**Пермский национальный исследовательский политехнический университет**

Кафедра “Информационные технологии и автоматизированные системы”

**Лабораторная работа №24**

По дисциплине «Основы алгоритмизации и программирования»

**Тема:**

Шаблоны классов С++.

Выполнила:

Студентка группы ИВТ-19-1б

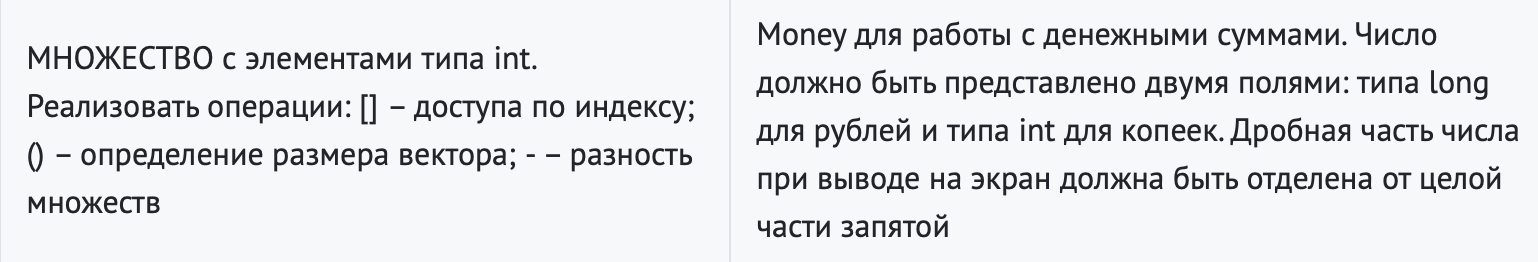
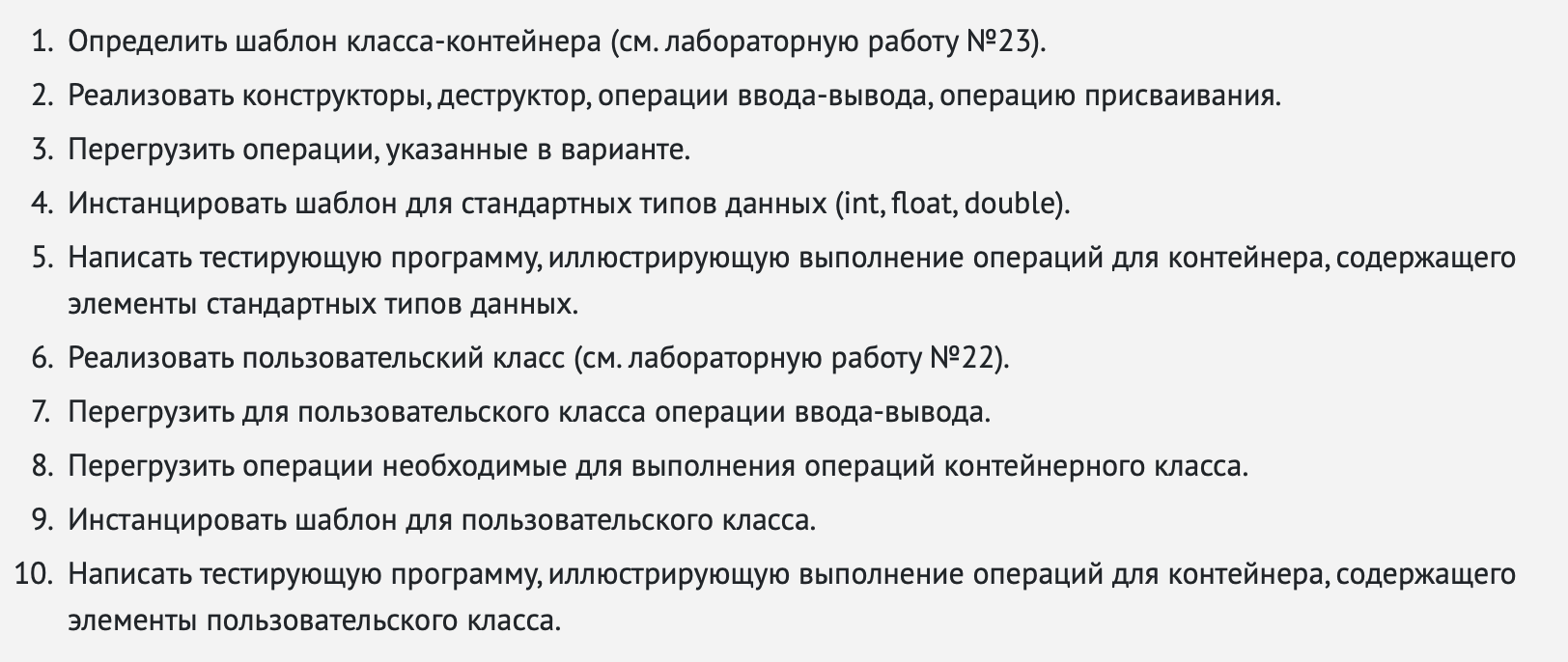
Еремеева Полина Алексеевна

Проверила:

доцент кафедры “ИТАС”

Полякова О.А.

