O`zbekiston Respublikasi Oliy va O`rta maxsus ta`lim vazirligi

Jizzax Davlat Pedagogika Instituti

“ Fizika - Matematika ” fakulteti

“ Informatika va Axborot texnologiyalari ” kafedrasi

“Algoritmlash va dasturlash asoslari ” fanidan

KURS ISHI

Bajardi: Avazov Z.

Qabul qildi: Sattorov A.

Komissiya a`zolari: _____

_____

_____

Jizzax - 2015 yil

C++ da massivlar bilan ishlash (kiritish, taxrirlash, tartiblash , . . .)

REJA:

I. Kirish

II. Asosiy qism . Massivlar bilan ishlash.

2.1 Massiv haqida umumiy tushuncha.

2.2 Bir o`lchovli massivlar.

2.3 Ko`p o`lchovli massivlar.

III. Massivlarning umumiy holda qo`llanilishi.

3.1 Dinamik massivlar.

3.2 Funksiyalarda massiv qo`llanishi.

IV. Xulosa.

VI. Foydalanilgan adabiyotlar.

Kirish

Ma`lumki, programma mashina kodlarining qandaydir ketma – ketligi bo`lib, aniq bir xisoblash vositasini amal qilishini boshqaradi. Programmata`minotini yaratish jarayonini osonlashtirish uchun yuzlab programma - lash tillari yaratilgan . Barcha programmalash tillarini ikki toifaga ajratish mumkin.

- **Quyi darajadagi programmalash tillari;**
- **Yuqori darajadagi programmalash tillari;**

Quyi darajadagi programmalash tillariga Assembler turidagi tillar kiradi . Bu tillar nisbatan qisqa va tezkor bajariluvchi kodlarni yaratish imkoniyatini beradi. Lekin, assembler tilida programma tuzish zaxmatli, nisbatan uzoq davom etadigan jarayondir . Bunga qarama – qarshi ravishda yuqori bosqich tillari yaratilganki , ularda tabiiy tilning cheklangan ko`rinishidan foydalangan holda programma tuziladi. Yuqori bosqich tillaridagi operatorlar , berilganlarning turlari , o`zgaruvchilar va programma yozishning turli usullari tilning ifodalash imkoniyatini oshiradi va programmani <<o`qimishli>> bo`lishini ta`minlaydi. Yuqori bosqich tillariga Fortran, PL/1, Prolog, Lisp, Basic, Pascal, C va boshqa tillarni misol keltirish mumkin. Kompyuter arxitekturasini takomillashuvi, kompyuter tarmog`ining rivojlanishi mos ravishda yuqori bosqich tillarini yangi variantlarini yuzaga kelishiga, yangi tillarni paydo bo`lishiga ayrim tillarni esa yo`qolib ketishiga olib keldi . Xozirda keng tarqalgan tillarga Object Pascal, C++, C#, Php, Java , Asp tillari xisoblanadi. Xususan , C tilining takomillashgan variant C++ tilini olishimiz mumkin.

1972 – yilda Denis Ritch va Brayan Kernegi tomonidan C tili yaratildi .

1980 – yilda Byarn Straustrop C tilining avlodi C++ tilini yaratdiki,

unda strukturali va obektga yo`naltirilgan programmalash texnologiyasiga tayangan xolda programma yaratish imkoniyati tug`ildi.

C++ funksiya va obyektlarning juda boy kutubxonasiga ega . Yani C++ da dasturlashni o`rganish ikki qismga bo`linadi . Birinchisi bu C++ ni o`zini o`rganish , ikkinchisi esa C++ ning standart kutubxonasidagi tayyor obyekt funksiyalarni qo`llashni o`rganishdir.

Massiv – bu bir toifali , chekli qiymatlarning tartiblangan to`plamidir . Massivlarga misol qilib matematika kursidan ma`lum bo`lgan vektorlar , matritsalarni ko`rsatish mumkin.

Massivlar odatda bir o`lchovli va ko`p o`lchovli turlarga bo`linadi.

Massiv bir o`lchamli deyiladi, agar uning elementiga bir indeks orqali murojat qilish mumkin bo`lsa.

C\C++ dasturlash tillaridagi massiv elementlar indekislari har doim noldan boshlanadi (birdan emas) .Bizga char tipidagi m nomli massiv berilgan bo`lsin . Va u 3 ta elementdan tashkil topgan bo`lsin.

m[0] → -9 ;
m[1] → 15;
m[2] → 3;

Demak, elementga murojat qilish uchun massiv nomi va [] qavslar ichida element indeksi yoziladi.

Bu yerda birinchi element qiymati -9 , ikkinchi element – 1 nomerli indeksda -15 qiymati bor ekan. Oxirgi element indeksi n-1 bo`ladi (n-massiv elementlari soni). [] qavs ichidagi indeks butun son yoki butun songa olib keluvchi ifoda bo`lmog`i lozim. Masalan:

```
int  n=6, m=4;
L[n-m]=33;  // L[2]=33;
Cout<<m[2];// ekranda : 3;
```

Massiv elementlariga murojat qilish oddiy o`zgaruvchilarga murojat qilishdan biroz farq qiladi . Massiv elementiga murojat qilish indeksi orqali bo`ladi.

```
a[1] = 5;  a massivning indeksi 1 bo`lgan elementi 5 qiymat o`zlashtirilsin.
cin>>a[2]; a massivning elementi 2 bo`lgan elementi kiritilsin;
cout<<a[3]; a massivning indeksi 3 bo`lgan elementi ekranga chiqarilsin;
```

Bir o`lchamli massivlarni e`lon quyidagicha bo`ladi :

```
<Toifa><massiv_nomi> [elementlar _soni] = { boshlang`ich qiymatlar };
1)float a[5]   2) int b[6]   3) boll c[7];
```

1) a elementi haqiqiy sondan iborat bo`lgan , 4 ta elementdan

tashkil topgan massiv. Indekslari esa 0 dan 3 gacha bo`lgan sonlar.

