O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAXSUS KASB-HUNAR TA’LIMI VAZIRLIGI

O’RTA MAXSUS, KASB-HUNAR TA’LIMI MARKAZI

SAMARQAND VILOYATI O’RTA MAXSUS KASB-HUNAR TA’LIMI BOSHQARMASI

PASTDARG’OM QURILISH TEXNOLOGIYALARI KASB-HUNAR KOLLEJI

Mavzu: Yirik panelli (to’rt, besh va yuqori qavatli) sinchsiz bino konstruksiyalaridan bo’lgan uy-joy va jamoat binolarining montaji.

Bajardi: Oltiboyeva J 208g

Tekshirdi: Xasanov A

Pastdarg’om - 2015

Mavzu: C++ Dasturlash tilida massivlar ustida amallar bajarish.

Reja:

1. **Umumiy ma’lumotlar.**
2. **Massivlarni navlarga ajratish.**
3. **Oddiy tanlash usuli bilan navlarga ajratish.**
4. **Ko’p o’lchamli massivlar.**
5. **Belgili axborot va satrlar.**

Massiv tushunchasi

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

type a[length];

Bu maxsus a[0], a[1], ..., a[length-1] nomlarga yega bo'lgan type turidagi o'zgaruvchilarning ye'lon qilinishiga to'g'ri keladi. Massivning har bir yelementi o'z raqamiga - indeksga yega. Massivning x-nchi yelementiga kirish indekslash operasiyasi yordamida amalga oshiriladi:

```
int x=...;                  //butun sonli indeks
TYPE valuye=a[x];       //ch-nchi yelementni o'qish
a[x]=valuye;              //x-yxb yelementga yozish
```

Indeks sifatida butun tur qiymatini chiqarib beradigan har qanday ifoda qo'llanishi mumkin: char, short, int, long. Si da massiv yelementlarining indekslari 0 dan boshlanadi (1 dan yemas), LENGTH yelementdan iborat bo'lgan massivning oxirgi yelementining indeksi yesa - bu LENGTH-1 (LENGTH yemas). S'Huning uchun massivning barcha yelementlari 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 yelementni o'qish/yozishga urinish) dastur xulq-atvorida kutilmagan natijalarga olib kelishi mumkin. S'Huni ta'kidlab o'tamizki, bu yeng ko'p tarqalgan xatolardan biridir.

Statik massivlarni nomlab ye'lon qilish mumkin, bunda massivlar yelementlarining qiymatlari vergul bilan ajratilgan shakldor qavs {} ichida sanab o'tiladi. Agar massiv uzunligiga qaraganda kamroq yelement berilgan bo'lsa, qolgan yelementlar 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 obyektlarni 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 yega. Murakkablashtirilgan usullar kamroq sonli operasiyalarni talab qiladi, biroq operasiyalarning o'zi murakkabroq, shuning uchun uncha katta bo'lmagan massivlar uchun oddiy usullar ko'proq samara beradi.

Oddiy sullar uchta asosiy kategoriyaga bo'linadi:

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

Oddiy kiritish usuli bilan navlarga ajratish

Massiv yelementlari avvaldan tayyor berilgan va dastlabki ketma-ketliklarga bo'linadi. I=2 dan boshlab, har bir qadamda dastlabki ketma-ketlikdan I-nchi yelement 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 pozisiyadan tanlab olingan yelementni uzatish amalga oshiriladi, ya'ni tanlab olingan yelement, J:=I-1 dan boshlab, navlarga ajratib bo'lingan qismning navbatdagi yelementi bilan qiyoslanadi. Agar tanlab olingan yelement a[I] dan katta bo'lsa, uni navlarga ajratish qismiga qo'shadilar, aks holda a[J] bitta pozisiyaga suriladi, tanlab olingan yelementni yesa navlarga ajratilgan ketma-ketlikning navbatdagi yelementi bilan qiyoslaydilar. To'g'ri keladigan joyni qidirish jarayoni ikkita turlicha shart bilan tugallanadi:

- agar a[J]>a[I] yelementi topilgan bo'lsa;
- agar tayyor ketma-ketlikning chap uchiga yetilgan bo'lsa.

```
int i, j, x;
fjr(i=1; i<n; i++)
{
x=[i];// kiritib qo'ishimiz lozim bo'lgan yelementni yesda 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;//yelementni kiritish
}
```

Oddiy tanlash usuli bilan navlarga ajratish;

Massivning minimal yelementi tanlanadi hamda massivning birinchi yelementi bilan joy almashtiriladi. Keyin jaranyon qolgan yelementlar 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 yeng kichik yelementi uning yeng chapki yelementiga aylanadi. Jarayon massivning qolgan yelementlari bilan davom yettiriladi.

