C++visual dasturlash muhitida massivlar va satrlar

R e j a

KIRISH

Vaqt o‘tishi bilan kompyuterlar tobora kengroq qo‘llana boshladi hamda yuqoriroq darajadagi protsedura dasturlash tillari paydo bo‘ldi.

Hozirgi kunda respublikamizdagi texnika oliy o`quv yurtlarida “Informatika va axborot texnologiyalari” yo`nalishi va mutaxassisliklariga turli xil dasturlash tillarini o`rgatish mo`ljallangan. Bizga ma`lumki, dasturlash tillarining yuzdan ortiq ko`rinishlari mavjud, lekin qo`llanilishi ko`lamiga qarab C/C++ va C# dasturlash tillari yuqori dasturlash sinfiga mansubdir.

Mutaxassislarning fikriga ko`ra C++ dasturlash tili Assembler dasturlash tiliga eng yaqin bo`lib, tezlik jihatidan 10 % ortda qolar ekan.

Keyingi yillarda amaliy dasturchilarga juda ko`p integratsion dastur tuzish muhitlari taklif etilmoqda. Bu muhitlar u yoki bu imkoniyatlari bilan bir–biridan farq qiladi. Aksariyat dasturlashtirish muhitlarining fundamental asosi C/C++ tiliga borib taqaladi.

Ushbu kurs ishi hozirgi kunda komp’yuterda berilgan masalalarni dasturlash tillari orqali echih va dasturlashda massivlar va satrlar bilan ishlash hamda boshqa o’eklar bilan ishlash kabi ko’nikmalarni o`rganishga bag’shlanadi.

1.1.Massiv tushunchasi

Massiv - bu bitta turga mansub bir nechta o‘zgaruvchilar to‘plami. TYPE turidagi LENGTH ta elementdan iborat a nomli massiv shunday e’lon qilinadi:

type a[length];

Bu maxsus a[0], a[1], ..., a[length-1] nomlarga ega bo‘lgan type turidagi o‘zgaruvchilarning e’lon qilinishiga to‘g‘ri keladi. Massivning har bir elementi o‘z raqamiga - indeksga ega. Massivning x-nchi elementiga kirish indekslash operatsiyasi yordamida amalga oshiriladi:

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

Indeks sifatida butun tur qiymatini chiqarib beradigan har qanday ifoda qo‘llanishi mumkin: char, short, int, long. Si da massiv elementlarining indekslari 0 dan boshlanadi (1 dan emas), LENGTH elementdan iborat bo‘lgan massivning oxirgi elementining indeksi esa - bu LENGTH-1 (LENGTH emas). SHuning uchun massivning barcha elementlari bo‘yicha davr - bu

```
TYPE a[LENGTH]; int indx;
fjr(indx< LENGTH; indx++)
...a[indx]...;
```

indx< LENGTH ning qiymati indx<= LENGTH-1 qiymatiga teng. Massiv chegarasidan tashqariga chiqish (ya’ni mavjud bo‘lmagan elementni o‘qish/yozishga urinish) dastur xulq-atvorida kutilmagan natijalarga olib kelishi mumkin. SHuni ta’kidlab o‘tamizki, bu eng ko‘p tarqalgan xatolardan biridir.

Statik massivlarni nomlab e’lon qilish mumkin, bunda massivlar elementlarining qiymatlari vergul bilan ajratilgan shakldor qavs {} ichida sanab o‘tiladi. Agar massiv uzunligiga qaraganda kamroq element berilgan bo‘lsa, qolgan elementlar 0 hisoblanadi:

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

Agar nomlangan massivning tavsifida uning o‘lchamlari ko‘rsatilmagan bo‘lsa, u kompilyator tomonidan sanab chiqiladi:

int a3[]={1, 2, 3}; //go‘yo a3[3]

Massivlarni navlarga ajratish

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

Massivlarning navlarga ajratilishi tez xarakatlanuvchiligiga ko‘ra farqlanadi. Navlarga ajratishning n*n ta qiyoslashni talab qilgan oddiy usuli va n*In(n) ta qiyoslashni talab qilgan tez usuli mavjud. Oddiy usullar navlarga ajratish tamoyillarini tushuntirishda qulay hisoblanadi, chunki sodda va kalta algoritmlarga ega. Murakkablashtirilgan usullar kamroq sonli operatsiyalarni talab qiladi, biroq operatsiyalarning o‘zi murakkabroq, shuning uchun uncha katta bo‘lmagan massivlar uchun oddiy usullar ko‘proq samara beradi.

