МАССИВЫ, УКАЗАТЕЛИ И АДРЕСНАЯ АРИФМЕТИКА

Одномерный массив

Массив – это группа расположенных друг за другом в памяти переменных одного типа и имеющих одно общее имя.
Элементы массива - это последовательно расположенные ячейки памяти. Каждому элементу массива отводится одна ячейка памяти. Все элементы имеют одно имя – *имя массива* и отличаются индексами – *порядковыми номерами* в массиве.
Количество элементов в массиве называется его размером.

*Определение массива***:**

Type ID_array **[Size];**

Type – это любой тип определенный в программе, кроме *функции*, типа *void и ссылки*.

ID_array – это скрытый константный указатель на первый элемент массива.

Size (размер) для любого типа массива – это целая положительная константа, которая может быть задана с помощью константы или константного выражения.

Примеры определения массива
const int N = 6; //определение размера массива
double arr[N]; //определение массива вещественных чисел

arr (имя массива)**–** это адрес первого элемента массива, т.е.**arr** – это **&arr [0]**

Чтобы обратиться к элементу массива, надо указать имя массива и номер элемента в массиве (индекс):
arr[ind];// это число вещественного типа
где **индекс** это *целое положительное число*, которое может меняться в интервале **[0,N-1]**, где **N –** *размер*.

Инициализация массива
1.
const int N = 6; //определение размера массива
int arr[N]={23, -34, 90, 34,-45, 10};

2.
const int N = 6;
int arr[N]={23, -34, 90, 34,-45, 10, 23};
Если количество значений, перечисленных при инициализации больше, чем может поместиться в самом массиве, то при компиляции, программа обнаружит ошибку.

3.
const int N = 6;
int arr[N]={23, -34, 90};
Если количество значений, перечисленных при инициализации меньше, то остальное содержание массива автоматически обнуляется.

4. *Обнуление значений массива* при его определении:
int arr[N]={0};

5.
int arr[]={23, -34, 90, 5, 345};
Размер массива может вычисляться компилятором по количеству значений, перечисленных при инициализации. Следовательно, размер данного массива будет равен 5.

Ввод и вывод элементов массива

```
const int N = 10;//определение размера массива
double x[N];//определение массива вещественных чисел
int ind;//определение индекса массива

//ввод значений
 for(ind = 0; ind <= N-1; ind++){
//выводим номер ячейки, куда вводится значение
   printf("[%d]=",ind);
```

```
// ввод значения в ind-ый номер ячейки
   scanf("%lf", &x[ind]);
 }
```

Просмотр элементов массива
const int N = 10;//определение размера массива
double x[N];//определение массива вещественных чисел
int ind;//определение индекса массива

1. *Слева направо с шагом отличным от 1*

for(ind = 0; ind < N; ind += step) {обработка x[ind] **;**}

2. Справа налево с шагом 1
for(ind = N-1; ind >= 0; ind--) {обработка x[ind] **;**}

3. Обработка элементов массива с обеих сторон, пока не дойдем до его середины
int j;
for(ind = 0, j = N-1; j > ind; ind++, j--) {обработка x[ind] и x[j] ;}

Одномерные массивы и указатели
const int N = 6;
double arr[N]={23.1, -34, 90.7, 34,-45, 10.8};
double *p;

Направить указатель на массив можно двумя способами:

1. p = arr;
2. p = &arr[0];

Если p = p + 3; // p + 3*sizeof(double)

Если на ячейку памяти указывает указатель, то:
p == &arr[3] == &p[3];
*p == arr[3] == p[3];

Ввод и вывод элементов массива через указатель

```
const int N = 6;
double arr[N];
int ind;

//ввод значений через указатель
for (ind = 0; ind < N; ind++) {
       printf("[%d]=",ind);
      scanf("%lf", (arr + ind));
}

//вывод значений через указатель
 for(ind = 0; ind < N; ind++)
       printf("[%d]=%.2lf\n", ind, *(arr + ind));
```

Передача одномерного массива в функцию имеет вид:

ТипВозврЗн **Имя**Функции **(ТипДанных *, int);**

При вызове функции параметры необходимо передавать следующие образом:

