Московский авиационный институт
(Национальный исследовательский университет)

Институт: 8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 1

Тема: Простые классы на языке C++

Студент: Моисеенков Илья Павлович
Группа: 80-208
Преподаватель: Чернышов Л.Н.
Дата: 28.09.2020
Оценка: 13/15

Москва, 2020

1. Постановка задачи

a. Ознакомиться с теоретическим материалом по классам в C++
b. Создать класс **Budget** для работы с бюджетом. Класс состоит из двух вещественных чисел (a,b). Где a – собственная часть средств бюджета в рублях, b – заемная часть средств бюджета рублях. Оба числа должны округляться до второго знака после запятой. Реализовать арифметические операции сложения, вычитания, умножения и деления, а также операции сравнения.
c. Настроить CMake файл для сборки программы
d. Продумать варианты тестирования работы методов класса и подготовить тестовые данные
e. Отправить файлы лабораторной работы в репозиторий Github
f. Подготовить отчет о лабораторной работе

2. Описание программы

Программа имеет многофайловую структуру: описание класса и реализация его методов выделены в отдельные файлы.

Budget.h

- Заголовочный файл - описание класса для работы с бюджетом.
- Содержит приватные атрибуты owned и protected для обозначения собственной и заемной части бюджета соответственно.
- Публичные методы: конструкторы, операторы сравнения и арифметические операторы.
- Присутствует приватный метод round_data для округления размера бюджета до 2 знаков после запятой.

Budget.cpp

- Реализация всех методов из Budget.h.
- Реализация операций сравнения основана на идее, что “Бюджеты сортируются по возрастанию собственной части и по возрастанию заемной части”.
- Арифметические операции с бюджетом выполняются отдельно для собственных частей и для заемных. Например, если мы сложим два бюджета, то в результате получим бюджет, собственная часть которого равна сумме собственных частей, а заемная часть - сумме заемных частей. Аналогично с остальными бинарными операциями.
- Для удобства реализованы операторы типа +=

- Реализовано 2 типа конструктора: первый принимает размер собственной и заемной частей бюджета, второй устанавливает значения по умолчанию (нули)
- Реализованы методы get_owned(), get_borrowed() и get_total() для доступа к приватным атрибутом вне класса
- Метод round_data() реализован на основе функции round из библиотеки cmath

main.cpp

- Предназначена для тестирования работоспособности класса
- При вызове предлагает пользователю выбрать файл с тестовыми данными, либо пропустить ввод и ввести данные вручную
- Необходимо ввести данные для двух объектов класса Budget
- Функция демонстрирует работу методов округления, операций сравнения объектов и арифметические операции

3. Набор тестов

На вход подается два набора чисел: собственная и заёмная часть первого и второго бюджета.

test1.txt

```
55.569 100.11111
20 98.3
```

test2.txt

```
0.01 52.345
1000.001 100.23812
```

test3.txt

```
300 500
300 500
```

4. Результаты выполнения тестов

test1

```
Let's create 2 objects for demonstration of class methods.
```

Type in the name of file with data you want to use (e.g test1.txt) or type 'skip' to enter them yourself:test1.txt

So we have budget a: 55.57 & 100.11

Total budget a: 155.68

And budget b: 20 & 98.3

Total budget b: 118.3

a is not equal to b: a is greater than b

Sum of budgets is 75.57 & 198.41

Difference between budgets is 35.57 & 1.81

Product of budgets is 1111.4 & 9840.81

Quotient of budgets is 2.78 & 1.02

We can multiply the initial budget a by 2

We'll get 111.14 & 200.22

And let's divide budget b by 3

We'll get 6.67 & 32.77

test2

Let's create 2 objects for demonstration of class methods.

Type in the name of file with data you want to use (e.g test1.txt) or type 'skip' to enter them yourself:test2.txt

So we have budget a: 0.01 & 52.35

Total budget a: 52.36

And budget b: 1000 & 100.24

Total budget b: 1100.24

a is not equal to b: a is lower than b

Sum of budgets is 1000.01 & 152.59

Difference between budgets is -999.99 & -47.89

Product of budgets is 10 & 5247.56

Quotient of budgets is 0 & 0.52

We can multiply the initial budget a by 2

We'll get 0.02 & 104.7

And let's divide budget b by 3

We'll get 333.33 & 33.41

test3

Let's create 2 objects for demonstration of class methods.