Oddiy sullar uchta asosiy kategoriyaga bo‘linadi:

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

1.2.Oddiy kiritish usuli bilan navlarga ajratish

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

44	55	12	42	94	18

Tayyor dastlabki ketma-ketlik

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

- agar a[J]>a[I] elementi topilgan bo‘lsa;
- agar tayyor ketma-ketlikning chap uchiga etilgan bo‘lsa.

```
int i, j, x;
fjr(i=1; i<n; i++)
{
x=[i];// kiritib qo‘ishimiz lozim bo‘lgan elementni esda saqlab qolamiz
j=i-1;
while(x<a[j]&&j>=0)//to‘g‘ri keladigan joyni qidirish
}
a[j+1]=a[j]$//o‘nga surilish
j--;
```

```
}
a[j+1]=x;//elementni kiritish
}
```

Oddiy tanlash usuli bilan navlarga ajratish;

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

44	55	12	42	94	18

1 min

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

Oddiy almashtirish usuli bilan navlarga ajratish

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

44	55	12	42	94	18

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

1.3. Ko‘p o‘lchamli massivlar

C++ da massivning eng umumiy tushunchasi - bu kuo‘rsatkichdir, bunda har xil turdagi ko‘rstakich bo‘lishi mumkin, ya’ni massiv har qanday turdagi elementlarga, shu jumladan, massiv bo‘lishi mumkin bo‘lgan ko‘rsatkichlarga ham ega bo‘lishi mumkin. O‘z tarkibida boshqa massivlarga ham ega bo‘lgan massiv ko‘p o‘lchamli hisoblanadi.

Bunday massivlarni e’lon qilishda kompyuter xotirasida bir nechta turli xildagi ob’ekt yaratiladi. Masalan, int arr[4][3] int int int

Arr

↓

arr[0] → arr[0][0] arr[0][1] arr[0][2]

arr[1] → arr[1][0] arr[1][1] arr[1][2]

arr[2] → arr[2][0] arr[2][1] arr[2][2]

arr[3] → arr[3][0] arr[3][1] arr[3][2]

Shunday qilib, arr[4][3] ning e’lon qilinishi dasturda uchta turli xildagi ob’ektlarni yuzaga keltiradi: arr identifikatorli ko‘rsatkichni, to‘rtta ko‘rsatkich dan iborat nomsiz massivni va int turidagi o‘n ikkita sondan iborat nomsiz massivni. Nomsiz massivlarga kirish huquqiga ega bo‘lish uchun arr ko‘rsatkichli adresli ifodalar qo‘llanadi. Ko‘rsatkichlar massivi elementlariga kirish huquqi arr[2] yoki *(arr+2) shaklidagi indeksli ifodaning bittasini ko‘rsatish orqali amalga oshiriladi.

Int turidagi ikki o‘lchamli sonlar massiviga kirish uchun arr[1][2] shaklidagi ikkita indeksli ifoda yoki unga ekvivalent bo‘lgan *(*(arr+1)+2) va (*(arr+1))[2] shaklidagi ifodalar qo‘llanishi kerak. Shuni ham hisobga olish kerakki, Si tili sintaksisi nuqtai nazaridan arr ko‘rsatkichi va arr[0], arr[1], arr[2]? arr[3] ko‘rsatkichlari konstantalardir hamda ularning qiymatlarini dasturni bajarish paytida o‘zgartirish mumkin emas. Uch o‘lchamli massivni joylashtirish ham xuddi shunga o‘xshash amalga oshiriladi hamda float arr3[3][4][5] ning e’lon qilinishi dasturda, float turidagi oltmishta sondan iborat uch o‘lchamli massivning o‘zidan tashqari, float òóðèää tuzilgan òo‘rtta ko‘rsatkichdan iborat massivni, float êo‘rsatkichlar massiviga tuzilgan uchta ko‘rsatkichdan iborat massivni va float ää tuzilgan êo‘rsatkichlar massivining massivlariga ko‘rsatkichni yuzaga keltiradi. Ko‘p o‘lchamli massivlar elementlarini joylashtirishda ular xotirada satrlar bo‘yicha bir tartibda joylashtiriladi., ya’ni oxirgi indeks hammadan tezroq o‘zgaradi, birinchisi esa sekinroq o‘zgaradi. Bunday tartib, ko‘p o‘lchamli massiv boshlang‘ich elementining adresini hamda faqat bitta indeks ifodasini qo‘llab, ko‘p o‘lchamli massivning har qanday elementiga murojaat qilish imkonini beradi.