1. первый параметр – это имя массива;
2. второй параметр – это размер массива.

```
#include <stdio.h>

void vvod_mas(double *,int);
void vivod_mas(double *, int);

void main(){
      const int n=5,m=6;
      double  x[n], y[m];
```

```
      vvod_mas(x,n);
      vvod_mas(y,m);

      printf("Содержание первого массива:\n");
      vivod_mas(x,n);
      printf("Содержание второго массива:\n");
      vivod_mas(y,m);
}

void vvod_mas(double *p, int k){
      for(int i=0;i<k;i++){
            printf("[%d]=",i);
            scanf("%lf", p++);
      }
}

void vivod_mas(double *p, int k){
      for(int i=0;i< k; i++)
      printf("[%d]=%.2lf\n",i,*(p++));
}
```

АЛГОРИТМЫ СОРТИРОВКИ И ПОИСКА МАССИВА

1. **Сортировка массива по возрастанию, *пузырьковым методом*.**

```
for( j = 1;j < n;j++){
      for(i = 0;i < n-j;i++){
            if( x[i] > x[i+1]){
                buf = x[i];
                x[i] = x[i+1];
                x[i+1] = buf;
            }
      }
}
```

2. **Сортировка массива по возрастанию, *методом простого выбора***

```
for( i = 0;i < n-1; i++){
   k = i;// место min – го элемента

   for( j = i+1;j < n;j++)
      if( x[k] > x[j])
          k = j;

   buf = x[i];
   x[i] = x[k];
   x[k] = buf;
}
```

3. **Сортировка массива по возрастанию, *методом простого включения***

```
for(i = 1;i < n; i++){
//запомнили элемент, для которого ищем место
     y = a[i];
// место куда поместить элемент
    j = i-1;

//поиск подходящего места
   while(y < a[j] && j >= 0){
       a[j+1] = a[j];//сдвиг вправо
       j--;
   }

   a[j+1] = y;//помещение элемента
}
```

4. **Сортировка массива по возрастанию, *методом Шелла***

```
// k – это шаг просмотра
```

```
for( k = n/2; k > 0; k /= 2) {
     do {
      k1 = 0;
      for( i = 0, j = k; j < n; i++, j++)
          if(x[i]>x[j]) {
             buf=x[i];
             x[i]=x[j];
             x[j]=buf;
             k1++;
          }
     // пока был хотя бы один обмен элементов
     }while(k1);
}
```

5. **Сортировка массива по возрастанию,** ***методом Хоара***

```
//левая и правая границы
void sort_mas(int *a,int L,int R) {
      int i, j;
      int SR = a[(L+R)/2], buf;
      i = L;
      j = R;

      do{
      //ищем слева элемент > среднего
            while(a[i]<SR) i++;
      //ищем справа элемент < среднего
            while(a[j]>SR) j--;

            if(i <= j){
                 buf=a[i];
                 a[i]=a[j];
                a[j]=buf;
                  i++;
                   j--;
            }
      }while(i <= j);

      if( i < R)
        sort_mas(a, i, R);

      if( j > L)
        sort_mas(a, L, j);
}
```

6. **Алгоритм** ***бинарного поиска***

```
const int N1 = 6, N2 = 10;
int i,k,k1,j;
int arr1[N1] = {45, -90, 78, 2, 67, -7};
int arr2[N2] = {-100, -90, -56, 2, 12, 45, 67,
                 78, 100, 598};

for(i = 0; i < N1; ++i){
      if( arr1[i] == arr2[N2/2]){
            printf("%d ",arr1[i]);
            continue;
      }

      if( arr1[i] > arr2[N2/2]){
            k = N2/2+1;
            k1 = N2;
      }
      else{
            k = 0;
            k1 = N2/2;
      }
```

```
        for(j = k; j < k1; ++j)
              if(arr1[i] == arr2[j])  {
                    printf("%d ", arr1[i]);
                    break;
        }
}
```

7. ***Сортировка слиянием***

```
const int n1 = 5, n2 = 7, m = n1 + n2;
int arr1[n1] = {-90, -45, 67, 89, 100};
int arr2[n2] = {-34, -12,- 6, 3, 7, 28, 200}, ptr[m];
int i1 = 0, i2 = 0, i3 = 0;

while ((i1 < n1) && (i2 < n2))
        if ( arr1[i1] < arr2[i2] )
              ptr[i3++] = arr1[i1++];
        else
              ptr[i3++] = arr2[i2++];

        while((i1<n1) || (i2<n2))
              if (i1 < n1)
                 ptr[i3++] = arr1[i1++];
              else
                 ptr[i3++] = arr2[i2++];

        for (i1 = 0; i1 < m; i1++)
           printf("%d ", ptr[i1]);
```

МНОГОМЕРНЫЕ МАССИВЫ

Многомерный массив – это массив, элементами которого служат массивы.

