O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
NUKUS FILIALI

Kompyuter injiniringi fakulteti Dasturiy injiniring yo’nalishi

2-b guruhi talabasi, Bozorboeva Dilraboning

“ C++ dasturlash tili ” fanidan

KURS ISHI.

Mavzu: C++ dasturlash tilida o’zgaruvchilar.

Bajargan: _____

Qabul qilgan: _____

Mavzu: C++ dasturlash tilida o’zgaruvchilar.

Mavzu: C++ dasturlash tilida o’zgaruvchilar.

Reja:

I.Kirish:

a) Hisoblash mashinalari tarixi.
b) C++ dasturlash tili haqida qisqacha malumot.

II. Asosiy qism:

a) C++ dasturlash tilida o’zgaruvchilar.
b) O’zgaruvchi turlari haqida malumot.
c) O’zgaruvchi tiplari.
d) O’zgaruvchilarni tariflash va qiymat berish.
e) O’zgaruvchining xotirada saqlanish uslublari

III. Xulosa.

IV. Amaliy qism.

V. Foydalanilgan adabiyot va internet saytlari.

Kirish.

Biz bu kurs ishi davomida C++ dasturlash tilida o’zgarmaslar haqida bir qancha malumotlarga to’xtalib o’tamiz. Bu kurs ishida men o’zgaruvchilarga kirishishdan oldin uning kelib chiqishiga turtki bo’lgan hisoblash mashinalari va C++ dasturlash tili tarixiga to’xtalib o’tmoqchiman. Kurs ishim davomida men V reja asosida mavzuni o’rganishga qaror qildim. Birinchi reja “Kirish” bo’lib, unda yuqorida aytganizdek hisoblash mashinalari va C++ dasturlash tili yaralishiga to’xtalib o’tamiz. Biz bunda hisoblash mashinalarining yaralishi davrlari, dastlabki insoniyat o’ylab topgan sanoq toshlaridan tortib hozirgi kundagi supper kampyuterlarigacha bo’lgan davrlarni ko’rib o’tamiz. Keyin C++ dasturlash tilining kelib chiqish tarixiga qisqacha to’xtalib o’tamiz. Keyigi reja “Asosiy qisim” bo’lib bu qismni 5 qadamga bo’lib o’rganamiz. Bu qismda o’zgaruvchilar haqida ancha malumotga ega bo’lishimiz mumkin. Bunda o’zgaruvchi turlari, o’zgaruvchi tipi, xotirada joy egallashi va boshqa xususiyatlarini ko’rib o’tamiz. Keyingi reja “xulosa” bu kurs ishini o’rganish davomida olgan bilimlarimizni taxlil qilamiz. Bu kurs ishi faqat nazariy malumotlar jamlamasi bo’libgina qolmay balki, IV bo’limda biz o’zgaruvchilarni amaliy mashqlarda ham ko’rishimiz mumkin. Amaliy mashqlarda biz ozgaruvchi turlariga har biriga misollar keltirib o’tishga harakat qilamiz. Mana shu kurs ishini yoritishda foydalangan adabiyotlar va saytlar bilan oxirgi rejada tanishishimiz mumkin. Kurs ishini mana shu tartibda o’rganib o’ylaymanki biror malumotlarga ega bo’lasiz. Endi malumotlar bilan to’liq tanishib chiqamiz.

Hisoblash mashinalari tarixi.

Insoniyat yatilgandan boshlab turli qurilmalarni yaratdi. Ular nafaqat mahnat qurollari balki boshqa sohalarda ham ko’pgina yangiliklar yaratib, vaqt o’tishi mobaynida takomillasha borgan. Shu jumladan hisob mkitob ishlarini bajarish ishlarini osonlashtirish maqsadida sanoq cho’plaridan tortib chotlargacha ixtiro qilganlar. Informatika fanining rivojlanishiga XX asrda yaratilgan Elektron hisoblash mashinalari sezilarli darajada ta’sir o’tkazdi. Elektron hisoblash mashinalari informatsiya bilan ishlash uchun universal texnik vosita hisoblanadi. Uning hayotimizga kirib kelishi axborotlarni saqlash , ishlov berish va uzatish sohasida tub burulish yasadi. EXMlarning rivojlanish tarixida aloxida o’rinni kompyuterlar egallab kelmoqdalar. Kompyuterlar davri 1971 yilda AKShda mikroprotsessor kashf etilgandan boshlangan desa bo’ladi. Kompyuterlarni ishlab chiqarish avvaliga asosan APPLE firmasi, keyinchalik esa IBM firmasi maxsulotlari xisobiga kengayib bordi. Hozirgi eng zamonaviy IBM PC tizimidagi kompyuterlari hisoblangan 5-avlod EHM lari mantiqiy masalalarni hal qila oladi. Rasm va chizmalarni taniydi. Matnlarni tarjima qila oladi. Multimediya sistemasi yordamida musiqa eshitish, tasvirlarning harakatini ko`rish mumkin. Hozirgi kunda kompyuter va axborot texnologiyalari jadal suratlar bilan rivojlanib borish bilan birga kundalik turmushimizning asosiga aylanib qolmoqda. Jumladan, fan va texnikada, ishlab chiqarish, bank tizimi va matbuot, radio va televideniya, maishiy xizmat ko'rsatish, savdo-sotiq, huquqni muhofaza qilish organlari, tibbiyot va boshqa barcha sohalarida kompyuterlar soha xodimlari uchun asosiy ish quroli vositasi hisoblanadi. Dastlabki va eng sodda suniy hisob asboblaridan biri birkadir. Birka 10 yoki 12 ta tayoqchadan iborat bo`lib, tayoqchalar turli-tuman shakllar bilan o`yilgan. Hisoblash texnikasida mexanik moslamalar davrini boshlab bergan mashinalardan biri nemis olimi Vilgelm Shikkard tomonidan 1623 yili ixtiro qilindi. Biroq, bu hisoblash mashinasi juda tor doiradagi

kishilargagina malum bo`lganligi sababli uzoq vaqtlargacha bu boradagi birinchi ixtirochi 1645 yili arifmometr yasagan frantsuz matematigi Blez Paskal deb hisoblanib kelingan. Lekin, 1958 yili Shtutgart shahri kutubxonasida İ. Keplerning qo`lyozma va hujjatlari orasidan topilgan hisoblash mashinasi chizmasi bu boradagi birinchi ixtirochi Shikkard ekanligini uzil-kesil tasdiqladi. Lekin, Shikkardning mashinasi ham birinchi emas edi. 1967 yili Madriddagi milliy kutubxonada Leonardo da Vinchining nashr qilinmagan ikki jildli qo`lyozmasi topildi. Qo`lyozmaning birinchi jildi mexanikaga bag'ishlangan bo`lib, undagi chizmalar orasida hisoblash qurilmasining chizmasi ham chiqqan. Shu chizma asosida mashina yaratilganda, u qo`shish va ayirish amallarini bajaruvchi qurilma ekanligi malum bo`ldi. Mexanik hisoblash mashinalarida qurilmalar qo`l kuchi bilan harakatga keltirilar edi. Endi mana shu vazifani elektr energiyasi yordamida amalga oshiruvchi hisoblash mashinalari paydo bo`la boshladi. Shuning uchun ham bunday mashinalar elektromexanik hisoblash mashinalari deyiladi. EHM yaratilishi va elementli asoslaridan foydalanish bosqichlari bo'yicha shartli ravishda quyidagi avlodlarga bo'linadi:

