O’ZBEKISTON RESPUBLIKASI ALOQA,
AXBOROTLASHTIRISH VA TELEKOMUNIKATSIYA DAVLAT QO’MITASI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
FARG’ONA FILIALI

”Axborot texnologiyalari” kafedrasi

”C++ da dasturlash”
fanidan

KURS ISHI

Bajardi: 611-12 Guruh talabasi
Turg’unov Hayotbek

Qabul qildi: Bazarbayev M.
Hashimov A

Farg’ona-2015

Mavzu: Shart operatori imkoniyatlari

Reja

1. **Kirish**
   a) **Dsturlash haqida qisqacha ma’lumot**
2. **Asosiy qism.**

a) Dasturlash tillari imkoniyatlari

b) C++ dasturlash tili haqida.

c) Shart operatori imkoniyatlari

d) Massivlar haqida.

3. **Dasturiy qism**
   a) **Dastur vazifasi.**
   b) **Dastur ko’rinishi.**
   c) **Dasturni ishga tushirish**
4. **Xulosa**
5. **Foydalanilgan adabiyotlar.**

Kirish

Dasturlash texnologiyasi bo’yicha qisqacha ma’lumot

Dasturlash– dastur yaratishga yo’naltirilgan amallar sohasidir.

Dastur – bu qo’yilgan masalani yechishga olib keluvchi kompyuter buyruqlarining ketma-ketligi.

Ilova – masalaning yechilishini kompyuterda dasturiy amalga oshirishdir.

Dasturiy tahminot (DT)) – dasturiy maxsulotlar va ularga texnik xujjatlardir.

Dasturiy maxsulot (DM) – mahlum masalani amalga oshirish uchun mo’ljallangan o’zoro bog’liq dasturlar kompleksi.

Dasturiy tahminot sifatining xarakteristikalari:

- Xujjatlashtirilganlik;
- Effektivlik;
- foydalanishga qulayligi;
- mobilligi;
- narxi va boshqalar.

Dastur quyidagi talablarga javob berishi kerak:

1. Texnik topshiriqqa asosan ishlashi;
2. Tezkorlik va xotira bo’yicha effektivlik;
3. Ishlatish va takomillashtirish uchun qulay bo’lishi;
4. Xatoliklarni aniqlashga moslashgan bo’lishi va boshqalar.

Translyator dasturni dasturlash tilidan mashina kodlariga o’tkazib beradi.

Translyatorlarning turlari:

- Inter’retator – satrma–satr o’tkazadi va bajaradi. Interpretator – sekin ishlaydi lekin sozlashga qulay.
- Kompilyator – dasturni to’lig’icha o’tkazadi va keyin bajaradi. Kompilyator – sozlashga qulay emas, lekin tayyor dasturni tez bajaradi

Algoritm va dasturlarni avtomatlashtirilmagan loyihalash tarkibi murakkab bo’lmagan nisbatan kichik dasturiy maxsulotlarni ishlab chiqishda ishlatiladi.

Avtomatlashtirilgan loyihalash katta firmalarda dasturiy maxsulotlarning ayrim klasslarini ishlab chiquvchilar kollektivlari tomonidan қo’llaniladi.

Algoritmlar va va dasturlarni loyihalash dasturiy maxsulotlarning ҳayot tsiklidagi asosiy bosqichlardan biridir.

Avtomatlashtirilgan loyihalash - loyihalash ishlariga ketadigan sarflarni kamaytirish, ularni bajarish muddatlarini qisqartirish, har xil loyihalarda ko’plab marta nusha olish mumkin bo’lgan tayyor ishlanmalardan foydalanish va dasturiy maxsulot ishlab chiqaruvchilarning katta kollektivining ishini muvofiqlashtirish zarurligi tufayli yuzaga kelgan

Tarkibiy loyihalash – loyihalash masalasini maqsadga yo’naltirilgan holda ayrim tashkil etuvchilarga bo’lib chiqishdir..

Tarkibiy loyihalash quyidagilarni o’z ichiga oladi:

- yuqoridan pastga yo’naltirilgan loyihalash;
- modulli dasturlash;
- tarkibiy dasturlash.

Yuqoridan pastga ypnaltirilgan loyihalash metodi mahlumotlarni qayta ishlashning umumiy funktsiyasini sodda funktsional elementlarga bo'lib chiqishni ko'zda tutadi.

Tarkibiy dasturlash dasturiy maxsulotning modulli tarkibiga va asosiy algoritmik tarkiblarga asoslangan.

Obyektga yo'naltirilgan loyihalash quyidagilarga asoslangan:

- obyektlarning klasslarini ajratish;
- obhektlarning xossalari va ularni o'zgartirish (qayta ishlash) metodlari;
- klasslarning ierarxiyasini hosil qilish, obyektlarning xossalari va ularni qayta ishlash metodlarini meros qilish.

Har bir oyhekt mahslumotlar va ularni qayta ishlash dasturini o'z ichiga oladi va mahlum klassga mansub bo'ladi.

Obyektga yo'naltirilgan loyihalashning asosiy maqsadi yuoridan pastga loyihalashning quyidagi kamchiliklarini bartaraf qilishdir:

- mahlumotlar strukturalariga ehtiborning yetarli emasligi;
- mahlumotlar strukturalari bilan ularni qayta ishlash jarayonlari orasidagi bog'lanishning kuchsiz bog'langanligi.

Obyektga yo'naltirilgan loyihalash bir-biriga bog'langan ko'plab obyektlarni o'z ichiga oluvchi dasturlarning xajmini va ularni tayyorlashga ketadigan mehnat sarflarini keskin kamaytirish imkoniyatini beradi.

Dasturiy maxsulotlarni yaratish bosqichlari

1. Dasturlash uchun texnik topshiriqni tuzish.

Ushbu bosqichda quyidagilar bajariladi:

operatsion sistemaning turi (OS - MS DOS, Windows, Windows NT) tanlanadi,
dastur ishlashining tarmoq varianti zarurligi ko'rib chiqiladi,
dastur tuzishning zarurligi aniqlanadi,
masalani yechish usuli tanlanadi va algoritmi tuziladi,
foydalanuvchining interfeysiga talablar aniqlanadi.

Texnik loyiha

Ushbu bosqichda quyidagilar bajariladi:

Mahlumotlarni qayta ishlashning batafsil algoritmi tuziladi,
Dasturiy tahminotning tarkibi aniqlanadi,
Dasaturiy maxsulotning ayrim dasturiy modullardan tashkil topgan ichki tarkibi ishlab chiqiladi,
Dasturiy vositalarni ishlab chiqish vositalari tanlanadi.

Ishchi xujjatlar (ishchi loyiha).

Ushbu bosqichda quyidagilar amalga oshiriladi:

Dasturiy modullar ishlab chiqiladi,
Dastur tayyorlanadi,
Dasturiy maxsulot sozlanadi,
Dasturiy modullar va vositalarning ishga yaroqligi sinab ko'riladi,
Dasturiy maxsulotning to'shirikka mos kelishini tekshirib ko'rish uchun nazorat misoli tayyorlanadi.

Dasturiy maxsulotlarning tuzilishi

Dasturiy maxsulot o’zoro bog’langan dasturiy modullardan tashkil topgan bo’ladi.

Modul – dasturning mustaqil qismi bo’lib u boshqa modullardan avtonom ravishda berilgan funktsiyalarni tahminlaydi va mahlum vazifaga ega bo’ladi.

Modullarning quyidagi turlarini ko’rsatish mumkin:

bosh modul – dasturiy maxsulotni ishga tushirishni boshqaradi;

boshqaruvchi modul – boshqa modullarni chaqirishni tahminlaydi;

ishchi modullar - qayta ishlash funktsiyalarini bajaradi;

servis modullar va bibliotekalar – xizmat ko’rsatish funktsiyalarini amalga oshiradi.

Har bir modul mustaqil saqlanuvchi fayl ko’rinishida bo’ladi.

Dasturning ichki tuzilishga ega bo’lishi uni loyihalash, dasturlash, sozlash va o’zgartirishlar kiritish uchun qulay bo’lishini tahminlaydi.

Dasturlash usullari dasturni o’qilishini qulayligi bilan o’zaro bog’liq. Usullar deganda tajribali dastur tuzuvchilar to’g’ri natijali, foydalanishda va o’qishda qulay bo’lgan dasturlarni tuzishda foydalanadigan usul va metodlar nazarda tutiladi.

Dasturlash usullari dasturni o’qilishini qulayligi bilan o’zaro bog’liq. Usullar deganda tajribali dastur tuzuvchilar to’g’ri natijali, foydalanishda qulay bo’lgan dasturlarni tuzishda foydalanadigan usul va metodlar nazarda tutiladi.

Dastur usulining eng yaxshi qoidasi - bu tajribali dasturchilar o’rtasidagi kelishishlardir. CHunki bitta tajribali dasturchiga nisbatan 2 ta yoki 3 ta tajribali dasturchilar tuzgan dastur yaxshiroqdir.

Dastur shunday tuzilishi kerakki, uni birinchi navbatda mashina yoki EHM emas, balki inson o’qib tushunsin. CHunki dastur insonlarga dasturlarni tuzishda, shakllantirishda va ularni qo’llashda kerak bo’ladi.

Dastur - bu keyinchalik ishlatishga va takomillashtirishga mo’ljallangan hujjat, algoritmlarni kodlashtirish uchun o’quv materiali va hokazolardir.

