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 №18.2**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование. Использование конструкторов”

Вариант 12

Выполнила:

Студент группы ИВТ-20-2Б Ананина Арина Юрьевна

Проверила:

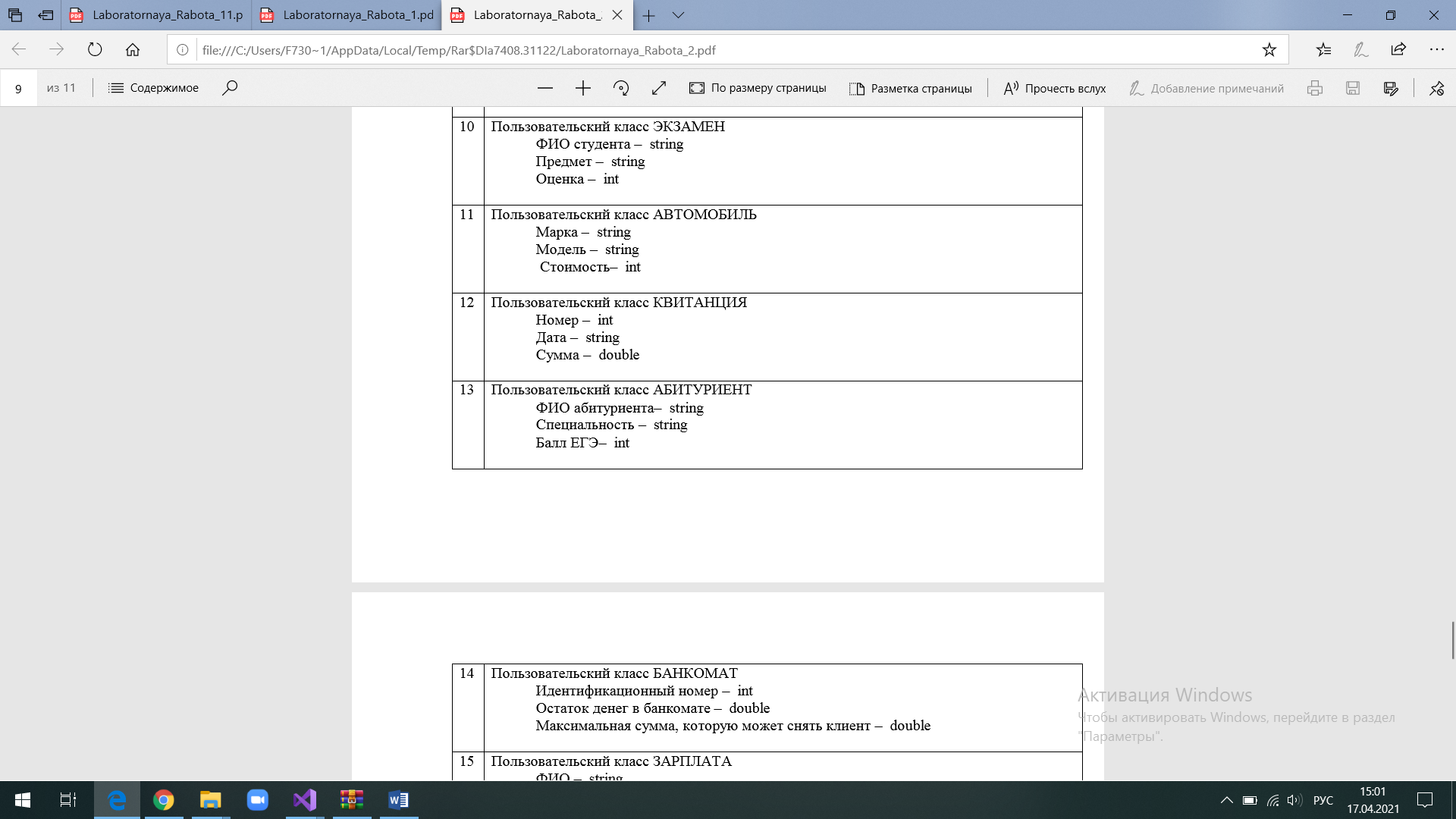
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

Определить пользовательский класс. Определить в класс следующие конструкторы: без параметров, с параметрами, копирования. Определить в классе деструктор.



