

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа
по курсу «ООП»**

**Тема:
Простые классы.**

Студент:	Петрин С.А.
Группа:	М80-207Б-18
Преподаватель:	Журавлев А.А.
Вариант:	17
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

budget_class.h:

```
#ifndef BUDGET_CLASS_H
#define BUDGET_CLASS_H 1

class Budget
{
    double a, b;
public:
    Budget(double first, double second);
    Budget();
    double rounding(double num) const;
    Budget plus(Budget const op2) const;
    Budget assign(Budget const op2);
    Budget minus(Budget const op2) const;
    Budget multiply(Budget const &op2) const;
    Budget divide(Budget const &op2) const;
    bool eq(Budget const &op2) const;
    bool ne(Budget const &op2) const;
    bool lt(Budget const &op2) const;
    bool gt(Budget const &op2) const;
    bool le(Budget const &op2) const;
    bool ge(Budget const &op2) const;
    void write(std::ostream &out);
    void read(std::istream &in);
};

#endif // BUDGET_CLASS_H
```

budget_class.cpp:

```
#include <iostream>
#include "budget_class.h"

Budget::Budget(double first, double second)
{
    a = rounding(first);
    b = rounding(second);
}

Budget::Budget()
{
    a = 0.0;
    b = 0.0;
}
```

```
double Budget::rounding(double num) const
{
    if (num >= 0)
        num = (int) (num * 100 + 0.5) / 100.0;
    else
        num = (int) (num * 100 - 0.5) / 100.0;

    return num;
}
```

```
Budget Budget::plus(Budget const op2) const
{
    Budget temp;

    temp.a = a + op2.a;
    temp.b = b + op2.b;

    return temp;
}
```

```
Budget Budget::assign(Budget const op2)
{
    a = op2.a;
    b = op2.b;

    return *this;
}
```

```
Budget Budget::minus(Budget const op2) const
{
    Budget temp;

    temp.a = a - op2.a;
    temp.b = b - op2.b;

    return temp;
}
```

```
Budget Budget::multiply(Budget const &op2) const
{
    Budget temp;

    temp.a = rounding(a * op2.a);
    temp.b = rounding(b * op2.b);
}
```

```

    return temp;
}

Budget Budget::devide(Budget const &op2) const
{
    Budget temp;

    temp.a = rounding(a / op2.a);
    temp.b = rounding(b / op2.b);

    return temp;
}

bool Budget::eq(Budget const &op2) const
{
    return ((a == op2.a) && (b == op2.b));
}

bool Budget::ne(Budget const &op2) const
{
    return ((a != op2.a) || (b != op2.b));
}

bool Budget::lt(Budget const &op2) const
{
    if (a < op2.a)
    {
        return true;
    }
    else if (a == op2.a)
    {
        return b < op2.b;
    }
    else
    {
        return false;
    }
}

bool Budget::gt(Budget const &op2) const
{
    if (a > op2.a)
    {
        return true;
    }
}

```

```

        else if (a == op2.a)
        {
            return b > op2.b;
        }
        else
        {
            return false;
        }
    }
}

```

```

bool Budget::le(Budget const &op2) const
{
    if (a < op2.a)
    {
        return true;
    }
    else if (a == op2.a)
    {
        return b <= op2.b;
    }
    else
    {
        return false;
    }
}

```

```

bool Budget::ge(Budget const &op2) const
{
    if (a > op2.a)
    {
        return true;
    }
    else if (a == op2.a)
    {
        return b >= op2.b;
    }
    else
    {
        return false;
    }
}

```

```

void Budget::write(std::ostream &out)
{
    out.precision(2);
}

```

```

        out.setf(ios::fixed);
        out << a << " " << b << "\n";
        out.unsetf(ios::fixed);
    }

void Budget::read(std::istream &in)
{
    in >> a >> b;
    a = rounding(a);
    b = rounding(b);
}

```

main.cpp:

```

#include <iostream>
#include "budget_class.h"

#define UNUSED(x) (void)x

int main(int argc, char *argv[])
{
    Budget ob1, ob2;

    ob1.read(std::cin);
    ob2.read(std::cin);

    std::cout << "ob1:\n";
    ob1.write(std::cout);
    std::cout << "ob2:\n";
    ob2.write(std::cout);

    std::cout << "Addition:\n";
    ob1.plus(ob2).write(std::cout);

    std::cout << "Subtraction:\n";
    ob1.minus(ob2).write(std::cout);

    std::cout << "Multiplication:\n";
    ob1.multiply(ob2).write(std::cout);

    std::cout << "Division:\n";
    ob1.devide(ob2).write(std::cout);

    std::cout << "ob1 = ob2 ? ";
    if (ob1.eq(ob2))

```

```

        std::cout << "YES\n";
    else std::cout << "NO\n";

    std::cout << "ob1 != ob2 ? ";
    if (ob1.ne(ob2))
        std::cout << "YES\n";
    else std::cout << "NO\n";

    std::cout << "ob1 < ob2 ? ";
    if (ob1.lt(ob2))
        std::cout << "YES\n";
    else std::cout << "NO\n";

    std::cout << "ob1 > ob2 ? ";
    if (ob1.gt(ob2))
        std::cout << "YES\n";
    else std::cout << "NO\n";

    std::cout << "ob1 <= ob2 ? ";
    if (ob1.le(ob2))
        std::cout << "YES\n";
    else std::cout << "NO\n";

    std::cout << "ob1 >= ob2 ? ";
    if (ob1.ge(ob2))
        std::cout << "YES\n";
    else std::cout << "NO\n";

    UNUSED(argc);
    UNUSED(argv);

    return 0;
}

