi ikkita butun son talab qiladi, siz esa ikki butun sonni unga yuboryapsiz; hech kim bu sonlar musbat bo’lishi xaqida gapirmagan.

Nazorat savollari

1. To’rtta asosiy xatoliklar turini ayting va ularni qisqacha ifodalab bering.
2. Tinglovchilar dasturidagi qaysi xatolarni tashlab ketishimiz mumkin?
3. Xar qanday yakunlangan proyekt nimani kafotlashi kerak?
4. Xatoliklarni topish va bartaraf etishning uchta asosiy uslubini sanab o’ting.
5. Nega biz xato qidirishni yoqtirmaymiz?
6. Sintaktik xato nima? Beshta misol keltiring.
7. Turlar xatosi nima? Beshta misol keltiring.
8. Mantiqiy xato nima? Uchta mmisol keltiring.

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

3-Ma’ruza. C++ da o’zgaruvchilar

Reja:

3.1 O’zgaruvchilar
3.2 Konstantalar
3.3Turlar bilan ishlash

Kalit so’zlar:*bad(), iostream, kiritish amali, buffer, istream, chiqarish amali, clear(), ofstream, kiritish tugallanganligi belgisi, close(), open(), eof() oqimining holati, ostream, kiritish qurilmasi, fail(), unget(), chiqarish qurilmasi, good(), qurilma drayveri, fayl, ifstream.*

3.1. O’zgaruvchilar

O’zgaruvchilar ob’ekt sifatida. C tilining asosiy tushunchalaridan biri nomlangan xotira qismi – ob’ekt tushunchasidir. Obektning xususiy xoli bu o’zgaruvchidir. O’zgaruvchiga qiymat berilganda unga ajratilgan xotira qismiga shu qiymat kodi yoziladi. O’zgaruvchi qiymatiga nomi orqali murojaat qilish mumkin, xotira qismiga esa faqat adresi orqali murojaat qilinadi. O’zgaruvchi nomi bu erkin kiritiladigan identifikatordir. O’zgaruvchi nomi sifatida xizmatchi so’zlarni ishlatish mumkin emas.

O’zgaruvchilarni ta’riflash. C tilida o’zgaruvchini aniqlash uchun kompyuterga uning turi (masalan, int, charyoki float) hamda ismi haqida ma’lumot beriladi. Bu axborot asosida kompilyatorga o’zgaruvchi uchun qancha joy ajratish lozim va bu o’zgaruvchida qanday turdagi qiymat saqlanishi mumkinligi haqida ma’lumot aniq bo’ladi. O’zgaruvchi nomi identifikator bo’lib, xizmatchi so’zlardan farqli bo’lishi kerak.

Har bir yacheyka bir bayt o’lchovga ega. Agar o’zgaruvchi uchun ko’rsatilgan tur 4 baytni talab qilsa, uning uchun to’rtta yacheyka ajratiladi. Aynan o’zgaruvchini turiga muvofiq ravishda kompilyator bu o’zgaruvchi uchun qancha joy ajratish kerakligini aniqlaydi.

Kompyuterda qiymatlarni ifodalash uchun bitlar va baytlar qo’llaniladi va xotira baytlarda hisoblanadi.

O’zgaruvchilar turlari. O’zgaruvchilarning quyidagi turlari mavjud:

char – bitta simvol;
long char – uzun simvol;
int – butun son;
short yoki **short int** – qisqa butun son;
long yoki **long int** – uzun butun son;

float haqiqiy son;

long float yoki **double** – ikkilangan haqiqiy son;

long double – uzun ikkilangan haqiqiy son.

Butun sonlar ta'riflanganda ko'rilgan turlar oldiga unsigned (ishorasiz) ta'rifi qo'shilishi mumkin. Bu ta'rif qo'shilgan butun sonlar ustida amallar mod 2^n arifmetikasiga asoslangandir. Bu yerda n soni int turi xotirada egallovchi razryadlar sonidir. Agar ishoraciz k soni uzunligi int soni razryadlar sonidan uzun bo'lsa, bu son qiymati k mod 2^n ga teng bo'ladi. Ishorasiz k son uchun ga –k amali 2^n–k formula asosida hisoblanadi. Ishorali, ya'ni signed turidagi sonlarning eng katta razryadi son ishorasini ko'rsatish uchun ishlatilsa unsigned (ishorasiz) turdagi sonlarda bu razryad sonni tasvirlash uchun ishlatiladi.

O'zgaruvchilarni dasturning ixtiyoriy qismida ta'riflash yoki qayta ta'riflash mumkin.

Misol uchun:

int a, b1, ac; yoki

int a;

int b1;

int ac;

O'zgaruvchilar ta'riflanganda ularning qiymatlari aniqlanmagan bo'ladi. Lekin o'zgaruvchilarni ta'riflashda inisializasiya ya'ni boshlang'ich qiymatlarini ko'rsatish mumkin.

Misol uchun:

int i = 0;

char c = 'k';

Typedef ta'riflovchisi yangi turlarni kiritishga imkon beradi.

Misol uchun yangi COD turini kiritish:

typedef unsigned char COD;

COD simbol;

Butun turlar o'lchami. Bir xil turdagi o'zgaruvchilar uchun turli kompyuterlarda xotiradan turli hajmdagi joy ajratilishi mumkin. Lekin bitta kompyuterda bir xil turdagi ikkita o'zgaruvchi bir xil miqdorda joy egallaydi.

Masalan, char turli o'zgaruvchi bir bayt hajmni egallaydi. Ko'pgina kompyuterlarda short int (qisqa butun) turi ikki bayt, long int turi esa 4 bayt joy egallaydi. Butun qiymatlar o'lchovini kompyuter sistemasi va ishlatiladigan kompilyator aniqlaydi. 32 – razryadli kompyuterlarda butun o'zgaruvchilar 4 bayt joy egallaydi.

3.2. Konstantalar

Konstantalar turlari. Konstanta bu o'zgartirish mumkin bo'lmagan qiymatdir. C tilida besh turdagi konstantalar ishlatilishi mumkin: simvollar, butun sonlar, haqiqiy sonlar, sanovchi konstantalar va nul ko'rsatkich.

Belgili o'zgarmaslar. Belgili o'zgarmaslar odatda bir bayt joyni egallaydi va bu 256 xil belgini saqlash uchun yetarlidir. Char turi qiymatlarini 0..255 sonlar to'plamiga yoki ASCII belgilar to'plamiga interpretasiya qilish mumkin.

ASCII belgilari deganda kompyuterlarda qo'llaniladigan standart belgilar to'plami tushuniladi. ASCII - bu American Standard Code for Information Interchange (Amerikaning axborot almashinishi uchun standart kodi) degan ma'noni anglatadi.

Misol uchun 'x','*','\012','\0','\n'- bitta simvolli konstanta; 'dd','\n\t','\x07\x07' ikki simvolli konstantalar.

C kompilyatorida tekstlarni formatlovchi bir nechta maxsus belgilardan foydalaniladi. (Ulardan eng ko'p tarqalgani jadvalda keltirilgan).

Maxsus belgilar axborotlarni ekranga, faylga va boshqa chiqarish qurilmalariga chiqarishda formatlash uchun qo'llaniladi.

Maxsus '\' simvolidan boshlangan simvollar eskeyp simvollar deyiladi. Simvolli konstanta qiymati simvolning kompyuterda qabul qilingan sonli kodiga tengdir.

ESC (eskeyp) simvollar jadvali:

Yozilishi	Ichki kodi	Simvoli(nomi)	Ma'nosi
\a	0x07	bel (audible bell)	Tovush signali
\b	0x08	bs (bascspase)	Bir qadam qaytish
\f	0x0C	ff (form feed)	Sahifani o'tkazish
\n	0x0A	lf (line feed)	Qatorni o'tkazish
\r	0x0D	cr (carriage return)	Karetkani qaytarish
\t	0x09	ht (horizontal tab)	Gorizontal tabulyasiya
\v	0x0B	vt (vertical tab)	Vertikal tabulyasiya
\\	0x5C	\ (bacslash)	Teskari chiziq
\'	0x27	' (single out)	Apostrif (oddiy qavs)
\"	0x22	" (double quote)	Ikkilik qavs
\?	0x3F	? (questionmark)	Savol belgisi
\000	000	ixtiyoriy (octal	Simvol sakkizlik kodi

		number)	
\xhh	0xhh	ixtiyoriy (hex number)	Simvol o'noltilik kodi

Ma'lumotlarning butun son turi. Butun sonlar o'nlik, sakkizlik yoki o'n oltilik sanoq sistemalarida berilishi mumkin.

O'nlik sanoq sistemasida butun sonlar 0-9 raqamlari ketma-ketligidan iborat bo'lib, birinchi raqami 0 bo'lishi kerak emas. Lekin yagona 0 bo'lishi mumkin.

Sakkizlik sanoq sistemasida butun sonlar 0 bilan boshlanuvchi 0-7 raqamlaridan iborat ketma-ketlikdir.

O'n oltilik sanoq sistemasida butun son 0x yoki 0X bilan boshlanuvchi 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma-ketlikdir.

Masalan, 15 va 22 o'nlik sonlari sakkizlikda 017 va 026, o'n oltilikda 0xF va 0x16 shaklda tasvirlanadi.

Ma'lumotlarning uzun butun son turi.

Oxiriga l yoki L harflari qo'yilgan o'nlik,sakkizlik yoki o'n oltilik butun son.

Ma'lumotlarning ishorasiz (unsigned) butun son turi:

Oxiriga u yoki U harflari qo'yilgan o'nlik, sakkizlik yoki o'n oltilik oddiy yoki uzun butun son.

Ma'lumotlarning haqiqiy son turi. Ma'lumotlarning haqiqiy son turi olti qismdan iborat bo'lishi mumkin: butun qism, nuqta, kasr qism, e yoki E belgisi, o'nlik daraja va F yoki f suffikslari.

Masalan : 66., .0, .12, 3.14F, 1.12e-12.

Ma'lumotlarning uzun haqiqiy son turi:

Oxiriga L yoki l suffikslari qo'yilgan haqiqiy son.

Masalan: 2E+6L;

Sanovchi konstanta. Sanovchi konstantalar enum xizmatchi so'zi yordamida kiritilib, int turidagi sonlarga qulay so'zlarni mos qo'yish uchun ishlatiladi.

Misol uchun:

enum{one = 1,two = 2,three = 3};

Agar son qiymatlari ko'rsatilmagan bo'lsa eng chapki so'zga 0 qiymati berilib qolganlariga tartib bo'yicha o'suvchi sonlar mos qo'yiladi:

enum{zero,one,two};

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni qabul kiladi:

zero = 0, one = 1, two = 2;

Konstantalar aralash ko'rinishda kiritilishi ham mumkin:

enum(zero,one,for = 4,five,seeks}.

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni qabul kiladi:

zero = 0, one = 1, for = 4;five = 5,seeks = 6;

Yana bir misol:

Enum BOOLEAN {NO, YES};

Konstantalar qiymatlari:

NO = 0, YES = 1;

Nul ko'rsatkich. NULL- ko'rsatkich yagona arifmetik bo'lmagan konstantadir. Konkret realizasiyalarda null ko'rsatkich 0 yoki 0L yoki nomlangan konstanta NULL orqali tasvirlanishi mumkin. Shuni aytish lozimki, bu konstanta qiymati 0 bo'lishi yoki '0' simvoli kodiga mos kelishi shart emas.

Mantiqiy konstanta. Mantiqiy konstantalar true(rost) va false(yolg'on) qiymatlardan iborat. C tilida butun sonlar va ifodalar mantiqiy konstantalar sifatida qaraladi. Ichki ko'rinishi false – 0, ixtiyoriy boshqa qiymat true deb qaraladi.

Satrli konstanta. Satrli konstantalar C tili konstantalariga kirmaydi, balki leksemalari alohida turi hisoblanadi. Shuning uchun adabiyotlarda satrli konstantalar satrli leksemalar deb ham ataladi.

Satrli konstanta bu ikkilik qavslarga olingan ixtiyoriy simvollar ketma-ketligidir. Misol uchun "Men satrli konstantaman".

Satrlar orasiga eskeyp simvollar ham kirishi mumkin. Bu simvollar oldiga \ belgisi qo'yiladi. Misol uchun :

"\n Bu satr \n uch qatorga \n joylashadi".

Satr simvollari xotirada ketma-ket joylashtiriladi va har bir satrli konstanta oxiriga avtomatik ravishda kompilyator tomonidan '\0' simvoli qo'shiladi. Shunday satrning xotiradagi hajmi simvollar soni+1 baytga tengdir.

Ketma-ket kelgan va bo'shliq, tabulyasiya yoki satr oxiri belgisi bilan ajratilgan satrlar kompilyasiya davrida bitta satrga aylantiriladi. Misol uchun:

"Salom""Toshkent"

satrlari bitta satr deb qaraladi.

"Salom Toshkent"

Bu qoidaga bir necha qatorga yozilgan satrlar ham bo'ysunadi. Misol uchun:

"O'zbekistonga "
"bahor "
"keldi"

qatorlari bitta qatorga mos:

"O'zbekistonga bahor keldi"

Agar satrda '\' belgisi uchrasa va bu belgidan so'ng to '\n' satr oxiri belgisigacha bo'shliq belgisi kelsa bu bo'shliq belgilari '\' va '\n' belgisi bilan birga satrdan o'chiriladi. Satrning o'zi keyingi satrda kelgan satr bilan qo'shiladi.

"O'zbekistonga \
 bahor \
keldi"

qatorlari bitta qatorga mos:

"O'zbekistonga bahor keldi"

Nomlangan konstantalar. C tilida o'zgaruvchilardan tashqari nomlangan konstantalar kiritilishi mumkin. Bu konstantalar qiymatlarini dasturda o'zgartirish mumkin emas. Konstantalar nomlari dasturchi tomonidan kiritilgan va xizmatchi so'zlardan farqli bo'lgan identifikatorlar bo'lishi mumkin. Odatda nom sifatida katta lotin harflari va ostiga chizish belgilari kombinasiyasidan iborat identifikatorlar ishlatiladi. Nomlangan konstantalar quyidagi shaklda kiritiladi:

const tur konstanta_nomi = konstanta_qiymati.

Misol uchun:

const double EULER = 2.718282;
const long M = 99999999;
const R = 765;

Oxirgi misolda konstanta turi ko'rsatilmagan, bu konstanta int turiga tegishli deb hisoblanadi.

3.3. Turlar bilan ishlash

Turlarni keltirish. Turlarni keltirish (type casting) ma'lum turdagi o'zgaruvchi boshqa turdagi qiymat qabul qilganda foydalaniladi. Ba'zi turlar uchun keltirish avtomatik ravishda bajariladi. Avtomatik turlarni keltirish o'zgaruvchi turi hajmi qiymatni saqlashga yetarli bo'lganda bajariladi. Bu jarayon kengaytirish *(widening)* yoki yuksaltirish *(promotion)*

deb ataladi, chunki, kichik razryadli tur katta razryadli turga kengaytiriladi. Bu holda turlarni avtomatik keltirish xavfsiz deb ataladi. Masalan int turi char turidagi qiymatni saqlashga yetarli, shuning uchun turlarni keltirish talab qilinmaydi. Teskari jarayon toraytirish *(narrowing)* deb ataladi, chunki qiymatni o'zgartirish talab etiladi. Bu holda turlarni avtomatik keltirish xavfli deb ataladi. Masalan haqiqiy turni butun turga keltirilganda kasr qism tashlab yuboriladi.

Amallarda turlarni avtomatik keltirish. Binar arifmetik amallar bajarilganda turlarni keltirish quyidagi qoidalar asosida amalga oshiriladi:

short va char turlari int turiga keltiriladi;

Agar operandlardan biri long turiga tegishli bo'lsa ikkinchi operand ham long turiga keltiriladi va natija ham long turiga tegishli bo'ladi;

Agar operandlardan biri float turiga tegishli bo'lsa ikkinchi operand ham float turiga keltiriladi va natija ham float turiga tegishli bo'ladi;

Agar operandlardan biri double turiga tegishli bo'lsa ikkinchi operand ham double turiga keltiriladi va natija ham double turiga tegishli bo'ladi;

Agar operandlardan biri long double turiga tegishli bo'lsa ikkinchi operand ham long double turiga keltiriladi va natija ham long double turiga tegishli bo'ladi;

Ifodalarda turlarni avtomatik keltirish. Agar ifodada short va int turidagi o'zgaruvchilar ishlatilsa, butun ifoda tiri int ga ko'tariladi. Agar ifodada biror o'zgaruvchi turi— long bo'lsa, butun ifoda turi long turga ko'tariladi. Ko'zda tutilganidek hamma butun konstantalar int turiga ega deb qaraladi. Hamma butun konstantalar oxirida L yoki 1 simvoli turgan bo'lsa, long turiga ega.

Agar ifoda float turidagi operandga ega bo'lsa, butun ifoda float turiga ko'tariladi. Agar biror operand double turiga ega bo'lsa, butun ifoda turi double turiga ko'tariladi.

Turlar bilan ishlovchi amallar. Turlarni o'zgartirish amali quyidagi ko'rinishga ega:

(tur_nomi) operand;

Bu amal operandlar qiymatini ko'rsatilgan turga keltirish uchun ishlatiladi. Operand sifatida konstanta, o'zgaruvchi yoki qavslarga olingan ifoda kelishi mumkin. Misol uchun (long)6 amali konstanta qiymatini o'zgartirmagan holda operativ xotirada egallagan baytlar sonini oshiradi. Bu misolda konstanta turi o'zgarmagan bo'lsa, (double)6 yoki (float)6

amali konstanta ichki ko’rinishini ham o’zgartiradi. Katta butun sonlar haqiqiy turga keltirilganda sonning aniqligi yo’qolishi mumkin.

Masalan:

int x = 1.7+1.8;

int y = (int)1.7+(int)1.8;

Bu amallar bajarilishi natijasida x o’zgaruvchi qiymati 3 ga y o’zgaruvchi qiymati ikkiga teng bo’ladi.

sizeof amali operand sifatida ko’rsatilgan ob’ektning baytlarda xotiradagi hajmini hisoblash uchun ishlatiladi. Bu amalning ikki ko’rinishi mavjud:

sizeof ifoda

sizeof (tur)

Shuni ta’kidlab o’tish lozimki sizeof funksiyasi preprosessor qayta ishlash jarayonida bajariladi, shuning uchun dastur bajarilish jarayonida vaqt talab etmaydi.

Misol uchun:

sizeof 3.14 = 8

sizeof 3.14f = 4

sizeof 3.14L = 10

sizeof(char) = 1

sizeof(double) = 8.

Nazorat savollari

1.Zamonaviy kompyuterlarning kiritish va chiqarish vositalari qanchalik turlicha bo’lishi mumkin?

2. istream oqimi nima qiladi?

3. ostream oqimi nima qiladi?

4. Fayl nima?

5. Fayl formati nima?

6. Dasturga ma’lumotlar kiritish va chiqarish uchun ishlatiladigan 4 xil tipdagi qurilmalarni ayting..

7.Faylni o’qishning 4 bosqichini aytib bering.

8.Faylga yozishning 4 bosqichini aytib bering.

9.Oqimlarning 4 xil holatini ayting va aniqlang.

10.Ushbu kiritishga oid masalalarni yechish usullari haqida gapiring.

10.1. Foydalanuvchi mumkin bo’lgan diapazondan oshib ketuvchi qiymat kiritdi.

10.2. Ma’lumotlar tugadi (fayl oxiri).

10.3. Foydalanuvchi noto’g’ri tipdagi qiymat kiritdi.

11. Kiritishning chiqarishga nisbatan qiyinligi nimada?

12. Chiqarishning kiritishga nisbatan qiyinligi nimada?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

3. http://www.stroustrup.com/4th.html

4. http://www.cplusplus.com/

4-Ma’ruza.Grafika. C++ da kiritish chiqarish operatorlari va arifmetik amallar

Ma’ruza rejasi:

4.1. Ma’lumotlarni kiritish va chiqarish

4.2. Amallar

Kalit so’zlar:.

4.3. Ma’lumotlarni kiritish va chiqarish

Formatli chiqarish – printf. Chiqarishprintf funksiyasi ko’rsatilgan parametrlarni standart oqimga chiqarish uchun ishlatiladi. Standart oqim tushunchasi keyingi boblarda yoritiladi. Xozircha standart oqim sifatida monitor tushunilishi yetarlidir.

Funksiya **stdio.h** modulida joylashgan bo’lib, umumiy ko’rinishi quyidagichadir:

printf(control,arg1,arg2,…)

Bunda control boshqaruvchi qator deb atalib ikki turdagi simvollardan iborat bo’ladi: oddiy chiqariluvchi simvollar va navbatdagi parametrni o’zgartirib chiqaruvchi spesifikasiyalar.

Har bir spesifikasiya % simvolidan boshlanib o’zgartirish turini ko’rsatuvchi simvol bilan tugaydi.

O’zgartirish simvollari quyidagilardan iborat.

Butun sonlar uchun:

d– parametr ishorali o’nlik butun songa aylantiriladi.

u - parametr ishorasiz o’nlik butun songa aylantiriladi.

o – parametr ishorasiz va birinchi raqami 0 bo’lmagan sakkizlik songa aylantiriladi.

x – parametr ishorasiz va 0x belgisiz o’n oltilik songa aylantiriladi.

X – parametr xuddi x kabi. Faqat harf bilan ko’rsatiluvchi raqamlar katta harf ya’ni A,B,C,D,E,F sifatida yoziladi.

Haqiqiy sonlar uchun:

e – parametr float yoki double turidagi son deb qaraladi va ishorali m.nnnnnne+-xx ko’rinishidagi o’nlik songa keltiriladi.

E – parametr xuddi e kabi. Faqat mantissa belgisi katta harf ya’ni E sifatida yoziladi.

f - parametr float yoki double turidagi son deb qaraladi va ishorali m.nnnnnn ko’rinishidagi o’nlik songa keltiriladi.

g – parametr berilgan son qiymati va aniqligi uchun eng ixcham %e yoki %f tanlaydi.

G – parametr xuddi g kabi. Faqat mantissa belgisi katta harf ya'ni E sifatida yoziladi.

