21-mavzu: Dasturlash tilining asosiy konstruktsiyalari, ulardan foydalanish xususiyatlari

REJA:

1. **Massiv tushunchasi. Statik massivlar.**
2. **Ko'rsatkichlar. Ko'rsatkichlarni o'zlashtirish.**
3. **Dinamik massivlar.**

Kompyuter xotirasida ketma-ket (regulyar) joylashgan bir xil turdagi qiymatlarga *massiv* deyiladi.

Katta hajmdagi cheklangan va tartiblangan qiymatlarni qayta ishlash bilan bog'liq masalalarni echishda massivlardan foydalanishga zarurat tuьiladi.

Massivlarni matematikadagi sonlar vektoriga o'xshatish mumkin, chunki vektor ham o'zining individual nomiga ega va u chekli miqdordagi bir turdagi qiymatlardan, ya'ni sonlardan iborat.

Demak, *massiv* – bu chekli miqdordagi bir turdagi qiymatlar (massiv elementlari)ning tartiblangan majmuasidir. Ya'ni massivdagi barcha elementlar bir xil turga tegishli bo'lishi lozim. Bunday tur massiv elementining turi yoki massiv uchun tayanch tur deb ataladi. Dasturda ishlatiladigan har bir massiv o'zining individual nomiga ega bo'lishi kerak. Bu nom massiv o'zgaruvchisi deyiladi. Massivning har bir elementi massiv nomi, hamda kvadrat qavsga olingan va element selektori deb nomlanuvchi indeksni ko'rsatish orqali oshkor ravishda belgilanadi. Unga murojaat sintaksisi quyidagicha:

<massiv nomi >[<indeks>];

Bu ko'rinishga xususiy o'zgaruvchi deyiladi, chunki uning qiymati massivning alohida elementidir.

Umuman olganda indeks sifatida ifoda ishlatilishi mumkin. Ifoda qiymati massiv elementi nomerini aniqlaydi. Ifoda sifatida o'zgaruvchi ham olinishi mumkin, bunda o'zgaruvchining qiymati o'zgarishi bilan murojaat qilinayotgan massiv elementini aniqlovchi indeks ham o'zgaradi. Shunday qilib, dasturdagi bitta indeksli o'zgaruvchi orqali massivning barcha elementlarini aniqlash mumkin.

Haqiqiy turdagi (*float, double*) qiymatlar to'plami cheksiz bo'lganligi sababli ular massiv indeksi sifatida ishlatilmaydi.

Bu bo'limda bir o'lchovli *statik* va *dinamik* massivlar bilan tanishamiz.

C++ tilida massiv indeksi doimo 0(nol) dan boshlanadi, uning eng katta qiymati massiv e'lonidagi uzunlikdan bittaga kam bo'ladi.

Massiv quyidagicha e'lon qilinadi:

<tur> <massiv o'zgaruvchisi nomi> [<o'lchami>] ={<boshlanьich qiymatlar>};

Bu erda *<o'lcham>* - butun son ko'rinishidagi o'zgarmas ifoda. Bir o'lchovli massivlarni e'lon qilishga doir misollar:

int a[5]={4,-5,2,10,3};

float n[4];

Massiv *statik* yoki *dinamik* turda bo'lishi mumkin. *Statik* massivning uzunligi oldindan ma'lum bo'ladi va uning elementlari xotirada aniq bir adresdan boshlab ketma-ket joylashadi. *Dinamik* massivning uzunligi dastur bajarilishi jarayonida aniqlanadi va uning elementlari dinamik xotirada ayni paytda bo'sh bo'lgan adreslarga joylashadi. *Masalan,*

int a[5]; ko'rinishida e'lon qilingan bir o'lchovli massiv elementlari xotirada quyidagicha joylashadi:

Adres	***Qiymatlar***					
a	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

26.1-jadval. Bir o'lchovli massivning xotiradagi joylashuvi.

Massivning *i*-elementiga *a[i]* yoki **(a+i)* – vositali murojaat qilish mumkin. Massiv uzunligi *sizeof(a)* amali orqali aniqlanadi.

Massiv e’lonida uning elementlariga boshlanьich qiymatlar berish (initsializatsiyalash) mumkin va uning bir nechta variantlari mavjud.

