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 № 18.3**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Перегрузка операций

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

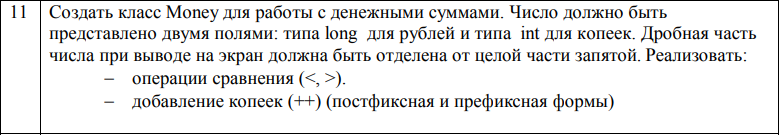
**Пермь, 2021**

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Использование перегруженных операций в классах.

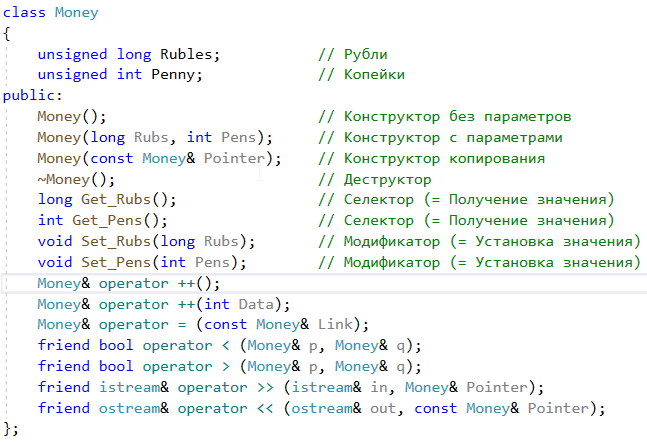
**Постановка задачи**

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 операцию ввода и вывода объектов с помощью потоков.
7. Перегрузить операции, указанные в варианте.
8. Написать демонстрационную программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.



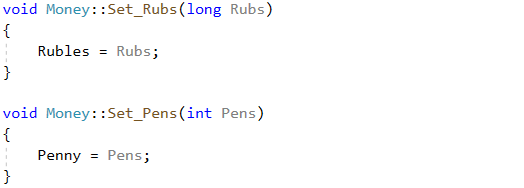
**Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать определение класса Money.

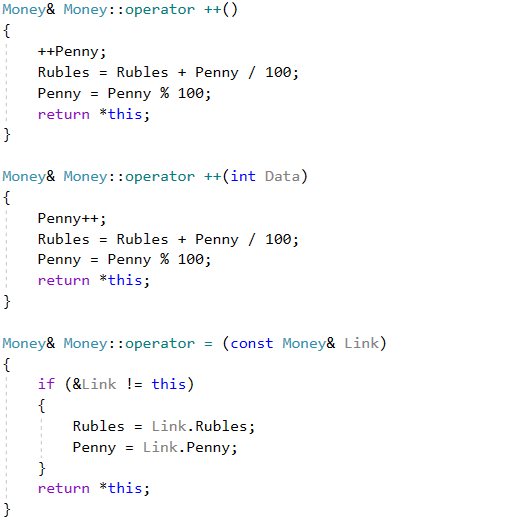


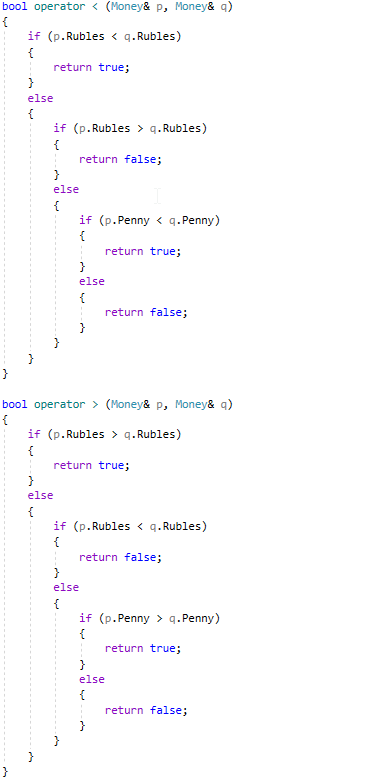
* 1. Реализовать определение конструкторов без параметра, с параметрами и копирования, деструктора, селекторов и модификаторов класса.