- 1-avlod, 1950 yillar. Elektron vakuum lampalarda ishlovchi EHMlar.
- 2-avlod, 1960 yillar. Diskret yarim o'tkazgichli asboblar, ya'ni tranzistorlarda ishlovchi EHMlar.
- 3-avlod, 1970 yillar. Kichik va o'ta yuqori darajada integratsiyasi bo'lgan yarim utkazgichli integral sxemalarda ishlovchi EHMlar.
- 4-avlod, 1980 yillar. Katta va o'ta katta integral sxemalar-mikroprotsessorlarda ishlovchi EHMlar.
- 5-avlod, 1990 yillar. Bilimlarni qayta ishlashning samarali tizimlarini ko'rishga imkon beruvchi, bir qancha o'nlab parallel ishlovchi mikroprotsessorlari bo'lgan EHMlar.

C++ dasturlash tili haqida qisqacha malumot.

C++ dasturlash tili C tiliga asoslangan. C esa o'z navbatida B va BCPL tillaridan kelib chiqqan. BCPL 1967 yilda Martin Richards tomonidan tuzilgan va operatsion sistemalarni yozish uchun mo'ljallangan edi. Ken Thompson o'zining B tilida BCPL ning ko'p hossalarini kiritgan va B da UNIX operatsion sistemasining birinchi versiyalarini yozgan. BCPL ham, B ham tipsiz til bo'lgan. Yani o'garuvchilarning ma'lum bir tipi bo'lmagan - har bir o'zgaruvchi kompyuter hotirasida faqat bir bayt yer egallagan. O'zgaruvchini qanday sifatda ishlatish esa, yani butun sonmi, kasrli sonmi yoki harfdekmi, dasturchi vazifasi bo'lgan. C tilini Dennis Ritchie B dan keltirib chiqardi va uni 1972 yili ilk bor Bell Laboratoriyasida, DEC PDP-11 kompyuterida qo'lladi. C o'zidan oldingi B va BCPL tillarining juda ko'p muhim tomonlarini o'z ichiga olish bilan bir qatorda o'zgaruvchilarni tiplashtirdi va bir qator boshqa yangiliklarni kiritdi. Boshlanishda C asosan UNIX sistemalarida keng tarqaldi. Hozirda operatsion sistemalarning asosiy qismi C/C++ da yozilmoqda. C mashina arxitekturasiga bog'langan tildir. Lekin yaxshi rejalashtirish orqali dasturlarni turli kompyuter platformalarida ishlaydigan qilsa bo'ladi. 1983 yilda, C tili keng tarqalganligi sababli, uni standartlash harakati boshlandi. Buning uchun Amerika Milliy Standartlar Komiteti (ANSI) qoshida X3J11 tehnik komitet tuzildi va 1989 yilda ushbu standart qabul qilindi. Standartni dunyo bo'yicha keng tarqatish maqsadida 1990 yilda ANSI va Dunyo Standartlar Tashkiloti (ISO) hamkorlikda C ning ANSI/ISO 9899:1990 standartini qabul qilishdi. Shu sababli C da yozilgan dasturlar kam miqdordagi o'zgarishlar yoki umuman o'zgarishlarsiz juda ko'p kompyuter platformalarida ishlaydi. C++ 1980 yillar boshida Bjarne Stroustrup tomonidan C ga asoslangan tarzda tuzildi. C++ juda ko'p qo'shimchalarni o'z ichiga olgan, lekin eng asosiysi u ob'ektlar bilan dasturlashga imkon beradi.

Dasturlarni tez va sifatli yozish hozirgi kunda katta ahamiyat kasb etmoda. Buni ta'minlash uchun ob'ektli dasturlash g'oyasi ilgari surildi. Huddi 70-chi yillar boshida strukturali dasturlash kabi, programmalarni hayotdagi jismlarni modellashtiruvchi ob'ektlat orqali tuzish dasturlash sohasida inqilob qildi. C++ dan tashqari boshqa ko'p ob'ektli dasturlshga yo'naltirilgan tillar paydo bo'ldi. Shulardan eng ko'zga tashlanadigani Xeroxning Palo Altoda joylashgan ilmiy-qidiruv markazida (PARC) tuzilgan Smalltalk dasturlash tilidir. Smalltalk da hamma narsa ob'ektlarga asoslangan. C++ esa gibrid tildir. Unda C ga o'hshab strukturali dasturlash yoki yangicha, ob'ektlar bilan dasturlash mumkin. Yangicha deyishimiz ham nisbiydir. Ob'ektli dasturlash falsafasi paydo bo'lganiga ham yigirma yildan oshayapti. C++ sistemasi asosan quyidagi qismlardan iborat. Bular dasturni yozish redaktori, C++ tili va standart kutubxonalardir. C++ dasturi ma'lum bir fazalardan o'tadi. Birinchisi dasturni yozish va tahrirlash, ikkinchisi preprosessor amallarini bajarish, kompilyatsiya, kutubxonalardagi ob'yekt va funksiyalarni dastur bilan bog'lash (link), xotiraga yuklash (load) va bajarish (execute).

C++ dasturlash tilida o’zgaruvchilar.

Dastur ishlashi mobaynida qiymatlari o‘zgarishi mumkin bo‘lgan identifikatorga o‘zgaruvchilar deyiladi. Dasturlash tillarida dastur bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o‘zgaruvchilar va o‘zgarmaslardan foydalaniladi.O‘zgaruvchi-dastur obyekti bo‘lib, xotiradagi bir nechta yacheykalarni egallaydi va berilganlarni saqlash uchun xizmat qiladi. O‘zgaruvchi nomga, o‘lchamga va boshqa atributlarga – ko‘rinish sohasi, amal qilish vaqti va boshqa xususiyatlarga ega bo’ladi. O‘zgaruvchilarni ishlatish uchun ular albatta e’lon qilinishi kerak. E’lon natijasida o‘zgaruvchi uchun xotiradan qandaydir soha zahiralanadi, soha o‘lchami esa o‘zgaruvchining aniq turiga bog‘liq bo’ladi. Shuni qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin.

O’zgaruvchi turlari haqida malumot.

C++ tilida o‘zgaruvchi e’loni uning turini aniqlovchi kalit so‘zi bilan boshlanadi va ‘=’ belgisi orqali boshlang‘ich qiymat beriladi (shart emas). Bitta kalit so‘z bilan bir nechta o‘zgaruvchilarni e’lon qilish mumkin. Buning uchun o‘zgaruvchilar bir-biridan ‘,’ belgisi bilan ajratiladi. E’lonlar ‘;’ belgisi bilan tugaydi. O‘zgaruvchi nomi 255 belgidan oshmasligi kerak. O‘zgaruvchilarni e’lon qilish dastur matnining istalgan joyida amalga oshirilishi mumkin.