**Анализ задачи**

**1.** Для решения задачи необходимо:

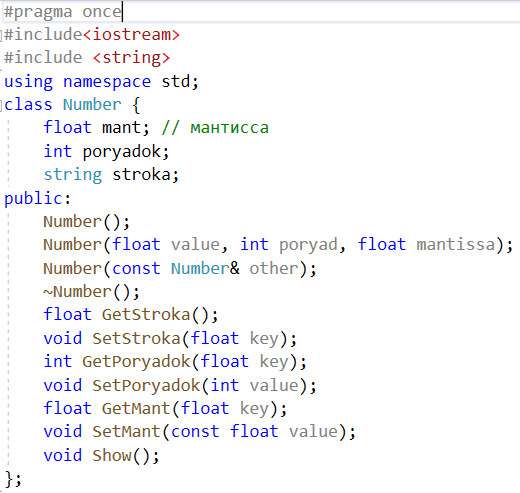
**1.1.** Организовать класс Number для хранения мантиссы в типе float, порядка типа int и самого числа в формате string.

**1.2.** Организовать необходимые методы для ввода данных в поля mant и poryadok, stroka: гетторы, сетторы, конструкторы, деструктор.

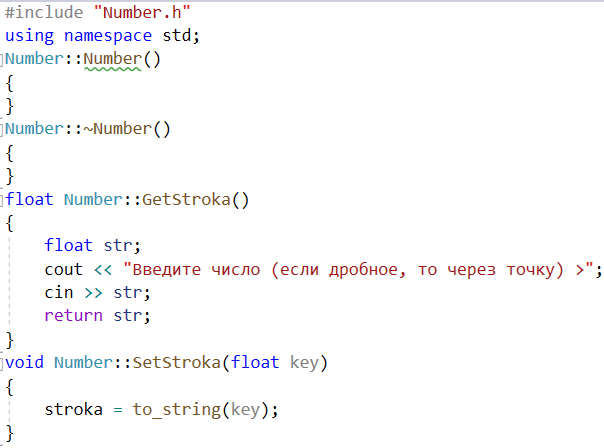
**1.3.** В функции main() использовать множественный if() для выбора пользователем, как он(-а) хочет ввести данные (через конструктор без параметров, с параметрами и конструктор копирования).

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

**2.1.** Организовать класс Number для хранения мантиссы в типе float, порядка типа int и самого числа в формате string в заголовочном файле Number.h.



**2.2.** Организовать соответствующий файл Number.cpp, в котором будут описаны методы класса Number.



**2.3.** Переменные типа float: key, Por, mantissa, method, где key – введённое число, Por – переменная для хранения порядка числа, mantissa – переменная для выбора пользователем, как он хочет выполнить задачу.

float key = 0;

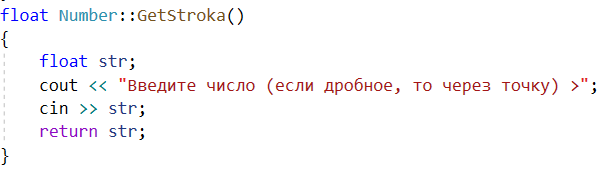
int Por = 0;

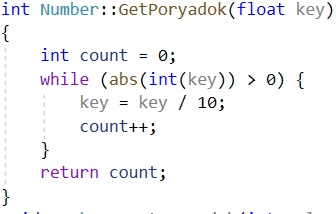
float mantissa = 0;

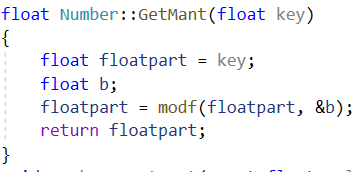
int method;

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

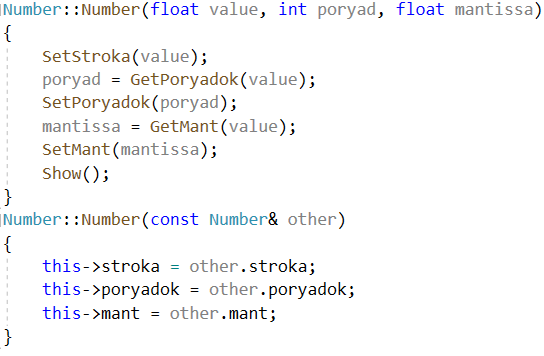
**3.1.** Данные вводятся через объекты класса Number через гетторы.