Demak, dasturlash tillari bizga o’qishda qulay bo’lgan dasturni yaratishni taominlab berishi kerak. Dastur iloji boricha algoritimning tuzilishini va mantig’ini bizga tushunarliroq qilib yetkazib berishi kerak.

Asosiy qism.
Dasturlash tillari imkoniyatlari

Usullarni standartlash.

Albatta biz juda ko’p joylarda va jarayonlarda standartlarga asosan ish olib boramiz. Lekin dasturlash usullarida bunga qarshi bir necha fikrlar aytilgan. Ulardan birini keltirib o’tamiz: dasturlash usuli - bu shaxsiy fikr va didga bog’liqdir. SHuning uchun unga hech qanday chegaralanishlar qo’yilmasligi kerak.

Usullarni standartlash quyidagilardan iborat:

Agar biron - bir narsani yaratishda bir necha usul mavjud bo’lib, tanlash sizga bog’liq bo’lsa, unda siz 1 ta usulni tanlab, oxirigacha shu usuldan foydalaning.

Ammo standartlash jarayonining o’ziga xos kamchiliklari bor:

standartlashni qo’llaganda kelajakda dasturlash yo’llarini sekinlashtiribgina qolmay, balki foydalanishda qulay va ortiqcha chegaralarga ega bo’lishi mumkin.

CHunki ular yangidan - yangi vazifalarni bajarishga sizning aqlingizni bog’lab qo’yadi.

"O’qilmaydigan " programmalar.

Dastur ishlab chiqilgandan so’ng uni va javobgar shaxs, bevosita loyihalovchi tasdiqlaydi. Aniqlikka talab standart usuli so’rab o’tirilmaydi. Faqat ishlaydimi ?, ishlamasa qachon ishlaydi ? . Bunday dasturlar kirayotgan maolumotni qabul qiladi va chiqaruvchi maolumotni chiqaradi. Xuddi “oldim, berdim”. SHuning uchun bu dasturlar katta o’zgarishlar sodir bo’lgunga qadar kerakdir. Bu o’zgarishlardan so’ng bunday dasturlar tashlab yuborilib yangilari paydo bo’ladi. CHunki eskisini takomillashtirishdan ko’ra yangisini loyihalash osonroq. SHuning uchun bunday dasturlarga ikkinchi shaxslar qo’l qo’yishi va dasturning usuli va o’lchami, xamda ishlashiga javob berishlari maqsadga muvofikdir.

Qavslar. (Skobki).

Qavslardan to’g’ri foydalanish dasturni o’qishni osonlashtiradi, chunki operatsiyalar bajarilayotgan vaqtda priariatatsiya maolum bo’ladi va bunda 1dasturchi bir necha qavslardan foydalanishi mumkin, lekin bunda dasturni o’qish va tuzatish qiyinlashadi.

Asosiy qoidalardan biron - gumonli joylarda qavs qo’ying. Bu esa dasturni tushunarli qilib, xato qilishni oldini oladi.

Qator boshidan o’tish.

Operatorlarni yozishda ularni o’rtasidagi bog’liqni ko’rsatish uchun bir satrlar boshidan o’tishlar bajariladi. To’g’ri bajarilgan o’tishlar dasturni tarkibini ko’rsatadi. Bu tekstni tushunarli tilda paragrflarga guruhiy so’zlarni mantiqiy bog’lab bo’ladi. TSikllar - o’tishlarida foydalanishdagi oddiy xolat. O’tishlarda faqat tsikllarda emas, balki buyruqlarni qulay guruhlashda ham kiritiladi.

Operator IF yordamida aniqlanadigan operatorlarga qator boshidan bir o’tishlar bilan yeziladi. Kiritish - chiqarish operatoriga yozishda ham o’tishlardan foydalaniladi. teng xukukli fayllarni taoriflashda ham o’tishlardan foydalaning.

Agar dasturchi tuzayotgan dasturni biron - bir boshqa dasturchi yoki kishini o’qishini bilsa, u shu dasturni kodlashdagi fikri keskin o’zgaradi. U dasturni programmalarga bo’lib, izohlar bilan to’ldiradi. Natijada dastur aniq va tushunarli bo’ladi.

Bir xilgi loyihalarda dasturni oddiyligini taominlovchi metod dasturni nazaorat qiluvchi sistemadan foydalaniladi.

Bu sistemada dastur loyihalanaetgan paytda kamida 2 ta dasturchidan foydalaniladi. Birinchisi dastur tuzsa. ikkinchisi shu dasturni tuzish uchun ko’rib, o’qib chiqadi. Bu sistemani 2 ta yutug’i bor. Birinchidan u oddiy va tushunarli dastur ishlab chikishni taominlaydi. Ikkinchidan, birinchi dasturchi loyihani tugata olmasa, ikkinchi dasturchi uni o’rniga tugatib qo’yadi.

Dasturni standartlarini bajarishlikka baxolash metodi ham shunga o’xshash metoddir.

Dasturchiga maslahatlar.

- Keragidan ko’proq va yana ko’proq izoh bering.
- Kiritish izohlaridan foydalaning.
- Katta dasturlarda mundarija qiling.
- Izohlar boshini aylantirmasdan qo’shimcha maolumotlar bersin.
- Noto’g’ri izohdan ko’ra, uning yukligi yaxshi.
- Dasturni oson uqilishi uchun oraliqlar bajaring.
- Fayllarni nomlashda prefiks va suffikslardan foydalaning.
- Alvafit buyicha ro’yxatlar tuzing.
- Dastur tarkibini aniqlash uchun o’tishlar xosil qiling.
- Maolumot tarkibini aniqlashda ham erinmang. Qator boshidan o’tishlardan foydalaning.

Dasturni loyihalash bosqichi dasturlash usuliga, ishonchligiga, sozlashga, foydalanish tarkibiga, va xokazolarga tahsir ko’rsatadi.

Dasturning loyihalashning oson kechishi - bu dasturni oson o’qilishi uchun qo’yilgan birinchi qadam. Kodlashtirish oddiyroq bo’lsin.

Ortiqcha shakllantirilgan dastur sozlashda yoki modifikatsiyalashda kimmatga tushadi. Dasturdan faqat uning muallifi emas balki boshqa shaxslar ham foydalanib, ham sozlashda kiynalmasligi kerak. Dasturning tarkibi uning mantig’ini ko’rsatishi shart, kodlarning doimiyligi dasturni yanada osonlashtiradi. Masalan: dasturni o’zgartiruvchilar doimo bir xil ravishda foydalanish kerak. "1" belgisi yoqilgan. "0" belgisi o’chirilgan. Bundan tashkari, dasturdagi har bir jadval bir xil taosvirlansin.

Masalani qo’yilishi va tarkibi.

Loyihalash vaqtida har - xil muammolarni yuzaga kelishini hisobga olish kerak. Asosan bu muammolar shu dasturni buyurtmachilarining aniq maqsadga ega bulmasliklaridan kelib chikadi. Foydalanuvchi noaniq bergan so’rovga aniq va to’g’ri javob beruvchi dastur - bu eng yuksak dasturdir. Vazifa yaxshi yoki yomon deb aniqlanishi mumkin. Yomon shakllantirilgan vazifaga aniq bir dasturni loyihalab bo’lmaydi.

Aniq belgilangan vazifa esa loyihalash davomida ortiqcha ishdan, vaqt sarflashdan ozod bo’ladi. Vazifa dasturning xajmiga qarab ixcham va tushunarli bo’lishi kerak. Agar dastur kichkina bo’lsa vazifa bir necha qatordan, katta bo’lsa butun - bir kitobdan iborat bo’lishi mumkin. Bu tavsiflar katta songa ega bo’lgan hujjatlarga asos bo’ladi. Keyinchalik vazifalarga kiritilgan o’zgartirishlar yoki qo’shimcha vazifalar dasturni bo’g’ibgina qolmay uni tugatishni umuman sekinlatib qo’yadi. Vazifani aniqlashda qo’yilgan lokaydsizliklar keyinchalik katta muammolar keltirib chiqaradi.

Algoritm tanlash.

Algoritmni to’g’ri tanlash bu dasturni to’g’ri va samarali bo’lishiga quyiladigan asosiy qoidadir. Bunda dasturni tilini va tarkibini to’g’ri tanlash mumkin. SHunday qilib yaxshi algoritm kerakli, lekin dastur uchun yetarli sharoit emas.

Birinchidan - miyaga to’g’ri kelgan algoritmga dastur tuzmaslik kerak. Bir necha variantlarni ko’rib chiqib, ularning ichidan eng yaxshisini tanlash zarur.

Maolumotlarni tavsiflash.

Yaxshi tavsiflangan maolumotlar dasturni ancha qiskartiradi. Demak, maolumotlar massivni tashkillashda zarur bo'lgandagina qo'llash kerak. Boshqacha usulida tayanch va ko'rsatkichlardan foydalanish kerak. Maolumotlar haqida ko'rilayotgan vazifaga mos xolda taosurotga ega bo'lish kerak.

Hisobda har bir maolumotning tarkibini hamma tilga ko'chirish mumkin. Lekin siz mo'ljallangan maolumot tarkibini mujassamlagan dasturlash tilida foydalanishingiz makulroq.

Yetarli murakkablikdagi strukturasi modullarini tuzish.