Simvol va satr uchun:

c – parametr bitta simvol deb qaraladi.

s – parametr satr simvollar no'linchi simvol uchramaguncha yoki ko'rsatilgan sondagi simvollar bosiladi.

Misol:

```
#include <stdio.h>
int main()
{
int num = -27; int number = 27; float f = 123.456;
char r = 'a'; char str[4] = "abc";
printf("%d\n", num);        /*    -27        */
printf("%u\n", number);     /*    27         */
printf("%o\n", number);     /*    33         */
printf("%x\n", number);     /*    lb      */
printf(      "%f\n",      f);       /*    123.456001    */
printf("%e\n", f);              /*    1.23456e+02        */
printf("%E\n", f);              /*    1.23456E+02        */
printf("%c\n", r);              /*    a          */
printf("%s\n", str);        /*    abc        */
return 0;
}
```

Prosent % belgisi va o'zgartirish simvoli orasiga quyidagi simvollarni qo'yish mumkin.

Chiqarilayotgan argument chapga tekislash lozimligini ko'rsatuvchi minus belgisi.

Maydon minimal uzunligini ko'rsatuvchi raqamlar qatori.

Maydon uzunligini keyingi raqamlar qatoridan ajratuvchi nuqta.

Biror qatordan qancha simvol ajratib olish lozimligini hamda float yoki double turidagi sonlarda nuqtadan keyin qancha kasr raqamlari bosib chiqarilishini ko'rsatuvchi raqamlar ketma-ketligi.

Chiqarilayotgan son long turiga tegishli ekanligini ko'rsatuvchi uzun o'nlik markeri l.

% dan keyingi simvol o'zgartirish simvoli bo'lmasa u bosmaga chiqariladi.

% simvolini o'zini bosmaga chiqarish uchun %% belgisini berish lozim.

Quyidagi jadval har xil spesifikasiyalarni "HELLO, WORLD" (12 simvolli) so'zini bosishga ta'sirini ko'rsatadi. Bu yerda har bir maydon uzunligini ko'rsatish uchun maydon oxiriga ikki nuqta qo'yilgan.

:%10S:	:HELLO, WORLD:
:%10-S:	:HELLO, WORLD:
:%20S:	: HELLO, WORLD:
:%-20S:	:HELLO, WORLD :
:%20.10S:	: HELLO, WOR:
:%-20.10S:	:HELLO, WOR :
:%.10S:	:HELLO, WOR:

Turlar maksimal va minimal qiymatlari. Turli turlar maksimal va minimal qiymatlari <LIMITS.H> faylidagi konstantalarda saqlanadi.

Quyidagi dasturda char turidagi bitlar soni va char turi maksimal va minimal qiymati ekranga chiqariladi

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("CHAR_BIT = %d\n",CHAR_BIT);
printf("CHAR_MIN = %d CHAR_MAX =
%d\n",CHAR_MIN,CHAR_MAX);
return 0;
}
```

Quyidagi dasturda short, int, long turlari maksimal va minimal qiymati ekranga chiqariladi

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("SHRT_MIN = %d SHRT_MAX =
%d\n",SHRT_MIN,SHRT_MAX);
printf("INT_MIN = %d INT_MAX = %d\n",INT_MIN,INT_MAX);
printf("LONG_MIN = %ld LONG_MAX =
%ld\n",LONG_MIN,LONG_MAX);
return 0;
}
```

Quyidagi dasturda unsigned char, unsigned short, unsigned int, unsigned long turlari maksimal va minimal qiymati ekranga chiqariladi

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("UCHAR_MAX = %u\n",UCHAR_MAX);
printf("USHRT_MAX = %u\n",USHRT_MAX);
printf("UINT_MAX = %u\n",INT_MAX);
printf("ULONG_MAX = %ul\n",ULONG_MAX);
return 0;
}
```

Formatli kiritish Scanf. Scanf funksiyasi stdio.h modulida joylashgan bo'lib, umumiy ko'rinishi quyidagichadir:

Scanf(control, arg1, arg2,…)

Funksiya standart oqimdan simvollarni o'qib boshqaruvchi qator asosida formatlab mos parametrlarga yozib qo'yadi. Parametr ko'rsatkich bo'lishi lozim.

Boshqaruvchi qator quyidagi o'zgartirish spesifikasiyalaridan iborat:

Bo'shliq, tabulyasiya, keyingi qatorga o'tish simvollari;

Oddiy simvollar (% dan tashqari) kiritish oqimidagi navbatdagi simvollar bilan mos kelishi lozim;

% simvolidan boshlanuvchi spesifikasiya simvollari;

% simvolidan boshlanuvchi qiymat berishni ta'qiqlovchi * simvoli;

% simvolidan boshlanuvchi maydon maksimal uzunligini ko'rsatuvchi son;

quyidagi spesifikasiya simvollarini ishlatish mumkin:

d – ishorali o'nli butun son kutilmoqda.

o – ishorali sakkizlik butun son kutilmoqda.

x –ishorali o'n oltilik butun son kutilmoqda.

h - ishorasiz o'nlik son kutilmoqda.

c – bitta simvol kutilmoqda.

s – satr kutilmoqda.

f - float turidagi son kutilmoqda. Kiritilayotgan sonning butun raqamlari va nuqtadan so'ng kasr raqamlari soni va E yoki e belgisidan so'ng mantissa raqamlari soni ko'rsatilishi mumkin.

```
#include <stdio.h>
int main(void)
```

```
{
 unsigned width, precision;
 int number = 256;
 double weight = 242.5;
 printf("What field width?\n");
 scanf("%d", &width);
 printf("The number is :%*d:\n", width, number);
 printf("Now enter a width and a precision:\n");
 scanf("%d %d", &width, &precision);
 printf("Weight = %*.*f\n", width, precision, weight);
 printf("Done!\n");
 return 0;
}
```

Lokal va global o'zgaruvchilar. C tilida o'zgaruvchi ta'rifi albatta blok boshida joylashishi lozim.

O'zgaruvchi mavjudlik sohasi deb shu o'zgaruvchiga ajratilgan xotira mavjud bo'lgan dastur qismiga aytiladi. O'zgaruvchi ko'rinish sohasi deb o'zgaruvchi qiymatini olish mumkin bo'lgan dastur qismiga aytiladi. Biror blokda ta'riflangan o'zgaruvchi lokal o'zgaruvchi deyiladi. Har qanday blokdan tashqarida ta'riflangan o'zgaruvchi global o'zgaruvchi deyiladi.

Lokal o'zgaruvchi mavjudlik va ko'rinish sohasi ta'rifdan to shu ta'rif joylashgan blok oxirigachadir.

Tashqi blokdagi o'zgaruvchi nomi shu blokda joylashgan yoki shu blokdagi ichki blokda o'zgaruvchi nomi bilan bir xil bo'lmasligi kerak.

Global o'zgaruvchi mavjudlik sohasi ta'rifdan to dastur oxirigachadir.

Agar ichki blokdagi o'zgaruvchi nomi global o'zgaruvchi nomi bilan bir xil bo'lsa lokal o'zgaruvchi ko'rinish sohasida global o'zgaruvchi ko'rinmay qoladi.

Misol:

```
#include<stdio.h>
int i = 5;
int k = 6;
int main()
{
int i = 9;
printf("%d\n",i);
printf("%d\n",k);
```

```
return 0;
}
```
Natija:
9
6

4.2. Amallar

Arifmetik amallar. Amallar odatda unar, ya'ni bitta operandga qo'llaniladigan amallarga va binar, ya'ni ikki operandga qo'llaniladigan amallarga ajratiladi.

Binar amallar additiv ya'ni + qo'shish va – ayirish amallariga, hamda multiplikativ, ya'ni * ko'paytirish, / bo'lish va % modul olish amallariga ajratiladi.

Butun sonni butun songa bo'lganda natija butun songacha yaxlitlanadi. Misol uchun, 20/3 = 6; (-20)/3 = -6; 20/(-3) = -6.

Modul amali butun sonni butun songa bo'lishdan hosil bo'ladigan qoldiqqa tengdir. Agar modul amali musbat operandlarga qo'llanilsa, natija ham musbat bo'ladi, aks holda natija ishorasi kompilyatorga bog'liqdir.

Unar amallarga ishorani o'zgartiruvchi unar minus – va unar plyus + amallari kiradi. Bundan tashqari inkrement ++ va dekrement -- amallari ham unar amallarga kiradi.

Inkrement ++ unar amali qiymatni 1 ga oshirishni ko'rsatadi. Amalni prefiks, ya'ni ++i ko'rinishda ishlatish oldin o'zgaruvchi qiymatini oshirib, so'ngra foydalanish lozimligini, postfiks esa i++ ko'rinishda ishlatish oldin o'zgaruvchi qiymatidan foydalanib, so'ngra oshirish kerakligini ko'rsatadi. Misol uchun, i ning qiymati 2 ga teng bo'lsin, u holda 3+(++i) ifoda qiymati 6 ga, 3+i++ ifoda qiymati 5 ga teng bo'ladi. Ikkala holda ham i ning qiymati 3 ga teng bo'ladi.

Dekrement -- unar amali qiymatni 1 ga kamaytirishni ko'rsatadi. Bu amal ham prefiks va postfiks ko'rinishda ishlatilishi mumkin. Bu ikki amalni faqat o'zgaruvchilarga qo'llash mumkin.

Amallar ustivorligi. Murakkab ifodalarda qaysi amal birinchi navbatda bajarilishi operator prioritetiga bog'liq.

Masalan: x = 5+3*8.

Ko'paytirish qo'shishga nisbatan yuqoriroq prioritetga ega. Shuning uchun bu ifoda qiymati 29 ga teng bo'ladi.

Agarda ikkita matematik ifodaning prioriteti teng bo'lsa, ular chapdan o'ngga qarab ketma-ket bajariladi.

Masalan: x = 5+3+8*9+6*4.

Bu ifodada birinchi ko'paytirish amallari chapdan o'ngga qarab bajariladi 8*9 = 72 va 6*4 = 24. Keyin qo'shish amallari bajariladi. Natijada x = 104 qiymatga ega bo'ladi.

Lekin, barcha amallar ham bu tartibga amal qilmaydi. Masalan, o'zlashtirish amali o'ngdan chapga qarab bajariladi.

Additiv amallarining ustivorligi multiplikativ amallarining ustivorligidan pastrokdir.

Unar amallarning ustivorligi binar amallardan yuqoridir.

Razryadli amallar. Razryadli amallar natijasi butun sonlarni ikkilik ko'rinishlarining har bir razryadiga mos mantiqiy amallarni qo'llashdan hosil bo'ladi. Masalan, 5 kodi 101 ga teng va 6 kodi 110 ga teng.

6&5 qiymati 4 ga, ya'ni 100 ga teng.

6|5 qiymati 7 ga, ya'ni 111 ga teng.

6^5 qiymati 3 ga, ya'ni 011 ga teng.

~6 qiymati 4 ga, ya'ni 010 ga teng.

Bu misollarda amallar ustivorligi oshib borishi tartibida berilgandir.

Bu amallardan tashqari M<<N chapga razryadli siljitish va M>>N o'ngga razryadli siljitish amallari qo'llaniladi. Siljitish M butun sonning razryadli ko'rinishiga qo'llaniladi. N nechta pozisiyaga siljitish kerakligini ko'rsatadi.

Chapga N pozisiyaga surish bu operand qiymatini ikkining N chi darajasiga ko'paytirishga mos keladi. Misol uchun 5<<2 = 20. Bu amalning bitli ko'rinishi: 101<<2 = 10100.

Agar operand musbat bo'lsa, N pozisiyaga o'ngga surish chap operandni ikkining N chi darajasiga bo'lib kasr qismini tashlab yuborishga mosdir. Misol uchun 5>>2 = 1. Bu amalning bitli ko'rinishi 101>>2 = 001 = 1. Agarda operand qiymati manfiy bo'lsa ikki variant mavjuddir: arifmetik siljitishda bo'shatilayotgan razryadlar ishora razryadi qiymati bilan to'ldiriladi, mantiqiy siljitishda bo'shatilayotgan razryadlar nullar bilan to'ldiriladi.

Razryadli surish amallarining ustivorligi o'zaro teng, razryadli inkor amalidan past, qolgan razryadli amallardan yuqoridir. Razryadli inkor amali unar amalga qolgan amallar binar amallarga kiradi.

Nisbat amallari. Nisbat amallari qiymatlari 1 ga teng agar nisbat bajarilsa va aksincha 0 ga tengdir. Nisbat amallari arifmetik turdagi operandlarga yoki ko'rsatkichlarga qo'llaniladi.

Misollar:

1! = 0 qiymati 1 ga teng;

1 == 0 qiymati 0 ga teng;

3> = 3 qiymati 1 ga teng;

3>3 qiymati 0 ga teng;

2< = 2 qiymati 1 ga teng;

2<2 qiymati 0 ga teng;

Katta >, kichik <, katta yoki teng > = , kichik yoki teng < = amallarining ustivorligi bir xildir.

Teng = = va teng emas ! = amallarining ustivorligi o'zaro teng va qolgan amallardan pastdir.

Mantiqiy amallar. C tilida mantiqiy tur yo'q. Shuning uchun mantiqiy amallar butun sonlarga qo'llanadi. Bu amallarning natijalari quyidagicha aniqlanadi:

x||y amali 1 ga teng agar x>0 yoki y>0 bo'lsa, aksincha 0 ga teng

x&&y amali 1 ga teng agar x>0 va y>0 bo'lsa, aksincha 0 ga teng

!x amali 1 ga teng agar x>0 bo'lsa, aksincha 0 ga teng

Bu misollarda amallar ustivorligi oshib borish tartibida berilgandir.

Inkor ! amali unar qolganlari binar amallardir.

Qiymat berish amali. Qiymat berish amali = binar amal bo'lib chap operandi odatda o'zgaruvchi o'ng operandi esa ifodaga teng bo'ladi. Misol uchun

z = 4.7+3.34

Bu qiymati 8.04 ga teng ifodadir. Bu qiymat Z o'zgaruvchiga ham beriladi.

Bu ifoda oxiriga nuqta vergul (;) belgisi qo'yilganda operatorga aylanadi.

z = 4.7+3.34

Bitta ifodada bir necha qiymat berish amallari qo'llanilishi mumkin. Misol uchun:

c = y = f = 4.2+2.8;

Bundan tashqari C tilida murakkab qiymat berish amali mavjud bo'lib, umumiy ko'rinishi quyidagichadir:

O'zgaruvchi_nomi **amal** = ifoda;

Bu yerda **amal** quyidagi amallardan biri *,/,%,+,-, &,^,|, <<,>>.

Misol uchun:

x+ = 4 ifoda x = x+4 ifodaga ekvivalentdir;

x* = a ifoda x = x*a ifodaga ekvivalentdir;

x/ = a+b ifoda x = x/(a+b) ifodaga ekvivalentdir;

x>> = 4 ifoda x = x>>4 ifodaga ekvivalentdir;

Imlo belgilari amal sifatida. C tilida ba'zi bir imlo belgilari ham amal sifatida ishlatilishi mumkin. Bu belgilar oddiy () va kvadrat [] qavslardir. Oddiy qavslar binar amal deb qaralib ifodalarga yoki funksiyaga murojaat qilishda foydalaniladi. Funksiyaga murojaat qilish quyidagi shaklda amlga oshiriladi:

<funksiya nomi> (<argumentlar ro'yxati>). Misol uchun sin(x) yoki max(a,b).

Kvadrat qavslardan massivlarga murojaat qilishda foydalaniladi. Bu murojaat quyidagicha amalga oshiriladi:

<massiv nomi>[<indeks>]. Misol uchun a[5] yoki b[n][m].

Vergul simvolini ajratuvchi belgi sifatida ham amal sifatida ham qarash mumkin. Vergul bilan ajratilgan amallar ketma-ketligi bir amal deb qaralib, chapdan o'ngga hisoblanadi va oxirgi ifoda qiymati natija deb qaraladi. Misol uchun:

d = 4,d+2 amali natijasi 6 ga teng.

Shartli amal. Shartli amal ternar amal deyiladi va uchta operanddan iborat bo'ladi:

<1-ifoda>?<2-ifoda>:<3-ifoda>

Shartli amal bajarilganda avval 1- ifoda hisoblanadi. Agar 1-ifoda qiymati 0 dan farqli bo'lsa 2- ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi, aks holda 3-ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi.

Misol uchun modulni hisoblash: x<0?-x:x yoki ikkita son kichigini hisoblash a<b?a:b.

Shuni aytish lozimki shartli ifodadan har qanday ifoda sifatida foydalanish mumkin. Agar F FLOAT turpga,a N – INT turga tegishli bo'lsa,

(N > 0) ? F : Nifoda N musbat yoki manfiyligidan qat'iy nazar DOUBLE turiga tegishli bo'ladi.

Shartli ifodada birinchi ifodani qavsga olish shart emas.

Amallar ustivorligi jadvali

Rang	Amallar	Yo'nalish
1	() [] -> :: .	Chapdan o'ngga
2	! ~ + - ++ -- & * (tur) sizeof new delete tur()	O'ngdan chapga
3	. * ->*	Chapdan o'ngga
4	* / % (multiplikativ binar amallar)	Chapdan o'ngga
5	+ - (additiv binar amallar)	Chapdan o'ngga

6	<<>	Chapdan o'ngga
7	<< = >= >	Chapdan o'ngga
8	= !=	Chapdan o'ngga
9	&	Chapdan o'ngga
10	^	Chapdan o'ngga
11	|	Chapdan o'ngga
12	&&	Chapdan o'ngga
13	||	Chapdan o'ngga
14	?:(shartli amal)	O'ngdan chapga
15	= *= /= %= += -= &= ^= |= <<=>>=	O'ngdan chapga
16	, (vergul amali)	Chapdan o'ngga

Nazorat savollari

1. Butun sonli va haqiqiy turlarni qanday farqi bor?
2. Ishorasiz unsigned turining xossalarini ko'rsating.
3. Ishorasiz unsigned short int va long int turlarining o'zaro farqi nimada?
4. Birinchi qaysi funksiya bajariladi?
5. Simvolli kiritish funksiyalari.
6. Shartli amal umumiy ko'rinishi.
7. Turlarni keltirish qoidalari.
8. Quyidagi #includedirektivasi qanday vazifani bajaradi.
9. Bosh main() funksiyasining o'ziga xos xususiyati nimadan iborat?
10. Izohlar bir necha qatorda yozilishi mumkinmi?

5-Ma’ruza. C++ da satrli kattaliklar va ular bilan ishlash

Ma’ruza rejasi:

5.1 Belgili axborot va satrlar

5.2 So’zlar massivlari

5.3 Izlash va tartiblash

Kalit so’zlar: *delete, masofa keltirish, delete[], new, indeks, this, indeksirlash, [] bo’sh xotira, void*, konteyner, 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.*

5.1. Belgili axborot va satrlar

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

Misollar:

```
const char c = 'c';
char a,b;
```

S 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 capital[] = "TASHKENT"; Bu holda avtomatik ravishda massiv elementlari soni aniqlanadi va massiv oxiriga satr ko’chirish '\0'simvoli qo’shiladi.

Yuqoridagi inisializasiyani quyidagicha amalga oshirish mumkin:

```
char capital[] = {'T','A','S','H','K','E','N','T','\0'};
```

Bu holda so’z oxirida '\0' simvoli aniq ko’rsatilishi shart.

Qiymat berish operatori yordamida satrga qiymat berish mumkin emas. Satrni massivga yoki kiritish paytida yoki nomlantirish yordamida joylashtirish mumkin.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
```

```
char s1[10] = "string1";
int k = sizeof(s1);
printf("\n%s %d",s1,k);
char s2[] = "string2";
k = sizeof(s2);
printf("\n%s %d",s2,k);
char s3[] = {'s','t','r','i','n','g','3','\0'};
k = sizeof(s3);
printf("\n%s %d",s3,k);
char *s4 = "string4";//satr ko'rsatkichi, uni o'zgartirib bo'lmaydi
k = sizeof(s4);
printf("\n%s %d",s4,k);
system("pause");
return 0;
}
```

Natija:

string1 10
string2 8
string3 8
string4 4

Keyingi misolda kiritilgan so’zdan berilgan harfni olib tashlash dasturi berilgan.

```
#include <stdio.h>
int main()
{
char s[100];
scanf("%s",&s);
int i, j;
for ( i = j = 0; s[i] ! = '\0'; i++)
if ( s[i] ! = 'c' )
s[j++] = s[i];
s[j] = '\0';
printf("%s",s);
return 0;
}
```

Xar safar 's' dan farqli simvol uchraganda, u j pozisiyaga yoziladi va faqat shundan so’ng j ning qiymati 1 ga oshadi. Bu quyidagi yozuvga ekvivalent:

```
if ( s[i] ! = c )
s[j] = s[i];
j++;
```

Funksiyalar va satrlar. Funksiyalarda satrlar ishlatilganda ularning chegarasini ko'rsatish shart emas. Satrlarning uzunligini hisoblash len funksiyasini quyidagicha ta'riflash mumkin:

```
int len(char c[])
{ int m = 0;
for(m = 0; c[m]! = '0'; m++);
return m;
};
```

Shu funksiyadan foydalanilgan dasturni keltiramiz:

```
#include <stdio.h>
int len(char c[])
{
int m = 0;
while(c[m++]);
return m-1;
};
int main()
{
char e[] = "Pro Tempore!";
printf("\n%d", len(e));
return 0;
}
```

Bu funksiyaning standart varianti strlen deb ataladi va bu funksiyadan foydalanish uchun string.h sarlavha faylidan foydalanish lozim.

Satrdan nusxa olish funksiyasi strcpy ni C tilida quyidagicha ta'riflash mumkin:

```
#include <stdio.h>
void strlen(char s1[], char s2[])
{
int i = 0;
while(s2[i]! = '\0') s1[i++] = s2[i];
s1[i] = s2[i];
```

```
}

int main()
{
char s1[] = "aaa";
char s2[] = "ddd";
strcpy(s1,s2);
printf("%s",s1);
return 0;
}
```

Natija:

ddd

Berilgan satrni teskariga aylantiruvchi funksiya:

```
reverse(char s[])
{
int c, i, j;
for(i = 0, j = strlen(s) - 1; i < j; i++, j--)
c = s[i];
s[i] = s[j];
s[j] = c;
}
```

Keyingi misolimizda T qatorni S qator oxiriga ulovchi STRCAT(S,T) funksiyasini ko'rib chiqamiz:

```
strcat(char s[], t[])
 {
int i, j;
i = j = 0;
while (s[i] ! = '\0')
i++;
while((s[i++] = t[j++]) ! = '\0')
}
```

5.2. So'zlar massivlari

So'zlar massivini kiritish. C tilida so'zlar massivlari ikki o'lchovli simvolli massivlar sifatida ta'riflanadi. Misol uchun:

```
char name[4][5].
```

Bu ta’rif yordamida har biri 5 ta harfdan iborat bo’lgan 4 ta so’zli massiv kiritiladi. So’zlar massivlari quyidagicha inisializasiya qilinishi mumkin:

char Name[3][8] = { "Anvar","Mirkomil","Yusuf"}.

