



Контейнеры



Контейнеры – это объекты

- Контейнеры-это объекты, содержащие другие объекты
- Существует несколько различных типов контейнеров. Например, классы `vector`, `queue` и `list` – *последовательные контейнеры*
- Также есть *ассоциативные контейнер*, которые позволяют находить нужные значения на основе заданных ключевых значений



Векторы

- Векторы представляют собой динамические массивы, которые при необходимости могут увеличивать свой размер.
 - Для доступа к его элементам можно использовать стандартное обозначение индексации массивов, несмотря на то что вектор – это динамический массив
 - Объявление вектора
 - `vector<int> iv; //Вектор нулевой длины`
 - `vector<int> cv(5); //5-элементный вектор`
 - `vector<int> iv2(iv); /*Создание вектора на основе вектора iv */`
-



Функции-члены, определенные в классе `vector`

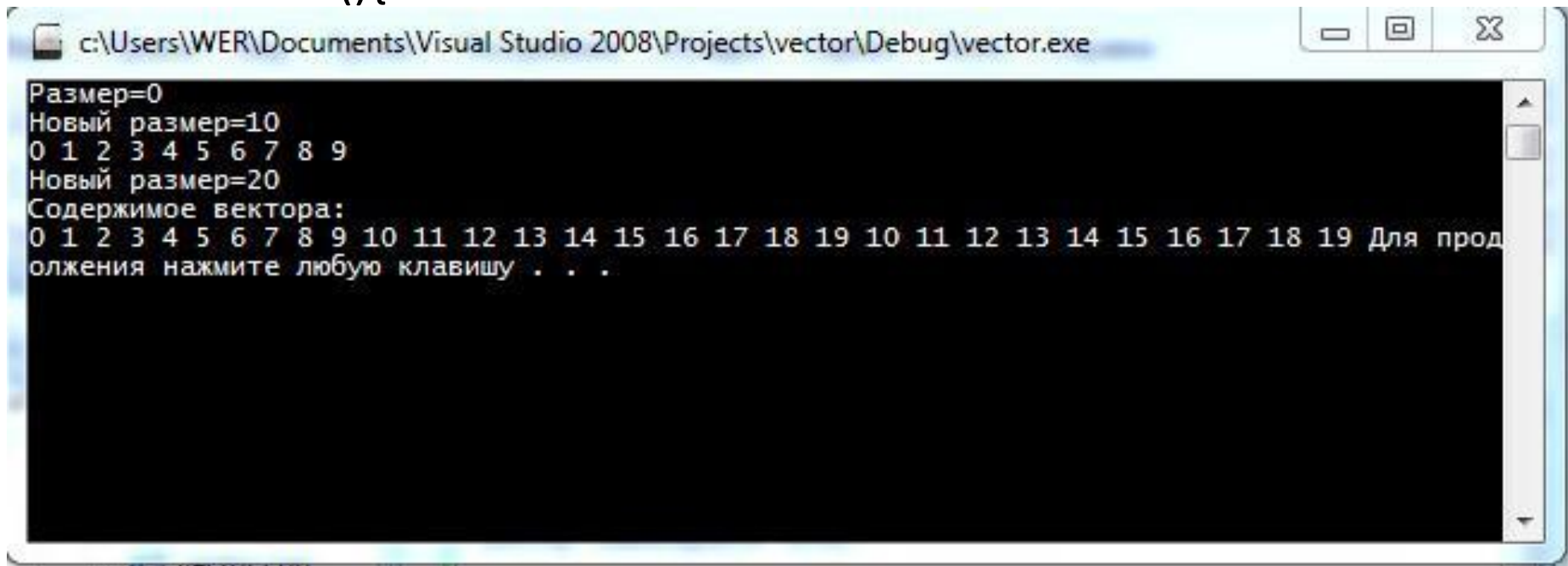
Функция-член	Описание
<code>Void clear()</code>	Удаляет все элементы из вектора
<code>Size_type max_size() const;</code>	Возвращает максимальное число элементов, которое может содержать вектор
<code>Void pop_back();</code>	Удаляет последний элемент в векторе
<code>Void push_back(const T &val);</code>	Добавляет в конец вектора элемент, значение которого задано параметром <code>val</code>
<code>Iterator erase(iterator i);</code>	Удаляет элемент, адресуемый итератором <code>i</code> ;



Демонстрация базового поведения вектора

```
#include<iostream>
#include<vector>
using namespace std;
```

```
void main(){
```



```
c:\Users\WER\Documents\Visual Studio 2008\Projects\vector\Debug\vector.exe
Размер=0
Новый размер=10
0 1 2 3 4 5 6 7 8 9
Новый размер=20
Содержимое вектора:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 10 11 12 13 14 15 16 17 18 19 Для продолжения нажмите любую клавишу . . .
```

```
for (i=0, i<v.size(), i++)
```

