15-Ma’ruza. Massivlar

Ma’ruza rejasi:

15.1 Bir o’lchovli massivlar;

15.2 Massivlarni navlarga ajratish.

Kalit so’zlar:*, ro’yxat, manzil, nolinchi ko’rchsatkich, tugun, adres olish &, bo’shatish, ko’rsatkich, virtual destruktor, xotira, xotira chiqishi, destruktor, toifani o’zlashtirish, resurslar chiqishi, a’zo destruktori.*

15.1. Bir o’lchovli massivlar

Massiv tushunchasi. Massiv bu bir turli nomerlangan ma’lumotlar jamlanmasidir. Massiv indeksli o’zgaruvchi tushunchasiga mos keladi. Massiv ta’riflanganda turi, nomi va indekslar chegarasi ko’rsatiladi. Masalan, 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 murojaat indekslash operasiyasi yordamida amalga oshiriladi:

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

Indeks sifatida butun tur qiymatini qaytaradigan har qanday ifoda qo’llanishi mumkin: char, short, int, long. C 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) ga teng. Massivning int z[3] shakldagi ta’rifi, int turiga tegishli z[0], z[1], z[2] elementlardan iborat massivni aniqlaydi.

Massiv chegarasidan tashqariga chiqish (ya’ni mavjud bo’lmagan elementni o’qish/yozishga urinish) dastur bajarilishida kutilmagan natijalarga olib kelishi mumkin. Shuni ta’kidlab o’tish lozimki, bu eng ko’p tarqalgan xatolardan biridir.

Agar massiv inisializasiya qilinganda elementlar chegarasi ko’rsatilgan bo’lsa, ro’yxatdagi elementlar soni bu chegaradan kam bo’lishi mumkin, lekin ortiq bo’lishi mumkin emas.

Misol uchun int a[5] = {2,-2}. Bu holda a[0] va a[1] qiymatlari aniqlangan bo'lib, mos holda 2 va –2 ga teng. 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, kompilyator tomonidan massiv chegarasi avtomatik aniqlanadi:

int a3[] = {1, 2, 3};

Massivda musbat elementlar soni va summasini hisoblash.

```
#include<stdio.h>
int main()
{
int s = 0,k = 0;
int x[] = {-1,2,5,-4,8,9};
for(int i = 0; i<6; i++)
{
if (x[i]< = 0) continue;
k++;
s+ = x[i];
};
printf("%d\n",k);
printf("%d\n",s);
return 0;
};
```

Massivning eng katta, eng kichik elementi va o'rta qiymatini aniqlash:

```
#include<stdio.h>
int main()
{
int i,j,n;
float min,max,s = 0;
float a,b,d,x[100];
while(1)
{
printf("\n n = ");
scanf("%d",&n);
if ( n>0 && n< = 100 ) break;
printf("\n Hato 0<n<101 bo'lishi
kerak");
}
printf("\n elementlar qiymatlarini
kiriting:\n");
for (i = 0;i<n;i++)
{
printf("x[%d] = ",i);
scanf("%f",&x[i]);
}
max = x[0];
min = x[0];
for(i = 0;i<n;i++)
{
s+ = x[i];
if (max<x[i]) max = x[i];
if (min>x[i]) min = x[i];
};
```

```
s/ = n;                                  printf("\n o'rta qiymat = %f",s);
printf("\n max = %f",max);               return 0;
printf("\n min = %f",min);               };
```

15.2 Massivlarni navlarga ajratish

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

Massivlarning navlarga ajratilishi tez bajarilishiga ko'ra farqlanadi. Navlarga ajratishning n*n ta qiyoslashni talab qilgan oddiy usuli va n*log(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 operasiyalarni talab qiladi, biroq operasiyalarning o'zi murakkabroq, shuning uchun uncha katta bo'lmagan massivlar uchun oddiy usullar ko'proq samara beradi.

Oddiy usullar 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 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.