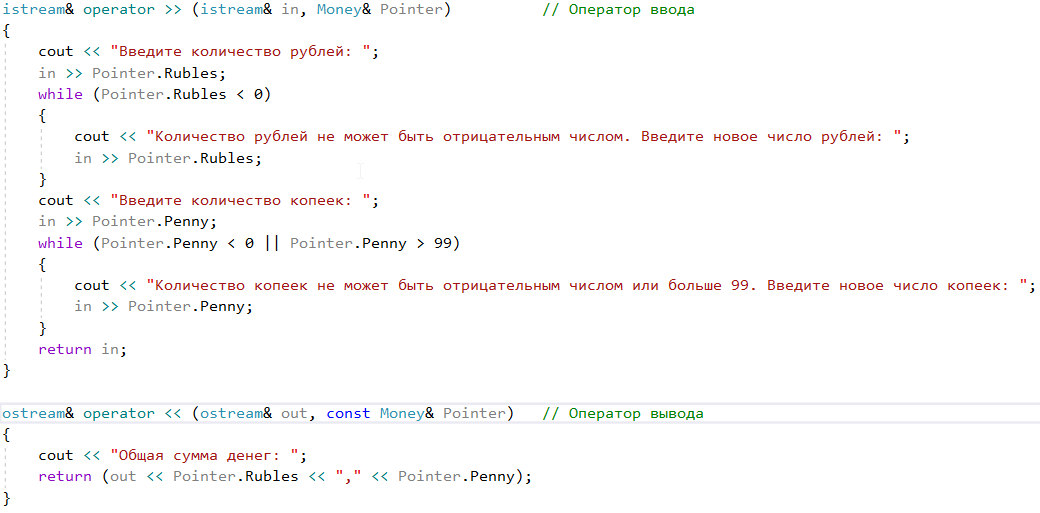




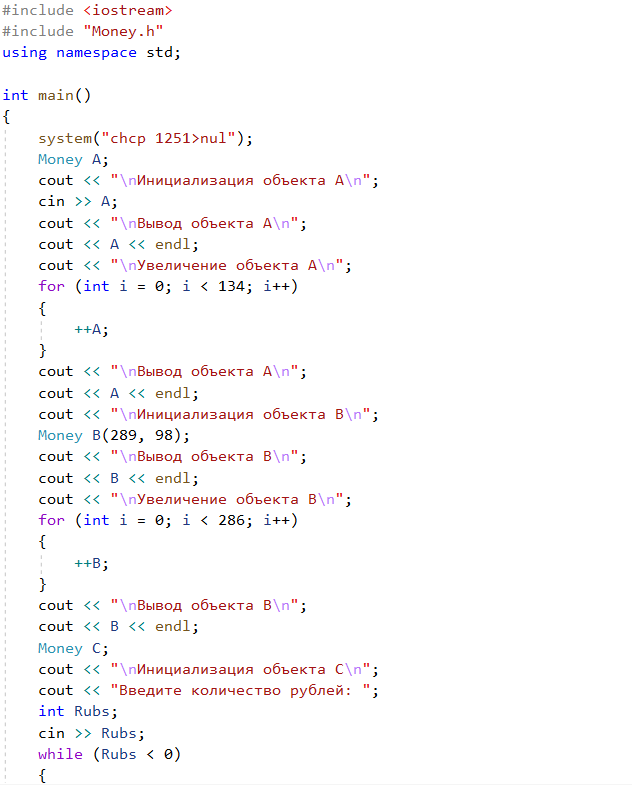
* 1. Реализовать определение перегрузки операторов.

