Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.7**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Шаблоны классов”

Вариант 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О.Ю.

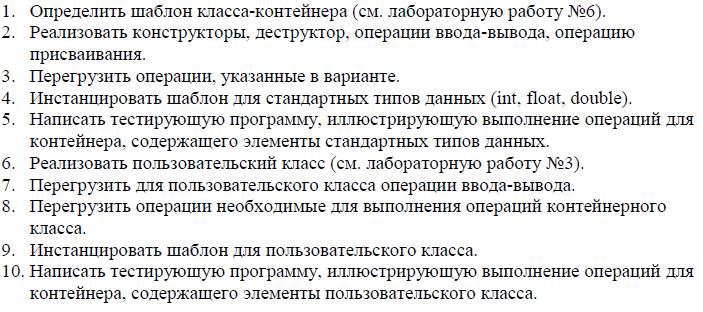
Проверила:

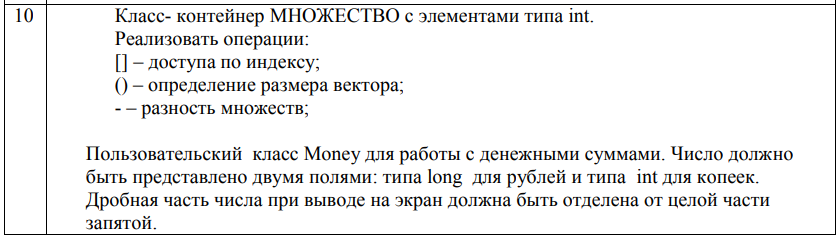
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Money для хранения рублей (rub), копеек (kop).

**1.2.** Организовать необходимые методы для ввода данных в поля rub и kop: гетторы, сетторы, конструкторы, деструктор.

**1.3.** Организовать перегрузку оператора >> дружественной классу Money.

**1.4.** Организовать перегрузку оператора << дружественной классу Money.

**1.5.** Организовать перегрузку оператора = дружественной классу Money.

**1.6.** Организовать перегрузку оператора != дружественной классу Money.

**1.7.** Организовать перегрузку оператора -- дружественной классу Money.

**1.8.** Организовать класс Vector с полями \_size для размерности массива и указатель data на первый элемент массива.

**1.9.** Организовать перегрузку оператора << дружественной классу Vector.

**1.10.** Организовать перегрузку оператора () дружественной классу Vector.

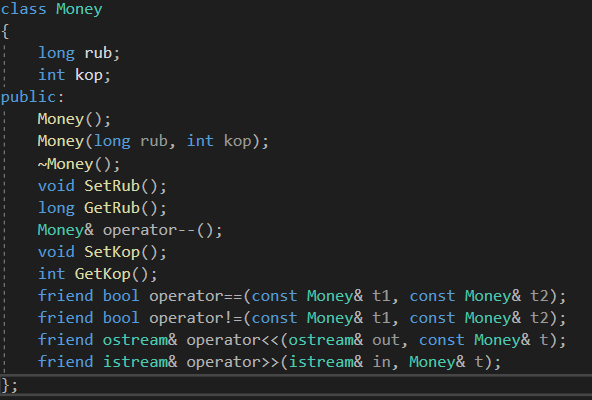
**1.11.** Организовать перегрузку оператора [] дружественной классу Vector.

**1.12.** Организовать перегрузку оператора = дружественной классу Vector.

**1.13.** Организовать перегрузку оператора - дружественной классу Vector.

**2.** В ходе работы были использованы следующие типы данных:

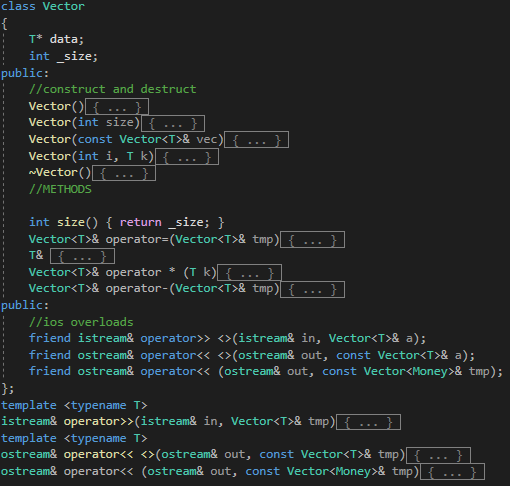
**2.1.** Организовать класс Money для хранения рублей (rub), копеек (kop). в заголовочном файле Money.h.