Type in the name of file with data you want to use (e.g test1.txt) or type 'skip' to enter them yourself:test3.txt

So we have budget a: 300 & 500

Total budget a: 800

And budget b: 300 & 500

Total budget b: 800

a is equal to b

Sum of budgets is 600 & 1000

Difference between budgets is 0 & 0

Product of budgets is 90000 & 250000

Quotient of budgets is 1 & 1

We can multiply the initial budget a by 2

We'll get 600 & 1000

And let's divide budget b by 3

We'll get 100 & 166.67

5. Листинг программы

Budget.h

```
#ifndef OOP_LAB1_BUDGET_H
#define OOP_LAB1_BUDGET_H


class Budget {
public:
    Budget();
    Budget(double, double);

    double get_owned() const;

    double get_borrowed() const;

    double get_total() const;

    Budget& operator=(Budget rhs);

    bool operator==(Budget& rhs) const;

    bool operator!=(Budget& rhs) const;

    bool operator<(Budget& rhs) const;

    bool operator<=(Budget& rhs) const;

    bool operator>(Budget& rhs);

    bool operator>=(Budget& rhs);

    Budget operator+(Budget& rhs) const;

    Budget operator+=(Budget& rhs);

    Budget operator-(Budget& rhs) const;

    Budget operator-=(Budget& rhs);

    Budget operator*(Budget& rhs) const;

    Budget operator*(double rhs) const;
```

```
    Budget operator*=(Budget& rhs);

    Budget operator/(Budget& rhs) const;

    Budget operator/(double rhs) const;

    Budget operator/=(Budget& rhs);

private:
    double owned; // owned part of budget
    double borrowed; // borrowed part of budget

    // rounds to 2 decimal places
    void round_data();
};


#endif //OOP_LAB1_BUDGET_H
```

Budget.cpp

```
#include <cmath>
#include "Budget.h"

Budget::Budget() : owned(0), borrowed(0) {}

Budget::Budget(double owned_, double borrowed_) : owned(owned_),
borrowed(borrowed_) {
    round_data();
}

double Budget::get_owned() const {
    return owned;
}

double Budget::get_borrowed() const {
    return borrowed;
}

double Budget::get_total() const {
    return owned + borrowed;
}

Budget& Budget::operator=(Budget rhs) {
    owned = rhs.owned;
    borrowed = rhs.borrowed;
    return *this;
}

bool Budget::operator==(Budget &rhs) const {
    return owned == rhs.owned && borrowed == rhs.borrowed;
}
```

```
bool Budget::operator!=(Budget &rhs) const {
    return !(*this == rhs);
}

bool Budget::operator<(Budget &rhs) const {
    if (owned == rhs.owned) {
        return borrowed < rhs.borrowed;
    }
    return owned < rhs.owned;
}

bool Budget::operator<=(Budget &rhs) const {
    return (*this < rhs) || (*this == rhs);
}

bool Budget::operator>(Budget &rhs) {
    return rhs < *this;
}

bool Budget::operator>=(Budget &rhs) {
    return rhs <= *this;
}

Budget Budget::operator+(Budget &rhs) const {
    return Budget(owned + rhs.owned, borrowed + rhs.borrowed);
}

Budget Budget::operator+=(Budget &rhs) {
    owned += rhs.owned;
    borrowed += rhs.borrowed;
    return *this;
}

Budget Budget::operator-(Budget &rhs) const {
    return Budget(owned - rhs.owned, borrowed - rhs.borrowed);
}

Budget Budget::operator-=(Budget &rhs) {
    owned -= rhs.owned;
    borrowed -= rhs.borrowed;
    return *this;
}

Budget Budget::operator*(Budget &rhs) const {
    return Budget(owned * rhs.owned, borrowed * rhs.borrowed);
}

Budget Budget::operator*(double rhs) const {
    return Budget(owned * rhs, borrowed * rhs);
}

Budget Budget::operator*=(Budget &rhs) {
    owned *= rhs.owned;
    borrowed *= rhs.borrowed;
    round_data();
```

```
    return *this;
}

Budget Budget::operator/(Budget &rhs) const {
    return Budget(owned / rhs.owned, borrowed / rhs.borrowed);
}

Budget Budget::operator/(double rhs) const {
    return Budget(owned / rhs, borrowed / rhs);
}

Budget Budget::operator/=(Budget &rhs) {
    owned /= rhs.owned;
    borrowed /= rhs.borrowed;
    round_data();
    return *this;
}

// round to 2 decimal places
void Budget::round_data() {
    owned = round(owned * 100) / 100;
    borrowed = round(borrowed * 100) / 100;
}
```

