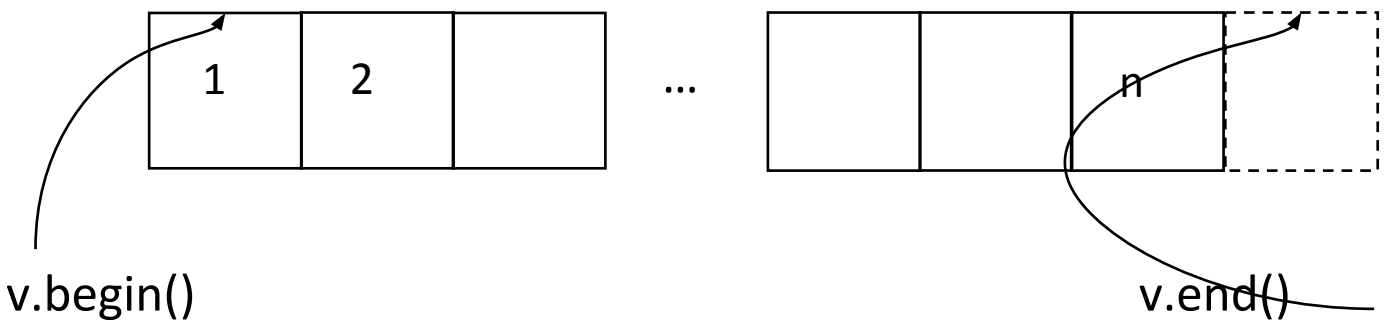
# Работа с классом vector

# Методы доступа к элементам контейнера

- begin(),  begin() const – итератор начала
- end(),  end() const – итератор конца
- rbegin(),  rbegin() const – реверсивный итератор начала (конец)
- rend(),  rend() const – реверсивный итератор конца (начало)



# Методы общие для всех контейнеров

- size() – количество элементов;
- max_size() – максимальное количество элементов до перераспределения;
- empty() – проверка на пустоту.

# Создание элементов vector

```
#incude <vector>
int main(){
vector <int> v1;         //пустой вектор целых чисел
vector <int> v2(10);     //вектор из 10 целых чисел
vector <int> v3(15, 5);//из 15 целых чисел "5"
vector <int> v6 {1,2,3};
vector <int> v4(v2);     //копия вектора v2
//копия части вектора v3 - с 0 по 2 элемент
vector <int> v5(v3.begin(), v3.begin() + 2);
vector <char> str(3);  //вектор из 3-х символов
```

# Присваивание в векторах

- vector<T>& operator = (const vector<T>& x);
- void assign(size_type n, const T& value);
- template <class InputIter> void assign(InputIter first, InputIter last);

```
vector <int> v1, v2;
v1 = v2;
v1.assign(15, 5);
v2.assign(v1.begin() + 2, v1.begin() + 6);
```

# Доступ к элементам вектора

- operator [ ] (size_type n);
- at(size_type n) ; // генерирует out_of_range;
- front() – ссылка на 1-й элемент;
- back() – ссылка на последний элемент;

# Пример доступа к элементам вектора

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

int main(){
try{
  vector <int> v(15, 5);
  v.front() = 100;
  v.back()  = 200;
  cout << v[0] << " "
    << v [v.size() - 1] << " "
    << v.at(v.size());
  }
catch(out_of_range){
  cout << "where are we go?";
  }
}
```

# Изменение объектов класса vector

- void push_back (const T& value);
- void pop_back ();
- iterator insert (iterator position, const T& value);
- iterator erase(iterator position);
- iterator erase(iterator first, iterator last);
- void swap();
- void clear() – удаляет все элементы вектора

# Пример изменения векторов

```
vector <int> v(2), v1(4,1); //v: 0 0; v1: 1 1 1 1
for (vector<int>::iterator i = v.begin(); i != v.end(); ++i)
    cout << *i;
int a[5] = {1, 2, 3, 4, 5};
vector <int> v2(a, a + 5); //1 2 3 4 5
v.insert(v.begin() + 1, 1);// v: 0 1 0
for (int i = 0; i < v.size(); i++)
    cout << v[i];
v.push_back(5);      // v: 0 1 0 5
v.erase(v.begin()); // v: 1 0 5
v1.swap(v); // v: 1 1 1 1

for (auto i = v.begin(); i != v.end(); ++i)
    cout << *i;
```

# Операции сравнения векторов

```
vector <int> v, u;
for (int i = 0; i < 6; i++)
   v.push_back(i);
for (int i = 0; i < 3; i++)
   u.push_back(i + 1);
if (v == u) //!=, <, >, …
   cout << "equal" << endl:
else
   cout << "not equal" << endl;
}
```

# Range-based for loop (since C++11)