**2.2.** Переменные типа long, int: rub, kop, где rub – введённое количество рублей, kop – введённое количество копеек.



**2.3.** Организовать класс Vector с полями \_size для размерности массива и указатель data на первый элемент массива.



**2.4.** Объект класса Vector N – массив типа Money.

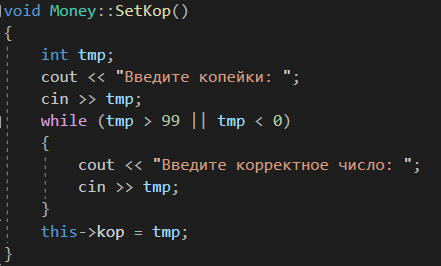
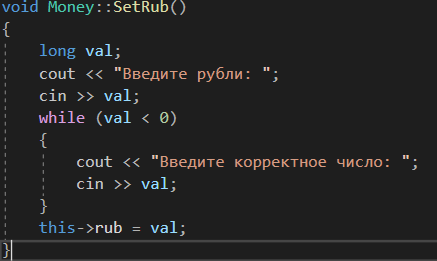


**2.5.** Объект m класса Money.

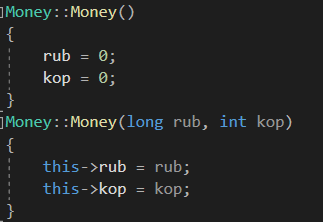
****

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через объекты класса Money через cетторы.

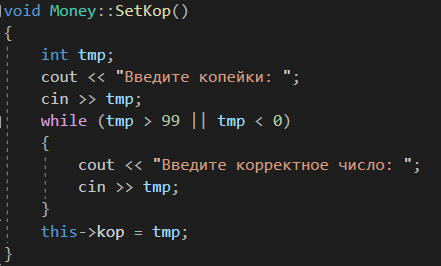
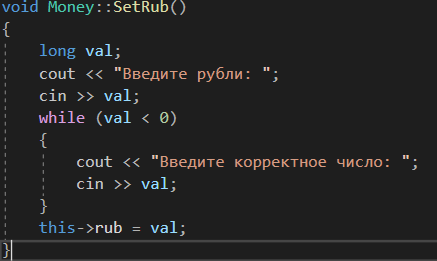


**3.2.** Данные также вводятся через объекты класса Money через конструктор с параметрами.

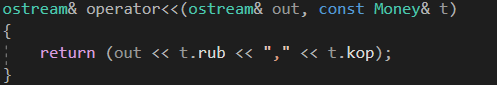


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода временных интервалов используются cетторы (методы, описанные в классе для ввода), в которых используется проверка на отрицательное число, а для копеек на число больше 100.

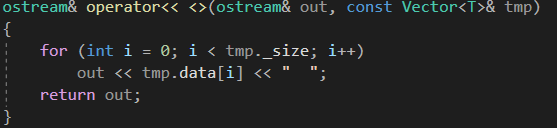


**4.2.** Вывод на консоль может быть осуществлён с помощью перегрузки оператора << для класса Money.

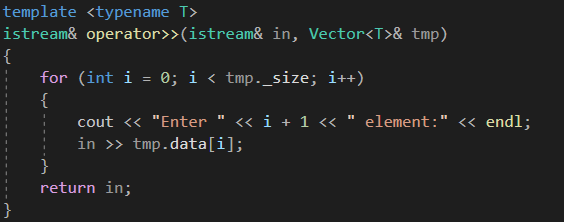
****

**4.3.** Вывод на консоль может быть осуществлён с помощью перегрузки оператора << для класса Vector.



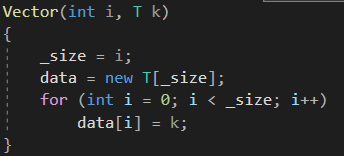


**4.4.** Для ввода элементов в массив класса Vector также может быть использована перегрузка оператора >>

****

**5.** Поставленные задачи будут решены следующими действиями:

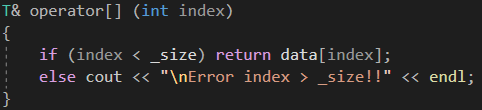
**5.1.** В конструкторе класса Vector в качестве параметров передаётся размерность массива и объект, который надо поместить в массив.