Bu ta’rifda har bir so’z uchun xotiradan 8 bayt joy ajratiladi va har bir so’z oxiriga ' \0' belgisi kuyiladi.

So’zlar massivlari inisializasiya qilinganda so’zlar soni ko’rsatilmasligi mumkin. Bu holda so’zlar soni avtomatik aniqlanadi:

char comp[][9] = { "kompyuter","printer","kartridj"}.

Quyidagi dasturda berilgan harf bilan boshlanuvchi so’zlar ro’yxati bosib chiqariladi:

```
#include <stdio.h>
int main()
{
char a[10][10];
char c = 'a';
int i;
for (i = 0;i<3;i++) scanf("%s",&a[i]);
for (i = 0;i<3;i++)
if (a[i][0] == c) printf("\n%s",a[i]);
return 0;
}
```

Quyidagi dasturda fan nomi, talabalar ro’yxati va ularning baholari kiritiladi. Dastur bajarilganda ikki olgan talabalar ro’yxati bosib chiqariladi:

```
#include <stdio.h>
int main()
{
char a[10][10];
char s[10];
int k[10];
scanf("%s",&s);
for (int i = 0;i<3;i++)
{
scanf("%s",&a[i]);
scanf("%d",&k[i]);
};
for (int i = 0;i<3;i++)
```

```
if (k[i] == 2) printf("%s\n",a[i]);
return 0;
}
```

Funksiyalar va so'zlar massivlari. Satrli massivlar funksiya argumenti sifatida ishlatilganda satrlarning umumiy uzunligi aniq ko'rsatilishi shart.

Misol tariqasida ixtiyoriy sondagi satrlar massivini alfavit bo'yicha tartiblash funksiyasidan foydalanilgan dasturni ko'rib chiqamiz:

```
#include <stdio.h>
#define m 10
void sort(int n, char a[][m])
{
char c;
int i,j,l;
for (i = 0;i<n;i++)
for (j = i+1;j<m;j++)
if (a[i][0]<a[j][0] )
for(l = 0;l<m;l++)
{
c = a[i][l];
a[i][l] = a[j][l];
a[j][l] = c;
};
};
int main()
{
char aa[][m] = {"Alimov","Dadashev","Boboev"};
sort(3,aa);
for(int i = 0; i<3;i++) printf("%s\n",aa[i]);
return 0;
}
```

5.3. Izlash va tartiblash

Ikkiga bo'lib izlash. Quyidagi dasturda tartiblangan massivda ikkiga bo'lib kalit sonni izlash algoritmi asosida tuzilgan funksiyadan foydalanish keltirilgan:

```
#include<stdio.h>
#include<conio.h>
```

```
int bsearch(int a[],int key,int n)
{
int m1,m2,m;
m1 = 0;m2 = n-1;
while(m1< = m2)
{
m = (m2+m1)/2;
if (a[m] == key) return m;
if (a[m]>key) m2 = --m;
if (a[m]<key) m1 = ++m;
}
return -1;
};
int main(int argc, char* argv[])
{
int m;
int a[] = {5,6,9,11};
m = bsearch(a,7,4);
printf("%d",m);
getch();
return 0;
}
```

Keyingi misolda shu funksiya satrlar uchun varianti keltirilgan:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#define size 5
int strbsearch(char a[][size],char key[],int n)
{
int m1,m2,m,pr;
m1 = 0;m2 = n-1;
while(m1< = m2)
{
m = (m2+m1)/2;
pr = strcmp(a[m],key);
if (pr == 0) return m;
if (pr == -1) m2 = --m;
```

```
if (pr == 1) m1 = ++m;
}
return -1;
};
int main(int argc, char* argv[])
{
int m;
char a[][size] = {"aaa","aab","aac","aad"};
m = strbsearch(a,"aab",4);
printf("%d",m);
getch();
return 0;
}
```

Tezkor tartiblash. Quyidagi dasturda tezkor tartiblash algoritmiga asoslangan funksiyadan foydalanilgan. Algoritm mohiyati shundan iboratki, avval yetakchi element tanlanadi. Funksiyada yetakchi element sifatida boshlang'ich element tanlanadi. Shundan so'ng massiv ikki qismga ajratiladi. Yetakchi elementdan kichik elementlar past qismga, katta elementlar yuqori qismga to'planadi. Shundan so'ng rekursiya asosida algoritm ikkala qismga alohida qo'llanadi.

```
#include<stdio.h>
#include<conio.h>
int a[] = {5,4};
void sqsort(int k1, int k2)
{
if(k1 < k2){
int i, j, k;
i = k1; j = k2;
while(i < j)
{
if (a[k1] > a[i]) {i++; continue;}
if (a[k1] < a[j]) {j--; continue;}
k = a[j]; a[j] = a[i]; a[i] = k;
}
k = a[k1]; a[k1] = a[i]; a[i] = k;
sqsort(k1, i);
sqsort(i+1, k2);
```

```
};
}
int main(int argc, char* argv[])
{
int i;
sqsort(0, 1);
for(i = 0; i < 2; i++) printf("%d ", a[i]);
getch();
return 0;
}
```

Nazoratsavollari

1. Satr simvolli massivdan qanday farq qiladi?
2. Bir o'lchovli massivlarni inisializasiya qilish usullariniko'rsating.
3. Ko'p o'lchovli massiv ta'rifi xususiyatlarini keltiring.
4. Ko'p o'lchovli masivlar inisializasiyasi xususiyatlari.
5. Satrlarni inisializasiya qilish usullariniko'rsating.
6. So'zlar massivi qanday kiritiladi?
7. Qanday qilib bir o'lchovli massivlar formal parametrlar sifatida ishlatilishi mumkin?
8. Qanday qilib ko'p o'lchovli massivlar formal parametrlar sifatida ishlatilishi mumkin?
9. Satr ta'riflash usullari.
10. Satrlar funksiya parametri sifatida.

6-Ma’ruza. Tarmoqlanish operatorlari: sonlar va belgilarni solishtrish

Ma’ruza rejasi:

6.1 O’tish operatori.

6.2 Shartli operatorning qisqa ko’rinishi.

6.3 Shartli operatorning uzun ko’rinishi.

6.4 Tanlash operatori.

Kalit so’zlar: *mantiqiy qo’shish va ko’paytirish, inkor amali, tarmoqlanish, shartli operator, o’tish operatori, tanlash operatori*

Ko’pmasalalarningyechimima’lumbirshartyokishartlarningqo’yilishi gaqarabbajariladi.
Bundayjarayonlarnitarmoqlanuvchihisoblashjarayonideyiladi.
Tarmoqlanuvchihisoblashjarayonlaritarkibidayanatarmoqlanishbo’lishimumkin. Bundaylarnimurakkabtarmoqlanuvchijarayonlardebataladi.
Algoritmiktildakattaliklarningistalganxossasishuondagiqiymatlariuchunbajarilishiyokibajarilmasligi**shart**deyiladi. Masalan: a = v tenglik uchun a=3 va v=3,1 bo’lganda shart bajarilmaydi. N tub son deyilsa va N=19 bo’lsa, shart bajariladi, N=15 bo’lganda shart bajarilmaydi.

Tarmoqlanuvchi jarayonlarni tashkil etishda shartsiz o’tish va shartli o’tish operatorlaridan foydalaniladi. Shunday jarayonlar mavjudki, shartning bajarilishiga qarab, dasturning u yoki bu qismiga o’tishga to’g’ri keladi. Bunday hollarda shartsiz o’tish operatori ishlatiladi.

1. Shartsizo’tishoperatori: **goto n;**buyerda n – metka, belgibo’lib, jarayono’tishikerakbo’lganjoyniko’rsatadi. Metkaxarf, sonvaulararalashmasidaniboratbo’lishimumkin.
1taoperatorgabirnechtametkalarniqo’yishmumkin. (Ustadasturchilar goto n operatoridankamroqfoydalanadilar.)

2. Shartlio’tishoperatori: **if (shart) operator;**Uningishlashiquyidagicha:
agarshartrostbo’lsakeltirilganoperatorbajariladi, agarshartyolg’onbo’lsa, keyingiqatorgao’tiladi. Ko’pinchabuko’rinishishlatilganda 2taoperatorlararalashibketmasligiuchun goto operatoriishlatildi. Agar if so’zidankeyinbirnechtaoperatorlarkeladiganbo’lsa, ularnialohida {} qavslargaolinadi. (buusulkamroqishlatiladi) Masalan:

$$Y=\begin{cases}\sin x, & \text{agar } x<5\\ \end{cases}$$

$\sqrt[3]{x^2}$, agar $x \geq 5$

```
# include <iostream.h>
# include <math.h>
void main ( )
{  floatx, y;                    // xvauningtoifasixaqiqiy
cin>>x;                    // xningsonqiymatikiritiladi
   if  (x<5)                              // agar  x<5 bo'lsa
  { y=sin(x); goto cc; }                  // 1-funksiyaishlaydi
    y=pow(x, 2/3.);                 // aksxolda2-funksiya ishlaydi
cc: cout<< "y="<<y<<endl;    // uningjavobiberiladi. ss-metka
    }                                    // main funksiyasiberkitildi.
```

3. if (shart) 1-operator(lar); else 2-operator(lar);

Masalan: yukoridagi misolni kurib utamiz:

```
# include <iostream.h>
# include <math.h>
void main ( )
{  float x, y;
   cin >> x;
   if  (x<5)   y=sin(x); else   y=pow(x, 2/3.);
   cout << "y="<< y<< endl;   }
```

Izox: if – else konstruksiyasiichidayana if – else konstruksiyasiishlatilishimumkin. Bunda bir nechta if operatoridan iborat ichma-ich joylashgan konstruksiya xosil buladi. Bunday xollarda else sO'zi O'ziga yakin turgan if ga tegishli buladi.

shart ? 1-operator (lar) : 2-operator (lar);

Masalan: x<5 ? y=sin(x) : y=pow(x,2/3.);

(agar shart rost bulsa, 1-operator, aks xolda 2-operator bajariladi)

Tanlash operatori – O'zgaruvchining kiymatiga karab kup tarmok ichidan bittasi tanlanadi. Buoperatorningkurinishikuyidagicha:

```
switch  (ifodayokio'zgaruvchi)
{ case1-qiymat:  operator(lar); break;
case 2-qiymat:  operator(lar); break;
casen - qiymat: operator(lar); break;
default : aksholdagioperator (lar);   }
```

Masalan:

$$\begin{cases} \sin x, \text{ agar } x=1 \\ \\ \\ \end{cases}$$

Y= cos x, agar x=2

tgx, agar x=3

$\sqrt{x}$, boshka barcha xollarda (x>0)

```
# include<iostream.h>
# include <math.h>
void main ( )
{ int x;  float y;
cin >> x;
switch  (x)
{ case  1 :  y=sin(x);  break;
   case  2 :  y=cos(x); break;
   case  3 : tan(x);       break;
   default :  y=sqrt(x);  }
cout << "y="<<y<<"x="<<x<<endl;
getch ( );  }
```

Izox: 1) switchoperatoridagiifodayokio'zgaruvchibutuntoifalibo'lishishart!

2) switchoperatorisatrlaridagibreaksO'zitushibkolsa, joriy case operatoridankeyingi case bloklariichidagiifodalarxambajarilaveradi.

Tekshirilayotganshartlarbirnechtabo'lishixammumkin. Bunday xollarda ularni murakkab shart deyiladi. Bunday shartlarni kuyidagi mantikiy amallar orkali ifoda etiladi:

&& - mantikiy kupaytirish (va)

|| – mantikiy kushish (yoki)

! - mantikiy inkor (emas)

Masalan: $6 \le x \le 10$ bulsa, (x>=6) && (x<=10)

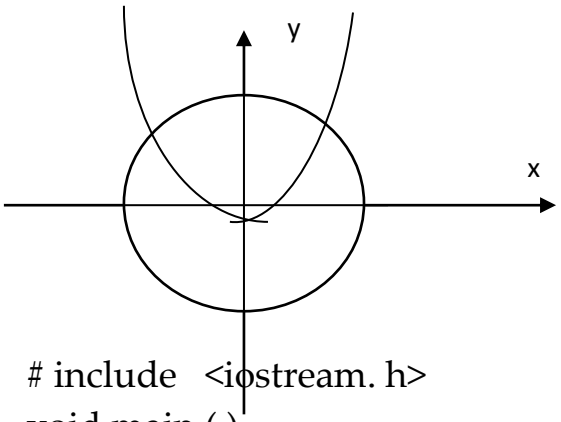
$Y > 0$ va $x < 4$ yoki $z \ge 5$ bylsa, (y>0) && (x<4) || (z>5)

if (sin(x) > 1) && (5 / 2 = 0) y:=1; elsey:=0; {ikkala shart xam yolgon kiymatga ega, shuning uchun y=0 buladi.}

Mantikiy kiymatlar ustida amallar bajarilganida kuyidagi natijalar olinadi: (+ true 1; - false 0 degan ma'noda)

		A	**B**	**A&&B**	**A || B**
				+	+
				-	+
				-	+
				-	-

Mantiqiy masala: ixtiyoriy berilgan M(x,u) nukta $u=x^2$ va $x^2+u^2=4$ aylana bilan kesishgan sohaga yoki shu aylananing 4-choragi tashqarisiga tushishini tekshiring.

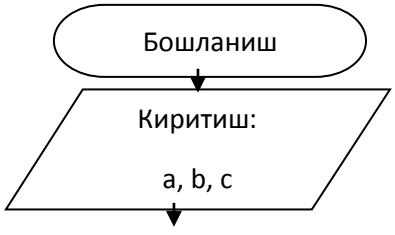


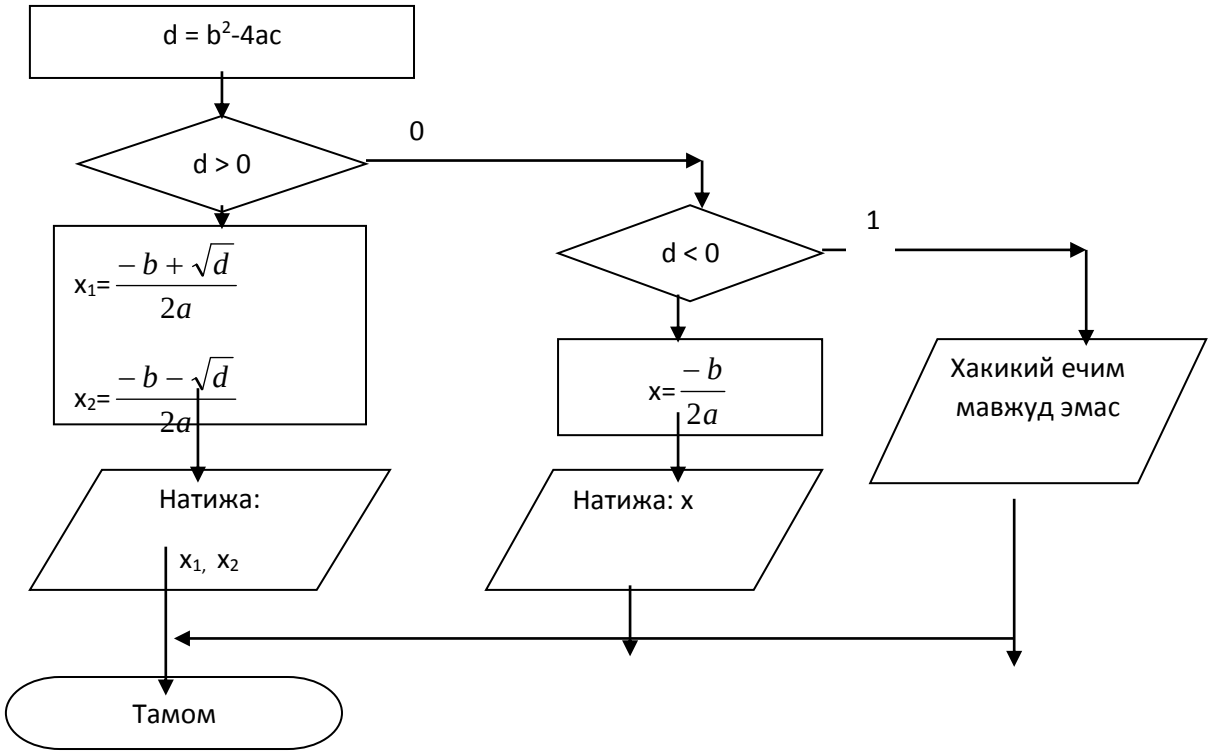
Демак:

$y > x^2$ and $x^2 + y^2 <= 4$
or x > 0 and y < 0 and
$x^2 + y^2 > = 4$
x=1, y=1 → false
x=1, y=0 → true
x=-2, y=0.5 → false
x=2, y=-2 → true

```
# include <iostream. h>
void main ( )
{ float x, y; int n;
cout << "nuqtaning koordinatalarini k
 cin >> x>>y;
if ((y>=x*x && x*x+y*y<=4) ||
(x>0 && y<0 && x*x+y*y>=4)) n=1;
else n=0;
cout << "x="<<x<<endl;
cout << "y="<<y<< endl; cout << "n="<<n<< endl; }
```

3-misol: $ax^2+bx+c=0$ kurinishdagikvadrattenglamaningxakikiyyechimlarinitopishalgoritminitO'zing. (a≠0; b≠0; c≠0;)






```
        # include <iostream.h>
        # include <math.h>
        void main ( )
        float a, b, c, d, x, x1, x2;
        cout << "Tenglamaning
koeffisientlarini kiriting: ";
        cin >> a>>b>>c;
        d = b*b - 4*a*c;
        if  ( d == 0 )  { x=- b / (2*a);
        cout <<"x="<<x<<endl; goto  b15;
}
        if  ( d > 0 )
        { x1 = (- b + sqrt(d)) / (2*a);
          x2 = (- b - sqrt(d)) / (2*a);
        cout
<<"x1="<<x1<<"x2="<<x2<<endl; }
        else  cout <<"yechimiyuk"<<
endl;
        b15 : }
```

```
        # include <iostream.h>
        # include <math.h>
        void main ( )
        float a, b, c, d, x, x1, x2;  int v;
        cout << "Tenglamaning
koeffisientlari:";
        cin >> a>>b>>c;
        d = b*b - 4*a*c;
        if  (d<0)  v=0;
        if  (d = = 0)  v=1; else v=2;
        switch ( v )
        { case 0: cout
<<"yechimiyuk"<< endl; break;
        case  1 : { x=- b / (2*a);
        cout <<"  x="<<x<<endl; }
break;
        case  2:  { x1 = (- b + sqrt(d)) /
(2*a);
          x2 = (- b - sqrt(d)) / (2*a);
        cout
<<"x1="<<x1<<"x2="<<x2<<endl; }
break; }
```

Nazorat savollari:

1. Shartli o’tish operatorining vazifasi
2. Shartli o’tish operatorlarining ko’rinishlari
3. Murakkab operatorlar qachon va qanday qo’llaniladi?
4. Shartli o’tish operatori ichida yana shartli operator qatnashishi mumkinmi?
5. Shartni ifodalovchi blok sxema tuzing.
6. Tanlash operatorining vazifasi.
7. Tanlash operatoridagi **break** ning vazifasi.
8. Tanlash operatoridagi o’zgaruvchilarning tiplari qanday bo’lishi kerak?

7-Ma’ruza. Ko’p tarmoqlanishlar va variant tanlash operatorlari

Ma’ruza rejasi:

7.1 Tanlash operatorlari

Kalit so’zlar: *delete, masofa keltirish, delete[], new, indeks, this, indeksirlash, [] bo’sh xotira, void*, konteyner, 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.*

Tanlash operatorlari

Shartli operator. Shartli operator ikki ko’rinishda ishlatilishi mumkin:

if (ifoda)
1- operator
else
2- operator
yoki
if (ifoda**)**
1-operator

Shartli operator bajarilganda avval ifoda hisoblanadi; agar qiymat rost ya’ni noldan farqli bo’lsa 1- operator bajariladi. Agar qiymat yolg’on ya’ni nol bo’lsa va else ishlatilsa 2-operator bajariladi. Operator else qismi har doim eng yaqin if ga mos qo’yiladi.

```
if( n>0)
if(a>b)
Z = a;
else
Z = b;
```

Agar else qismniyuqori if gamosqo’yishlozim bo’lsa, figurali qavslarishlatishlozim.

```
if( n>0) {
if(a>b)
z = a;
}
else
```

```
z = b;
```

Misol tariqasida uchta berilgan sonning eng kattasini aniqlash dasturi:

```
#include<stdio.h>
int main()
{
float a,b,c,max;
scanf("%f",&a);
scanf("%f",&b);
scanf("%f",&c);
if (a>b)
if (a>c) max = a; else max = c;
else
if (b>c) max = b; else max = c;
printf("\n max = %f", max);
return 0;
}
```

Keyingi misolda kiritilgan ball va maksimal ball asosida baho aniqlanadi:

```
#include<stdio.h>
int main()
{
int ball,max_ball,baho;
printf( "\n ball = ");
scanf("%d",&ball);
printf("\n max_ball = ");
scanf("%d",&max_ball);
float d = (float)ball/max_ball;
if (d>0.85) baho = 5; else
{
if (d>0.71) baho = 4; else
{
if (d>0.55) baho = 3; else baho = 2;
}
}
printf("\n baho = %d",baho);
```

```
return 0;
}
```

Kalit bo’yicha tanlash operatori. Kalit bo’yicha tanlash switch operatori umumiy ko’rinishi quyidagicha:

```
switch(<ifoda>){
case <1-qiymat>:<1-operator>
…
break;
        …
default: <operator>
        …
case: <n-operator>;
   }
```

Oldin qavs ichidagi butun ifoda hisoblanadi va uning qiymati hamma variantlar bilan solishtiriladi. Biror variantga qiymat mos kelsa shu variantda ko’rsatilgan operator bajariladi. Agar biror variant mos kelmasa default orqali ko’rsatilgan operator bajariladi. Uzish break operatori ishlatilmasa shartga mos kelgan variantdan tashqari keyingi variantdagi operatorlar ham avtomatik bajariladi. Quyidagi default, break va belgilangan variantlar ixtiyoriy tartibda kelishi mumkin. Umuman default yoki break operatorlarini ishlatish shart emas. Belgilangan operatorlar bo’sh bo’lishi ham mumkin.

Misol tariqasida bahoni son miqdoriga qarab aniqlash dasturini ko’ramiz.

