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

Ma’ruza rejasi:

2.1 Hello, World!

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 Hello, World!

Dasturlar. Kompyuterni biron bir amalni bajprishga 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 "std_lib_facilities.h"
intmain() // 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:

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

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

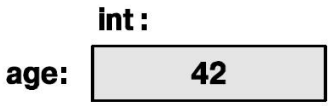
```
cout << "Hello, World!\n"; // "Hello, World!" chiqish
return 0;
```

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

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.

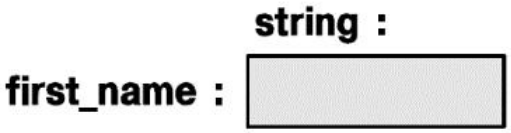
```
cout << "Iltimos, ismingizni kiriting (keyin 'enter' tugmasini bosing):\n";
```

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.



Xotiraan 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:

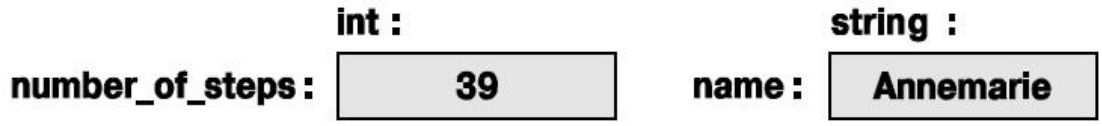
```
Ism — Nicolas
```

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 yoza olmaymiz.

```
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 Carlos ni 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 (whitespace) 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 berish)	=	=	=	=	=
Qo'shish			+	+	
Ulash					+
Ayirish			–	–	
Ko'paytirish			*	*	
Bo'lish			/	/	

Qoldiq			%		
Birga inkrement (oshirish)			++	++	
1ga			– –	– –	
n ga inkrement			+=n	+=n	
Ohiriga qo’shish					+=
n ga qo’shish			-=n	-=n	
Ko’paytirib tenglash			*=	*=	
Bo’lib tenglash			/=	/=	
Qoldiq olish va tenglash			%=		
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 esa n ning kvadrat ildizini topish 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.

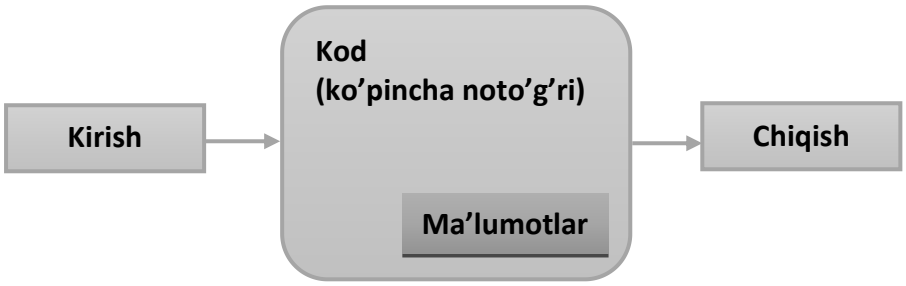
```
// ismni kiritish va taqqoslash
int main()
{
cout << "Iltimos ikkita ism kiriting\n";
string first;
```

```
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.



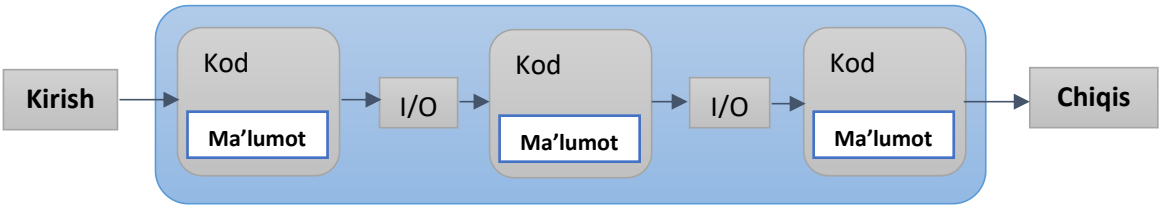
Kiruvchi ma’lumotlar klaviaturadan, sichqonchadan, sensor ekrandan, fayllardan, kiritishning boshqa qurilmalaridan va dasturning boshqa qismlaridan kiritilishi mumkin. "Kiritishning boshqa qurilmalari" kategoriyasi qiziqarli ma’lumotlar manbaini o’z ichiga oladi: musiqiy klavishli pultlar, videoyozuv 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 int toifali 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. Kompilyator aniq 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 taminot va/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.