Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

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

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №13

Тема: «Стандартные обобщенные алгоритмы библиотеки STL»

Выполнили

Студент группы РИС-22-1б

Анистратов А. Д.

Проверил доц. 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 15:

Задача 1

1. Контейнер - список

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

Задача 2 Адаптер контейнера – очередь с приоритетами.

Задача 3 Ассоциативный контейнер - словарь

Задание 3

Найти среднее арифметическое и добавить его в конец контейнера

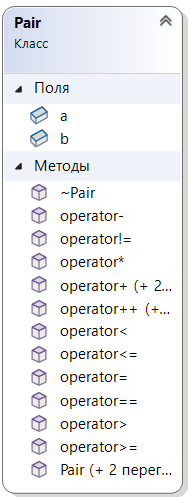
Задание 4

Найти элементы ключами из заданного диапазона и удалить их из контейнера

Задание 5

К каждому элементу добавить сумму минимального и максимального элементов контейнера

UML – диаграмма



Код программы

**Pair.h**

#pragma once

#include <iostream>

using namespace std;

class Pair

{

public:

int a;

double b;

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

Pair(int a, double b);

Pair();

Pair(const Pair& p);

~Pair();

Pair& operator=(const Pair& p);

Pair operator-(const Pair& p);

Pair& operator+(int a);

Pair& operator+(double b);

Pair& operator++();

Pair operator++(int);

Pair operator+(const Pair& p);

bool operator!=(const Pair& p) { return !((this->a == p.a) && (this->b == p.b)); }

bool operator==(const Pair& p) { return ((this->a == p.a) && (this->b == p.b)); }

bool operator>(const Pair& p) { return (this->a > p.a) || (this->b > p.b); }

bool operator>=(const Pair& p) { return (this->a >= p.a) || (this->b >= p.b); }

bool operator<=(const Pair& p) { return (this->a <= p.a) || (this->b <= p.b); }

bool operator<(const Pair& p) { return (this->a < p.a) || (this->b < p.b); }

Pair operator \* (Pair& pair)

{

Pair new\_pair;

new\_pair.a = this->a \* pair.a;

new\_pair.b = this->b \* pair.b;

return new\_pair;

}

};

**Pair.cpp**

#include "Pair.h"

Pair::Pair(int a, double b)

{

this->a = a;

this->b = b;

}

istream& operator>>(istream& in, Pair& p)

{

cout << "(int)a : ";

in >> p.a;

cout << "(double)b :";

in >> p.b;

return in;

}

ostream& operator<<(ostream& out, const Pair& p)

{

return (out << p.a << " : " << p.b);

}

Pair::Pair()

{

a = 0;

b = 0;

}

Pair::~Pair()

{

}

Pair::Pair(const Pair& p)

{

a = p.a;

b = p.b;

}

Pair& Pair::operator=(const Pair& p)

{

if (&p == this) return \*this;

a = p.a;

b = p.b;

return \*this;

}

Pair Pair::operator-(const Pair& p)

{

Pair res(a - p.a, b - p.b);

return res;

}

Pair& Pair::operator+(int a)

{

this->a += a;

return \*this;

}

Pair& Pair::operator+(double b)

{

this->b += b;

return \*this;

}

Pair& Pair::operator++()

{

++a; ++b;

return \*this;

}

Pair Pair::operator ++(int)

{

Pair temp = \*this;

this->a++; this->b++;

return temp;

}

Pair Pair::operator+(const Pair& p)

{

this->a += p.a;

this->b += p.b;

return \*this;

}

**main.cpp**

#include "Pair.h"

#include <iostream>

#include <list>

#include <queue>

#include <numeric>

#include <algorithm>

#include <map>

using namespace std;

void ex\_1()

{

cout << "Введите размер листа: " << endl;

int n;

cin >> n;

list<Pair> lst;

Pair p;

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

{

cin >> p;

lst.push\_back(p);