```
#include <stdio.h>
int main()
{
int baho;
scanf("%d", &baho);
switch(baho)
{
case 2:printf("\n yomon");break;
case 3:printf("\n o'rta");break;
case 4:printf("\n yahshi");break;
case 5:printf("\n alo");break;
default: printf("\n noto'g'ri kiritilgan");
};
```

```
return 0;
}
```
Keyingi misolda kiritilgan simvol unli harf ekanligi aniqlanadi:
```
#include <stdio.h>
int main()
{
char c;
scanf("%c", &c);
switch(c)
{
case 'a':
case 'u':
case 'o':
case 'i':
printf("\n Simvol unli");break;
default: printf("\n Simvol unli emas");
};
return 0;
}
```

8,9-Ma’ruza. Takrorlanuvchi jarayonlar va ularni dasturlash. For, while va do operatorlari yordamida takrorlanishlarni dasturlash

Ma’ruza rejasi:

8.1 Sikl operatorlari

8.2 O’tish operatorlari

Kalit so’zlar: *delete, masofa keltirish, delete[], new, indeks, this, indeksirlash, [] bo’sh xotira, void*, konteyner, 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.*

8.1 Sikloperatorlari

Oldingi shartli whileoperatori. Oldingishartli whileoperatoriquyidagiumumiyko’rinishgaegadir:

```
while(ifoda)
Operator
```

Bu operator bajarilganda avval ifoda hisoblanadi. Agar uning qiymati 0 dan farqli bo’lsa operator bajariladi va ifoda qayta hisoblanadi. To ifoda qiymati 0 bo’lmaguncha sikl qaytariladi.

Agar dasturda while (1); satr qo’yilsa bu dastur hech qachon tugamaydi.

Misol. Berilgan n gacha sonlar yig’indisi.

```
#include <stdio.h>
void main()
{
long n,i = 1,s = 0;
scanf("%d",&n);
while (i< = n )
 s+ = i++;
printf("\n s = %d",s);
}
```

Bu dasturda s+ = i++ ifoda s = s+i; i = i+1 ifodalarga ekvivalentdir.

Quyidagi dastur to nuqta bosilmaguncha kiritilgan simvollar va qatorlar soni hisoblanadi:

```
#include <stdio.h>
int main()
{
int nc = 0,nl = 0;
char c;
while ((c = getchar())! = '.' )
{
++nc;
if (c = = '\n') ++nl;
};
printf("satrlar = %d simvollar = %d  \n",nl,nc);
return 0;
}
```

Keyingi shartli do-while operatori. Keyingi shartli do-while operatoriumumiyko'rinishiquyidagicha:

```
do
Operator
while(ifoda)
```

Sikl operatorining bu ko'rinishida avval operator bajariladi so'ngra ifoda hisoblanadi. Agar uning qiymati 0 dan farqli bo'lsa operator yana bajariladi va hokazo. To ifoda qiymati 0 bo'lmaguncha sikl qaytariladi.

Misol. Berilgan n gacha sonlar yig'indisi.

```
#include <stdio.h>
int main()
{
long n,i = 1,s = 0;
scanf("%d",&n);
do
s+ = i++;
while (i< = n);
printf("\n s = %d",s);
return 0;
}
```

Bu dasturning kamchiligi shundan iboratki agar n qiymati 0 ga teng yoki manfiy bo'lsa ham, sikl tanasi bir marta bajariladi va s qiymati birga teng bo'ladi.

Parametrli for operatori. Parametrli for operatori umumiy ko'rinishi quyidagicha:

for(1-ifoda;2- ifoda; 3-ifoda)
Operator
Bu operator quyidagi operatorga mosdir.

1-ifoda;
while(2-ifoda) {
operator
3-ifoda
}

Misol. Berilgan n gacha sonlar yig'indisi.

```
#include <stdio.h>
int main()
{
int n;
scanf("%d",&n);
int s = 0;
for(int i = 1;i< = n; i++) s+ = i;
printf("\n%d",s);
return 0;
}
```

Sikldabirnechtaschyotchikniqo'llanilishi.

Parametrliforsikliningsintaksisiundabirnechtao'zgaruvchi - schyotchikniqo'llanilishiga, siklnidavometishinimurakkabshartlarinitekshirishgavasiklschyotchiklariustidaketma-ketbirnechtaoperasiyanibajarilishigaimkonberadi.

Agardabirnechtaschyotchikkaqiymato'zlashtirilsayokiularo'rtasidabirnechtaoperasiyabajarilsa, buifodalarvergulbilanajratilganholdaketma – ketyoziladi.

for siklidabirnechtaschyotchikniqo'llanilishi

```
#include <stdio.h>
#include<conio.h>
```

```
int main()
{
int i,j;
for (i = 0, j = 0; i<3; i++, j++)
printf("i:%d j:%d \n",i,j);
getch();
return 0;
}
```

Hatija:

```
i: 0        j: 0
i: 1        j: 1
i: 2    j: 2
```

8.2 O’tishoperatorlari

Uzish break operatori. Ba’zi hollarda sikl bajarilishini ixtiyoriy joyda to’xtatishga to’g’ri keladi. Bu vazifani break operatori bajarishga imkon beradi. Bu operator darxol sikl bajarilishini to’xtatadi va boshqaruvni sikldan keyingi operatorlarga uzatadi.

Misol:

```
#include <stdio.h>
int main()
{
int n;
while(1)
{
scanf("%d",&n);
if(n == 1||n == 0) break;
}
printf("Sikl tugadi");
return 0;
}
```

Bumisolda while(1) operatoriyordamidacheksizsiklhosilqilinadi. Agar 1 yoki 0 sonikiritilsasiklto’xtatiladi.

Qaytarish continue operatori. Siklbajarilishigata’siro’tkazishgaimkonberadiganyanabiroperator continue operatoridir. Buoperatorsiklqadaminibajarilishinito’xtatib for va while dako’rsatilganshartlitekshirishgao’tkazadi.

Misol:

```
#include <stdio.h>
int main()
{
int n;
for(;;)
{
scanf("%d",&n);
if(n == 1||n == 0) continue;
break;
}
printf("Sikl tugadi");
return 0;
}
```

Bumisolda for(;;)operatoriyordamidacheksizsiklhosilqilinadi. Agar 1 yoki 0 sonlardan farqli sonkiritilsasiklto'xtatiladi.

O'tishoperatorigoto. O'tish operatorining ko'rinishi:

goto<identifikator>. Bu operator identifikator bilan belgilangan operatorga o'tish kerakligini ko'rsatadi.

Misol uchun gotoA1;…;A1:y = 5;

Strukturali dasturlashda goto operatoridan foydalanmaslik maslahat beriladi. Lekin ba'zi hollarda o'tish operatoridan foydalanish dasturlashni osonlashtiradi.

Misol uchun bir necha sikldan birdan chiqish kerak bo'lib qolganda, to'g'ridan-to'g'ri break operatorini qo'llab bo'lmaydi, chunki u faqat eng ichki sikldan chiqishga imkon beradi.

```
#include <stdio.h>
int main()
{
int n = 16,s = 0;
int i,j;
for(i = 1;i<5;i++)
for(j = 1;j<5;j++)
{
if(i*j>n) goto A;
C++;
}
```

```
A:printf("Sikl tugadi s = %d",s);
return 0;
}
```

10-Ma’ruza. Umumiy takrorlanish algoritmlari va ichma-ich takrorlanishlar

Ma’ruza rejasi:

10.1 C++tilida takrorlanuvchi jarayonlarni dasturlash

10.2 Continue operatori

Kalit so’zlar: *delete, masofa keltirish, delete[], new, indeks, this, indeksirlash, [] bo’sh xotira, void*, konteyner, 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.*

8.1 C++tilida takrorlanuvchi jarayonlarni dasturlash

Agar dastur bajarilish jarayonida operator yoki operatorlar guruhi bir necha marta qayta-qayta bajarilsa, bunday jarayonlarni takrorlanuvchi (siklik) jarayon deyiladi. C++ tilida siklni 3 xil ko’rinishda tashkil qilish mumkin.

1. Sharti avval tekshiriladigan takrorlanish (oldshartli sikl):

while (shart) operator (lar);

Bu yerda operatorlar while da ko’rsatilgan shart yolg’on bo’lgunicha takrorlanadi. Takrorlanish tanasi murakkab bo’lsa, ya’ni 1 tadan ortiq operatorlar qatnashsa, ularni alohida {} ichiga olish kerak bo’ladi.

Masalan: b = 2*(a+5); a € [1, 10]; h=1;

```
# include <iostream.h>
# include <math.h>
void main ( )
{  int a=1, b;
 while (a<=10)
 {  b = 2*(a+5); cout << “b=” <<b;
 cout << “a=” <<a << endl;
 a++ ; }
}
```

Ekranda 10 ta a va b larning qiymatlari paydo bo’ladi.

2- misol.

```
# include <iostream.h>
 void main ( )
 {  int i = 10;
```

while (i++ < =15)
cout << “Salom!!!”<< endl; }

Ekranda 5 marta “Salom!!!” yozuvi paydo bo’ladi.

2. Sharti keyin tekshiriladigan takrorlanish (so’ngshartli sikl):

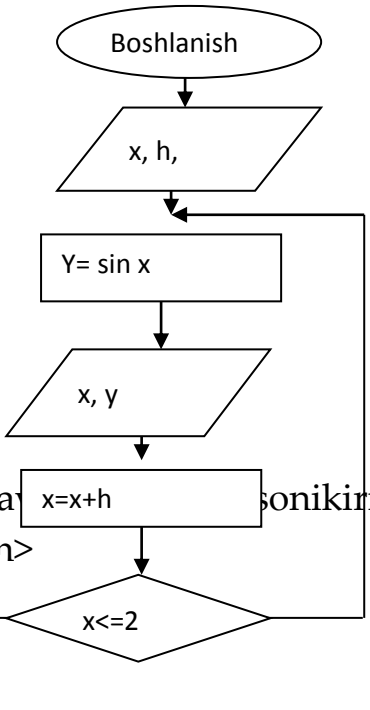
do
operator (lar)
while (shart);

Takrorlanish while da ko’rsatilgan shart yolg’on bo’lgunicha davom etadi.

Masalan: y=sinx; x € [1,2] ; h=0.1

```
# include <iostream.h>
# include <math.h>
void main ( )
{ float x=1, y;
do
  { y=sin(x); cout << “x=”<<x<<”  y=”<<y<<endl;    x+ = 0.1;  }
   while (x<=2); getch(); }
```

Masalaning algoritmi quyidagi ko’rinishga ega bo’ladi:



2- misol. Dastur klav sonikiritilishini kutadi.

```
# include <iostream>
void main (
{ int n;
  do
{ cin >> n;
```

```
cout << “Sonni qayta kiriting!=”<<n; }
while (n!=20);
}
```

3. Parametrli takrorlanish (sikl):

Umumiy ko’rinishi

```
for (bosh qiymat; shart; o’zgarish qadami)
operator (lar);
```

Operatorlar 1 tadan ortiq bo’lsa ularni alohida qavslar -{} ichiga olinadi.

1-misol. Y = cos x; x € [2,3] ; h=0,2;

```
# include <iostream.h>
# include <math.h>
void main ( )
{ float x, y;
for (x=2; x<=3; x+ = 0.2)
  { y=cos(x);  cout << “x=”<< x << “  y=”<< y << endl; }
}
```

2-misol. 100 gacha bo’lgan juft sonlarni ekranga chiqarish dasturi.

```
# include <iostream.h>
  void main ( )
 {   int i = 2;
  while (i<=100)
  { cout << “i=”<< i;  i += 2; }  }
```

…. for (int i=2; i<=100; i +=2) cout << “i=”<< i;	do cout << “i=”<< i; i += 2; while (i<=100);

3-misol: 1 dan 100 gacha bo’lgan 3 raqami bilan tugaydigan sonlarni ekranga chiqarish dasturini tuzing (2 xil usulda).

……
int i=3;
while (i <=100)
{ cout << “i=”<< i;
  i + = 10; }
	…..
	for (i = 3;i <= 100; i + = 10)
	cout << “i=”<< i;

4-misol. Qadimiy masala. Bir odam 100 so’m bilan bozorga bordi. Bozorda 1 ta sigir 10 so’m, 1 ta qo’y 3 so’m, 1 ta echki 0.5 so’m va xarid qilingan qoramollarning umumiy soni 100 ta bo’lsa, nechta sigir, qo’y va echki sotib olindi?

Sigirlar soni: x, qo’ylar soni y, echkilar soni z deb olinsa,

```
# include <iostream.h>
int main ( )
{
int x, y, z, s;
for (x=1; x<=100; x++)
for (y=1; y<=100; y++)
if (19*x + 5*y = = 100)
{ z = 100 – x – y;
  cout << “x=”<<x;
  cout << “y=”<<y;
cout << “z=”<<z; }
   return 0;
   }
```

```
# include <iostream.h>
int main ( )
{
int x, y, z, s;
for (x=1; x<=100; x++)
for (y=1; y<=100; y++)
for (z=1; z<=100; z++)
if (x + y + z = = 100)
{  cout << “x=”<<x;
cout << “y=”<<y;
  cout << “z=”<<z; }
return 0;
   }
```

10.2 Continue operatori

Bu operator yordamida sikl parametrining biror qiymatida hisoblashni to’xtatib, keyingi qiymatida hisoblashni davom ettirish mumkin. Masalan: y = 2x funksiyasini x € [1,18] oraliqda h=1 qiymat bilan hisoblash kerak, lekin x=6 yoki x=13 qiymatlarida hisoblashni bajarmaslik kerak.

```
# include <iostream.h>
void main ( )
{ int x, y;
for (x=1; x<=18; x++)
 { if (( x = = 6) || (x = = 13)) continue;
y = 2*x; cout << “x=”<< x << “  y=” << y << endl;
 } }
```

2 - misol. 10 ta ketma-ket kiritiladigan butun musbat sonlar yig’indisini hisoblash dasturini tuzing. Agar son manfiy bo’lsa, yig’indiga qo’shmaslik kerak.

```
# include <iostream.h>
void main ( )
```

```
{   int x, i, s=0;
for ( i =1; i <=10; i ++)
  { cin >> x;
if ( x < 0)  continue;
  s = s + x; }          // s +=x deb yozsa ham bo'ladi.
  cout << "s="<< s << endl;    }
```

3-misol. Y = x^n funksiyasini rekurrent formula orqali hisoblash dasturini tuzing. Bu yerda n - butun son, x – ixtiyoriy haqiqiy son.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ float x=2.56, y=1;
for ( int n=1; n<=10; n++)
y = y * x;              // y * = x;
cout <<"y="<<y<<endl;
getch ( );  }
```

4-misol. $Y= \sum_{n=1}^{8} \frac{x^2}{n!} = x^2 + \frac{x^2}{1*2} + \frac{x^2}{1*2*3} + ... + \frac{x^2}{1*2*3*4*5*6*7*8}$

x - ixtiyoriy haqiqiy son.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ float x=3.75, y=0; long p=1;
for ( int n=1; n<=8; n++)
{ p = p * n; y = y + x*x / p; }
cout << "y="<<y<<endl;
getch ( );
}
```

5-misol. $S=\cos x + \frac{\cos 2x}{2} + \frac{\cos 3x}{3} + ... + \frac{\cos nx}{n}$;

bu yerda $\frac{\pi}{5} \le x \le \frac{9\pi}{5}$ n=10

```
# include <iostream.h>
# include <conio.h>
# include <math.h>
   void main ( )
{ float a, b, h, x, s, pi=3.14;
```

```
a = pi / 5;  b=9 * pi / 5; h=(b-a) / 10;
x = a; cout. precision (3);
while ( x<=b )
{ s = 0;
for ( int n=1; n<=10; n++)
s = s + cos(n*x) / n;
cout <<"s="<<s<<endl;
x = x + h;
}
getch ( ); }
```

6-misol. Boy bilan olim bahslashibdilar. Olim boyga har kuni (30 kun) 100000 so'm beradigan bo'libdi. Boy esa olimga 1-kun 1 tiyin, 2-kun 2 tiyin, 3-kun 4 tiyin, 4-kun 8 tiyin va h.k. pul beradigan bo'libdi. Bahsda kim yutadi? Dasturini tuzing.

Olim → 30*100000 = 3000000 so'm

Boy → $\sum_{i=0}^{30} 2^i$ sum → 10737418 so'm

```
# include <iostream.h>
# include <conio.h>
# include <math.h>
void main ( )
{ int s, s1=1, k = 0;
for ( int i=1; i<=30; i++)
{ k = k + s1;
s1 = s1*2; }
k = k / 100;
s = 30*100000;
cout <<"boy olimga beradi:"<<k<<endl;
cout <<"olim boyga beradi:" << s << endl;
getch ( );
}
```

Nazorat savollari:

1. Sharti avval tekshiriladigan takrorlanish
2. Sharti keyin tekshiriladigan takrorlanish
3. Parametrli takrorlanish
4. Dasturda takrorlanishlarni tashkil etish.
5. Takrorlanuvchi dastur nima?

6. Murakkab takrorlanishlar
7. **continue** operatori
8. **return** operatori

11,12-Ma’ruza. C++ da funksiyalar: tuzilishi va undan foydalanish. . Qiymat qaytarmaydigan funksiyalar va ular yordamida masala yechish

Ma’ruza rejasi:

11.1 Funksiya tushunchasi

11.2 Funksiya parametrlari

11.3 Funksiyadan foydalanish

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.*

Programma ta’minotini yaratish amalda murakkab jarayon hisoblanadi. Programma tuzuvchi programma kompleksini bir butunlikdagi va uning har bir bo‘lagining ichki mazmunini va ularning sezilmas farqlarini hisobga olishi kerak bo‘ladi.

Programmalashga tizimli yondoshuv shundan iboratki, program-ma tuzuvchi oldiga qo‘yilgan masala oldindan ikkita, uchta va undan ortiq nisbatan kichik masala ostilarga bo‘linadi. O‘z navbatida bu masalaostilari ham yana kichik masalaostilariga bo‘linishi mumkin. Bu jarayon toki mayda masalalarni oddiy standart amallar yordamida echish mumkin bo‘lguncha davom etadi. SHu yo‘l bilan masalani dekompozitsiyalash amalga oshiriladi.

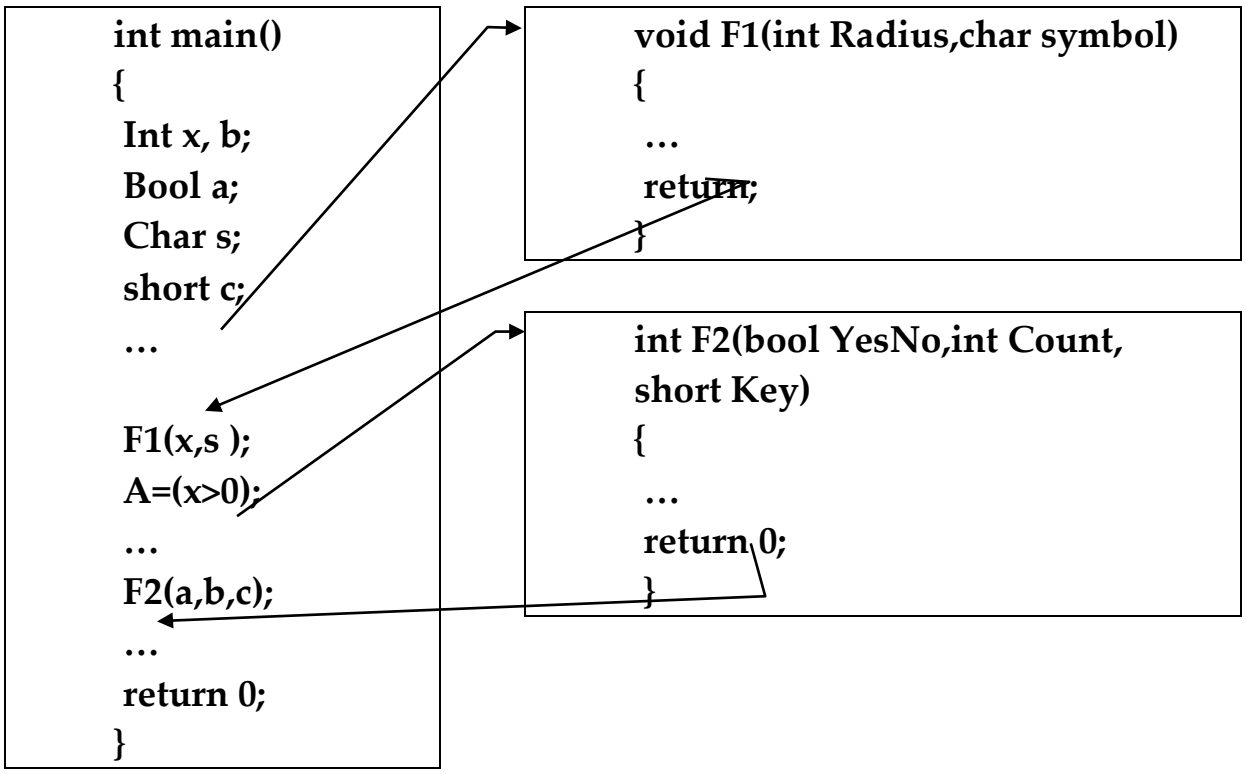
Ikkinchi tomondan, programmalashda shunday holatlar kuzatiladiki, unda programmaning turli joylarida mazmunan bir xil algo-ritmlarni bajarishga to‘g‘ri keladi. Algoritmning bu bo‘laklari asosiy echilayotgan masaladan ajratib olingan qandaydir masala ostini echishga mo‘ljallangan bo‘lib, etarlicha mustaqil qiymatga (natijaga) egadir. Misol uchun quyidagi masalani ko‘raylik:

Berilgan $a_0,a_1,...,a_{30}$, $b_0,b_1,...,b_{30}$, $c_0,c_1,...,c_{30}$ va x,y,z haqiqiy sonlar uchun

$$\frac{\left(a_0x^{30}+a_1x^{29}+...+a_{30}\right)^2-\left(b_0y^{30}+b_1y^{29}+...+b_{30}\right)}{c_0(x+z)^{30}+c_1(x+z)^{29}+...+c_{30}}$$

ifodaningqiymatihisoblansin.

Funksiya tanasidagi return operatori yoki oxirgi operator bajargandan keyin avtomatik ravishda bosh funksiyaga qaytish amalga oshiriladi.



11.1-rasm. Bosh funksiyadan boshqa funksiyalarni chaqirish va qaytish

Aksariyat hollarda main() funksiyasining parametrlar ro‘yxati bo‘sh bo‘ladi. Agar yuklanuvchi programmani ishga tushirishda,buyruq satri orqali yuklanuvchi programma ishga tushirilganda, unga parametrlarni uzatish (berish) zarur bo‘lsa, main() programmasi funksiyasining sintaksisi o‘zgaradi:

int main(int argc, char* argv[]);

Bu erda argc - uzatiladigan parametrlar soni, argv[]- bir-biridan punktuatsiya belgilari (va probel) bilan ajratilgan parametrlar ro‘yxatini o‘z ichiga olgan massivga ko‘rsatkich.

Quyida funksiyalarni e’lon qilish, chaqirish va aniqlashga misollar keltirilgan:

```
// funksiyalar e’loni
int Mening_funksiyam(int Number, float Point);
char Belgini_uqish();
void bitni_urnatish(short Num);
void Amal_yoq(int,char);
// funksiyalarni chaqirish
result=Mening_funksiyam(Varb1,3.14);
symb=Belgini_uqish();
bitni_urnatish(3);
Amal_yoq(2,Smbl);
```

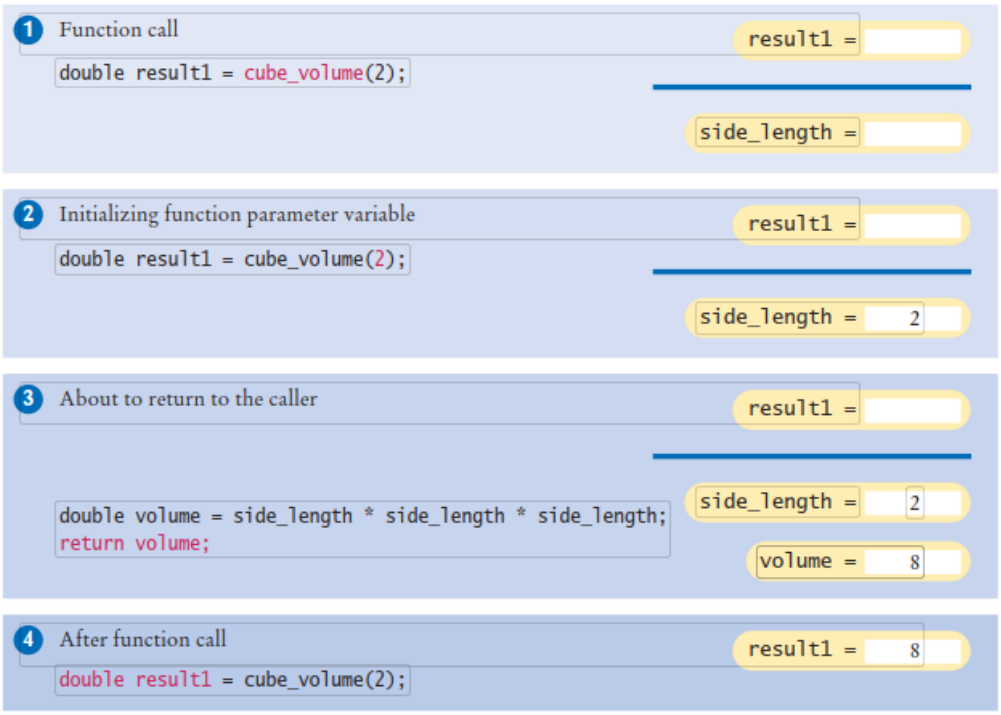
```
// funksiyalarni aniqlash
int Mening_funksiyam(int Number,float Point);
{int x;
...
 return x;}
char Belgini_uqish()
{
char Symbol;
 cin>>Symbol;
 return Symbol;
};
void bitni_urnatish(short number)
{
global_bit=global_bit | number;
};
void Amal_yoq(int x, char ch){};
```

Misollar

ch05/cube.cpp

```
1 #include <iostream>
2
3using namespace std;
4
5 /**
6Computes the volume of a cube.
7@param side_length the side length of the cube
8@return the volume
9 */
10double cube_volume(double side_length)
11 {
12double volume = side_length * side_length * side_length;
13return volume;
14 }
15
16int main()
17 {
18double result1 = cube_volume(2);
```

```
19double result2 = cube_volume(10);
20 cout <<"A cube with side length 2 has volume "<< result1 << endl;
21 cout <<"A cube with side length 10 has volume "<< result2 << endl;
22
23return 0;
24 }
```



ch05/intname.cpp

```
1 #include <iostream>
2 #include <string>
3
4using namespace std;
5
6 /**
7Turns a digit into its English name.
8@param digit an integer between 1 and 9
9@return the name of digit (“one” ... “nine”)
10 */
11 string digit_name(int digit)
12 {
13if (digit == 1) return"one";
14if (digit == 2) return"two"; 15if (digit == 3) return"three";
16if (digit == 4) return"four";
17if (digit == 5) return"five"; 18if (digit == 6) return"six";
19if (digit == 7) return"seven";
20if (digit == 8) return"eight"; 21if (digit == 9) return"nine";
22return"";
23 }
24
25 /**
26Turns a number between 10 and 19 into its English name.
27@param number an integer between 10 and 19
```

```
28@return the name of the given number (“ten” ... “nineteen”)
29 */
30 string teen_name(int number)
31 {
32if (number == 10) return"ten";
33if (number == 11) return"eleven";
34if (number == 12) return"twelve";
35if (number == 13) return"thirteen";
36if (number == 14)return"fourteen";
37if (number == 15) return"fifteen";
38if (number == 16) return"sixteen";
39if (number == 17) return"seventeen";
40if (number == 18) return"eighteen";
41if (number == 19) return"nineteen";
42return"";
43 }
44
45 /**
46Gives the name of the tens part of a number between 20 and 99.
47@param number an integer between 20 and 99
48@return the name of the tens part of the number (“twenty” ... “ninety”) 49 */
50 string tens_name(int number)
51 {
52if (number >= 90) return"ninety";
53if (number >= 80) return"eighty";
54if (number >= 70) return"seventy";
55if (number >= 60) return"sixty";
56if (number >= 50) return"fifty";
57if (number >= 40) return"forty";
58if (number >= 30) return"thirty";
59if (number >= 20) return"twenty";
60return"";
61 }
62
63 /**
64Turns a number into its English name.
65@param number a positive integer < 1,000
66@return the name of the number (e.g. “two hundred seventy four”)
67 */
68 string int_name(int number)
69 {
```

```
70int part = number; // The part that still needs to be converted
71string name; // The return value
72
73if (part >= 100)
74 {
75name = digit_name(part / 100) + " hundred"; 76part = part % 100;
77 }
78
79if (part >= 20)
80 {
81name = name + " " + tens_name(part);
82part = part % 10;
83 }
84else if (part >= 10)
85 {
86name = name + " " + teen_name(part);
87part = 0;
88 }
89
90if (part >0)
91 {
92name = name + " " + digit_name(part);
93 }
94
95return name;
96 }
97
98int main()
99 {
100 cout <<"Please enter a positive integer: "; 101int input;
102 cin >> input;
103 cout << int_name(input) << endl;
104return0;
105 }
```

program run
Please enter a positive integer: 729
seven hundred twenty nine

Nazorat savollari

1. C++da funksiya qanday ishlaydi?
2. funksiyaga kutubxona kerakmi?
3. For operatori funksiyada qanday ishlatiladi?
4. Matematik funksiyalar qanday ishlaydi?
5. Funksiya parametrlar nima?
6. Funksiya qanday chaqiriladi?
7. Funksiya parametrlari orqali nima uzatiladi?
8. If operatorining nechta turi bor?

9. O’zgaruvchilar nima uchun qo’llaniladi?

13-Ma’ruza. Funksiyalarda argument sifatida local, global o’zgaruvchilardan va havolalardan foydalanish.

Ma’ruza rejasi:

13.1 Lokal o’zgaruvchilar

13.2 Global o’zgaruvchilar

13.3 Havolalar tushunchasi

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.*

Ko‘rinish sohasi. Lokal va global o‘zgaruvchilar

O‘zgaruvchilar funksiya tanasida yoki undan tashqarida e’lon qilinishi mumkin. Funksiya ichida e’lon qilingan o‘zgaruvchilarga *lokal o‘zgaruvchilar* deyiladi. Bunday o‘zgaruvchilar xotiradagi prog-ramma stekida joylashadi va faqat o‘zi e’lon qilingan funksiya tanasida amal qiladi. Boshqaruv asosiy funksiyaga qaytishi bilan lokal o‘zgaruvchilar uchun ajratilgan xotira bo‘shatiladi (o‘chiriladi).

Har bir o‘zgaruvchi o‘zining amal qilish sohasi va yashash vaqti xususiyatlari bilan xarakterlanadi.

O‘zgaruvchi *amal qilish sohasi* deganda o‘zgaruvchini ishlatish mumkin bo‘lgan programma sohasi (qismi) tushuniladi. Bu tushuncha bilan o‘zgaruvchining *ko‘rinish sohasi* uzviy bog‘langan. O‘zgaruvchi amal qilish sohasidan chiqqanda ko‘rinmay qoladi. Ikkinchi tomondan, o‘zgaruvchi amal qilish sohasida bo‘lishi, lekin ko‘rinmas-ligi mumkin. Bunda ko‘rinish sohasiga ruxsat berish amali «::» yordamida ko‘rinmas o‘zgaruvchiga murojat qilish mumkin bo‘ladi.

O‘zgaruvchining *yashash vaqti* deb, u mavjud bo‘lgan programma bo‘lagining bajarilishiga ketgan vaqt intervaliga aytiladi.

Lokal o‘zgaruvchilar o‘zlari e’lon qilingan funksiya yoki blok chegarasida ko‘rinish sohasiga ega. Blokdagi ichki bloklarda xuddi shu nomdagi o‘zgaruvchi e’lon qilingan bo‘lsa, ichki bloklarda bu lokal o‘zgaruvchi ham amal qilmay qoladi. Lokal o‘zgaruvchi yashash vaqti - blok yoki funksiyani bajarish vaqti bilan aniqlanadi. Bu hol shuni anglatadiki, turli funksiyalarda bir-biriga umuman bog‘liq bo‘lma-gan bir xil nomdagi lokal o‘zgaruvchilarni ishlatish mumkin.

Quyidagi programmada main() va sum() funksiyalarida bir xil nomdagi o‘zgaruvchilarni ishlatish ko‘rsatilgan. Programmada ikkita sonning yig‘indisi hisoblanadi va chop etiladi:

```
#include <iostream.h>
// funksiyaprototipi
int sum(int a;int b);
int main()
{
 // lokal o‘zgaruvchilar
 int x=r;
int y=4;
 cout<<sum(x, y);
 return 0;
}
int sum(int a,int b)
{
 // lokal o‘zgaruvchi
 int x=a+b;
 return x;
}
```

Global o‘zgaruvchilar programma matnida funksiya aniqlanishi-dan tashqarida e’lon qilinadi va e’lon qilingan joyidan boshlab programma oxirigacha amal qiladi.

```
#include <iostream.h>
int f1(); int f2();
int main()
{
 cout<<f1()<<" "<<f2()<<endl;
 return 0;
}
int f1()
{
 return x;// kompilyasiya xatosi ro‘y beradi
}
int x=10; // global o‘zgaruvchi e’loni
int f2(){ return x*x;}
```

YUqorida keltirilgan programmada kompilyasiya xatosi ro‘y beradi, chunki f1() funksiya uchun x o‘zgaruvchisi noma’lum hisob-lanadi.

Programma matnida global o‘zgaruvchilarni ular e’lonidan keyin yozilgan ixtiyoriy funksiyada ishlatish mumkin. SHu sababli, global o‘zgaruvchilar programma matnining boshida yoziladi. Funksiya ichidan

global o‘zgaruvchiga murojat qilish uchun funksiyada uning nomi bilan mos tushadigan lokal o‘zgaruvchilar bo‘lmasligi kerak. Agar global o‘zgaruvchi e’lonida unga boshlang‘ich qiymat berilmagan bo‘lsa, ularning qiymati 0 hisoblanadi. Global o‘zgaruvchining amal qilish sohasi uning ko‘rinish sohasi bilan ustma-ust tushadi.

SHuni qayd etish kerakki, tajribali programma tuzuvchilar imkon qadar global o‘zgaruvchilarni ishlatmaslikka harakat qilishadi, chunki bunday o‘zgaruvchilar qiymatini programmaning ixtiyoriy joyidan o‘zgartirish xavfi mavjudligi sababli programma ishlashida mazmunan xatolar yuzaga kelishi mumkin. Bu fikrimizni tasdiqlovchi programmani ko‘raylik.

```
# include <iostream.h>
// global o‘zgaruvchi e’loni
inttest=100;
void Chop_qilish(void );
int main()
{
 //lokal o‘zgaruvchi e’loni
 int test=10;
 //global o‘zgaruvchi chop qilish funksiyasini chaqirish
 Chop_qilish();
 sout<<”Lokal o’zgaruvchi: ”<<test<<‘\n’;
 return 0;
}
void Chop_qilish(void)
{
 cout<<”Global o’zgaruvchi: ”<<test<<‘\n’;
}
```

Programma boshida test global o‘zgaruvchisi 100 qiymati bilan e’lon qilinadi. Keyinchalik, main() funksiyasida test nomi bilan lokal o‘zgaruvchisi 10 qiymati bilan e’lon qilinadi. Programmada, Chop_qilish() funksiyasiga murojaat qilinganida, asosiy funksiya tanasidan vaqtincha chiqiladi va natijada main() funksiyasida e’lon qilingan barcha lokal o‘zgaruvchilarga murojaat qilish mumkin bo‘lmay qoladi. SHu sababli Chop_qilish() funksiyasida global test o‘zgaruvchisining qiymatini chop etiladi. Asosiy programmaga qaytilgandan keyin, main() funksiyasidagi lokal test o‘zgaruvchisi global test o‘zgaruvchisini «berkitadi» va lokal test

o‘zgaruvchini qiymati chop etiladi. Programma ishlashi natijasida ekranga quyidagi natijalar chop etiladi:

Global o‘zgaruvchi: 100

Lokal o‘zgaruvchi: 10

:: amali

YUqorida qayd qilingandek, lokal o‘zgaruvchi e’loni xuddi shu nomdagi global o‘zgaruvchini «berkitadi» va bu joydan global o‘zgaruvchiga murojat qilish imkoni bo‘lmay qoladi. C++ tilida bunday holatlarda ham global o‘zgaruvchiga murojat qilish imko-niyati saqlanib qolingan. Buning uchun «ko‘rinish sohasiga ruxsat berish» amalidan foydalanish mumkin va o‘zgaruvchi oldiga ikkita nuqta - «::» qo‘yish zarur bo‘ladi. Misol tariqasida quyidagi programani keltiramiz:

```
#include <iostream.h >
//global o‘zgaruvchi e’loni
int uzg=5;
int main()
{
//lokal o‘zgaruvchi e’loni
 int uzg=70;
 //lokal o‘zgaruvchini chop etish
 cout<<uzg<<‘/n’;
 //global o‘zgaruvchini chop etish
 cout<<::uzg <<’/n’;
 return 0;
}
```

Programma ishlashi natijasida ekranga oldin 70 va keyin 5 sonlari chop etiladi.

Xotira sinflari

O‘zgaruvchilarning ko‘rinish sohasi va amal qilish vaqtini aniqlovchi o‘zgaruvchi modifikatorlari mavjud (5.1-jadval).

5.1-jadval. O‘zgaruvchi modifikatorlari

Modifikator	**Qo‘llanishi**	**Amal qilish sohasi**	**YAshash davri**
auto	lokal	blok	vaqtincha
register	lokal	blok	vaqtincha
exteru	Global	blok	vaqtincha
static	Lokal	blok	doimiy
	Global	fayl	doimiy

volatile	Global	fayl	doimiy

Avtomat o‘zgaruvchilar. auto modifikatori lokal o‘zgaruvchilar e’lonida ishlatiladi. Odatda lokal o‘zgaruvchilar e’lonida bu modifikator kelishuv bo‘yicha qo‘llaniladi va shu sababli amalda uni yozishmaydi:

```
#include <iostream.h>
int main()
{
 auto int X=2; // int X=2; bilan ekvivalent
cout<<X;
returu 0;
}
```

auto modifikatori blok ichida e’lon qilingan lokal o‘zgaruvchi-larga qo‘llaniladi. Bu o‘zgaruvchilar blokdan chiqishi bilan avtoma-tik ravishda yo‘q bo‘lib ketadi.

Registr o‘zgaruvchilar. register modifikatori kompilyatorga, ko‘rsatilgan o‘zgaruvchini protsessor registrlariga joylashtirishga harakat qilishni tayinlaydi. Agar bu harakat natija bermasa o‘zga-ruvchi auto turidagi lokal o‘zgaruvchi sifatida amal qiladi.

O‘zgaruvchilarni registrlarda joylashtirish programma kodini bajarish tezligi bo‘yicha optimallashtiradi, chunki protsessor xotiradagi berilganlarga nisbatan registrdagi qiymatlar bilan ancha tez ishlaydi. Lekin registrlar soni cheklanganligi uchun har doim ham o‘zgaruvchilarni registrlarda joylashtirishning iloji bo‘lmaydi.

```
#include < iostream.h >
int main()
{
 register int Reg;
 ...
 return 0;
}
```

registermodifikatori faqat lokal o‘zgaruvchilariga nisbatan qo‘llaniladi, global o‘zgaruvchilarga qo‘llash kompilyasiya xatosiga olib keladi.

Tashqi o‘zgaruvchilar. Agar programma bir nechta moduldan iborat bo‘lsa, ular qandaydir o‘zgaruvchi orqali o‘zaro qiymat alma-shishlari mumkin (fayllar orasida). Buning uchun o‘zgaruvchi birorta modulda global tarzda e’lon qilinadi va u boshqa faylda (modulda) ko‘rinishi uchun u erda extern modifikatori bilan e’lon qilinishi kerak bo‘ladi. extern

modifikatori o‘zgaruvchini boshqa faylda e’lon qilinganligini bildiradi. Tashqi o‘zgaruvchilar ishlatilgan prog-rammani ko‘raylik.

