Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Стандартные обобщенные алгоритмы библиотеки STL.»

Выполнил работу

Студент группы РИС-23-3Б

Епин Т. Е.

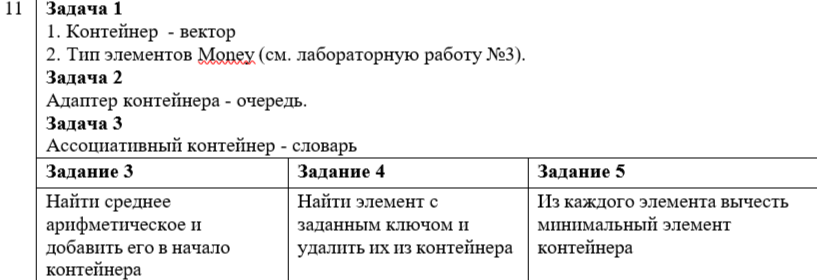
Проверил

Доцент кафедры ИТАС

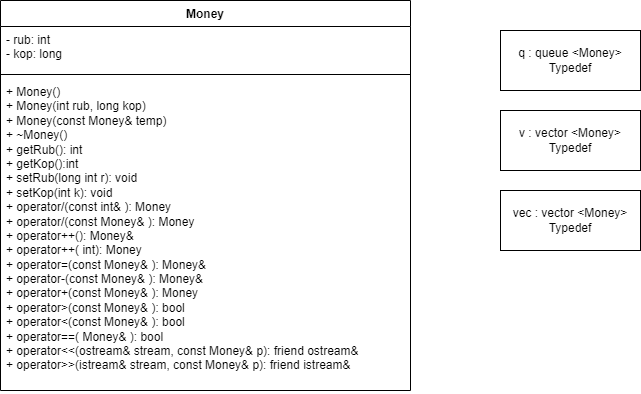
Полякова О.А.

Г. Пермь-2024

**Постановка задачи:**



**Uml диаграмма:**

****

**Код программы:**

**Money.h**

#pragma once

#include<iostream>

using namespace std;

class Money {

long int rub;

int kop;

public:

Money() { rub = 0; kop = 0; };

Money(long int r, int k) { rub = r; kop = k; };

Money(const Money& temp) { rub = temp.rub; kop = temp.kop; };

~Money() {};

int getRub() { return rub; };

int getKop() { return kop; };

void setRub(long int r) { rub = r; }

void setKop(int k) { kop = k; }

Money& operator =(const Money&);

Money& operator ++();

Money operator / (const Money&);

Money operator / (const int&);

Money operator ++(int);

Money& operator -(const Money&);

Money operator +(const Money&);

bool operator > (const Money&);

bool operator < (const Money&);

bool operator ==(Money&);

friend istream& operator >>(istream& in, Money& temp);

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

};

**Money.cpp**

#include<iostream>

#include"money.h"

using namespace std;

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

if (rub == m.rub && kop == m.kop) return 1;

else return 0;

}

Money& Money::operator - (const Money& a) {

Money\* tmp = new Money;

tmp->setRub(this->rub); tmp->setKop(this->kop);

tmp->rub -= a.rub;

tmp->kop -= a.kop;

return \*tmp;

}

Money Money::operator / (const Money& m) {

int tmp1 = rub \* 100 + kop;

int tmp2 = m.rub \* 100 + m.kop;

Money p;

p.rub = (tmp1 + tmp2) / 100;

p.kop = (tmp1 + tmp2) % 100;

return p;

}

Money Money::operator / (const int& i) {

Money p;

p.rub = rub / i;

p.kop = kop / i;

return p;

}

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

if (rub < m.rub) return true;

if (rub == m.rub && kop < m.kop) return true;

return false;

}

bool Money::operator >(const Money& m) {

if (rub > m.rub) return true;

if (rub == m.rub && kop > m.kop) return true;

return false;

}

Money& Money::operator =(const Money& temp) {

if (&temp == this) {

return \*this;

}

rub = temp.rub;

kop = temp.kop;

return \*this;

}

Money& Money::operator ++() {

int temp = rub \* 100 + kop;

temp++;

kop = temp / 100;

kop = temp % 100;

return \*this;

}

Money Money::operator ++(int) {

int temp = rub \* 100 + kop;

temp++;

Money ex1(rub, kop);

kop = temp / 100;

kop = temp % 100;

return ex1;

}

Money Money::operator +(const Money& temp) {

int ex1 = rub \* 100 + kop;

int ex2 = temp.rub \* 100 + temp.kop;

Money temp2;

temp2.rub = (ex1 + ex2) / 100;

temp2.kop = (ex1 + ex2) % 60;

return temp2;

}

istream& operator >>(istream& in, Money& temp) {

cout << "rubles: ";

in >> temp.rub;

cout << "kopecks: ";

in >> temp.kop;

return in;