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;}
}
```

Ko'p o'lchamli massivlar

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

Bunday massivlarni ye'lon qilishda kompyuter xotirasida bir nechta turli xildagi obyekt 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 ye'lon qilinishi dasturda uchta turli xildagi obyektlarni 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 yega bo'lish uchun arr ko'rsatkichli adresli ifodalar qo'llanadi. Ko'rsatkichlar massivi yelementlariga 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 yekvivalent 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 yemas. Uch o'lchamli massivni joylashtirish ham xuddi shunga o'xshash amalga oshiriladi hamda float arr3[3][4][5] ning ye'lon qilinishi dasturda, float turidagi oltmishta sondan iborat uch o'lchamli massivning o'zidan tashqari, float òóðèãà tuzilgan òo'rtta ko'rsatkichdan iborat massivni, float êo'rsatkichlar massiviga tuzilgan uchta ko'rsatkichdan iborat massivni va float ãà tuzilgan êo'rsatkichlar massivining massivlariga ko'rsatkichni yuzaga keltiradi. Ko'p o'lchamli massivlar yelementlarini joylashtirishda ular xotirada satrlar bo'yicha bir tartibda joylashtiriladi., ya'ni oxirgi indeks hammadan tezroq o'zgaradi, birinchisi yesa sekinroq o'zgaradi. Bunday tartib, ko'p o'lchamli massiv boshlang'ich yelementining adresini hamda faqat bitta indeks ifodasini qo'llab, ko'p o'lchamli massivning har qanday yelementiga murojaat qilish imkonini beradi.

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

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

Massivlar ustida amallar

Massivlarning qo'lanilishiga doir misollar ko'rib o'tamiz. 1 - **m i s** o 1.

$$S = \sum_{j=1}^{10} \left[\prod_{i=1}^{5} b_{ij} + \sum_{i=1}^{5} b_{ij} \right].$$

Dasturini tuzamiz:

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

```
{ float p,c;
int i , j;
double b[\Q] [5]
printf(«\ n   Massiv elementlarini kiriting») ;
for (/=1 ;/<10 ;./++)

for ( i=l ; i<5 ; i++)
{
printf(«6[ %d ] [ %d]=»,j,i )
scanf(« %f %f , & x[ /] [?]»)

 for ( s=0.0 , /=0, ; /< 10 ; /++) { ior (p=1.0 i c=0. 0 ,
  ;=0 ; /<5

        P*= a[ /] [/] ; c += a[ /] [«] ;

        s+= c+p ; }
```

2 - mi s o 1.

Bir o'lchamli massiv elementlarini tartiblash. *a(n)* 1< *n*< 100 massivning elementlarini qiymat-lari o'sib borish tartibida joylashtiring. # include <stdio./z> main (){ int *n , i, j* ; floata[100],6 ; while (1) {

```
printf(«\n   Elementlar sonini kiriting n=») ;
scanf(«%d»,& n)\
if ( n> 1 && < = 100) break ;
printf(«xato! Kn< = 100 bo'lishi zarur !»)    ; }
printf(«\/i massiv elementlari qiymatlarini   kiriting  )
```

```
for ( /=0 ; j

    printf( a [%d ],
    scanf(«%f» , &a[

for (i=0; i<n—l; i
    for (j=i
            ])