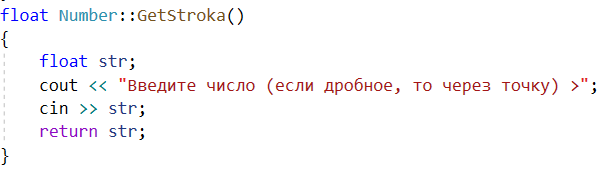


**3.2.** Данные также вводятся через объекты класса Number через конструктор с параметрами и конструктор копирования.

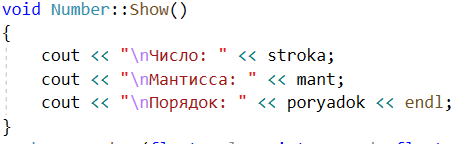


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода числа используется геттор (метод, описанный в классе для ввода), в котором используется функция cin.

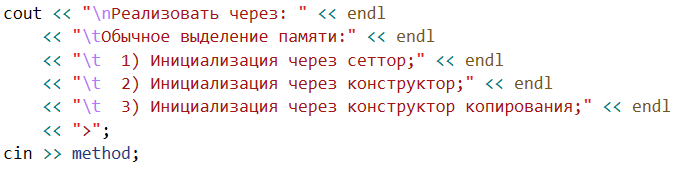


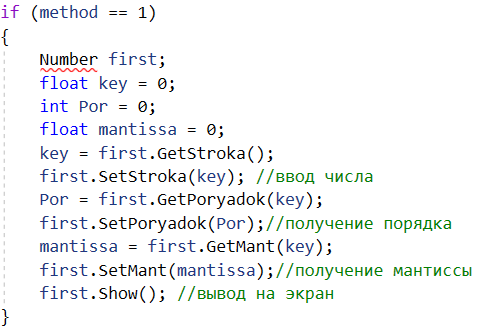
**4.2.** Для вывода данных на консоль используется метод Show().

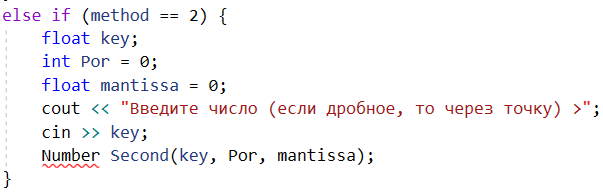


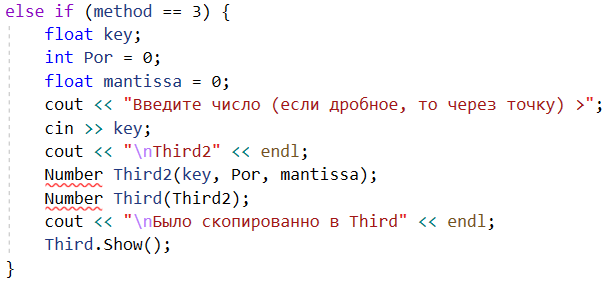
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции main() через if() пользователь выбирает, какми методом выполнить поставленную задачу.

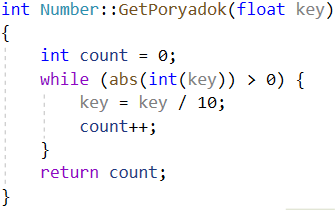




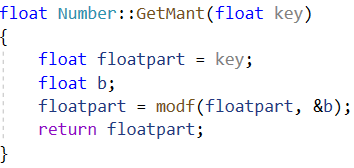




**5.2.** Метод GetPoryadok() класса Number cчитает порядок в цикле while(), в котором число целочисленно делится на 10 и с каждым заходом в цикл увеличивается счётчик count.



**5.3.** Метод GetMant() класса Number cчитает мантиссу через функцию modf, которая записывает в переменную floatpart дробную часть введённого числа, а в параметр b записывает целую часть числа.



