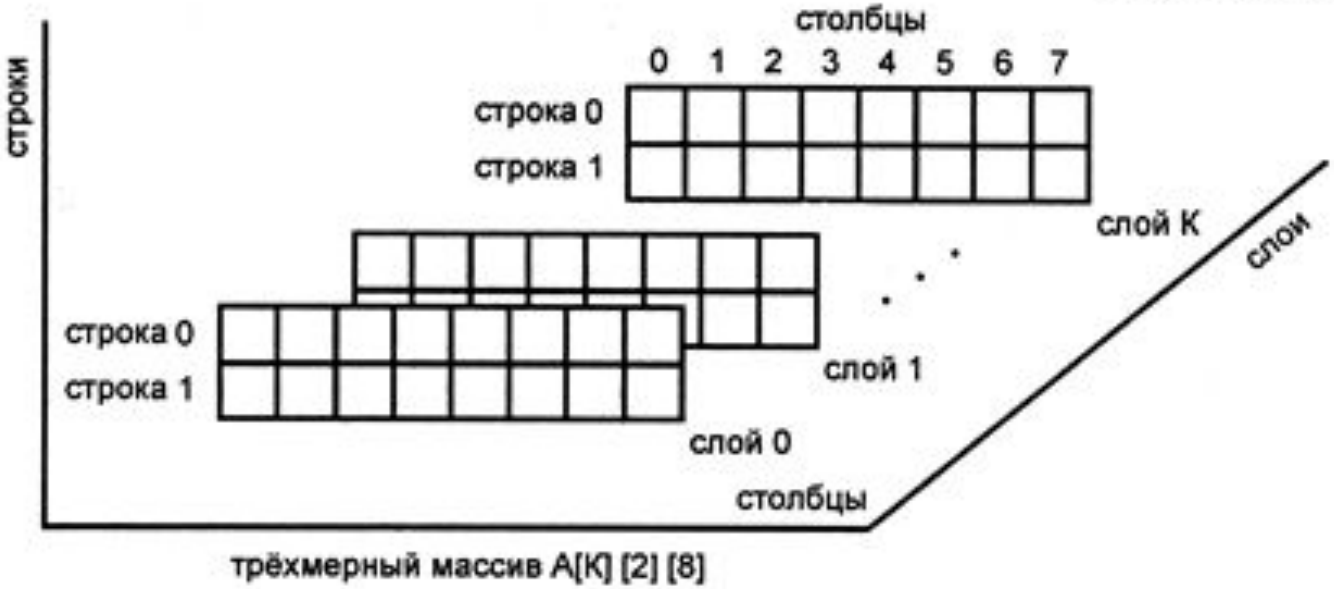
ЛЕКЦИЯ № 1.
Массивы в C++

Массив – именованная последовательность областей памяти, хранящих однотипные элементы



Синтаксис определения массива без дополнительных *спецификаторов* и

<Тип> <ИмяМассива> [<Выражение Типа Константы>];

или

<Тип> <ИмяМассива>[];

Тип – тип элементов объявляемого массива. Элементами массива не могут быть функции, файлы и элементы типа void.

- при объявлении массив инициализируется;
- массив объявлен как *формальный параметр* функции;
- массив объявлен как ссылка на массив, явно определенный в другом файле.

Например:

```
int a[100]; //массив из 100 элементов целого типа
double d[14]; // массив из 14 элементов типа double
char s[]="Программирование"; //символьный массив
const int t=5, k=8;float wer[2*t+k]; //массив из 2*t+k элементов вещественного типаint sample[853]; /*массив из элементов sample[0], sample[1], sample[2],...,sample[852] типа int*/ равносильно объявлению
const int N_max=853;int sample[N_max]; //равносильно объявлению
#define N_max 853...int sample[N_max];
```

Инициализация одномерных массивов

- Например:

```
float t[5]={1.0, 4.3, 8.1, 3.0, 6.74};
char b[7]={'П','р','и','в','е','т'};/*в данных примерах длину массива компилятор вычисляет по количеству начальных значений, перечисленных в фигурных скобках*/
int d[10]={1, 2, 3};
char a[10]="Привет";/*в данных примерах определяется значение только заданных переменных d[0],d[1],d[2] и a[0],a[1],...,d[9], остальные элементы не инициализируются*/
```

