STL

Тема 2. Обобщенные алгоритмы.

Часть 2. Изменяющие алгоритмы.

Изменяющие алгоритмы

- copy
- copy_backward
- fill
- generate
- partition
- random_shuffle
- remove
- replace
- remove
- rotate
- swap
- swap_ranges
- transform
- unique

copy \ copy_backward

Задача: Копирование из одного диапазона последовательности в другой

Сложность: линейная

```
vector <int> v1, v2;
for ( int i = 0 ; i <= 5 ; i++ )
            v1.push_back( 10 * i );
for (int  ii = 0 ; ii <= 10 ; ii++ )
            v2.push_back( 3 * ii );

copy_backward( v1.begin( ), v1.begin( ) + 3, v2.begin( ) + 7 );
```

fill \ fill_n

Задача: Помещает копии данного значения во все позиции диапазона

Сложность: линейная

```
vector <int> v1;
for (int i = 0 ; i <= 9 ; i++ )
{
        v1.push_back( 5 * i );
}

fill( v1.begin( ) + 5, v1.end( ), 2 );

fill_n( v1.begin( ) + 7, 3, 2 );
```

generate

Задача: Заполняет диапазон значениями генерируемыми подставленной функцией

Сложность: линейная

```
template <typename T>
class calc_square {
  T i;
 public:
              calc_square(): i(0) {}
              T operator()() { ++i; return i * i; }
};

int main()
{
             vector<int> vector1(10);
             generate(vector1.begin(), vector1.end(), calc_square<int>());
}
```

Partition \ stable_partition

Задача: для данного диапазона и унарного предиката перегруппировывает последовательность так, что бы удовлетворяющие предикату элементы находились перед неудовлетворяющими

Сложность: линейная

```
bool greater5 ( int value )
{
        return value >5;
}

vector <int> v1;
// заполнение вектора

partition ( v1.begin( ), v1.end( ), greater5 );
```

random_shuffle

Задача: случайным образом пересортировать элементы в диапазоне

Сложность: линейная

```
vector <int> v1;
 for (int i = 1 ; i <= 9 ; i++ )
                v1.push_back( i );

random_shuffle( v1.begin( ), v1.end( ) );
```

remove

Задача: удалить из диапазона элементы равное заданному

Сложность: линейная

```
const int  N = 11;
int array1[N] = {1, 2, 0, 3, 4, 0, 5, 6, 7, 0, 8};
vector<int> vector1;

for (int i = 0; i < N; ++i)
    vector1.push_back(array1[i]);

vector<int>::iterator new_end = remove(vector1.begin(), vector1.end(), 0);

 vector1.erase(new_end, vector1.end());
```

replace

Задача: Заменить элементы равные заданному значению другим значением

Сложность: линейная

```
vector <int> v1;

for (int i = 0 ; i <= 9 ; i++ )
            v1.push_back( i );
for (int ii = 0 ; ii <= 3 ; ii++ )
             v1.push_back( 7 );

random_shuffle (v1.begin( ), v1.end( ) );

replace (v1.begin( ), v1.end( ), 7 , 700);
```

reverse

Задача: Обратить элементы в заданном диапазоне

Сложность: линейная

Итератор: двунаправленный

```
vector <int> v1;
for (int i = 0 ; i <= 9 ; i++ )
{
        v1.push_back( i );
 }

reverse (v1.begin( ), v1.end( ) );
```

rotate

Задача: Выполнить циклический сдвиг диапазона

Сложность: линейная

Итератор: двунаправленный

```
vector <int> v1;
for (int i = -3 ; i <= 5 ; i++ )
{
        v1.push_back( i );
 }

rotate ( v1.begin ( ) , v1.begin ( ) + 3 , v1.end ( ) );
```

swap

Задача: Поменять значения контейнеров местами

Сложность: для встроенных типов константная

```
vector <int> v1, v2;
for (int i = 0 ; i <= 10 ; i++ )
{
            v1.push_back( i );
}
for (int ii = 0 ; ii <= 4 ; ii++ )
{
             v2.push_back( 5 );
 }

swap( v1, v2 );
```

swap_ranges

Задача: Обменивает два диапазона значение (возможно из разных контейнеров)

Сложность: линейная

```
vector <int> v1;
deque <int> d1;
for (int i = 0 ; i <= 5 ; i++ )
            v1.push_back( i );
for (int ii =4 ; ii <= 9 ; ii++ )
            d1.push_back( 6 );

swap_ranges ( v1.begin ( ) , v1.end ( ) , d1.begin ( ) );
```

transform

Задача: применить унарную(бинарную) функцию ко все элементам диапазона и сохранить полученные значения в другом диапазоне

Сложность: линейная

```
int sum(int val1, int val2) { return val2 + val1; }

int main()
{
   int array1[5] = {0, 1, 2, 3, 4};
  int array2[5] = {6, 7, 8, 9, 10};
  ostream_iterator<int> out(cout, " ");

   transform(&array1[0], &array1[5], &array2[0], out, sum);

  cout << endl;
  return 0;
}
```

unique

Задача: Удалить все последовательные дублирующие элементы

Сложность: линейная

```
int main()
{
  const int N = 11;
  int array1[N] = {1, 2, 0, 3, 3, 0, 7, 7, 7, 0, 8};
  vector<int> vector1;
  for (int i = 0; i < N; ++i)
    vector1.push_back(array1[i]);

  vector<int>::iterator new_end;
  new_end = unique(vector1.begin(), vector1.end());

 vector1.erase(new_end, vector1.end());
 copy(vector1.begin(), vector1.end(),
       ostream_iterator<int>(cout, " "));

  return 0;
}
```

Практическое задание

- Создать контейнер состоящий из 1 – 1, 2 -2, 3 -3 …10-10
- Перераспределить его произвольным образом
- Сдвинуть влево не 3 позиции
- Сдвинуть по кругу на 5 позиций
- Создать контейнер состоящий из 55 первых чисел Фибоначчи
- Поменять местами первые 20 членов двух контейнеров местами
- Отсортировать второй контейнер, удалить подряд идущие одинаковые элементы
- Скопировать из первого контейнера только четные элементы