МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

##### «МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ»

##### ***Группа 82 ТП***

## **Отчет**

## **по лабораторной работе № 25**

# "РАЗРАБОТКА И ОТЛАДКА АЛГОРИТМОВ И ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ"

учебный предмет

«Основы алгоритмизации и программирования»

**Исполнитель: Шатуха А.К.**

**Руководитель: Д.А.Федкевич**

Минск 2025

Цель работы:

Применить на практике знания:

* по созданию производных классов, образованных от нескольких базовых;
* по вызову методов базовых классов.

## ЗАДАНИЯ

Для выполнения задания внесите дополнения в Лабораторную работу №24. Варианты заданий остаются прежними.

### Вариант 14

Задание: Множественное наследование.

Разработайте базовый класс BANK. Элемент класса:

* название банка получателя.

Методы:

* метод, осуществляющий ввод значений полей класса с клавиатуры;
* метод, осуществляющий вывод значений полей класса на экран.

Разработайте класс RENTA – производный от класса ORDER1 и BANK. В этот класс добавьте метод, в котором сравниваются названия банка отправителя и банка получателя. Если они совпадают, то выводится на экран сообщение «Без комиссии», если нет – выводится сообщение «Комиссия 0.5%» и оставшаяся сумма.

Листинг или ссылка на проект:

#include <iostream>

#include <string>

using namespace std;

class ORDER {

protected:

string orderNumber;

double amount;

public:

ORDER() : orderNumber(""), amount(0.0) {

cout << "Вызван конструктор без параметров для ORDER" << endl;

}

ORDER(const string& number, double amt) : orderNumber(number), amount(amt) {

cout << "Вызван конструктор с параметрами для ORDER" << endl;

}

ORDER(const ORDER& other) : orderNumber(other.orderNumber), amount(other.amount) {

cout << "Вызван конструктор копирования для ORDER" << endl;

}

virtual ~ORDER() {

cout << "Вызван деструктор для ORDER" << endl;

}

virtual void input() {

cout << "Введите номер: ";

getline(cin, orderNumber);

cout << "Введите сумму: ";

cin >> amount;

cin.ignore();

}

virtual void display() const {

cout << "Номер заказа: " << orderNumber << endl;

cout << "Сумма заказа: " << amount << endl;

}

};

class BANK {

protected:

string receiverBank;

public:

BANK() : receiverBank("") {

cout << "Вызван конструктор без параметров для BANK" << endl;

}

BANK(const string& bank) : receiverBank(bank) {

cout << "Вызван конструктор с параметрами для BANK" << endl;

}

void input() {

cout << "Введите название банка получателя: ";

getline(cin, receiverBank);

}

void display() const {

cout << "Банк получателя: " << receiverBank << endl;

}

};

class ORDER1 : public ORDER {

private:

string senderBank;

public:

ORDER1() : ORDER(), senderBank("") {

cout << "Вызван конструктор без параметров для ORDER1" << endl;

}

ORDER1(const string& number, double amt, const string& bank)

: ORDER(number, amt), senderBank(bank) {

cout << "Вызван конструктор с параметрами для ORDER1" << endl;

}

ORDER1(const ORDER1& other) : ORDER(other), senderBank(other.senderBank) {

cout << "Вызван конструктор копирования для ORDER1" << endl;

}

~ORDER1() override {

cout << "Вызван деструктор для ORDER1" << endl;

}

void input() override {

ORDER::input();

cout << "Введите банк-отправитель: ";

getline(cin, senderBank);

}

void display() const override {

ORDER::display();

cout << "Банк-отправитель: " << senderBank << endl;

}

string getSenderBank() const {

return senderBank;

}

};

class RENTA : public ORDER1, public BANK {

public:

RENTA() : ORDER1(), BANK() {

cout << "Вызван конструктор без параметров для RENTA" << endl;

}

RENTA(const string& number, double amt, const string& sender, const string& receiver)

: ORDER1(number, amt, sender), BANK(receiver) {

cout << "Вызван конструктор с параметрами для RENTA" << endl;

}

void input() override {

ORDER1::input();

BANK::input();

}

void display() const override {

ORDER1::display();

BANK::display();

}

void checkCommission() const {

if (getSenderBank() == receiverBank) {

cout << "Без комиссии" << endl;

}

else {

double commission = amount \* 0.005;

double remainingAmount = amount - commission;

cout << "Комиссия 0.5%: " << commission << endl;

cout << "Остаточная сумма: " << remainingAmount << endl;