}

//1

Pair sr = accumulate(lst.begin(), lst.end(), Pair(0, 0));

sr.a = sr.a / lst.size();

sr.b = sr.b / lst.size();

lst.push\_back(sr);

cout << "Среднее:\n";

for\_each(lst.begin(), lst.end(), [](Pair p) {cout << p << endl; });

//2

cout << "Введите удаляемое: " << endl;

cin >> p;

auto it = remove(lst.begin(), lst.end(), p);

lst.erase(it, lst.end());

cout << "Удаление заданного: " << endl;

for\_each(lst.begin(), lst.end(), [](Pair p) {cout << p << endl; });

//3

cout << "Добавление макс и мин: " << endl;

Pair max = \*max\_element(lst.begin(), lst.end());

Pair min = \*min\_element(lst.begin(), lst.end());

for (Pair& c : lst)

{

c = c + max + min;

cout << c << endl;

}

}

bool operator<(const Pair& pr1, const Pair& pr2)

{

return ((pr1.a < pr2.a) && (pr1.b < pr2.b));

}

void ex\_2()

{

priority\_queue <Pair> lst;

priority\_queue <Pair> tmp;

lst.push(Pair(1, 1.5));

lst.push(Pair(-2, 2.7));

lst.push(Pair(3, -3.2));

cout << "Изначальный список:\n";

//1

Pair sr(0, 0);

while (!lst.empty())

{

cout << lst.top() << endl;

sr = sr + lst.top();

tmp.push(lst.top());

lst.pop();

}

sr.a /= tmp.size();

sr.b /= tmp.size();

cout << "Среднее ариф., добавленное в конец:\n";

while (!tmp.empty())

{

cout << tmp.top() << endl;

lst.push(tmp.top());

tmp.pop();

}

lst.push(sr);

cout << sr << endl;

//2

int k = 1;

int i, j;

Pair res;

cout << "Диапозон: "; cin >> i >> j;

cout << "Ключ: "; cin >> res;

while (!lst.empty())

{

if ((k < i) || (k > j) || (res != lst.top()))

{

tmp.push(lst.top());

}

lst.pop();

k++;

}

cout << "Удаленные элементы из заданного диапозона:\n";

Pair max = tmp.top();

Pair min = tmp.top();

while (!tmp.empty())

{

if (max < tmp.top()) max = tmp.top();

if (tmp.top() < min) min = tmp.top();

cout << tmp.top() << endl;

lst.push(tmp.top());

tmp.pop();

}

//3

cout << "Добавить макс. и мин. к каждому элементу:\n";

while (!lst.empty())

{

tmp.push(max + min + lst.top());

lst.pop();

}

while (!tmp.empty())

{

lst.push(tmp.top());

cout << tmp.top() << endl;

tmp.pop();

}

}

void ex\_3()

{

map <int, int> myMap = { {1, 2}, {2, 4}, {3, 6} };

cout << "Добавляем среднее ариф среди значений в конец:\n";

int sum = 0;

for (auto& p : myMap) {

sum += p.second;

}

int avg = sum / myMap.size();

myMap[0] = avg;

for (auto& p : myMap) {

cout << p.first << ": " << p.second << endl;

}

cout << "Удаляем ключ 2:\n";

myMap.erase(2);

cout << "Добавляем макс и мин значения к значениям ключей:\n";

auto maxIt = std::max\_element(myMap.begin(), myMap.end(),

[](const auto& p1, const auto& p2) { return p1.second < p2.second; });

auto minIt = std::min\_element(myMap.begin(), myMap.end(),

[](const auto& p1, const auto& p2) { return p1.second < p2.second; });

int maxVal = maxIt->second;

int minVal = minIt->second;

for (auto& p : myMap) {

p.second += maxVal + minVal;

cout << p.first << " : " << p.second << endl;

}

}

int main()

{

setlocale(0, "rus");

// ex\_1();

// ex\_2();

ex\_3();

return 0;

}