1) O’lchami ko’rsatilgan massiv elementlarini to’liq initsializatsiyalash:

int b[5]={8,5,-11,41,39};

Bunda 5 ta elementdan iborat bo’lgan b nomli bir o’lchovli massiv e’lon qilingan va uning barcha elementlariga boshlanьich qiymatlar berilgan. Bu e’lon quyidagi e’lon bilan ekvivalent:

int b[5];

b[0]=8; b[1]=5; b[2]=-11; b[3]=41; b[4]=39;

2) O’lchami ko’rsatilgan massiv elementlarini to’liqmas initsializatsiyalash:

int b[5]={-11,5,29};

Bu erda faqat massiv boshidagi uchta elementga boshlanьich qiymatlar berilgan. Shuni aytib o’tish kerakki, massivni initsializatsiyalashda uning boshidagi ementlariga boshlanьich qiymatlar bermasdan turib, oxiridagi elementlariga boshlanьich qiymatlar berish mumkin emas. Agarda massiv elementlariga boshlanьich qiymat berilmasa, unda kelishuv bo’yicha *static* va *extern* modifikatori bilan e’lon qilingan massiv uchun elementlarning boshlanьich qiymati *0(nol)* soniga teng bo’ladi. Automatik massiv elementlarining boshlanьich qiymatlari esa noma’lum hisoblanadi.

3) O’lchami ko’rsatilmagan massiv elementlarini to’liq initsializatsiyalash:

int d[]={-15,7,15,24};

Bu misolda massivning barcha elementlariga qiymatlar berilgan hisoblanadi, massiv uzunligi esa kompilyator tomonidan boshlanьich qiymatlar soniga qarab aniqlanadi.

Shuni ta’kidlash kerakki, massivning uzunligi berilmasa, unga boshlanьich qiymatlar berilishi shart.

Bir o’lchovli statik massivlarni e’lon qilish:

char s[5]={’d’,‘a’,‘b’,’c’,’e’}; //belgilar massivi

int n[5] ={1,26,34,14}; // butun sonlar massivi

*char str[]=”defa”; /*satr uzunligi 5 ga teng, chunki, uning oxiriga ‘\0’ belgisi qo’shiladi.*/*

*char str[]={‘d’,‘e’,‘f’,’a’}; /*yuqoridagi satrning boshqacha yozilishi.*/*

Statik massivlarning kamchiliklari shundaki, ularning o’lchami oldindan ma’lum bo’lishi kerak, bundan tashqari bu o’lcham berilganlarga ajratilgan xotira segmentining o’lchami bilan chegaralangan. Ikkinchi tomondan, etarlicha katta o’lchamdagi massiv e’lon qilib, masala echilishida ajratilgan xotira to’liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular dastur ishlashi jarayonida kerakli o’lchamdagi massivlarni yaratish va ularga zarurat qolmaganda yo’qotish imkoniyatini beradi.

Dinamik massivlarga xotiradan joy ajratish uchun *malloc(), calloc()* funksiyalaridan yoki new operatoridan foydalaniladi. Dinamik massivga ajratilgan xotirani bo’shatish uchun *free()* funksiyasi yoki *delete* operatori ishlatiladi.

Yuqorida qayd qilingan funksiyalar *alloc.h* kutubxonasida joylashgan.

malloc() funksiyasining sintaksisi

*void * malloc(size_t ulcham) ;*

ko’rinishida bo’lib, u xotiraning uyum qismidan *‘ulcham’* bayt o’lchamidagi uzluksiz sohani ajratadi. Agar xotira ajratish muvaffaqiyatli bo’lsa, *malloc()* funksiyasi shu soha boshlanishining adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo’lsa, funksiya *NULL* qiymatini qaytaradi.

Sintaksisdan ko’rinib turibdiki, funksiya *void* turidagi qiymat qaytaradi. Amalda esa aniq bir turdagi massiv ob’ekti uchun xotiradan joy ajratish zarur bo’ladi. Shu bois *void* turini aniq bir turga keltirish texnologiyasidan foydalaniladi. Masalan, butun turdagi uzunligi 3 ga teng massivga joy ajratishni quyidagicha amalga oshirish mumkin:

*int * d_mas=(int*)malloc(3*sizeof(int));*

malloc() funksiyasidan farqli ravishda *calloc*() funksiyasi massiv uchun joy ajratishdan tashqari massiv elementlarini *0*(nol) qiymati bilan initsializatsiyalaydi. Bu funksiya sintaksisi quyidagi:

*void * calloc (size_t miqdor, size_t ulcham) ;*

ko’rinishda bo’lib, *‘miqdor’* parametri ajratilgan sohada nechta element borligini, *‘ulcham’* esa element o’lchamini bildiradi.

free() dinamik xotirani bo’shatish funksiyasi bo’lib, ko’rsatilgan dinamik massiv egallab turgan xotira qismini bo’shatadi:

*void free (void * blok);*

free() funksiyasi parametrining *void* turida bo’lishi, ixtiyoriy turdagi xotira bo’lagini o’chirish imkonini beradi.