Float a[5]					
Massiv elementlari	a [0]	a [1]	a [2]	a [3]	a [4]
qiymati	4	11	-8	12	122

2) b elementi butun sondan iborat bo`lgan , 6 ta elementdan tashkil topgan massiv. Indekslari esa 0 dan 5 gacha bo`lgan sonlar.

int a[6]						
Massiv elementlari	a [0]	a [1]	a [2]	a [3]	a [4]	a [5]
qiymati	2	99	-5	28	112	54

3) c elementlari mantiqiy qiymatlardan (true, false) iborat bo`lgan 7 ta elementdan tashkil topgan massiv. Indekslari esa 0 dan 6 gacha bo`lgan sonlardir.

Massivni e`lon qilishda uning elementlariga boshlang`ich qiymat berish mumkin va buning bir necha usuli mavjud.

1) O`lchami ko`ratilgan massivni to`liq initsializatsiyalash.

```
int k[5] = {2, 15 , -9, 45, 3 , 7};
```

Bu yerda 5 ta elementdan iborat k massivi e`lon qilingan va massivning barcha elementlariga boshlang`ich qiymat berilgan.

2) O`lchami ko`rsatilgan massivni to`liqmas to`liqmas initsializatsiyalash.

```
int k[5] = {2, 15, -9 };
```

Bu yerda 5 ta elementdan iborat bo`lgan k massivi e`lon qilingan va dastlabki 3 ta elementlariga boshlang`ich qiymat berilgan.

3) O`lchami ko`rsatilmagan massivni to`liq initsializatsiyalash.

```
int k[] = {2, 15, -9, 45, 3, 7};
```

Shuni takidlash lozimki , agar massiv o`lchami ko`rsatilmasa , uni to`liq initsializatsiyalash shart. Bu xolda massiv o`lchami kompilyatsiya jarayonida massiv elementlar soniga qarab aniqlanadi. Bu yerda massiv o`lchami 5 ga teng.

4) O`lchami ko`rsatilgan massivning barcha elementlariga boshlang`ich qiymat 0 berish.

intk[5] = {0};

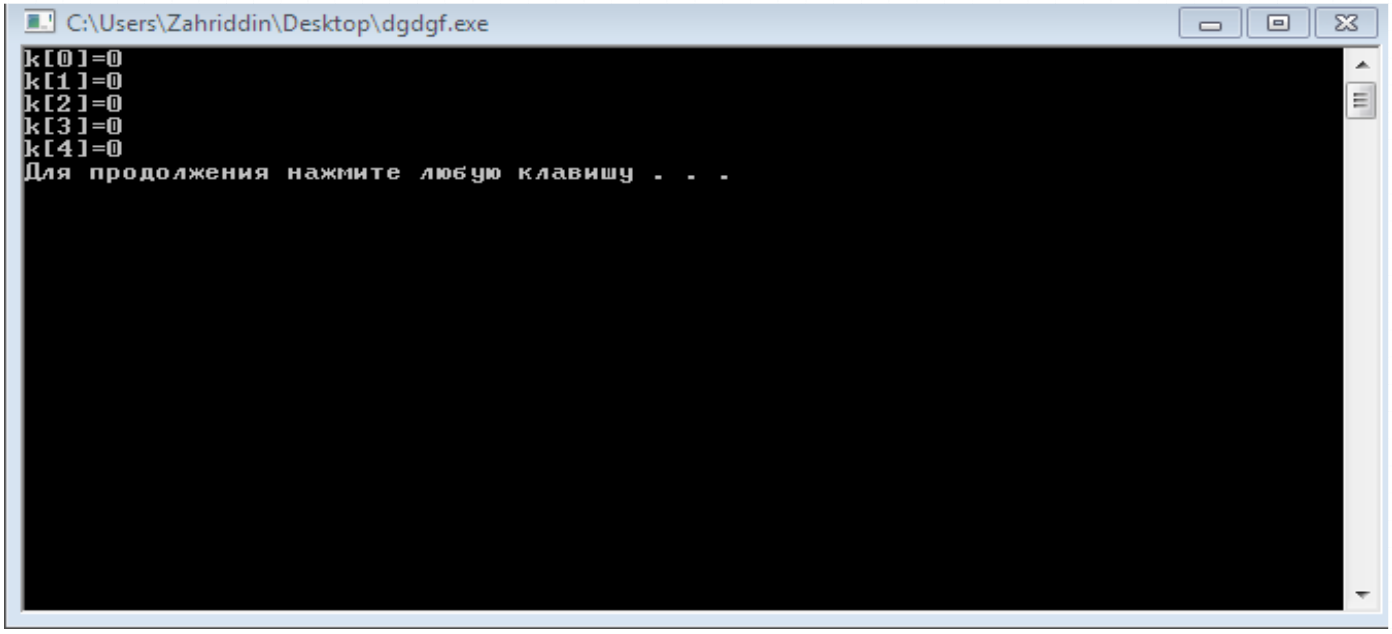
Masalan:

1-misol

O`lchami ko`rsatilgan massivning barcha elementlariga boshlang`ichqiymat 0 berish.

```
#include<iostream.h>

  int main ()
  {
     int k[5]={0};   //  massivning  barcha  elementlariga   0  qiymat  berish.

     for (int i=0; i<5; i++ )

    cout<<"k["<<i<<"]="<<k[i]<<endl;

     return 0;
  }
```

