STL

Тема 2. Обобщенные алгоритмы.

Часть 1. Неизменяющие алгоритмы.

Обобщенные алгоритмы

- Неизменяющие алгоритмы
- Изменяющие алгоритмы
- Связанные с сортировкой алгоритмы
- Обобщенные числовые алгоритмы

Обобщенные алгоритмы

- Изменяющие себя

```
 int a[1000];
int i;
for (i = 0; i < 1000; ++i)
  a[i] = 1000 - i – 1;
sort(&a[0], &a[1000]);
for (i = 0; i < 1000; ++i)
  assert (a[i] == i);
```

- Копирующие

```
int a[1000], b[1000];
 int i;
 for (i = 0; i < 1000; ++i)
   a[i] = I;
 reverse_copy(&a[0], &a[1000], &b[0]);
 for (i = 0; i < 1000; ++i)
    assert (a[i] == i && b[i] == 1000 - i - 1);
```

Алгоритмы с предикатами

```
template<typenama Type>
struct greater : public binary_function <Type, Type, bool>
{
     bool operator()( const Type& _Left, const Type& _Right ) const;
};
int main( )
{
    using namespace std;
    vector <int> v1;
     for (int i = 0 ; i < 8 ; i++ )
            { v1.push_back( rand( ) ); }

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

Неизменяющие алгоритмы

- find
- adjacent_find
- count
- for_each
- mismatch
- equal
- search

find и find_if

Задача: Поиск элемента в последовательности

Сложность : линейная

```
class GreaterThan50 {
 public:
  bool operator()(int x) const { return x > 50; }
};

int main()
{
   vector<int> vector1;
  for (int i = 0; i < 13; ++i)
     vector1.push_back(i * i);

  vector<int>::iterator where;
  where = find_if(vector1.begin(), vector1.end(),
                  GreaterThan50());

  assert (*where == 64);
  return 0;
}
```

adjacent_find

Задача: Поиск подряд стоящий одинаковых элементов

Сложность : линейная

```
bool twice (int elem1, int elem2 )
 { return elem1 * 2 == elem2; }

 int main( )
 {
        list <int> L;
        list <int>::iterator Iter;
        list <int>::iterator result1, result2;
        L.push_back(50); L.push_back(40); L.push_back(10); L.push_back(20);
        L.push_back(20);
        result1 = adjacent_find( L.begin( ), L.end( ) );
        result2 = adjacent_find( L.begin( ), L.end( ), twice );
}
```

count

Задача: поиск количества значений равных данному

Сложность: линейная

```
int a[] = {0, 0, 0, 1, 1, 1, 2, 2, 2};

  // Count the number of values in the array a
  // that are equal to 1:
  int final_count = count(&a[0], &a[9], 1);
```

for_each

Задача: Применение функции к каждому элементу последовательности

Сложность: линейная

```
void print_list(string s)
{
  cout << s << endl;
}

int main()
{
   list<string> dlist;
  dlist.insert(dlist.end(), “Ivan");
  dlist.insert(dlist.end(), “Petr");
  dlist.insert(dlist.end(), “Elena");

  for_each(dlist.begin(), dlist.end(), print_list);
  return 0;
}
```

mismatch и equal

Задача: сравнение двух последовательностей

Сложность: линейная

```
bool b;
b = equal( v1.begin( ), v1.end( ), v2.begin( ) );
```

```
pair<vector <int>::iterator, list <int>::iterator> results1;
results1 = mismatch (v1.begin( ), v1.end( ), L1.begin( ));
```

search

Задача: Поиск позиции в которой вторая последовательность входит в первую

Сложность: O(nm)

```
int i;
for (i = 0; i < 20; ++i)
  vector1[i] = I;

for (i = 0; i < 5; ++i)
  deque1[i] = i + 5;

vector<int>::iterator k =
  search(vector1.begin(), vector1.end(),
         deque1.begin(), deque1.end());
```

Практическое задание

В файле записана последовательность натуральных чисел. Требуется:

1. Найти сколько раз в последовательности встречается число равное заданному
2. Найти сколько раз встречается число для кадратный корень которого равен заданному
3. Найти пару рядом стоящих равных чисел и все пары различающиеся в три раза
4. Каждый элемент возвести в квадрат и записать в файл
5. Выяснить входит ля заданная подпоследовательность в исходную