Пусть необходимо проинициализировать массивы для *создания таблицы* сообщений об ошибках:

```
char e1[12] = "read error\n";
char e2[13] = "write error\n";
char e3[18] = "cannot open file\n";
```

- *Компилятор* C++ сам сформирует нужное *значение* по количеству инициализирующих данных. В нашем случае под *массив* e2 будет отведено 13 байтов, включая последний *байт* с нулевым кодом, завершающий каждую строку. Оператор
- printf("%s имеет длину, равную %d\n",e2,sizeof (e2)); выведет на экран write error имеет длину, равную 13

Обращение к элементам одномерного массива

- *Адресация* элементов массива осуществляется с помощью индексированного имени. *Синтаксис* обращения к элементу массива:

ИмяМассива[*ВыражениеТипаКонстанты*];

или

ИмяМассива[*ЗначениеИндекса*];

- Таким образом, чтобы обратиться к элементу массива, надо указать имя массива и номер элемента в массиве (*индекс*).
- Например:
- a[0] – *индекс* задается как константа,
- d[55] – *индекс* задается как константа,
- s[i] – *индекс* задается как *переменная*,
- w[4*p] – *индекс* задается как *выражение*.
- Следует помнить, что *компилятор* в процессе генерации кода задет начальный *адрес* массива, который в дальнейшем не может быть *переопределен*. Начальный *адрес* массива – это *адрес* первого элемента массива. Вообще в программе начальным адресом массива считается ИмяМассива либо &ИмяМассива[0]. Имя массива считается константой-указателем, ссылающимся на *адрес* начала массива.

Определение размера памяти для одномерных массивов

- *Массив* занимает *непрерывную* область памяти. Для одномерного массива полный объем занимаемой памяти в байтах вычисляется по формуле:

  Байты = sizeof (тип) * размер массива

- Например, пусть *одномерный массив* A состоит из элементов, расположенных в памяти подряд по возрастанию индексов, и каждый элемент занимает по k *байт*. Тогда *адрес* i -того элемента вычисляется по формуле:

  адрес(A[i]) = адрес(A[0]) + i*k

Пример 1. Определение размера памяти одномерного массива.

```
1.  #include "stdafx.h“
2.  #include <iostream>
3.  using namespace std;
4.  #define v 4
5.  #define p 3
6.  int _tmain(int argc, _TCHAR* argv[]){
7.  const int q=4, r=1;
8.  int i_mas[10];
9.  int k=sizeof(i_mas);
10. cout << "i_mas[10] занимает " << k << " байт\n";
11. float f_mas[7]={2.0,4.5,8.3,7.0,1.0};
12. int t=sizeof(f_mas);
13. cout << "f_mas[7]={2.0,4.5,8.3,7.0,1.0} занимает "<< t<<"байт\n";
```

```
14. double d_mas[2*q-r];
15. int w=sizeof(d_mas);
16. cout << "d_mas[2*q-r] занимает " << w << " байт\n";
17. double d1_mas[2*v/p];
18. int w1=sizeof(d1_mas);
19. cout << "d1_mas[2*v/p] занимает " << w1 << " байт\n";
20. char c_mas[]="Программирование";
21. int s=sizeof(c_mas);
22. cout << "c_mas[]=\"Программирование\"занимает"<< s <<"
    байт\n";
23. system("pause");
24. return 0;
25. }
```

Результат выполнения программы:

- i_mas[10] занимает 40 *байт* – 4 байта (тип int) * 10 (количество элементов массива)
- f_mas[7] = {2.0,4.5,8.3,7.0,1.0} занимает 28 *байт* – 4 байта (тип float) * 7 (объявленное количество элементов массива)
- d_mas[2*q-r] занимает 56 *байт* – 8 *байт* (тип double) * 7 (вычисленное через формулу количество элементов массива)
- d1_mas[2*v/p] занимает 16 *байт* – 8 *байт* (тип double) * 2 (вычисленное через формулу количество элементов массива)
- c_mas[]="Программирование" занимает 17 *байт* – 1 *байт* (тип char) * 17 (16 знаков + нулевой *байт* '\0')

Указатели и одномерные массивы

- Поскольку имя массива является указателем, допустимо, например, такое *присваивание*:

  int array[25];int *ptr;ptr=array;

- Здесь *указатель ptr* устанавливается на *адрес* первого элемента массива, причем *присваивание ptr* = array можно записать в эквивалентной форме *ptr*=&array[0].