Ekranga quyidagicha natija chiqadi:

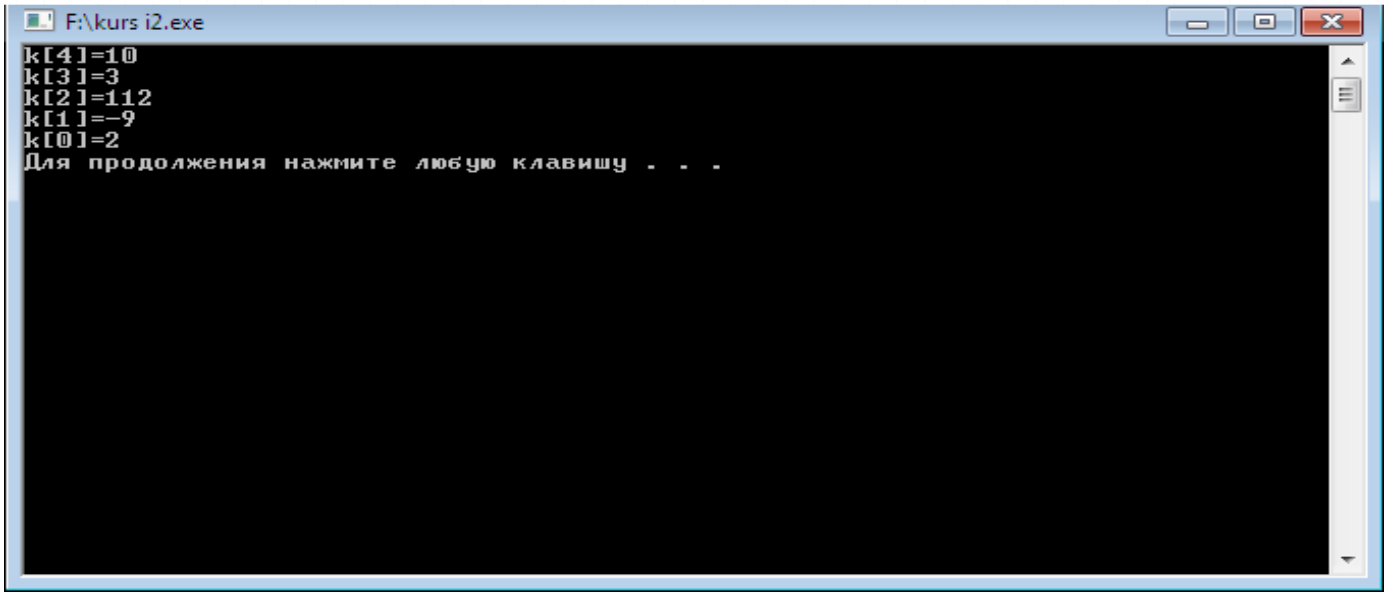


2-misol.

O`lchami ko`rsatilgan massivni to`liq initsializatsiyalash.

```
#include<iostream.h>

 int main ()
{
     int k[5] = { 2, -9, 112, 3, 8 };

    for (int i=4;  i>=0;  i-- )  //  indekslarini  teskari  tartibda  chop  etish.

     cout<<"k["<<i<<"]="<<k[i]<<endl;

    return 0;
 }
```

Ekranga quyidagicha natija chiqadi:

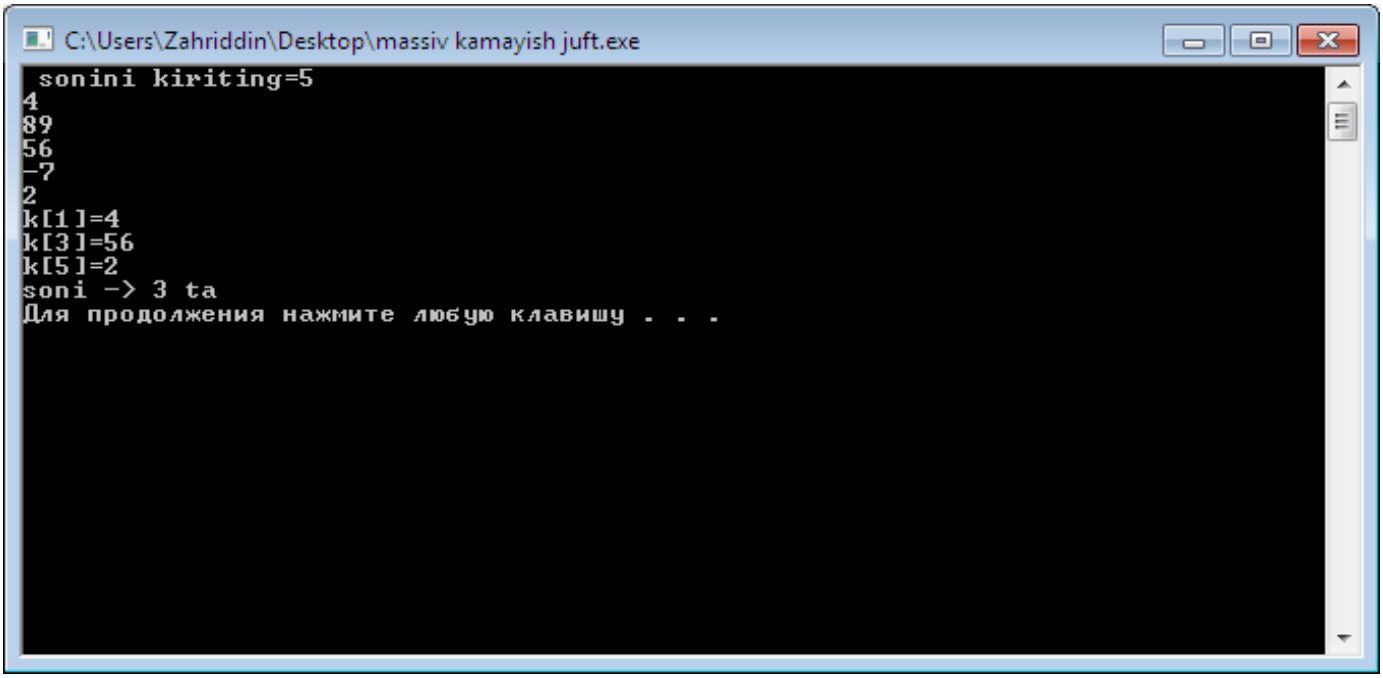


3-misol.

n o`lchamli butun sonlardan iborat massiv berilgan . Bu massivning toq elementlarini indekslarini o`sib borish tartibida chop etish va toq elementlar sonini hisoblash dasturi tuzilsin.

```
#include<iostream.h>

  int main ()

{
  int k[100];
  int i,n,s;
  cout<<" sonini kiriting=";
  cin>>n;
  for ( i=1; i<=n; i++)
  cin>>k[i];
  s=0;
  for (i=1; i<=n; i+=2)
  {
     cout<<"k["<<i<<"]="<<k[i]<<endl;
      s++;
         }
  cout<<"soni"<<" "<<"->"<<" "<<s<<" "<<"ta"<<endl;
  system("pause");
  return 0;
}
```

Ekranga quyidagicha natija chiqadi:



Ko`p o`lchovli statik massivlar

C++ tilida massivlar elementining turiga cheklovlar qo`yilmaydi , lekin bu turlar chekli o`lchamdagi obyektlarning turi bo`lishi kerak.
CHunki kompliyator massivning hotiradan qancha joy (bayt) egallashini xisoblay olish kerak. Xususan , massiv komponentasi massiv bo`lish mumkin (“vektorlar - vektori”) , natijada matritsa deb nomlanuvchi ikki o`lchamli massiv xosil bo`ladi.

Agar matritsaning elementi xam vektor bo`lsa , uch o`lchamli massivlar - kub xosil bo`ladi. Shu yo`l bilan yechilayotgan masalaga bog`liq ravishda ixtiyoriy o`lchamdagi massivlarni yaratish mumkin.
Ikki o`lchamli massivda birinchi indeks satrlar sonini , ikkinchisi esa ustunlar sonini bildiradi.
Birinchi satrning dastlabki elementi a_{10} – a biri nol element deb o`qiladi . a o`n deyilmaydi.
M ta satr n ta ustunga ega bo`lgan massivga (mxn)o`lchamli massiv deyiladi. Agar m=n (satrlar va ustunlar soni teng) bo`lsa kvadrat massiv deyiladi .

Ikki o`lchamli massivning sintaksi quyidagi ko`rinishda bo`ladi:

<tur><nom>[<uzunlik>][<uzunlik>]

Masalan, 10X20 o`lchamli xaqiqiy sonlar massivning e`loni:

Float a[10][20];

E`lon qilingan a matritsa ko`rinishi quyidagicha ko`rinishda bo`ladi.

J

$a_{[0]}$: ($a_{[0][0]}$, $a_{[0][2]}$, …. , ….$a_{[0][18]}$, $a_{[0][19]}$,)

$a_{[1]}$:($a_{[1][0]}$, $a_{[1][1]}$, …. , ….$a_{[1][18]}$, $a_{[1][19]}$,)

….

i $a_{[i]}$:(…, … , …. , ….$a_{[i][j]}$ …. , ……)

….

$a_{[9]}$:($a_{[9][0]}$, $a_{[9][1]}$, …. , …. $a_{[9][18]}$, $a_{[9][19]}$,).

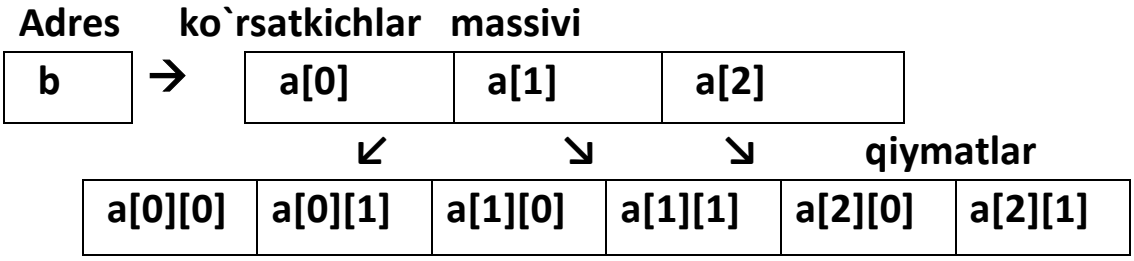
Ikki o`lchamli massivning hotirada joylashuvi

Endi adres nuqtayi - nazaridan ko`p o`lchamli massiv elementlariga murojat qilishni ko`raylik. Quyidagi elonlar berilgan bo`lsin:

Int a[3][2];

Float b[2][2][2];