Masalan, arr[1][2] elementiga murojaatni ptr2 ko‘rsatkichi yordamida amalga oshirsa bo‘ladi. Bu ko‘rsatkich esa ptr2[1*4+2] () murojaati yoki ptr2[6] murojaati sifatida int *ptr2=arr[0] shaklida e’lon qilingan bo‘ladi. Ta’kidlab o‘tishimiz lozimki, tashqi tomondan o‘xshash arr[6] murojaatini bajarish mumkin emas, chunki 6 indeksli ko‘rsatkich mavjud emas.

Shuningdek, uch o‘lchamli massivga kiradigan arr3[2][3][4] elementiga murojaat uchun float *ptr3=arr3[0][0] ko‘rinishida tavsiflangan, ptr3[3*2+4*3+4] yoki ptr3[22] shaklidagi bitta indeksli ifodaga ega bo‘lgan ko‘rsatkichni qo‘llash mumkin.

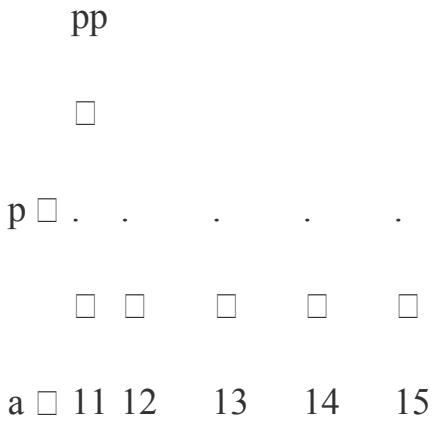
Ko‘rsatkichlar massilari

Si tilida massivlar elementlari har qanday turga ega bo‘lishi mumkin, xususan, har qanday turdagi ko‘rsatkichlar bo‘lishi mumkin. Ko‘rsatkichlar qo‘llangan bir nechta misolni ko‘rib chiqamiz.

¤zgaruvchilarning quyidagi e’lonlari:

```
int a[]={10,11,12,13,14};
int a[]={a, a+1, a+2, a+2, a+3, a+4};
int **pp=p;
```

mana bu sxemada ko‘rsatilgan dasturiy ob’ektlarni yuzaga keltiradi.



E’lon qilishda o‘zgaruvchilarni joylashtirish ɛemasi.

Rr-r operatsiyasini bajarishda nol qiymatga ega bo‘lamiz, chunki rr va r iqtiboslari o‘zaro teng hamda ko‘rsatkichlar massivining boshlanQich elementiga ishora qiladi. Bu erda ko‘rsatkichlar massivi r ko‘rsatkichi (r[0] elementga) bilan boQliq.

Rr+=2 operatsiyasi bajarilgandan keyin, sxema o‘zgaradi hamda quyidagi tasvirlangandek ko‘rinishga ega bo‘ladi.

p □ . . . . .

□ □ □ □ □

a □ 10 11 12 13 14

rr+=2 operatsiyasi bajarilgandan keyin o‘zgaruvchilarni joylashtirish sxemasi.

Rr-r ni ayirish natijasi 2 ga teng bo‘ladi, chunki rr ning qiymati r massividagi uchinchi element adresi hisoblanadi. *rr-a iqtibosi ham 2 qiymatni beradi, chunki *rr murojjati a massivi uchinchi elementining adresidir, a murojaati esa a massivi boshlanQich elementining adresidir.**rr iqtibosi yordamidagi murojaatda 12 ga ega bo‘lamiz - u a massivi uchinchi elementining qiymatidir. *rr++ iqtibosi r massivi to‘rtinchi elementining qiymatini, ya’ni a massivi to‘rtinchi elementining adresini beradi.

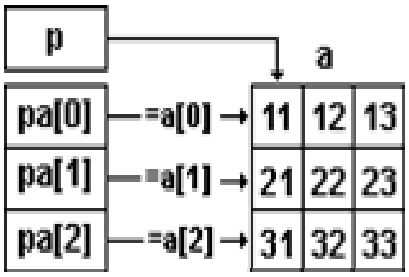
Rr=r deb hisoblasak, u holda *++rr murojaati a massivi birinchi elementining qiymati bo‘ladi (ya’ni 11 qiymati), ++*rr operatsiyasi r[0] ko‘rsatkichining ichidagisini shunday o‘zgartiradiki, u a elementi adresining qiymatiga teng bo‘lib qoladi.