main.cpp

```
/* Моисеенков Илья М8О-208Б-19
 *
 * github: mosikk
 *
 * Создать класс Budget для работы с бюджетом. Класс состоит из двух
вещественных чисел (a,b).
 * Где a – собственная часть средств бюджета в рублях, b – заемная
часть средств бюджета рублях.
 * Оба числа должны округляться до второго знака после запятой.
 * Реализовать арифметические операции сложения, вычитания,
умножения и деления,
 * а также операции сравнения.
 */


#include <iostream>
#include <fstream>
#include "Budget.h"

int main() {
    std::cout << "Let's create 2 objects for demonstration of class
methods." << std::endl;
    std::string files_name;
    std::cout << "Type in the name of file with data you want to use
";
```

```
    std::cout << "(e.g test1.txt) or type 'skip' to enter them
yourself: ";
    std::cin >> files_name;
    double own_a, borrow_a, own_b, borrow_b;
    if (files_name != "skip") {
        std::ifstream input;
        input.open(files_name);
        input >> own_a >> borrow_a >> own_b >> borrow_b;
    }
    else {
        std::cout << "Enter owned and borrowed parts for budget 'a':
";
        std::cin >> own_a >> borrow_a;
        std::cout << "Enter owned and borrowed parts for budget 'b':
";
        std::cin >> own_b >> borrow_b;
        std::cout << std::endl;
    }
    std::cout << std::endl;

    Budget a(own_a, borrow_a);
    Budget b(own_b, borrow_b);
    std::cout << "So we have budget a: " << a.get_owned() << " & "
<< a.get_borrowed() << std::endl;
    std::cout << "Total budget a: " << a.get_total() << std::endl;
    std::cout << "And budget b: " << b.get_owned() << " & " <<
b.get_borrowed() << std::endl;
    std::cout << "Total budget b: " << b.get_total() << std::endl <<
std::endl;

    if (a != b) {
        std::cout << "a is not equal to b: ";
        if (a > b) {
            std::cout << "a is greater than b" << std::endl;
        }
        else {
            std::cout << "a is lower than b" << std::endl;
        }
    }
    else {
        std::cout << "a is equal to b" << std::endl;
    }
    std::cout << std::endl;

    Budget sum = a + b;
    Budget difference = a - b;
    Budget product = a * b;
    Budget quotient = a / b;
    std::cout << "Sum of budgets is " << sum.get_owned() << " & " <<
sum.get_borrowed() << std::endl;
    std::cout << "Difference between budgets is " <<
difference.get_owned() << " & ";
    std::cout << difference.get_borrowed() << std::endl;
    std::cout << "Product of budgets is " << product.get_owned() <<
" & " << product.get_borrowed() << std::endl;
```

```
    std::cout << "Quotient of budgets is " << quotient.get_owned()
<< " & " << quotient.get_borrowed();
    std::cout << std::endl << std::endl;

    std::cout << "We can multiply the initial budget a by 2" <<
std::endl;
    a = a * 2;
    std::cout << "We'll get " << a.get_owned() << " & " <<
a.get_borrowed() << std::endl;
    std::cout << "And let's divide budget b by 3" << std::endl;
    b = b / 3;
    std::cout << "We'll get " << b.get_owned() << " & " <<
b.get_borrowed() << std::endl << std::endl;
}
```

6. Выводы

Выполнив данную лабораторную работу, я изучил основы и принципы построения классов на языке C++ на примере реализации простого класса “Budget”. Я научился реализовывать методы для работы с классами и перегружать операции. После выполнения лабораторной работы я убедился в том, что применять идеи ООП полезно и удобно при проектировании программ.

7. Список используемых источников

1. Руководство по языку C++ [Электронный ресурс]. - URL: https://www.cplusplus.com/.
2. Построение классов C++ [Электронный ресурс]. - URL: http://cppstudio.com/post/439/