Programmani modullarga bo'lish va yuqoridan pastga loyixalash murakkab strukturalarni nazorat qilish imkoniyatini berib, uni darajasini belgilaydi.
Modullarga bo'lish - bu murakkablik darajasini boshqarish vositasidir. Agar pastgi kismlaridagi informatsiyalarga talab bulmasa ular qoldirilishi mumkin.
Birinchidan modullar o'zgaruvchilar to'plamini bo'lishga sharoit yaratadi.
Buning okibatida modullar zarur bulmagan o'zgaruvchilardan xalos bo'lib, kuzatilishi mumkin bo'lgan xatoliklardan maolum darajada ximoyalanadi.
Bundan tashkari, modullar qanchalik katta bo'lsa, uni taxlil qilish shunchalik qiyin. SHunday qilib modullarni kiritilishi programmada bo'lishi mumkin bo'lgan yo'llar sonini kamaytiradi va uni boshqarishni yengillashtiradi.

Programmalash tilini tanlash.

Ko'pincha dasturlash tilini hisoblash sistemasi yoki dasturchi foydalanuvchiga asosan tanlangan bo'ladi.

Agar dasturchi tanlashga ega bo'lib qolsa u albatda vazifaga mos eng yuqori dasturlash tilini tanlashi kerak. Agar tanlangan dasturlash tili berilgan vazifaga mos kelmasa dasturlashda va sozlashda muammolar kelib chiqishi mumkin. Tanlangan til loyihalashga ham ancha taosir ko'rsatadi.

Dialog rejim

Ko'pgina dasturiy maxsulotlar dialog rejimida ishlaydi.
Dialog rejimda foydalanuvchi tomonidan funktsiyalar ishga tushiriladi, obyektlarning xossalari o'zgartiridadi va h.k.
Menyular ierarxik bo'lishi va ost menyularni o'z ichiga olishi mumkin.

Dialog tizimlarning tarkibi:

Menyu – foydalanuvchiga ro'yhatdan alternativ funktsiyalarni tanlash imkoniyatini beradi;

Menyuda ichki (ost) menyu xam bo'lishi mumkin.

So'rov-javob amali– ro'yxatdan tanlab olinishi mumkin bo'lgan qiymatlar yoki Ha/Yo'q turdagi javoblar;

Format bo'yicha so'rov – kalit so'zlar yoki iboralar yordamida.

Dialog jarayoni yaratilgan stsenariyaga asosan boshqariladi. Dialog jarayoni uchun quyidagilar aniqlanadi:

Dialogning boshlanish momenti;

Dialogni boshlovchi– foydalanuvchi yoki dasturiy maxsulot;

Dialogning parametrlari va mazmuni – axborotlar,
menyuning tarkibi sostav, ekran formalari;
Dasturiy maxsulotning dialog tugashiga reaktsiyasi.

Dialog jarayonni va foydalanuvchining interfeysini yaratish uchun dasturlarni ishlab chiqishning obyektga yo'naltirilgan vositalaridan foydalaniladi. Ularning tarkibiga quyidagilar kiradi:
Menyu qurgich (bosh menyu va unga biriktirilgan ost menyularni yaratish uchun);
Ekran formalarining konstruktori (ekran orqali kiritish formatlarini ishlab chiqish va mahlumotlarni tahrirlash uchun).
Dialog oynada quyidagi boshqarish elementlari bo'lishi mumkin:
Axborot matnlari;
Foydalanuvchining axborotlarini kiritish uchun maydonlar;
Tanlash uchun mumkin bo'lgan alternativlar ro'yxatlari;
tugmalar, ulab uzgichlar va h.k.

Foydalanuvchining grafik interfeysi

Grafik interfeys zamonaviy dasturiy maxsulotlar uchun zaruriy komponent bo'lib hisoblanadi va ochiluvchi menyular ko'rinishida quriladi. Grafik interfeys yordamida foydalanuvchi ekran formalari bilan ishlaydi. Ekran formalarida boshqarish elementlari, asboblar panellari va qayta ishlash komandalari bo'lishi mumkin.

Menyu dastur bajaradigan funktsiyalarga mos holda tushunarli punktlarga ega bo'lishi kerak.
Tarkibiy loyihalash va dasturlash

Tarkibiy loyihalash quyidagilarni o'z ichiga oladi:
yuqoridan pastga loyipalash,
modulli dasturlash,
tarkibiy dasturlash.

Yuqoridan pastga loyihalashda axborotlarni qayta ishlash funktsiyasi yuqoridan pastga yo'nalishda sodda funktsional elementlarga bo'lib chiqiladi. Natijada ilova algoritmining funktsional tuzilishi hosil qilinadi. Unda quyidagilar o'z aksini topadi:
maqsad-ost maqsad;
qo'yilgan maqsadni amalga oshirishni tahminlaydigan ilovalarning tarkibi va o'zaro bog'lanishlari;
axborotlarni qayta ishlash funktsiyalari.

Modul mantiqiy bog'langan elementlar majmuidir. Modul boshqa modullar yoki dasturlar foydalanishi uchun mo'ljallangan bo'ladi.
Modul tayyor dasturlarni saqlash uchun mo'ljallangan.

Modulning o'zi bajariluvchi dastur bo'lmaydi. Uning obYektlarini boshqa dasturlar (protseduralar, funktsiyalar) ishlatadi.

Modulda bitta kirish va bitta chiqish bo'ladi. Kirishiga boshlang'ich mahlumotlar beriladi, modul ularni qayta ishlaydi va natijalarni chiqishga uzatadi, ya'ni quyidagi printsipni amalga oshiradi

IpO (Input – process – Output) – kirish-jarayon-chiqish;

Har bir modul o’zidan yuqoridagi modul tomonidan chaqiriladi va ishini tugatgandan so’ng boshqarishni o’zini chaqirgan modulga topshiradi.

C++ dasturlash tili haqida

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. Yan i 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 Laboratories da, 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 arhitekturasiga bog'langan tildir. Lekin yahshi 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 Xerox ning 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++ funksiya va ob'ektlarning juda boy kutubhonasiga ega. Yani C++ da dasturlashni o'rganish ikki qismga bo'linadi. Birinchisi bu C++ ni o'zini o'rganish, ikkinchisi esa C++ ning standart kutubhonasidagi tayyor ob'ekt/funksiyalarni qo'llashni o'rganishdir.

C++ DA DASTURLASHNING ASOSIY QISMLARI

C++ sistemasi asosan quyidagi qismlardan iborat. Bular dasturni yozish redaktori, C++ tili va standart kutubhonalardir. C++ dasturi ma'lum bir fazalardan o'tadi. Birinchisi dasturni yozish va tahrirlash, ikkinchisi preprosessor amallarini bajarish, kompilyatsiya, kutubhonalardagi ob'ekt va funksiyalarni dastur bilan bog'lash (link), hotiraga yuklash (load) va bajarish (execute).

DASTUR IJRO STRUKTURALARI

Asosan dasturdagi ifodalar ketma-ket, navbatiga ko'ra ijro etiladi. Gohida bir shart bajarilishiga ko'ra, ijro boshqa bir ifodaga o'tadi. Navbatdagi emas, dasturning boshqa yerida joylashgan ifoda bajariladi. Yani sakrash yoki ijro ko'chishi vujudga keladi.

60-chi yillarga kelib, dasturlardagi ko'pchilik hatolar aynan shu ijro ko'chishlarining rejasiz ishlatilishidan kelib chiqishi ma'lum bo'ldi. Bunda eng katta aybdor deb bu ko'shishlarni amalga oshiruvchi goto (...ga bor) ifodasi belgilandi. goto dastur ijrosini deyarli istalgan yerga ko'chirib yuborishi mumkin. Bu esa programmani o'qishni va uning strukturasini murakkablashtirib yuboradi. Shu sababli "strukturali dasturlash" atamasi "goto ni yo'q qilish" bilan tenglashtirilardi. Shuni aytib o'tish kerakki, goto kabi shartsiz sakrash amallarini bajaruvchi ifodalar boshqa dasturlash tillarida ham bor.

Tadqiqotlar shuni ko'rsatdiki, istalgan programma goto siz yozilishi mumkin ekan. goto siz yozish uslubi strukturali dasturlash deb nom oldi. Va bunday dastur yozish metodi katta iqtisodiy samara beradi.

Strukturali dasturlash asosi shundan iboratki, har bir programma faqatgina uch hil boshqaruv strukturalaridan iboratdir.Bular ifodalarni ketma-ket ijro etish strukturasi (sequence structure), tanlash strukturasi (selection structure) va amalni qayta ijro etish strukturasidir (repetition structure).

Ifodalarni ketma-ket ijro etish strukturasi C++ tomonidan ta'minlanadi. Normal sharoitda C++ ifodalari dasturdagi navbatiga ko'ra bajariladi.

Tanlash buyruqlari uchtadir. Bular if, if/else va switch dir. Qayta ijro etish buyruqlari gurugiga ham uchta a'zo bor, bular while, do/while va for. Bularni har birini keyinroq tahlil qilib chiqamiz.

Yuqoridagi buyruqlar nomlari C++ dasturlash tilining mahsus so'zlaridir. Dasturchi bu so'zlarni o'zgaruvchi yoki funksiyalar nomi sifatida qo'llashi ta'qiqlanadi. Quyida C++ ning ajratilgan so'zlarining to'liq ro'yhati berilgan.

