Московский Авиационный Институт
(Национальный исследовательский Университет)

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

Лабораторная работа
по курсу «ООП»

Тема:
Операторы, литералы.

Студент:	Петрин С.А.
Группа:	М80-207Б-18
Преподаватель:	Чернышов Л.Н.
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) : a(first), b(first) {};
    Budget() : a(0), b(0) {};
    Budget operator+(Budget const op2) const;
    Budget operator=(Budget const op2);
    Budget operator-(Budget const op2) const;
    Budget operator*(double const &num) const;
    Budget operator/(double const &num) const;
    bool operator==(Budget const &op2) const;
    bool operator!=(Budget const &op2) const;
    bool operator<(Budget const &op2) const;
    bool operator>(Budget const &op2) const;
    bool operator<=(Budget const &op2) const;
    bool operator>=(Budget const &op2) const;
    friend std::ostream &operator<<(std::ostream &out, Budget const &ob);
    friend std::istream &operator>>(std::istream &in, Budget &ob);
    friend double rounding(double num);
};

Budget operator"" _Budget(long double);

#endif //BUDGET_CLASS_H
```

budget_class.cpp:

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

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

   return num;
}

Budget Budget::operator+(Budget const op2) const
{
    Budget temp;
```

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

    return temp;
}

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

    return *this;
}

Budget Budget::operator-(Budget const op2) const
{
    Budget temp;

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

    return temp;
}

Budget Budget::operator*(double const &num) const
{
    Budget temp;

    temp.a = rounding(a * num);
    temp.b = rounding(b * num);

    return temp;
}

Budget Budget::operator/(double const &num) const
{
    Budget temp;

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

    return temp;
}
```

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

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

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

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

bool Budget::operator<=(Budget const &op2) const
{
    if (a < op2.a)
```

```
    {
        return true;
    }
    else if (a == op2.a)
    {
        return b <= op2.b;
    }
    else
    {
        return false;
    }
}

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

Budget operator"" _Budget(long double op)
{
    Budget temp{(double) op, (double) op};

    return temp;
}

std::ostream &operator<<(std::ostream &out, Budget const &ob)
{
    out.precision(2);
    out.setf(std::ios::fixed);
    out << ob.a << " " << ob.b << "\n";
    out.unsetf(std::ios::fixed);

    return out;
}
```

```
std::istream &operator>>(std::istream &in, Budget &ob)
{
    in >> ob.a >> ob.b;
    ob.a = rounding(ob.a);
    ob.b = rounding(ob.b);

    return in;
}
```

main.cpp:

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

#define UNUSED(x) (void)x

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

    std::cin >> ob1 >> ob2;

    std::cout << "ob1:\n" << ob1 << "ob2:\n" << ob2;

    std::cout << "ob1 + ob2:\n" << ob1 + ob2;

    std::cout << "ob1 - ob2\n" << ob1 - ob2;

    double n{};
    std::cin >> n;
    std::cout << "ob1 * " << std::fixed << n << ":\n" << ob1 * n;
    std::cin >> n;
    std::cout << "ob2 * " << std::fixed << n << ":\n" << ob2 * n;

    std::cin >> n;
    std::cout << "ob1 / " << std::fixed << n << ":\n" << ob1 / n;
    std::cin >> n;
    std::cout << "ob2 / " << std::fixed << n << ":\n" << ob2 / n;

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

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

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

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

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

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

    Budget ob3 = 4.5_Budget;
    std::cout << ob3;

    UNUSED(argc);
    UNUSED(argv);

    return 0;
}
```

CmakeLists.txt:

```
cmake_minimum_required (VERSION 3.2)

project(lab2)

add_executable(oop_exercise_02 main.cpp budget_class.cpp)

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

set_target_properties(oop_exercise_02 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_02

3. Набор testcases.

test_00.test:
```
100.123 100.125
200.12555 200.12333
2 3
10 20
```

test_01.test:
```
0.12976 99.654
883.3123 10.2123
-30 100
0.100 10.2148888
```

test_02.test:
```
100.123 100.125
100.124 100.126
-0.747 -2.501
100.12 -100.13
```

test_03.test:
```
-100.103 -100.105
-300.108 -250.1204
99.999 743.214
943.1 250.00
```

test_04.test:
```
982.555 -5351.316
982.559 3215.12345
0 33.333
10.00 25.00
```

4. Результаты выполнения тестов.

test_00.result:
ob1:
100.12 100.13
ob2:
200.13 200.12
ob1 + ob2:
300.25 300.25
ob1 - ob2
-100.01 -99.99
ob1 * 2.00:
200.24 200.26
ob2 * 3.00:
600.39 600.36
ob1 / 10.00:
10.01 10.01
ob2 / 20.00:
10.01 10.01
ob1 = ob2 ? NO
ob1 != ob2 ? YES
ob1 < ob2 ? YES
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? NO
ob3:
4.50 4.50

test_01.result:
ob1:
0.13 99.65
ob2:
883.31 10.21
ob1 + ob2:
883.44 109.86
ob1 - ob2
-883.18 89.44
ob1 * -30.00:
-3.90 -2989.50
ob2 * 100.00:
88331.00 1021.00
ob1 / 0.10:
1.30 996.50
ob2 / 10.21:
86.47 1.00
ob1 = ob2 ? NO

ob1 != ob2 ? YES
ob1 < ob2 ? YES
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? NO
ob3:
4.50 4.50