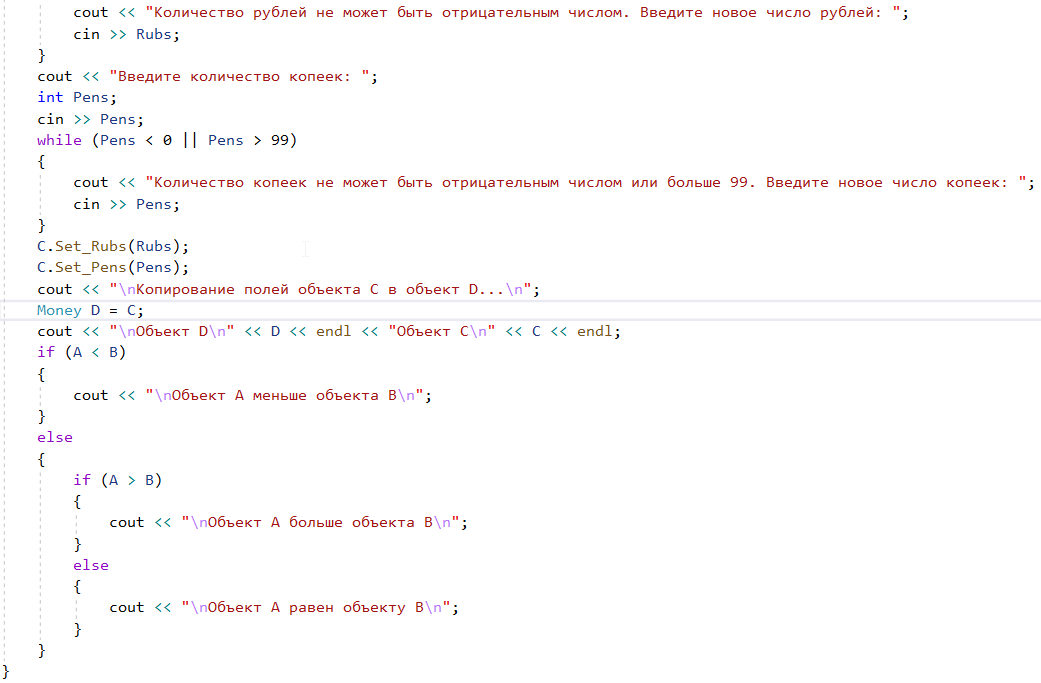






* 1. Реализовать применение этих функций в главной функции.





1. В ходе работы были использованы типы данных:
   1. Для конструктора без параметров Money класса Money не используются аргументы:

Сам конструктор заполняет атрибуты объекта класса “пустотами”

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_gDUB2zbPet.png

* 1. Для конструктора с параметрами Money класса Money используются следующие аргументы:
  2. Тип int: количество рублей.
  3. Тип int: количество копеек.

Сам конструктор заполняет атрибуты объекта класса параметрами.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_OPh5HkgEj3.png

* 1. Для конструктора копирования Money класса Money используются следующие аргументы:
  2. Адресация типа const Money: объект копирования.

Сам конструктор заполняет атрибуты объекта класса атрибутами объекта копирования.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_zoUJiZwFzm.png

* 1. Для деструктора Money класса Money не используются аргументы:

Сам деструктор удаляет атрибуты объекта класса.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_ncigfS8PfL.png

* 1. Для селектора Get\_Rubs не используются аргументы.

Сам селектор имеет тип int и возвращает значение поля Rubles.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_6Sem7W8rR6.png

* 1. Для селектора Get\_Pens не используются аргументы.

Сам селектор имеет тип int и возвращает значение поля Penny.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_UiulFygU8n.png

* 1. Для модификатора Set\_Rubs используются следующие аргументы:

1. Тип int: количество рублей.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_S9bQlBw6XS.png

* 1. Для модификатора Set\_Pens используются следующие аргументы:

1. Тип int: количество копеек.

Сам модификатор имеет тип void, поскольку при работе селектора не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_42DhuICUNa.png

* 1. Для метода перегрузки оператора префиксного инкремента не используются аргументы.

Сам метод имеет тип Money& и возвращает указатель на объект this.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_uGe84IKY02.png

* 1. Для метода перегрузки постфиксного инкремента используются следующие аргументы:
  2. Тип int: показатель того, что происходит перегрузка постфиксного инкремента.

Сам метод имеет тип Money& и возвращает указатель на объект this.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_DSb639ql27.png

* 1. Для метода перегрузки оператора присваивания используются следующие аргументы:
  2. Тип const Money&: ссылка на объект.

Сам метод имеет тип Money& и возвращает указатель на объект this.

* 1. Для дружественной функции перегрузки оператора ввода используются следующие аргументы:
  2. Тип istream&: поток ввода.
  3. Тип const Money&: ссылка на объект.

Сама функция имеет тип istream& и возвращает вводимые данные.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_LZ4ZrF89TI.png

* 1. Для дружественной функции перегрузки оператора вывода используются следующие аргументы:
  2. Тип ostream&: поток вывода.
  3. Тип const Money&: ссылка на объект.

Сама функция имеет тип ostream& и возвращает выводимые данные.



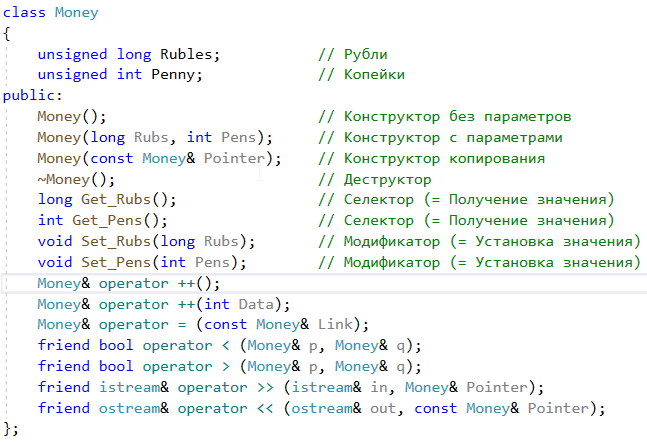
* 1. Для метода перегрузки операторов сравнения используются следующие аргументы:
  2. Тип Money&: ссылка на 1-ый сравниваемый объект.
  3. Тип Money&: ссылка на 2-ой сравниваемый объект.