C++ va C ga tegishli:

auto do goto signed unsigned
break double if sizeof void

case else int static volatile
char enum long struct while
const extern register switch
continue float return typedef
default for short union
Faqat C++ ga qarashli:
asm explicit operator this virtual
bool false private throw wchar_t
catch friend protected true
class inline public try
const_cast mutable reinterpret_cast typeid
delete namespace static_cast typename
dynamic_cast new template using
C++ dagi yetita boshqaruv strukturasini aytib o'tdik. Ular bittagina boshlanish nuqtasiga va bittagina chiqish nuqtasiga egadirlar. Demak biz bu dastur bo'laklarini ketma-ket ulab ketishimiz mumkin. Boshqaruv strukturalarining bu kabi ulanishini devorning g'ishtlarini ustma-ust qalashga ham taqqoslasak bo'ladi. Yoki biz bu bloklarni bir-birining ichiga joylashtirishimiz mumkin. Bu kabi qo'llashish ikkinchi uslub bo'ladi. Mana shu ikki yo'l bilan bog'langan yetita blok yordamida biz istalgan dasturimizni yoza olamiz.

Shart operatorlari

if STRUKTURASI

Biz shartga ko'ra bir necha harakat yo'lidan bittasini tanlaymiz. Misol uchun agar bolaning yoshi 7 ga teng yoki katta bo'lsa u maktabga borishi mumkin bo'lsin. Buni C++ da if ni qo'llab yozaylik.

```
if (yosh >= 7)
maktab();
```

Bu yerda shart bajarilishi yoki bajarilmasligi mumkin. Agar yosh o'zgaruvchisi 7 ga teng yoki undan katta bo'lsa shart bajariladi va maktab() funksiyasi chaqiriladi. Bu holat true (to'g'ri) deyiladi. Agar yosh 7 dan kichik bo'lsa, maktab() tashlab o'tiladi. Yani false (noto'g'ri) holat yuzaga keladi. Biz shart qismini mantiqiy operatorlarga asoslanganligini ko'rib chiqqan edik. Aslida esa shartdagi ifodaning ko'rinishi muhim emas - agar ifodani nolga keltirish mumkin bo'lsa false bo'ladi, noldan farqli javob bo'lsa, musbatmi, manfiymi, true holat paydo bo'ladi va shart bajariladi. Bunga qo'shimcha qilib o'tish kerakki, C++ da mahsus bool tipi mavjud. Bu tipdagi o'zgaruvchilarning yordamida bul (mantiqiy) arifmetikasini amalga oshirish mumkin. bool o'zgaruvchilar faqat true yoki false qiymatlarini olishlari mumkin.

if/else STRUKTURASI

if ni qo'llaganimizda ifoda faqat shart haqiqat bo'lgandagina bajariladi, aks holda tashlanib o'tiladi. if/else yordamida esa shart bajarilmaganda

(false natija chiqqanda) else orqali boshqa bir yo'ldan borishni belgilash mumkin. Misolimizni takomillashtirsak. Bola 7 yosh yoki undan katta bo'lsa maktabga, 7 dan kichkina bo'lsa bog'chaga borsin.

```
if (yosh >= 7)
maktab(); //nuqta-vergul majburiydir
else
bogcha();
```

Yuqorida if ga tegishli bo'lgan blok bitta ifodadan (maktab()) iborat. Shu sababli nuqta-vergul qo'yilishi shart. Buni aytib o'tishimizning sababi, masal Pascalda hech narsa qo'yilmasligi shart.

C++ da bitta ifosa turgan joyga ifodalar guruhini {} qavslarda olingan holda qo'ysa bo'ladi. Masalan:

```
if (yosh >= 7){
cout << "Maktabga!\n";
maktab();
}
else{
cout << "Bog'chaga!\n";
bogcha();
}
```

Aslida har doim {} qavslarni qo'yish yahshi odat hisoblanadi; keyinchalik bir ifoda turgan joyga qo'shimcha qilinganda qavslardan biri unutilib qolmaydi.

Strukrurali dasturlashning yana bir harakterli joyi shundaki tabulyatsiya, bo'sh joy va yangi satrlar ko'p qo'llaniladi. Bu programmani o'qishni osonlashtirish uchun qilinadi. C++ uchun bo'sh joyning hech ahamiyati yo'q, lekin dasturni tahrir qilayatgan odamga buyruqlar guruhini, bloklarni tabulyatsiya yordamida ajratib bersak, unga katta yordam bo'ladi. Yuqoridagini quyidagicha ham yozish mumkin:

```
if(yosh>=7){cout<<"Maktabga!\n";maktab()}else{cout<<"Bog'chaga!\n";bogcha()
};
```

Biroq buni o'qish ancha murakkab ishdir.

C++ da if/else strukturasiga o'hshash ?: shart operatori (conditional operator) ham bordir. Bu C++ ning bittagina uchta argument oluvchi operatori. Uch operand va shart operatori shart ifodasini beradi. Birinchi operand orqali shartimizni beramiz. Ikkinchi argument shart true (haqiqat) bo'lib chiqqandagi butun shart ifodasining javob qiymatidir. Uchinchi operand shartimiz bajarilmay (false) qolgandagi butun shart ifodasining qiymatidir. Masalan:

```
bool bayroq;
int yosh = 10;
bayroq = ( yosh >= 7 ? true : false );
```

Agar yosh 7 ga teng yoki katta bo'lsa, bool tipidagi o'zgaruvchimiz true qiymatini oladi, aks taqdirda false bo'ladi. Shart operatori qavslar ichida bo'lishi zarur, chunki uning kuchi katta emas. Javob qiymatlar bajariladigan

funksiyalar ham bo'lishi mumkin:
yosh >= 7 ? maktab() : bogcha();
if/else strukturalarini bir-birining ichida yozishimiz mumkin. Bunda ular bir-biriga ulanib ketadi. Misol uchun tezlikning kattaligiga qarab jarimani belgilab beruvchi blokni yozaylik.

```
if (tezlik > 120)
cout << "Jarima 100 so'm";
else if (tezlik > 100)
cout << "Jarima 70 so'm";
else if (tezlik > 85)
cout << "Jarima 30 so'm";
else
cout << "Tezlik normada";
```

Agar tezlik 120 dan katta bo'lsa birinchi if/else strukturasining haqiqat sharti bajariladi. Va bu holda albatta tezlik o'zgaruvchimizning qiymati ikkinchi va uchinchi if/else imizni ham qoniqtiradi. Lekin solishtirish ulargacha bormaydi, chunki ular birinchi if/else ning else qismida, yani noto'g'ri javob qismida joylashgandir. Solishtirish birinchi if/else da tugashi (aynan shu misolda) tanlash amalini tezlashtiradi. Yani bir-biriga bog'liq if/else lar alohida if strukturalari blokidan tezroq bajarilishi mumkin, chunki birinchi holda if/else blokidan vaqtliroq chiqish imkoni bor. Shu sababli ich-ichiga kirgan if/else lar guruhida true bo'lish imkoni ko'proq bo'lgan shartlarni oldinroq tekshirish kerak.
switch STRUKTURASI
if-else-if yordami bilan bir necha shartni test qilishimiz mumkin. Lekin bunday yozuv nisbatan o'qishga qiyin va ko'rinishi qo'pol bo'ladi. Agar shart ifoda butun son tipida bo'lsa yoki bu tipga keltirilishi mumkin bo'lsa, biz switch (tanlash) ifodalarini ishlata olamiz.
switch strukturasi bir necha case etiketlaridan (label) va majburiy bo'lmagan default etiketidan iboratdir. Etiket bu bir nomdir. U dasturnig bir nuqtasidaga qo'yiladi. Programmaning boshqa yeridan ushbu etiketga o'tishni bajarish mumkin. O'tish yoki sakrash goto bilan amalga oshiriladi, switch blokida ham qo'llaniladi.
5 lik sistemadagi bahoni so'zlik bahoga o'tqizadigan blokni yozaylik.

```
int baho;
baho = 4;
switch (baho) {
case 5: cout << "A'lo";
break;
case 4: cout << "Yahshi";
break;
case 3: cout << "Qoniqarli";
break;
```

```
case 2:
case 1: cout << "A'lo";
break;
default: cout << "Baho hato kiritildi!";
break;
}
```

switch ga kirgan o'zgaruvchi (yuqorigi misolda baho) har bir case etiketlarining qiymatlari bilan solishtirilib chiqiladi. Solishtirish yuqoridan pastga bajariladi. Shartdagi qiymat etiketdagi qiymat bilan teng bo'lib chiqqanda ushbu case ga tegishli ifoda yoki ifodalar bloki bajariladi. So'ng break (buzmoq, tugatmoq) sakrash buyrug'i bilan switch ning tanasidan chiqiladi. Agar break qo'yilmasa, keyingi etiketlar qiymatlari bilan solishtirish bajarilmasdan ularga tegishli ifodalar ijro ko'raveradi. Bu albatta biz istamaydigan narsa. default etiketi majburiy emas. Lekin shart chegaradan tashqarida bo'lgan qiymatda ega bo'lgan hollarni diagnostika qilish uchun kerak bo'ladi.

case va etiket orasida bo'sh joy qoldirish shartdir. Chunki, masalan, case 4: ni case4: deb yozish oddiy etiketni vujudga keltiradi, bunda sharti test qilinayotgan ifoda 4 bilan solishtirilmay o'tiladi.

