Занятие № 7. Использование массивов

Инициализация одномерных массивов

C++ предоставляет удобный механизм инициализации одномерных массивов. Вам нужно только задать список значений, которыми инициализируется массив, заключенный в фигурные скобки ({ }). Список должен быть разделен запятыми и может занимать несколько строк. Если данных в списке меньше, чем размер массива, компилятор допишет в остальные элементы нули. В случае же, если данных больше, чем элементов массива, компилятор выдаст сообщение об ошибке.

```
1:  // C++ программа иллюстрирует использование одномерных массивов
2:  // при расчете среднего значения.
3:  // Данные задаются при инициализации массива.
4:
5:  #include <iostream.h>
6:
7:  const int MAX = 10;
8:
9:  int main()
10:{
11:    double array[MAX] = { 12.2, 45.4, 67.2, 12.2, 34.6, 87.4,
12:                          83.6, 12.3, 14.8, 55.5 };
13:    int num_elem = MAX;
14:
15:    double sum = 0;
16:    for (int ix = 0; ix < num_elem; ++ix)
17:       {
18:       sum += array[ix];
19:       cout << "array[" << ix << "]: " << array[ix]  << endl;
20:       }
21:    cout << endl << "Average: " << sum / num_elem << endl;
22:    return 0;
23: }
```

C++ может **автоматически создавать массив** размера, равного количеству элементов в списке инициализации. В этом случае при объявлении массива вам не нужно указывать в скобках размерность массива, компилятор определит это число сам.

Чтобы использовать описанное выше свойство в рассматриваемой программе, необходимо переписать строки с 11-й по 13-ю следующим образом:

```
double array[] = { 12.2, 45.4, 67.2, 12.2, 34.6, 87.4,
  83.6, 12.3, 14.8, 55.5 };
int num_elem = sizeof(array) / sizeof(array[0]);
```

Массивы — параметры функции

C++ позволяет определять массивы в качестве параметров функции.

C++ позволяет задавать массивы-параметры точно или в общем виде: можно указать размер массива при объявлении параметра или объявить параметр с пустыми скобками.

Массив-параметр фиксированного размера

Общая форма объявления в качестве параметра массива фиксированного размера:

type parameterName[arraySize];

Пример:

```
int minArray(int  arr[100]);
void  sort(unsigned dayNum[7]);
```

Массив-параметр неопределенной длины

Общая форма объявления в качестве параметра массива неопределенного размера (открытого массива) такова:

```
type parameterName[]
```

Пример:

```
int minArray(int  arr[],   int num_elem);
void  sort   (unsigned dayNum[],   int num_elem);
```

Так как функции неизвестна размерность массива при объявлении, она должна быть указана через дополнительный параметр.

Составить программу, которая выполняет следующие действия:

- предлагает ввести количество обрабатываемых в программе чисел — от двух до десяти,
- затем просит ввести указанное количество целых чисел, которыми заполняется массив,
- выводит минимальное число среди элементов массива, выводит максимальное число среди элементов массива.

```
1:  // программа C++, определяющая наибольший
2:  // и наименьший элементы массива
3:
4:  #include <iostream.h>
5:
6:  const int MIN = 2;
7:  const int MAX = 10;
8:
9:  int getNumPoints(int low, int high)
10:{
11:    int numPoints;
12:    do
13:       {
14:       cout << "Enter number of data points ["
15:            << low << " to "
16:            << high << "]: ";
17:       cin >> numPoints;
18:       } while (numPoints < low || numPoints > high);
19:   return numPoints;
20: }
21:
22: int findMin(int array[], int size)
23: {
24:    int small = array[0];
25:    for (int i = 1; i < size; i++)
26:       if (array[i] < small)
27:          small = array[i];
```

```
28:          return small;
29: }
30:
31: int findMax(int array[], int size)
32: {
33:     int big = array[0];
34:     for (int i = 1; i < size; i++)
35:        if (array[i] > big)
36:           big = array[i];
37:     return big;
38: }
39:
40: int main()
41: {
42:     int array(MAX], num_elem;
43:
44:         num_elem = getNumPoints(MIN, MAX);
45:
46:         //  Запрос ввести данные
47:         for (int i = 0; i < num_elem; i++)
48:         {
49:             cout << "array[" << i << "]: ";
50:             cin >> array[i];
51:         }
52:
53:     cout << "Smallest value in array is "
54:          << findMin(array, num_elem) << endl
55:          << "Biggest value in array is "
56:          << findMax(array, num_elem) << endl;
57:     return 0;
58: }
```