Murakkab murojaatlar ichidan turib ochiladi. Masalan, *(++(*rr)) murojaatini quyidagi amallarga bo‘lish mumkin: *rr r[0] massivi boshlang‘ich elementining qiymatini beradi, keyin bu qiymat ++(*r) ga inkrementatsiya bo‘ladi, buning natijasida r[0] ko‘rsatkichi a[1] elementi adresining qiymatiga teng bo‘lib qoladi, va, nihoyat, oxirgi amal -bu olingan adres bo‘yicha qiymatlarni tanlash, ya’ni 11 qiymati.

Avvalgi misollarda bir o‘lchamli massiv qo‘llangan edi. Endi ko‘p o‘lchamli massiv va ko‘rsatkichlar berilgan misolni ko‘rib chiqamiz. Quyidagi o‘zgaruvchilarning e’lonlari:

```
int a[3][3]={  {11,12,13},
          {21,22,23},
           {31,32,33}   };
int *pa[3]={a,a[1],a[2]};
int *p=a[0];
```

dasturda mana bu sxemada ko‘rsatilgan ob’ektlarni yuzaga keltiradi



Dinamik massivlar

C++tilida o‘zgaruvchilar yo statik tarzda - kompilyatsiya paytida, yoki standart kutubxonadan funktsiyalarni chaqirib olish yo‘li bilan dinamik tarzda - dasturni bajarish paytida joylashtirilishi mumkin. Asosiy farq ushbu usullarni qo‘llashda ko‘rinadi - ularning samaradorligi va moslashuvchanligida. Statik joylashtirish samaraliroq, chunki bunda xotirani ajratish dastur bajarilishidan oldin sodir bo‘ladi. Biroq bu usulning moslashuvchanligi ancha past, chunki bunda biz joylashtirilayotgan ob’ektning turi va o‘lchamlarini avvaldan bilishimiz kerak bo‘ladi. Masalan, matniy faylning ichidagisini satrlarning statik massivida joylashtirish qiyin: avvaldan uning o‘lchamlarini bilish kerak bo‘ladi. Noma’lum sonli elementlarni oldindan saqlash va ishlov berish kerak bo‘lgan masalalar odatda xotiraning dinamik ajratilishini talab qiladi.

Xotirani dinamik va statik ajratish o‘rtasidagi asosiy farqlar quyidagicha:

- statik ob’ektlar nomlangan o‘zgaruvchilar bilan belgilanadi, hamda ushbu ob’ektlar o‘rtasidagi amallar to‘g‘ridan-to‘g‘ri, ularning nomlaridan foydalangan holda, amalga oshiriladi. Dinamik ob’ektlar o‘z shaxsiy otlariga ega bo‘lmaydi, va ular ustidagi amallar bilvosita, ko‘rsatkichlar yordamida, amalga oshiriladi;
- statik ob’ektlar uchun xotirani ajratish va bo‘shatish kompilyator tomonidan avtomatik tarzda amalga oshiriladi. Dasturchi bu haqda o‘zi qayg‘urishi kerak emas. Statik ob’ektlar uchun xotirani ajratish va bo‘shatish to‘laligicha dasturchi zimmasiga yuklatiladi. Bu anchayin qiyin masala va uni echishda xatoga yo‘l qo‘yish oson.

Dinamik tarzda ajratilayotgan xotira ustida turli xatti-harakatlarni amalga oshirish uchun new va delete poeratorlari xizmat qiladi.

Shu paytga qadar barcha misollarda statik xotira ajratish qo‘llanadi. Masalan, i o‘zgaruvchisini aniqlash:

int i=1024;

Bu komanda xotirada shunday sohani ajratib beradiki, u int turidagi o‘zgaruvchini saqlash, ushbu soha bilan i nomini bog‘lash hamda u erga 1024 qiymatini joylashtirish uchun etarli bo‘ladi. Bularning hammasi dastur bajarilishidan oldin kompilyatsiya bosqichida amalga oshiriladi.

Biroq o‘zgaruvchiga xotirani ajratib berish uchun yana bir usul mavjud bo‘lib, u new operatorini qo‘llashdan iborat.