MASSIVLAR

Bu qismda dasturdagi ma'lumot strukturalari bilan tanishishni boshlaymiz. Dasturda ikki asosiy tur ma'lumot strukturalari mavjuddir. Birinchisi statik, ikkinchisi dinamikdir. Statik deganimizda hotirada egallagan joyi o'zgarmas, dastur boshida beriladigan strukturalarni nazarda tutamiz. Dinamik ma'lumot tiplari dastur davomida o'z hajmini, egallagan hotirasini o'zgartirishi mumkin.

Agar struktura bir hil kattalikdagi tiplardan tuzilgan bo'lsa, uning nomi massiv (array) deyiladi. Massivlar dasturlashda eng ko'p qo'laniladigan ma'lumot tiplaridir. Bundan tashqari strukturalar bir necha farqli tipdagi o'zgaruvchilardan tashkil topgan bo'lishi mumkin. Buni biz klas (Pascalda record) deymiz. Masalan bunday strukturamiz ichida odam ismi va yoshi bo'lishi mumkin.

Bu bo'limda biz massivlar bilan yaqindan tanishib o'tamiz. Bu bo'limdagi massivlarimiz C uslubidagi, pointerlarga (ko'rsatkichlarga) asoslan strukturalardir. Massivlarning boshqa ko'rinishlarini keyingi qismlarda o'tamiz.

Massivlar hotirada ketma-ket joylashgan, bir tipdagi o'zgaruvchilar guruhidir. Alohida bir o'zgaruvchini ko'rsatish uchun massiv nomi va kerakli o'zgaruvchi indeksini yozamiz. C/C++ dagi massivlardagi elementlar indeksi har doim noldan boshlanadi. Bizda char tipidagi m nomli massiv bor bo'lsin. Va uning 4 dona elementi mavjud bo'lsin. Shemada bunday ko'rsataylik:

m[0] -> 4
m[1] -> -44

m[2] -> 109
m[3] -> 23
Ko'rib turganimizdek, elementga murojaat qilish uchun massiv nomi va [] qavslar ichida element indeksi yoziladi. Bu yerda birinchi element qiymati 4, ikkinchi element - 1 nomerli indeksda -44 qiymatlari bor ekan. Ohirgi element indeksi n-1 bo'ladi (n - massiv elementlari soni).
[] qavslar ichidagi indeks butun son yoki butun songa olib keluvchi ifoda bo'lmog'i lozim. Masalan:
```
...
int k = 4, l = 2;
m[ k-l ] = 77; // m[2] = 77
m[3] *= 2; // m[3] = 46
double d = m[0] * 6; // d = 24
cout << m[1]; // Ekranda: -44
...
```
Massivlarni ishlatish uchun ularni e'lon qilish va kerak bo'lsa massiv elementlarini initsalizatsiya qilish kerak. Massiv e'lon qilinganda kompilyator elementlar soniga teng hajmda hotira ajratadi. Masalan yuqorida qo'llanilgan char tipidagi m massivini e'lon qilaylik.
```
char m[4];
```
Bu yerdagi 4 soni massivdagi elementlar miqdorini bildiradi. Bir necha massivni e'londa bersak ham bo'ladi:
```
int m1[4], m2[99], k, l = 0;
```
Massiv elementlari dastur davomida initsalizatsiya qilishimiz mumkin, yoki boshlang'ich qiymatlarni e'lon vaqtida, {} qavslar ichida ham bersak bo'ladi. {} qavslardagagi qiymatlar massiv initsalizaytsiya ro'yhati deyiladi.
```
int n[5] = {3, 5, -33, 5, 90};
```
Yuqorida birinchi elementning qiymati 3, ikkinchiniki 5 ... ohirgi beshinchi element qiymati esa 90 bo'ldi. Boshqa misol:
```
double array[10] = {0.0, 0.4, 3.55};
```
Bu yerdagi massiv tipi double bo'ldi. Ushbu massiv 10 ta elementdan iboratdir. {} qavslar ichida esa faqat boshlangich uchta element qiymatlari berildi. Bunday holda, qolgan elementlar avtomatik tarzda nolga tenglashtiriladi. Bu yerda aytib o'tishimiz kerakki, {} qavslar ichida berilgan boshlangish qiymatlar soni massivdagi elementlar sonidan katta bo'lsa, sintaksis hatosi vujudga keladi. Masalan:
```
char k[3] = {3, 4, 6, -66, 34, 90}; // Hato!
```
Uch elementdan iborat massivga 6 dona boshlangich qiymat berilyapti, bu hatodir. Boshqa misolni ko'rib chiqaylik:
```
int w[] = {3, 7, 90, 78};
```
w nomli massiv e'lon qilindi, lekin [] qavslar ichida massivdagi elementlar soni berilmadi. Bunday holda necha elementga joy ajratishni kompilyator {} qavslar ichidagi boshlangich qiymatlar miqdoriga qarab biladi. Demak,

yuqoridagi misolda w massivimiz 4 dona elementdan iborat bo'ladi.
E'lon davridagi massiv initsalizatsiya ro'yhati dastur ijrosi vaqtidagi initsalizatsiyadan ko'ra tezroq ishlaydigan mashina kodini vujudga keltiradi.
Bir misol keltiraylik.

```
// Massivlar bilan ishlash.
# include <iostream.h>
# include <iomanip.h>
const int massiv = 8; // massiv kattaligi uchun konstanta
int k[massiv];
char c[massiv] = {5,7,8,9,3,44,-33,0}; // massiv initsalizatsiya ro'yhati
int main()
{
for (int i = 0; i < massiv; i++) {
k[i] = i + 1; // dastur ichida inisalizatsiya
}
for (int j = 0; j < massiv; j++) {
cout << k[j]
<< setw(4)
<< c[j]
<< endl;
}
return (0);
}
```

Ekranda:

```
1 5
2 7
3 8
4 9
5 3
6 44
7 -33
8 0
```

Yuqorida <iomanip.h> faylini dasturimizga kiritdik. Bu e'lon faylida standart kirish/chiqish oqimlari bilan ishlaydigan buyruqlar berilgan. Dasturimizda qo'llanilgan setw() manipulyatori chiqish oqimiga berilayatgan ma'lumotlarning eng kichik kengligini belgilaydi, biz setw() parametrini 4 deb berdik, demak c[] massivi elementlari 4 harf kenglikda ekranga bosiladilar. Agar kenglik kamlik qilsa, u kattalashtiriladi, agar bo'sh joy qolsa, elementlar chapga yondashilib yoziladi. Biz <iomanip.h> va manipulyatorlarni keyinroq to'la ko'rib chiqamiz.
Misolimizda massiv nomli konstantani qo'lladik. Uning yordamida massiv chegaralarini va for strukturasidagi chegaraviy qiymatlarni berdik. Bunday o'zgarmasni qo'llash dasturda hatoga yo'l qo'yishni kamaytiradi. Massiv

chegarasi o'zgarganda, dasturning faqat bir joyiga o'zgarish kiritiladi.
Massiv hajmi e'lonida faqat const sifatli o'zgaruvchilar qo'llanilishi mumkin.
Massivlar bilan ishlaganda eng ko'p yo'l qoyiladigan hato bu massivga
0 dan kichkina va (n-1) dan (n: massivdagi elementlar soni) katta indeks bilan
murojaat qilishdir. Bunday hato dastur mantig'i hatosiga olib keladi.
Kompilyator bu turdagi hatolarni tekshirmaydi. Keyinroq o'zimiza yozgan
massiv klaslarida ushbu hatoni tekshiriladigan qilishimiz mumkin.
10 ta sonning tushish ehtimilini ko'rsaturvchi dastur yozaylik.

```
// Ehtimollar va massivlar
# include <iomanip.h>
# include <iostream.h>
# include <time.h>
# include <stdlib.h>
int main ()
{
const int massivHajmi = 10;
int m[massivHajmi] = {0}; // hamma 10 ta element
// 0 ga tenglashtirildi
srand( time(NULL) );
for(int i = 0; i < 1000; i++) {
++m[ rand() % 10 ];
}
for(int j = 0; j < massivHajmi; j++) {
cout << j << setw(4) << m[j] << endl;
}
return (0);
}
```

Ekranda:

```
0 96
1 89
2 111
3 97
4 107
5 91
6 100
7 118
8 99
9 92
```

Ko'rib turganimizdek, sonlarning tushish ehtimoli nisbatan tengdir. Albatta,
bu qiymatlar dasturning har yangi ishlashida o'zgaradi.

```
++m[ rand() % 10 ];
```

Yozuvi bilan biz massivning rand() % 10 indeksli elementini birga
oshirmoqdamiz. Bunda rand () % 10 ifodasidan chiqadigan qiymatlar [0;9] ichida

yotadi.
Satrlar, yani harflar ketma-ketligi ("Toshkent", "Yangi yilingiz bilan!"...) C/C++ da char tipidagi massivlar yordamida beriladi. Bunday satrlar bilan islovlar juda tez bajariladi. Chunki ortiqcha tekshirishlar bajarilmaydi. Bundan tashqari C++ da ancha rivojlangan String klasi mavjuddir, u oddiy char bilan berilgan satrlardan ko'ra qulayroqdir. Lekin ushbu klas ko'proq joy egallaydi va massivli satrlardan ko'ra sekinroq ishlaydi. String klasini keyingi qismlarda o'tamiz. Qolaversa, satrlar bilan ishlash uchun biz o'zimiz klas yoki struktura yozishimiz mumkin. C dan meros bo'lib qolgan satrlar ustida amallar bajarish uchun biz dasturimizga <string.h> (yangi ismi <cstring>) e'lon faylini kiritishimiz kerak. Ushbu e'lon faylida berilgan funksiyalar bilan keyingi bo'limda ishlaymiz.
Harflar, yani literalar, aytib o'tganimizdek, C++ da char tipi orqali beriladi. Literalar apostroflarga ('S', '*' ...) olinadi. Satrlar esa qo'shtirnoqlarga olinadi. Satrlar e'loniga misol beraylik.

```
char string[] = "Malibu";
char *p = "Ikkinchi!?!";
```

