

2 ЛАБОРАТОРНАЯ РАБОТА №2

«ИССЛЕДОВАНИЕ МЕХАНИЗМА ВИРТУАЛЬНЫХ ФУНКЦИЙ»

2.1 Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизмов наследования и виртуальных функций. Освоение особенностей отладки объектно-ориентированных программ.

2.2 Вариант задания – 8

Требуется создать абстрактный базовый класс - число с виртуальной функцией изменения знака числа. Создать производные классы: целое (int), вещественное (double) и комплексное (float, float). Для заданной иерархии описать классы, конструкторы и деструктор, функции ввода и вывода информации на экран. Базовый класс определить как абстрактный, а заданную функцию — как чисто виртуальную в базовом классе и переопределить ее в остальных классах иерархии. Проиллюстрировать корректную работу виртуальных функций и механизма наследования.

2.3 Ход работы

2.3.1 В программе создан абстрактный класс Number в котором содержится виртуальная функция, меняющая знак перед числом. Далее созданы 3 класса – наследника в каждом из которых есть по 2 конструктора, деструктор, функции ввода и вывода числа а также виртуальная функция. В функции main проведены действия с объектами этих трёх классов.

2.3.2 Написана программа на C++ согласно вышеописанного алгоритма.

```
#include <iostream>

#include <windows.h>

using namespace std;

class Number {
```

```

public:

    virtual void change_the_sign() = 0;

};

class Int_Number : public Number {

    int i;

public:

    Int_Number() { i = 0; cout << "Int_Number - конструктор по умолчанию" <<
endl; }

    Int_Number(int num) { i = num; cout << "Int_Number - конструктор с
параметрами" << endl; }

    ~Int_Number() { cout << "Int_Number - деструктор" << endl; system("pause");
}

    void input_int(int num) { i = num; }

    void show_int() { cout << "Целое число: " << i << endl; }

    void change_the_sign() override {

        if (i == 0) return;

        i = -i;

    }

};

class Double_Number : public Number {

    double d;

public:

    Double_Number() { d = 0; cout << "Double_Number - конструктор по умолчанию"
<< endl; }

    Double_Number(double num) { d = num; cout << "Double_Number - конструктор с
параметрами" << endl; }

    ~Double_Number() { cout << "Double_Number - деструктор" << endl; }

    void input_double(double num) { d = num; }

```

```

void show_double() { cout << "Double число: " << d << endl; }

void change_the_sign() override {

    if (d == 0) return;

    d = -d;

}

};

//float real, imaginary;

class Complex_Number : public Number {

    float real, imagin;

public:

    Complex_Number() { real = imagin = 0; cout << "Complex_Number - конструктор
по умолчанию" << endl; }

    Complex_Number(float num1, float num2) { real = num1; imagin = num2; cout
<< "Complex_Number - конструктор с параметрами" << endl; }

    ~Complex_Number() { cout << "Complex_Number - деструктор" << endl; }

    void input_Complex(float num1, float num2) { real = num1; imagin = num2; }

    void show_Complex() { cout << "Complex число: " << real << " + " << imagin
<< "i" << endl; }

    void change_the_sign() override {

        if ((real == 0) && (imagin == 0)) return;

        real = -real; imagin = -imagin;

    }

};

int main()

{

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

