

**Московский авиационный институт  
(Национальный исследовательский университет)**

Факультет: «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: Семенов Илья

Преподаватель: Журавлев А.А.

Дата:

Оценка:

Москва, 2019

## 1. Постановка задачи

Создать класс Bottle для работы с емкостями. Класс должен состоять из двух вещественных чисел: a – объем емкости в литрах и b – процент наполнения емкости (0 – пустая, 1 – полная). Реализовать операции сложения и вычитания, а также сравнения объектов класса бутылка. При сложении должен складываться фактический объем заполнения бутылок

## 2. Репозиторий github

[https://github.com/ilya89099/oop\\_exercise\\_01/](https://github.com/ilya89099/oop_exercise_01/)

## 3. Описание программы

Реализован класс Bottle, в котором хранятся две переменные, отображающие объем и процент заполнения. Написаны Get функции для получения общего объема, процента заполнения и фактического объема каждой бутылки(GetVolume(), GetFillPercent(), GetFilledVolume()). Также реализованы функции, указанные в задании, для получения суммы и разности(Sum, Subtract), и для сравнения различных объектов класса(Less, More, Equal). Функции Sum и Subtract возвращают вещественное число, которое является суммой/разностью фактических объемов переданных бутылок.

Для удобства пользования создано меню с тремя командами:

- create SIZE FILL\_PERCENT – создает новую бутылку и выводит на экране ее идентификатор.
- compare OPERATION ID1 ID2 – принимает операцию сравнения в виде символа(=, <, >), а также два уникальных идентификатора, выведенных в результате работы команды create. Команда выводит true или false как результат сравнения
- operation OPERATION ID1 ID2 – принимает операцию в формате символа(+,-), два уникальных идентификатора, выведенных командой create. Производит сложение/вычитание соответствующих фактических объемов жидкости в бутылках.

## 4. Набор testcases

Тестовые файлы: test\_01.test, test\_02.test, test\_03.test

### test\_01.test:

```
create 20 0.5
```

```
create 10 1
```

create -5 0.2

create 10 2

create 15 0.3

Проверка правильности конструируемых объектов и корректности обработки ошибок.

### **Результат работы программы**

You created bottle number 1

Bottle size: 20

Bottle fill percent: 0.5

You created bottle number 2

Bottle size: 10

Bottle fill percent: 1

Incorrect parameters

Incorrect parameters

You created bottle number 3

Bottle size: 15

Bottle fill percent: 0.3

### **test\_02.txt:**

create 20 0.5

create 10 0.5

create 20 0.25

compare = 1 2

compare = 2 1

compare = 3 2

compare = 2 3

compare > 1 2

compare > 2 1

compare < 1 2

compare < 2 1

compare a 2 1

compare >= 2 1

Проверка корректности работы операций сравнения для класса Bottle.

### **Результат работы программы**

You created bottle number 1

Bottle size: 20

Bottle fill percent: 0.5

You created bottle number 2

Bottle size: 10

Bottle fill percent: 0.5

You created bottle number 3

Bottle size: 20

Bottle fill percent: 0.25

1 2 = false

2 1 = false

3 2 = true

2 3 = true

1 2 > true

2 1 > false

1 2 < false

2 1 < true

Incorrect parameters

Incorrect parameters

### **test\_03txt:**

create 20 0.5

create 10 0.5

create 7 0.2

operation + 1 2

operation + 2 1

operation - 1 2

operation - 2 1

operation - 3 2

operation - 2 3

Проверка корректности работы операций сложения и вычитания для класса Bottle.

### **Результат работы программы**

You created bottle number 1

Bottle size: 20

Bottle fill percent: 0.5

You created bottle number 2

Bottle size: 10

Bottle fill percent: 0.5

You created bottle number 3

Bottle size: 7

Bottle fill percent: 0.2

1 2 + 15

2 1 + 15

1 2 - 5

2 1 - -5

3 2 - -3.6

2 3 - 3.6

## 5. Результаты выполнения тестов

Все тесты успешно пройдены, программа выдаёт верные результаты.