C++ tilida ham o’zgaruvchilarning turlari bir necha guruhlarga ajraladi. Ularni quyida qarab chiqamiz.

Butun son turlari. Butun son qiymatlarni qabul qiladigan o‘zgaruvchilar int(butun), short(qisqa) va 1ong(uzun) kalit so‘zlar bilan aniqlanadi. O‘zgaruvchi qiymatlari ishorali bo‘lishi yoki unsigned kalit so‘zi bilan ishorasiz son sifatida qaralishi mumkin.

Belgi turi. Belgi turidagi o‘zgaruvchilar char kalit so‘zi bilan beriladi va ular o‘zida belgining ASCII kodini saqlaydi. Belgi turidagi qiymatlar nisbatan murakkab bo‘lgan tuzilmalar – satrlar, belgilar massivlari va hokazolarni hosil qilishda ishlatiladi.

Haqiqiy son turi. Haqiqiy sonlar float kalit so‘zi bilan e’lon qilinadi. Bu turdagi o‘zgaruvchi uchun xotiradan 4 bayt joy ajratiladi va <ishora><tartib><mantissa> qolipida sonni saqlaydi. Agar kasrli son juda katta (kichik) qiymatlarni qabul qiladigan bo‘lsa, u xotirada 8 yoki 10 baytli ikkilangan aniqlik ko‘rinishida saqlanadi va mos double va long double kalit so‘zlari bilan e’lon qilinadi. Oxirgi holat 32-razryadli platformalar uchun o‘rinli.

Mantiqiy tur. Bu turdagi o‘zgaruvchi bool kalit so‘zi bilan e’lon qilinib, xotiradan 1 bayt joy egallaydi va 0 (false, yolg‘on) yoki (true, rost) qiymat qabul qiladi. Mantiqiy tur o‘zgaruvchilar qiymatlar o‘rtasidagi munosabatlarni

ifodalaydigan mulohazalarni rost (true) yoki yolg‘on (false) ekanligi tavsifida qo‘llaniladi va ular qabul qiladigan qiymatlar matematik mantiq qonuniyatlariga asoslanadi.

O’zgaruvchi tiplari.

C++ tilida o‘zgaruvchilar ma’lumotni saqlash uchun qo‘llaniladi. O‘zgaruvchining dasturda foydalanish mumkin bo‘lgan qandaydir qiymatlarni saqlaydigan komьyuter xotirasidagi yacheyka ko‘rinishda ifodalash mumkin. Kompyuter xotirasini yacheykalardan iborat qator sifatida qarash mumkin. Barcha yacheykalar ketma – ket nomerlangan. Bu nomerlar yacheykaning adresi deb ataladi. O‘zgaruvchilar biror – bir qiymatni saqlash uchun bir yoki bir nechta yacheykalarni band qiladi. O‘zgaruvchining nomini (masalan, MyVariable) xotira yacheykasi adresi yozilgan yozuv deb qarash mumkin. Masalan MyVariable o‘zgaruvchisi 102 – adresdagi yacheykadan boshlab saqlanadi. O‘zining o‘lchoviga muvofiq MyVariable o‘zgaruvchisi xotiradan bir yoki bir necha yacheykani band qilishi mumkin.

O‘zgaruvchilarning quyidagi tiplari mavjuddir:

bool – mantiqiy;

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 xaqiqiy son;

long float yoki double – ikkilangan xaqiqiy son

long double – uzun ikkilangan xaqiqiy son

Butun sonlar o‘lchami. Bir xil tipdagi o‘zgaruvchilar uchun turli kompyuterlarda xotiradan turli hajmdagi joy ajratilishi mumkin. Lekin, bitta kompyuterda bir xil tipdagi ikkita o‘zgaruvchi bir xil miqdorda joy egallaydi. Char tipli o‘zgaruvchi bir bayt hajmni egallaydi. Ko‘pgina komьpyuterlarda short int (qisqa butun) tipi ikki bayt, long int tipi esa 4 bayt joy egallaydi. Butun qiymatlar o‘lchovini kompьyuter sistemasi va ishlatiladigan kompilyator aniqlaydi. 32 – razryadli kompьyuterlarda butun o‘zgaruvchilar 4 bayt joy egallaydi.

O’zgaruvchilarni tariflash va qiymat berish.

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 initsializatsiya ya’ni boshlang‘ich qiymatlarini ko‘rsatish mumkin.

Misol uchun:

```
int i=0;

char c='k';
```

O‘zgaruvchilarga qiymat berish uchun o‘zlashtirish operatori qo‘llaniladi. Masalan, Width o‘zgaruvchisiga 5 qiymatni berish uchun quyidagilarni yozish lozim:

```
unsigned short Width;

Width = 5;
```

Bu ikkala satrni Width o‘zgaruvchisini aniqlash jarayonida birgalikda yozish mumkin.

```
unsigned short Wigth = 5;
```

Bir necha o‘zgaruvchilarni aniqlash vaqtida ham ularga qiymat berish mumkin:

```
long width = 5, length = 7;
```

Bu misolda long tipidagi width o‘zgaruvchisi 5 qiymatni, shu tipdagi length o‘zgaruvchisi esa 7 qiymatni qabul qildi. Const_cast esa o'zgaruvchilardan const (o'zgarmas) va volatile (o'zgaruvchan, uchuvchan) sifatlarini olib tashlashda qo'llaniladi. Odatda const o'zgaruvchining qiymatini o'zgartirib bo'lmaydi. Ushbu holda const_cast qo'llaniladi. reinterpret_cast odatiy bo'lmagan keltirishlarni bajarishda qo'llaniladi (masalan void* ni int ga). reinterpret_cast o'zgaruvchining bitlarini boshqa ma'noda qo'llashga imkon beredi. Bu operator bilib ishlatilinishi kerak.

O’zgaruvchining xotirada saqlanish uslublari

O'zgaruvchilarning kattaligi, ismi va turidan tashqari yana bir necha boshqa hossalari bor. Bulardan biri hotirada saqlanish tipidir. O'zgaruvchilar hotirada ikki uslubda saqlanishi mumkin. Birinchisi avtomatik, ikkinchisi statik yo'ldir. Avtomatik bo'lgan birlik u e'lon qilingan blok bajarilishi boshlanganda tuziladi, va ushbu blok tugaganda buziladi, u hotirada egallagan joy esa bo'shatiladi. Faqat o'zgaruvchilar avtomatik bolishi mumkin. Avtomatik sifatini berish uchun