Satrlarni yuqoridagi ikkita yo'l bilan initsalizatsiya qilsa bo'ladi.
Satrlar ikkinchi uslubda e'lon qilinganda, yani pointer mehanizmi qo'llanilganda, ba'zi bir kompilyatorlar satrlarni hotiraning konstantalar saqlanadigan qismiga qo'yadi. Yani ushbu satrlarga o'zgartirish kiritilishi ta'qiqlanadi. Shu sababli satrlarni hardoim o'zgartirish imkoni bo'lishi uchun ularni char tipidagi massivlar ko'rinishida e'lon qilish afzaldir.
Satrlar massiv yordamida berilganda, ularning uzunligi noma'lumdir. Shu sababli satrning tugaganligini bildirish uchun satr ohiriga mahsus belgi nol literasi qo'yiladi. Uning dastursa belgilanishi '\0' ko'rinishga ega. Demak, yuqorida berilgan "Malibu" satriga ohiriga '\0' belgisi qo'shiladi, yani massivning umumiy uzunligi "Malibu":6 + '\0':1 = 7 ta char tipidagi elementga teng bo'ladi. Satrlarni massiv initsalizatsiya ro'yhati ko'rinishida ham bersak bo'ladi:

```
char c[6] = {'A', 'B', 'C', 'D', 'E' , '\0'};
...
cout << c;
...
```

Ekranda:

```
ABCDE
```

Biz cout bilan c ning qiymati ekranga bosib chiqardik. Aynan shunday klaviaturadan ham o'qib olishimiz mumkin:

```
char string[80];
cin >> string;
```

Eng muhimi satr bilan '\0' belgisi uchun yetarli joy bo'lishi kerak.
Satrlar massiv yordamida berilganligi uchun, alohida elementlarga indeks orqali yetishish mumkin, masalan:

```
char k[] = "Bahor keldi, gullar ochildi.";
for (int i = 0; k[i] != '\0'; i++)
if ( (i mod 2) == 0 ) // juft sonlar uchun haqiqat bo'ladi
cout << k[i] << " ";
```
Ekranda:
B h r k l i u l r o h l i
Yuqoridagi misolda, sikl tugashi uchun k[i] element '\0' belgiga teng bo'lishi kerak.

Dasturiy qism

Dastur vazifasi: Biz tuzadigan dasturning asosiy vazifasi shundan iboratki
“ **O’nlik sanoq sistemasidagi sonlar ichidan ikkita raqami bir xil bo’lmagan barcha to’rt xonali sonlarni chop etish” dan iborat**

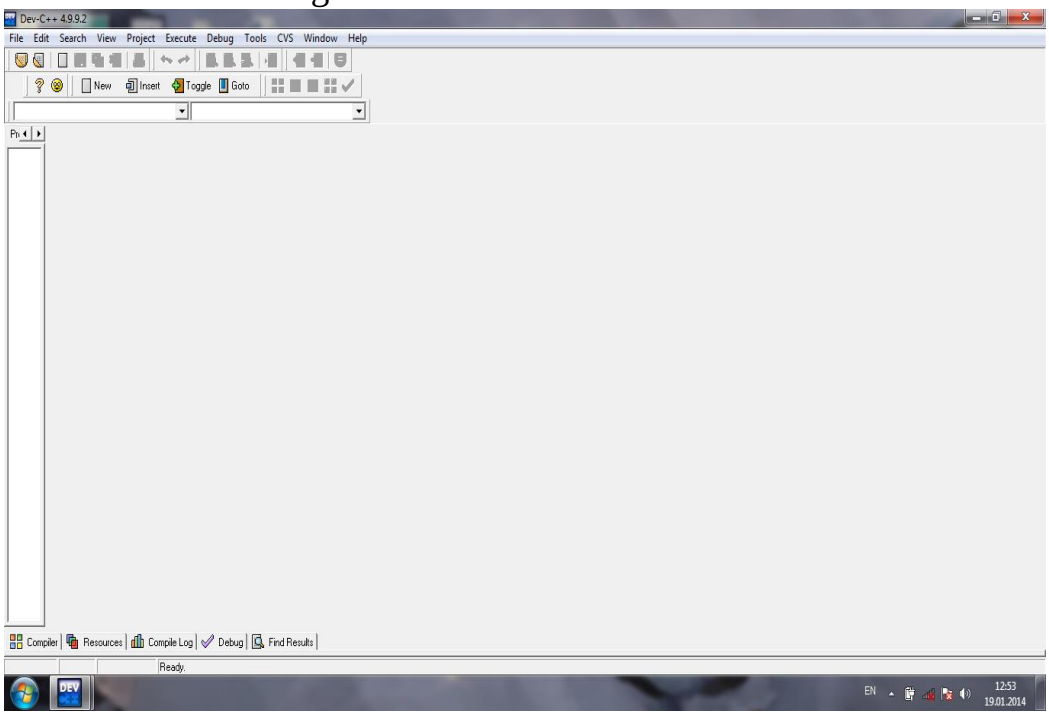
Ushbu dasturning umumiy ko’rinishi quyidagicha

```
//Muallif: Hakimov J
//Maqsad: Kiritilgan sonlar ichidan raqami turli to'rt xonali sonalrni aniqlash
//Sana: 15.01.2014

#include <iostream>
#include <cstdlib>

using namespace std;

int main(){
	//============================================
O'zgaruvchilar e'loni
	int n;//Kiritiladigan sonlar soni
	int A[100]; //Kiritiladigan sonlarni o'zlashtirish uchun massiv
	int raqam_soni = 4;//4 ta raqam
	int B[raqam_soni];//4 xonali son raqamlarini o'zlashtirish uchun
	int k;//4 xonali sonlarni o'zlashtirish uchun

	cout << "Nechta son kiritasiz? "; cin >> n;
	for(int i = 0; i < n; i++)
	{
		cout << i + 1 << " - son : "; cin >> A[i];
	}
	//============================================ Raqami
har-xil 4 xonali sonlarni aniqlash
	cout << endl;
	for(int i = 0; i < n; i++)
	{
		if(i == 0 ) cout << "Siz kiritgan sonlar ichida raqami turli 4 xonali
sonlar quyidagilar:\n";
```

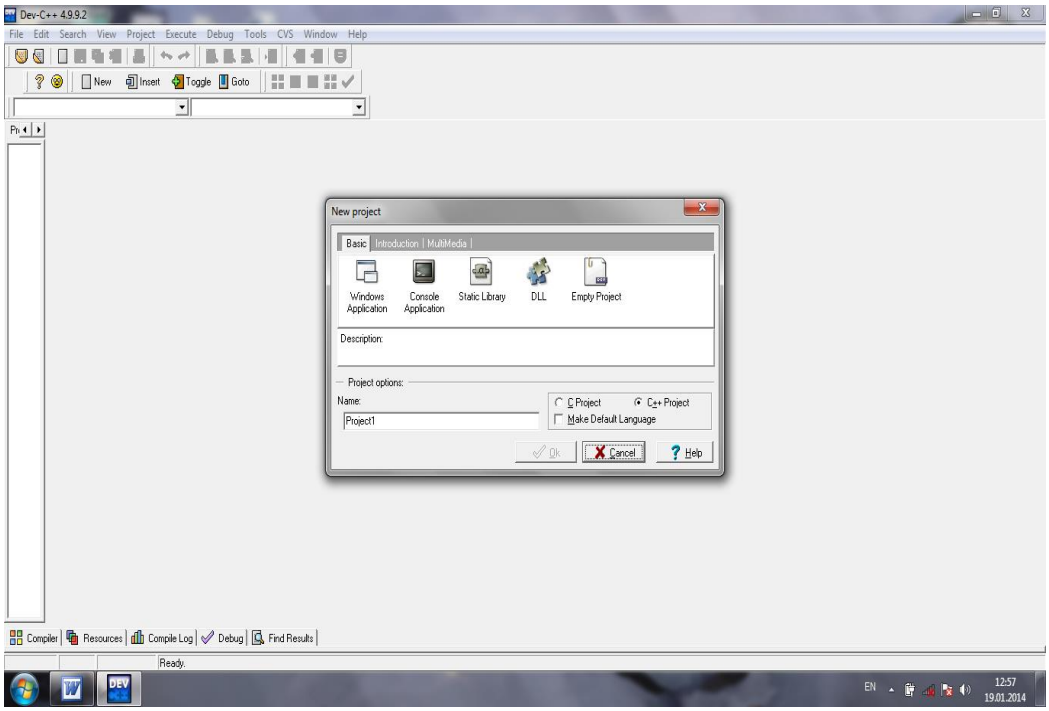
```
                if(A[i] >= 1000 && A[i] <= 9999)//Faqat 4 xonali sonlar uchun
                {
                        k = A[i];//Sonni k ga o'zlashtirish, A[i]massiv elementini
keyinchalik
                                //chop etishimiz mumkin
                    //Son raqamlarini massivga o'zlashtirish
                    for(int j = 0; j < raqam_soni; j++)
                    {
                            B[j] = k % 10 ; k = k / 10;
                            //Ushbu jarayonda k ning qiymati o'zgaradi
                        }//end for j

                    //Tekshirish va raqamlari har xil bo'lganini chop etish
                    if((B[0] != B[1]) && (B[0] != B[2]) && (B[0] != B[3]) && (B[1]
!= B[2]) && (B[1] != B[3]) && (B[2] != B[3]))
                    {
                            cout << A[i] << " ";
                    }// end if2
                }//end if1
        }//end for i
        system("Pause");
        return 0;
}
```

