МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №3**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Перегрузка операций

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

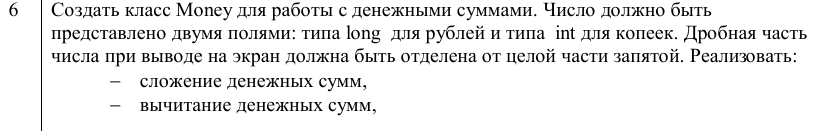
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2025

ВВЕДЕНИЕ



ОСНОВНАЯ ЧАСТЬ

**Блок-схема:**

**Код:**

**Main:**

// Lab3\_main.cpp

#include "Money.h"

#include <iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

Money m1(10, 50);

Money m2(5, 75);

cout << "Сумма 1: " << m1 << endl;

cout << "Сумма 2: " << m2 << endl;

Money sum = m1 + m2;

cout << "Сумма: " << sum << endl;

Money diff = m1 - m2;

cout << "Разница: " << diff << endl;

Money m3;

cout << "Сумма 3: " << endl;

cin >> m3;

cout << "Сумма 3: " << m3 << endl;

return 0;

}

**Money.cpp:**

// Money.cpp

#include "Money.h"

using namespace std;

Money::Money() : rubles(0), kopecks(0) {}

Money::Money(long rubles, int kopecks) : rubles(rubles), kopecks(kopecks) {

normalize(); // Функция для приведения копеек к рублям, если нужно

}

Money::Money(const Money& other) : rubles(other.rubles), kopecks(other.kopecks) {}

Money::~Money() {}

long Money::getRubles() const {

return rubles;

}

int Money::getKopecks() const {

return kopecks;

}

void Money::setRubles(long rubles) {

this->rubles = rubles;

}

void Money::setKopecks(int kopecks) {

this->kopecks = kopecks;

normalize(); // При изменении копеек, нужно нормализовать

}

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

if (this != &other) {

rubles = other.rubles;

kopecks = other.kopecks;

}

return \*this;

}

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

Money result(rubles + other.rubles, kopecks + other.kopecks);

result.normalize();

return result;

}

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

Money result(rubles - other.rubles, kopecks - other.kopecks);

result.normalize();

return result;

}

ostream& operator<<(ostream& os, const Money& money) {

os << money.rubles << "," << (money.kopecks < 10 ? "0" : "") << money.kopecks;

return os;

}

istream& operator>>(istream& is, Money& money) {

cout << "Введите рубли: ";

is >> money.rubles;

cout << "Введите копейки: ";

is >> money.kopecks;

money.normalize();

return is;

}

void Money::normalize() {

rubles += kopecks / 100;

kopecks %= 100;

if (kopecks < 0) {

rubles--;

kopecks += 100;

}

}

**Money.h:**

#pragma once

// Money.h

#ifndef MONEY\_H

#define MONEY\_H

#include <iostream>

#include <string>

class Money {

private:

long rubles;

int kopecks;

public:

// Конструкторы

Money();

Money(long rubles, int kopecks);

Money(const Money& other);

// Деструктор

~Money();

// Селекторы

long getRubles() const;

int getKopecks() const;

// Модификаторы

void setRubles(long rubles);

void setKopecks(int kopecks);

// Перегрузка оператора присваивания

Money& operator=(const Money& other);

// Перегрузка операторов сложения и вычитания

Money operator+(const Money& other) const;

Money operator-(const Money& other) const;

// Дружественные функции для перегрузки операторов ввода/вывода

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

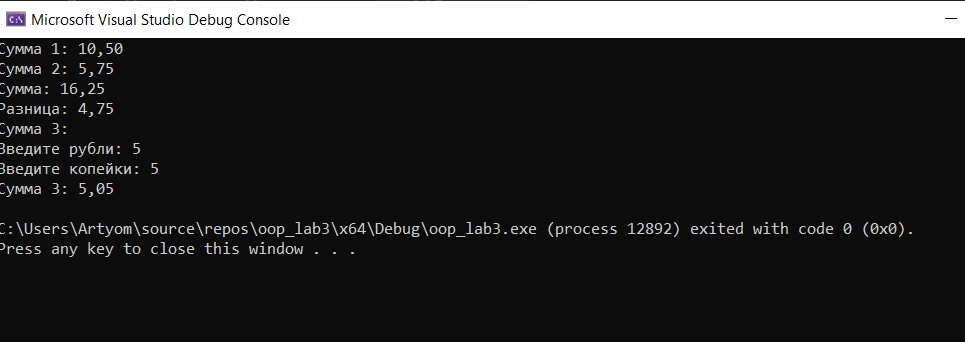
friend std::istream& operator>>(std::istream& is, Money& money);

void normalize();

};

#endif

**Результат работы кода:**

****

**Контрольные вопросы:**

1. Для чего используются дружественные функции и классы?

