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 ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Классы и объекты. Использование конструкторов."

Выполнила работу

Студентка группы РИС-22-1Б

Лихачев Д.А.

Проверила

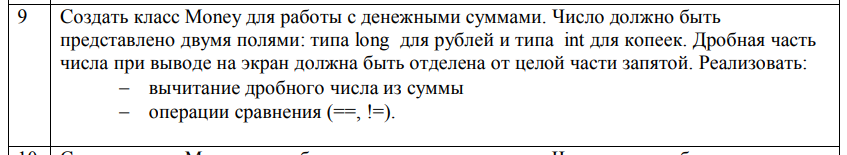
Доцент кафедры ИТАС

Полякова О.А.

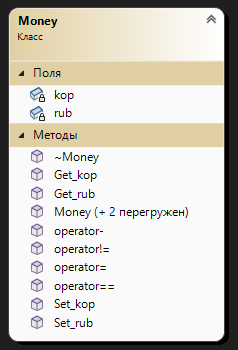
Г. Пермь-2023

**Постановка задачи**

1. Определить пользовательский класс.  
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.  
3. Определить в классе деструктор.  
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).  
5. Перегрузить операцию присваивания.  
6. Перегрузить операции ввода и вывода объектов с помощью потоков.  
7. Перегрузить операции указанные в варианте.  
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.



**Диаграмма классов**



**Код программы**

Money.h:

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Money

{

private:

long rub;

int kop;

public:

//Конструкторы + деструктор

Money();

Money(long, int);

Money(const Money& p);

~Money();

//Селекторы и модификаторы

long Get\_rub();

int Get\_kop();

void Set\_rub(long rub);

void Set\_kop(long kop);

//Перегруженные операторы

Money& operator-(double money);

Money& operator=(const Money& p);

bool operator==(const Money& p);

bool operator!=(const Money& p);

//Перегруженные методы ввода/вывода

friend ostream& operator<<(ostream& fout, const Money& p);

friend istream& operator>>(istream& fin, Money& p);

};

Money.cpp:

#include <iostream>

#include <fstream>

#include "Money.h"

using namespace std;

Money::Money() {

this->rub = 0;

this->kop = 0;

}

Money::Money(long rub, int kop) {

this->rub = rub;

this->kop = kop;

}

Money::Money(const Money& p) {

rub = p.rub;

kop = p.kop;

}

Money::~Money() {

}

long Money::Get\_rub() {

return rub;

}

int Money::Get\_kop() {

return kop;

}

void Money::Set\_rub(long rub) {

this->rub = rub;

}

void Money::Set\_kop(long kop) {

this->kop = kop;

}

Money& Money::operator-(double money) {

int celaya = (int)money;

int drobnaya = int((money - celaya) \* 100.0);

rub -= celaya;

kop -= drobnaya;

return \*this;

}

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

this->rub = p.rub;

this->kop = p.kop;

return \*this;

}

bool Money::operator==(const Money& p) {

return this->rub == p.rub && this->kop == p.kop;

}

bool Money::operator!=(const Money& p) {

return !(this->rub == p.rub && this->kop == p.kop);

}

ostream& operator<<(ostream& fout, const Money& p) {

fout << p.rub << "," << p.kop << "\n";

return fout;

}

istream& operator>>(istream& fin, Money& p) {

fin >> p.rub >> p.kop;

return fin;

}

Lab3\_main.cpp:

#include <iostream>

#include <fstream>

#include "Money.h"

using namespace std;

int main() {

Money a;

Money b;

Money c;

cout << "Enter a:" << endl;

cin >> a;

cout << "Enter b:" << endl;

cin >> b;

cout << "Enter c:" << endl;

cin >> c;

c = a;

(a == b) ? cout << "a==b" << endl : cout << "a!=b"<<endl;

(a == c) ? cout << "a==c" << endl : cout << "a!=c" << endl;

cout<<b - 24.25 << endl;

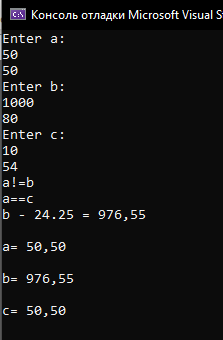
cout << "a= "<< a << endl;

cout << "b= " << b << endl;

cout << "c= " << c << endl;