Пермь, 2020

**Постановка задачи**

**Код**

**Main.cpp**

#include "Lots.hpp"

#include "Money.hpp"

#include <iostream>

**using** **namespace** std;

**int** main()

{

setlocale(LC\_ALL, "rus");

cout << "Работаем с классом множество!\n";

cout << "Создаем переменную А с 5 элементами:\n";

Lots<**int**> a(5,0);

cout << a << "\n";

cin >> a;

cout << a << "\n";

cout << endl;

cout << "Создаем переменную В с 10 элементами:\n";

Lots<**int**> b(10,45);

cout << b << "\n";

cout << endl;

cout << "Приравниваем В к А:\n";

b = a;

cout << b << "\n";

cout << endl;

cout << "Элемент переменной А с индексом 2 равен 100:\n";

a[2] = 100;

cout << endl;

cout << "Создаем переменную С с 10 элементами\n";

Lots<**int**> c(10,0);

cout << endl;

cout << "переменная С равена разности элементов B и A:\n";

c = b - a;

cout << c << "\n";

cout << "Вывод длины А= " << a() << endl;

cout << endl;

cout << "Работаем с классом Money!\n";

Money F;

cin >> F;

cout << F;

cout << endl;

Money L;

cin >> L;

cout << L;

cout << endl;

Money FL;

FL = F - L;

cout << FL;

cout << endl;

cout << "Выполним тестировани параметризированного класса Множество:\n";

cout << "Для пользовательского типа данных Money:\n";

cout << "Создаем переменную класса Money:\n";

Money m;

cin >> m;

cout << m;

cout << endl;

cout << "Создаем переменную класса Множества с пользовательским типо данных Money:\n";

Lots<Money>Al(5, m);

cin >> Al;

cout << Al << endl;

cout << endl;

cout << "Создаем вторую переменную класса Множества с пользовательским типо данных Money:\n";

Lots<Money>Bl(10, m);

cout << Bl << endl;

cout << endl;

cout << "Выводим 3 элемент первой переменной:\n";

cout << Al[2] << endl;

cout << "Размер первой переменной = " << Al() << endl;

Lots<Money>Cl(10, m);

cout << endl;

cout << "Вычитаем из второй переменной элементы первой:\n";

Cl = Bl - Al;

cout << Cl;

cout << endl;

cout << "Приравниваем вторую переменную первой и выводи её:\n";

Bl = Al;

cout << Bl << endl;

}

Money.cpp

#include "Money.hpp"

Money::Money(**void**)

{

rub = cop = 0;

}

Money::Money(**long** r, **int** c)

{

cop = c;

rub = r;

}

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

{

rub = m.rub;

cop = m.cop;

}

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

{

rub = m.rub;

cop = m.cop;

**return** \***this**;

}

istream& **operator**>> (istream& in, Money& m)

{

cout << "rub?"; in >> m.rub;

cout << "cop?"; in >> m.cop;

**return** in;

}

ostream& **operator**<<(ostream& out, **const** Money& m)

{

out << m.rub << "," << m.cop;

**return** out;

}

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

{

**long** minus = (rub \* 100 + cop) - (m.rub \* 100 + m.cop);

Money temp(((minus - (minus % 100)) / 100), minus % 100);

**return** temp;

}

Money.hpp

#pragma once

#include <iostream>

**using** **namespace** std;

**class** Money

{

**public**:

Money(**void**);

Money(**long**, **int**);

Money(**const** Money&);

Money& **operator** = (**const** Money&);

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

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

Money **operator**-(**const** Money&m);

**public**:

**virtual** ~Money(**void**) {};

**private**:

**long** rub;

**int** cop;

};

Lots.hpp

#pragma once

#include <iostream>

#include <initializer\_list>

**using** **namespace** std;

**template** <**class** T> **class** Lots;

**template** <**class** T> ostream& **operator**<<(ostream&, **const** Lots<T>&);

**template** <**class** T> istream& **operator**>>(istream& in, Lots<T>& v);

**template** <**class** T>

**class** Lots

{

**public**:

Lots(**int** s, T k );

Lots(**const** Lots<T>& v);

~Lots();

Lots& **operator**=(**const** Lots<T>& v);

T& **operator**[](**int** index);

Lots **operator**-(**const** Lots<T>& v);

**int** **operator**()();

**friend** ostream& **operator** << <T> (ostream&, **const** Lots<T>&);

**friend** istream& **operator**>> <T>(istream& in, Lots<T>& v);

**private**:

**int** size;

T\* data;

};

**template** <**class** T>

Lots<T>::Lots(**const** Lots& v)

{

size = v.size;

data = **new** T[size];

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

{

data[i] = v.data[i];

}

}

**template** <**class** T>

Lots<T>::Lots(**int** s, T k)