**Код**

Header.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Kvitantsiya

{

int Number;

string Date;

double Summ;

public:

Kvitantsiya();

Kvitantsiya(int Number, string Date, double Summ);

Kvitantsiya(const Kvitantsiya& p);

~Kvitantsiya();

int Get\_Number();

string Get\_Date();

double Get\_Summ();

void Set\_Number(int N);

void Set\_Date(int D);

void Set\_Summ(int S);

void Show();

};

Header.cpp

#include <iostream>

#include <string>

#include "Header.h"

using namespace std;

Kvitantsiya :: Kvitantsiya()

{

Number = 12;

Date = "2 ноября";

Summ = 12.4;

}

Kvitantsiya :: Kvitantsiya(int Number, string Date, double Summ)

{

this->Number = Number;

this->Date = Date;

this->Summ = Summ;

}

Kvitantsiya :: Kvitantsiya(const Kvitantsiya& p)

{

Number = p.Number;

Date = p.Date;

Summ = p.Summ;

}

Kvitantsiya :: ~Kvitantsiya() {};

//Вывод

int Kvitantsiya::Get\_Number()

{

return Number;

}

string Kvitantsiya :: Get\_Date()

{

return Date;

}

double Kvitantsiya :: Get\_Summ()

{

return Summ;

}

// Ввод

void Kvitantsiya :: Set\_Number(int N)

{

Number = N;

}

void Kvitantsiya :: Set\_Date(int D)

{

Date = D;

}

void Kvitantsiya :: Set\_Summ(int S)

{

Summ = S;

}

void Kvitantsiya :: Show()

{

cout << "Номер квитанции: " << this->Number << endl;

cout << "Дата: " << this->Date << endl;

cout << "Сумма (в рублях): " << this->Summ << endl;

}

Main.cpp

#include <iostream>

#include <string>

#include "Header.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

Kvitantsiya t1;

Kvitantsiya t2(10, "13 октября", 1235.5);

Kvitantsiya t3(t2);

cout << "Объект без параметров: " << endl;

t1.Show();

cout << "Объект с параметрами: " << endl;

t2.Show();

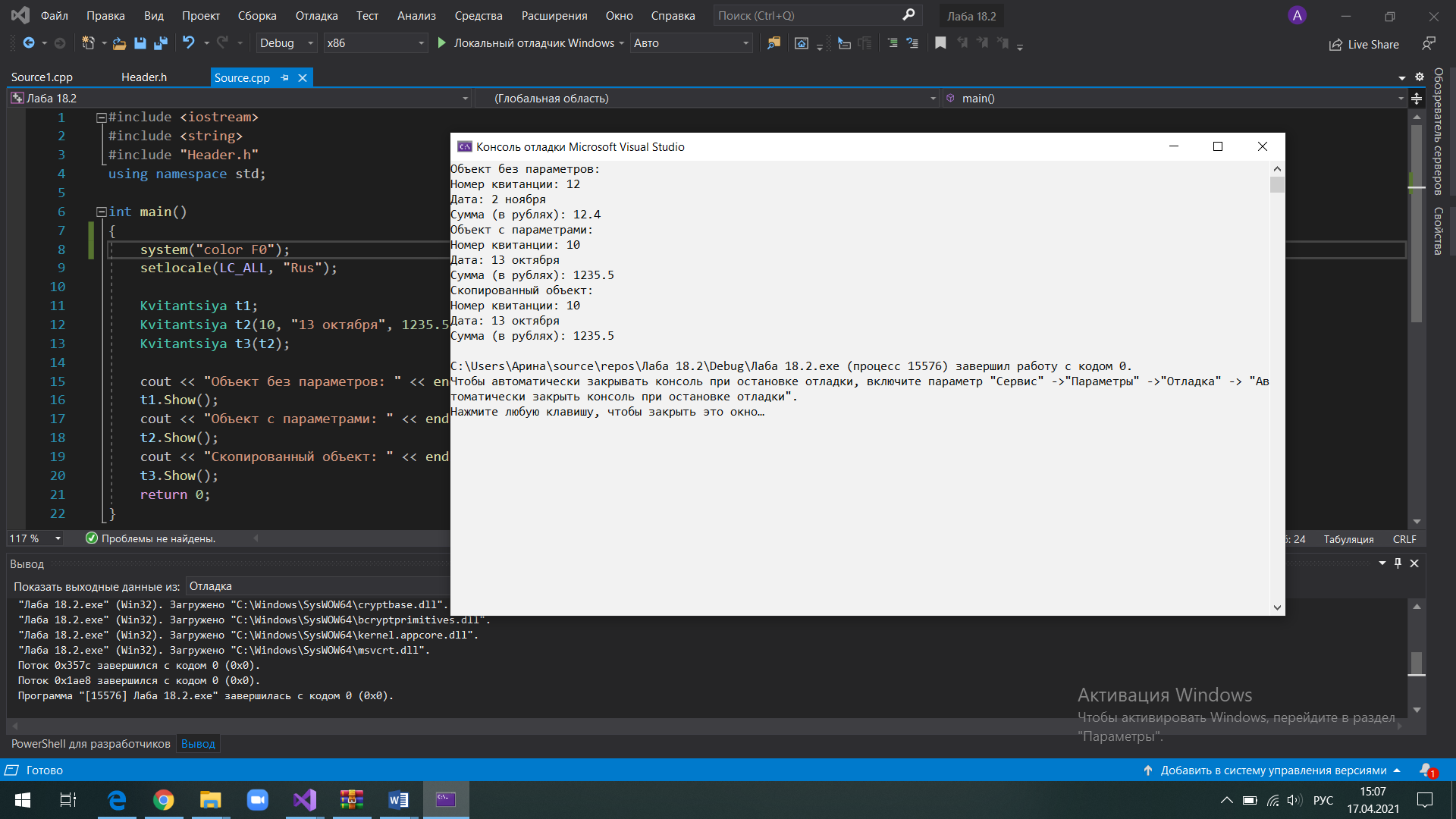
cout << "Скопированный объект: " << endl;