```
//Sarlavha.h faylida
void Bayroq_Almashsin(void);
// modul_1.cpp faylida
bool Bayroq;
void Bayroq_Almashsin(void){Bayroq=!Bayroq;}
// masala.cpp faylida
#include < iostream.h>
#include <Sarlavha.h>
#include <modul_1.cpp>
extern bool Bayroq;
int main()
{
Bayroq_Almashsin();
 if(Bayroq)
 cout<<”Bayroq TRUE”<<endl;
 else cout<<”Bayroq FALSE”<<endl;
 return 0;
}
```

Oldin sarlavha.h faylida Bayroq_Almashsin( ) funksiya sarlav-hasi e’lon qilinadi, keyin modul_1.srr faylida tashqi o‘zgaruvchi e’lon qilinadi va Bayroq_Almashsin() funksiyasining tanasi aniqla-nadi va nihoyat, masala.cpp faylida Bayroq o‘zgaruvchisi tashqi deb e’lon qilinadi.

Statik o‘zgaruvchilar. Statik o‘zgaruvchilar static modifika-tori bilan e’lon qilinadi va o‘z xususiyatiga ko‘ra global o‘zgaruvchi-larga o‘xshaydi. Agar bu turdagi o‘zgaruvchi global bo‘lsa, uning amal qilish sohasi - e’lon qilingan joydan programma matnining oxirigacha bo‘ladi. Agar statik o‘zgaruvchi funksiya yoki blok ichida e’lon qilinadigan bo‘lsa, u funksiya yoki blokka birinchi kirishda initsializatsiya qilinadi. O‘zgaruvchining bu qiymati funksiya keyingi chaqirilganida yoki blokka qayta kirishda saqlanib qoladi va bu qiymatni o‘zgartirish mumkin. Statik o‘zgaruvchilarni tashqi deb e’lon qilib bo‘lmaydi.

Agar statik o‘zgaruvchi initsializatsiya qilinmagan bo‘lsa, uning birinchi murojatdagi qiymati 0 hisoblanadi.

Misol tariqasida birorta funksiyani necha marotaba chaqiril-ganligini aniqlash masalasini ko‘raylik:

```
#include <iostream.h >
```

```
int Sanagich(void);
int main()
{
int natija;
 for (int i=0; i<30; i++)
 natija=Sanagich();
 cout<<natija;
 return 0;
}
int Sanagich(void)
{
 static short sanagich=0;
 ...
 sanagich++;
 return sanagich;
}
```

Bu erda asosiy funksiyadan counter statik o‘zgaruvchiga ega Sanagicht() funksiyasi 30 marta chaqiriladi. Funksiya birinchi marta chaqirilganda sanagich o‘zgaruvchiga 0 qiymatini qabul qiladi va uning qiymati birga ortgan holda funksiya qiymati sifatida qaytariladi. Statik o‘zgaruvchilar qiymatlarini funksiyani bir chaqirilishidan ikkinchisiga saqlanib qolinishi sababli, keyingi har bir chaqirishlarda sanagich qiymati bittaga ortib boradi.

Masala. Berilgan ishorasiz butun sonning barcha tub bo‘luv-chilari aniqlansin. Masalani echish algoritmi quyidagi takrorla-nuvchi jarayondan iborat bo‘ladi: berilgan son tub songa (1-qadamda 2 ga) bo‘linadi. Agar qoldiq 0 bo‘lsa, tub son chop qilinadi va bo‘linuv-chi sifatida bo‘linma olinadi, aks holda navbatdagi tub son olinadi. Takrorlash navbatdagi tub son bo‘linuvchiga teng bo‘lguncha davom etadi.

Programma matni:

```
#include<iostream.h>
#include<math.h>
int Navb_tub();
int main()
{
 unsigned int n,p;
 cout<<"\nn qiymatini kiritng: ";
cin>>n;
```

```
 cout<<"\n1";
 p=Navb_tub();
 while(n>=p)
 {
  if(n%p==0)
  {
   cout<<"*"<<p;
   n=n/p;
  }
  else p=Navb_tub();
  }
 return 0;
}
int Navb_tub()
{
 static unsigned int tub=1;
 for(;;)
 {
  tub++;
  short int ha_tub=1;
  for(int i=2;i<=tub/2;i++)
if(tub%i==0)ha_tub=0;
  if(ha_tub)return tub;
 }
 return 0;
}
```

Programmada navbatdagi tub sonni hosil qilish funksiya ko‘ri-nishida amalga oshirilgan. Navb_tub() funksiyasining har chaqirili-shida oxirgi tub sondan keyingi tub son topiladi. Oxirgi tub sonni «eslab» qolish uchun tub o‘zgaruvchisi static qilib aniqlangan.

Programma ishga tushganda klaviaturadan n o‘zgaruvchisining qiymati sifatida 60 soni kiritilsa, ekranga quyidagi ko‘paytma chop etiladi:

1*2*2*3*5

volatile sinfi o‘zgaruvchilari. Agar programmada o‘zgaruvchini birorta tashqi qurilma yoki boshqa programma bilan bog‘lash uchun ishlatish zarur bo‘ladigan bo‘lsa, u volatile modifikatori bilan e’lon qilinadi. Kompilyator bunday modifikatorli o‘zgaruvchini registrga

joylashtirishga harakat qilmaydi. Bunday o‘zgaruvchilar e’loniga misol quyida keltirilgan:

volatile short port_1;

volatile const int Adress=0x00A2;

Misoldan ko‘rinib turibdiki, volatile modifikatorli o‘zgarmas ham e’lon qilinishi mumkin.

onsider this sample program:

```
1   int x;
2   int mystery(int x)
3   {
4    int s = 0;
5    for (int i = 0; i < x; i++)
6    {
7   int x = i + 1;
8   s = s + x;
9    }
10    return x;
11   }
12   int main()
13   {
14    x = 4;
15    int s = mystery(x);
16    cout << s << endl;
17   }
```

28. Which line defines a global variable?

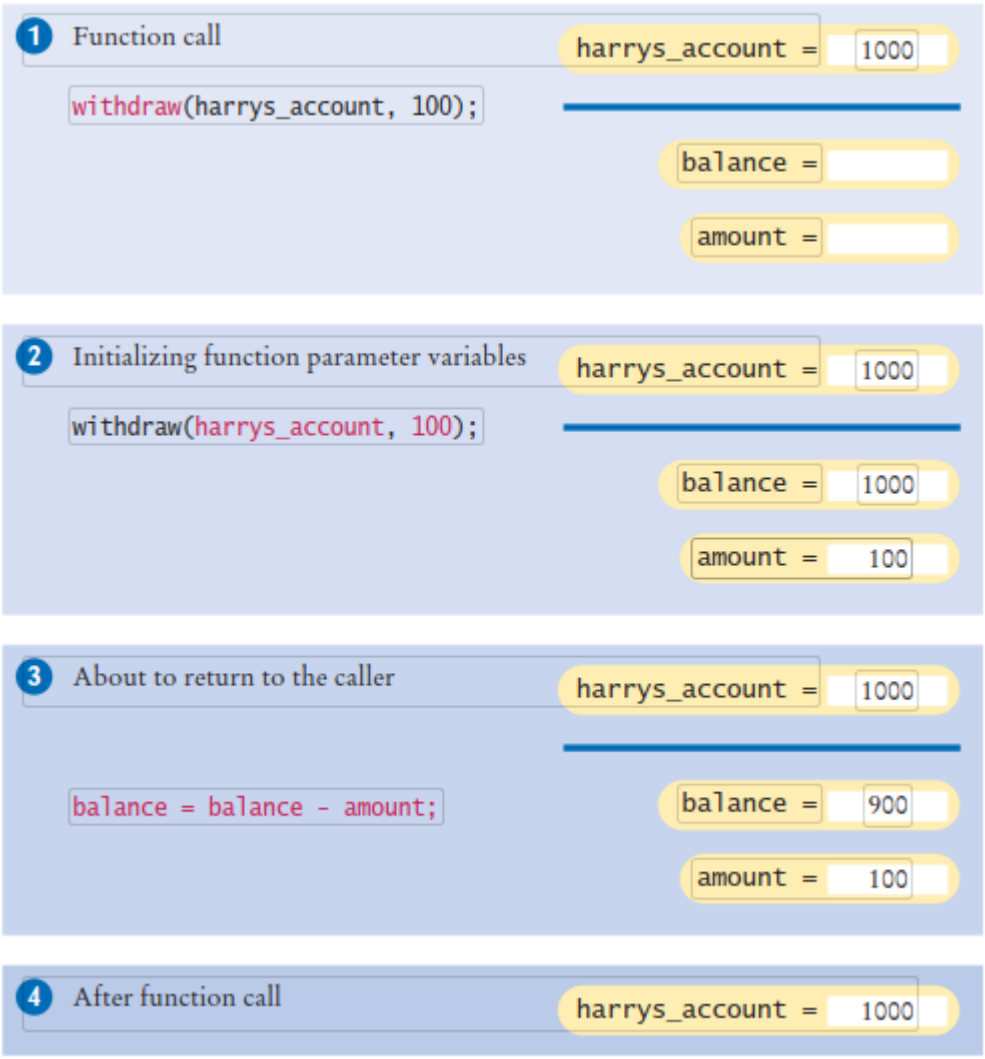
29. Which lines define local variables named x?

30. Which lines are in the scope of the definition of x in line 2?

31. Which variable is changed by the assignment in line 14?

32. This program defines two variables with the same name whose scopes don’t

overlap. What are they?



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

ch05/intname.cpp

```
1 #include <iostream>
2 #include <string>
3
4using namespace std;
5
6 /**
7Turns a digit into its English name.
8@param digit an integer between 1 and 9
9@return the name of digit (“one” ... “nine”)
10 */
11 string digit_name(int digit)
12 {
13if (digit == 1) return"one"; 14if (digit == 2) return"two"; 15if (digit ==
3) return"three";
```

```
16if (digit == 4) return"four"; 17if (digit == 5) return"five"; 18if (digit ==
6) return"six";
19if (digit == 7) return"seven"; 20if (digit == 8) return"eight"; 21if (digit
== 9) return"nine";
22return"";
23 }
24
25 /**
26Turns a number between 10 and 19 into its English name.
27@param number an integer between 10 and 19
28@return the name of the given number (“ten” ... “nineteen”)
29 */
30 string teen_name(int number)
31 {
32if (number == 10) return"ten";
33if (number == 11) return"eleven";
34if (number == 12) return"twelve";
35if (number == 13) return"thirteen";
36if (number == 14)return"fourteen";
37if (number == 15) return"fifteen";
38if (number == 16) return"sixteen";
39if (number == 17) return"seventeen";
40if (number == 18) return"eighteen";
41if (number == 19) return"nineteen";
42return"";
43 }
44
45 /**
46Gives the name of the tens part of a number between 20 and 99.
47@param number an integer between 20 and 99
48@return the name of the tens part of the number (“twenty” ...
“ninety”) 49 */
50 string tens_name(int number)
51 {
52if (number >= 90) return"ninety";
53if (number >= 80) return"eighty";
54if (number >= 70) return"seventy";
55if (number >= 60) return"sixty";
```

```
56if (number >= 50) return"fifty";
57if (number >= 40) return"forty";
58if (number >= 30) return"thirty";
59if (number >= 20) return"twenty";
60return"";
61 }
62
63 /**
64Turns a number into its English name.
65@param number a positive integer < 1,000
66@return the name of the number (e.g. “two hundred seventy four”)
67 */
68 string int_name(int number)
69 {
70int part = number; // The part that still needs to be converted
71string name; // The return value
72
73if (part >= 100)
74 {
75name = digit_name(part / 100) + " hundred"; 76part = part % 100;
77 }
78
79if (part >= 20)
80 {
81name = name + " " + tens_name(part);
82part = part % 10;
83 }
84else if (part >= 10)
85 {
86name = name + " " + teen_name(part);
87part = 0;
88 }
89
90if (part >0)
91 {
92name = name + " " + digit_name(part);
93 }
94
```

```
95return name;
96 }
97
98int main()
99 {
100 cout <<"Please enter a positive integer: "; 101int input;
102 cin >> input;
103 cout << int_name(input) << endl;
104return0;
105 }
```

program run
Please enter a positive integer: 729
seven hundred twenty nine

Nazorat savollari

10. C++da funksiya qanday ishlaydi?
11. funksiyaga kutubxona kerakmi?
12. For operatori funksiyada qanday ishlatiladi?
13. Matematik funksiyalar qanday ishlaydi?
14. Funksiya parametrlar nima?
15. Funksiya qanday chaqiriladi?
16. Funksiya parametrlari orqali nima uzatiladi?
17. If operatorining nechta turi bor?
18. O'zgaruvchilar nima uchun qo'llaniladi?

14-Ma’ruza. Rekursiv funksiyalar

Ma’ruza rejasi:

14.1 Rekursiv jarayon nima
14.2 Rekursiv funksilari
14.3 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:

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. C++ tilida rekursiya matematikadagi rekursiyaga o‘xshash. Buni yuqoridagi misollar uchun tuzilgan funksiyalarda ko‘rish mumkin. Faktorial uchun:

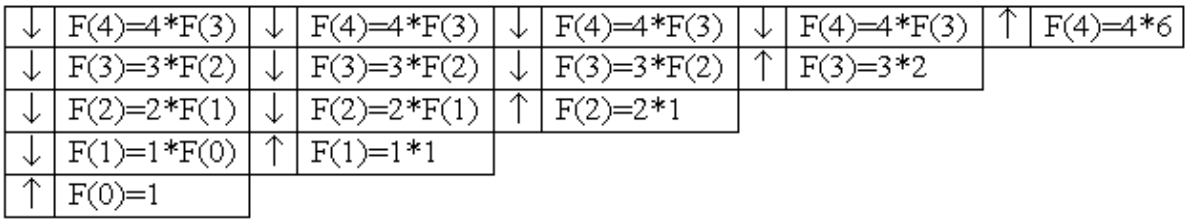
```
long F(int n)
{
 if(!n) return 1;
 else return n*F(n-1);
}
```

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:



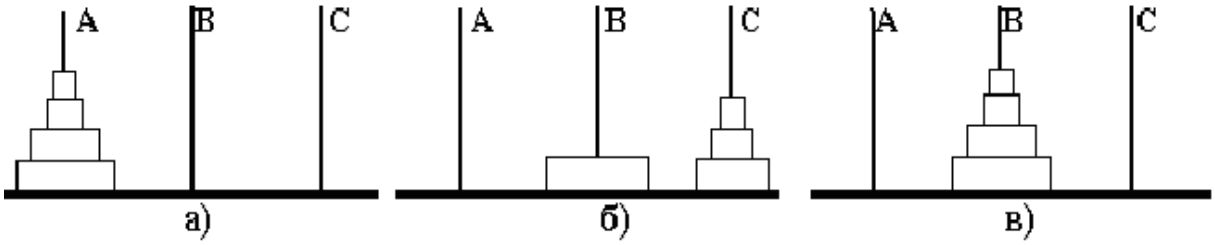
14.1-rasm. 4! hisoblash sxemasi

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. Funksiya qanday chaqiriladi?
9. Funksiya parametrlari orqali nima uzatiladi?
10. If operatorining nechta turi bor?
11. O’zgaruvchilar nima uchun qo’llaniladi?

15-Ma’ruza. Massivlar

Ma’ruza rejasi:

15.1 Bir o’lchovli massivlar

15.2 Massivlarni navlarga ajratish

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.*

15.1. Bir o’lchovli massivlar

Massiv tushunchasi. Massiv bu bir turli nomerlangan ma’lumotlar jamlanmasidir. Massiv indeksli o’zgaruvchi tushunchasiga mos keladi. Massiv ta’riflanganda turi, nomi va indekslar chegarasi ko’rsatiladi. Masalan, type turidagi length ta elementdan iborat a nomli massiv shunday e’lon qilinadi:

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

Bu maxsus a[0], a[1], ..., a[length -1] nomlarga ega bo’lgan type turidagi o’zgaruvchilarning e’lon qilinishiga to’g’ri keladi.

Massivning har bir elementi o’z raqamiga - indeksga ega. Massivning x-nchi elementiga murojaat indekslash operasiyasi yordamida amalga oshiriladi:

```
int x = ...;     //butun sonli indeks
TYPE value = a[x];   //x-nchi elementni o‘qish
a[x] = value;    //x- elementga yozish
```

Indeks sifatida butun tur qiymatini qaytaradigan har qanday ifoda qo’llanishi mumkin: char, short, int, long. C da massiv elementlarining indekslari 0 dan boshlanadi (1 dan emas), length elementdan iborat bo’lgan massivning oxirgi elementining indeksi esa - bu length -1 (length emas) ga teng. Massivning int z[3] shakldagi ta’rifi, int turiga tegishli z[0],z[1],z[2] elementlardan iborat massivni aniqlaydi.

Massiv chegarasidan tashqariga chiqish (ya’ni mavjud bo’lmagan elementni o’qish/yozishga urinish) dastur bajarilishida kutilmagan

natijalarga olib kelishi mumkin. Shuni ta’kidlab o’tish lozimki, bu eng ko’p tarqalgan xatolardan biridir.

Agar massiv inisializasiya qilinganda elementlar chegarasi ko’rsatilgan bo’lsa, ro’yxatdagi elementlar soni bu chegaradan kam bo’lishi mumkin, lekin ortiq bo’lishi mumkin emas.

Misoluchun int a[5] = {2,-2}. Buholda a[0] va a[1] qiymatlarianiqlanganbo’lib, mosholda 2 va –2 gateng. Agarmassivuzunligigaqaragandakamroqelementberilganbo’lsa, qolganelementlar 0 hisoblanadi:

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

Agar nomlangan massivning tavsifida uning o’lchamlari ko’rsatilmagan bo’lsa, kompilyator tomonidan massiv chegarasi avtomatik aniqlanadi:

int a3[] = {1, 2, 3};

Massivda musbat elementlar soni va summasini hisoblash.

```
#include<stdio.h>
int main()
{
int s = 0,k = 0;
int x[] = {-1,2,5,-4,8,9};
for(int i = 0; i<6; i++)
{
if (x[i]< = 0) continue;
k++;
s+ = x[i];
};
printf("%d\n",k);
printf("%d\n",s);
return 0;
};
```

Massivning eng katta, eng kichik elementi va o’rta qiymatini aniqlash:

```
#include<stdio.h>
int main()
{
int i,j,n;
float min,max,s = 0;
```

```
float a,b,d,x[100];
while(1)
{
printf("\n n = ");
scanf("%d",&n);
if ( n>0 && n< = 100 ) break;
printf("\n Hato 0<n<101 bo'lishi kerak");
}
printf("\n elementlar qiymatlarini kiriting:\n");
for (i = 0;i<n;i++)
{
printf("x[%d] = ",i);
scanf("%f",&x[i]);
}
max = x[0];
min = x[0];
for(i = 0;i<n;i++)
{
s+ = x[i];
if (max<x[i]) max = x[i];
if (min>x[i]) min = x[i];
};
s/ = n;
printf("\n max = %f",max);
printf("\n min = %f",min);
printf("\n o'rta qiymat = %f",s);
return 0;
};
```

15.2 Massivlarni navlarga ajratish

Navlarga ajratish - bu berilgan ko’plab ob’ektlarni biron-bir belgilangan tartibda qaytadan guruhlash jarayoni.

Massivlarning navlarga ajratilishi tez bajarilishiga ko’ra farqlanadi. Navlarga ajratishning n*n ta qiyoslashni talab qilgan oddiy usuli va n*log(n) ta qiyoslashni talab qilgan tez usuli mavjud. Oddiy usullar navlarga ajratish tamoyillarini tushuntirishda qulay hisoblanadi, chunki sodda va kalta algoritmlarga ega. Murakkablashtirilgan usullar kamroq sonli operasiyalarni talab qiladi, biroq operasiyalarning o’zi murakkabroq,

shuning uchun uncha katta bo'lmagan massivlar uchun oddiy usullar ko'proq samara beradi.

Oddiy usullar uchta asosiy kategoriyaga bo'linadi:

- oddiy kiritish usuli bilan navlarga ajratish;
- oddiy ajratish usuli bilan navlarga ajratish;
- oddiy almashtirish usuli bilan navlarga ajratish.

Oddiy kiritish usuli bilan navlarga ajratish

Massiv elementlari avvaldan tayyor berilgan va dastlabki ketma-ketliklarga bo'linadi. i = 2 dan boshlab, har bir qadamda dastlabki ketma-ketlikdan i-nchi element chiqarib olinadi hamda tayyor ketma-ketlikning kerakli o'rniga kiritib qo'yiladi. Keyin i bittaga ko'payadi va h.k.

Kerakli joyni izlash jarayonida, ko'proq o'ngdan bitta pozisiyadan tanlab olingan elementni uzatish amalga oshiriladi, ya'ni tanlab olingan element, j: = i-1 dan boshlab, navlarga ajratib bo'lingan qismning navbatdagi elementi bilan qiyoslanadi. Agar tanlab olingan element a[i] dan katta bo'lsa, uni navlarga ajratish qismiga qo'shadilar, aks holda a[j] bitta pozisiyaga suriladi, tanlab olingan elementni esa navlarga ajratilgan ketma-ketlikning navbatdagi elementi bilan qiyoslaydilar. To'g'ri keladigan joyni qidirish jarayoni ikkita turlicha shart bilan tugallanadi:

agar a[j]>a[i] elementi topilgan bo'lsa;

agar tayyor ketma-ketlikning chap uchiga bo'lsa.

```
int i, j, x;
for(i = 1; i < n; i++)
{
x = [i]; // kiritib qo'yishimiz lozim bo'lgan elementni esda saqlab qolamiz
j = i - 1;
while(x<a[j]&&j> = 0)//to'g'ri keladigan joyni qidirish
}
a[j+1] = a[j]; //o'ngga surish
j--;
}
a[j+1] = x; //elementni kiritish
}
```

Oddiy tanlash usuli bilan navlarga ajratish

Massivning minimal elementi tanlanadi hamda massivning birinchi elementi bilan joy almashtiriladi. Keyin jarayon qolgan elementlar bilan takrorlanadi va h.k.

```
int i, min, n_min, j;
for(i = 0; i < n-1; i++)
{
min = a[i]; n_min = i; //minimal qiymatni qidirish
for(j = i + 1; j < n; j++)
if(a[j] < min){min = a[j]; n + min = j;}
a[n_min] = a[i]; //almashtirish
a[i] = min;
}
```

Oddiy almashtirish usuli bilan navlarga ajratish

Elementlar juftlari oxirgisidan boshlab qiyoslanadi va o'rin almashinadi. Natijada massivning eng kichik elementi uning eng chapki elementiga aylanadi. Jarayon massivning qolgan elementlari bilan davom ettiriladi.

```
for(int i = 1; i < n; i++)
for(int j = n - 1; j > = i; j—
if(a[j] < a[j-1])
{int r = a[j]; a[j] = a[j-1]; a[j - 1] = r;}
}
```

Bir o'lchamli massivlarni funksiya parametrlari sifatida uzatish. Massivdan funksiya parametri sifatida foydalanganda, funksiyaning birinchi elementiga ko'rsatkich uzatiladi, ya'ni massiv hamma vaqt adres bo'yicha uzatiladi. Bunda massivdagi elementlarning miqdori haqidagi axborot yo'qotiladi, shuning uchun massivning o'lchamlari haqidagi ma'lumotni alohida parametr sifatida uzatish kerak.

Misol:

Massiv barcha elementlari chiqarilsin:

```
#include <stdio.h>
#include <stdlib.h>
int form(int a[100])
{
```

```
int n;
printf("\nEnter n:");
scanf("%d",&n);
for(int i = 0;i < n; i++)
a[i] = rand()%100;
return n;
}
void print(int a[100], int n)
{
for(int i = 0;i < n; i++)
printf("%d ", a[i]);
printf("\n");
}
int main()
{
int a[100];
int n;
n = form(a);
print(a,n);
return 0;
}
```

Funksiyagamassivboshlanishiuchunko'rsatkichuzatilganitufayli (adresbo'yichauzatish), funksiyatanasiningoperatorlarihisobigamassivo'zgarishimumkin.

Funksiyalarda bir o'lchovli sonli massivlar argument sifatida ishlatilganda ularning chegarasini ko'rsatish shart emas.

Misol. Massivdan barcha juft elementlar chiqarilsin.