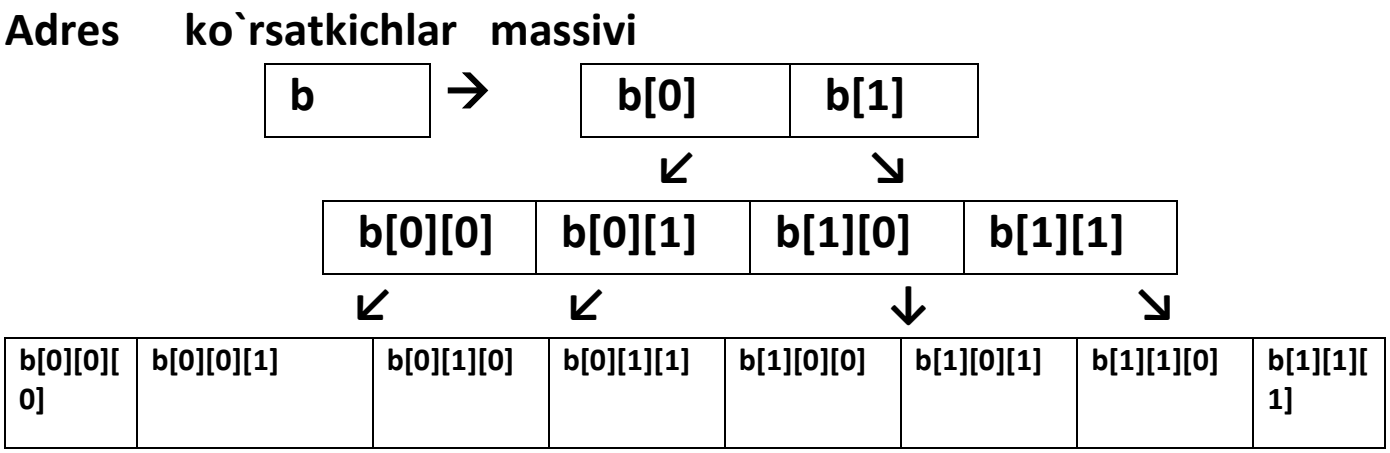
Birinchi elonda ikki o`lchamli massiv, yani 2 ta satr va 3 ustundan iborat matritsa e`lon qilingan , ikkinchisida uch o`lchamli - 3 ta 2x2 matritsadan iborat bo`lgan massiv e`lon qilingan . Uning elementlariga murojat sxemasi:



Ikki `olchamli massiv elementlariga murojat ;

Bu yerda a[i] ko`rsatkichida i-chi satrning boshlang`ich adresi joylashadi, massiv elementiga a[i][j] ko`rinishidagi asosiy murojatdan tashqari vositali murojat qilish mumkin: *(*(a+i)+j) yoki *(a[i]+j).

Uch o`lchamli massivning xotirada tashkil bo`lishi:



Massiv elementlariga murojat qilish uchun nomdan keyin kvadrat qavsda xar bir o`lcham uchun indeks yozilishi kerak , masalan b[i][j][k]. Bu elementga vositali murojat xam qilish mumkin va uning variantlari:
***(*(*(b+i)+j)+k) yoki *(*(b[i]+j)+k) yoki *(b[i][j]+k);**

Ko`p o`lchovli massivlarni initsializatsiyalash

```
Int  a[2][3] = {2, 6, 8,  7, 12, 5};
Int  b[3][3] = {{2, 6, 8}, {7, 12, 5}, {20, 21, 22 }}
```

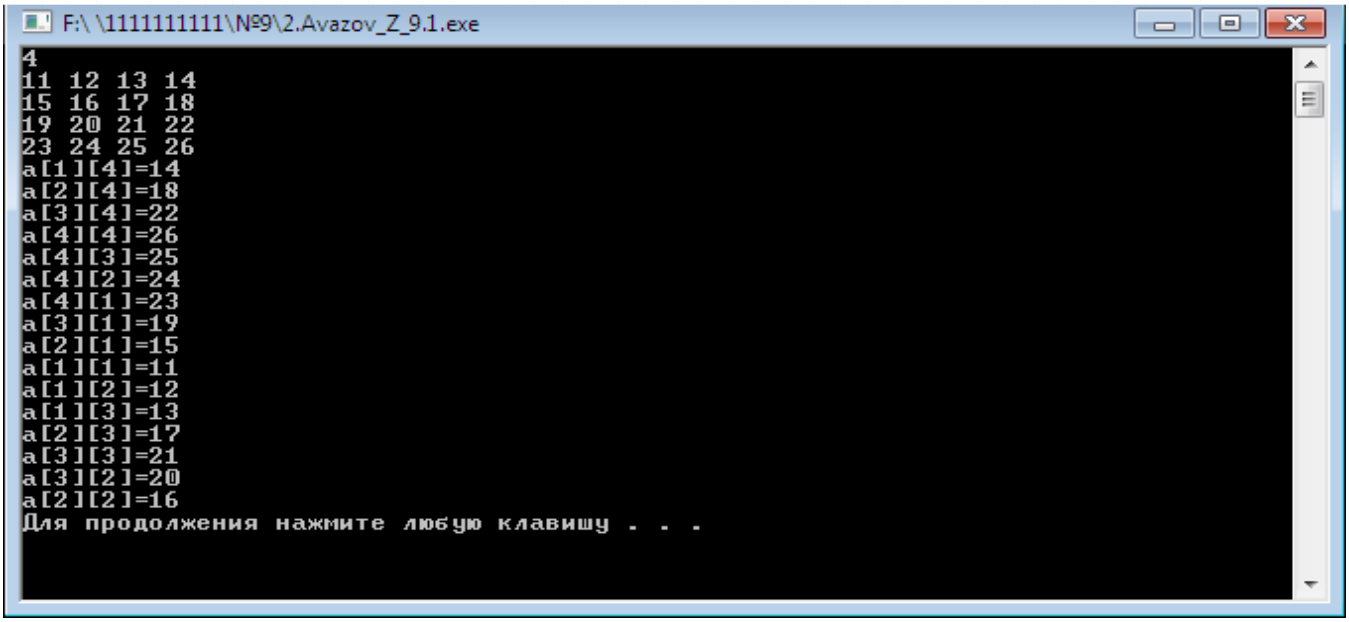
Birinchi operatorda boshlang`ich qiymatlar ketma – ket yozilgan, Ikkinchi operatorda qiymatlar guruxlangan.
Misollar:

1-misol.