Сам метод имеет тип Money& и возвращает указатель на объект this.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_OziWnES2Y3.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_orIBY4ls1t.png

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.



1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin и getline, используемых при реализации в главной функции и функций перегрузки оператора.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_vFBKX3dMBB.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_KNTULcXt7b.png

* 1. Вывод данных реализован с помощью оператора cout, используемого при реализации в главной функции и функций перегрузки операторов ввода-вывода.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_DXNSzzoJ5j.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_GIa8OQ3H9K.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_gaIKFzqZyr.png

1. Поставленные задачи решены следующими действиями:
   1. Определение класса Money было реализовано в заголовочном файле Money.h, определение конструкторов, деструктора, селекторов и модификаторов класса – в файле Money.cpp. Основной блок программы описан в главном файле Лабораторная работа № 18.3.cpp.
   2. При работе с объектами класса Money обращение к конструктору без параметра аналогично выделению памяти под переменную. Обращение к конструктору с параметрами аналогично обращению к методу Init. Обращение к конструктору копирования аналогично оператору присваивания. Обращение к селекторам и модификаторам аналогично обращению к методам или к полям структуры.

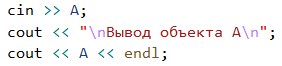
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_ACinEOepfF.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_SHVS1qZcNW.png

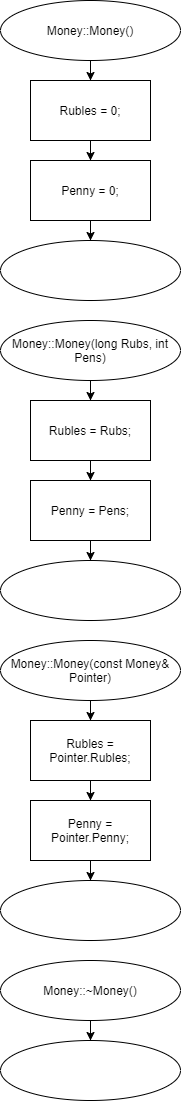
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_cD4BdjRlkh.png

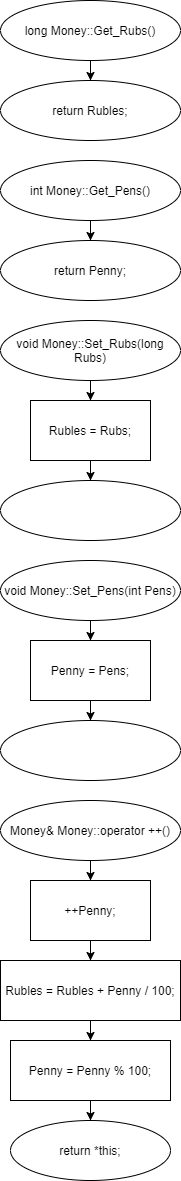
* 1. При работе с перегруженными операторами работа без объектов созданного класса происходит по умолчанию. Если при вызове перегруженных операторов используются объекты класса, то операторы работают согласно их определению в этом классе.

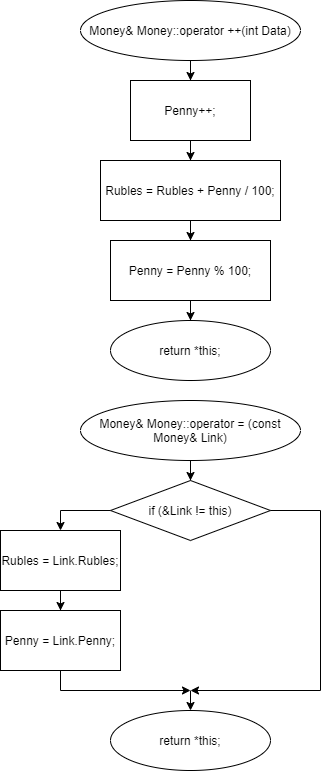
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_YMnnLesxGK.png