```

```
cout << "Здравствуйте!" << endl;

Number *obj;

Int_Number obj1;

obj1.show_int();

obj = &obj1;

int num;

cout << "Введите целое число и мы его сделаем противоположным знаком :)" <<
endl;

cout << " -> "; cin >> num;

obj1.input_int(num);

obj1.show_int();

obj->change_the_sign();

obj1.show_int();

Double_Number obj2;

obj2.show_double();

obj = &obj2;

double num2;

cout << "Введите double число и мы его сделаем противоположным знаком :)"
<< endl;

cout << " -> "; cin >> num2;

obj2.input_double(num2);

obj2.show_double();

obj->change_the_sign(); //изменили знак

obj2.show_double();

Complex_Number obj3;

obj3.show_Complex();

obj = &obj3;
```

```

float num3, num4;

cout << "Введите два числа типа flout; это будет комплексное число, где" <<
endl;

cout << "первое число - реальная часть, а второе число мнимая часть" <<
endl;

cout << " -> "; cin >> num3;

cout << " -> "; cin >> num4;

obj3.input_Complex(num3, num4);

obj3.show_Complex();

obj->change_the_sign(); //изменили знак

obj3.show_Complex();

system("pause");

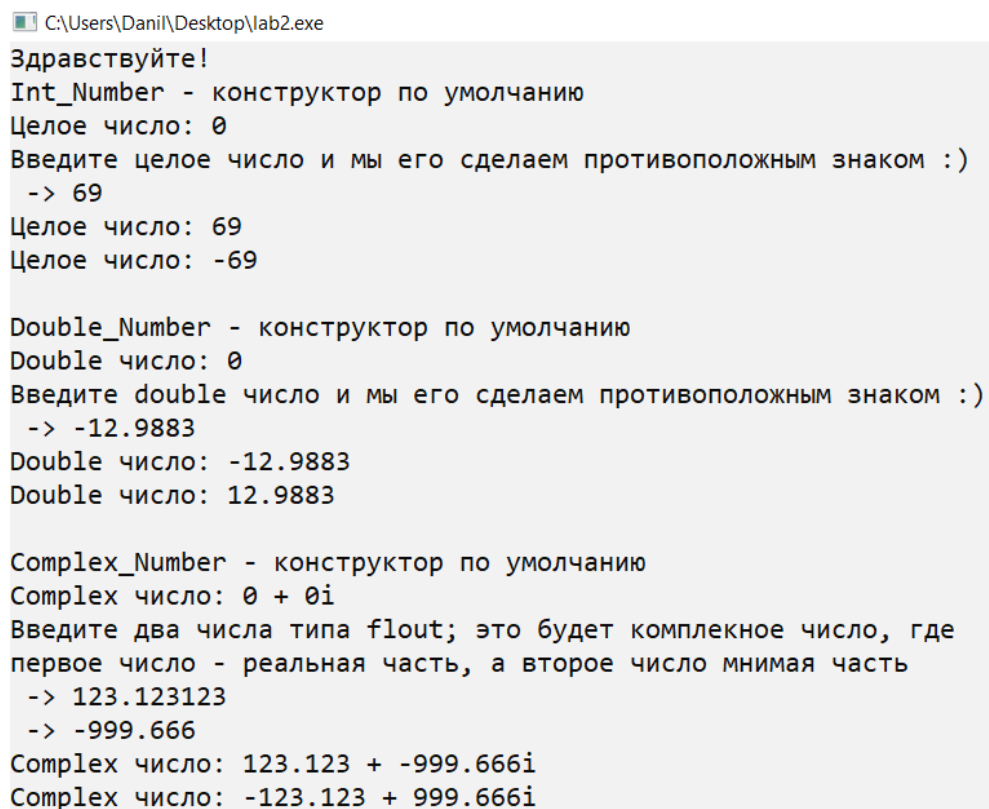
return 0;

}

```

2.3.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунке 2.1. На изображении видно как мы вводим число, и оно выводится с обратным значением.



```

C:\Users\Danil\Desktop\lab2.exe
Здравствуйте!
Int_Number - конструктор по умолчанию
Целое число: 0
Введите целое число и мы его сделаем противоположным знаком :)
-> 69
Целое число: 69
Целое число: -69

Double_Number - конструктор по умолчанию
Double число: 0
Введите double число и мы его сделаем противоположным знаком :)
-> -12.9883
Double число: -12.9883
Double число: 12.9883

Complex_Number - конструктор по умолчанию
Complex число: 0 + 0i
Введите два числа типа flout; это будет комплексное число, где
первое число - реальная часть, а второе число мнимая часть
-> 123.123123
-> -999.666
Complex число: 123.123 + -999.666i
Complex число: -123.123 + 999.666i

```

Рисунок 2.1 – Работа с программой

Результаты тестирования полностью соответствуют ожиданиям.

Выводы

В ходе выполнения данной лабораторной работы были получены навыки разработки программ, использующих виртуальные функции, абстрактные классы. Было изучено такое явление, как полиморфизм. Закреплены навыки разработки и отладки программ, использующих классы и объекты. Полученные во время разработки навыки помогут разрабатывать более сложные программы с использованием классов и объектов, более эффективные по времени выполнения алгоритмы.