Ushbu dasturni C++ dasturlash tilida ishga tushiramiz buning uchun biz eng avvalo

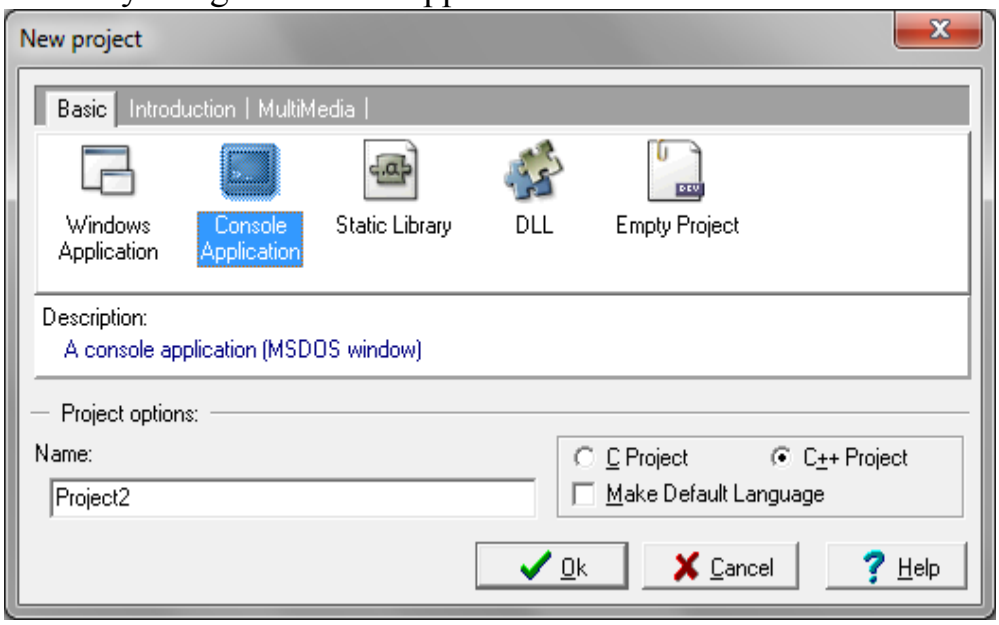
1. C++ dasturini ishga tushiramiz



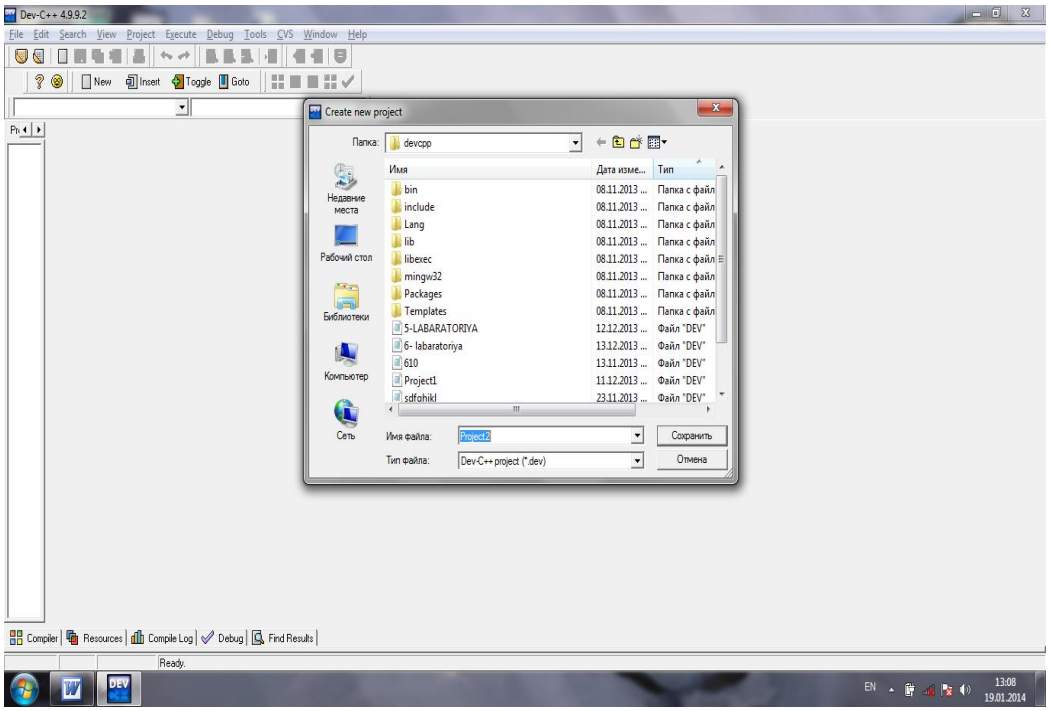
2. Endi ishchi oynasini ochamiz, buning uchun “Project” ni bosamiz va quyidagi oynaga ega bo’lamiz



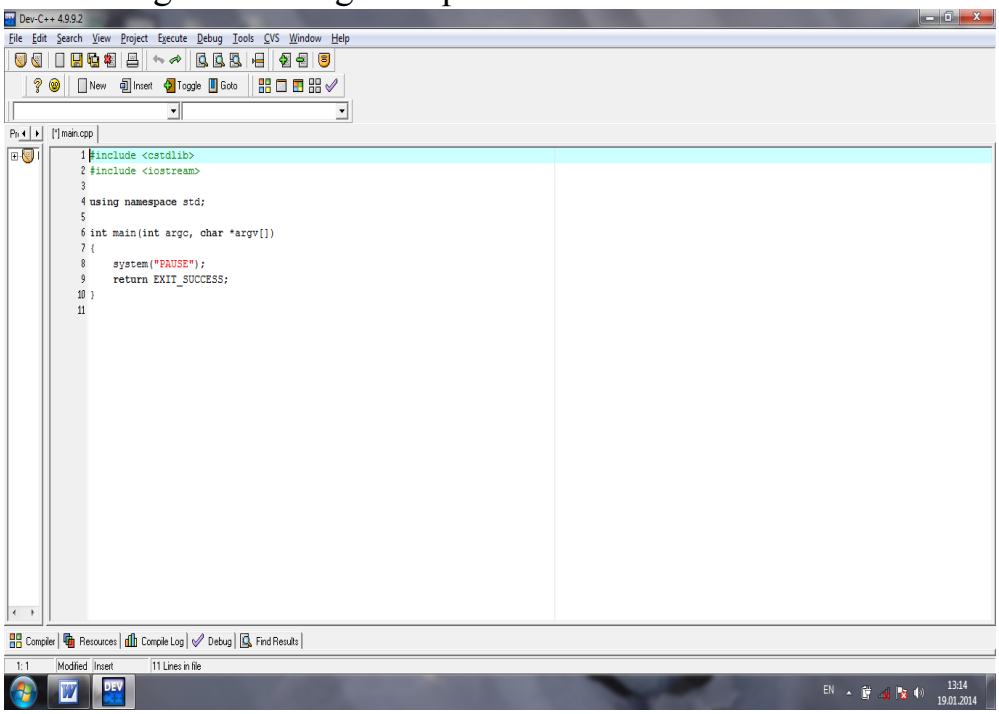
4. ushbu oynadagi “Consola Application”→ :Ok”ni bosamiz

5. 