test_02.result:
ob1:
100.12 100.13
ob2:
100.12 100.13
ob1 + ob2:
200.24 200.26
ob1 - ob2
0.00 0.00
ob1 * -0.75:
-74.79 -74.80
ob2 * -2.50:
-250.40 -250.43
ob1 / 100.12:
1.00 1.00
ob2 / -100.13:
-1.00 -1.00
ob1 = ob2 ? YES
ob1 != ob2 ? NO
ob1 < ob2 ? NO
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? YES
ob3:
4.50 4.50

test_03.result:
ob1:
-100.10 -100.11
ob2:
-300.11 -250.12
ob1 + ob2:
-400.21 -350.23
ob1 - ob2
200.01 150.01
ob1 * 100.00:
-10009.90 -10010.90

ob2 * 743.21:
-223045.95 -185892.69
ob1 / 943.10:
-0.11 -0.11
ob2 / 250.00:
-1.20 -1.00
ob1 = ob2 ? NO
ob1 != ob2 ? YES
ob1 < ob2 ? NO
ob1 > ob2 ? YES
ob1 <= ob2 ? NO
ob1 >= ob2 ? YES
ob3:
4.50 4.50

test_04.result:
ob1:
982.56 -5351.32
ob2:
982.56 3215.12
ob1 + ob2:
1965.12 -2136.20
ob1 - ob2
0.00 -8566.44
ob1 * 0.00:
0.00 0.00
ob2 * 33.33:
32751.67 107169.59
ob1 / 10.00:
98.26 -535.13
ob2 / 25.00:
39.30 128.60
ob1 = ob2 ? NO
ob1 != ob2 ? YES
ob1 < ob2 ? YES
ob1 > ob2 ? NO
ob1 <= ob2 ? YES
ob1 >= ob2 ? NO
ob3:
4.50 4.50

5. Объяснение результатов работы программы.

1) Считываются данные для объекта с помощью перегрузки оператора ввода «>>». В данном методе считываемые числа с плавающей точкой округляются до двух знаков после точки, и возвращается ссылка на поток ввода.
2) Благодаря перегрузке оператора вывода «<<» выводятся сначала данные объекта ob1, а затем объекта ob2.
3) Выводится сумма объекта ob1 и ob2.
Перегрузка оператора «+»:

- Создается временный объект temp класса Budget.
- Поле a первого объекта складывается с полем a второго объекта и результат сохраняется в поле a объекта temp.
- Аналогично с полем b.
- Возвращается объект temp.

4) Выводится разность объекта ob1 и ob2.
Перегрузка оператора «-»:

- Создается временный объект temp класса Budget.
- Из поля a первого объекта вычитается с поле a второго объекта и результат сохраняется в поле a объекта temp.
- Аналогично с полем b.
- Возвращается объект temp.

5) Создается переменная n для умножения на объекты ob1, ob2 и деления объектов ob1, ob2 на это число.
6) Считывается число n для умножения на ob1.
7) Выводится произведение числа n и объект ob1.
Перегрузка оператора «*»:

- Создается временный объект temp класса Budget.
- В поле a временного объекта temp сохраняется значение произведения поля a вызывающего объекта и числа n, округленное до 2 знаков после точки.
- Аналогично с полем b.
- Возвращается объект temp.

8) Считывается число n для умножения на ob2.
9) Выводится произведение числа n и объекта ob2.
10) Считывается число n для того, чтобы разделить ob1 на n.
11) Выводится частное от деления ob1 на n.
Перегрузка оператора «/»:

- Создается временный объект temp класса Budget.
- В поле a временного объекта temp сохраняется результат деления поля a вызывающего объекта на число n, округленное до 2 знаков после точки.
- Аналогично с полем b.
- Возвращается объект temp.

12) Выводится либо «YES», либо «NO» в зависимости оттого, что равны ли два объекта или нет. Объекты равны, когда равны и поля a, и поля b.

13) Выводится либо «YES», либо «NO» в зависимости оттого, что не равны ли два объекта или нет. Объекты не равны, когда хотя бы одно из полей первого объекта не равно соответствующему полю другого.
14) Выводится либо «YES», либо «NO» в зависимости оттого, меньше ли первый объект второго. Первый объект меньше второго, если:

- Если поле a первого объекта меньше соответствующего поля второго объекта.
- Если поля a равны, а поле b первого объекта меньше соответствующего поля второго объекта.

15) Выводится либо «YES», либо «NO» в зависимости оттого, больше ли первый объект второго. Первый объект больше второго, если:

- Если поле a первого объекта больше соответствующего поля второго объекта.
- Если поля a равны, а поле b первого объекта больше соответствующего поля второго объекта.

16) Выводится либо «YES», либо «NO» в зависимости оттого, меньше или равен ли первый объект второму. Первый объект меньше или равно второго, если:

- Если поле a первого объекта меньше соответствующего поля второго объекта.
- Если поля a равны, а поле b первого объекта меньше или равно соответствующему поля второго объекта.

17) Выводится либо «YES», либо «NO» в зависимости оттого, больше или равен ли первый объект второму. Первый объект больше второго, если:

- Если поле a первого объекта больше соответствующего поля второго объекта.
- Если поля a равны, а поле b первого объекта больше или равно соответствующему поля второго объекта.

18) Выводится объект ob3, значения полей которого инициализируются с помощью пользовательского литерала _Budget.

6. Вывод.

Перегрузка операторов нужна для удобной работы с объектами, чтобы можно было совершать с классами те же операции, что и со стандартными типами данных.
Механизм пользовательских литералов — это полезный в некоторых случаях инструмент. Использовать его где попало не стоит, т. к. они могут как повысить читаемость кода, так и понизить.