**5.2.** В перегрузке оператора () класса Vector производится вывод размерности объекта класса Vector.

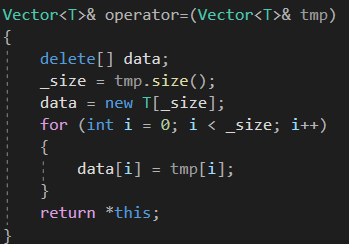


**5.3.** В перегрузке оператора [] класса Vector производится вызов элемента множества по индексу объекта класса Vector.

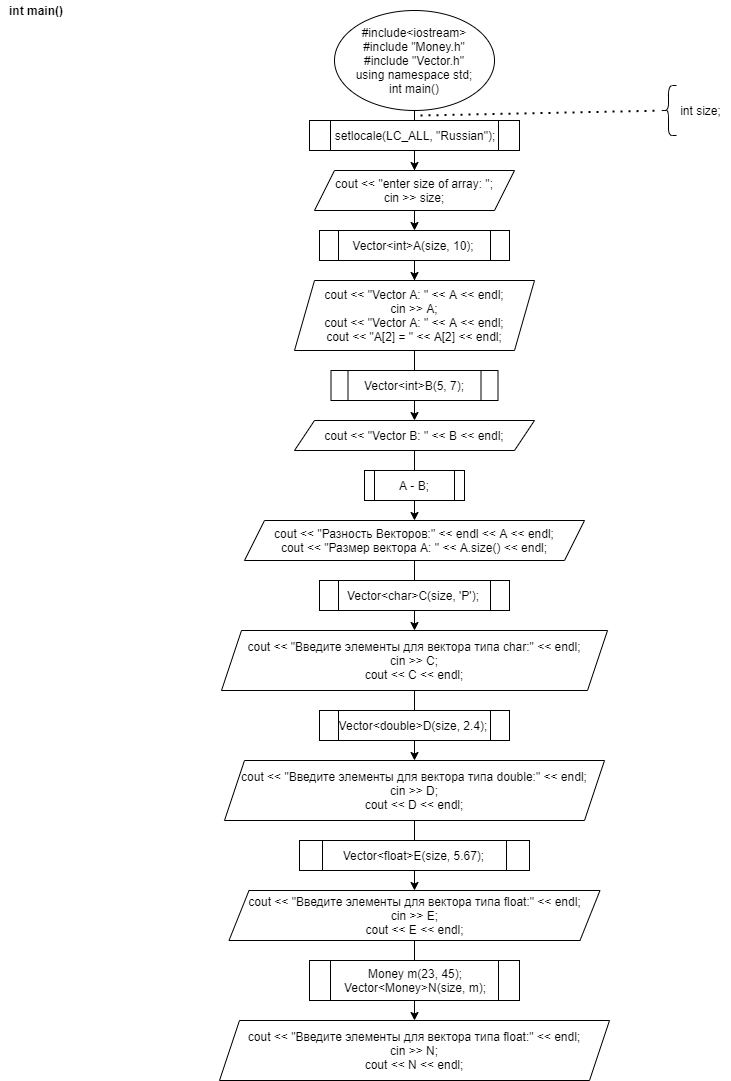


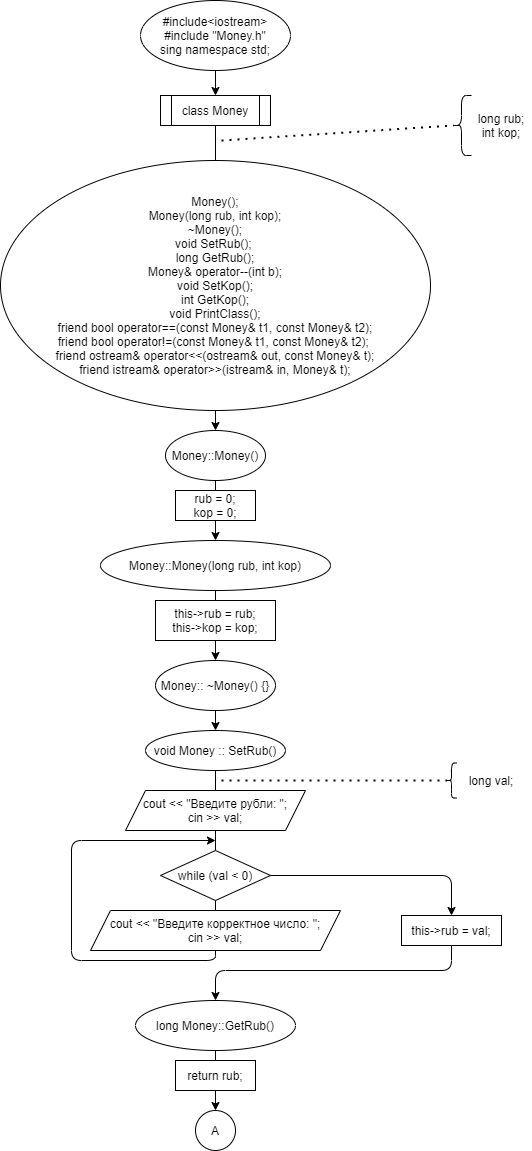
**5.4.** В перегрузке оператора = класса Vector производится вызов элемента множества по индексу объекта класса Vector.