Адрес каждого элемента массива можно получить, используя одно из трех выражений:

Адрес в памяти	**1000**	**1002**	**1004**	**1008**	**1008**	**...**	
Индекс	0	1	2	3	4	...	24
Значения	1	2	3	4	5	...	25
Адрес i - го элемента	array&array[0]ptr	array+1&array[1]ptr + 1	array+2&array[2]ptr + 2	array+3&array[3]ptr + 3	array+4&array[4]ptr + 4	...	array+24&array[24]ptr + 24

- А обращение к пятому элементу массива можно записать как: array[4], *(array + 4), *(ptr + 4). Эти *операторы* вернут пятый элемент массива.

Указатели в одномерном массиве

Пример 2:

```
1.  #include "stdafx.h "
2.  #include <iostream>
3.  using namespace std;
4.  int _tmain(int argc, _TCHAR* argv[]){
5.  int array[10];
6.  int *p;
7.  p=&array[0];        /*эквивалентные операции присваивания*/
    *array=2; printf("%d\n", array[0]);
8.   array[0]=2; printf("%d\n", array[0]);
9.  *(array+0)=2; printf("%d\n", array[0]);
10. *p=2; printf("%d\n", array[0]);
11.  p[0]=2; printf("%d\n", array[0]);
12. *(p+0)=2; printf("%d\n", array[0]);
13. system("pause");
14. return 0;}
```

Использование элементов массивов в выражениях (операции с элементами массивов)

- С элементами объявленного массива можно выполнять все действия, допустимые для обычных переменных этого типа (выше был приведен пример целочисленного массива, т.е. типа int). Например, возможны *операторы присваивания*:

  hours[4] = 34;hours[5] = hours[4]/2; или *логические выражения* с участием элементов массива:

  if (number < 4 && hours[number] >= 40) { ... }

- Присвоить значения набору элементов массива часто бывает удобно с помощью циклов for или while. Для массивов не допустима операция прямого присваивания.

Генерация одномерных массивов

Генерацию массива (массивов) в программе оформляют в виде отдельной функции. Стандартными способами генерация массивов являются:

- ввод данных с клавиатуры,
- формирование значений через *генератор* случайных чисел,
- *вычисление значений* по формуле,
- ввод данных из файла.

При этом при формировании значений элементов используют цикл по индексам элементов или *арифметические операции* с указателем на *массив*. В данной работе рассмотрим первые три способа генерации массивов (*Примеры 3, 4, 6*).

Вывод одномерных массивов

- Одномерные массивы удобно выводить в строку или в столбец в зависимости от задачи (*Пример 3* и *4*).
- *Пример 3*.

```
1.  /*Генерация целочисленного массива числами с клавиатуры и вывод массива в строку*/
2.  #include "stdafx.h "
3.  #include <iostream>
4.  using namespace std;
5.  #define max 20
6.  void gen (int k,int *pp);//прототип функции генерации массива
7.  void out (int k,int x[max]);//прототип функции вывода массива
8.  int _tmain(int argc, _TCHAR* argv[]){
9.  int a[max],n,*p;
10. do {
```

```
1.  printf("\nВведите количество элементов массива n (n<=20):");
2.  scanf ("%d",&n);     }
3.  while (n>max); //проверка выхода за границы массива
4.  p=a;
5.  gen(n,p);
6.  out(n,a);
7.  system("pause");
8.  return 0;}
9.  //Описание функции генерации массива с клавиатуры
10. void gen(int k,int *pp){
11.  /*передача указателя как параметра позволяет вернуть
    сформированный массив в основную программу*/
12. int i;  printf("\nВведите значения %d элементов массива: \n",k);
    for (i=0;i<k;i++){
13.  printf("x[%d]= ",i);
14. scanf("%d",pp++);
```

```
15.  }}
16.   //Описание функции вывода массива в строку
17.  void out (int k,int x[max]){
18.   int i;
19.  printf("\nВывод значений %d элементов массива в строку: \n",k);
20.  for (i=0;i<k;i++)
21.  printf("%d\t",x[i]);
22.  }
```