o'zgaruvchi boshiga auto yoki register so'zlari qo'yiladi. Aslida lokal o'zgaruvchilar oldiga hech narsa yozilmasa, ularga auto sifati beriladi. Dastur ijro etilganda o'zgaruvchilar markaziy prosessor registrlariga yuklanib ishlov ko'radilar. Keyin esa yana hotiraga qaytariladilar. Agar register sifatini qo'llasak, biz kompyuterga ushbu o'zgaruvchini ishlov ko'rish payti davomida registrlarning birida saqlashni tavsiya etgan bo'lamiz. Bunda hotiraga va hotiradan yuklashga vaqt ketmaydi. Albatta bu juda katta vaqt yutug'i bermasligi mumkin, lekin agar sikl ichida ishlatilsa, yutuq sezilarli darajada bo'lishi mumkin. Shuni atish kerakki, hozirgi kundagi kompilyatorlar bunday ko'p ishlatiladigan o'zgaruvchilarni ajrata olishdi va o'zlari ular bilan ishlashni optimizatsiya qilishadi. Shu sababli o'zgaruvchini register deb e'lon qilish shart bo'lmay qoldi. Hotirada boshqa tur saqlanish yo'li bu statik saqlanishdir. Statik sifatini o'zgaruvchi va funksiyalar olishlari mumkin. Bunday birliklar dastur boshlanish nuqtasida hotirada quriladilar va dastur tugashiga qadar saqlanib turadilar. O'zgaruvchi va funksiyalarni statik qilib e'lon qilish uchun static yoki extern (tashqi) ifodalari e'lon boshiga qo'yiladi. Statik o'zgaruvchilar dastur boshida hotirada quriladilar va initsalizatsiya qilinadilar. Fuksiyalarning ismi esa dastur boshidan bor bo'ladi. Lekin statik birliklar dastur boshidan mavjud bo'lishi, ularni dasturning istalgan nuqtasida turib qo'llasa bo'ladi degan gap emas. Hotirada saqlanish uslubi bilan qo'llanilish sohasi tushunchalari farqli narsalardir. O'zgaruvchi mavjud bo'lishi mumkin, biroq ijro ko'rayatgan blok ichida ko'rinmasligi mumkin. Dasturda ikki xil statik birliklar bor. Birinchi xili bu tashqi identefikatorlardir. Bular global sohada aniqlangan o'zgaruvchi va funksiyalardir. Ikkinchi tur statik birliklar esa static ifodasi bilan e'lon qilingan lokal o'zgaruvchilardir. Global o'zgaruvchi va funksiyalar oldida extern deb yozilmasa ham ular extern sifatiga ega bo'ladilar. Global o'zgaruvchilar ularning e'lonlarini funksiyalar tashqarisida yozish bilan olinadi. Bunday o'zgaruvchi va funksiyalar o'zlaridan faylda keyin keluvchi har qanday funksiya tomonidan qo'llanilishi mumkin. Global o'zgaruvchilarni ehtiyotorlik bilan ishlatish kerak. Bunday o'zgaruvchilarni harqanday funksiya

o'zgartirish imkoniga ega. O'zgaruvchiga aloqasi yo'q funksiya uning qiymatini bilib-bilmasdan o'zgartirsa, dastur mantig'i buzilishi mumkin. Shu sababli global sohada iloji boricha kamroq o'zgaruvchi aniqlanishi lozim. Faqat bir joyda ishlatilinadigan o'zgaruvchilar o'sha blok ichida aniqlanishi kerak. Ularni global qilish noto'g'ridir. Lokal o'zgaruvchilarni, yani funksiya ichida e'lon qilingan o'zgaruvchilarni static so'zi bilan e'lon qilish mumkin. Bunda ular ikkinchi hil statik birliklarni tashkil qilishgan bo'lishadi. Albatta ular faqat o'sha funksiya ichida qo'llanishlari mumkin. Ammo funksiya bajarilib tugaganidan so'ng statik o'zgaruvchilar o'z qiymatlarini saqlab qoladilar va keyingi funksiya chaqirig'ida saqlanib qolingan qiymatni yana ishlatishlari yoki o'zgartirishlari mumkin. Statik o'zgaruvchilar e'lon paytida initsalizatsiya qilinadilar. Agar ularga e'lon paytida ochiqchasiga qiymat berilmagan bo'lsa, ular nolga tenglashtiriladi.

static double d = 0.7; // ochiqchasiga qiymat berish,

static int k; // qiymati nol bo'ladi. Agar static yoki extern ifodalari global identefikatorlar bilan qo'llanilsa, ushbu identefikatorlar mahsus ma'noga egadirlar. Biz u hollarni keyin ko'rib o'tamiz. O'zgaruvchi dasturning faqat ma'lum sohasida ma'moga egadir. Yani faqat biror bir blok, yoki bu blok ichida joylashgan bloklar ichida qo'llanilishi mumkin. Bunday blokni soha (qo'llanilish sohasi - scope) deb ataylik. Identefikator (oz'garuvchi yoki funksiya ismi) besh hil sohada aniqlanishi mumkin. Bular funksiya sohasi, fayl sohasi, blok sohasi, funksiya prototipi sohasi va klas sohasi. Agar identefikator e'loni hech bir funksiya ichida joylashmagan bo'lsa, u fayl sohasiga egadir. Ushbu identefikator e'lon nuqtasidan to fayl ohirigacha ko'rinadi. Global o'zgaruvchilar, funksiya prototiplari va aniqlanishlari shunday sohaga egadirlar. Etiketlar (label), yani identefikatorlardan keyin ikki nuqta (:) keluvchi ismlar, masalan: chiqish: mahsus ismlardir. Ular dastur nuqtasini belgilab turadilar. Dasturning boshqa yeridan esa ushbu nuqtaga sakrashni (jump) bajarish mumkin. Va faqat etiketlar funksiya sohasiga egadirlar. Etiketlarga ular e'lon qilingan

funksiyaning istalgan joyidan murojaat qilish mumkin. Lekin funksiya tashqarisidan ularga ishora qilish ta'qiqlanadi. Shu sababli ularning qo'llanilish sohasi funksiyadir. Etiketlar switch va goto ifodalarida ishlatilinadi. goto qo'llanilgan bir blokni misol qilaylik.

```
int factorial(int k) {
if (k<2)
   goto end;
else
   return ( k*factorial(k-1) );
 end:
   return (1);
}
```

Bu funksiya sonning faktorialini hisoblaydi. Bunda 0 va 1 sonlari uchun faktorial 1 ga teng, 1 dan katta x soni uchun esa x!=x*(x-1)*(x-2)...2*1 formulasi bo'yicha hisoblanadi. Yuqoridagi funksiya rekursiya metodini ishlatmoqda, yani o'zini-o'zini chaqirmoqda. Bu usul dasturlashda keng qo'llaniladi. Funksiyamiz ichida bitta dona etiket - end: qollanilmoqda. Etiketlarni qo'llash strukturali dasturlashga to'g'ri kelmaydi, shu sababli ularni ishlatmaslikga harakat qilish kerak. Blok ichida e'lon qilingan identefikator blok sohasiga egadir. Bu soha o'zgaruvchi e'lonidan boshlanadi va } qavsda (blokni yopuvchi qavs) tugaydi. Funksiyaning lokal o'zgaruvchilari hamda funksiyaning kiruvchi parametrlari blok sohasiga egadirlar. Bunda parametrlar ham funksiyaning lokal o'zgaruvchilari qatoriga kiradilar. Bloklar bir-birining ichida joylashgan bo'lishi mumkin. Agar tashqi blokda ham, ichki blokda ham ayni ismli identefikator mavjud bo'lsa, dastur ijrosi ichki blokda sodir bo'layatgan bir vaqtda ichki identefikator tashqi blokdagi identefikatorni to'sib turadi. Yani ichki blokda tashqi blok identefikatorining ismi ko'rinmaydi. Bunda ichki blok faqat o'zining o'zgaruvchisi bilan ish yuritishi mumkin. Ayni ismli tashqi blok identefikatorini ko'rmaydi. Lokal o'zgaruvchilar static deya belgilanishlariga qaramay, faqat aniqlangan bloklaridagina qo'llanila