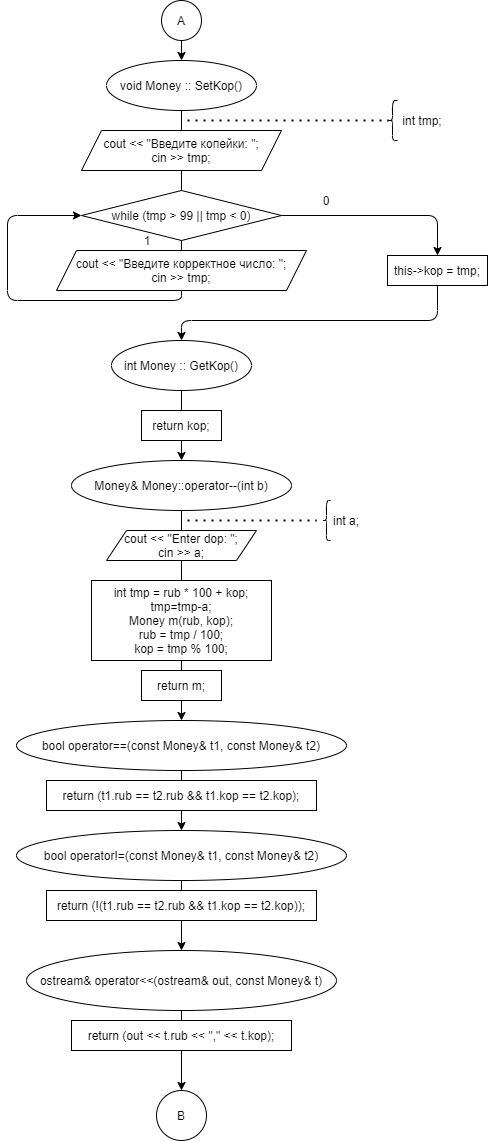
В поле \_size присваивается значение size нового элемента (tmp), создаётся новый массив с размерностью \_size и в цикле for заполняется новый массив из массива объекта tmp.