```
#include <stdio.h>
#include <stdlib.h>
void form(int a[], int n)
{
for(int i = 0; i < n; i++)
a[i] = rand()%100;
}
void print(int a[],int n)
{
```

```
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
}
int Dell(int a[],int n)
{
int j = 0, i, b[100];
for(i = 0; i < n; i++)
if(a[i]%2! = 0)
{
b[j] = a[i]; j++;
}
for(i = 0; i < j; i++) a[i] = b[i];
return j;
}
int main()
{
int a[100];
int n;
printf("\nEnter n:");
scanf("%d",&n);
form(a, n);
print(a, n);
n = Dell(a, n);
print(a, n);
system("pause");
return 0;
}
```

Funksiyalardabiro'lchovlisonlimassivlarargumentsifatidaishlatilgandaularningchegarasiniko'rsatishshartemas. Misol tariqasida n o'lchovli vektorlar bilan bog'liq funksiyalarni ta'riflashni ko'rib chiqamiz.

Vektorninguzunliginianiqlashfunksiyasi:

```
float mod_vec(int n, float x[])
{
float a = 0;
for (int i = 0; i < n; i++)
```

```
a+ = x[i] * x[i];
return sqrt(double(a));
}
```

Funksiyalardabiro'lchovlimassivlarqaytariluvchiqiymatlarsifatida ham kelishimumkin. Misol uchun ikki vektor summasini hisoblovchi funksiya prosedurani ko'ramiz:

```
void sum_vec(int n, float a, float b, float c)
{
for(int i = 0;i < n; i++,c[i] = a[i] + b[i]);
}
```

Bu funksiyaga quyidagicha murojaat qilish mumkin:

```
float a[] = {1, -1.5, -2};
b[] = {-5.2, 1.3, -4},c[3];
sum_vec(3, a, b, c);
```

Polinom. Polinom qiymatini hisoblash funksiyasi poly deb nomlanadi.

Prototipi:

```
double poly(double x, int degree, double coeffs[]);
```

Algebraik polinom koeffisientlari coeffs[0], coeffs[1], ..., coeffs[degree] massiv elementlarida beriladi.

Misol:

```
 #include <stdio.h>
 #include <math.h>
/* polynomial: x**3 - 2x**2 + 5x - 1 */
 int main(void)
{
 double array[] = { -1.0, 5.0, -2.0, 1.0};
 double result;
 result = poly(2.0, 3, array);
 printf("The polynomial: x**3 - 2.0x**2 + 5x - 1 at 2.0 is %lf\n", result);
 return 0;
 }
```

16-Ma’ruza. Massivlar uchun umumiy algoritmlar

Ma’ruza rejasi:

16.1 Umumiy massiv algoritmlari

16.1.1To’ldirish

16.1.2 Nusxa ko’chirish

16.1.3 Yig’indi va O’rtacha Qiymat

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.*

16.1 Umumiy massiv algoritmlari

Quyidagi bo’limlarda biz, qiymatlar ketma- ketligini qayta ishlash uchun ba’zi eng keng tarqalgan algoritmlarni muhokama qilamiz.Biz algoritmlarni shunday taqdim etamizki, siz ularni to’liq va qisman to’ldirilagan massivlar hamda (6.7 bo’limda tanishtiradigan)vektorlar yordamida foydalanishingiz mumkin.Qiymatlar miqdori (size of values) ifodasini ishlatganimizda, siz uni massivda elementlar miqdorini anglatuvchi doimiy yoki o’zgaruvchan bilan almashtirishingiz kerak.

(yoki values.size() if values is a vector ifodasi.)

16.1.1To’ldirish

Bu sikl massivni nollar bilan to’ldiradi:

```
for (int i = 0; i < size of values; i++)
{
  values[i] = 0;
}
```

So’ngra, keling kvadratlar massivini 0, 1, 4, 9, 16, va h.z sonlar bilanto’ldiramiz. E’tibor bering, 0 indeksli element 02 ni , 1 indeksli element 12va h.z ni o’z ichiga oladi.

```
for (int i = 0; i < size of squares; i++)
{
  squares[i] = i * i;
}
```

16.1.2 Nusxa ko’chirish

Ikkita massivni ko’rib chiqamiz:

```
int squares[5] = { 0, 1, 4, 9, 16 };
```

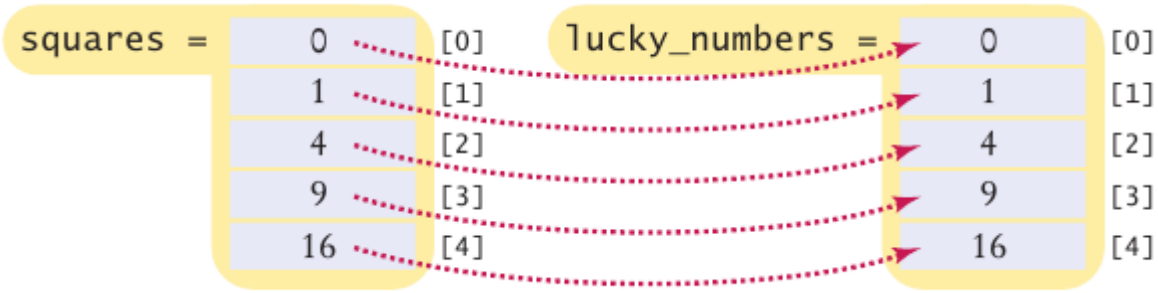
int lucky_numbers[5];

Hozir aytaylik, siz birinchi massivdagi barcha qiymatlarni ikkinchisigako'chirmoqchisiz. Quyidagi ifoda xato hisoblanadi:

lucky_numbers = squares; // Error

C++da siz bir massivni ikkinchisiga o'tkaza olmaysiz.Uning o'rniga siz barchaelementlarni ko'chirish uchun sikldan foydalanishingiz kerak:

```
for (int i = 0; i < 5; i++)
{
  lucky_numbers[i] = squares[i];
}
```



16.1 shakl massivndan nusxa olishuchun elementlarniko'chirish

16.1.3 Yig'indi va O'rtacha Qiymat

Siz allaqachon bu algoritm bilan 4.7.1. bo'limda duch kelgansiz.Quyidamassivning barcha elementlarining yig'indisini hisoblash kodi berilgan.

```
double total = 0;
for (int i = 0; i < size of values; i++)
{
  total = total + values[i];
}
```

O'rtacha qiymatni topish uchun elementlar miqdorinibo'ling:

double average = total / size of values;

miqdor nol emasligini tekshirishni unutmang.

16.1.4 Maksimum va minimum

O'zida eng katta element uchun o'zgaruvchini saqlovchi va massivlarni joriy etishda uchratgan, 4.7.4 - bo'limdagi algoritmni qo'llang:

```
double largest = values[0];
for (int i = 1; i < size of values; i++)
{
  if (values[i] > largest)
```

```
  {
  largest = values[i];
  }
 }
```

Biz eng katta qiymatni [0] qiymati bilan inisializasiyalashtirganligimiz tufayli, sikl 1 dan boshlanishiga e'tibor qarating.Eng kichik qiymatni hisoblash uchun solishtirishni orqaga qaytaring. Bu algoritmlar massiv kamida bitta elementni o'z ichiga olishni talab qiladi.

16.1.5 Element ajratuvchilari razdeliteli

Siz element to'plamlarini namoyish etganingizda, siz odatda ularni vergul yoki quyidagicha vertikal chiziqlar bilan bo'lishni hohlaysiz:

1 | 4 | 9 | 16 | 25

E'tibor bering, ajratuvchi sonlarga qaraganda bittaga kam. dastlabki(0 indeksli) dan tashqari har qaysi elementdan oldin ajratuvchini yozing.

```
for (int i = 0; i < size of values; i++)
{
  if (i > 0)
  {
  cout << " | ";
  }
  cout << values[i];
}
```

16.1.6 Bir chiziqli qidiruv

Uni o'rnini almashtirish yoki olib tashlash uchunn siz tez- tez elemento'rnini topishingiz kerak bo'ladi. Massivning oxiriga kelmaguncha yokio'xshashlik topmaguningizcha, barcha elementlarni ko'rib chiqing. Quyida biz100ga teng bo'lgan birinchi element o'rnini qidiramiz.

```
bool found = false;
while (pos < size of values && !found)
{
  if (values[pos] == 100)
  {
  found = true;
  }
  else
```

```
{
poC++;
}
}
```

Agar topilma to’g’ri bo’lsa, joy birinchi moslikning o’rni hisoblanadi.

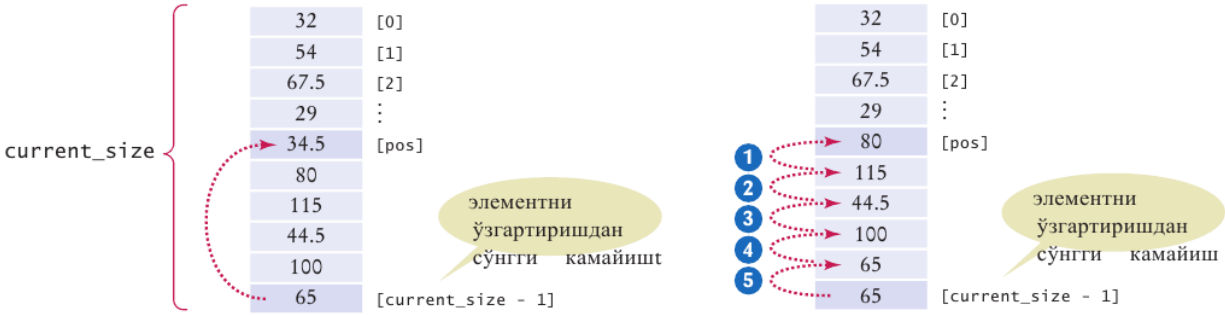
6.2.7 Elementni olib tashlash

Joriy miqdori o’zgaruvchan joriy miqdorda saqlanovchi qisman to’ldirilgan massiv miqdorini ko’rib chiqamiz. Deylik, siz pos indeksli elementni qiymatdanolib tashlamoqchisiz.Agar elementlar hyech qanday aniq tartibda bo’lmasa, buvazifani amalag oshirish yengil kechadi.Shunchaki, oxirgi elementdan olibtashlangan elementni qaytadan yozing.So’ng, miqdorni kuzatuvchi o’zgaruvchinikamaytiring. (6 - shaklga qarang).

```
values[pos] = values[current_size - 1];
current_size--;
```

Agar elementlar tartibi muhim bo’lsa vaziyat yanada murakkablashadi. Bundayholda siz barcha elementlarni keyingi indeksi pastroq elementgaalmashtirishingiz kerak.So’ng massivni miqdorini saqlagan holda o’zgaruvchinikamaytirishingiz kerak. (7 - shaklga qarang).

```
for (int i = pos + 1; i < current_size; i++)
{
  values[i - 1] = values[i];
}
current_size--;
```



6-7- shakl. Tartibsiz massivda elementni olib tashlash va Tartibli massivda elementni olib tashlash

6.2.8 Element kiritish

Agar element tartibi muhim bo’lmasa, siz oddiygina miqdorni kuzatuvchio’zgaruvchini ko’paytirib yangi elementni oxiriga kiritishingiz mumkin.

Qisman to’ldirilganmassiv uchun:

```
if (current_size < CAPACITY)
{
  current_size++;
  values[current_size - 1] = new_element;
}
```

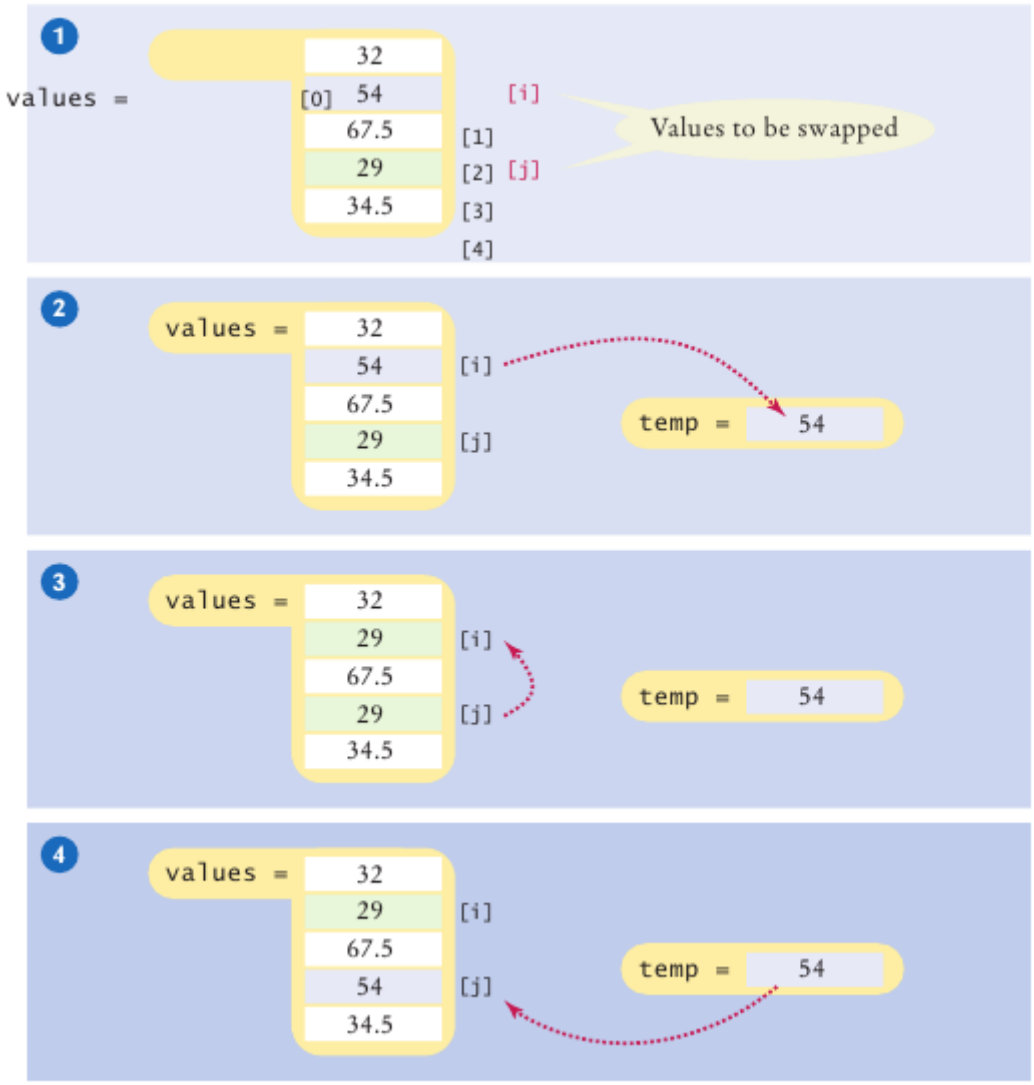
Ketma - ketlikning o’rtasida muayan holatda element kiritish ko’proq ishni talabqiladi. Birinchidan, joriy miqdorni o’zida ushlab turgan o’zgaruvchiniko’paytiring.Keyin, barcha elementlarni qo’shish o’rnidan yuqoriroq indeksgako’chiring. Nihoyat, yangi elementni joylashtiring.Quyida qisman to’ldirilgan massiv uchun kod berilgan:

```
if (current_size < CAPACITY)
{
  current_size++;
  for (int i = current_size - 1; i > pos; i--)
  {
  values[i] = values[i - 1];
  }
  values[pos] = new_element;
}
```

Harakatlarning tartibiga e’tibor qarating: Elementni olib tashlaganingizda,birinchi bo’lib keyingi elementni pastroq indeksga o’zgartiring. Keyin esa shu ketma- ketlikda massivni oxiriga yetib borguningizgacha davom ettiring.Yangi elementkiritishda siz massivni oxiridan boshlaysiz.Shu elementni yuqoriroq indeksgao’zgartiringg vaa keyinn undann oldingisigaa vaa xakozoo nihoyatt qo’shishsh joyigakelguningizgachaa (9- shaklga qarang).

6.2.9 Elementtlar n ailmashtirish

Siz massiv elementlarini tez -tezalmashtirishingiz kerak bo’ladi.Misol uchun, 263 - sahifadagi 6.2 - maxsus mavzudagi saralash algoritmlari elementlarni almashtirish orqali takroran massivni saralaydi.



Massiv qiymatlarining i va j o'rnidagielementlarni almashtirish vazifasini ko'rib chiqamiz.Biz [i] qiymatlarini [j] qiymatlariga o'rnatishnihoxlar edik, lekin bu ayni paytda [i]qiymatlaridasaqlangan qiymatni qaytadan yozganligi sababli bizbirinchisini saqlashni istaymiz.

double temp = values[i];

values[i] = values[j];

Endi biz [j]qiymatlarini saqlangan qiymatga o'rniatishimiz mumkin.

values[j] = temp;

10- shaklda jarayonni ko'rishingiz mumkin.

16.1.10 Kiritishni o'qish

Bilsangiz edi foydalanuvchi qancha kiritish qiymatlari bilan ta'minlashikerak, shunchaki ularni bitta massivga joylashtirish uchun.

double values[NUMBER_OF_INPUTS];

for (i = 0; i < NUMBER_OF_INPUTS; i++)

```
{
  cin >> values[i];
```

17-Ma’ruza. Massivlar yordamida obyektlarni tadqiq qilish

Ma’ruza rejasi:

17.1 Dinamik massivlar bilan ishlash

17.2 Funksiya va massivlar

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.*

Statikmassivlarningkamchiliklarishundaki, ularningo‘lchamlarioldindanma’lumbo‘lishikerak, bundantashqaribuo‘lchamlarberilganlargaajratilganxotirasegmentiningo‘lchamibilanchegaralangan. Ikkinchitomondan, etarlichakattao‘lchamdagimassive’lonqilib, konkretmasalaechilishidaajratilganxotirato‘liqishlatilmasligimumkin. Bukamchiliklardinamikmassivlar-danfoydalanishorqalibartarafetiladi, chunkiularprogrammaishlashijarayonidakerakbo‘lgano‘lchamdagimassivlarniyaratishvazaruratqolmagandayo‘qotishimkoniyatiniberadi.

Dinamikmassivlargaxotiraajratishuchun malloc(), calloc() funksiyalaridanyoki new operatoridanfoydalanishmumkin. Dinamikob’ektgaajratilganxotiranibo‘shatishuchun free() funksiyasiyoki delete operatoriishlatiladi.

YUqoridaqaydqilinganfunksiyalar «alloc.h» kutubxonasidajoylashgan.

malloc() funksiyasiningsintaksisi

void * malloc(size_t size);

ko‘rinishidabo‘lib, uxotiraninguyumqismidan size bayto‘lchamidagiuzluksizsohaniajratadi. Agarxotiraajratishmuvaffaqiyatlibo‘lsa, malloc() funksiyasiajratilgansohaningboshlanishadresiniqaytaradi. Talabqilinganxotiraniajratishmuvaffaqiyatsizbo‘lsa, funksiya NULL qiymatiniqaytaradi.

Sintaksisdanko‘rinibturibdiki, funksiya void turidagiqiymatqaytaradi. Amalda esa konkret turdagi ob’ekt uchun xotira ajratish zarur bo‘ladi.Buninguchunvoid turinikonkretturgakeltirishtexnologiyasidanfoydalaniladi. Masalan,

butunturdagiuzunligi 3 gatengmassivgajoyajratishniquyidagichaamalgaoshirishmumkin:

int * pInt=(int*)malloc(3*sizeof(int));

calloc() funksiyasi malloc() funksiyasidanfarqliravishdamassivuchunjoyajratishdantashqarimassivelementlarini 0 qiymatibilaninitsializatsiyaqiladi. Bufunksiyasintaksisi

void * calloc(size_t num, size_t size);

ko‘rinishdabo‘lib, num parametriajratilgansohadanechtaelementborligini, size harbirelemento‘lchaminibildiradi.

free() xotiranibo‘shatishfunksiyasio‘chiriladiganxotirabo‘lagigako‘rsatkichbo‘lganyagonaparametrgaegabo‘ladi:

void free(void * block);

free() funksiyasiparametrining void turidabo‘lishiixtiyoriyturdagixotirabo‘laginio‘chirishimkoniniberadi.

Quyidagiprogrammada 10 tabutunsondaniboratdinamikmassivyaratish, ungaqiymatberishvao‘chirishamallaribajarilgan.

```
#include <iostream.h>
#include <alloc.h>
int main()
{
 int * pVector;
 if ((pVector=(int*)malloc(10*sizeof(int)))==NULL)
 {
  cout<<”Xotira etarli emas!!!”;
  return 1;
 }
 // ajratilganxotirasohasinito‘ldirish
 for(int i=0;i<10;i++) *(pVector+i)=i;
 // vektorelementlarinichopetish
 for(int i=0; i<10; i++) cout<<*(pVector+i)<<endl;
 // ajratilganxotirabo‘laginiqaytarish (o‘chirish)
 free(pVector);
 return 0;
}
```

Keyingiprogrammada n×n o‘lchamlihaqiqiysonlarmassivi-ningboshdiagonalidanyuqoridajoylashganelementlaryig‘indi-sinihisoblashmasalasiechilgan.

