Тема 4

Массивы

Массивы

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнять однообразные действия, им дают одно имя, а различают по порядковому номеру (индексу).

Конечная именованная последовательность однотипных величин, доступ к каждой из которых осуществляется с помощью индексации, называется массивом.

Массивы делятся на:

- статические;
- динамические.

Статические массивы

Количество элементов в статическом массиве известно при написании программы и никогда не меняется. Память под такой массив выделяет компилятор.

Описание статического массива:

тип имя [количество_элементов] [инициализатор]

Доступ к элементу массива:

имя_массива [индекс]

Индексация массивов начинается с нуля.

Выход индекса за пределы массива никак не контролируется, последствия такого выхода непредсказуемы!

Пример работы со статическим массивом

```
// подсчёт суммы элементов массива
//
#include <iostream>
using namespace std;
int main(){
 setlocale (LC_ALL, ”.1251”);
 const int n = 10;
 int sum = 0;
 int marks[n] = {3, 4, 5, 4, 4};
     // инициализация первых пяти элементов массива
     // оставшиеся элементы инициализируются нулём
 for (int i = 0; i<n; i++)
  sum += marks[i];
 cout << "Сумма элементов: " << sum << endl;
 return 0;
}
```

Организация массивов с использованием указателей

```
int Mas [10];
```

Реализация этого описания выполнена так: Mas представляет собой константный указатель, содержащий адрес первого из элементов массива. Память под элементы выделена подряд:

```
int * const Mas
```

*Mas – обращение к начальному элементу (с индексом 0)

*(Mas+1) – обращение к элементу с индексом 1

*(Mas+i) – обращение к элементу с индексом i

Таким образом, запись Mas[i] – синоним записи *(Mas+i)

Динамические массивы

Количество элементов динамического массива на этапе компиляции не известно и, как правило, зависит от входных данных. Память под динамический массив выделяется во время выполнения программы с помощью операций выделения памяти:

```
int N;
cout << ”Введите количество элементов массива: ”;
cin >> N;
double *p; // писать double p[N] нельзя!
p = new double [N];
cout << ”Введите элементы массива:\n”;
for (int i=0; i<N; i++)
  cin >> p[i];
…
delete [] p;
// квадратные скобки означают освобождение памяти
// для всех элементов массива
```

Динамические массивы (продолжение)

Элементы динамических массивов не инициализированы, в отличие от статических. Однако можно выполнить инициализацию нулями с помощью конструкции

```
p = new double [N] ();
```

Многомерные массивы

Многомерные массивы фиксированного размера задаются указанием каждого измерения в квадратных скобках, например, оператор

```
int matr [6][8];
```

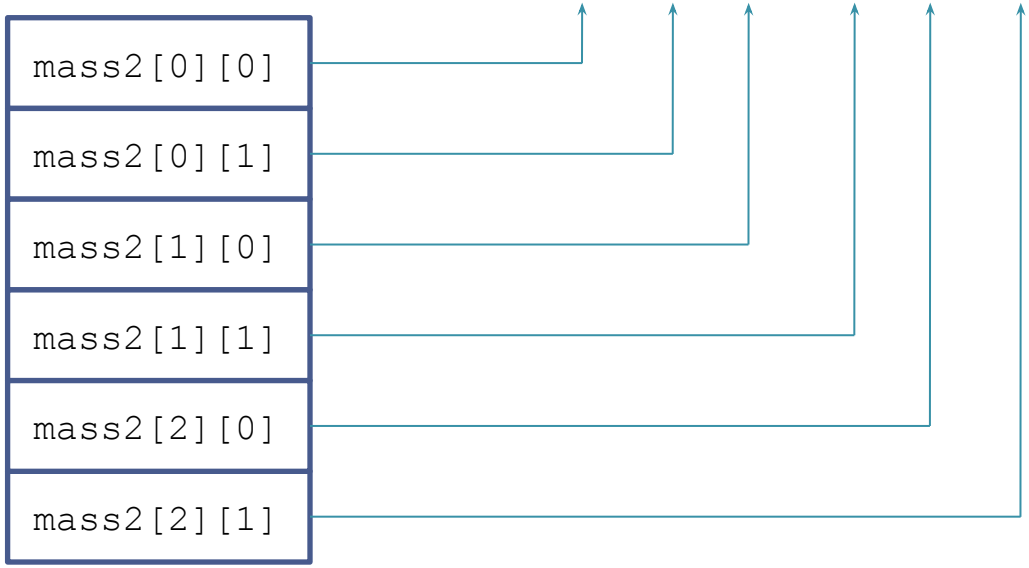
задает описание двумерного массива из 6 строк и 8 столбцов. В памяти такой массив располагается в последовательных ячейках построчно. Многомерные массивы размещаются так, что при переходе к следующему элементу быстрее всего изменяется последний индекс.