t3.Show();

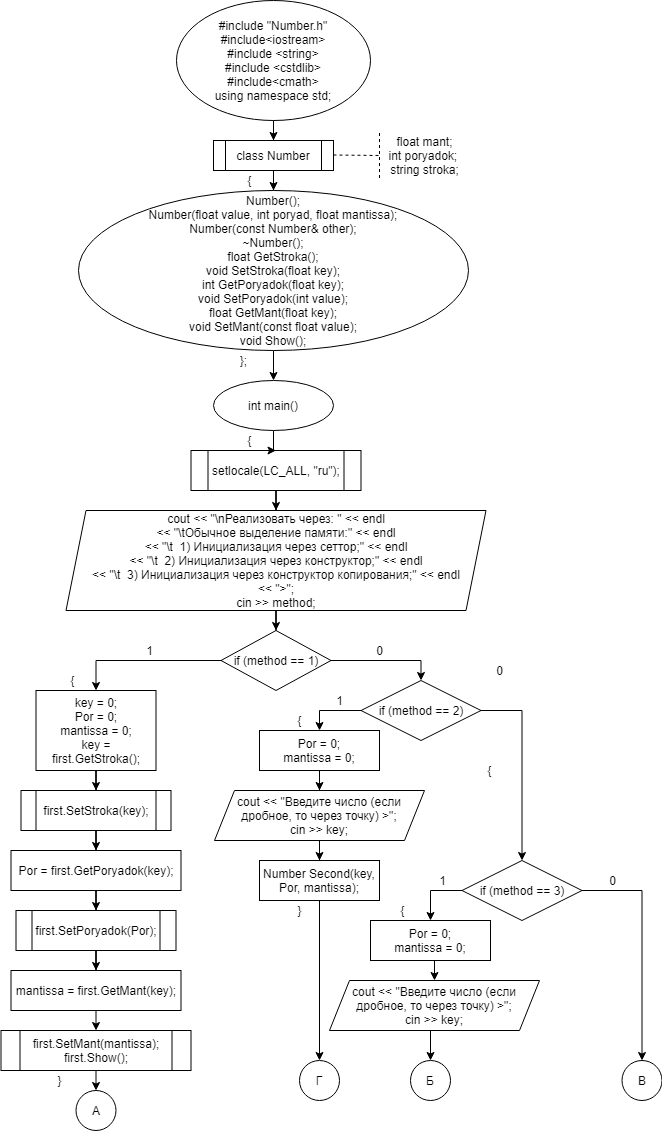
return 0;

