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

Студенты группы РИС-22-2б

Третьяков Н. А

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()).

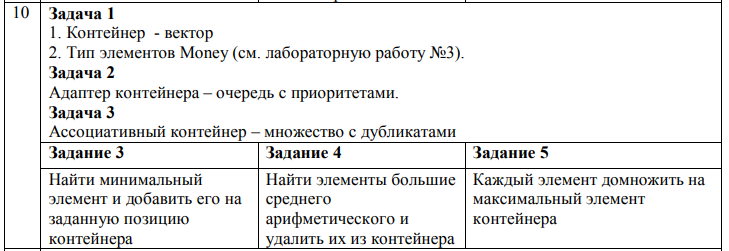
# 4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

# 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.



Код программы

Задача 1

Money.h

#pragma once

#include <iostream>

#include <vector>

using namespace std;

class Money

{

long rubles;

int kopecks;

public:

Money();

Money(long r, int k);

Money(const Money&);

long GetMoneyR();

int GetMoneyK();

void SetMoney(long r, int k);

Money& operator \*(const Money&);

Money& operator = (const Money&);

Money& operator +(const Money&);

Money& operator /(const int);

Money& operator --();

Money& operator --(int);

Money srednee(vector<Money>);

friend std::ostream& operator << (std::ostream& out, const Money&);

friend std::istream& operator >> (std::istream& in, Money&);

bool operator == (const Money&);

bool operator != (const Money&);

bool operator < (const Money&);

~Money() {};

};

Money.cpp

#include "Money.h"

#include <iostream>

#include <vector>

using namespace std;

Money::Money()

{

rubles = 0;

kopecks = 0;

}

Money::Money(long r, int k)

{

rubles = r;

if (k < 100)

{

kopecks = k;

}

else

{

rubles = rubles + k / 100;

kopecks = k % 100;

}

}

Money::Money(const Money& m)

{

rubles = m.rubles;

kopecks = m.kopecks;

}

long Money::GetMoneyR()

{

return rubles;

}

int Money::GetMoneyK()

{

return kopecks;

}

void Money::SetMoney(long r, int k)

{

rubles = r;

kopecks = k;

}

Money& Money::operator\*(const Money& m)

{

int s = (kopecks + rubles \* 100) \* (m.kopecks + m.rubles \* 100);

rubles = s / 100;

kopecks = s % 100;

return \*this;

}

Money& Money::operator/(const int n)

{

int S = rubles \* 100 + kopecks;

S = S / n;

rubles = S / 100;

kopecks = S % 100;

return \*this;

}

Money& Money::operator+(const Money& m)

{

if (kopecks + m.kopecks > 100)

{

rubles = rubles + m.rubles + (kopecks + m.kopecks) / 100;

kopecks = (kopecks + m.kopecks) % 100;

}

else

{

rubles += m.rubles;

kopecks += m.kopecks;

}

return \*this;

}

Money& Money::operator = (const Money& m)

{

rubles = m.rubles;

kopecks = m.kopecks;

return \*this;

}

Money& Money::operator -- ()

{

kopecks = kopecks - 1;

return \*this;

}

Money& Money::operator--(int)

{

kopecks = kopecks - 1;

return \*this;

}

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

{

return(out << p.rubles << ", " << p.kopecks);

}

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

{

in >> p.rubles;

in >> p.kopecks;

return in;

}

bool Money::operator!=(const Money& m)

{

if (this->rubles != m.rubles)

{

return 1;

}

else

{

if (this->kopecks != m.kopecks)

{

return 1;

}

else

{

return 0;

}

}

}

bool Money::operator==(const Money& m)

{

if (this->rubles == m.rubles)

{

if (this->kopecks == m.kopecks)

{

return 1;

}

else

{

return 0;

}

}

else

{

return 0;

}

}

bool Money:: operator <(const Money& m)

{

if (this->rubles < m.rubles)

{

return 1;

}

else

{

if (this->rubles == m.rubles)

{

if (this->kopecks < m.kopecks)

return 1;

else

return 0;

}

else

return 0;

}

}

Money Money::srednee(vector<Money> M)

{

Money m;

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

m = m + M[i];

int n = M.size();

return m / n;

}

Main.cpp

#include <iostream>

#include <vector>

#include "Money.h"

#include <algorithm>

using namespace std;

typedef vector <Money> Vec;

Money M;

Vec MVector(int n)

{

Vec v;

Money m;

int k;

long r;

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

{

k = rand() % 100;

r = rand() % 100;

m.SetMoney(r, k);

v.push\_back(m);

}

return v;

}

void Print(vector<Money> m)

{

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

cout << m[i] << endl;

cout << "------------------------------------" << endl;

}

struct Greater\_s

{

bool operator() (Money m)

