**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №6**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «Использование стандартной библиотеки шаблонов»

**Вариант №9**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6307 |  | Лазарев С.О.  Медведев Е.Р. |
| Преподаватель |  | Колинько П.Г. |

Санкт-Петербург

2018

Содержание

[ЦЕЛЬ 3](#_Toc514756635)

[ЗАДАНИЕ 3](#_Toc514756636)

[ФОРМАЛИЗАЦИЯ ЗАДАНИЯ 4](#_Toc514756637)

[НАБОР ПОДХОДЯЩИХ КОНТЕЙНЕРОВ, ФУНКЦИИ STL 4](#_Toc514756638)

[ВРЕМЕННАЯ СЛОЖНОСТЬ 5](#_Toc514756639)

[ВЫВОДЫ 7](#_Toc514756640)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8](#_Toc514756641)

[ПРИЛОЖЕНИЕ 9](#_Toc514756642)

# 

# ЦЕЛЬ

Получить практические навыки по работе со стандартной библиотекой шаблонов.

# ЗАДАНИЕ

Составить и отладить программу для использования контейнеров из стандартной библиотеки шаблонов.

F = A ⊕ B ∩ C ∪ D ∪ E.

# ФОРМАЛИЗАЦИЯ ЗАДАНИЯ

Для хранения множеств выбрать контейнер подходящего типа и доработать его для поддержки операций с последовательностями. Для реализации операций с контейнерами использовать возможности библиотеки алгоритмов. Программа должна реализовывать цепочку операций с последовательностями.

- EXCL – вторая последовательность исключается из первой, если она является ее частью.

- SUBST – вторая последовательность включается в первую с указанной позиции p.

- CONCAT – вторая последовательность подсоединяется к концу первой, образуя ее продолжение.

# НАБОР ПОДХОДЯЩИХ КОНТЕЙНЕРОВ, ФУНКЦИИ STL

Структура данных представлена в форме контейнера стандартной библиотеки шаблонов set. Данная структура данных хранит множество в виде дерева двоичного поиска с автобалансировкой (красно-черное дерево), причем контейнер set хранит множество ключей

Также для реализации функций над последовательностями был использован контейнер вектор указателей на значения элементов множества.

Из библиотеки algorithm были использованы методы set\_difference, set\_symmetri\_difference, set\_intersection для вычитания, пересечения множеств.

**Пример**:

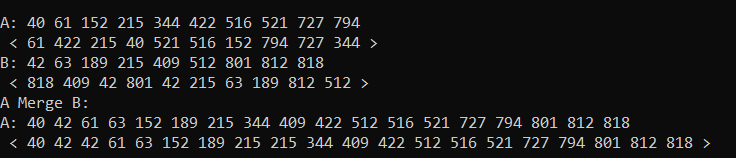


Рис 1. MERGE

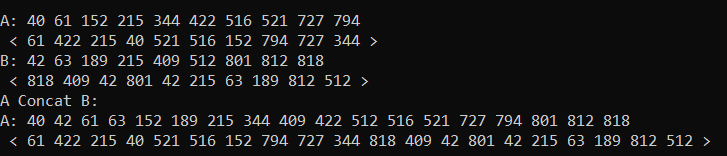


Рис 2. CONCAT

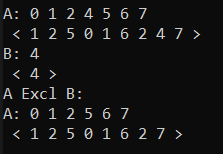


Рис 3. EXCL

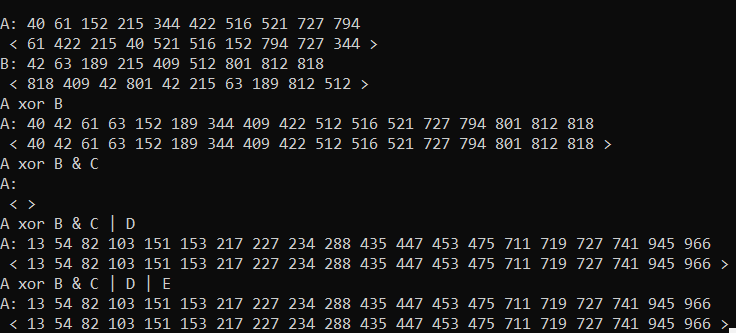


Рис 4. Множества

# ВРЕМЕННАЯ СЛОЖНОСТЬ

Временная сложность алгоритмов STL представлена в таблице 1.

Временная сложность функций программы представлена в таблице 2.