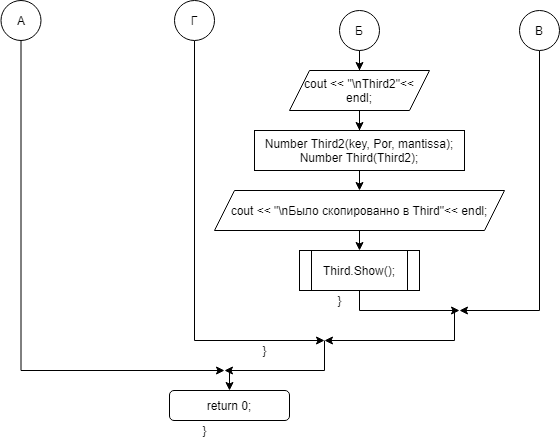
}

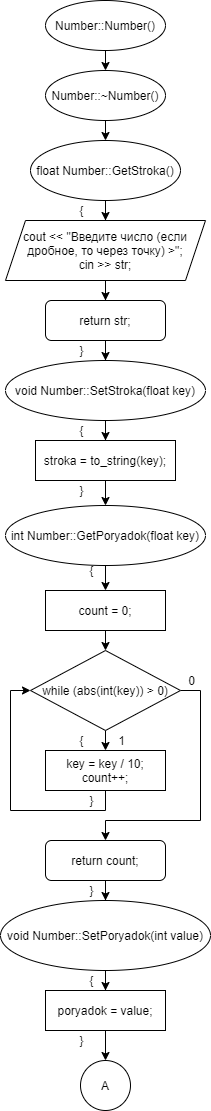
**Выполнение**

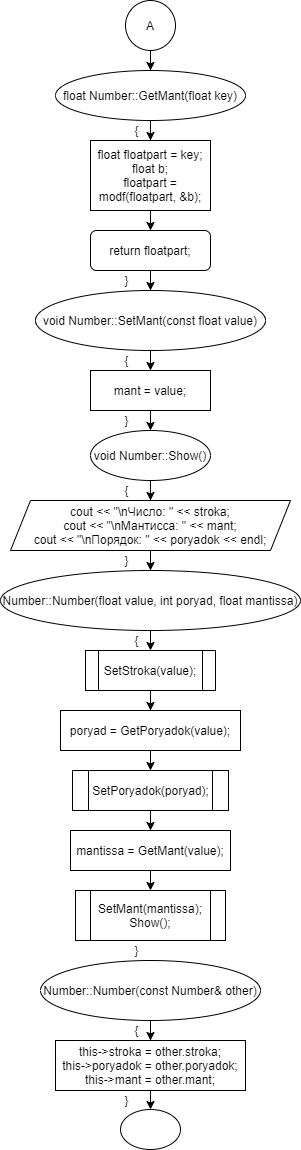


**Блок-схема**

****





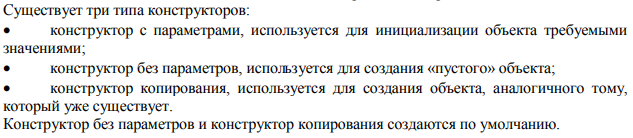


**Контрольные вопросы**



1. Конструктор необходим для инициализации объекта.



2. 

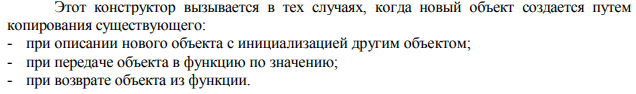


3. Деструктор – это особый вид метода, применяющийся для освобождения ресурсов, выделенных конструктором объекту. Описывать деструктор явным образом требуется в случае, когда объект содержит указатели на память, выделяемую динамически.



4. Конструктор с параметрами используется для инициализации объекта требуемыми значениями. Конструктор без параметров используется для создания «пустого» объекта. Конструктор копирования – это специальный вид конструктора, получающий в качестве единственного параметра указатель на объект этого же класса.