}

}

};

int main() {

setlocale(LC\_ALL, "ru");

cout << "\nСоздание объекта RENTA с конструктором без параметров:\n";

RENTA renta;

renta.input();

renta.display();

renta.checkCommission();

cout << "\nСоздание объекта RENTA с конструктором с параметрами:\n";

RENTA renta2("ORD123", 500.75, "Sberbank", "Sberbank");

renta2.display();

renta2.checkCommission();

cout << "\nСоздание объекта RENTA с различными банками:\n";

RENTA renta3("ORD456", 300.50, "Sberbank", "VTB");

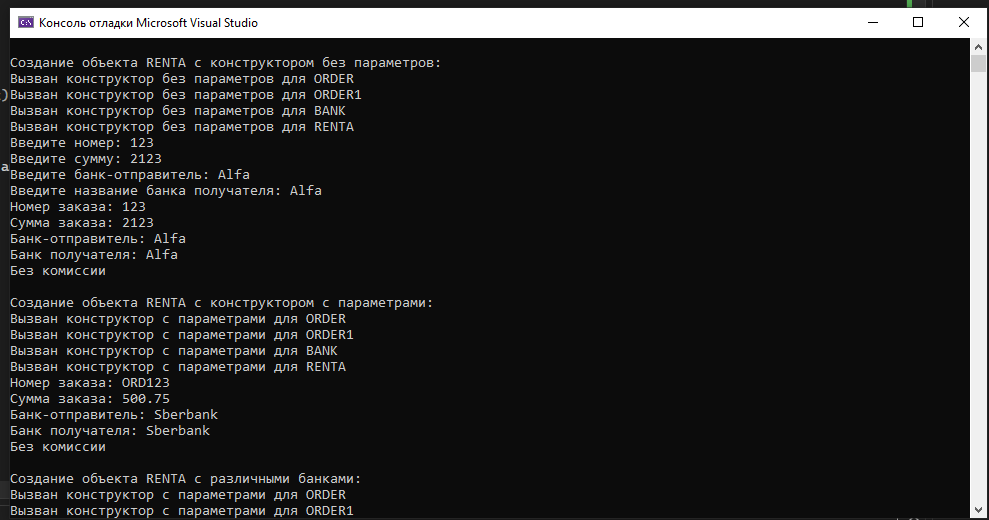
renta3.display();

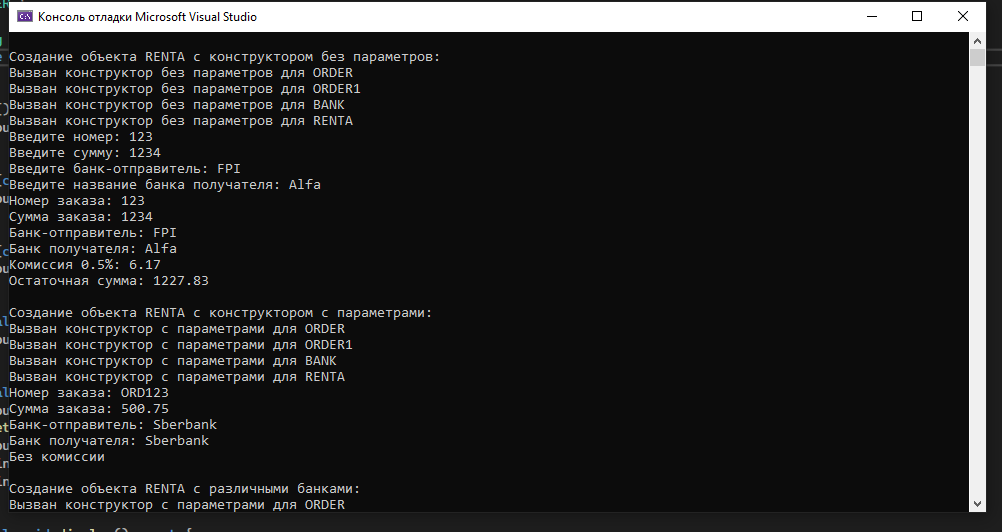
renta3.checkCommission();

return 0;

}

Скриншот:





## КОНТРОЛНЫЕ ВОПРОСЫ

1. Поясните, чем отличаются различные типы наследования: public, protected, private.
2. Объясните, что представляет собой множественное наследование.
3. Приведите синтаксис описания производного класса при множественном наследовании.