                    b= a[i\; i
            } printf(«\n Tartiblangan massiv: ) n»);
                    for 0-0; j<n; /++) printf(«a(%rf)=%f
            \n»,/+l,a[/]);
```

Dasturning bajarilishi
Elementlar sonini kiriting n=3
a[l]=15.8
fl[2]=11.2
a[3]=—2.3
Tartiblangan massiv:
c[l]=—2.3
a[2]=11.2
a[3]=15.8

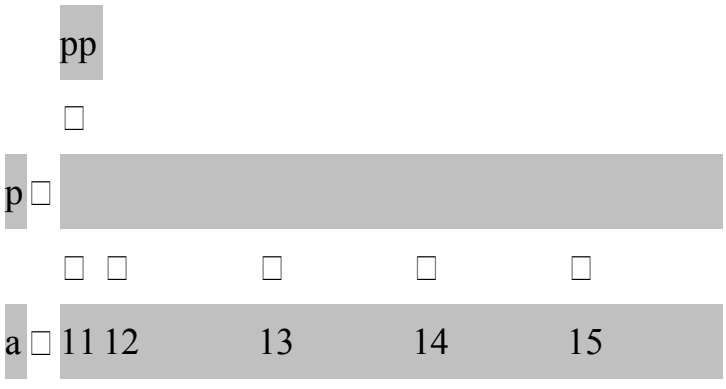
Ko'rsatkichlar massilari

Si tilida massivlar yelementlari har qanday turga yega bo'lishi mumkin, xususan, har qanday turdagi ko'rsatkichlar bo'lishi mumkin. Ko'rsatkichlar qo'llangan bir nechta misolni ko'rib chiqamiz.

¤zgaruvchilarning quyidagi ye'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 obyektlarni yuzaga keltiradi.

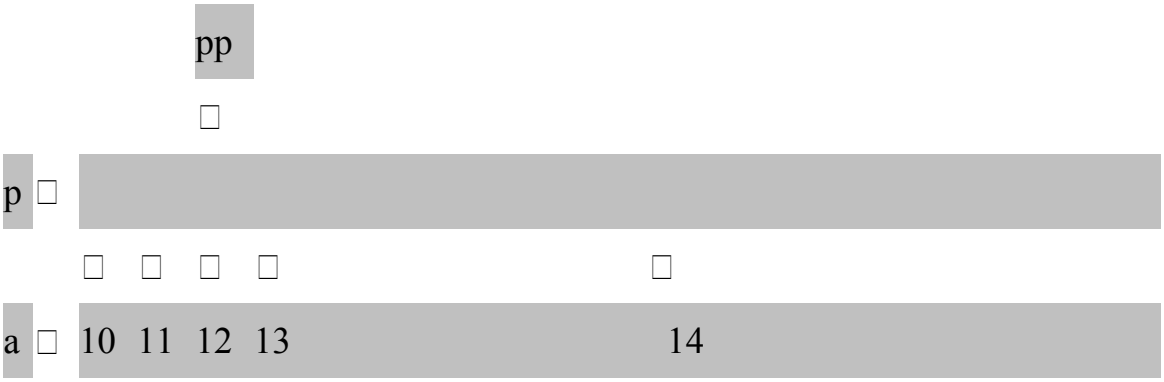


Ye'lon qilishda o'zgaruvchilarni

ylashtirish sxemasi.

Rr-r operasiyasini bajarishda nol qiymatga yega bo'lamiz, chnuki rr va r iqtiboslari o'zaro teng hamda ko'rsatkichlar massivining boshlanQich yelementiga ishora qiladi. Bu yerda ko'rsatkichlar massivi r ko'rsatkichi (r[0] yelementga) bilan boQliq.

Rr+=2 operasiyasi bajarilgandan keyin, sxema o'zgaradi hamda quyidagi tasvirlangandek ko'rinishga yega bo'ladi.



rr+=2 operasiyasi bajarilgandan keyin o'zgaruvchilarni joylashtirish sxemasi.

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

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

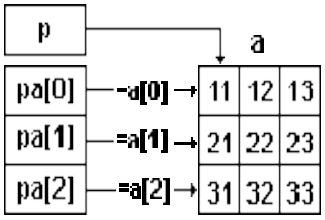
Murakkab murojaatlar ichidan turib ochiladi. Masalan, *(++(*rr)) murojaatini quyidagi amallarga bo'lish mumkin: *rr r[0] massivi boshlang'ich yelementining qiymatini beradi, keyin bu qiymat ++(*r) ga inkrementasiya bo'ladi, buning natijasida r[0] ko'rsatkichi a[1] yelementi adresining qiymatiga teng 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 yedi. Yendi ko'p o'lchamli massiv va ko'rsatkichlar berilgan misolni ko'rib chiqamiz. Quyidagi o'zgaruvchilarning ye'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 obyektlarni yuzaga keltiradi



Dinamik massivlar

C++tilida o'zgaruvchilar yo statik tarzda - kompilyasiya paytida, yoki standart kutubxonadan funksiyalarni 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 obyektning 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 yelementlarni 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 obyektlar nomlangan o'zgaruvchilar bilan belgilanadi, hamda ushbu obyektlar o'rtasidagi amallar to'g'ridan-to'g'ri, ularning nomlaridan foydalangan holda, amalga oshiriladi. Dinamik obyektlar o'z shaxsiy otlariga yega bo'lmaydi, va ular ustidagi amallar bilvosita, ko'rsatkichlar yordamida, amalga oshiriladi;
- statik obyektlar uchun xotirani ajratish va bo'shatish kompyuter tomonidan avtomatik tarzda amalga oshiriladi. Dasturchi bu haqda o'zi qayg'urishi kerak yemas. Statik obyektlar uchun xotirani ajratish va bo'shatish to'laligicha dasturchi zimmasiga yuklatiladi. Bu anchayin qiyin masala va uni yechishda xatoga yo'l qo'yish oson.

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

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