New operatori ikkita shaklga ega. Birinchi shakl ma’lum bir turdagi yakka ob’ektga xotirani ajratib beradi;

int*pint=new int(1024)

Bu erda new operatori int turidagi nomsiz ob’ektga xotirani ajratib beradi, uni 1024 qiymati bilan nomlantiradi (initsiallashtiradi) hamda yaratilgan ob’ekt adresini qaytarib beradi. Bu adres pint ko‘rsatkichiga joylashtiriladi. Ushbu nomsiz

ob’ekt ustidagi barcha xatti-harakatlar shu ko‘rsatkich bilan ishlash orqali amalga oshiriladi, chunki dinamik ob’ekt bilan to‘g‘ridan-to‘g‘ri ish olib borish (manipulyatsiyalar o‘tkazish) mumkin emas.

New operatorining ikkinchi shakli ma’lum bir turdagi elementlardan tashkil topgan berilgan o‘lchamlardagi massivga xotira ajratib beradi:

int *pia=new int[4];

Bu misolda xotira int turidagi to‘rtta elementdan iborat massivga xotira ajratiladi. Afsuski, new operatorining bu shakli massiv elementlarini nomlantirish (initsiallashtirish) imkonini bermaydi.

New operatorining har ikkala shakli ham bir xil ko‘rsatkichni qaytarishi (keltirilgan misolda bu butun sonning ko‘rsatkichi) ayrim chalkashliklarga olib keladi. pint ham pia ham aynan bir xil e’lon qilingan Biroq pint operatori int turidagi bitta ob’ektni ko‘rsatadi, pia esa int turidagi to‘rtta ob’ektdan iborat massivning birinchi elementini ko‘rsatadi.

Dinamik ob’ekt kerak bo‘lmay qolganda, unga ajratilgan xotirani to‘g‘ridan-to‘g‘ri bo‘shatish kerak. Bu ish delete operatori yordamida amalga oshiriladi:

delete pint;

Ob’ektning bo‘shatilishi, new kabi, ikkita shaklga ega - yakka ob’ekt uchun va massiv uchun:

delete[] pia;

Agar ajratilgan xotirani bo‘shatish esdan chiqqudek bo‘lsa, bu xotira bekordan-bekorga sarflana boshlaydi, foydalanilmay qoladi, biroq, agar uning ko‘rsatkichi o‘z qiymatini o‘zgartirgan bo‘lsa, uni tizimga qaytarish mumkin emas. Bu hodisa *xotiraning yo‘qotilishi (utechka pamyati)* degan maxsus nom bilan

ataladi. Pirovard natijada dastur xotira etishmagani tufayli avariya holatida tugallanadi (agar u ancha vaqt ishlayversa).

1.4. Satrlar. Belgili axborot va satrlar

Si++ da belgili ma'lumotlar uchun char turi qabul qilingan. Beligili axborotni taqdim etishda belgilar, simvolli o‘zgaruvchilar va matniy konstantalar qabul qilingan.

Misollar:

sonst char c='c';//belgi - bir baytni egallaydi, uning qiymati o‘zgarmaydi

char a,b;//belgili o‘zgaruvchilar, bir baytdan joy egallaydi, qiymatlari o‘zgaradi.

const char *s= ‘‘\n satrining misoli’’;//matniy konstanta

Si++ dagi satr - bu nul-belgi - ‘\0’ (nul-terminator)- bilan tugallanuvchi beliglar massivi. Nul-terminatorning holatiga qarab satrning amaldagi uzunligi aniqlanadi. Bunday massivdagi elementlar soni, satr tasviriga qaraganda, bittaga ko‘p.

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

Misol:

```
void main()
{
        char s1[10]=''string1'';
        int k=sizeof (s1);
        cout<<s1<<''\t''<<k<<endl;
        char s2[]=''string2'';
        k=sizeof(s2);
        cout<<s2<<''\t''<<k<<endl;
        char s3[]={'s','t','r','i','n','g','3'};
        k=sizeof(s3);
        cout<<s3<<''\t''<<k<<endl;
        char *s4=''string4'';//satr ko'rsatkichi, uni o'zgartirib bo'lmaydi
        k=sizeof(s4);
        cout<<s4<<''\t''<<k<<endl;
}
```

Natijalar:

string1 10 - 10 bayt ajratilgan, shu jumladan \0 ga
string2 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)
string3 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)
string4 4 - ko'rsatkichning o'lchamlari

1.5. Bir o'lchamli massivlarni funktsiya parametrlari sifatida uzatish

Massivdan funktsiya parametri sifatida foylalanganda, funktsiyaning birinchi elementiga ko'rsatkich uzatiladi, ya'ni massiv hamma vaqt adres bo'yicha uzatiladi. Bunda massivdagi elementlarning miqdori haqidagi axborot yo'qotiladi, shuning uchun massivning o'lchamlari haqidagi ma'lumotni alohida parametr sifatida uzatish kerak. Funktsiyaga massiv boshlanishi uchun ko'rsatkich uzatilgani

tufayli (adres bo‘yicha uzatish), funktsiya tanasining operatorlari hisobiga massiv o‘zgarishi mumkin.