▶ cout<<v[i]<<" ";

Доступ к вектору с помощью итератора

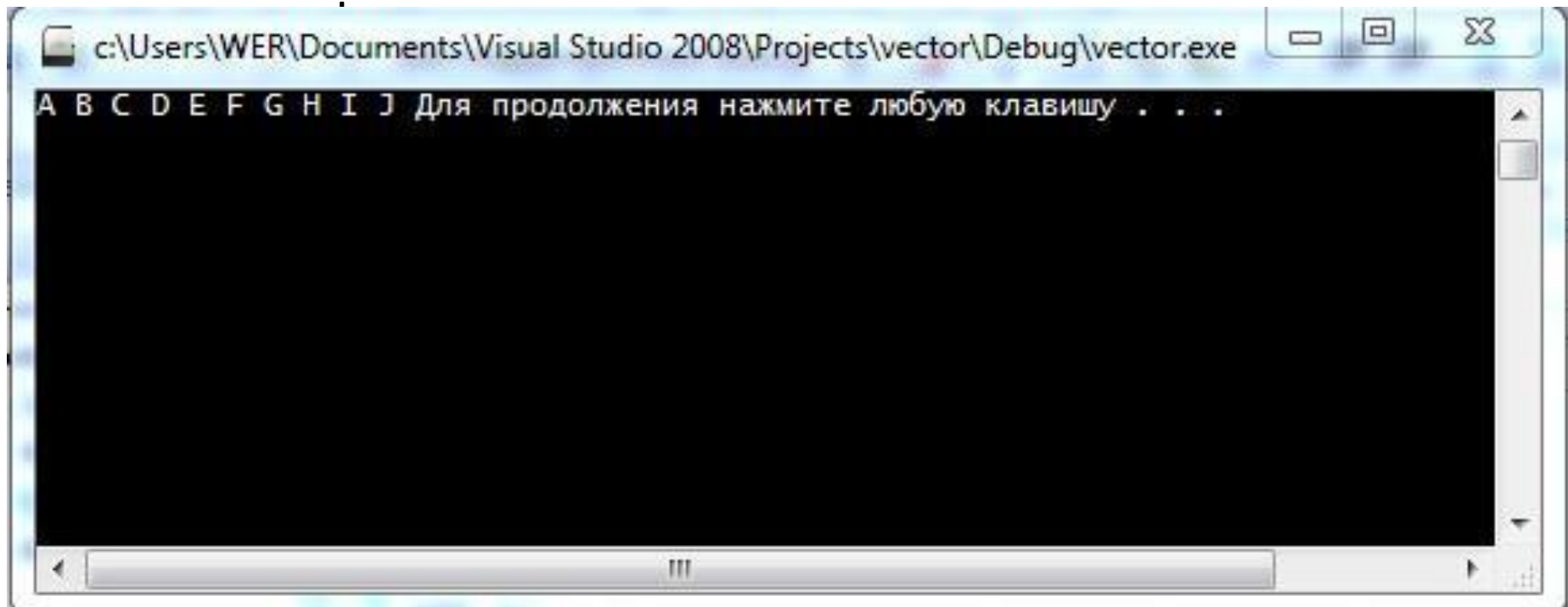
- В C++ массивы и указатели тесно связаны между собой
- В библиотеке STL аналогичная связь между векторами и итераторами
- Это значит, что к членам вектора можно обращаться с помощью индекса и с помощью итератора



Доступ к вектору с помощью итератора

```
#include<iostream>
#include<vector>
using namespace std;

void main(){
```



cout << r << endl;

Вставка и удаление элементов из вектора

- Есть возможность вставлять элементы в середину вектора, используя функцию `insert()`. А удалять с помощью `erase()`.



Вставка и удаление элементов из вектора

```
#include<iostream>
#include<vector>
using namespace std;
void main(){
vector<char>v;
int l;
for(i=0;i<10;i++)
v.push_back('A'+i);

for(i=0;i<v.size();i++)
cout<<v[i]<<" ";

vector<char>::iterator p=
```

Списки

- Список-это контейнер с двунаправленным последовательным доступом к его элементам
- Класс list позволяет получать к своим элементам только последовательный доступ в двух направлениях: от начала к концу и от конца к началу



Элементы можно помещать в список как с начала, так и с конца

```
#include<list>
```

```
void main(){
```

```
list<char> lst;
```

```
list<char> revlst;
```

```
int i;
```

```
for(i=0;i<10;i++)
```

```
lst.push_back('A'+i);
```

```
list<char>::iterator p;
```

```
while(!lst.empty()){
```

```
p=lst.begin();
```

```
cout<<*p;
```

```
▶ revlst.push_front(*p);
```