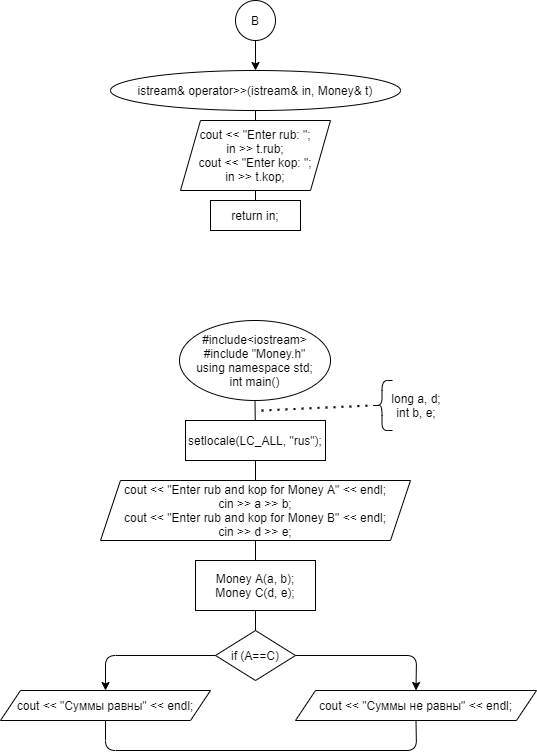


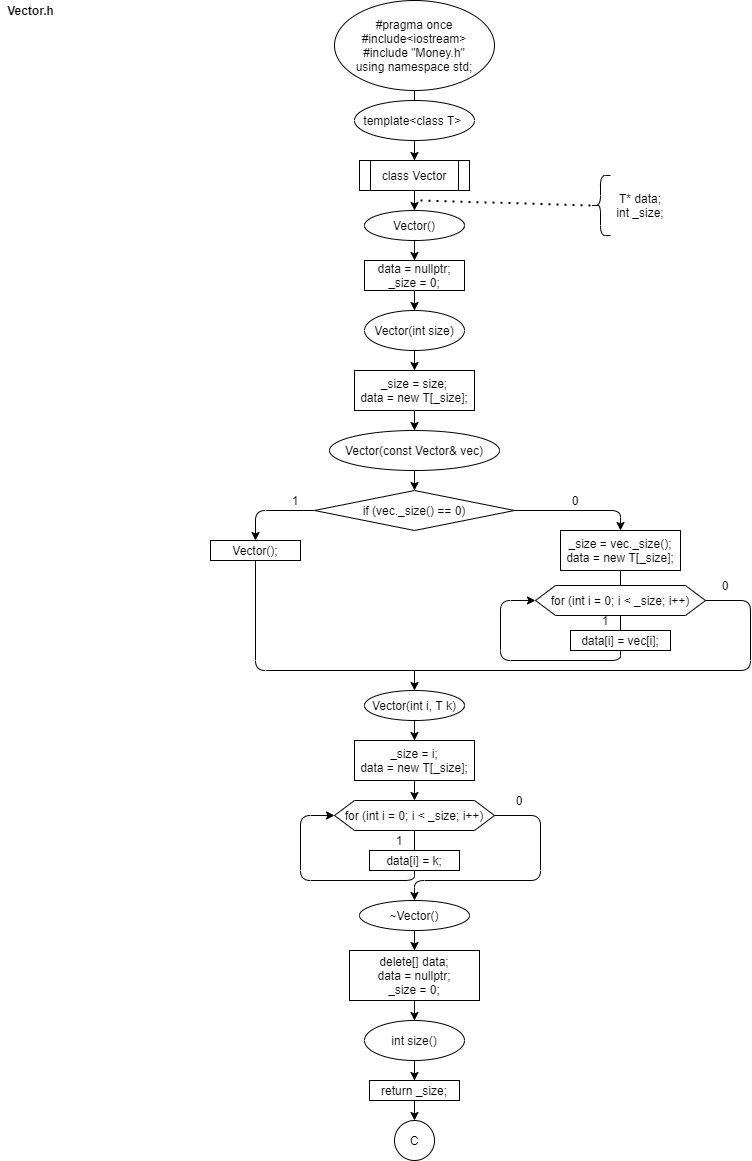
**Блок-схема**

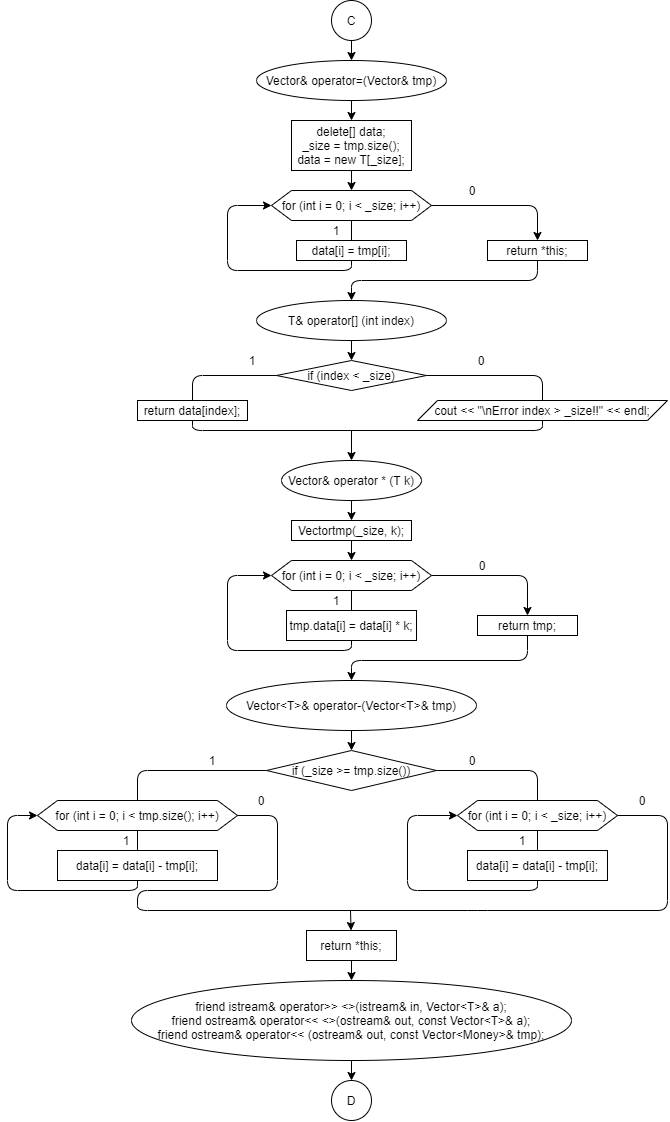
****

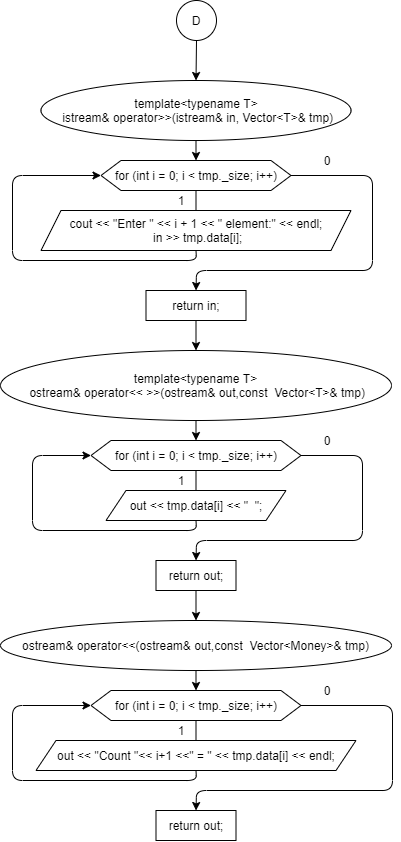












**Код**

#include <iostream>

#include "Money.h"

#include "Vector.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int size;

cout << "enter size of array: ";

cin >> size;

Vector<int>A(size, 10);

cout << "Vector A: " << A << endl;

cin >> A;

cout << "Vector A: " << A << endl;

cout << "A[2] = " << A[2] << endl;

Vector<int>B(5, 7);

cout << "Vector B: " << B << endl;

A - B;

cout << "Разность Векторов:" << endl << A << endl;

cout << "Размер вектора А: " << A.size() << endl;

Vector<char>C(size, 'P');

cout << "Введите элементы для вектора типа char:" << endl;

cin >> C;

cout << C << endl;

Vector<double>D(size, 2.4);

cout << "Введите элементы для вектора типа double:" << endl;

cin >> D;

cout << D << endl;

Vector<float>E(size, 5.67);

cout << "Введите элементы для вектора типа float:" << endl;

cin >> E;

cout << E << endl;

Money m(23, 45);

Vector<Money>N(size, m);

cout << "Введите элементы для вектора типа float:" << endl;

cin >> N;

cout << N << endl;

}

#pragma once

#include<iostream>

using namespace std;

class Money

{

long rub;

int kop;

public:

Money();

Money(long rub, int kop);

~Money();

void SetRub();

long GetRub();

Money& operator--();

void SetKop();

int GetKop();

friend bool operator==(const Money& t1, const Money& t2);

friend bool operator!=(const Money& t1, const Money& t2);

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

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

};

#include <iostream>

#include "Money.h"

using namespace std;

Money::Money()

{

rub = 0;

kop = 0;

}

Money::Money(long rub, int kop)

{

this->rub = rub;

this->kop = kop;

}

Money:: ~Money() {}

void Money::SetRub()

{

long val;

cout << "Введите рубли: ";

cin >> val;

while (val < 0)

{

cout << "Введите корректное число: ";

cin >> val;

}

this->rub = val;

}

long Money::GetRub() { return rub; }

void Money::SetKop()

{

int tmp;

cout << "Введите копейки: ";

cin >> tmp;

while (tmp > 99 || tmp < 0)

{

cout << "Введите корректное число: ";

cin >> tmp;

}

this->kop = tmp;

}

int Money::GetKop() { return kop; }

Money& Money::operator--()

{

int a;

cout << "Enter dop: ";

cin >> a;

int tmp = rub \* 100 + kop;

tmp = tmp - a;

Money m(rub, kop);

rub = tmp / 100;

kop = tmp % 100;

return m;

}

bool operator==(const Money& t1, const Money& t2)

{

return (t1.rub == t2.rub && t1.kop == t2.kop);

