

2.3. Контейнеры

2.3.1. Контейнер vector

Тип `vector` представляет собой набор элементов одного типа. Тип элементов вектора указывается в угловых скобках. Классический сценарий использования вектора — сохранение последовательности элементов.

Создание вектора требуемой длины. Ввод и вывод с консоли

Напишем программу, которая считывает из консоли последовательность строк, например, имен лекторов. Сначала на вход подается число элементов последовательности:

```
int n;  
cin >> n;
```

Поскольку известно количество элементов последовательности, его можно указать в конструкторе вектора (то есть в круглых скобках после названия переменной):

```
vector<string> v(n);
```

После этого можно с помощью цикла `for` перебрать все элементы вектора по ссылке:

```
for (string& s : v) {  
    cin >> s;  
}
```

Каждый очередной элемент `s` — ссылка на очередной элемент вектора. С помощью этой ссылки считывается очередная строка.

Теперь остается вывести вектор на экран, чтобы проверить, что все было считано правильно. Для этого удобно написать специальную функцию, которая выводит все значения вектора. Вызываем функцию следующим образом:

```
PrintVector(v);
```

А само определение функции `PrintVector` располагаем над функцией `main`:

```
void PrintVector(const vector<string>& v) {  
    for (string s : v) {  
        cout << s << endl;  
    }  
}
```

Запустим программу и проверим, что она работает:

```
> 2
> Anton
> Ilia
Anton
Ilia
```

Отлично: мы успешно считали элементы вектора и успешно их вывели.

Добавление элементов в вектор. Методы `push_back` и `size`

Можно реализовать эту программу несколько иначе с помощью цикла `while`. Также считаем число элементов вектора `n`, но создадим пустой вектор `v`.

```
int n;
cin >> n;
vector<string> v;
```

Создадим переменную `i`, в которой будет храниться индекс считываемой на данной итерации строки.

```
int i = 0;
```

В цикле `while` считываем строку из консоли в локальную вспомогательную переменную `s`, которая добавляется к вектору с помощью метода `push_back`:

```
while (i < n) {
    string s;
    cin >> s;
    v.push_back(s);
    cout << "Current size = " << v.size() << endl;
    ++i;
}
```

В конце каждой итерации значение `i` увеличивается на 1. Чтобы продемонстрировать, что размер вектора меняется, на каждой итерации его текущий размер выводится на экран.

После завершения цикла чтения, как и в предыдущем примере, выведем значения вектора на экран с помощью функции `PrintVector`:

```
PrintVector(v);
```