{

size = s;

data = **new** T[size];

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

{

data[i] =k;

}

}

**template** <**class** T>

Lots<T>::~Lots()

{

**delete**[]data;

data = 0;

}

**template** <**class** T>

Lots<T>& Lots<T>::**operator**=(**const** Lots<T>& v)

{

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

size = v.size;

**if** (data != 0) **delete**[] data;

data = **new** T[size];

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

{

data[i] = v.data[i];

}

**return** \***this**;

}

**template** <**class** T>

T& Lots<T>::**operator**[] (**int** index)

{

**if** (index < size) **return** data[index];

**else** cout << "\nError! Index > size"; exit(0);

}

**template** <**class** T>

Lots<T> Lots<T>::**operator**-(**const** Lots<T>& v)

{

Lots<T> temp(size, data[0]);

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

{

temp.data[i] = v.data[i] - data[i];

}

**return** temp;

}

**template** <**class** T>

**int** Lots<T>::**operator**()()

{

**return** size;

}

**template** <**class** T>

ostream& **operator**<<(ostream& out, **const** Lots<T>& v)

{

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

{

out << v.data[i] << " ";

}

**return** out;

}

**template** <**class** T>

istream& **operator**>>(istream& in, Lots<T>& v)

{

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

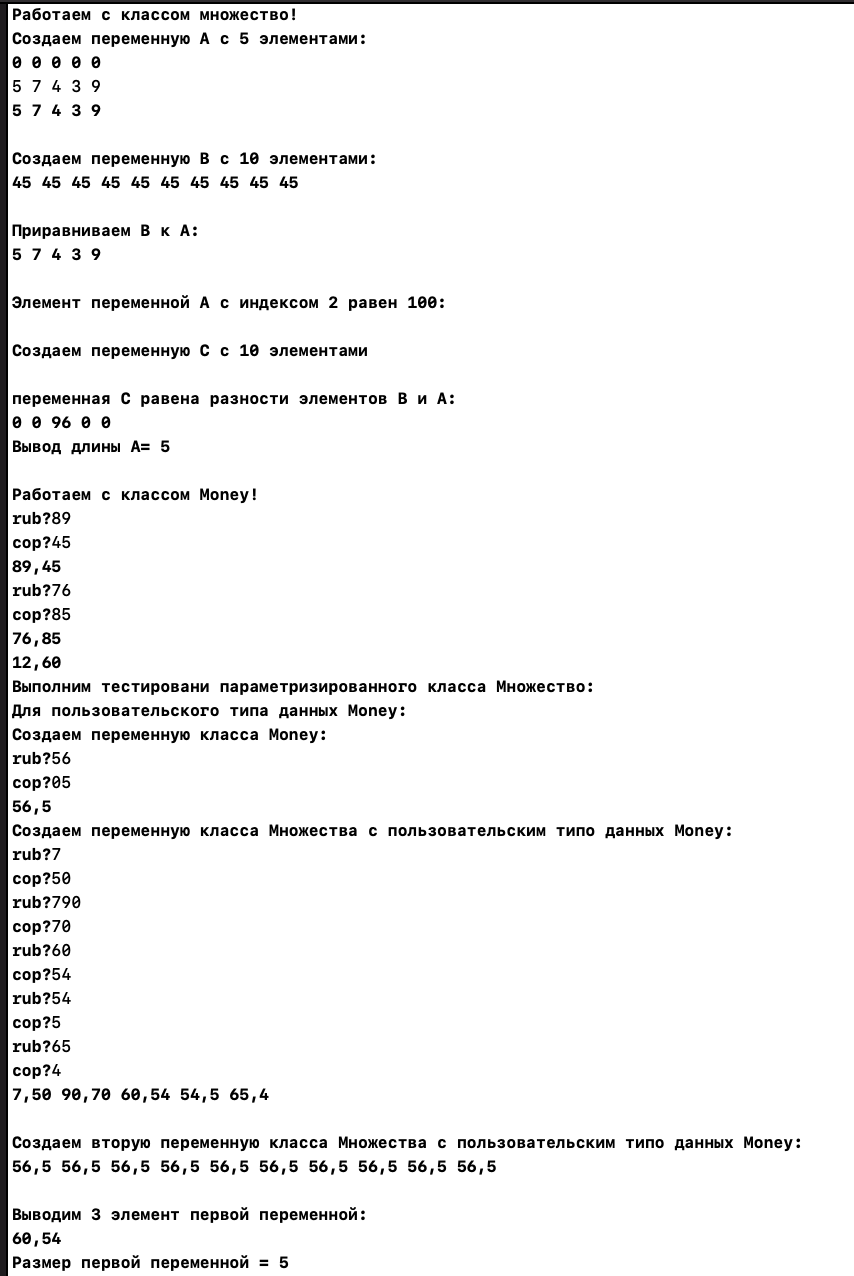
{

in >> v.data[i];

}

**return** in;

}

**Тестирование**