oladilar. Ular dasturning butun hayoti davomida mavjud bo'lishlari ularning qo'llanilish sohalariga ta'sir ko'rsatmaydi. Funksiya prototipi sohasiga ega o'zgaruvchilar funksiya e'lonida berilgan identefikatorlardir. Aytib o'tkanimizdek, funksiya prototipida faqat o'zgaruvchi tipini bersak yetarlidir. identefikator ismi berilsa, ushbu ism kompilyator tomonidan hisobga olinmaydi. Bu ismlarni dasturning boshqa yerida hech bir qiyinchiliksiz qo'llash mumkin. Kompilyator hato bermaydi. Klas sohasiga ega ismlar klas nomli bloklarda aniqlanadilar. Bizlar klaslarni keyinroq o'tamiz. Hozir soha va hotirada saqlanish tipi mavzusida bir misol keltiraylik.

```
//Qo'llanilish sohasi, static va auto
//o'zgaruvchilarga misollar.
 # include <iostream.h>
 long r = 100; //global o'zgaruvchi,
                //funksiyalar tashqarisida aniqlangan
 void staticLocal(); //funksiya prototipi yoki e'loni
void globalAuto(int k /* k funksiya prototip sohasiga
ega */); //f-ya e'loni
 int main ()
{
   staticLocal();
   staticLocal();
    int m = 6;
   globalAuto(m);
   ::r = ::r + 30;
   cout "main da global long r: ";
   cout << ::r << endl; //global long r to'liq
aniqlangan
                        //ismi o'rqali qo'llanilmoqda
   m++;//m = 7
   globalAuto(m);
    int r = 10; //tashqi sohadagi main ga nisbatan lokal
o'zgaruvchi;
               //long r ni to'sadi
   cout << "tashqi sohadagi lokal r: " << r << endl;
      { //ichki blok
         short r = 3; //ichki sohadagi lokal
o'zgaruvchi;
                       //int r ni to'sadi
```

```
            cout << "ichki sohadagi lokal r: " << r <<
endl;
        }
    cout << "tashqi sohadagi lokal r: " << r << endl;
    return (0);
}
 void staticLocal() {
    static  int s = 0; //statik o'zgaruvchi
     cout << "staticLocal da: " << s << endl;
    s++; //s = 1;
}
 void globalAuto(int i) {
    int g = 333; //avtomatik o'zgaruvchi
     cout << "globalAuto da: " << i << " ";
    cout << g << " ";
    g++;
    cout << r << endl; //global long r ekranga bosiladi
}
```

Ekranda:

```
 staticLocal da: 0
staticLocal da: 1
globalAuto da: 6 333 100
main da global long r: 130
globalAuto da: 7 333 130
tashqi sohadagi lokal r: 10
ichki sohadagi lokal r: 3
tashqi sohadagi lokal r: 10
```

Tiplarning hotiradagi kattaligini kopsatadigan, bir parametr oladigan sizeof() (sizeof - ning kattaligi) operatori mavjuddir. Uning yordamida tiplarning, o'zgaruvchilarning yoki massivlarning kattaliklarini aniqlash mumkin. Agar

o'zgaruvchi nomi berilsa, () qavslar berilishi shart emas, tip, massiv va pointer nomlari esa () qavslar ichida beriladi. Bir misol beraylik.

```
// sizeof() operatori
# include <iostream.h>
 int k;
int *pk;
 char ch;
char *pch;
 double dArray[20];
 int main()
{
   cout << sizeof (int) << " - " << sizeof k << " - "
<< sizeof (pk) << endl;
      //         tip  nomi                  o'zgaruvchi
pointer
   cout <<sizeof (char) << " - " <<sizeof ch << " - "
<<sizeof (pch) << endl;
    cout  <<  "\nMassiv  hotirada  egallagan  umumiy  joy
(baytlarda): "
        << sizeof (dArray) << endl;
   cout << "Massivning alohida elementi egallagan joy:
"
        << sizeof (double) << endl;
   cout << "Massivdagi elementlar soni: "
        << sizeof (dArray) / sizeof (double) << endl;
    return (0);
}
```

Ekranda:

```
4 - 4 - 4
```

```
1 - 1 - 4
```

Massiv hotirada egallagan umumiy joy (baytlarda): 160

Massivning alohida elementi egallagan joy: 8

Massivdagi elementlar soni: 20

Kompyuter hotirasidan 4 va 5 soni joylashishi uchun alohida joy ajratiladi, ularning yig'indisini saqlovchi uchinchi son uchun yana bir qism hotira va so'nggi natijani saqlash uchun yana bir qism hotira. Albatta yuqoridagi misol juda sodda, lekin kompyuteringiz bunga o'xshash misollardan milliontasini bir vaqtning o'zida bajara oladi.

Yuqoridagi misolda sonlarni yoki boshqa ma'lumotlar ifodalovchilarini dasturlash tilida o'zgaruvchilar deb nomlanadi. Ular turli ma'lumotlarni o'zida saqlab turishi uchun kompyuterdan ma'lum bir hotira ajratiladi. C++ tilida o'zgaruvchilarni istalgan nom bilan nomlash mumkin. Masalan yuqoridagi "a" bilan nomlangan sonni "birinchi_son" deb e'lon qilish ham mumkin edi.

Ma'lumot turlari asosan 4 qismga bo'linadi:

```
Sonli – int (1, 2, 67, ….)
Belgili – char ('a', '7', '%',…)
Suzuvchi – float, double  (2.45, 3.33, 1.03, ….)
Mantiqiy – boolean (true, false yoki 0, 1)
```

O'zgaruvchilar va ma'lumotning sonli turidan foydalanib yuqoridagi misolni dastur ko'rinishini yozamiz

```
#include <iostream> ;
using namespace std;
int main ()
{
// o'zgaruvchilarni e'lon qilish:
```

```
int birinchi_son, b, summa;
int natija;
// asosiy o’zlashtirish va hisob jarayoni:
birinchi_son = 4;
b = 5;
summa = birinchi_son + b;
natija = summa - 3;
// natijani chop qilamiz:
cout<<natija;
dasturdan chiqish:
return 0;
}
```

Dastur tuzish mobaynida, hosil bo‘ladigan natijalar ma’lum bir joyda saqlanishi lozim. Bu joy operativ xotiradir. Operativ xotira hajmi, dasturchilar uchun juda muhim hisoblanadi. Bundan ko‘rinib turibdiki, dasturchilar «**xotira**» so‘zini ishlatishsa, «**operativxotira**» tushunilishi lozim. Bu xotirada qiymatlar(natijalar) ma’lum bir ism bilan saqlanishi lozim(bo‘lmasa kerakli qiymatni u yerdan qanday topasiz), bu ism dasturlashda «**o‘zgaruvchi nomi**» deyiladi. **O‘zgaruvchilar** — ma’lum bir nomga va tipga ega bo‘lib, o‘zida qandaydir qiymatlarni saqlash uchun ishlatiladi. Bu qiymatlar keyinchalik, o‘zgaruvchi nomi orqali chaqirib, ishlatilishi mumkin bo‘ladi. Xotira bir necha logik qisimlarga bo‘linadi, bu qismlar **yacheyka** deb yuritiladi. Har bir yacheyka nomerlangan bo‘ladi va bu nomerlar **xotira adresi** deyiladi. Biz o‘rganayotgan o‘zgaruvchilar, bitta yoki bir necha xotira adresini egallashi mumkin bo‘ladi(albatta tipiga qarab). Har bir yacheyka **1 bayt** o‘lchamida bo‘ladi.