M o`lchamli kvadrat matrisa berilgan . Bu massivning elementlarini spiral shaklida chop etish dasturi tuzilsin : avval oxirgi ustun , keyin oxirgi qator teskari tartibda , keyin birinchi ustun teskari tartibda, keyin birinchi qator. Ichki elementlar ham shu tartibda chop etiladi. Eng oxirida matrisaning markaziy elementi chop etiladi.

```
#include <iostream.h>
using namespace std;

int main()
{
  short k,i,j,m,x,y,z,w;
  float a[100][100];
  cin>>m;
  for(i=1;i<=m;i++)
  for(j=1;j<=m;j++)
     cin>>a[i][j];
  x=m; y=m;  z=1;    w=1;
 for(k=1;k<=m/2;k++)
 {
  for(i=z;i<=x;i++)
    cout<<"a["<<i<<"]["<<x<<"]="<<a[i][x]<<endl;

  for(j=y-1;j>=w;j--)
    cout<<"a["<<y<<"]["<<j<<"]="<<a[y][j]<<endl;

  for(i=x-1;i>=z;i--)
    cout<<"a["<<i<<"]["<<z<<"]="<<a[i][z]<<endl;

  for(j=w+1;j<=y-1;j++)
    cout<<"a["<<w<<"]["<<j<<"]="<<a[w][j]<<endl;
  x--;y--;z++;w++;
 }
 //  bu dastur toq sonlar uchun  ham  o`rinli
 if(m%2==1)
    cout<<"a["<<m/2+1<<"]["<<m/2+1<<"]="<<a[m/2+1][m/2+1]<<endl;
    system ("pause");
  return 0;
}
```

Ekranga quyidagicha natija chiqadi:

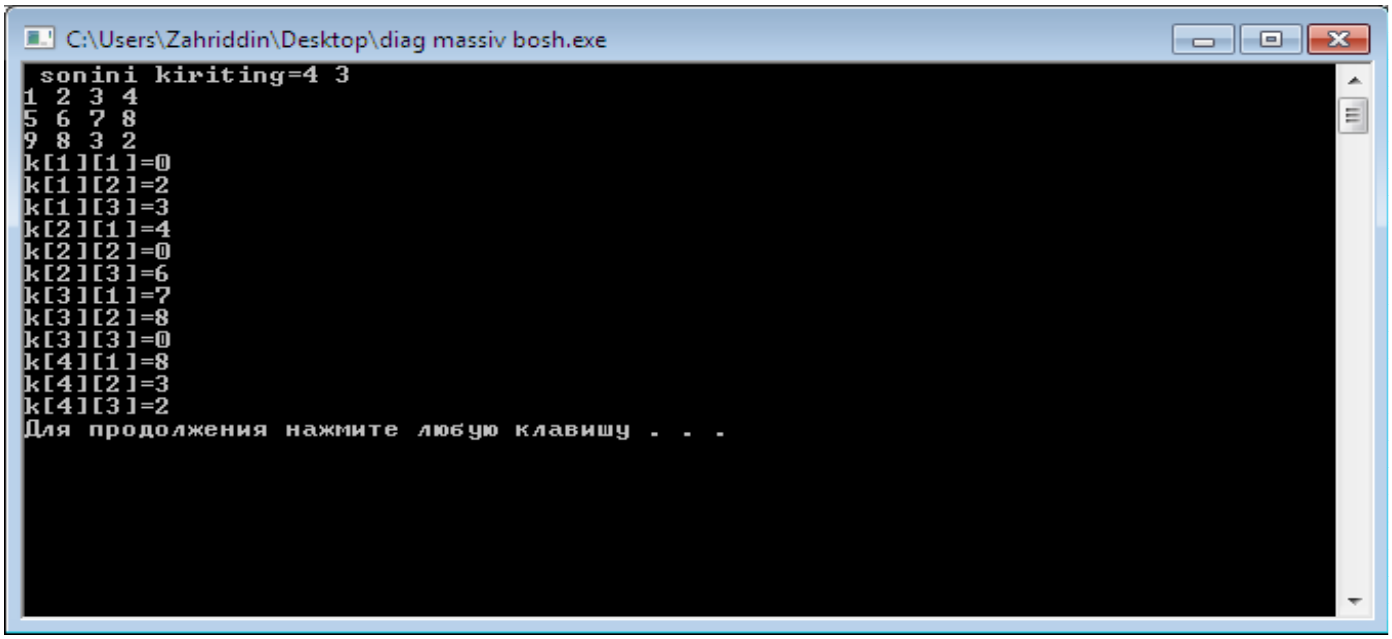


2-misol.

Berilgan mxn o`lchamli matrisaning bosh diaganali elementlarini nollarga aylantirish dasturi tuzilsin.

```
#include<iostream.h>
   int main ()

   {
   int k[100][100];
   int i,j,n,m;
   cout<<" sonini kiriting=";
   cin>>n>>m;
 for ( i=1; i<=n; i++)
   for ( j=1; j<=m; j++)
   cin>>k[i][j];
    for ( i=1; i<=n; i++)
     for ( j=1; j<=m; j++)
    {if (i==j)
    k[i][j]=0;
      cout<<"k["<<i<<"]["<<j<<"]="<<k[i][j]<<endl;}
   system("pause");
   return 0;
   }
```

Ekranga quyidagicha natija chiqadi:



Dinamik massivlar bilan ishlash.