```
vector <int> v {4, 3, 2, 1};
for (auto t : v)
    std::cout << t << ' ';
for (auto t : v)
    std::cin >> t;
```

# Пример применения обратного итератора

```
vector <int> v {1, 2, 3, 4, 5};
auto revIt = v.rbegin();
while(revIt != v.rend())
    cout << *revIt++ << ' ';
//5 4 3 2 1
```

# Алгоритмы STL

min_element/max_element – поиск минимального / максимального элемента

find – первый элемент с указанным значением

count – количество элементов с указанным значением

search – последовательность значений одного контейнера идентичная последовательности другого

swap – обмен значений

sort – сортировка последовательности

merge – слияние двух отсортированных последовательностей

# Алгоритм min_element/max_element

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

int main() {
vector<int> v{2, 1, 3, 4, 6, 5};
auto vmin = min_element(v.begin(), v.end());
cout << "min element at: " << distance(v.begin(), vmin);
auto vmax = max_element(v.begin(), v.end());
cout << "max element is: " << *vmax;
}
```

# Алгоритм find

```
vector<int> v{1, 2, 3, 4, 5, 6, 7, 8};
vector<int>::iterator ptrV;
ptrV = find(v.begin(), v.begin() + 8, 3);
cout << "element`s pos with value 3: "
     << (ptrV - v.begin()) << endl;
```

# Алгоритм count

```
vector<int> v{1, 3, 2, 3, 4, 5, 3, 7, 8};
cout << count(v.begin(), v.begin() + v.size(), 3);
```

# Алгоритм search

```
vector<int> v {1, 2, 3, 4, 5, 3, 4, 8},
            sv {3, 4};
auto ptrV = search(v.begin(), v.end(),
           sv.begin(), sv.end());
cout  << "1-st sv position in v: "
    << (ptrV - v.begin()) << endl;
```

# Алгоритм swap

```
vector<int> v1 {1, 2, 3, 4, 5, 3, 4, 8},
            v2 {3, 4};
v1.swap(v2);
```

# Алгоритм sort

```
vector<int> v {1.1, 5.1, 6.1, 3.1, 2.1, 7.1};
sort(v.begin(), v.end());
sort(v.begin(), v.begin() + 6, greater<float>());
```

# Алгоритм merge

```
vector<int> v {1, 2, 3, 4, 5, 8},
      sv {6, 7},
      mv(8);
merge(v.begin(), v.end(),
    sv.begin(), sv.end(),
    mv.begin());
```

# В массиве действительных чисел размера n выполнить циклический сдвиг элементов влево на 1 ячейку

```
int n;
cin >> n;
vector <float> mas(n);
for (unsigned i = 0; i < mas.size(); i++)
   cin >> mas[i];
float first = mas.front();
mas.erase(mas.begin());
mas.push_back(first);
for (unsigned i = 0; i < mas.size(); i++)
    cout << mas[i] << " ";
```

# В массиве действительных чисел размера n выполнить сортировку первых k элементов массива по возрастанию (k – натуральное число, k < n)

```
int k;
cout << "input number of sorting elements\n";
cin >> k;
if ((k > n)||(k < 0)){
   cout << "k must be >= 0 and <= array size";
     return 0;
   }
sort(mas.begin(), mas.begin() + k);
```

# В массиве действительных чисел размера n удалить все элементы, кратные 3 или 5

```
int n;
cin >> n;
vector <float> mas(n);
for (unsigned i = 0; i < mas.size(); i++)
   cin >> mas[i];
int i = 0;
while (i < mas.size())
   if ((fmod(mas[i], 3) == 0) || (fmod(mas[i], 5) == 0))
      mas.erase(mas.begin() + i);
   else
      i++;
```

# Двумерный динамический массив на базе vector (фиксированное число элементов в каждой строке)

```
int val, cols, rows;
vector< vector<int> > mas(rows, vector<int>());
for(int i = 0; i < mas.size(); i++){
   for(int j = 0; j < cols; j++){
      cin >> val;
      mas.at(i).push_back(val);
      }
   }
for(int i = 0; i < mas.size(); i++){
   for(int j = 0; j < mas.at(i).size(); j++)
    cout << mas[i][j] << " ";
cout << endl;
}
```

# Двумерный динамический массив на базе vector (произвольное число элементов в каждой строке)

```
int val, cols, rows;
vector< vector<int> > mas(rows, vector<int>());
for(int i = 0; i < mas.size(); i++){
   cout << "input size for the row " << i + 1 << endl;
   cin  >> cols;
   cout << "input " << cols << " elements for row" << i + 1 << endl;
   for(int j = 0; j < cols; j++){
      cin >> val;
      mas.at(i).push_back(val);
      }
   }
for(int i = 0; i < mas.size(); i++){
for(int j = 0; j < mas.at(i).size(); j++)
    cout << mas[i][j] << " ";
cout << endl;
}
```