Многомерные массивы

В многомерных массивах каждое дополнительное измерение имеет свой параметр доступа, индекс. Двумерные массивы (или матрицы, если хотите), являются наиболее популярными многомерными массивами. Трехмерные менее популярны, и так далее.

Многомерный массив является множеством одномерных массивов.

Двумерные и трехмерные массивы

Общая форма объявления двумерных и трехмерных массивов:

```
тип  array[размер1][размер2];
тип  array[размер1][размер2][размер3];
```

Как и в одномерных массивах, нижнее значение индекса по каждому измерению равно 0, а в скобках указывается количество элементов по каждому измерению.

Примеры

```
double  matrixA[100][10];
char  table [41][22][3];
int   index[7][12];
```

Инициализация многомерных массивов

C++ позволяет инициализировать как одномерные, так и многомерные массивы. Большинство компиляторов хранит элементы многомерного массива непрерывным списком, как один большой одномерный массив.

```
1:  // пример работы в C++ с двумерным массивом.
2:  // рассчитывается среднее значение по каждому столбцу.
3:
4:  #include <iostream.h>
5:
6:  const int MAX_COL = 3;
7:  const int MAX_ROW = 3;
8:
9:  int main()
10: {
11:    double matrix[MAX_ROW][MAX_COL]=
12:       { 1, 2, 3,       // строка №1
13:         4, 5, 6,       // строка №2
14:         7, 8, 9        // строка №3
15:       };
16:    double sum, average;
17:    int rows = MAX_ROW, cols = MAX_COL;
18:
19:    // вывод элементов матрицы
20:    cout << "Matrix is:" << endl;
21:    for (int i = 0; i < rows; i++)
22:       {
23:       for (int j = 0; j < cols; j++)
24:          {
25:          cout.width(4);
26:          cout.precision(1);
27:          cout << matrix[i][j]<< " ";
28:          }
29:       cout << endl;
30:       }
31:     cout << endl;
32:
33:    // вычисляется среднее значение по каждому столбцу
```

```
34:     for (int j = 0; j < cols; j++)
35:        {
36:           sum =0.0;              // инициализация sum
37:           for (int i = 0; i < rows; i++)
38:              sum += matrix[i][j];
39:           average = sum / rows;
40:           cout << "Average for column " << j
41:                << " = " << average << endl;
42:        }
43:      return 0;
44:      }
```

Многомерные массивы — параметры функции

C++ позволяет вводить в качестве параметров функции многомерные массивы. Как и в случае одномерных массивов, вы можете точно указать размер массива либо задать массив неопределенной длины. В последнем случае вы можете оставить неопределенным размер только по одному измерению, а именно по первому. Если вы хотите определить в качестве параметра массив фиксированной длины, вы должны определить размер по каждому измерению.

Массив-параметр фиксированного размера

Общая форма объявления в качестве параметра массива фиксированного размера:

```
тип  имяПараметра[dim1Size][dim2Size] . . .
```

Пример:

```
int minMatrix(int intMat[100] [20], int rows, int
  cols);
void sort(unsigned mat[23][55],int rows, int cols, int
  collndex);
```

Массив-параметр неопределенной длины

Общая форма объявления в качестве параметра массива неопределенной длины (открытого массива) такова:

```
type  parameterName[][dim2Size]...
```

Пример:

```
int  minMatrix(int   intMat[][20],   int   rows,   int
  cols);
void sort(unsigned mat[][55],int rows, int cols, int
  collndex);
```