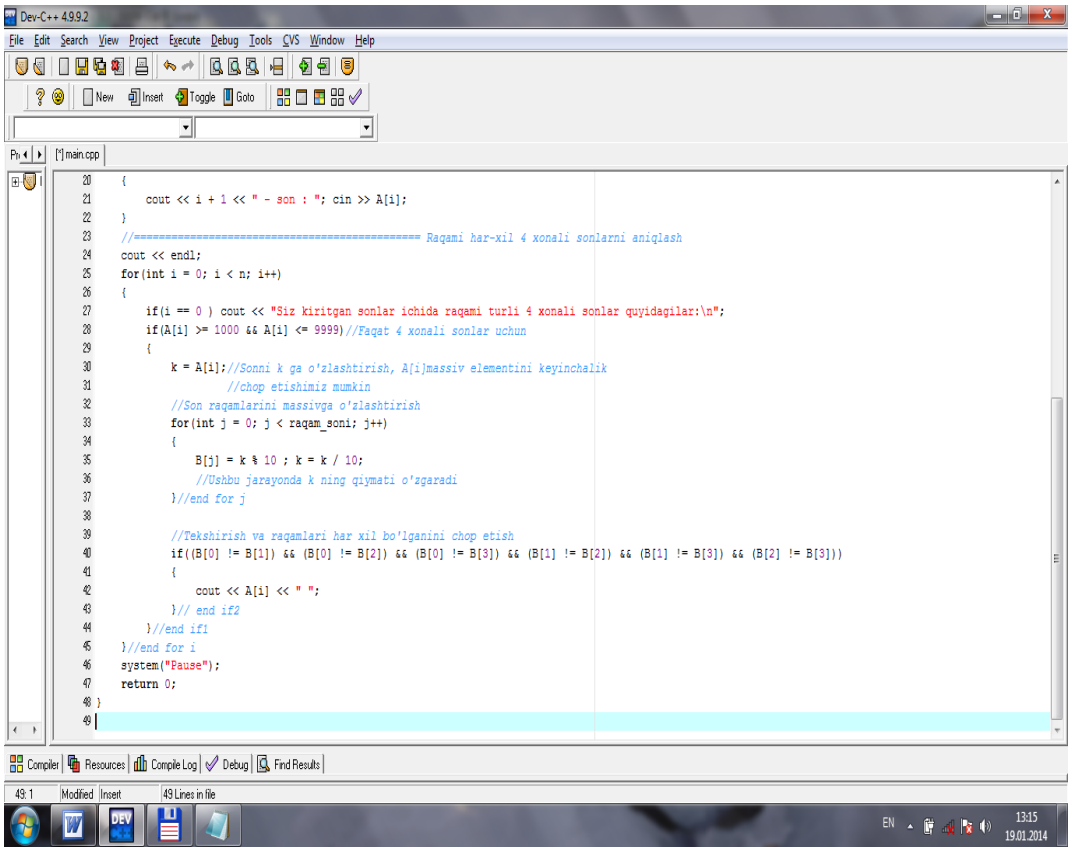
6. Nom beramiz va qayerga saqlashmni hal qilamiz va OK ni bosamiz.



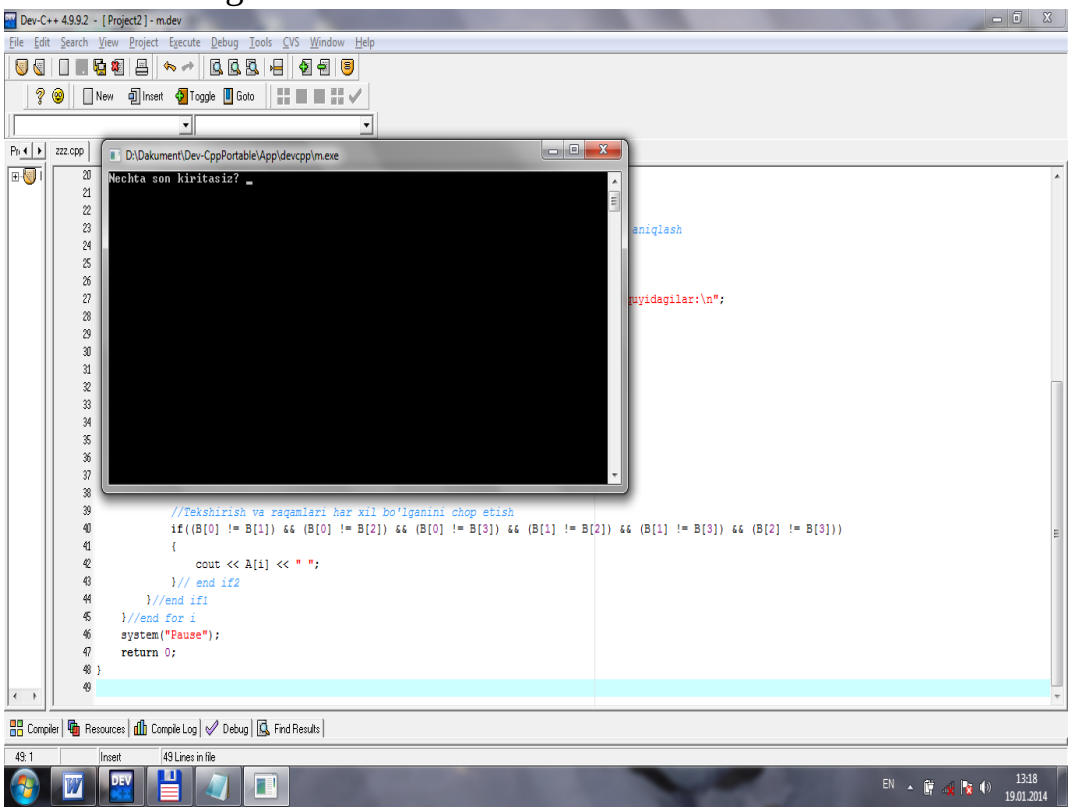
6. So’ng ishchi oynasi ochiladi.
7. Nomlagandan so’ng “Сохранит” ni bosamiz



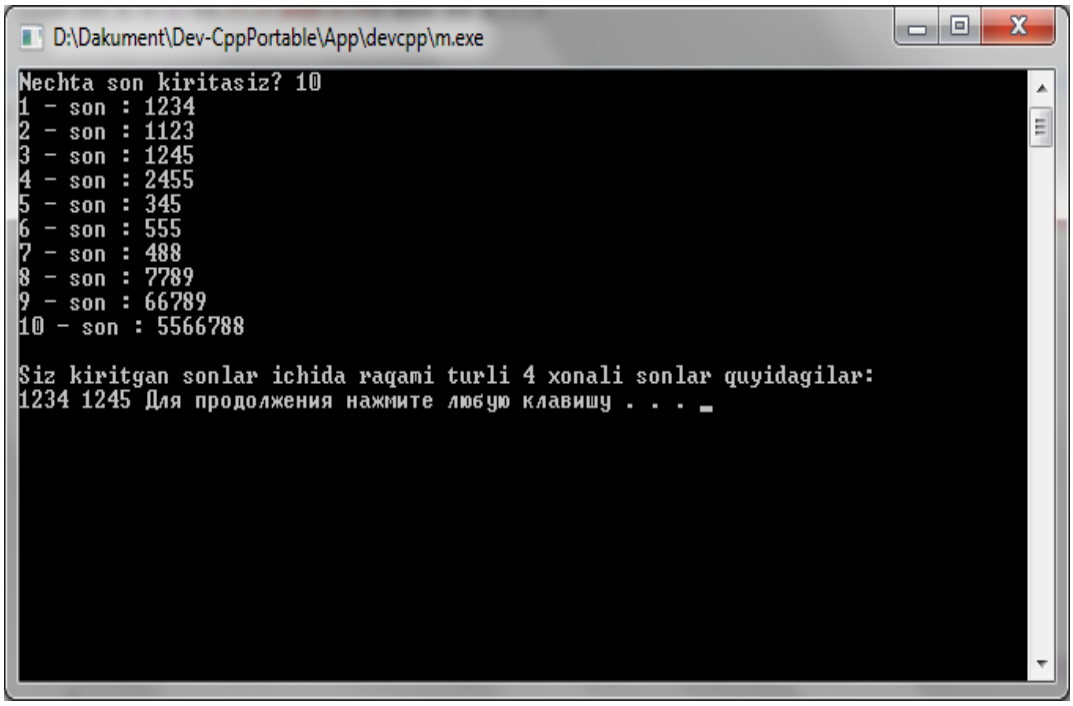
8. { belgisidan keyin dasturni kiritamiz



9. dasturni ishga tushirish uchun F9 ni bosamiz



10.ushbu oynaga sonni kiritamiz



Biz quyidagi natijaga ega bo’lamiz

Xulosa

Men ushbu kurs ishimnii bajarish davomida Dev C++ tilining massivlar bilan ishlash imkoniyati va shart operatorlari bilan ishlashni qangicha usullari bilan o’rganib chiqdim . Dev C++ tilida amaliy dastur yaratish mobaynida shunga amin bo’ldimki, berilgan masalani yechish uchun birinchi navbatda uning mohiyatini to’liq anglab olish kerakligini tushundim. Chunki, masalani tushunib olish masalani yarim yechimi xisoblanishini bilamiz. Dastur tuzishda uning har operatorlaridan kerakli joyda kerakli tartibda foydalanish dasturning hajm jihatdan va sifat ko’rsatkichini yuqori darajada ekanligini ifodalaydi. Eng asosiysi dasturning foydalanuvchi bilan interfeysini yaxshi tashkillashimiz kerakligi o’rgandim. Massivlar ustida juda ko’plab masalalarni hal qilishimiz mumkin. Massivlarni juda mukkamal ishlashni bilgan dasturchi har qanday dasturni hal qila olishiga ishondim. Massivlarni yaxshi bilmagan dasturchi bu juda qoloq dasturchi bo’lib qolaveradi.

Foydalanilgan adabiyotlar

1. G’ulomov S. S. va boshqalar. Axborot tizimlari va texnologiyalari: Oliy o‘kuv yurti talabalari uchun darslik Akademik S. S. G’ulomovning umumiy taxriri ostida.—T.: «Sharq», 2000.

1. **N.G. Mardanova . Programmalash bo`yicha chuqurlashtirilgan kurs T 2006**
2. **Arslonov K. C++ dan qo’llanma T .:2010**
3. **Ahunjonov M. C++ tili dasturlash asoslari .Farg’ona – 2009**
4. **www.acm.tuit.uz**
5. **www.ziyonet.uz**
6. **www.google.ru**
7. **www.dastur.uz**