```
#include <iostream.h>
#include <alloc.h>
int main()
{
 int n;
 float * pMatr, s=0;
 cout<<”A(n,n): n=”;
 cin>>n;
 if((pMatr=(float*)malloc(n*n*sizeof(float)))==NULL)
 {
  cout<<”Xotira etarli emas!!!”;
  return 1;
 }
 for(int i=0;i<n;i++)
  for(int j=0;j<n;j++)cin>>*(pMatr+i*n+j);
 for(int i=0;i<n;i++)
  for(int j=i+1;j<n;j++)s+=*(pMatr+i*n+j);
 cout<<”Matritsa bosh diagonalidan yuqoridagi ”;
 cout<<”elementlar yig`indisi S=”<<s<<endl;
 return 0;
}
```

new operatoriyordamida,massivgaxotiraajratishdaob’ektturidankeyinkvadratqavsichidaob’ektlarsoniko‘rsatiladi.Masalan, butunturdagi 10 tasondaniboratmassivgajoyajratishuchun

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

ifodasiyozilishikerak. Bungaqarama-qarshiravishda, buusuldaajratilganxotiranibo‘shatishuchun

```
      delete [] pVector;
```

ko‘rsatmasiniberishkerakbo‘ladi.

Ikkio‘lchamlidinamikmassivnitashkilqilishuchun

```
int **a;
```

ko‘rinishidagi «ko‘rsatkichgako‘rsatkich» ishlatiladi.

Boshdamassiv satrlarisonigaqarabko‘rsatkichlarmassivigadinamikxotiradan joy ajratishkerak:

a=new int *[m] // buerdammassiv satrlarisoni

Keyin, harbirsatruchuntakrorlashoperatoriyordamidaxotiraajratishvaularningboshlang‘ichadreslarini a massivelementlarigajoylashtirishzarurbo‘ladi:

for(int i=0;i<m;i++)a[i]=new int[n];//n ustunlarsoni

SHuniqaydetishkerakki, dinamikmassivningharbirsatrixotiraningturlijoylaridajoylashishimumkin (7.1 va 7.3-rasmlar).

Ikkio‘lchamlimassivnio‘chirishdaoldinmassivningharbirelementi (satri), so‘ngramassivningo‘ziyo‘qotiladi:

```
for(i=0;i<m;i++) delete[]a[i];
delete[]a;
```

Matritsanivektorgako‘paytirishmasalasiuchundinamikmassivlardanfoydalanishgamisol:

```
void main ()
{
 int n,m;
 int i,j; float s;
 cout<<"\n n="; cin>>n; // matritsasatrlarisoni
 cout<<"\n m="; cin>>m; // matritsaustunlarisoni
 float *b=new float[m];
 float *c=new float[n];
 // ko‘rsatkichlarmassivigaxotiraajratish
 float **a=new float *[n] ;
 for(i=0;i<n;i++)            // harbirsatruchun
 a[i]=new float[m];       //dinamikxotiraajratish
 for(j=0;j<m;j++)cin>>b[j];
for(i=0;i<n;i++)
for(j=0;j<m;j++)cin>>a[i][j];
 for(i=0;i<n;i++)
 {
  for(j=0,s=0;j<m;j++)s+=a[i,j]*b[j];
  c[i]=s;
 }
 for(i=0;i<n;i++)cout<<"\t c["<<i<<"]="<<c[i];
```

```
 delete[]b;
 delete[]c;
 for (i=0;i<n;i++) delete[]a[i];
 delete[]a;
 return;
}
```

Funksiyavamassivlar

Funksiyalarmassivniparametrsifatidaishlatishivaunifunksiyaningnatijasisifatidaqaytarishimumkin.

Agarmassivparametrorqalifunksiyagauzatilsa, elementlarsoninianiqlashmuammositug‘iladi, chunkimassivnomidanuninguzunliginianiqlashningilojiyo‘q. Ayrimhollarda, masalan, belgilarmassivisifatidaaniqlangansatr (ASCIIZ satrlar) bilanishlagandamassivuzunliginianiqlashmumkin, chunkisatrlar ‘\0’ belgisibilantugaydi.

Misoluchun:

```
#include <iostream.h>
int len(char s[])//massivniparametrsifatidaishlatish
{
int m=0;
while(s[m++]);
 return m-1;
}
void main ()
{
 char z[]=”Ushbu satr uzunligi = ”;
 cout<<z<<len(z);
}
```

Funksiyaparametrisatrbo‘lmaganhollardafiksirlanganuzunlikdagimassivlarishlatiladi. Agarturliuzunlikdagimassivlarniuzatishzarurbo‘lsa, massivo‘lchamlariniparametrsifatidauzatishmumkinyokibumaqsaddaglobalo‘zgaruvchidanfoydalanishgato‘g‘rikeladi.

Misol:

```
#include <iostream.h>
float sum(int n,float *x) //buikkinchiusul
{
 float s=0;
```

```
 for (int i=0;i<n;i++)s+=x[i];
 return s;
}
void main()
{
 float E[]={1.2,2.0,3.0,4.5,-4.0};
 cout<<sum(5,E);
}
```

Massivnomiko‘rsatkichbo‘lganligisabablimassivelement-larinifunksiyadao‘zgartirishmumkinvabuo‘zgartirishlarfunk-siyadanchiqqandankeyinhamsaqlanibqoladi.

```
#include <iostream.h>
void vector_01(int n,int*x,int * y) //buikkinchiusul
{
 for (int i=0;i<n;i++)
  y[i]=x[i]>0?1:0;
}
void main()
{
 int a[]=(1,2,-4,3,-5,0,4};
 int c[7];
 vector_01(7,a,c);
 for(int i=0;i<7;i++) cout<<’\t’<<c[i];
}
```

Masala. Butunturdagivaelementlarikamaymaydiganholdatartiblanganbiro‘lchamliikkitamassivlarniyagonamassivga,tartiblanishsaqlanganholdabirlashtirishamalgaoshirilsin.

Programmamatni:

```
#include <iostream.h>
   \\butunturdagimassivgako‘rsatkichqaytaradigan
       \\funksiya
int * massiv_ulash(int,int*,int,int*);
 void main()
{
 int c[]={-1,2,5,10},d[]={1,7,8};
 int * h;
 h=massiv_ulash(5,c,3,d);
 for(int i=0;i<8;i++) cout<<’\t’<<h[i];
```

```
 delete[]h;
}
int * massiv_ulash(intn,int *a ,int m,int *b);
{
 int * x=new int[n+m];
 int ia=0,ib=0,ix=0;
 while (ia<n && ib<m)
  a[ia]>b[ib]?x[ix++]=b[ib++]:x[ix++]=a[ia++];
 while(ib<m)x[ix++]=b[ib++];
 while(ia<n)x[ix++]=a[ia++];
 return x;
}
```

Ko‘po‘lchamlimassivlarbilanishlashma’lumbirmurakkablikkaega, chunkimassivlarxotiradajoylashtartibiturlivariantdabo‘lishimumkin. Masalan, funksiyaparametrlarro‘yxatida $n \times n$ o‘lchamdagihaqiqiyturdagi x[n][n] massivgamoskeluvchiparametrni

```
float sum(float x[n][n])
```

ko‘rinishdayozibbo‘lmaydi. Muammoechimi - bumassivo‘lchaminiparametrsifatidauzatishvafunksiyasarlavhasiniquyidagichayozishkerak:

```
float sum(int n,float x[][]);
```

Ko‘po‘lchamlimassivlarniparametrsifatidaishlatishdabirnechtausullardan foydalanish mumkin.

1-usul. Massivningikkinchio‘lchaminio‘zgarmasifoda (son) bilanko‘rsatish:

```
float sum(int n,float x[][10])
{float s=0.0;
 for(int i=0;i<n;i++)
 for(int j=0;j<n;j++)
 s+=x[i][j];
 return s;}
```

2-usul. Ikkio‘lchamlimassivko‘rsatkichlarmassiviko‘rinishidaaniqlanganholatlaruchunko‘rsatkichlarmassivini (matritsasatrlaradreslarini) berishorqali:

```
float sum(int n,float *p[])
{
 float s=0.0;
 for(int i=0;i<n;i++)
```

```
 for(int j=0;j<n;j++)
 s+=p[i][j];\\"*p[i][j]" emas,chunkimassivgamurojat
 return s;
}
        void main()
        {
     float x[][4]={{11,-12,13,14},{21,22,23,24},
      {31,32,33,34},{41,42,43,44}};
 float *ptr[4];
 for(int i=0;i<4;i++) ptr[i]=(float *)&x[i];
 cout<<sum(4,ptr)<<endl;
}
```

3-usul. Ko‘rsatkichlargako‘rsatkichko‘rinishidaaniqlangandinamikmassivlarn iishlatishbilan:

```
float sum(int n,float **x)
{
 float s=0.0;
 for(int i=0;i<n;i++)for(int j=0;j<n;j++)s+=x[i][j];   return s;
}
   void main()
   {
    float **ptr;
    int n;
    cin>>n;
    ptr=new float *[n];
    for(int i=0;i<n;i++)
    {
     ptr[i]=new float [n];
     for(int j=0;j<n;j++)
      ptr[i][j]=(float)((i+1)*10+j);
    }
    cout<<sum(n,ptr);
    for(int i=0; i<n;i++) delete ptr[i];
    delete[]ptr;
   }
```

Navbatdagiprogrammadafunksiyatomonidannatijasifatidaikkio‘lcha mlimassivniqaytarishigamisolkeltirilgan.

Massivelementlarningqiymatlaritasodifiysonlardantashkiltopadi. Tasoddifiysonlar «math.h» kutubxonasidagi random() funksiyayordamidahosilqilinadi:

```
#include <iostream.h>
#include <math.h>
int **rmatr(int n,int m)
{
int ** ptr;
          ptr=new int *[n];
          for(int i=0;i<n;i++)
          {
ptr[i]=new int[m];
for(int j=0;j<m;j++) ptr[i][j]=random(100);
          }
          return ptr;
}
int sum(int n,int m,int **ix)
{
 float s=0;
for(int i=0;i<n;i++)
for(int j=0;j<m;j++)s+=ix[i][j];
return s;
}
void main()
{
 int n,m;
          cin>>n>>m;
          int **matr;
         randomize();
          matr=rmatr(n,m);
          for(int i=0;i<n;i++)
          {
cout<<endl<<i<<" - satr:"
  for (int j=0;j<m;j++) cout<<'\t'<<matr[i][j];
          }
          cout<<endl<<"Summa="<<sum(n,m,matr);
          for(int i=0;i<n;i++) delete matr[i];
delete[]matr;
```

```
}
```

ch06/medals.cpp

```
#include <iostream>
#include
<iomanip>yoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoy oyoyoyoyoyoy
oyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyo
yoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyoyo
#include <string>

using namespace std;

const int COLUMNS = 3;

/**
j a d v a l d a g i  u m u m i y  q a t o r l a r n i  h i s o b l a y d i .
@param table uch ustunli jadval
@param row umulashtirilishi kerak bo'lgan qator
@return berilgan qatordagi elementlar majmui
*/
double row_total(int table[][COLUMNS], int row)
{
int total = 0;
for (int j = 0; j < COLUMNS; j++)
{
total = total + table[row][j];
}
return total;
}

int main()
{
const int COUNTRIES = 7;
const int MEDALS = 3;

string countries[] =
{
"Canada",
```

```
33"China",
34"Germany",
35"Korea",
36"Japan",
37"Russia",
38"United States"
39};
40
41int counts[COUNTRIES][MEDALS] =
42 { 43{ 1, 0, 1 }, 44{ 1, 1, 0 }, 45{ 0, 0, 1 }, 46{ 1, 0, 0 }, 47{ 0, 1, 1 }, 48{ 0, 1,
1 }, 49{ 1, 1, 0 }
50 };
51
52 cout <<"Country Gold Silver Bronze Total"<< endl; 53
54// Print countries, counts, and row totals
55for (int i = 0; i < COUNTRIES; i++)
56 {
57cout << setw(15) << countries[i];
58// Process the ith row
59for (int j = 0; j < MEDALS; j++)
60{
61cout << setw(8) << counts[i][j];
62}
63int total = row_total(counts, i);
64cout << setw(8) << total << endl;
65 }
66
67return0;
68 }
```

Natija:

```
M a m l a k a t  O l t i n  K u m u s h  B r o n z a  U m u m i y
Kanada1012
Xitoy1102
Germaniya0011
Koreya1001
YAponiya0112
Rossiya0112 Qo‘shma SHtatlar 1102
```

Nazorat savollari

1. C++da Massiv qanday ishlaydi?
2. Massivga kutubxona kerakmi?
3. Rekursiv funksiya nima?
4. Massiv elementlarini bilan massiv indexlarini farqi nimada?
5. For operatori funksiyada qanday ishlatiladi?
6. Matematik funksiyalar qanday ishlaydi?
7. Massiv elementlarini Funksiya parametrlarida uzatish nima uchun ishlatiladi?
8. Massivlarni qabday turlari mavjud?
9. Funksiya parametrlari orqali nima uzatiladi?

18-Ma’ruza. Vektorlar va ko’p o’lchovli massivlar bilan ishlash

Ma’ruza rejasi:

17.1 Vektor tushunchasi

17.2 Ko’p o’lchovli massivlar

17.3 Ko‘p o‘lchamli statik massivlar

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.*

17.1 Vektor tushunchasi

Foydalanuvchi kiritmasidan qiymatlarni yig‘uvchi dasturni yozganingizda nechta qiymatlar bo‘lishini bilmaysiz. Afsuski, *dastur tuzilganda* jadval hajmi aniq bo‘lishi kerak.

Bo‘limda bu muammoni qisman to‘ldirilgan jadvallar qanday yo‘naltirishni ko‘rdingiz. biz quyidagi bo‘limlarda muhokama qiladigan bu vektor tuzilishi qulayroq echimni taklif etadi. Vektor qiymatlar ketma ketligini xuddi jadval qilgandek yig‘adi lekin uning hajmi o‘zgarishi mumkin.

Vektorni aniqlaganingizda burchak kronshteyndagi elementlar turini aniqlashtirasiz. Quyidagicha:

vector<double> values;

Boshlang‘ich hajmni erkin aniqlashingiz mumkin. Masalan, Bu erda boshlang‘ich

hajmi 10 ga teng bo‘lgan vektorning ta’rifi:

vector<double> values(10);

agar vektorni boshlang‘ich hajmsiz aniqlasangiz uning hajmi 0 ga teng.

Jadvalning 0 hajmini aniqlashda hech qanday nuqta bo‘lmaganda boshlang‘ich hajmi 0 ga teng vektorlarga ega bo‘lish foydali va keyin ular keraklicha o‘stiriladi. Dasturingizda vektorlardan foydalanish uchun siz vektor sarlavhasini ham kiritishingiz zarur

defining Vectors

vector<int> numbers(10);10 butun vektor

vector<string> names(3);3 trosli vektor

vector<double> values;0 hajmli vektor

vector<double> values();Xato**:** Vektorni aniqlamadi.

Agar qo‘shimcha elementlar kerak bo‘lsa elementni vektorning oxiriga qo‘shish uchun push_back funksiyasini qo‘llaysiz. bunda uning hajmi 1 ga ortadi. Push_back funksiyas i quyidagi nuqtali notatsiya bilan chaqirishingiz shart bo‘lgan a’zo funksiyasidir:

values.push_back(37.5);

Bu chaqiruvdan so‘ng 14 shakldagi vektor qiymati 3 hajmga teng va qiymatlar

values[2] qiymat 37.5 ga teng.

Bo‘sh vektor bilan boshlash va push_back funksiyasi bilan uni to‘ldirish ommalashgan. Masalan,

vector<double> values; // Dastlabki bo‘sh

values.push_back(32); // endi qiymatla 1 hajmga va element esa 32 hajmga ega values.push_back(54); // endi qiymatla 2 hajmga va element esa 32,54 hajmga ega values.push_back(37.5); // endi qiymatla 3 hajmga va element esa 32,54,37.5 hajmga ega

push_back a’zo funksiyasi uchun boshqa bir foydalanish bu vektorni kiritma qiymatlari bilan to‘ldirish.

```
vector<double> values; // Dastlabki bo‘sh
double input;
while (cin >> input)
{
 values.push_back(input);
}
```

Bu kiritmassikl 6.2.10 bo‘limdagidan osonroq va oddiyroq ekanini qayd eting. Boshqa a’zo funksiyasi, pop_back, vektorning oxirgi elementini olib tashlash, uning xajmini bittaga kichiklashtirish. (15 shaklga qarang):

values.pop_back();

Vektorlar va funksiyalar

Siz boshqa qiymatlar kabi vektorlarni funksiya parametri sifatida ishlatishingiz mumkin. Masalan, quyida funksiya flouting member nuqtasi sonlari vektorining umumiy sonini hisoblaydi:

```
double sum(vector<double> values)
{
 double total = 0;
 for (int i = 0; i < values.size(); i++)
 {
```

```
total = total + values[i];
 }
 return total;
}
```

Bu funksiya vektor elementlarini aylanib o‘tadi, lekin ularni o‘zgartirmaydi.

```
void multiply(vector<double>& values, double factor) // Note the &
{
 for (int i = 0; i < values.size(); i++)
 {
values[i] = values[i] * factor;
 }
}
```

Ba’zi programmistlar o‘zgartirilmaydigan vektor parametrlari uchun turg‘un yo‘nalishdan foydalanadilar. (Maxsus mavzu 5.2) masalan:

```
double sum(const vector<double>& values) // const &added for efficiency
```

funksiya vektorni qaytarishi mumkin. YAna vektorlar qolgan qiymatlardan

boshqacha emas. funksiyada natijani o‘rnating va uni qaytaring. bu misolda, squares funksiyasi vektorni 0^2 up to $(n-1)^2$ qaytaradi:

```
vector<int> squares(int n)
{
 vector<int> result;
 for (int i = 0; i < n; i++)
 {
result.push_back(i * i);
 }
 return result;
}
```

18.3Ko‘p o‘lchamli statik massivlar

C++ tilida massivlar elementining turiga cheklovlar qo‘yil-maydi, lekin bu turlar chekli o‘lchamdagi ob’ektlarning turi bo‘lishi kerak. CHunki kompilyator massivning xotiradan qancha joy (bayt) egallashini hisoblay olishi kerak. Xususan, massiv komponentasi massiv bo‘lishi mumkin

(«vektorlar-vektori»), natijada *matritsa* deb nomlanuvchi ikki o‘lchamli massiv hosil bo‘ladi.

Agar matritsaning elementi ham vektor bo‘lsa, uch o‘lchamli massivlar - *kub* hosil bo‘ladi. SHu yo‘l bilan echilayotgan masalaga bog‘liq ravishda ixtiyoriy o‘lchamdagi massivlarni yaratish mumkin.

Ikki o‘lchamli massivning sintaksisi quyidagi ko‘rinishda bo‘ladi:

<tur><nom> [<uzunlik >] [<uzunlik>]

Masalan, 10×20 o‘lchamli haqiqiy sonlar massivining e’loni:

float a[10][20];

E’lon qilingan A matritsani ko‘rinishi 7.2-rasmda keltirilgan.

$$
\begin{array}{cc}
 & \qquad\qquad\qquad\qquad\qquad\quad \mathbf{j} \\
 & a_0: \quad (a_{0\,0}, \quad a_{0\,2} \quad \ldots \quad \ldots \quad a_{0\,18}, \quad a_{0\,19}), \\
 & a_1: \quad (a_{1\,0}, \quad a_{1\,1}, \quad \ldots \quad \ldots \quad a_{1\,18}, \quad a_{1\,19}), \\
 & \ldots \\
\mathbf{i} & a_i: \quad (\ldots, \quad \ldots, \quad \ldots \quad \mathbf{a_{i\,j}} \quad \ldots \quad \ldots, \quad \ldots), \\
 & \ldots \\
 & a_9: \quad (a_{9\,0}, \quad a_{9\,1}, \quad \ldots \quad \ldots \quad a_{9\,18}, \quad a_{9\,19}).
\end{array}
$$

7.2-rasm. Ikki o‘lchamli massivning xotiradagi joylashuvi

Endi adres nuqtai - nazaridan ko‘p o‘lchamli massiv element-lariga murojaat qilishni ko‘raylik. Quyidagi e’lonlar berilgan bo‘lsin:

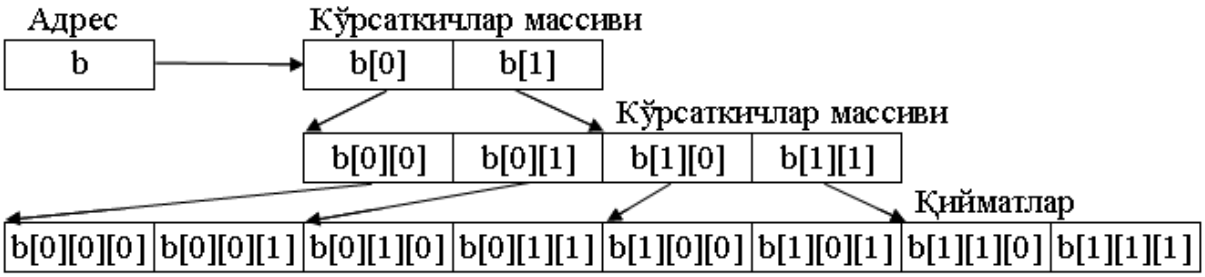
int a[3][2];

float b[2][2][2];

Birinchi e’londa ikki o‘lchamli massiv, ya’ni 2 satr va 3 ustundan iborat matritsa e’lon qilingan, ikkinchisida uch o‘lchamli - 3 ta 2x2 matritsadan iborat bo‘lgan massiv e’lon qilingan. Uning elementlariga murojaat sxemasi:

7.3-rasm. Ikki o‘lchamli massiv elementlariga murojaat

Bu erda a[i] ko‘rsatkichda i-chi satrning boshlang‘ich adresi joylashadi, massiv elementiga a[i][j] ko‘rinishidagi asosiy murojaatdan tashqari vositali murojaat qilish mumkin: *(*(a+i)+j) yoki *(a[i]+j).



7.3-rasm. Uch o‘lchamli massivning xotirada tashkil bo‘lishi

Massiv elementlariga murojaat qilish uchun nomdan keyin kvadrat qavsda har bir o‘lcham uchun indeks yozilishi kerak, masalan b[i][j][k]. Bu elementga vositali murojaat ham qilish mumkin va uning variantlari:

Mustaqil nazorat:

1. Besh boshlang‘ich sonni o‘z ichiga oluvchi vektorlar yaxlitligini aniqlang (2, 3, 5, 7, and 11). push_back funksiyasini element qo‘shish uchun ishlating.

2. o‘zingizni tekshiring 35 ga push_back ni ishlatmasdan javob bering.

3. Quyidagi jadvalda vektorlarning tarkibini aniqlang?

```
vector<string> names;
names.push_back("Ann");
names.push_back("Bob");
names.pop_back();
names.push_back("Cal");
```

4. Faraz qiling siz har besh minutda olingan obhavo o‘lchami to‘plamini saqlamoqchisiz. Vektordan foydalanasizmi yoki jadvaldan?

5. faraz qiling siz hafta kunlari nomlarini saqlamoqchisiz. Vektor yoki 7 string jadvalidan foydalanishingiz kerak?

6. Ikki vektorni to‘ldiruvchi, uchinchi vektorni olib keluvchi funksiyaning sarlavhasini toping. Funksiyani o‘zgartirmang.

7. Bir vektor elementini ikkinchisiga qo‘shuvchi, qisman to‘ldirilgan funksiyani faraz qiling.

```
void append(vector<double>__ target, vector<double>__ source)
{
 for (int i = 0; i < source.size(); i++)
 {
target.push_back(source[i]);
}
}
```

Nazorat savollari

19. C++da Massiv qanday ishlaydi?
20. Massivga kutubxona kerakmi?
21. Rekursiv funksiya nima?
22. Massiv elementlarini bilan massiv indexlarini farqi nimada?
23. For operatori funksiyada qanday ishlatiladi?
24. Matematik funksiyalar qanday ishlaydi?
25. Massiv elementlarini Funksiya parametrlarida uzatish nima uchun ishlatiladi?
26. Massivlarni qabday turlari mavjud?
27. Funksiya parametrlari orqali nima uzatiladi?
28. Massivning necha hil turi bor?