Misol:

Massivdan barcha juft elementlar chiqarilsin

```
#include <iostream.h>
#include <stdlib.h>
int form(int a[100])
{
int n;
        cout<<"\nEnter n";
        cin>>n;
        for(int i=0;i<n;i++)
                a[i]=rand()%100;
        return n;
}
void print(int a[100],int n)
{
        for(int i=0;i<n;i++)
                cout<<a[i]<<" ";
        cout<<"\n";
}
void Dell(int a[100],int&n)
{
        int j=0,i,b[100];
```

```
        for(i=0;i<n;i++)
                if(a[i]%2!=0)
                {
                        b[j]=a[i];j++;
                }
                n=j;
                for(i=0;i<n;i++)a[i]=b[i];
}
void main()
{
        int a[100];
        int n;
        n=form(a);
        print(a,n);
        Dell(a,n);
        print(a,n);
}
```

1.6. Satrlarni funktsiyalar parametrlari sifatida uzatish

Satrlar funktsiyaga char turidagi bir o‘lchamli massivlar sifatida yoki char* turidagi ko‘rsatkichlar sifatida uzatilishi mumkin. Oddiy massivlardan farqli o‘laroq, funktsiyada satr uzunligi ko‘rsatilmaydi, chunki satr oxirida satr oxiri /0 belgisi bor.

Misol:Berilgan belgini satrda qidirish funktsiyasi

```
int  find(char *s,char c)
```

```
{
for (int I=0;I<strlen(s);I++)
if(s[I]==c) return I;
return –1
}
```

Funktsiyaga ko‘p o‘lchamli massivlarni uzatish

Ko‘p o‘lchamli massivlarni funktsiyaga uzatishda barcha o‘lchamlar parametrlar sifatida uzatilishi kerak. Si va SI++ da ko‘p o‘lchamli massivlar aniqlanishi bo‘yicha mavjud emas. Agar biz bir nechta indeksga ega bo‘lgan massivni tavsiflasak (masalan, int mas[3][4]), bu degani, biz bir o‘lchamli mas massivini tavsifladik, bir o‘lchamli int [4] massivlarining ko‘rsatkichlari esa uning elementlaridir

Misol: Kvadrat matritsani uzatish (transportirovka qilish)

Agar void transp(int a[][],int n){.....} funktsiyasining sarlavhasini aniqlasak, bu holda biz funktsiyaga noma’lum o‘lchamdagi massivni uzatishni xohlagan bo‘lib qolamiz. Aniqlanishiga ko‘ra massiv bir o‘lchamli bo‘lishi kerak, hamda uning elementlari bir xil uzunlikda bo‘lishi kerak. Massivni uzatishda uning eelementlarining o‘lchamlari haqida ham biron narsa deyilmagan, shuning uchun kompilyator xato chiqarib beradi.

Bu muammoning eng sodda echimi funktsiyani quyidagicha aniqlashdir:

void transp(int a[][4],int n), bu holda har bir satr o‘lchami 4 bo‘ladi, massiv ko‘rsatkichlarining o‘lchami esa hisoblab chiqariladi.

```
#include<iostream.h>
const int N=4;//globalnaya peremennaya
void transp(int a[][N],int n)
{
int r;
```

```
for(int I=0;I<n;I++)
for(int j=0;j<n;j++)
if(I<j)
{
r[a[I][j];a[I][j]=a[j][I];a[j][I]=r;
}
}
void main()
{
int mas[N][N];
for(int I=0;I<N;I++)
for(int j=0;j<Nlj++)
cin>>mas[I][j];
for(I=0;I<N;I++)
{
for(j=0;j<N;j++)
cout<<mas[I][j]
cout<<"\n";
}
transp(N,mas);
for(I=0;I<N;I++)
{
for(j=0;j<N;j++)
cout<<mas[I][j]
```

```
cout<<"\n";
}
}
```