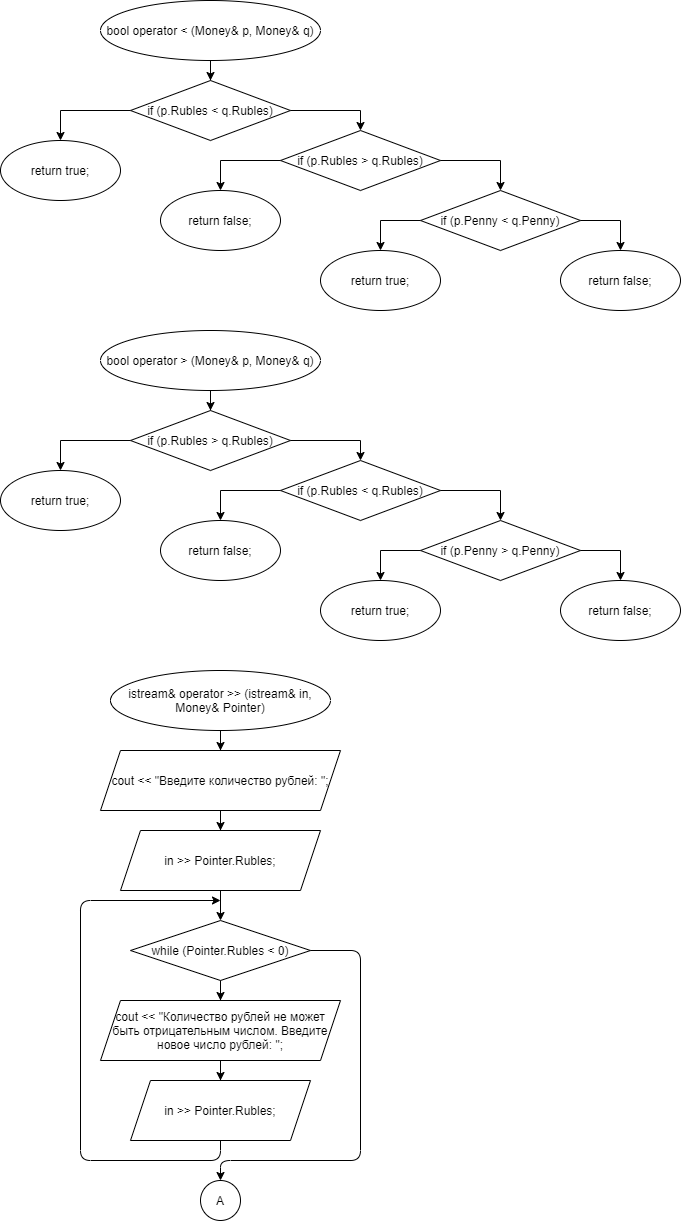


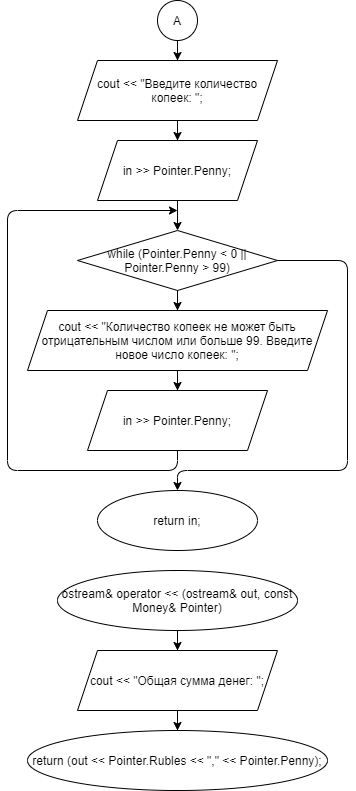
**Блок-схема программы**

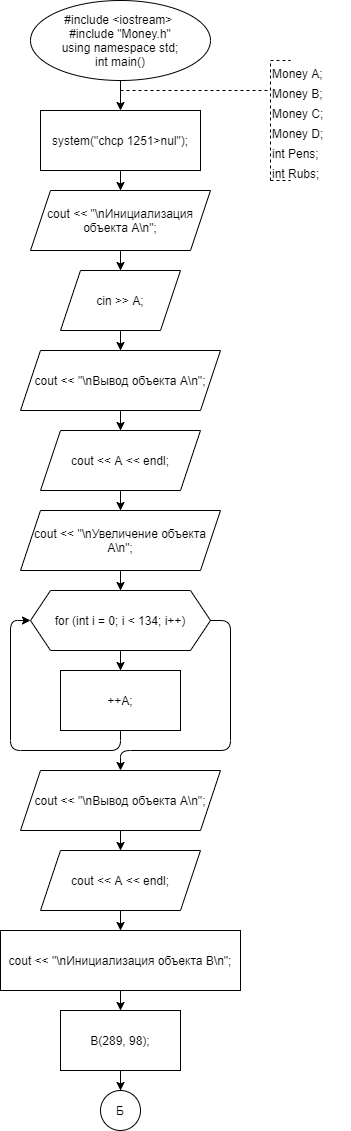


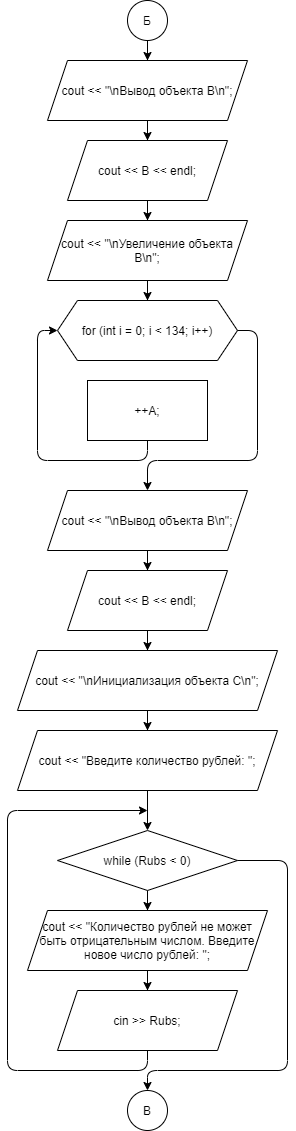


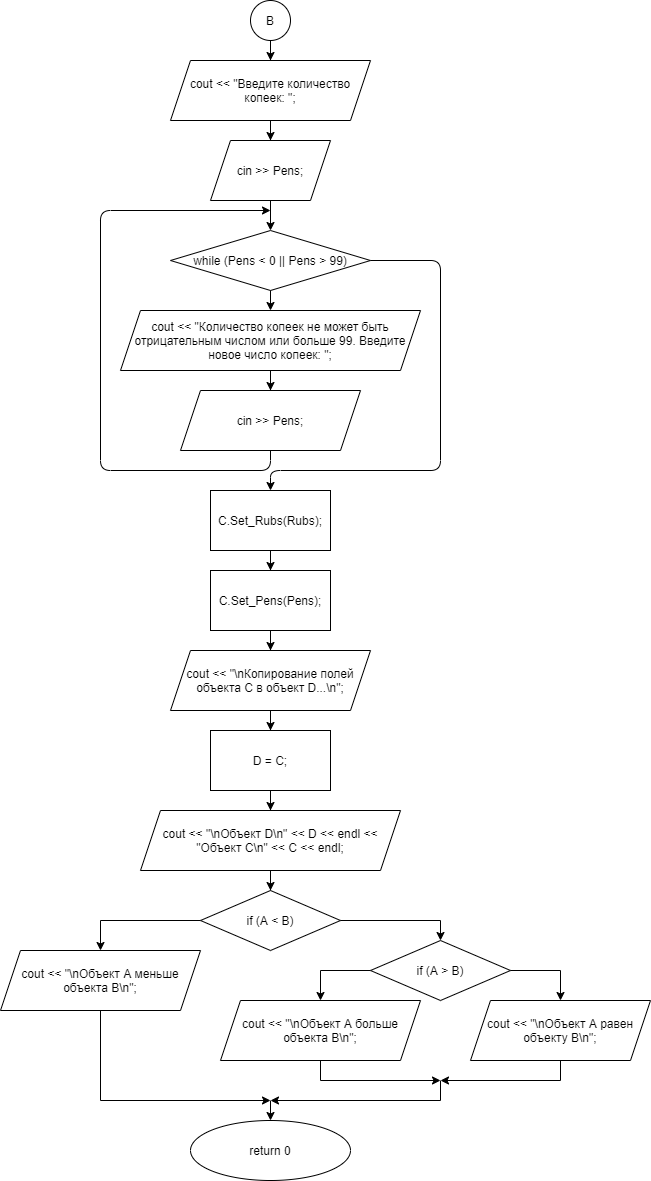












**Реализация задачи на языке С++**

**Заголовочный файл Money.h**

#pragma once

#include <iostream>

using namespace std;

class Money

{

unsigned long Rubles;

unsigned int Penny;

public:

Money();

Money(long Rubs, int Pens);

Money(const Money& Pointer);

~Money();

long Get\_Rubs();

int Get\_Pens();

void Set\_Rubs(long Rubs);

void Set\_Pens(int Pens);

Money& operator ++();

Money& operator ++(int Data);

