Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Лабораторная работа №13

Стандартные обобщенные алгоритмы библиотеки STL.

Выполнил

Студент группы РИС-22-2б

Вековшинин Д. А.

Проверила

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

Задача 1.

1. Создать последовательный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля

(использовать алгоритм sort()). 6. Найти в контейнере заданный элемент (использовать алгоритмы find(),

find\_if(), count(), count\_if()). 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) . 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки

STL.

Задача 2.

1. Создать адаптер контейнера.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля

(использовать алгоритм sort()).

6. Найти в контейнере элемент с заданным ключевым полем (использовать

алгоритмы find(), find\_if(), count(), count\_if()). 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) . 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 3

1. Создать ассоциативный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля

(использовать алгоритм sort()).

6. Найти в контейнере элемент с заданным ключевым полем (использовать

алгоритмы find(), find\_if(), count(), count\_if()). 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) . 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки

STL.

Вариант 10:

Задача 1

1. Контейнер - вектор

2. Тип элементов Money (см. лабораторную работу №3).

Задача 2

Адаптер контейнера – очередь с приоритетами.

Задача 3

Ассоциативный контейнер – множество с дубликатами

Задание 3

Найти минимальный элемент и добавить его на заданную позицию контейнера

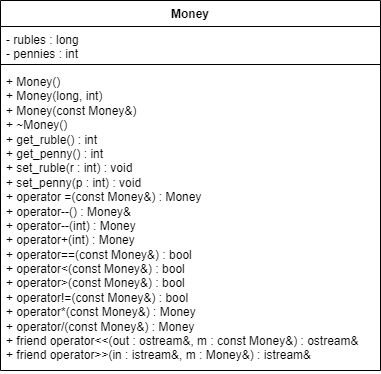
Задание 4

Найти элементы большие среднего арифметического и удалить их из контейнера

Задание 5

Каждый элемент домножить на максимальный элемент контейнера

UML-диаграмма



Код программы

#include "Money.h"

#include "vector"

#include <iostream>

#include <algorithm>

using namespace std;

typedef vector<Money> TVector;

//формирование вектора

TVector make\_vector(int n)

{

Money a;

TVector v;

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

{

cin >> a;

v.push\_back(a);

}

return v;

}

//печать вектора

void print\_vector(TVector v)

{

for (int i = 0; i < v.size(); i++)

cout << v[i] << endl;

cout << endl;

}

Money srednee(TVector v)

{

Money s = {0, 0};

for (int i = 0; i < v.size(); i++)

s = s + v[i];

int n = v.size();

return s / n;

}

typedef vector<Money> TVector;

Money s;

struct Greater\_s //больше, чем s

{

bool operator()(Money m)

{

if (m > s) return true; else return false;

}

};

void multi(Money& m)

{

m = m \* s;

}

void main()

{

int n, pos;

cout << "N?";

cin >> n;

TVector v;

v = make\_vector(n);

print\_vector(v);

TVector::iterator i;

//поставили итератор i на минимальный элемент

i = min\_element(v.begin(), v.end());

Money min = \*(i);

cout << "min: " << min << endl;

cout << "pos?"; cin >> pos;

v[pos] = min;

cout << "ZAMENA" << endl;

print\_vector(v);

s = srednee(v);

cout << "srednee: " << s << endl;

Money m = { 0, 0 };

replace\_if(v.begin(), v.end(), Greater\_s(), m);

cout << "UDALENIE" << endl;

print\_vector(v);

cout << "UMNOJENIE" << endl;

i = max\_element(v.begin(), v.end());

s = \*i;

cout << "max: " << s << endl;

//для каждого элемента вектора вызывается функция del

for\_each(v.begin(), v.end(), multi);

print\_vector(v);

}

Результаты работы программы