Пример 4.
Описание функции генерации массива значениями элементов арифметической прогрессии

```
1. void gen(int k,int x[max]) {
2. int i,d;
3. printf ("\nВведите нулевой элемент прогрессии: ");
4. scanf("%d",&x[0]);
5. printf ("\nВведите разность прогрессии: ");
6. scanf("%d",&d);
7. for (i=1;i<k;i++)     x[i]=x[i-1]+d;}
```

Пример 5.
Описание функции вывода массива в столбец

```
1. void out (int k,int x[max]){
2. int i;
3. printf("\nВывод значений %d элементов массива в столбец:
   \n",k);
4. for (i=0;i<k;i++)
5. printf("x[%i]= %d\n",i,x[i]);
6. }
```

Для использования функции *генерации случайных чисел* необходимо подключить библиотеку <time.h>.

Для написания кода генерации массива случайными целыми числами используется:

- Функция srand(). Синтаксис:
- void srand(unsigned seed); – функция устанавливает свой аргумент как основу (*seed*) для новой последовательности псевдослучайных целых чисел, возвращаемых функцией *rand*(). Сформированную последовательность можно воспроизвести. Для этого необходимо вызвать srand() с соответствующей величиной *seed*.
- Для использования данной функции необходимо подключить библиотечный файл <stdlib.h>.

- Функция *rand*(). Синтаксис:
- int rand(void); – функция возвращает *псевдослучайное число* в диапазоне от нуля до RAND_MAX. Для использования данной функции необходимо подключить библиотечный файл <stdlib.h>.
- Константа RAND_MAX определяет максимальное значение случайного числа, которое может быть возвращено функцией *rand*(). Значение RAND_MAX – это максимальное положительное целое число.
- //генерация случайных целых чисел на [a,b) x[i]=rand()%(b-a)+a;//генерация случайных вещественных чисел на [a,b) y[i]= rand()*1.0/(RAND_MAX)*(b-a)+a;

Пример 6.
Описание функции генерации массива случайными вещественными числами на[a,b)

```
1. void gen(int k,int a, int b, float x[max]){
2.  int i;  srand(time(NULL)*1000);   //устанавливает начальную точку генерации случайных чисел
3. for (i=0;i<k;i++){   x[i]=(rand()*1.0/(RAND_MAX)*(b-a)+a); //функция генерации случайных чисел на [a,b)
4. }
5. }
```

Ключевые термины:

- **Генерация массива** – это автоматическое формирование значений его элементов.
- **Значение элемента массива** – это *значение*, хранящееся по адресу, который соответствует данному элементу.
- **Измерение массива** – это количество индексов в определении массива.
- **Имя массива** – *идентификатор*, именующий выделенную под *массив* область памяти.
- **Индекс элемента массива** – это порядковый номер элемента в последовательности.
- **Инициализация массива** – это формирование значений его элементов.
- **Массив** – это именованная последовательность областей памяти, хранящих однотипные элементы.
- **Одномерный массив** – это *массив*, измерение которого равно единице.
- **Размер массива** – это количество элементов в массиве.
- *Тип массива* – это тип элементов массива.
- **Указатель на массив** – это *адрес* области памяти, выделенной под *массив*.
- **Элемент массива** – это каждая область памяти из последовательности областей, выделенных под *массив*.

Вопросы для самоконтроля

1. Почему в программе на C++ необходимо, чтобы был известен размер массива?
2. Можно ли выполнить прямое присваивание массивов объявленных так: int x[10], y[10];?
3. Когда, с какой целью и почему возможно объявление безразмерных массивов?
4. В чем отличие обращения к элементам массива с помощью индексированного имени и посредством арифметики с указателями?
5. Может ли значение элемента массива использоваться в качестве индекса другого элемента массива?
6. Эквивалентны ли для массива mas следующие обращения и почему: mas и &mas[0]?
7. Какие ограничения распространяются на тип массива?
8. Каким образом можно определить объем памяти, выделяемой под массив?
9. Каким образом можно составить выражение для генерации массива случайными целыми числами на заданном промежутке?