Таблица. 1. Временная сложность алгоритмов STL

|  |  |
| --- | --- |
| Функция | Временная сложность |
| vector::push\_back | O(1) |
| set::insert | O(ln n) |
| set\_intersection | O(n) |
| set\_difference | O(n) |
| set\_symmetric\_difference | O(n) |
| Swap | O(1) |
| Size | O(1) |

Таблица. 2. Временная сложность функций программы

|  |  |  |
| --- | --- | --- |
| Функция | Средняя | Худшая |
| setAnd | O(n) | O(n) |
| setDif | O(n) | O(n) |
| setSymDif | O(n) | O(n) |
| Excl | O(n) | O(n) |
| Subst | O(n) | O(n) |
| Concat | O(n) | O(n) |

# ВЫВОДЫ

В данной работе мы изучали некоторые элементы стандартной библиотеки шаблонов, в частности контейнеры set, vector, а также работу с ними. Мы научились использовать данные контейнеры, стандартные алгоритмы для работы с ними, изучили структуру контейнеров.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритмы и структуры данных. – Методические указания к лабораторным работам, практическим занятиям и курсовому проектированию, часть 2, глава 1 «Работа с иерархией объектов: наследование и полиморфизм».

# ПРИЛОЖЕНИЕ

***main.cpp***

#include <iostream>

#include <algorithm>

#include <set>

#include <ctime>

#include <iterator>

#include <chrono>

#include <vector>

using namespace std;

using MySet = std::set<int>;

using MyIt = std::set<int>::iterator;

using MySeq = std::vector<MyIt>;

const int lim = 1000; //ОГРАНИЧИТЕЛЬ для множества ключей

class MyCont

{

int power;

char tag;

MySet A;

MySeq sA;

MyCont& operator = (const MyCont&) = delete;

MyCont& operator = (MyCont&&) = delete;

public:

MyCont(int, char);

MyCont(const MyCont&);

MyCont(MyCont&&) noexcept;

MyCont& operator |= (const MyCont&);

MyCont operator | (const MyCont& rgt) const

{

MyCont result(\*this); return (result |= rgt);

}

MyCont& operator &= (const MyCont&);

MyCont operator & (const MyCont& rgt) const

{

MyCont result(\*this); return (result &= rgt);

}

MyCont& operator -= (const MyCont&);

MyCont operator - (const MyCont& rgt) const

{

MyCont result(\*this); return (result -= rgt);

}

void Merge(const MyCont&);

void Concat(const MyCont&);

void Mul(int);

void Erase(size\_t, size\_t);

void Excl(const MyCont&);

void Subst(const MyCont&, size\_t);

void Change(const MyCont&, size\_t);

void Show() const;

size\_t Power() const { return sA.size(); }

void PrepareExcl(const MyCont&); //Подготовка excl

friend void PrepareAnd(MyCont&, MyCont&, const int);

//подготовка and и sub

};

MyCont::MyCont(int p = 0, char t = 'R') : power(p), tag(t)

{

for (int i = 0; i < power; ++i)

{

sA.push\_back(A.insert(std::rand() % lim).first);

}

}

MyCont::MyCont(MyCont&& source) noexcept //Копия "с переносом"

: power(source.power), tag(source.tag),

A(std::move(source.A)), sA(std::move(source.sA)) { }

MyCont::MyCont(const MyCont& source) //Конструктор копии

: power(source.power), tag(source.tag)

{

for (auto x : source.A) sA.push\_back(A.insert(x).first);

}

void MyCont::Show() const

{

using std::cout;

cout << "\n" << tag << ": ";

/\* unsigned n = A.bucket\_count( ); //Вариант: выдача для ХТ

for (auto i = 0; i < n; ++i)

if (A.bucket\_size(i))

{

cout << "\n" << i << "("<< A.bucket\_size(i) << "):" ;

// auto it0 = A.begin(i), it1 = A.end(i);

for (auto it = A.begin(i); it != A.end(i); ++it) cout << " " << \*it;

} \*/

for (auto x : A)

{

cout << x << " "; //Выдача множества

}

cout << "\n < ";

for (auto x : sA)

{

cout << \*x << " "; //Выдача последовательности

}

cout << ">";

}

void PrepareAnd(MyCont& first, MyCont& second, const int quantity)

{

for (int i = 0; i < quantity; ++i)

{ //Подготовка пересечения:

int x = rand() % lim; // добавление общих эл-тов

first.sA.push\_back(first.A.insert(x).first);

second.sA.push\_back(second.A.insert(x).first);

}

}

MyCont& MyCont::operator -= (const MyCont& rgt)

{ //Разность мн-в

MySet temp;

MySeq stemp;

for (auto x : A)

{

if (rgt.A.find(x) == rgt.A.end())

{

stemp.push\_back(temp.insert(x).first);

}

}

temp.swap(A);

stemp.swap(sA);

return \*this;

}

MyCont& MyCont::operator &= (const MyCont& rgt) { //Пересечение

MySet temp;

MySeq stemp;

for (auto x : A)

{

if (rgt.A.find(x) != rgt.A.end())