Kerakli joyni izlash jarayonida, ko'proq o'ngdan bitta pozisiyadan 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 pozisiyaga 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 bo'lsa.

```
int i, j, x;
for(i = 1; i < n; i++)
{
```

```
x = [i]; // kiritib qo‘yishimiz 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‘ngga surish
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 jarayon qolgan elementlar bilan takrorlanadi va h.k.

```
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.

```
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;}
```

```
}
```

Bir o’lchamli massivlarni funksiya parametrlari sifatida uzatish. Massivdan funksiya parametri sifatida foydalanganda, funksiyaning 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.

Misol:

Massiv barcha elementlari chiqarilsin:

```
#include <stdio.h>
#include <stdlib.h>
int form(int a[100])
{
int n;
printf("\nEnter n:");
scanf("%d",&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++)
printf("%d ", a[i]);
printf("\n");
}
int main()
{
int a[100];
int n;
n = form(a);
print(a,n);
return 0;
}
```

Funksiyaga massiv boshlanishi uchun ko’rsatkich uzatilgani tufayli (adres bo’yicha uzatish), funksiya tanasining operatorlari hisobiga massiv o’zgarishi mumkin.

Funksiyalarda bir o’lchovli sonli massivlar argument sifatida ishlatilganda ularning chegarasini ko’rsatish shart emas.

Misol. Massivdan barcha juft elementlar chiqarilsin.

```
#include <stdio.h>
#include <stdlib.h>
void form(int a[], int n)
{
for(int i = 0; i < n; i++)
a[i] = rand()%100;
```

```
}
void print(int a[],int n)
{
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
}
int Dell(int a[],int n)
{
int j = 0, i, b[100];
for(i = 0; i < n; i++)
if(a[i]%2! = 0)
{
b[j] = a[i]; j++;
}
for(i = 0; i < j; i++) a[i] = b[i];
return j;
}
int main()
{
int a[100];
int n;
printf("\nEnter n:");
scanf("%d",&n);
form(a, n);
print(a, n);
n = Dell(a, n);
print(a, n);
system("pause");
return 0;
}
```

Funksiyalarda bir o'lchovli sonli massivlar argument sifatida ishlatilganda ularning chegarasini ko'rsatish shart emas. Misol tariqasida n o'lchovli vektorlar bilan bog'liq funksiyalarni ta'riflashni ko'rib chiqamiz.

Vektorning uzunligini aniqlash funksiyasi:

```
float mod_vec(int n, float x[])
{
float a = 0;
for (int i = 0; i < n; i++)
 a+ = x[i] * x[i];
return sqrt(double(a));
}
```

Funksiyalarda bir o'lchovli massivlar qaytariluvchi qiymatlar sifatida ham kelishi mumkin. Misol uchun ikki vektor summasini hisoblovchi funksiya prosedurani ko'ramiz:

```
void sum_vec(int n, float a, float b, float c)
{
for(int i = 0;i < n; i++,c[i] = a[i] + b[i]);
}
```

Bu funksiyaga quyidagicha murojaat qilish mumkin:

```
float a[] = {1, -1.5, -2};
b[] = {-5.2, 1.3, -4},c[3];
sum_vec(3, a, b, c);
```

Polinom. Polinom qiymatini hisoblash funksiyasi poly deb nomlanadi.

Prototipi:

```
double poly(double x, int degree, double coeffs[]);
```

Algebraik polinom koeffisientlari coeffs[0], coeffs[1], ..., coeffs[degree] massiv elementlarida beriladi.

Misol:

```
#include <stdio.h>
#include <math.h>
/* polynomial: x**3 - 2x**2 + 5x - 1 */
int main(void)
{
double array[] = { -1.0, 5.0, -2.0, 1.0};
double result;
result = poly(2.0, 3, array);
printf("The polynomial: x**3 - 2.0x**2 + 5x - 1 at 2.0 is %lf\n", result);
return 0;
}
```