{

if (M < m)

return false;

else

return true;

}

};

struct Comp\_less

{

bool operator()(Money m, Money m1)

{

if (m1 < m)

return true;

else

return false;

}

};

struct Equal\_s

{

bool operator() (Money m)

{

return m == M;

}

};

void del(Money& m)

{

m = m \* M;

}

int main()

{

Vec v1, v;

int n;

cout << "Enter the size of the container" << endl;

cin >> n;

v1 = MVector(n);

Vec::iterator i;

i = min\_element(v1.begin(), v1.end());

Money mn = \*(i);

cout << "Sort in ascending order: " << endl;

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

Print(v1);

i = max\_element(v1.begin(), v1.end());

Money mx = \*i;

M = mn;

replace\_if(v1.begin(), v1.end(), Greater\_s(), mx);

Print(v1);

cout << "Search" << endl;

cin >> M;

i = find\_if(v1.begin(), v1.end(), Equal\_s()); // оставить

if (i != v1.end())

cout << \*(i) << endl;

else

cout << "Not such element" << endl;

cout << "Dell" << endl;

M.srednee(v1);

i = remove\_if(v1.begin(), v1.end(), Equal\_s());// не работает

Print(v1);

for (int k = 0; k < v1.size();k++)

{

if (!(M < v1[k]))

{

v.push\_back(v1[k]);

}

}

v1 = v;

Print(v1);

cout << "Domnozhenie" << endl;

i = max\_element(v1.begin(), v1.end());

M = \*(i);

for\_each(v1.begin(), v1.end(), del); // изменить название на умножение

Print(v1);

}

Задача 2

Main.cpp

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

typedef priority\_queue<int> Pri;

Pri MQueue(int n)

{

Pri p;

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

p.push(rand() % 100);

return p;

}

void Print(Pri p)

{

while (!p.empty())

{

cout << p.top() << " ";

p.pop();

}

cout << endl;

}

Pri AddMin(Pri p)

{

Pri p1 = p;

int mn;

while (!p1.empty())

{

mn = p1.top();

p1.pop();

}

cout << "Enter the number to put the item on" << endl;

int n;

cin >> n;

int k = 0;

int temp;

while (k != n)

{

temp = p.top();

p1.push(temp);

p.pop();

k++;

}

p1.push(mn);

while (!p.empty())

{

temp = p.top();

p1.push(temp);

p.pop();

}

return p1;

}

Pri Del(Pri p)

{

int s = 0;

Pri p1 = p;

while (!p1.empty())

{

s += p1.top();

p1.pop();

}

s = s / p.size();

while (!p.empty())

{

if (p.top() < s)

{

p1.push(p.top());

}

p.pop();

}

return p1;

}

Pri M(Pri p)

{

int mx = p.top();

Pri p1;

while (!p.empty())

{

p1.push(p.top() \* mx);

p.pop();

}

return p1;

}

int main()

{

Pri p;

cout << "Enter the size of the container" << endl;

int n;

cin >> n;

p = MQueue(n);

Print(p);

p = AddMin(p);

Print(p);

p = Del(p);

Print(p);

p = M(p);

Print(p);

}

Задача 3

Main.cpp

#include<iostream>

#include <map>

#include<algorithm>

using namespace std;

int N;

typedef multimap<int, float> Mult;

typedef multimap<int, float>::iterator it;

Mult Mmap(int n)

{

Mult m;

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

{

m.insert(i, rand() % 100);

}

return m;

}

void Print(Mult m)

{

it iter;

for (iter = m.begin();iter != m.end(); iter++)

{

cout << iter->first + 1 << ") " << iter->second << endl;

}

}

struct Mn

{

bool operator ()(int i)

{

return i == N;

}

};

Mult MapMn(Mult m)

{

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

float mn = i->second;

cout << "" << endl;

int n;

cin >> n;

N = n;

replace\_if(m.begin(), m.end(), Mn(), mn);

return m;

}

Mult MapS(Mult m)

{

it i;

int n, n1 = 0;

float s;

for (i = m.begin(); i != m.end(); i++)

{

s += i->second;

n = i->first + 1;

}

s = s / n;

Mult m1;

for (i = m.begin(); i != m.end(); i++)

{

if (s > i->second)

{

m1.insert(make\_pair(n1, i->second));

n1++;

}

}

return m1;

}

Mult MapMax(Mult m)

{

it i;

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

float mx = i->second;

for (i = m.begin(); i != m.end(); i++)

{

i->second = i->second \* mx;

}

return m;

}

int main()

{

int n;

cout << "" << endl;

cin >> n;

Mult m;

m = Mmap(n);

Print(m);

m = MapS(m);

Print(m);

m = MapMax(m);

Print(m);

}