{

stemp.push\_back(temp.insert(x).first);

}

}

temp.swap(A);

stemp.swap(sA);

return \*this;

}

MyCont& MyCont::operator |= (const MyCont& rgt)

{ //Объединение

for (auto x : rgt.A)

{

sA.push\_back(A.insert(x).first);

}

return \*this;

}

void MyCont::Erase(size\_t p, size\_t q)

{ //Исключение фр-та от p до q

using std::min;

size\_t r(Power());

p = min(p, r);

q = min(q + 1, r);

if (p <= q)

{

MySet temp;

MySeq stemp;

for (size\_t i = 0; i < p; ++i)

{

stemp.push\_back(temp.insert(\*sA[i]).first);

}

for (size\_t i = q; i < r; ++i)

{

stemp.push\_back(temp.insert(\*sA[i]).first);

}

A.swap(temp);

sA.swap(stemp);

}

}

void MyCont::Mul(int k)

{ //Размножение (не более чем в 5 раз)

auto p = sA.begin(), q = sA.end();

if (p != q && (k = k % 5) > 1)

{ //Пропуск, если мн-во пусто или k < 2

std::vector<int> temp(A.begin(), A.end());

MySeq res(sA);

for (int i = 0; i < k - 1; ++i)

{

std::copy(p, q, back\_inserter(res));

A.insert(temp.begin(), temp.end());

}

sA.swap(res);

}

}

void MyCont::Merge(const MyCont& rgt)

{ //Слияние

using std::sort;

MySeq temp(rgt.sA), res;

auto le = [](MyIt a, MyIt b)->bool

{ return \*a < \*b; };//Критерий

sort(sA.begin(), sA.end(), le);

sort(temp.begin(), temp.end(), le);

std::merge(sA.begin(), sA.end(), temp.begin(), temp.end(),

std::back\_inserter(res), le); //Слияние для последовательностей...

A.insert(rgt.A.begin(), rgt.A.end()); //... и объединение множеств

sA.swap(res);

}

void MyCont::PrepareExcl(const MyCont& rgt)

{

//Подготовка объекта исключения в пустом контейнере...

int a = rand() % rgt.Power(), b = rand() % rgt.Power();

//... из случайного [a, b] отрезка rgt

if (b > a)

{

for (int x = a; x <= b; ++x)

{

int y = \*(rgt.sA[x]); sA.push\_back(A.insert(y).first);

}

}

}

void MyCont::Excl(const MyCont& rgt)

{ //Исключение подпоследовательности

size\_t n(Power()), m(rgt.Power());

if (m) for (size\_t p = 0; p < n; ++p)

{ //Поиск первого элемента

bool f(true);

// int a(\*sA[p]), b(\*rgt.sA[0]); //ОТЛАДКА

if (\*sA[p] == \*rgt.sA[0])

{ //Проверка всей цепочки

size\_t q(p), r(0);

if (m > 1)

{

do

{

++q, ++r;

size\_t c(\*sA[q]), d(\*rgt.sA[r]);

f &= c == d;

} while ((r < m - 1) && f);

}

if (f)

{//Цепочки совпали, удаляем

MySet temp;

MySeq stemp;

for (size\_t i = 0; i < p; ++i)

{

stemp.push\_back(temp.insert(\*sA[i]).first);

}

for (size\_t i = p + m; i < Power(); ++i)

{

stemp.push\_back(temp.insert(\*sA[i]).first);

}

A.swap(temp);

sA.swap(stemp);

break;

}

}

}

}

void MyCont::Concat(const MyCont& rgt)

{ //Сцепление

for (auto x : rgt.sA)

{

sA.push\_back(A.insert(\*x).first);

}

}

void MyCont::Subst(const MyCont& rgt, size\_t p)

{ //Подстановка

if (p >= Power())

{

Concat(rgt);

}

else

{

MySeq stemp(sA.begin(), sA.begin() + p); //Начало

std::copy(rgt.sA.begin(), rgt.sA.end(), back\_inserter(stemp)); //Вставка

std::copy(sA.begin() + p, sA.end(), back\_inserter(stemp)); //Окончание

MySet temp;

sA.clear();

for (auto x : stemp)

{

sA.push\_back(temp.insert(\*x).first);

}

A.swap(temp);

}

}

void MyCont::Change(const MyCont& rgt, size\_t p)

{ //Замена

if (p >= Power())

{

Concat(rgt);

}

else

{

MySeq stemp(sA.begin(), sA.begin() + p); //Начало

std::copy(rgt.sA.begin(), rgt.sA.end(), back\_inserter(stemp));

//Замена

size\_t q = p + rgt.Power();

if (q < Power())

{

std::copy(sA.begin() + q, sA.end(), back\_inserter(stemp));