}

**Результаты работы программы**

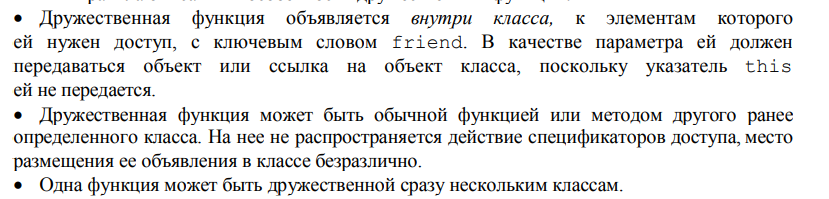


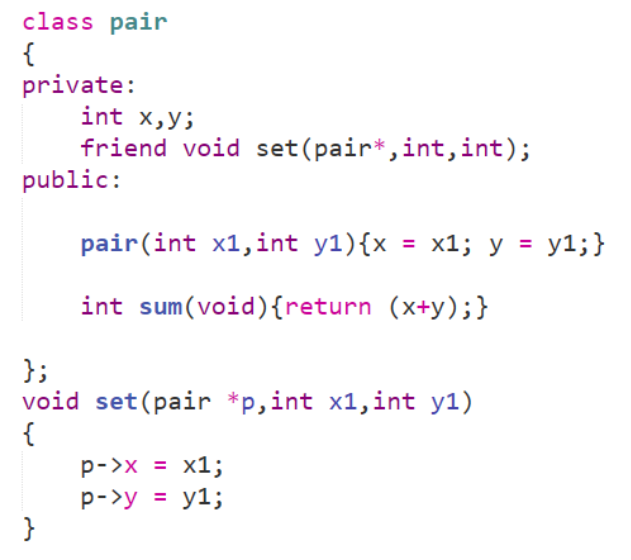
**Ответы на контрольные вопросы**

1. Для чего используются дружественные функции и классы?

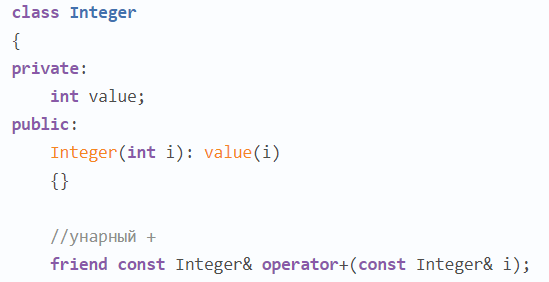
Дружественная функция - это функция, которая может получить доступ к закрытым членам класса, как если бы она была членом этого класса.

2. Сформулировать правила описания и особенности дружественных функций.



  
3. Каким образом можно перегрузить унарные операции?

Унарную операцию можно перегрузить:  
- Как компонентную функцию класса



- Как внешнюю (глобальную) функцию

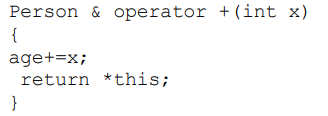
4. Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?

0   
5. Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?

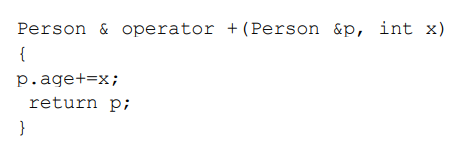
1

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

1. Ее операнд – вызвавший ее же объект.

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

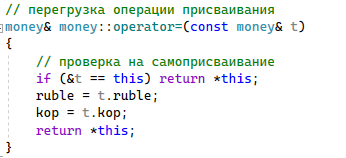
2 операнда типа класса.



8. Чем отличается перегрузка префиксных и постфиксных унарных операций?

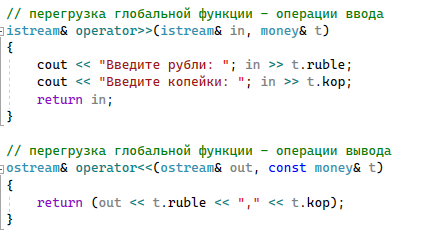
* если перегружается префиксная форма оператора ++, то в классе нужно реализовать операторную функцию operator++() без параметров;
* если перегружается префиксная форма оператора —, то в классе нужно реализовать операторную функцию operator—() без параметров;
* если перегружается постфиксная форма оператора ++, то в классе нужно реализовать операторную функцию operator++(int d) с одним целочисленным параметром. В этом случае параметр d не используется в функции. Он указывается только для того, чтобы указать что это именно постфиксная реализация оператора **++**. Имя d может быть заменено другим именем;
* если перегружается постфиксная форма оператора **—**, то в классе нужно реализовать операторную функцию operator—(int d) с одним параметром. Параметр d необходим для указания того, что перегружается именно постфиксная реализация оператора**—**

9. Каким образом можно перегрузить операцию присваивания?

  
10. Что должна возвращать операция присваивания?

Объект  
11. Каким образом можно перегрузить операции ввода-вывода?

При помощи дружественной глобальной ф-ии.



12. В программе описан класс class Student  
{  
Student& operator++();  
};  
и определен объект этого класса  
Student s;  
Выполняется операция  
++s;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Префиксный инкремент.

13. В программе описан класс  
class Student  
{  
friend Student& operator ++( Student&);  
};  
и определен объект этого класса  
Student s;  
Выполняется операция  
++s;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Постфиксный инкремент.

14. В программе описан класс class Student  
{  
bool operator<(Student &P);  
};  
и определены объекты этого класса  
Student a,b;  
Выполняется операция  
cout<<a<b;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Если а действительно меньше b вернется 1.

15. В программе описан класс  
class Student  
{  
friend bool operator >(const Person&, Person&)  
};  
и определены объекты этого класса  
Student a,b,  
Выполняется операция  
cout<<a>b;  
Каким образом, компилятор будет воспринимать вызов функции-операции?

Если а действительно больше b вернется 1.