Money& operator = (const Money& Link);

friend bool operator < (Money& p, Money& q);

friend bool operator > (Money& p, Money& q);

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

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

};

**Файл с описанием методов класса Money.cpp**

#include "Money.h"

#include <iostream>

using namespace std;

Money::Money()

{

Rubles = 0;

Penny = 0;

}

Money::Money(long Rubs, int Pens)

{

Rubles = Rubs;

Penny = Pens;

}

Money::Money(const Money& Pointer)

{

Rubles = Pointer.Rubles;

Penny = Pointer.Penny;

}

Money::~Money()

{

}

long Money::Get\_Rubs()

{

return Rubles;

}

int Money::Get\_Pens()

{

return Penny;

}

void Money::Set\_Rubs(long Rubs)

{

Rubles = Rubs;

}

void Money::Set\_Pens(int Pens)

{

Penny = Pens;

}

Money& Money::operator ++()

{

++Penny;

Rubles = Rubles + Penny / 100;

Penny = Penny % 100;

return \*this;

}

Money& Money::operator ++(int Data)

{

Penny++;

Rubles = Rubles + Penny / 100;

Penny = Penny % 100;

return \*this;

}

Money& Money::operator = (const Money& Link)

{

if (&Link != this)

{

Rubles = Link.Rubles;

Penny = Link.Penny;

}

return \*this;

}

bool operator < (Money& p, Money& q)

{

if (p.Rubles < q.Rubles)

{

return true;

}

else

{

if (p.Rubles > q.Rubles)

{

return false;

}

else

{

if (p.Penny < q.Penny)

{

return true;

}

else

{

return false;

}

}

}

}

bool operator > (Money& p, Money& q)

{

if (p.Rubles > q.Rubles)

{

return true;

}

else

{

if (p.Rubles < q.Rubles)

{

return false;

}

else

{

if (p.Penny > q.Penny)

{

return true;

}

else

{

return false;

}

}

}

}

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

{

cout << "Введите количество рублей: ";

in >> Pointer.Rubles;

while (Pointer.Rubles < 0)

{

cout << "Количество рублей не может быть отрицательным числом. Введите новое число рублей: ";

in >> Pointer.Rubles;

}

cout << "Введите количество копеек: ";

in >> Pointer.Penny;

while (Pointer.Penny < 0 || Pointer.Penny > 99)

{

cout << "Количество рублей не может быть отрицательным числом или больше 99. Введите новое число копеек: ";

in >> Pointer.Penny;

}

return in;

}

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

{

cout << "Общая сумма денег: ";

return (out << Pointer.Rubles << "," << Pointer.Penny);

}

**Файл с главной программой Лабораторная работа № 18.3.cpp**

#include <iostream>

#include "Money.h"

using namespace std;

int main()

{

system("chcp 1251>nul");

Money A;

cout << "\nИнициализация объекта A\n";

cin >> A;

cout << "\nВывод объекта A\n";

cout << A << endl;

cout << "\nУвеличение объекта A\n";

for (int i = 0; i < 134; i++)

{

++A;

}

cout << "\nВывод объекта A\n";

cout << A << endl;

cout << "\nИнициализация объекта В\n";

Money B(289, 98);

cout << "\nВывод объекта В\n";

cout << B << endl;

cout << "\nУвеличение объекта B\n";

for (int i = 0; i < 286; i++)

{

++B;

}

cout << "\nВывод объекта В\n";

cout << B << endl;

Money C;

cout << "\nИнициализация объекта C\n";

cout << "Введите количество рублей: ";

int Rubs;

cin >> Rubs;

cout << "Введите количество копеек: ";

int Pens;

cin >> Pens;

C.Set\_Rubs(Rubs);

C.Set\_Pens(Pens);

cout << "\nКопирование полей объекта C в объект D...\n";

Money D = C;

cout << "\nОбъект D\n" << D << endl << "Объект С\n" << C << endl;

if (A < B)

{

cout << "\nОбъект A меньше объекта B\n";

}

else

{

if (A > B)

{

cout << "\nОбъект A больше объекта B\n";