Funktsiya ko‘rsatkichi

Har bir funktsiya qaytarilayotgan qiymat turi, nomi va funktsiya parametrlari turlarining ro‘yxati bilan tavsiflanadi. Agar funktsiya nomidan keyinchalik qavslarsiz va parametrlarsiz foydalanilsa, bu holda u ushbu funktsiya ko‘rsatkichi sifatida amal qila boshlaydi, xotirada funktsiyani joylashtirish adresi esa uning qiymati bo‘lib qoladi. Bu qiymatni boshqa ko‘rsatkichga ham berish mumkin bo‘ladi. Bu holda ushbu yangi ko‘rsatkichdan funktsiyani chaqirib olish uchun foydalanish mumkin bo‘ladi. Funktsiyaga ko‘rsatkich quyidagicha aniqlanadi:

funktsiya_turi(ko‘rsatkich_*nomi)(parametrlar spetsifikatsiyasi)

Misol:

```
int f1(char c){.....}//funktsiyani aniqlash
int(*ptrf1)(char);//f1 funktsiyasiga ko‘rsatkichni aniqlash
```

Ko‘rsatkichni aniqlashda parametrlarning miqdori va turi ko‘rsatkich o‘rnatilayotgan funktsiyani aniqlashdagi tegishli turlarga mos kelishi kerak.

Ko‘rsatkich yordamida funktsiyani chaqirish quyidagi ko‘rinishga ega:

(ko‘rsatkich_*nomi)(faktik parametrlar ro‘yxati)

Misol:

```
#include <iostream.h>
void f1()
(cout<<''\nfunction f1'';)
```

```
void f2()
{cout<<''\nfunction f2'';}
void main()
{
void(*ptr)();//funktsiya ko‘rsatkichi
ptr=f2;//ko‘rsatkichga f2 funktsiyasining adresi beriladi
(*ptr)();// f2 funktsiyasini chaqirish
ptr=f1;//ko‘rsatkichga f1 funktsiyasining adresi beriladi
(*ptr)();//ko‘rsatkich yordamida f1 funktsiyasini chaqirish
}
```

Aniqlashda funktsiya ko‘rsatkichi shu paytning o‘zidayoq nomlantirilishi mumkin.

void (*ptr)()=f1;

Funktsiyalarga iqtiboslar

Funkiyaga ko‘rsatkich qanday aniqlansa funktsiyaga iqtibos ham xuddi shunday aniqlanadi:

funktsiya_turi(&iqtibos_nomi)(parametrlar)nomlantiruvchi_ifoda;

Misol:

int(&fret)(float,int)=f;// iqtibosni aniqlash

Funktsiya nomini parametrlarsiz va qavslarsiz qo‘llash funktsiya adresi sifatida qabul qilinadi. Funktsiyaga iqtibos funktsiya nomining sinonimi bo‘ladi. Funktsiyaga iqtibosning qiymatini o‘zgartirib bo‘lmaydi, shuning uchun ko‘p o‘rinda iqtibosga ko‘rsatkichlar emas, funktsiyaga ko‘rsatkichlar qo‘llanadi.

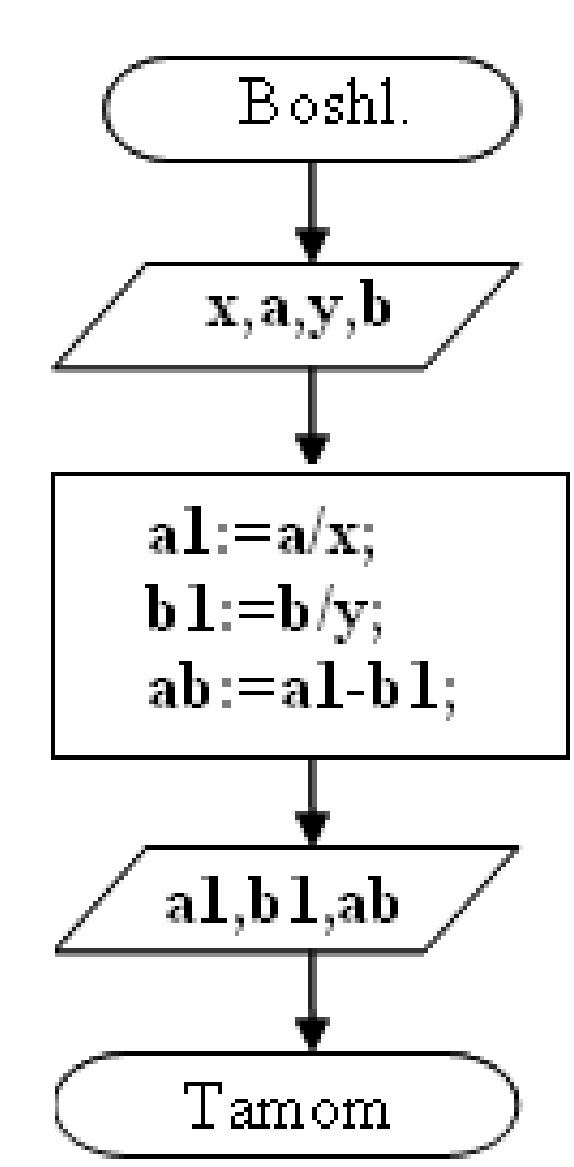
Misol:

```
#include <iostream.h>
void f(char c)
{cout<<’’\n<<c;}
void main()
{
void(*pf)(char);//funktsiya ko‘rsatkichi
void(&pf)(char);//èqtibos ko‘rsatkichi
f(‘A’);//nomi bo‘yicha chaqirish
pt=f;//ko‘rsatkich funktsiyaga ko‘yiladi
(*pt)(‘B’);//ko‘rsatkich yordamida chaqirish
rf(‘S’);//èqtibos bo‘yicha chaqirish
}
```