}

bool operator!=(const Money& t1, const Money& t2)

{

return (!(t1.rub == t2.rub && t1.kop == t2.kop));

}

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

{

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

}

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

{

cout << "Enter rub: ";

in >> t.rub;

cout << "Enter kop: ";

in >> t.kop;

return in;

}

#pragma once

#include<iostream>

#include "Money.h"

using namespace std;

template <class T>

class Vector

{

T\* data;

int \_size;

public:

//construct and destruct

Vector()

{

data = nullptr;

\_size = 0;

}

Vector(int size)

{

\_size = size;

data = new T[\_size];

}

Vector(const Vector<T>& vec)

{

if (vec.\_size() == 0)

Vector();

else

{

\_size = vec.\_size();

data = new T[\_size];

for (int i = 0; i < \_size; i++)

data[i] = vec[i];

}

}

Vector(int i, T k)

{

\_size = i;

data = new T[\_size];

for (int i = 0; i < \_size; i++)

data[i] = k;

}

~Vector()

{

delete[] data;

data = nullptr;

\_size = 0;

}

//METHODS

int size() { return \_size; }

Vector<T>& operator=(Vector<T>& tmp)

{

delete[] data;

\_size = tmp.size();

data = new T[\_size];

for (int i = 0; i < \_size; i++)

{

data[i] = tmp[i];

}

return \*this;

}

T& operator[] (int index)

{

if (index < \_size)

return data[index];

else

cout << "\nError index > \_size!!" << endl;

}

Vector<T>& operator \* (T k)

{

Vector<T>tmp(\_size, k);

for (int i = 0; i < \_size; i++)

tmp.data[i] = data[i] \* k;

return tmp;

}

Vector<T>& operator-(Vector<T>& tmp)

{

if (\_size >= tmp.size())

{

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

{

data[i] = data[i] - tmp[i];

}

}

else

{

for (int i = 0; i < \_size; i++)

{

data[i] = data[i] - tmp[i];

}

}

return \*this;

}

public:

//ios overloads

friend istream& operator>> <>(istream& in, Vector<T>& a);

friend ostream& operator<< <>(ostream& out, const Vector<T>& a);

friend ostream& operator<< (ostream& out, const Vector<Money>& tmp);

};

template <typename T>

istream& operator>>(istream& in, Vector<T>& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

{

cout << "Enter " << i + 1 << " element:" << endl;

in >> tmp.data[i];

}

return in;

}

template <typename T>

ostream& operator<< <>(ostream& out, const Vector<T>& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

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

return out;

}

ostream& operator<< (ostream& out, const Vector<Money>& tmp)

{

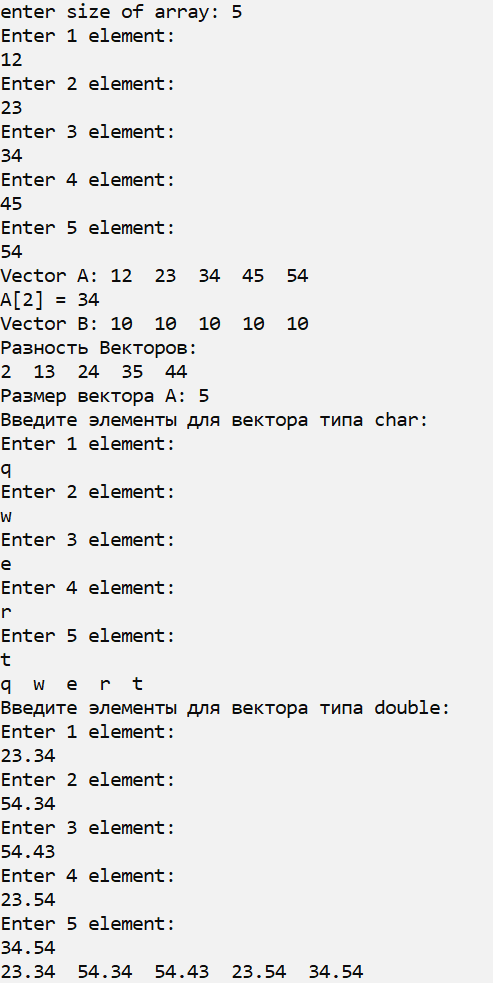
for (int i = 0; i < tmp.\_size; i++)

