Указатель - переменная, значением которой является адрес ячейки памяти.
& - операция взятия адреса

```
int var = 123; // инициализация переменной var числом 123

int *ptrvar = &var; // указатель на переменную var (присвоили адрес переменной указателю)

   cout << "&var    = " << &var << endl;// адрес переменной var содержащийся в памяти, извлечённый операцией взятия адреса      &var    = 0x7fff2bc96cac

   cout << "ptrvar  = " << ptrvar << endl;// адрес переменной var, является значением указателя ptrvar                                ptrvar  = 0x7fff2bc96cac

   cout << "var     = " << var << endl; // значение в переменной var
                                                                    var     = 123

   cout << "*ptrvar = " << *ptrvar << endl; // вывод значения содержащегося в переменной var через указатель, операцией разименования указателя
                                                                    *ptrvar = 123
```

Указатели в массивах

```
int a[3] = {101,105,107};
int *q = a;
cout << *q << endl;

int *p = &a[1];

cout << *p << endl;
cout << *(p + 1) << endl;
p--;
cout << *p << endl;
```

101

105
107

101

Контейнер вектор

```
#include <vector>
vector <string> s ;
vector <int> a = {1, 2, 3};

//Вывод:
for (int el:a)
        cout << el << " ";
```

Методы

a.size() -> 3

a.empty() -> false

a.pop_back() -> a = {1, 2};

a.push_back(5) -> a = {1, 2, 5};

a.front() -> 1

a.back() -> 5

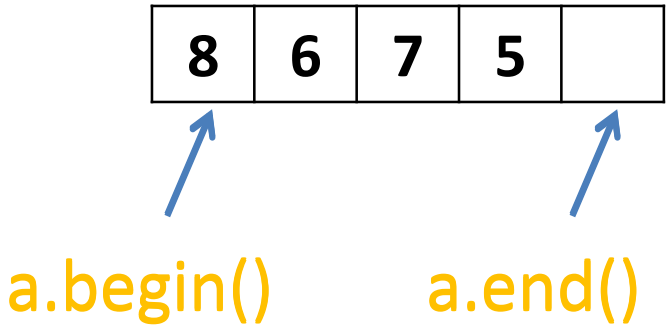
a.resize(2) -> a = {1, 2}

//осталось 2 элемента

a.resize(5, 3) -> 1 2 3 3 3

a.clear()

Итератор - это специальный класс, связанный с соответствующим классом контейнера.




```
vector<int> a = {8,6,7,5};
vector<int>::iterator it;
it = a.begin() + 3;//указывает на 3_ый элемент (5)
cout << *(it - 1);//содержимое 2-ого (7)
```

```
#include <iostream>
#include<vector>
using namespace std;
int main() {
 int x;    vector <int> a;
 //Ввод
while (cin>> x)
       a.push_back(x);

//Ввод n эл-тов
for(int i; i < n;i++){
     cin x;
     a.push_back(x);
}
//Вывод:
for (auto el:a)
       cout << el << " ";
//Вывод:
for (int i = 0;i < a.size();i++)
     cout << a[i] << " ";
}
```

```
#include <iostream>
#include<vector>
using namespace std;
int main()
{
 int x;    vector <int> a;
 while (cin>> x)
       a.push_back(x);

vector<int>::iterator it1;
for (it1 = a.begin(); it1< a.end(); it1++)
              cout << *it1<< " ";
cout << endl;

vector<int>::reverse_iterator it2;
for( it2 = a.rbegin(); it2 != a.rend(); it2++)
            cout<<*it2<<" ";
}
```

Ввод 1 0 7

1 0 7

7 0 1

#include <algorithm> ИЛИ #include <bits/stdc++.h>

```
vector<int>::iterator it;
 it =  find(A.begin(), A.end(), 27) ;
 cout << it -  A.begin();
```

4 1 27 1 2 3

Вывод 2

```
it =  find(A.begin() + 3 , A.end(), 27) ;
cout << it -  A.begin();
```

Вывод 6

```
 it =  find(A.begin() , A.end(), 27) ;
  cout << it -  A.end();
```

Вывод -4

```
 it =  find(A.begin() , A.end(), 7) ;
  cout << it -  A.end();
```

Вывод 0

```
 it =  find(A.begin() , A.end(), 7) ;
  cout <<( it ==  A.end());
```

Вывод 1

max, min, sort

4 1 27 1 2 3

```
int maxIndex = max_element(A.begin(), A.end()) - A.begin();
```

```
int maxEl = *max_element(A.begin(), A.end());
```

2 27

```
int minIndex = min_element(A.begin(), A.end()) - A.begin();
```

```
int minEl = *min_element(A.begin(), A.end());
```

1 1

```
sort(A.begin(), A.end());
```

1 1 2 3 4 27

```
sort(A.begin(), A.end(), greater());
```

27 4 3 2 1 1

Вставка и удаление

A = {4 1 27 3}

A.insert(A.begin(), 7) 7 4 1 27 3

A.insert(A.begin() + 2, 36) 7 4 36 1 27 3

A.insert(A.begin() + 6, 43) 7 4 36 1 27 3 43

R.insert(R.end(), A.begin() + 3, A.end()); дописываем часть массива A в конец массива R

it = A.insert(A.end(),7);
it указывает на добавленный элемент

Вставка и удаление

A = {7 4 36 1 27 3}

```
A.erase( A.begin( ) );
cout << *A.begin();
```

4

```
e = A.erase(A.begin( ) + 1, A.begin( ) + 3 );
cout << *e<< endl;
```

4 27 3
27

Метод удаления возвращает итератор, указывающий на первый элемент, оставшийся после удаленных элементов, или на указатель конца вектора, если такого элемента не существует.

rotate(first, middle, last)

A = {4 1 27 3}

<algorithm>

first: Итератор, который указывает на первый индекс или вектор массива, из которого мы хотим повернуть элементы.
middle: Итератор, который указывает на элемент массива или вектора, в котором будет выполняться вращение. Теперь этот элемент находится в начале массива.
last: Это итератор, который указывает на последний индекс массива или вектора до того места, где мы хотим повернуть элементы.

Сдвиг влево

A = {7, 4, 36, 1, 27, 3, 43}

```
rotate(A.begin(), A.begin() + 3, A.end());
```

1 27 3 43 7 4 36

```
rotate(A.begin() + 2, A.begin() + 3, A.end());
```

7 4 1 27 3 43 36

```
rotate(A.begin() + 2, A.begin() + 3, A.end() - 1);
```

7 4 1 27 3 36 43