}

ostream& operator <<(ostream& out, const Money& temp) {

return (out << temp.rub << "," << temp.kop);

}

**Vector.h**

#pragma once

#include"money.h"

#include<iostream>

#include<vector>

using namespace std;

typedef vector<Money>vec;

vec makeVector(int size) {

vec v;

for (int i = 1; i <= size; i++) {

Money tmp(rand() % 100, rand() % 100);

v.push\_back(tmp);

}

return v;

}

void printVector(vec v) {

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

cout << v[i] << " ";

}

cout << endl;

}

int average(vec v) {

int sum = 0;

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

sum += (v[i].getRub() + v[i].getKop());

}

int n = v.size();

cout << "Average: " << sum / n << endl;

return sum / n;

}

void minElem(vec v) {

Money min = v[0];

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

if (min > v[i]) { min = v[i]; }

}

cout << "Min element = " << min << endl;

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

v[i] = v[i] - min;

}

printVector(v);

}

**Queue.h**

#pragma once

#include<queue>

#include<vector>

#include<cmath>

#include"money.h"

using namespace std;

typedef queue<Money> q;

typedef vector<Money> v;

q makeQueue(int size) {

q queue;

Money m;

for (int i = 0; i < size; i++) {

cin >> m;

queue.push(m);

}

return queue;

}

v copyQueue(q queue) {

v vector;

while (!queue.empty()) {

vector.push\_back(queue.front());

queue.pop();

}

return vector;

}

q copyVector(v vector) {

q queue;

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

queue.push(vector[i]);

}

return queue;

}

void printQueue(q queue) {

cout << "Queue: ";

while (!queue.empty()) {

cout << queue.front() << " ";

queue.pop();

}

cout << endl;

}

Money averageQ(q queue) {

v vec = copyQueue(queue);

int n = 1;

Money sum = queue.front();

queue.pop();

while (!queue.empty()) {

sum = sum + queue.front();

queue.pop();

n++;

}

queue = copyVector(vec);

cout << "Average (queue): " << sum / n << endl;

return sum / n;

}

void addQueue(q& queue, Money m, int index) {

v vec; Money mm; int i = 1;

while (!queue.empty()) {

mm = queue.front();

if (i == index) { vec.push\_back(m); }

vec.push\_back(mm);

queue.pop();

i++;

}

queue = copyVector(vec);

}

Money minElem(q queue) {

Money min = queue.front();

v vec = copyQueue(queue);

while (!queue.empty()) {

if (queue.front() < min) { min = queue.front(); }

queue.pop();

}

queue = copyVector(vec);

cout << "Min elem: " << min << endl;

v vec2; Money m;

while (!queue.empty()) {

m = queue.front();

vec2.push\_back(m - min);

queue.pop();

}

queue = copyVector(vec2);

return min;

}

**Main.cpp**

#include <iostream>

#include"vector.h"

#include"queue.h"

using namespace std;

int main()

{

system("chcp 1251");

srand(time(0));

try {

vector<Money> v;

int size;

cout << "Enter size (vector): "; cin >> size;

v = makeVector(size);

cout << "Vector (money): "; printVector(v);

Money a(average(v), 0);

v.insert(v.begin(), a);

cout << "Result: "; printVector(v);

int elem;

cout << "Element to delete: "; cin >> elem;

printVector(v);

cout << "Vector - min Element" << endl;

minElem(v);

cout << endl;

Money mm;

q que;

int sizeq;

cout << "Enter size (queue): "; cin >> sizeq;

que = makeQueue(sizeq);

printQueue(que);

averageQ(que);

int index;

cout << "Index: "; cin >> index;

addQueue(que, mm, index);

cout << "Add elem: "; printQueue(que);

cout << endl;

minElem(que);

cout << "Result: "; printQueue(que);

cout << endl;

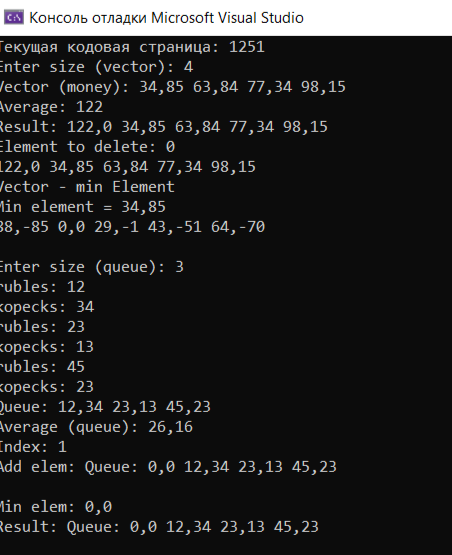
}

catch (int) { cout << "Error" << endl; }

return 0;

}

**Результат работы программы:**

****