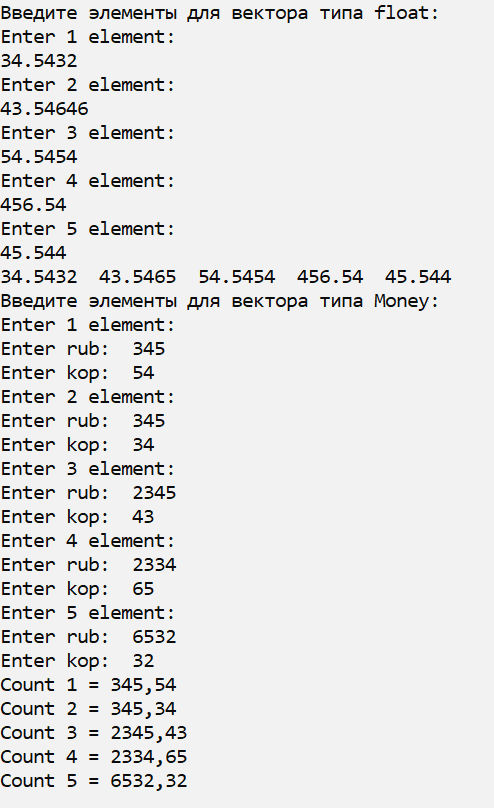
out << "Count "<< i+1 <<" = " << tmp.data[i] << endl;

return out;

}

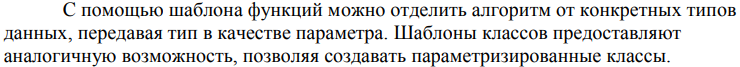
**Скриншоты**



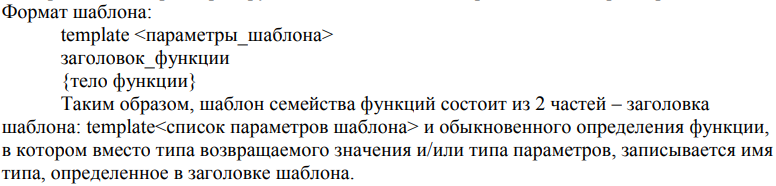


**Контрольные вопросы**

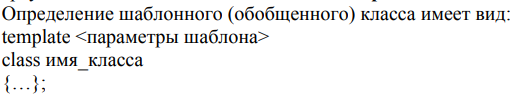


1. 



2. 



3. 



4. Параметры шаблона функции – это параметризированные типы данных, в зависимости от передаваемых данных в параметр.

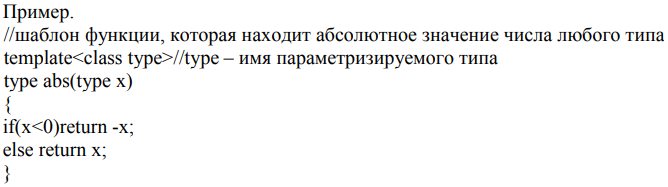


5. Шаблон функции объявляется один раз. Шаблон семейства функции состоит из 2 частей – заголовка шаблона и обыкновенного определения функции.

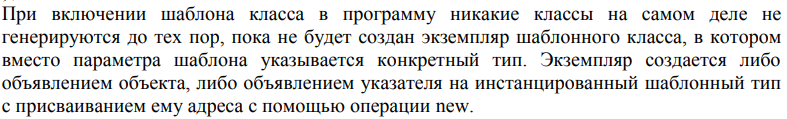


6. Параметр шаблона записывается в определении шаблона: template<typename Type>, далее в описании переменных и методов вместо типа данных необходимо записывать Type.



7. Да, можно. 



8. 



9. Компонентные функции параметризованного класса автоматически являются параметризованными.



10. Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, то есть по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону.



11. Шаблоны функций, которые являются членами классов, нельзя описывать как virtual.



12. Реализация компонентной функции шаблона класса, которая находится вне определения шаблона класса, должна включать дополнительно следующие два элемента:

* Определение должно начинаться с ключевого слова template, за которым следует такой же список\_параметров\_типов в угловых скобках, какой указан в определении шаблона класса.
* За именем\_класса, предшествующим операции области видимости (::), должен следовать список\_имен\_параметров шаблона.

template<список\_типов>

тип\_возвр\_значения имя\_класса<список\_имен\_параметров>::имя\_функции(список\_параметров)

{ ... }



13. Процесс генерации компилятором определения конкретного класса по шаблону класса и аргументам шаблона называется инстанцированием шаблона.



14. На этапе создания объекта перегруженного класса создаётся определение класса по шаблону.