Type ID_array [Size1] [Size2];

Type – это любой тип определенный в программе, кроме *функции*, типа *void и ссылки*.

ID_array – это адрес адреса, т.е. указатель на указатель.

Size1 – положительная константа, отражающая количество строк, т.е. количество указателей, которые содержат имена одномерных массивов. **Size2 –**положительная константа, отражающая количество столбцов, т.е. размер одномерного массива.

*Выделение памяти под массив, элементами которого являются массивы***,** представлен на рисунке 1.3

const int N = 4, M = 3 ;
double arr[N][M];

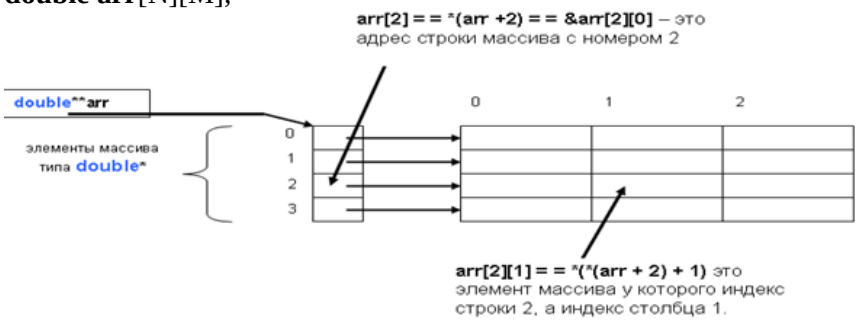

Рис. 1.3

Доступ к элементам многомерных массивов возможен и с помощью индексированных переменных и с помощью указателей:

a[2][1] – доступ с помощью индексированных переменных,
((a+2)+1) – доступ к этому же элементу с помощью указателей.

Примеры определения и инициализации двумерного массива
1. проинициализированы все элементы массива
int a[3][4] = {{11,22,33,44}, {55,66,77,88}, {99,110,120,130}};

2. проинициализированы первые элементы каждой строки
int b[3][4] = {{1},{2},{3}};

3. проинициализированы все элементы массива
int c[3][2]={1,2,3,4,5,6};

Ввод и вывод элементов двумерного массива

```
const int N = 10, M = 6;
//определение массива вещественных чисел
double x[N][M];
int ind1, ind2 ;//определение индексов массива

//ввод матрицы по строкам значений
 for(ind1 = 0; ind1 <= N-1; ind1++)
     for(ind2 = 0; ind2 <= M-1; ind2++) {
      printf("[%d][%d]=",ind1, ind2);
      scanf("%lf", &x[ind1][ind2]);
 }
//вывод в виде матрицы значений
 for(ind1 = 0; ind1 <= N-1; ind1++) {
       for(ind2 = 0; ind2 <= M-1; ind2++)
            printf("%-5.2lf", x[ind1][ ind2]);
            puts("");
}
```

*Передача многомерного массива в функцию имеет вид***:**

ReturnType *FunctionID*(ArrayType[][SIZE2], int);

При вызове функции параметры необходимо передавать следующие образом:
1. **первый параметр** – это имя массива и обязательно размер одномерного массива;
2. **второй параметр** – это количество строк массива, т.е. количество указателей.

Передача динамического многомерного массива в функцию имеет вид:

ReturnType *FunctionID* (ArrayType, int, int);**

При вызове функции параметры необходимо передавать следующие образом:

1. первый параметр – это имя массива;
2. второй параметр – это количество строк массива;
3. третий параметр – это количество столбцов в массива.

```
#include <stdio.h>

void transp(int [][4], int);
void print(int [][4], int);

void main(){
      const int N = 4;
      int mas[N][N] = {{1,2,3,4},    {5,6,7,8},
                       {9,10,11,12}, {13,14,15,16}};

      transp(mas,N);
      print(mas,N);
}
void transp(int a[][4], int n){
     for(int i=0;i<n;i++)
           for(int j=0;j<n;j++)
                 if(i<j){
                      int buf = a[i][j];
                      a[i][j] = a[j][i];
                     a[j][i] = buf;
```

```
                    }
}

void print(int a[][4], int n){
      for(int i=0;i<n;i++){
            for(int j=0;j<n;j++)
                  printf("%-5d",a[i][j]);
                  puts("");
      }
}
```