Quyidagi dasturda 10 ta elementga ega dinamik massivni yaratish, unga dastlabki 10 ta toq sonni qiymatlash va massiv elementlari qiymatini chop etish hamda shu massivni xotiradan o’chirish jarayonlari amalga oshirilgan.

```
#include <iostream.h>
#include <alloc.h>
int main()
{
int * d_mas;
if ((d_mas=(int*)malloc(10*sizeof(int)))==NULL)
{
cout<<”Xotira etarli emas!!!”;
return 1;
}
// ajratilgan xotira sohasini to’ldirish
for(int i=0; i<10; i++) * (d_mas+i)=2*i+1;
// dinamik massiv elementlarini chop etish
for(int i=0; i<10; i++)  cout<<*(d_mas+i)<<”end1”;
// ajratilgan xotira qismini bo’shatish
free (d_mas);
return 0;
}
```

Endi *new* va *delete* operatorlari bilan tanishamiz. *new* operatori yordamida massivga xotiradan joy ajratish uchun massiv turidan keyin kvadrat qavs ichida

massiv elementlari soni ko’rsatiladi. *Masalan,* butun turdagi 10 ta sondan iborat massivga joy ajratish quyidagi

d_mas=new int[10];

ko’rinishda bo’ladi. Bu usulda massivga ajratilgan xotirani bo’shatish uchun

delete [] d_mas;

buyruьini berish kerak bo’ladi.

Massivlar bilan ishlashda quyidagilarni hisobga olish zarur:

- massiv elementlariga qiymat kiritish siklllar yordamida amalga oshirilishi maqsadga muvofiq;

- statik massivlar bilan ishlashda kiritiladigan ma'lumotlar soni, massiv elementlari sonidan oshmasligi kerak;

- massiv elementlaridagi barcha ma'lumotlarni chiqarish uchun ham siklllardan foydalanish maqsadga muvofiq;

Quyida keltiriladigan masalada bu tasdiqlar o’z aksini topgan. Shuni alohida ta'kidlash lozimki, *for* takrorlanish operatorini algoritmda tasvirlash uchun qo’llanmada ikki xil shakldan foydalanilgan, chunki C++ tilida *for* operatori murakkab konstruksiyali bo’lib, uning parametrlariga tur nuqtai-nazaridan cheklovlar qoýilmaydi.

Qo’llanmaning 3-bobida takrorlanish operatorini tasvirlash uchun «*romb*» shaklidan foydalanilgan. Quyida keltirilgan masalaning algoritmida esa takrorlanish operatorini tasvirlash uchun «*oltiburchak*» shaklidan foydalanilgan. Chunki bu masalada barcha takrorlanish qadamlari 1 ga teng.

Yuqoridagilardan xulosa qiladigan bo’lsak, takrorlanishlar qadami 1 ga teng bo’lgan hol uchun algoritmlarni tasvirlashda «*oltiburchak*» shaklidan, qolgan hollarda esa «*romb*» shaklidan foydalanish maqsadga muvofiq bo’ladi.

Massiv bir xil turdagi bir nechta o’zgaruvchilarning to’plamidan tashkil topadi (ba’zi adabiyotlarda ular jadvallar deb ham nomlanadi). Nomi **a** bo’lgan LENGTH elementdan iborat TYPE turidagi massiv quyidagicha e’lon qilinadi:

type a[length];

Bu buyruqda type turiga tegishli maxsus a[0], a[1], ..., a[length-1] nomli o'zgaruvchilar e'lon qilinadi. Massivning har bir elementi o'z nomeri - indeksga ega bo'ladi. Massivning x - elementiga murojat qilish indeksatsiya amali yordamida amalga oshiriladi:

```
int x = ... ;                  // Butun qiymatli indeks
TYPE value = a[x];             // x - elementni o'qish
a[x] = value;                  // x- elementga yozish
```