2.1. Masalaning quyilishi va tahlili

Ma’lumki, ***x*** kg shokolad ***A*** so’m, ***y*** kg iris ***B*** so`m turadi. 1 kg shokoladniy konfet va 1 kg iris qancha turishini va shokoladniy konfet, irisdan qancha qimmatligi aniqlash algoritmi va dasturini tuzish.

2.2. Algoritm blok-sxemasi



2.3. Algoritm dasturiiy kodi va natijalar.

```
#pragma  hisoblash

#include <condefs.h>
#include <iostream.h>

//---------------------------------------------
#pragma argsused
int main(int argc, char **argv)
{
static float x,a,y,b;
static float a1,b1,ab;
cin>>x>>a>>y>>b;
a1=a/x;
b1=b/y;
ab=a1-b1;
cout<<a1<<' '<<b1<<' '<<ab;
cin>>"\n";
return 0;
}
```

Ma’lumki, ***x*** kg shokolad ***A*** so’m, ***y*** kg iris ***B*** so`m turadi. 1 kg shokoladniy konfet va 1 kg iris qancha turishini va shokoladniy konfet, irisdan qancha qimmatligi aniqlansin.

5 10000
2 2000
2000
1000
1000

Xulosa

Xulosa qilib aytganda, C++ dasturlash tili va unda ob’ektlar va sinflar bilan ishlash xaqida umumiy ma’lumotlarga ega bo`ldim. Borland C++ Builder - Windows muhitida ishlaydigan dastur tuzish uchun qulay bo`lgan vosita bo`lib, komp’yuterda dastur yaratish ishlarini avtomatlashtiradi, xatoliklarni kamaytiradi va dastur tuzuvchi mehnatini engillashtiradi. Borland C++ dastur zamonaviy vizual loyihalash texnologiyasi asosida ob`ektga yo`naltirilgan dasturlash nazariyasini hisobga olgan holda tuziladi.

Borland C++ Builder 6 sistemasi C++ tilining rivoji bo`lgan ob`ektga yo`naltirilgan Object C/C# dasturlash tillarini ishlatadi. Borland C++ Builder sistemasi dasturni loyihalash va yaratish vaqtini kamaytiradi, hamda Windows muhitida ishlovchi dastur ilovalarini tuzish jarayonini osonlashiradi.

Men C++ dasturi strukturasi xaqida, belgilar bayoni , Algoritm va dastur tushunchasi, ma’lumotlarni kiritish va chikarish operatorlari xamda dasturda massivlar va satrlar bilan ishlash xaqida bilim va kunikmalarga ega bo`ldim.

Foydalanilgan adabiyotlar.

1. Sh.A.Nazirov, R.V.Qobulov «Ob’ektga mo‘ljallangan dasturlash »
2. Xaldjigitov A.A., Madraximov Sh. F., Adambayev U.E., Eshboyev E.A., Informatika va programmalash. T.:O`zMU, 2005, -148.
3. Гради Буч. Объектно –ориентированной анализ и проектирование с примерами приложений на C++. Невский диалект, 560 стр, 2001 г.
4. Грехем И. Объектно ориентированные методы. Принципы и практика. Вильямс. 879 стр, 2004 г.
5. Иванова Г.С. Объектно ориентированное программирование. Учебник. МГТУ им Баумана. 320 стр, 2003 г.
6. www.ziyonet.uz
7. www.tuit.uz
8. www.tuit.kf.uz