C++ dasturlash tilida biror o‘zgaruvchi e’lon qilsangiz, albatta uning tipini ko‘rsatishingiz kerak bo‘ladi. Shu tip orqali kompilyator xotiradan nechta yacheyka ajratish kerakligini oldindan bilib oladi va shu joyni band qilib qo‘yadi. Agar int tipidagi o‘zgaruvchi e’lon qilsangiz, kompilyator xotiradan 4 bayt joyni

band qilib qo‘yadi, toki bu o‘zgaruvchi bo‘shatilib, o‘chirib tashlamaguncha. C++ dasturlash tilida asosan 3 xildagi tip ishlatiladi: **butun sonlar**, **haqiqiy sonlar** va **simvol**. Butun sonlar musbat va manfiy raqamlardan tashkil topgan sonlardir, haqiqiy sonlar esa kasr ko‘rinishidagi musbat va manfiy sonlardir. Simvollar esa tushunarli, simvollardan tashkil topgan qiymatlardir. Quyida tiplarning to‘liq ro‘yxati (nomi, hajmi, qiymati)ni ko‘rishingiz mumkin.

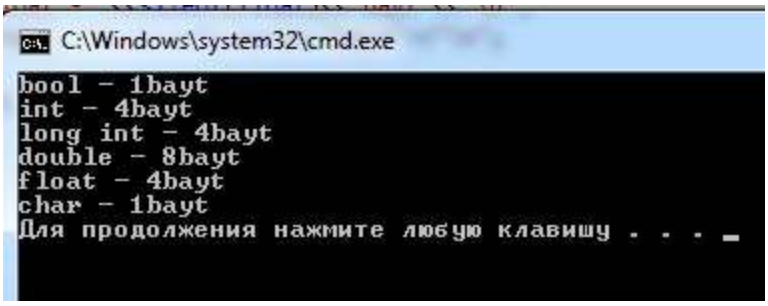
Тип	Ўлчам(байт)	Қиймат
bool	1	**True** ёки **false**
unsignet short int	2	**0 - 65535**
short int	2	**-32768 - 32768**
unsignet long int	4	**0 - 4294967295**
long int	4	**-2147483648 - 2147483647**
int(16 разрядли)	2	**-32768 - 32767**
int(32 разрядли)	4	**-2147483648 - 2147483647**
unsignet int(16 разрядли)	2	**0 - 65535**
unsignet int(32 разрядли)	4	**0 - 4294967295**
char	1	**256** та символ
float	4	**1,2e-38 - 3,4e38**
double	8	**2,2e-308 - 1,8e308**

Ko‘rib turganingizdek, «**unsignet**» so‘zi o‘sha tipni barcha maksimum qiymatlarini qo‘shib, faqat musbat qiymatlarni qabul qila olar ekan. Ba’zi kompyuterlarda o‘zgaruvchi tiplari xotiradan xar hil yacheyka miqdorini olishi mumkin(kompyuter razryadiga qarab). O‘zgaruvchi xotiradan nechi baytini band qilishini aniq bilishi uchun, C++da «**sizeof**» nomli funksiya mavjud. keling shu funksiyani misolda ko‘ramiz.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
cout<<"bool - "<<sizeof(bool)<<"bayt"<<"\n";
```

```
cout<<"int - "<<sizeof(int)<<"bayt"<<"\n";
cout<<"long int - "<<sizeof(long int)<<"bayt"<<"\n";
cout<<"double - "<<sizeof(double)<<"bayt"<<"\n";
cout<<"float - "<<sizeof(float)<<"bayt"<<"\n";
cout<<"char - "<<sizeof(char)<<"bayt"<<"\n";
return 0;
}
```

Natija:



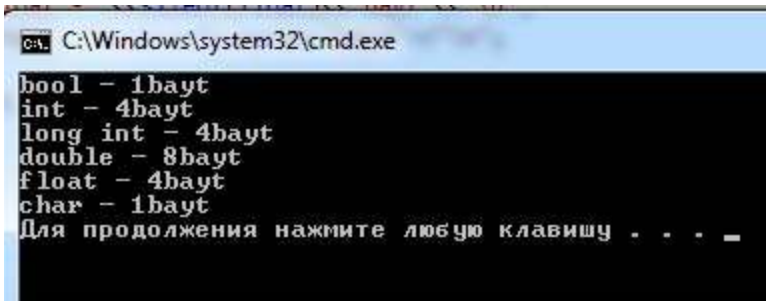
Natijadan ko‘rinib turibdiki, bu funksiya baytlarni aniq ko‘rsatib beradi.

Endi bu o‘zgaruvchilarni kod ichida qo‘llaymiz. Demak, bir nechta o‘zgaruvchilarni e’lon qilamiz va amallar bajaramiz.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int test1; test1 = 10;
short int test2=20;
long int test3, test4;
float test5, test6 = 1.5;test3 = test1+test2;
test4 = test3;
test5 = test6*2;cout <<"test1="<<test1<<"\n";
cout <<"test2="<<test2<<"\n";
```

```
cout <<"test3="<<test3<<"\n";
cout <<"test4="<<test4<<"\n";
cout <<"test5="<<test5<<"\n";
cout <<"test6="<<test6<<"\n";
return 0;
}
```

Natija:



Agar o‘zgaruvchi tipi qiymatga mos kelmasa xatolik chiqadi. Shuning uchun bir son yoki simvol ishlatishdan oldin, o‘zgaruvchi tipini to‘g‘ri qo‘ying. O‘zgaruvchi nomi sifatida harflar, sonlar, simvollar ishlatilishi mumkin. O‘zgaruvchi nomining dastlabki simvoli harf bo‘lishi lozim. Undan keyin C++ tilining xizmatchi so‘zlarini nom sifatida qo‘yish ham mumkin emas(if, while, cout). C++ dasturlash tili registrni ham inobatga oladi, ya’ni katta kichik harflarning farqi bor. Agar o‘zgaruvchi nomi sifatida «**test**«, «**Test**«, «**TEST**» so‘zlaridan foydalanchangiz, bularning barchasi boshqa-boshqa o‘zgaruvchi hisoblanadi(katta-kichik harflar farq qilgani uchun).