Indeks sifatida butun turdagi qiymat beruvchi ixtiyoriy ifodani ishlatish mumkin: char, short, int, long. C tilida massiv elementlarining indeksi 0 dan (1 dan emas) boshlanadi, LENGTH uzunlikdagi massivning oxirgi elementining indeksi esa LENGTH-1 (LENGTH emas). Shuning uchun ham massivning barcha elementlari bo'yicha sikl quyidagicha yoziladi:

```
TYPE a[LENGTH]; int indx;
for(indx=0; indx < LENGTH; indx++)
...a[indx]...;
```

Bu erda indx < LENGTH sharti indx <= LENGTH-1 ga teng kuchli. Massiv chegarasidan chiqish (mavjud bo'lmagan elementni o'qish/yozish) kutilmagan natijalarga va dastur ishida ham kutilmagan holatlarga olib kelishi mumkin. Bunday xatolar massivlar bilan ishlashdagi eng ko'p yo'l qo'yiladigan xatolar hisoblanadi.

Statik massivlarni uning elementlari qiymatlarini {} ichida vergul bilan ajratib yozish, ya'ni initsializatsiya qilish yo'li bilan ham e'lon qilish mumkin. Agar massiv uzunligidan kam elementlar berilgan bo'lsa, u holda qolgan elementlari nol deb hisoblanadi:

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

Agar massivlarni initsializatsiya qilishda uning o'lchovi berilmasa u kompilyator tomonidan hisoblanadi:

int a3[] = { 1, 2, 3 }; // Xuddi a3[3] kabi.

C++ tilida massiv elementlarining turiga cheklovlar qo'yilmaydi, lekin bu turlar chekli o'lchamdagi ob'ektlarning turi bo'lishi kerak. Chunki kompilyator massivning xotiradan qancha joy (bayt) egallashini bilishi zarur. Xususan, massiv komponentasi massiv bo'lishi ham mumkin, ya'ni «*vektorlar-vektori*» natijada matritsa deb nomlanuvchi ikki o'lchovli massivni hosil qiladi.

Agar matritsaning elementi ham vektor bo'lsa, uch o'lchamli massiv hosil bo'ladi. Shu yo'l bilan ixtiyoriy o'lchamdagi massivni hosil qilish mumkin.

Ikki o'lchovli *statik* massivning sintaksisi quyidagi ko'rinishda bo'ladi:

<tur><massiv o'zgaruvchisi nomi>[<o'lcham1>][<o'lcham2>];

Masalan, 5x15 o'lchamli haqiqiy sonlar qabul qiluvchi statik massivning e'loni quyidagicha bo'ladi:

float s_mas[5][15];

E'lon qilingan *s_mas* massiv o'zgaruvchisi 5 ta satr va 15 ta ustundan tashkil topgan matritsaga o'xshaydi, ya'ni uning 75 ta elementi mavjud.

Endi adres nuqtai–nazaridan ko'p o'lchovli massiv elementlariga murojaat qilishni ko'raylik. Quyidagi e'lonlar berilgan bo'lsin:

int a[4][5];

float b[3][2][4];

Birinchi e'londa ikki o'lchovli massiv, ya'ni 4 ta satr va 5 ta ustundan iborat matritsaga o'xshash massiv e'lon qilingan, ikkinchisida esa uch o'lchovli, ya'ni 3 ta satr, 2 ta ustun va 4 ta qavatdan iborat massiv e'lon qilingan.

Massiv elementlariga murojaat qilish uchun massiv nomidan keyin kvadrat qavsda har bir o'lcham uchun indeks yozilishi kerak, *masalan*, *a[i][j]* ko'rinishidagi asosiy murojaatdan tashqari, vositali murojaat qilish *(*(a+i)+j) yoki *(a[i]+j) va *b[i][j][k]* ko'rinishidagi asosiy murojaatdan tashqari, vositali murojaat qilish *(*(*(b+i)+j)+k) yoki *(*(b[i]+j)+k) yoki *(b[i][j]+k) ham mumkin.

Ko'p o'lchovli massivlarni initsializatsiyalash quyidagi misollar yordamida ko'rsatiladi:

int s_mas[3][4]={0,4,1,-5,2,-22,15,-46,18,98,12,-3};

int b[4][3]={{5,1,24},{1,34,-12},{76,-2,7},{7,-25,7}};

Birinchi e'londa boshlanьich qiymatlar ketma-ket yozilgan, ikkinchi e'londa esa qiymatlar guruhlashgan.

