Московский авиационный институт

(Национальный исследовательский университет)

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

Лабораторная работа № 2

Тема: Операторы, литералы

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(с помощью перегрузки операторов), а также сравнения объектов класс бутылка. При сложении должен складываться фактический объем заполнения бутылок. Реализовать пользовательский литерал для работы с объектами класса.

2. **Репозиторий github**

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()). Перегружены операторы сравнения (<,>,==) и операторы суммы и разности(возвращающие число). Создан пользовательский литерал с сигнатурой operator_bottle(const char* str, size_t). Работу литерала неудобно демонстрировать на тестовых примерах, поэтому в начале функции main проверяется работа пользовательского литерала.

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_04.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 <vector>
#include <string>
#include <cassert>
#include <cmath>
#include <limits>
#include "bottle.h"
int main() {
    std::vector<Bottle> bottles;
     std::string command;
    // я не знаю, как тут продемонстрировать пользовательский литерал,
пусть будет что то такое
    Bottle literal_constructed = "22.5,0.56"_bottle;
    assert(std::abs(literal_constructed.GetVolume() - 22.5) <
std::numeric_limits<double>::epsilon());
    assert(std::abs(literal_constructed.GetFillPercent() - 0.56) <
std::numeric_limits<double>::epsilon());
    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] <
bottles[rhs]) << "\n";
            } else if (compare == '=') {
                std::cout << std::boolalpha << (bottles[lhs] ==
bottles[rhs]) << "\n";
            } else if (compare == '>') {
                std::cout << std::boolalpha << (bottles[lhs] >
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] + bottles[rhs] << "\n";
            } else if (operation == '-') {
                std::cout << bottles[lhs] - 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
#include <iostream>
class Bottle {
public:
    Bottle(double volume, double fill_percent = 0);
    double GetVolume() const;
    double GetFillPercent() const;
    double GetFilledVolume() const;
    friend double operator + (const Bottle& lhs, const Bottle& rhs);
    friend double operator - (const Bottle& lhs, const Bottle& rhs);
    friend bool operator == (const Bottle& lhs, const Bottle& rhs);
    friend bool operator > (const Bottle& lhs, const Bottle& rhs);
    friend bool operator < (const Bottle& lhs, const Bottle& rhs);
private:
    double volume_;
    double fill_percent_;
};
Bottle operator""_bottle(const char* str, size_t size);
```

bottle.cpp

```
#include "bottle.h"

#include <exception>
#include <stdexcept>
Bottle::Bottle(double volume, double fill_percent)
: volume_(volume), fill_percent_(fill_percent) {
    if (volume < 0 || fill_percent < 0 || fill_percent > 1) {
        throw std::logic_error("Unacceptable parameters for constructor");
    }
}
double Bottle::GetVolume() const {
    return volume_;
}
double Bottle::GetFillPercent() const {
    return fill_percent_;
```

```
}
double Bottle::GetFilledVolume() const {
    return volume_ * fill_percent_;
}
double operator + (const Bottle& lhs, const Bottle& rhs) {
    return lhs.GetFilledVolume() + rhs.GetFilledVolume();
}
double operator - (const Bottle& lhs, const Bottle& rhs) {
    return lhs.GetFilledVolume() - rhs.GetFilledVolume();
}
bool operator == (const Bottle& lhs, const Bottle& rhs) {
    return lhs.GetFilledVolume() == rhs.GetFilledVolume();
}
bool operator > (const Bottle& lhs, const Bottle& rhs) {
    return lhs.GetFilledVolume() > rhs.GetFilledVolume();
}
bool operator < (const Bottle& lhs, const Bottle& rhs) {
    return lhs.GetFilledVolume() < rhs.GetFilledVolume();
}
Bottle operator""_bottle(const char* str, size_t size) {
    int idx = -1;
    for (int i = 0; i < size; ++i) {
        if (str[i] == ',') {
            idx = i;
        }
    }
    if (idx == -1 || idx == size - 1 || idx == 0) {
    }
    char* first_part = new char[idx];
    char* second_part = new char [size - idx - 1];
    std::copy(str, str + idx, first_part);
    std::copy(str + idx + 1, str + size, second_part);
    return Bottle(std::stod(first_part), std::stod(second_part));
}
```

7. Вывод

Предыдущая лабораторная работа переделана с использованием пользовательского литерала, перегрузок операторов и дружественных функций.

Список литературы

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