Statik massivlarning kamchiliklari shundaki, ularning o`lchamlari oldindan ma`lum bo`lishi kerak, bundan tashqari bu o`lchamlar berilganlarga ajratilgan xotira segmentining o`lchami bilan chegaralangan . Ikkinchi tomondan, yetarlicha kata o`lchamdagi massiv e`lon qilinib, konkret masala yechilishida ajratilgan xotira to`liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular programma ishlashi jarayonida kerak bo`lgan o`lchamdagi massivlarni yaratish va zarurat qolmaganda yo`qotish imkoniyatini beradi.

Dinamik massivlarga xotira ajratish uchun malloc(), calloc() funksiyalaridan yoki neu operatoridan foydalanish mumkin. Dinamik obyektga ajratilgan xotirani bo`shatish uchun delete operatori ishlatiladi

Yuqorida qayd qilingan funksiyalar <<alloc.h>> kutubxonasida joylashgan.

Malloc() funksiyasining sintaksisi

Void * malloc(size_t size);

Ko`rinishida bo`lib , u hotiraning uyum qismidan size bayt o`lchamdagi uzluksiz sohani ajratadi. Agar xotira ajratish

muvaffaqiyatli bo`lsa, malloc() funksiyasi ajratilgan sohaning boshlanish adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatli bo`lsa , funksiya NULL qiymatni qaytaradi.

Sintaksisdan ko`rinib turibdiki, funksiya void turidagi qiymat qaytaradi. Amalda esa konkret turdagi obyekt uchun xotira ajratish zarur bo`ladi. Buning uchun void konkret turga keltirish texnologiyasidan foydalaniladi. Masalan , butun turdagi uzunligi 3 ga teng massivga joy ajratishni quyidagicha amalga oshirish mumkin:

```
Int * pint=(int*)malloc(3*sizeof(int));
```

Calloc() funksiyasi malloc funksiyasidan farqli ravishda massiv uchun joy ajratishdan tashqari massiv elementlarini 0 qiymati bilan initsializatsiya qiladi.

Bu funksiya sintaksisi .

```
Void * calloc(size_t num, size_ t size);
```

Ko`rinishida bo`lib , num parametri ajratilgan sohada nechta element borligini, size xar bir element o`lchamini bildiradi.

Free() xotirani bo`shatish funksiyasi o`chiriladigan xotira bo`lagiga ko`rsatkich bo`lgan yagona parametrga ega bo`ladi:

```
Void free(void* block);
```

Free() funksiyasi parametrining void turida bo`lishi ixtiyoriy turdagi xotira bo`lagini ochirish imkonini beradi .

Quyidagi programmada 10 ta butun sondan iborat dinamik massiv yaratish, unga qiymat berish va o`chirish amallari bajarilgan.

```
#include<iostream.h>
#include<alloc.h>
int main()
{
int *  pvector;
if ((pvector=(int*)malloc(10*sizeof(int)))==NULL)
{
 Cout<<”xotira  yetarli  emas!!!”;
Return 1;
}
// ajratilgan  xotira  soxasini  to`ldirish
For  (int  i=0; i<10; i++)  *(pvektor+i)=I;
```

```
// vector elementlarini hop etish
For (int i=0; i<10; i++) cout<<*(pvector+i)<<endl;
// ajratilgan xotira bo`lagini qaytarish (o`chirish)
Free(pvector);
Return 0;
}
```

new operatori yordamida ,massivga hotira ajratishda obyrkt turidan keyin kvadrat qavs ichida obyektlar soni ko`rsatiladi.

Masalan , butun turdagi 10 ta sondan iborat massivga joy ajratish uchun

```
pVector=new int[10];
```

ifodasi yozilishi kerak. Bunga qarama – qarshi ravishda , bu usulda ajratilgan xotirani bo`shatish uchun

```
delete [] pVector;
```

ko`rsatmasini berish kerak bo`ladi;

Ikki o`lchamli dinamik massivni hosil qilish uchun

```
int **a;
```

ko`rinishidagi <<ko`rsatkichga ko`rsatkich >> ishlatiladi.

Boshqa massiv satrlari soniga qarab ko`rsatkichlar massiviga dinamik xotiradan joy ajratish kerak:

```
A=new int *[m] // bu yerda m massiv satrlar soni
```

Keyin , xar bir satr uchun takrorlash operatori yordamida xotira ajratish va ularning boshlang`ich adreslarini a massiv elementlariga joylashtirish zarur bo`ladi:

```
For (int i=0; i<m; i++) a[i] =new int [n] ;// n ustunlar soni
```

Shuni qayd etish kerakki , dinamik massivning har bir satri xotiraning turli joylarida joylashishi mumkin.