Дружественные функции и классы нужны, чтобы разрешить доступ к приватным (private) и защищенным (protected) членам класса извне этого класса. Обычно приватные и защищенные члены класса недоступны для других классов или функций, но с помощью дружбы можно сделать исключение. Это полезно, когда нужно предоставить особый доступ к классу, не нарушая принципы инкапсуляции.

2. Сформулировать правила описания и особенности дружественных функций.

• Дружба объявляется внутри класса, которому нужна дружба, с помощью ключевого слова friend.

• friend можно объявить для функции или другого класса.

• Объявление friend указывает, что данная функция или класс имеет право доступа к приватным и защищенным членам класса, в котором объявлена дружба.

• Дружба не является взаимной. Если класс A объявлен другом класса B, это не означает, что класс B является другом класса A.

• Дружба не наследуется. Если класс A является другом класса B, это не означает, что производные классы от A также являются друзьями класса B.

• Дружественные функции не являются членами класса, в котором они объявлены друзьями.

• Порядок объявления friend не важен

3. Каким образом можно перегрузить унарные операции?

Унарные операции (например, ++, --, -, !) можно перегрузить двумя способами:

• Как метод класса: В этом случае, унарная операция будет применяться к объекту этого класса.

• Как внешнюю функцию: В этом случае, функция принимает один аргумент типа этого класса.

4. Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?

Унарная функция-операция, определяемая внутри класса, не имеет явных операндов в списке параметров. Неявным операндом является сам объект, для которого вызывается метод То есть, список параметров будет пустым.

5. Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?

Унарная функция-операция, определяемая вне класса, должна иметь один операнд в списке параметров. Этот операнд является объектом класса, к которому применяется операция.

6. Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?

Бинарная функция-операция, определяемая внутри класса, должна иметь один явный операнд в списке параметров. Второй операнд – это сам объект, для которого вызывается метод (указатель this).

7. Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?

Бинарная функция-операция, определяемая вне класса, должна иметь два операнда в списке параметров. Оба операнда должны быть указаны явно.

8. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Для различения префиксной (++x) и постфиксной (x++) форм унарных операторов при перегрузке используется фиктивный параметр типа int для постфиксной формы.

• Префиксная форма (++x):

\* Определяется как T& operator++(). (Возвращает ссылку на измененный объект).

• Постфиксная форма (x++):

Определяется как T operator++(int). (Возвращает копию неизмененного\* объекта). Фиктивный параметр int служит только для сигнализации, что это постфиксная форма.

9. Каким образом можно перегрузить операцию присваивания?

Операция присваивания перегружается как метод класса с именем operator=. Он принимает один аргумент – объект того же класса, который находится справа от знака =.

c++

class MyClass {

public:

MyClass& operator=(const MyClass& other) {

// Код для присваивания значений полей из 'other' в текущий объект

return \*this;

}

};

10. Что должна возвращать операция присваивания?

Операция присваивания должна возвращать \*ссылку на текущий объект\* (\*this). Это позволяет выполнять цепочные присваивания (например, a = b = c;). Тип возвращаемого значения должен быть MyClass&, где MyClass - имя вашего класса.

11. Каким образом можно перегрузить операции ввода-вывода?

Операции ввода-вывода (например, << для вывода и >> для ввода) перегружаются как \*дружественные функции\* класса, так как левый операнд (std::ostream или std::istream) не является объектом вашего класса.

class MyClass {

friend std::ostream& operator<<(std::ostream& os, const MyClass& obj);

friend std::istream& operator>>(std::istream& is, MyClass& obj);

};

std::ostream& operator<<(std::ostream& os, const MyClass& obj) {

// Код для вывода объекта 'obj' в поток 'os'

return os;

}

std::istream& operator>>(std::istream& is, MyClass& obj) {

// Код для ввода данных в объект 'obj' из потока 'is'

return is;

}

**Ссылка на GitHub:**

<https://github.com/Drazda1488/oop_3lab>