STL

Тема 2. Обобщенные алгоритмы.

Часть 3. Алгоритмы связанные с сортировкой.

Sort\Stable_sort

Задача: сортировка последовательности

Сложность: N*log(N)

```
bool UDgreater ( int elem1, int elem2 )
{
        return elem1 > elem2;
}
{
    vector <int> v1;

    for (int i = 0 ; i <= 5 ; i++ )          v1.push_back( 2 * i );
    for (int ii = 0 ; ii <= 5 ; ii++ )   v1.push_back( 2 * ii + 1 );

    sort( v1.begin( ), v1.end( ) );
    sort( v1.begin( ), v1.end( ), greater<int>( ) );
    sort( v1.begin( ), v1.end( ), UDgreater );
}
```

partial_sort

Задача: сортировка части последовательности

Сложность: N*log(N)

```
bool UDgreater ( int elem1, int elem2 )
{
        return elem1 > elem2;
}
{
    vector <int> v1;

    for (int i = 0 ; i <= 5 ; i++ )          v1.push_back( 2 * i );
    for (int ii = 0 ; ii <= 5 ; ii++ )   v1.push_back( 2 * ii + 1 );

    partial_sort(v1.begin( ), v1.begin( ) + 6, v1.end( ) );
    partial_sort(v1.begin( ), v1.begin( ) + 4, v1.end( ), greater<int>( ) );
    partial_sort(v1.begin( ), v1.begin( ) + 8, v1.end( ), UDgreater );
    }
```

nth_element

Задача: Подставляет элемент в позицию, в которой он бы находился в случае отсортированной последовательности

Сложность: O(N)

```
vector<int> v(7);

 v[0] = 25; v[1] = 7; v[2] = 9;
 v[3] = 2; v[4] = 0; v[5] = 5; v[6] = 21;

nth_element(v.begin(), v.begin() + 4, v.end());
```

binary_search

Задача: Определить находится ли данный элемент в отсортированном контейнере.

Сложность: O(log(N))

```
list <int> L;
L.push_back( 50 );
L.push_back( 10 );
L.push_back( 30 );
L.push_back( 20 );
L.push_back( 25 );
L.push_back( 5 );

L.sort( );
bool b1 = binary_search( L.begin( ), L.end( ), 10 );

L.sort ( greater<int> ( ) );
bool b2 = binary_search( L.begin( ), L.end( ), 10 , greater<int> ( ) );
```

lower_bound / upper_bound / equal_range

Задача: Найти в отсортированном контейнере первую(последнюю) позицию куда может быть вставлен заданный элемент с сохранением порядка

Сложность: O(log(N))

```
vector<int> v(5);
v[1] = 7; v[2] = 7; v[3] = 7; v[4] = 8;

 vector<int>::iterator k;

 k = lower_bound(v.begin(), v.end(), 7);


 k = upper_bound(v.begin(), v.end(), 7);
```

merge

Задача: Объединить два отсортированных контейнера в один отсортированный

Сложность: O(M+N)

```
vector<int> vector1(5);
vector<int> vector2(5);
vector<int> vector3(10);

 for (int i = 0; i < 5; ++i)     vector1[i] = 2 * i;

 for (i = 0; i < 5; ++i)    vector2[i] = 1 + 2 * i;

 merge( vector1.begin(), vector1.end(),
            vector2.begin(), vector2.end(),
            vector3.begin());
```

max_element / min_element

Задача: Найти итератор максимального (минимального) элемента в заданном диапазоне

```
vector<int> vector1(5);
 for(int i = 0; i < 5; ++i)
   vector1[i] = i;

 random_shuffle(vector1.begin(), vector1.end());

vector<int>::iterator k =   max_element(vector1.begin(), vector1.end());


k = min_element(vector1.begin(), vector1.end());
```

Теоретико-множественные операции

includes

Задача: Проверить содержится ли элементы одного сортированного диапазона в другом сортированном диапазоне

```
bool result;

vector<char> vector1 = make< vector<char> >("abcde"),
             vector2 = make< vector<char> >("aeiou");


result = includes(vector1.begin(), vector1.end(),
                  vector2.begin(), vector2.end());


result = includes(vector1.begin(), vector1.end(),
                  vector2.begin(), vector2.begin() + 2);
```

set_union

Задача: объединить элементы двух сортированных диапазонов в один сортированный диапазон

```
bool result;

vector<char> vector1 = make< vector<char> >("abcde"),
             vector2 = make< vector<char> >("aeiou");


vector<char> setUnion;
set_union(vector1.begin(), vector1.end(),
          vector2.begin(), vector2.end(),
          back_inserter(setUnion));
```

set_intersection / set_difference / set_symmetric_difference

Задача: Получить для двух сортированных диапазонов диапазон элементов содержащихся только в первом диапазоне, элементов общих для обоих диапазонов, элементов содержащихся только в одном из диапазонов

```
vector<char> vector1 = make< vector<char> >("abcde"),
             vector2 = make< vector<char> >("aeiou");

vector<char> setIntersection;
set_intersection(vector1.begin(), vector1.end(),
     vector2.begin(), vector2.end(),      back_inserter(setIntersection));

assert (setIntersection == make< vector<char> >("ae"));

vector<char> setDifference;
set_symmetric_difference(vector1.begin(),  vector1.end(),
     vector2.begin(), vector2.end(),      back_inserter(setDifference));

assert (setDifference == make< vector<char> >("bcdiou"));
```

Задание

1. Считать из файла информацию об машинах находящихся на трассе:
    1. Положение (км)
    2. Скорость
    3. Гос. Номер
    4. Значение в километрах финиша маршрута
2. Вывести их в лексикографическом порядке, в порядке возрастания скорости, только автомобили 78 региона
3. По введенному Гос. Номеру определять есть ли данный автомобиль на трассе
4. Вывести положение автомобилей через заданное время T в упорядоченном относительно расстояния до финиша
5. Найти объединение, пересечение и разность следующих множеств: множества первых 20 чисел фибоначи и первых 50 четных .