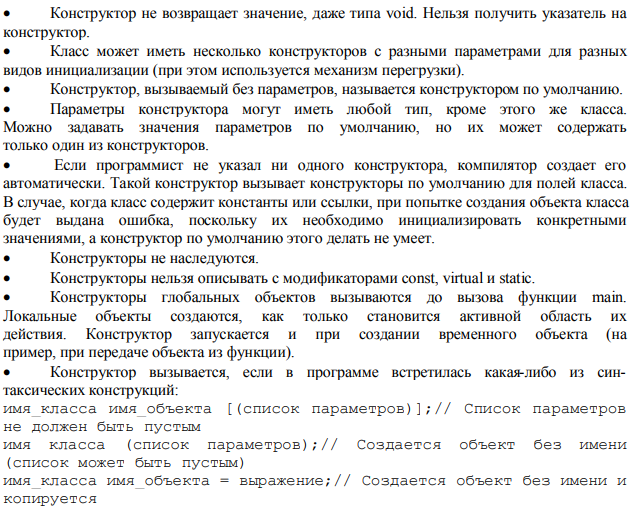


5. 

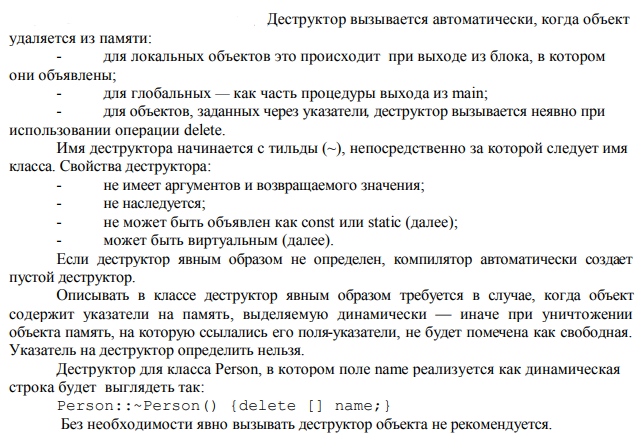


15

6.





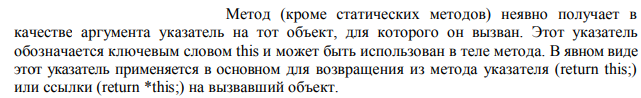
7. 

16

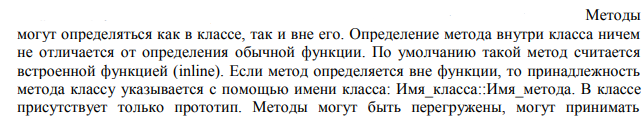


8.



9.



10.



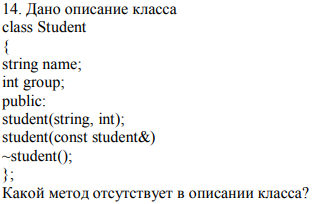
11. Конструктор не возвращает значения.



12. Конструкторы и деструкторы без параметров, если таковые не были прописаны явно.



13. Деструктор не возвращает значения.



14. Отсутствует конструктор без параметров в описании класса.

17



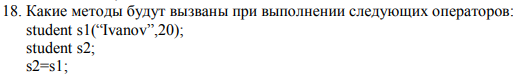
15. Будет вызван конструктор без параметров.



16. Будет вызван конструктор с параметрами.



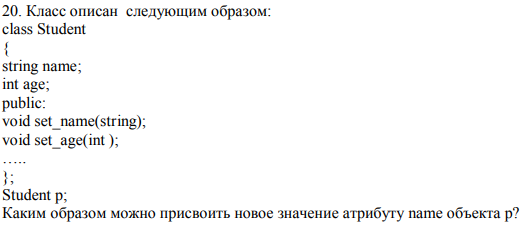
17. Будут вызваны конструктор с параметрами, а затем конструктор копирования.



18. Будут вызваны конструктор с параметрами, затем конструктор без параметров, а затем конструктор копирования.



19. Конструктор без параметров.



20. C помощью метода set\_name().

18