}

//Окончание

MySet temp;

sA.clear();

for (auto x : stemp)

{

sA.push\_back(temp.insert(\*x).first);

}

A.swap(temp);

}

}

int main()

{

using std::cout;

using namespace std::chrono;

setlocale(LC\_ALL, "Russian");

srand((unsigned int)7); //Пока здесь константа, данные повторяются

// srand((unsigned int)time(nullptr)); //Разблокировать для случайных данных

bool debug = true; //false, чтобы запретить отладочный вывод

auto MaxMul = 5;

int middle\_power = 0, set\_count = 0;

auto Used = [&](MyCont& t)

{

middle\_power += t.Power();

++set\_count;

};

auto DebOut = [debug](MyCont& t)

{

if (debug)

{

t.Show(); cout << " Press <Enter>"; cin.get();

}

};

auto rand = [](int d)

{

return std::rand() % d;

}; //Лямбда-функция!

// std::ofstream fout("in.txt"); //Открытие файла для результатов

int p = 10; //Текущая мощность (место для цикла по p)

//=== Данные ===

MyCont A(p, 'A');

MyCont B(p, 'B');

MyCont C(p, 'C');

MyCont D(p, 'D');

MyCont E(p, 'E');

MyCont F(p, 'F'); //Пустая заготовка для Excl

MyCont G(p, 'G');

MyCont H(p, 'H');

MyCont R(p);

A.Show();

B.Show();

std::cout << "\nA xor B";

((A | B) - (A & B)).Show();

std::cout << "\nA xor B & C";

(((A | B) - (A & B)) & C).Show();

std::cout << "\nA xor B & C | D";

(((A | B) - (A & B)) & C | D | E).Show();

std::cout << "\nA xor B & C | D | E";

(((A | B) - (A & B)) & C | D | E).Show();

//int q\_and(rand(MaxMul) + 1);

//PrepareAnd(A, R, q\_and);

//if (debug) A.Show(); Used(A);

//if (debug) R.Show(); Used(R);

////=== Цепочка операций ===

//// (Операция пропускается (skipped!), если аргументы некорректны)

////Идёт суммирование мощностей множеств и подсчёт их количества,

//// измеряется время выполнения цепочки

//auto t1 = std::chrono::high\_resolution\_clock::now();

//if (debug) cout << "\n=== R&=A === (q\_and=" << q\_and << ") ";

//R &= A; DebOut(R); Used(R);

//if (debug) B.Show(); Used(B);

//if (debug) cout << "\n=== R|=B ===";

//R |= B; DebOut(R); Used(R);

//int e = rand(R.Power());

//if (debug) cout << "\n=== R.Change (H, " << e << ") ===";

//H.Show(); Used(H);

//R.Change(H, e); DebOut(R); Used(R);

//int q\_sub(rand(MaxMul) + 1);

//PrepareAnd(C, R, q\_sub);

//if (debug) R.Show(), C.Show(); middle\_power += q\_sub; Used(C);

//if (debug) cout << "\n=== R-=C ===(" << q\_sub << ") ";

//R -= C; DebOut(R); Used(R);

//int a = rand(R.Power()), b = rand(R.Power());

//if (debug) cout << "\n=== R.Erase (" << a << "," << b << ")===";

//if (a > b) cout << "(skipped!)";

//R.Erase(a, b); DebOut(R); Used(R);

//if (debug) cout << "\n=== R.Concat(D) ===";

//D.Show(); Used(D);

//R.Concat(D); DebOut(R); Used(R);

//if (debug) cout << "\n=== R.Merge(E) ===";

//E.Show(); Used(E);

//R.Merge(E); DebOut(R); Used(R);

//if (debug) cout << "\n=== R.Excl(F) ===";

//F.PrepareExcl(R);

//if (debug && !F.Power()) cout << "(skipped)!";

//F.Show(); Used(F);

//R.Excl(F); DebOut(R); Used(R);

//int d = rand(R.Power());

//if (debug) cout << "\n=== R.Subst (G, " << d << ") ===";

//G.Show(); Used(G);

//R.Subst(G, d); DebOut(R); Used(R);

//int c = rand(MaxMul);

//if (debug) cout << "\n=== R.Mul(" << c << ")===";

//if (c < 2) cout << "(skipped!)";

//R.Mul(c); DebOut(R); Used(R);

//auto t2 = std::chrono::high\_resolution\_clock::now();

//auto dt = duration\_cast<duration<double>>(t2 - t1);

//middle\_power /= set\_count;

//// fout << p << ' ' << dt.count() << endl; //Выдача в файл

//cout << "\n=== Ready === \n(Power=" << p << " Set\_count = " << set\_count << " Middle\_power = " <<

// middle\_power << " DT=" << (dt.count()) << ")\n";

cin.get();

return 0;

}