Xulosa:

Xulosa qilib aytadigan bo’lsak biz bu kurs ishini o’rganish va tayorlash mobaynida o’zgaruvchilarning bir qancha xususyatlari bilan tanishib o’tdik. O’zgaruvchilar tipi, turi, xotirada egallashi mumkin bo’lgan joy, ularning tariflanishi, qo’llanilishi kabilar shular jumlasidan. O’zgaruvchilar dasturning bacha joylarida elon qilinishi mumkin. O’zgaruvchiga qisqacha to’xtaladigan bo’lsak, Dastur ishlashi mobaynida qiymatlari o‘zgarishi mumkin bo‘lgan identifikatorga o‘zgaruvchilar deyiladi. Dasturlash tillarida dastur bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o‘zgaruvchilar va o‘zgarmaslardan foydalaniladi.O‘zgaruvchi-dastur obyekti bo‘lib, xotiradagi bir nechta yacheykalarni egallaydi va berilganlarni saqlash uchun xizmat qiladi. O‘zgaruvchi nomga, o‘lchamga va boshqa atributlarga – ko‘rinish sohasi, amal qilish vaqti va boshqa xususiyatlarga ega bo’ladi. O‘zgaruvchilarni ishlatish uchun ular albatta e’lon qilinishi kerak. E’lon natijasida o‘zgaruvchi uchun xotiradan qandaydir soha zahiralanadi, soha o‘lchami esa o‘zgaruvchining aniq turiga bog‘liq bo’ladi. Shuni qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin. O’zgaruvchilarga quyida har bir turi uchun masalalar ko’rib chiqamiz. Shun bilan ularni amallda qo’llashni ham o’rganib olamiz. Manashu kurs ishi davomida o’zgaruvchilarga qisqacha to,xtalib o’tdik, ammo o’zgaruvchilarni to’liq o’rganish uchun bu juda kamlik qiladi. O’zgaruvchilarni qo’llay olishni o’rganish xar bir dasturchi uchun juda zarur.

Amaliy qism

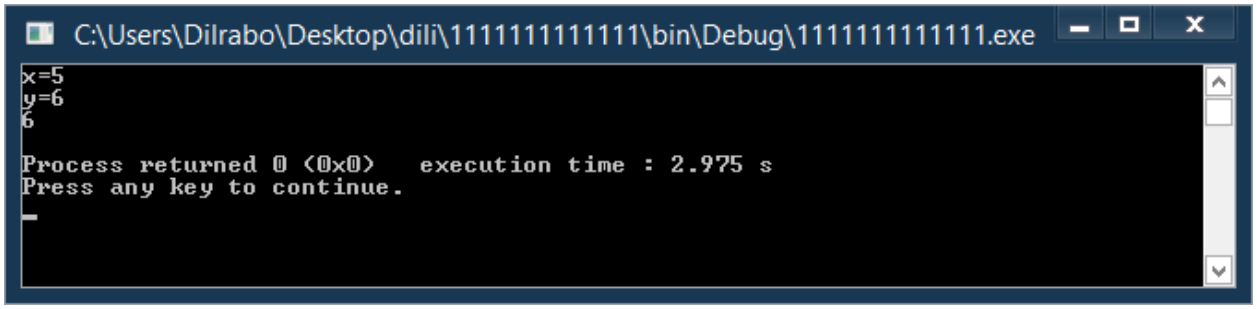
Butun son tur uchun misollar:

1. Masala sharti: x, y haqiyqiy sonlari berilgan ulardan kattasini chiqaruvchi dastur tuzing.

Dastur kodi:

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    int y, x, max;
    cout <<"x="; cin >>x;
cout<<"y=";cin>>y;
if ( x>y)
max=x;
Else
max=y;
    cout << max<< endl;
    return 0;
```

Dastur interfeysi:

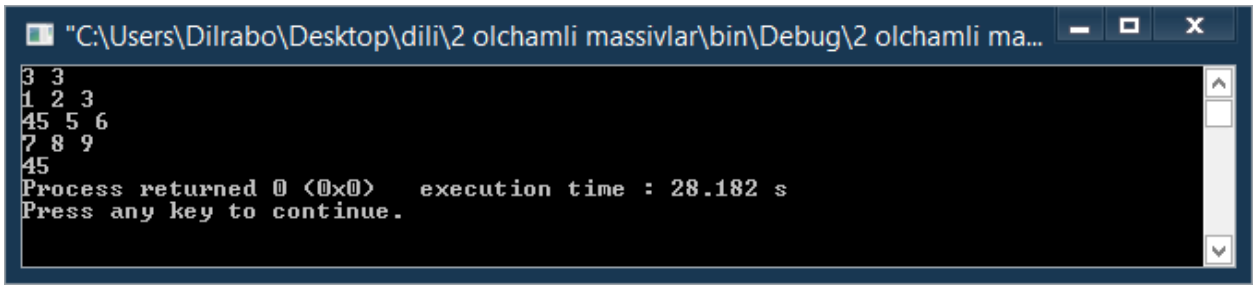


2. Masala sharti: a[n][m] Matritsa berilgan berilgan matritsa elementlari orasida eng kattasini aniqlovchi dastur tuzing:

Dastur kodi:

```
#include <iostream>
using namespace std;
int main()
{
    int i,j,n,m,max;
    cin>>n>>m;
    int a[n][m];
    for (i=0; i<n; i++)
        for (j=0; j<m; j++)
        cin>>a[i][j];
         max=a[0][0];
    for (i=0; i<n; i++)
   for (j=0; j<m; j++)
   if (max<a[i][j])
    max=a[i][j];
   cout<<max;
    return 0;
}
```

Dastur interfeysi:

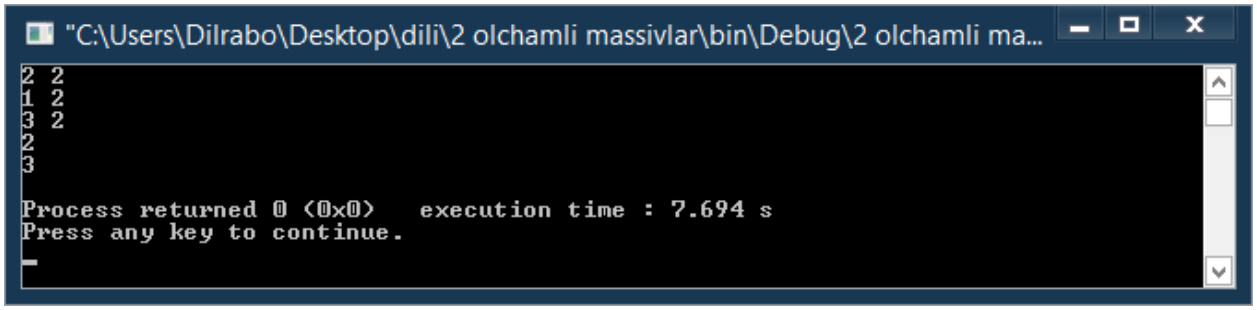


3. Masala sharti: Berilgan N:M matritsadan har bir ustundagi eng katta elementni hisoblash dasturi tuzilsin

Dastur kodi:

```
#include <iostream>
using namespace std;
int main()
{
    int i,j,n,m,max;
    cin>>n>>m;
    int a[n][m];
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
        cin>>a[i][j];
    for (i=0; i<n; i++)
    {
        max=a[i][0];
        for(j=0; j<m; j++)
            if (max<a[i][j])
                max=a[i][j];
        cout<<max<<endl;
    }
    return 0;
}
```

Dastur interfeysi:

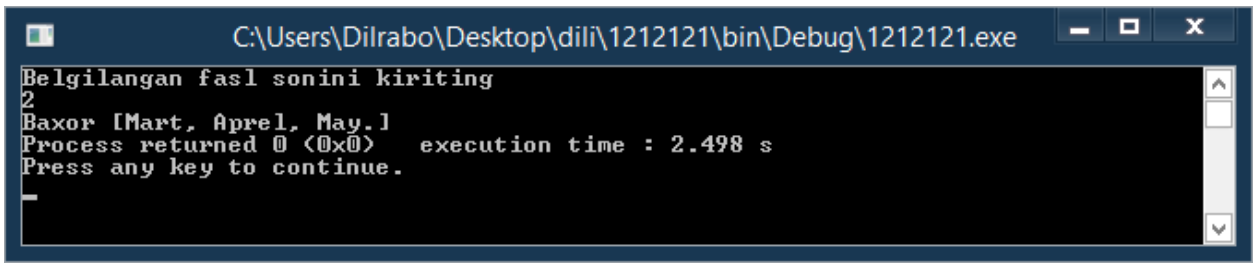


4. Masala sharti: Yil faslini bildiruvchi 1, 2, 3, 4 sonlaridan biri kiritilsin. Shu songa tegishli fasl oylarini chiqaruvchi dastur tuzilsin.

Dastur kodi:

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout <<"Belgilangan fasl sonini kiriting"<<endl;
    cin>>n;
    switch (n)
    {
case 1: cout << "Qish [Dekabr, Yanvar, Fevral.] ";
break;
case 2: cout << "Baxor [Mart, Aprel, May.] "; break;
case 3: cout << "Yoz [Iyun, iyul, avgust.] "; break;
case 4: cout << "Kuz [Sentabr, Oktabr, Noyabr.] ";
break;
        default : cout <<"Bir yilda to'rt fasl bor";
    }
    return 0;
}
```

Dastur interfeysi:



Haqiqiy turi uchun misol:

1. Masalaning berilishi: Kvadratning tomoni berilgan uning peremetirini aniqlovchi dastur tuzing.

Dastur kodi:

```
#include <iostream>
using namespace std;
int main()
{
     float a,p;
    cout<<"Kvadratning tomonini kiriting"<<endl;
    cout<<"a=";cin>>a;
    p=4*a;
    cout<<p<<endl;
    return 0;
}
```

Dastur interfeysi:

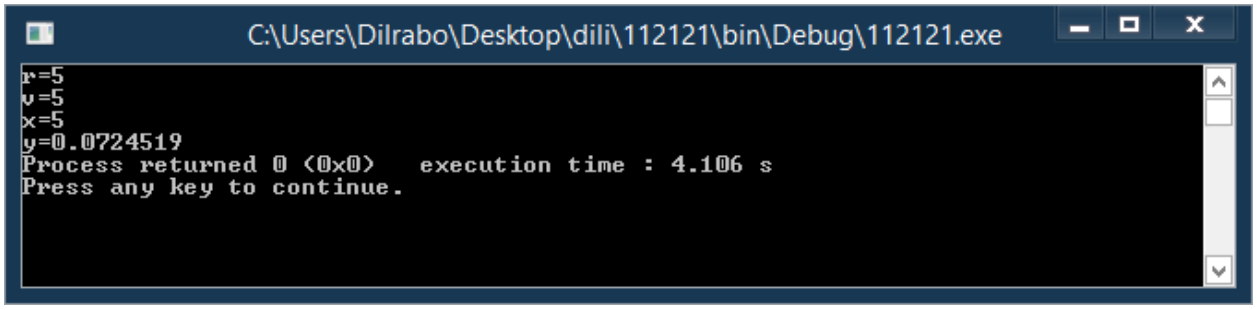
2. Masala sharti: Quyidagi funksiyani hisoblovchi dastur tuzilsin:

$$a)\ y=\frac{\frac{r^2}{4}ln\left|v+\sqrt{x^2-r^2}\right|-r^{-0.0004}}{\sqrt{x+\sqrt{9+x^2}+v^{r+1}}}c$$

Dastur kodi:

```
#include <iostream>
#include <math.h>
 using namespace std;
 int main()
{
    double v,r,x;
    double y;
    cout <<"r=";cin>>r;
    cout <<"v=";cin>>v;
    cout <<"x=";cin>>x;
  y=((((r*r)/4)*(log(v+sqrt(v*v-r*r)))-
1/pow(r,0.0004))/(sqrt(x+sqrt(9+x*x)+pow(v,r+1))));
    cout<<"y="<<y;
    return 0;
}
```

Dastur interfeysi:



Mantiqiy tur uchun misol:

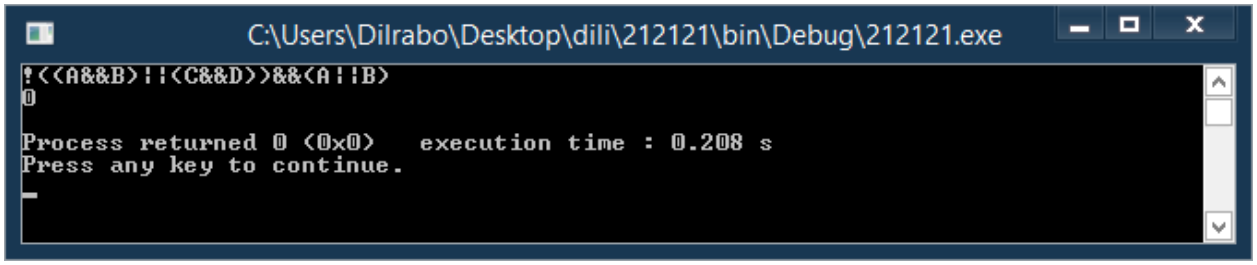
Masala sharti: A=true, B=true, C=false, D=true bolsa quyidagi ifoda qiymatini hisoblovchi dastur tuzilsin.

```
!((A&&B)||(C&&D))&&(A||B)
```

Dastur kodi:

```
#include <iostream>
using namespace std;
int main()
{
    bool A=true, B=true, C=false, D=true, Natija;
    Natija=!((A&&B)||(C&&D))&&(A||B);
    cout  <<  "!((A&&B)||(C&&D))&&(A||B)\n"<<Natija  <<
endl;
    return 0;}
```

Dastur interfeysi:



Foydalanilgan adabiyotlar ro’yxati.

1. **Aslonov. K. “C++ tilidan qo’llanma” Toshkent 2010.**
2. **Madraximov Sh. F, Gaynazarov S. M, “C++ dasturlash tili asoslari” Toshkent 2009.**
3. **J. Axmadaliyev. R. Holdarboyev, Uslubiy qo’llanma “C++ dasturlash tili” Andijon -2015**

Foydalanilgan saytlar ro’yxati.

1. **www.Google.uz**
2. **www.Ziyonet.uz**
3. **www.Arxiv.uz**
4. **www.kutubxona.uz**