}

else

{

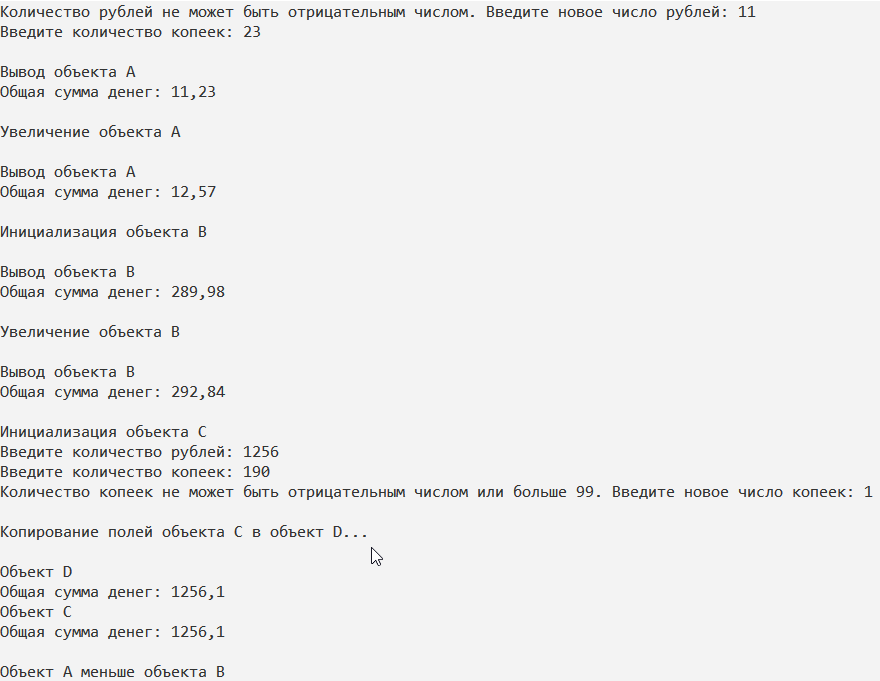
cout << "\nОбъект A равен объекту B\n";

}

}

}

**Скриншоты результатов работы программы**



**Ответы на вопросы**

1. Для чего используются дружественные функции и классы?

Дружественные функции применяются для доступа к скрытым полям класса и представляют собой альтернативу методам. Дружественный класс применяется для доступа методов одного класса к скрытым полям другого.

1. Сформулировать правила описания и особенности дружественных функций.

* Объявление дружественной функции – внутри нужного класса, ключевым словом friend. Передаваемый параметр - объект или ссылка на объект.
* Дружественная функция – обычная функция или метод другого класса. Не важно, где функция описана в классе + спецификаторы доступа не распространяются.
* Одна и та же дружественная функция может быть в нескольких классах.

1. Каким образом можно перегрузить унарные операции?

Унарную операцию можно перегрузить как:

* Компонентная функция класса
* Внешняя (глобальная) функция

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

Унарная функция-операция, определяемая внутри класса, не имеет операндов.

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

Унарная функция-операция, определяемая вне класса, имеет один операнд – ссылка на вызвавший её объект.

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

Бинарная функция-операция, определяемая внутри класса, имеет один операнд – переменная, хранящая значение.

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

Бинарная функция-операция, определяемая вне класса, имеет два операнда. Первый – вызвавший её объект, второй – переменная, хранящая значение.

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Перегрузка префиксной унарной операции не имеет передаваемых параметров, перегрузка постфиксной унарной операции имеет один передаваемый параметр – показатель того, что это постфиксная форма.

1. Каким образом можно перегрузить операцию присваивания?

Перегрузка операции присваивания описывается как метод класса. В качестве принимаемого аргумента используется ссылка на присваиваемый объект, в качестве возвращаемого значения – ссылку на объект, для которого она вызвана.

1. Что должна возвращать операция присваивания?

Операция присваивания возвращает ссылку на объект, для которого она вызвана.

1. Каким образом можно перегрузить операции ввода-вывода?

Перегрузка операций ввода-вывода описываются как дружественные функции, т. к. левый операнд – потоки ввода-вывода.

1. В программе описан класс

class Student

{

……

Student& operator ++();

……

};

И определён объект этого класса

Student s;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Компилятор воспринимает данный вызов как перегруженную функцию-операцию префиксного инкремента, определенную внутри класса.

1. В программе описан класс

class Student

{

……

friend Student& operator ++(Student&);

……

};

И определён объект этого класса

Student s;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Компилятор воспринимает данный вызов как дружественную перегруженную функцию-операцию префиксного инкремента.

1. 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?

Компилятор воспринимает данный вызов как перегруженную функцию-операцию оператора сравнения, определенную внутри класса.

1. В программе описан класс

class Student

{

……

friend bool operator > (const Person&, Person&);

……

};

И определён объект этого класса

Student a, b;

Выполняется операция

cout << a > b;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Компилятор воспринимает данный вызов как дружественную перегруженную функцию-операцию сравнения.