Многомерные массивы (продолжение)

Для доступа к элементу многомерного массива указываются все его индексы, например, matr[i][j].

Инициализация многомерного массива может быть выполнена одним из следующих способов:

```
int mass2 [3][2] = { {1, 1}, {0, 2}, {1, 0} };
int mass2 [3][2] = {1, 1, 0, 2, 1, 0};
```



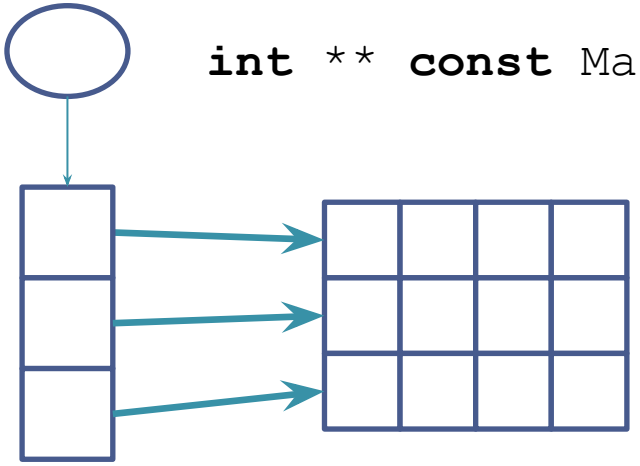
Пример работы с двумерным массивом

```
// подсчёт суммы элементов каждой строки двумерного
// массива
#include <iostream>
using namespace std;
int main(){
  setlocale(LC_ALL, ".1251");
  int sum;
  int matr[3][3] = {{5, 3, 2}, {2, -4, 5}, {10, 2, 8}};
  for (int i=0; i<3; i++) {
   sum = 0;
   for (int j=0; j<3; j++)
       sum+=matr[i][j];
   cout << "Сумма элементов в " << i <<
          "-й строке равна " << sum << "\n";
  }
  return 0;
}
```

Организация двумерных массивов с использованием указателей

```
int Matr [3][4];
```

Реализация этого описания выполнена с использованием указателей на указатели:




```
int * const Matr[3]
```

Двумерные динамические массивы

Выделение памяти для двумерных динамических массивов выполняется в соответствии с принципами организации двумерных массивов.

- Выделение памяти:

```
// пусть n – количество строк, k – столбцов
int **p;
p=new int*[n];
for (int i=0; i<n; i++)
  p[i] = new int[k];
```

- Освобождение памяти выполняется в обратном порядке:

```
for (int i=0; i<n; i++)
  delete [] p[i];
delete [] p;
```

Треугольные динамические матрицы

- Выделение памяти под верхний треугольник квадратной матрицы NxN:

```
int **p;
p=new int*[n];
for (int i=0; i<n; i++)
  p[i] = new int[n-i];
```

- Выделение памяти под нижний треугольник такой же матрицы:

```
int **p;
p=new int*[n];
for (int i=0; i<n; i++)
  p[i] = new int[i+1];
```