Ikki o`lchamli massivni o`chirishda oldin massivning har bir elementi (satri), so`ngra massivning o`zi yo`qotiladi.

```
For (i=0; i<m; i++) delete[] a[i];
delete []a;
```

FUNKSIYALARNING MASSIV KIRISH PARAMETRLARI

Funksiyalarga massivlarni kirish argument sifatida berish uchun parametr e'lonida [] qavslar qo'yiladi. Masalan:

```
...
void sortArray(int [], int );  // funksiya e'loni
void sortArray(int n[], int hajm) { // funksiya aniqlanishi
 ...
}
...
```

Dasturda esa, funksiya chaqirilganda, massivning faqat ismi beriladi halos, [] qavslarning keragi yo'q.

```
int size = 10;
int array[size] = {0};

...
void sortArray(array, size); // funksiya chaqirig'i,
                    // faqat massiv ismi - array berildi
...
```

Funksiyaga massivlarni berganimizda, eng katta muammo bu qanday qilib massivdagi elementlari sonini berishdir. Eng yaxshi usul bu massiv kattaligini qo'shimcha kirish parametri orqali funksiyaga bildirishdir. Bundan tashqari, massiv hajmini global konstanta orqali e'lon qilishimiz mumkin. Lekin bu ma'lumotni ochib tashlaydi, global sohani ortiqcha narsalar bilan to'ldirib tashlaydi. Undan tashqari massiv hajmini funksiyaning o'ziga yozib qoyishimiz mumkin. Biroq bunda bizning funksiyamiz faqat bitta kattalikdagi massivlar

bilan ishlaydigan bo'lib qoladi. Yani dasturimiz dimamizmni yo'qotadi. Klaslar yordamida tuzilgan massivlar o'z hajmini biladi. Agar bunday ob'ektlarni qo'llasak, boshqa qo'shimcha parametrlarni

qo'llashimizning keragi yo'q.

Funksiyalarga massivlar ko'rsatkich ko'rinishida beriladi. Buni C++, biz ko'rsatmagan bo'lsak ham, avtomatik ravishda bajaradi. Agar massivlar qiymat bo'yicha chaqirilganda edi, har bir massiv elementining nushasi olinishi kerak bo'lardi, bu esa dastur ishlash tezligiga salbiy ta'sir ko'rsatar edi.

Lekin massivning alohida elementi argument o'rnida funksiyaga berilganda, ushbu element, aksi ko'rsatilmagan bo'lsa, qiymat bo'yicha beriladi. Masalan:

```
...
double m[3] = {3.0, 6.88, 4.7};
void foo(double d){
  ...
}
...
int main()
{
...
void foo(m[2]); // m massivining uchinchi elementining qiymati - 4.7 berildi
...
return (0);
}
```

Agar kiritilayatgan massiv funksiya ichida o'zgarishi ta'qiqlansa, biz funksiya massiv parametri oldiga const sifatini qo'ysak bo'ladi:

```
foo(const char []);
```

Bunda funksiyaga kiradigan massiv funksiya tomonidan o'zgartirilmaydi. Agar o'zgartirishga urinishlar bo'lsa, kompilyator hato beradi.

Massivlar va funksiyalarning birga ko'llanilishiga misol beraylik.

```
// Massiv argumentli funksiyalar

# include <istream.h>

const int arraySize = 10;

double ortalama(int m[], int size) {
  double temp = 0;

  for (int i = 0; i < size; i++) {
    temp += m[i];
  }
  return ( temp / size );
}

void  printArray(const int n[], int size, int ortalama) {
  for (int i = 0; i < size; i++) {
    cout << n[i]; << endl;
  }
  cout << "Ortalama: " << ortalama << endl;
}
```

```
int main()
{
  int m[10] = {89,55,99,356,89,335,78743,44,767,346};

  printArray(m, arraySize, ortalama(m, arraySize)) ;
  return (0);
}
```

Ekranda:

89
55
99
356
89
335
78743
44
767
346
Ortalama: 8092.3

XULOSA .

Xulosa qilib shuni aytish mumkinki , C++ dasturlash tilida ishlash boshqa dasturlash tillariga nisbatan ancha qulay va imkonoyati ham kengroq. MenC++ dasturistrukturasixaqida, belgilarbayoni , Algoritmvadasturtushunchasi, ma’lumotlarnikiritishvachikarishoperatorlarixamdadasturdamassivlarvasatrlarbilanishlashxaqidao`zimga keraklicha bilimvako`nikmaga egabo`ldim.

C++ dasturi

Windowsmuhitidaishlaydigandasturtuzishuchunqulaybo`lganvositabo`lib,

kompyuterdadasturyaratishishlariniavtomatlashtiradi,

xatoliklarnikamaytiradivadasturtuzuvchimehnatini yengillashtiradi.

C++ dasturlash tilida massivlarning ishlatilishi boshqa dasturlash tillariga qaraganda bir muncha afzalliklarga ega. Massivlar bilan ishlash bazi hisoblash masalalarida ancha qulayliklar tug`diradi. Ularning xotirada egallaydigan joyini hisobga olsak dasturning ishlash tezligi xam bir necha marta ortadi.

Men bu kurs ishimda shuni yoritib berganmanki , massivlar xaqida umumiy ma`lumot (bir o`lchamli va ko`p o`lchamli), ularning xotiradan egallaydigan tartibi, saralash, tartiblash, funksiyalar bilan bog`lash , dasturlarda foydalanish kabiyechimlariga oid turli masalalarni misollar yordamida yoritib chiqdim. Bundan shuni xulosa qilishim mumkinki , bundan keyingi ishlarimni yanada boyitgan xolda massivlar ustida dastur tuzib ularni kengroq yoritib berishdir.

FOYDALANILGAN ADABIYOTLAR

1. A.A. Xoidjigitov , Sh.f.Madraximov, U.E.Adamboyev “Informatika va programmalash ” .O`quv qo`llanma, O`z.MU . 2005-yil.

2. B. Straustrop. “Yazik programmirovaniya C++. ” Binom press, 2006-yil.

3. I. Qobulov “C++ tili “
Toshkent nash. 2008-yil.

4. WWW.C++dastur.uz

5.Madraximov. F “C++ dasturlashtili” uslubiyqo`llanma.
2009-yil.

6. Sayfiyev J.F “C++ tiliga kirish”-uslubiy qo`llanma
Buxoro-2005.

7.http.//dastur.uz