## 6. Листинг программы

### main.cpp

```
#include <iostream>
#include "bottle.h"
#include <vector>
#include <string>
int main() {
    std::vector<Bottle> bottles;
    std::string command;
    while (std::cin >> command) {
        if (command == "create") {
            double size, percent;
            std::cin >> size >> percent;
            if (size < 0 || percent < 0 || percent > 1) {
                std::cout << "Incorrect parameters\n";
                continue;
            }
            bottles.emplace_back(size, percent);
            std::cout << "You created bottle number " << bottles.size()
<< "\n"
                                << "Bottle size: " << bottles.back().GetVolume() <<
"\n"
                                << "Bottle fill percent: " <<
bottles.back().GetFillPercent() << "\n";
        } else if (command == "compare") {
            std::string compare_string;
            int lhs, rhs;
            std::cin >> compare_string >> lhs >> rhs;
            if (lhs <= 0 || lhs > bottles.size() || rhs <= 0 || rhs >
bottles.size() || compare_string.size() != 1
```

```

        || (compare_string[0] != '=' && compare_string[0] !=
'>' && compare_string[0] != '<')) {
            std::cout << "Incorrect parameters" << "\n";
            continue;
        }
        char compare = compare_string[0];
        std::cout << lhs << " " << rhs << " " << compare << " ";
        lhs--;
        rhs--;
        if (compare == '<') {
            std::cout << std::boolalpha <<
bottles[lhs].Less(bottles[rhs]) << "\n";
        } else if (compare == '=') {
            std::cout << std::boolalpha <<
bottles[lhs].Equal(bottles[rhs]) << "\n";
        } else if (compare == '>') {
            std::cout << std::boolalpha <<
bottles[lhs].More(bottles[rhs]) << "\n";
        }
    } else if (command == "operation") {
        std::string operation_string;
        int lhs, rhs;
        std::cin >> operation_string >> lhs >> rhs;
        if ( lhs <= 0 || lhs > bottles.size() || rhs <= 0 || rhs >
bottles.size() || operation_string.size() != 1
            || (operation_string[0] != '-' && operation_string[0] !
= '+')) {
            std::cout << "Incorrect parameters" << "\n";
            continue;
        }
        char operation = operation_string[0];
        std::cout << lhs << " " << rhs << " " << operation << " ";
        rhs--;
        lhs--;
        if (operation == '+') {
            std::cout << bottles[lhs].Sum(bottles[rhs]) << "\n";
        } else if (operation == '-') {
            std::cout << bottles[lhs].Subtract(bottles[rhs]) << "\n";
        }
    } else if (command == "exit") {
        break;
    } else {
        std::cin.ignore(32767, '\n');
        std::cout << "Unknown command\n";
    }
}
return 0;
}

```

## bottle.h

```
#pragma once
class Bottle {
public:
    Bottle(double volume, double fill_percent);
    double GetVolume() const;
    double GetFillPercent() const;
    double GetFilledVolume() const;
    double Sum(const Bottle& other) const;
    double Substract(const Bottle& other) const;
    bool Less(const Bottle& other) const;
    bool More(const Bottle& other) const;
    bool Equal(const Bottle& other) const;
private:
    double volume_;
    double fill_percent_;
};
```

## bottle.cpp

```
#pragma once

#include "bottle.h"

Bottle::Bottle(double volume, double fill_percent)
: volume_(volume), fill_percent_(fill_percent) {}
double Bottle::GetVolume() const {
    return volume_;
}
double Bottle::GetFillPercent() const {
    return fill_percent_;
}
double Bottle::GetFilledVolume() const {
    return volume_ * fill_percent_;
}
double Bottle::Sum(const Bottle &other) const {
    return GetFilledVolume() + other.GetFilledVolume();
}
double Bottle::Substract(const Bottle& other) const {
    return GetFilledVolume() - other.GetFilledVolume();
}
bool Bottle::Less(const Bottle &other) const {
    return GetFilledVolume() < other.GetFilledVolume();
}
bool Bottle::More(const Bottle &other) const {
    return GetFilledVolume() > other.GetFilledVolume();
}
bool Bottle::Equal(const Bottle &other) const {
    return GetFilledVolume() == other.GetFilledVolume();
}
```



## **7. Вывод**

Реализована программа, включающая в себя простой класс с методами и переменными. Также получены навыки работы с git и stake.

### **Список литературы**

1. Шилдт, Герберт. С++: базовый курс, 3-е изд. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2018. - 624 с. : ил. - Парал. тит. англ.
2. Справочник по языку С++ [Электронный ресурс]. URL: <http://www.cplusplus.com/reference/deque/> (дата обращения: 14.09.2019).