Endi ikki o'lchovli dinamik massivlar tashkil qilishni qarab chiqamiz. Bunday massivlarni tashkil etishning sintaksisi quyidagi:

*<tur>**<massiv o'zgaruvchisi nomi>;*

ko'rinishda bo'lgan *«ko'rsatkichga ko'rsatkich»* ishlatiladi.

Dastlab massiv satrlari soniga qarab ko'rsatkichlar massiviga dinamik xotiradan joy ajratiladi:

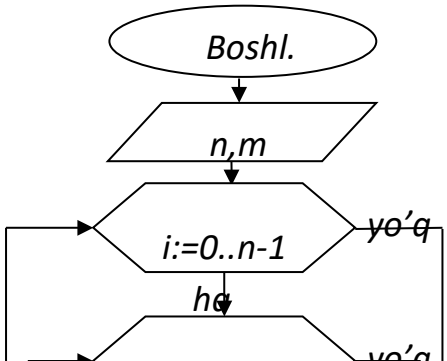
*d_mas=new int *[n]*; *// bu erda n massiv satrlari soni*

Keyin, dinamik massivning har bir satri uchun takrorlash operatoridan foydalanib xotiradan joy ajratish va ularning boshlanьich adreslarini *d_mas* massiv elementlariga joylashtirish zarur bo'ladi:

for(int i=0;i<n;i++) d_mas[i]=new int[m]; //m ustunlar soni

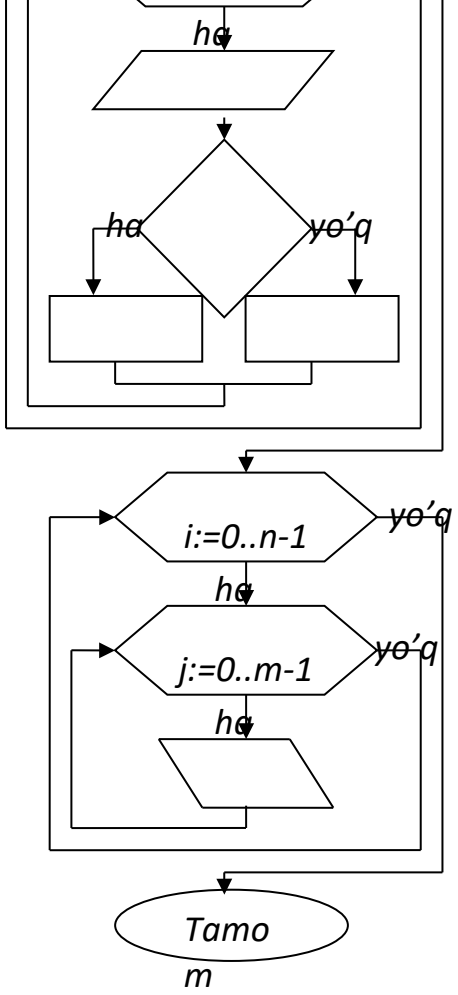
Shuni ta'kidlash kerakki, dinamik massivning har bir satri xotiraning turli joylarida joylashishi mumkin.

Ikki o'lchovli dinamik massivni o'chirish uchun esa oldin massivning har bir elementi (satri), so'ngra massivning o'zi o'chiriladi:



for (i=0;i<n;i++) delete [] d_mas[i];

delete [] d_mas;



Masala. O'lchamlari $n \times m$ bo'lgan ($n>0, m>0$) ikkita A va B matritsalar berilgan, ulardan foydalanib C matritsa tuzilsin: $C_{i,j} = \max(A_{i,j}, B_{i,j})$ (C matritsaning har bir elementi A va B matritsalardagi mos elementlarning kattasi hisoblanadi). Qo'yilgan masala uchun algoritm va dastur tuzilsin

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
 int n,m;
float a[100][100],b[100][100], c[100]
[100];
cin>>n>>m;
for(int i=0;i<=n-1;i++)
for(int j=0;j<=m-1;j++)
{
 cin>>a[i][j]>>b[i][j];
 if(a[i][j]>b[i][j])
 c[i][j]=a[i][j];
 else
 c[i][j]=b[i][j];
```

```
 }
for(int i=0;i<=n-1;i++)
for(int j=0;j<=m-1;j++)

cout<<c[i][j];

   system("PAUSE");
   return EXIT_SUCCESS;
 }
```