```

CmakeLists.txt:

```
cmake_minimum_required (VERSION 3.2)
```

```
project(lab1)
```

```
add_executable(oop_exercise_01 main.cpp budget_class.cpp)
```

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall")
```

```
set_target_properties(oop_exercise_01 PROPERTIES CXX_STANDART 14
CXX_STANDART_REQUIRED ON)
```

test.sh:

executable=\$1

for file in test_?.test

do

 \$executable < \$file > tmp

 if cmp tmp \${file%%.test}.result

 then

 echo Test "\$file": SUCCESS

 else

 echo Test "\$file": FAIL

 fi

 rm tmp

done

2. Ссылка на репозиторий на GitHub.

https://github.com/SergeiPetrin/OOP/tree/master/oop_exercise_01

3. Набор testcases.

test_00.test:

100.123 100.125

200.12555 200.12333

test_01.test:

0.12976 99.654

883.3123 10.2123

test_02.test:

100.123 100.125

100.124 100.126

test_03.test:

-100.103 -100.105

-300.108 -250.1204

test_04.test:

982.555 -5351.316

982.559 3215.12345

4. Результаты выполнения тестов.

test_00.result:
ob1:
100.12 100.13
ob2:
200.13 200.12
Addition:
300.25 300.25
Subtraction:
-100.01 -99.99
Multiplication:
20037.02 20038.02
Division:
0.50 0.50
ob1 = ob2 ? NO
ob1 != ob2 ? YES
ob1 < ob2 ? YES
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? NO

test_01.result:
ob1:
0.13 99.65
ob2:
883.31 10.21
Addition:
883.44 109.86
Subtraction:
-883.18 89.44
Multiplication:
114.83 1017.43
Division:
0.00 9.76
ob1 = ob2 ? NO
ob1 != ob2 ? YES
ob1 < ob2 ? YES
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? NO

test_02.result:
ob1:
100.12 100.13
ob2:

100.12 100.13

Addition:

200.24 200.26

Subtraction:

0.00 0.00

Multiplication:

10024.01 10026.02

Division:

1.00 1.00

ob1 = ob2 ? YES

ob1 != ob2 ? NO

ob1 < ob2 ? NO

ob1 > ob2 ? NO

ob1 <= ob2 ? YES

ob1 >= ob2 ? YES

test_03.result:

ob1:

-100.10 -100.11

ob2:

-300.11 -250.12

Addition:

-400.21 -350.23

Subtraction:

200.01 150.01

Multiplication:

30041.01 25039.51

Division:

0.33 0.40

ob1 = ob2 ? NO

ob1 != ob2 ? YES

ob1 < ob2 ? NO

ob1 > ob2 ? YES

ob1 <= ob2 ? NO

ob1 >= ob2 ? YES

test_04.result:

ob1:

982.56 -5351.32

ob2:

982.56 3215.12

Addition:

1965.12 -2136.20

Subtraction:

0.00 -8566.44

Multiplication:

965424.15 -17205135.96

Division:

1.00 -1.66

ob1 = ob2 ? NO

ob1 != ob2 ? YES

ob1 < ob2 ? YES

ob1 > ob2 ? NO

ob1 <= ob2 ? YES

ob1 >= ob2 ? NO

5. Объяснение результатов работы программы.

- 1) При запуске скрипта с аргументом `./test.sh ./oop_exercise_01` объекты `ob1`, `ob2` в основной программе получают данные из файлов `test_??.test`. Эти данные округляются до двух знаков после запятой.
- 2) Вывод данных объектов `ob1`, `ob2` в стандартный поток вывода.
- 3) Объекты `ob1` и `ob2` складываются с помощью функции-члена `plus()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 4) Из объекта `ob1` вычитается объект `ob2` с помощью функции-члена `minus()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 5) Объекты `ob1` и `ob2` умножаются с помощью функции-члена `multiply()` класса `Budget`, и результат сначала округляется до двух знаков после запятой, а затем выводится в стандартный поток вывода с помощью функции `write()`.
- 6) Объект `ob1` делится на `ob2` с помощью функции-члена `devide()` класса `Budget`, и результат сначала округляется до двух знаков после запятой, а затем выводится в стандартный поток вывода с помощью функции `write()`.
- 7) Объекты `ob1` и `ob2` проверяются на равенство с помощью функции-члена `eq()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 8) Объекты `ob1` и `ob2` проверяются на неравенство с помощью функции-члена `ne()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 9) Производится проверка на то, что `ob1` меньше `ob2`, с помощью функции-члена `lt()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 10) Производится проверка на то, что `ob1` больше `ob2`, с помощью функции-члена `gt()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 11) Производится проверка на то, что `ob1` меньше или равен `ob2`, с помощью функции-члена `le()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.
- 12) Производится проверка на то, что `ob1` больше или равен `ob2`, с помощью функции-члена `ge()` класса `Budget`, и результат выводится в стандартный поток вывода с помощью функции `write()`.

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий GitHub, а также изучил основы работы с классами в C++. Создал класс, соответствующий варианту моего задания, реализовал для него арифметические операции сложения, вычитания, умножения, деления, а также операции сравнения.