Вектора и массивы

В программировании массивом называется пронумерованная последовательность однотипных элементов. Номера элементов, называемые индексами, должны быть последовательными целыми числами в заданном диапазоне. В языке C++, как и во многих других, индексация элементов массива всегда начинается с нуля. Таким образом, если массив содержит n элементов, то индексом последнего является число $n-1$. Фундаментальными свойствами массива являются следующие:

- размер массива определяется при создании и не может быть изменен;
- все элементы располагаются в памяти последовательно;
- время доступа к элементу по его индексу является константой, не зависящей от размера массива и самого индекса.

Невозможность изменения размера массива часто является ограничением, существенно затрудняющим работу, поэтому в C++, как правило, используется контейнерный тип `vector`, обладающий функциональностью массива, но, кроме того, позволяющий менять свой размер.

Вектора в C++ относятся к шаблонным типам, которые используют свой синтаксис объявления переменных. При объявлении переменной шаблонного типа требуется указать параметр шаблона в угловых скобках. У векторов имеется один шаблонный параметр, указывающий тип элементов, хранящихся в векторе. Например, корректными являются следующие объявления переменных.

```
#include <vector>
using namespace std;
int main() {
  vector<int> a,b(10),c(10,-1);
  vector<double> x, y(10), z(10,0.5);
  ...
}
```

В первой строке программы указано имя библиотеки «`vector`», которую требуется подключить для использования векторов. В программе создаются три вектора из целых чисел и три вектора из чисел с плавающей точкой. При этом вектора `a` и `x` изначально пустые, вектора `b` и `y` содержат по 10 элементов, инициализированных значениями по умолчанию (нулями), а вектора `c` и `z` состоят из 10 значений -1 и 0.5 соответственно.

При создании вектора могут инициализироваться списками инициализаторов в фигурных скобках.

```
vector<int> d={10,20,30,40,50};
```

Размер вектора в этом случае вычисляется по количеству элементов.

Использование векторов

Если вектор содержит n элементов, то номерами этих элементов являются числа от 0 до $n-1$. Операция доступа к элементу по номеру называется индексацией и записывается в виде `v[i]`. Здесь `v` — это имя вектора, а `i` — порядковый номер (индекс) элемента.

Обычно для работы с массивами используются циклы, в которых индекс массива пробегает все возможные значения. Вместе с тем можно использовать и специальную форму цикла, называемую циклом по коллекции.

```
vector<string> v={"Humpty-Dumpty","Sat","on","a","wall"};
for (string x:v)
    cout << x <<" ";
```

В этом цикле заводится новая переменная `x` строкового типа, и на каждой итерации в нее последовательно заносятся все элементы вектора. Таким образом, в результате работы данной программы будет выведена строка

```
Humpty–Dumpty Sat on a wall
```

Вектора позволяют изменять свой размер. Наиболее распространены методы добавления элемента в конец вектора `push_back(x)` и удаления последнего элемента `pop_back()`. Узнать текущий размер вектора можно при помощи метода `size()`. Эти методы являются членами вектора, поэтому они используются с его именем, например, `v.push_back(x)`, `v.pop_back()` или `v.size()`. Изменить размер вектора можно методом `resize(n)`, где `n` — новый размер вектора или с двумя параметрами `resize(n,c)`, где `c` — значение для инициализации новых элементов вектора. При увеличении размера вектора вновь появившиеся ячейки инициализируются значением `c`, старые же значения остаются на своих местах.

Для доступа к последнему элементу вектора можно использовать метод `back()`.

Далее приведен пример программы, которая удаляет из вектора последний элемент, а потом добавляет новый.

```
vector<string> v={"Humpty–Dumpty","Sat","on","a","ball"};
cout << v.size() << endl;
for (string x:v)
    cout << x <<" ";
cout << endl << v.back() << endl;

v.pop_back();
cout << v.size() << endl;
for (string x:v)
    cout << x <<" ";
cout << endl << v.back() << endl;

v.push_back("wall");
cout << v.size() << endl;
for (string x:v)
    cout << x <<" ";
cout << endl << v.back() << endl;
```

Двумерные вектора

Для того, чтобы создать структуру данных, похожую на прямоугольную таблицу также можно использовать вектора. Вектор может содержать в себе элементы любых типов, в том числе, другие вектора.

```
vector<vector<double>> m;
```

В данном примере создана пустая двумерная таблица, размеры которой далее можно изменить при помощи метода `resize`.

```
v.resize(10,vector<double>(20));
```

Вторым параметром здесь указан константный вектор из 20 чисел с плавающей точкой. Таким образом, здесь создается двумерная таблица из 10 строк, в каждой строке по 20 чисел. Отметим, что размеры строк могут различаться.

Доступ к элементам двумерного вектора также использует оператор индексации. В следующем примере изменяется вся последняя строка таблицы.

```
v[9]={1.5,2.5,3.5};
```

Теперь последняя строка будет содержать только 3 элемента. При этом все остальные строки не изменятся и будут содержать по 20 чисел.

Для доступа к конкретной ячейке двумерной таблицы оператор индексации надо применить два раза

```
v[5][7]=0.5;
```

При этом первое применение индексации выбирает пятую строку, а второе — седьмой элемент в этой строке.