Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет: «Информационных технологий и прикладной математики»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

**Тема: Операторы, литералы**

|  |  |
| --- | --- |
| Cтудент: | Королев И.М. |
| Группа: | 8О-208Б |
| Преподаватель: | Чернышов Л.Н. |
| Дата: |  |
| Оценка: |  |

Москва 2020

**Оглавление**

[1. Постановка задачи 3](#_heading=h.1fob9te)

[2. Описание программы 4](#_heading=h.3znysh7)

[3. Тесты и результаты их выполнения 6](#_heading=h.3dy6vkm)

[4. Листинг программы 9](#_heading=h.1t3h5sf)

[5. Выводы 15](#_heading=h.4d34og8)

[Список используемых источников 16](#_heading=h.2s8eyo1)

# 1. Постановка задачи

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

Необходимо реализовать пользовательский литерал для работы с константами типа Bottle.

# 2. Описание программы

Программа выполняет чтение данных из консоли или файла (с помощью cat | ./исполняемый файл) и выполняет заданные действия. При запуске программы появляется меню, в котором будут предложены следующие варианты действий:

1. Выполнить арифметическую операцию (сложение или вычитание) или сравнение двух бутылок по объёму самой бутылки и объёму её наполнения.
2. Выполнить операцию сравнения (больше или меньше).
3. Выполнить завершение программы.

При сложении , бутылок и , выполняется переливание содержимого из бутылки в бутылку . Если в бутылку a переливают объём, который больше, чем объём незаполненной части бутылки, то в неё переливается допустимый объём, и она становится полностью заполненной. После выполнения операции, будут выведены изменённые параметры бутылки .

При вычитании , бутылок и , выполняется переливание содержимого из бутылки в бутылку . Если объём заполненной части бутылки мал по сравнению с пустым объёмом бутылки , то в бутылку переливается всё содержимое бутылки . После выполнения операции, будут выведены изменённые параметры бутылки .

При сравнении бутылок c помощью символа «|» будут выведены результаты сравнения бутылок по их объёму и объёму их наполнения.

При выполнении операции сравнения «>» и «<» будет выведено сообщение о верности данного утверждения.

Также был реализован пользовательский литерал \_bot для работы с константами типа Bottle.

# Переменные, методы и операторы класса Bottle

*double Volume*  – объём бутылки (ёмкости).

*double FillingPer* – процент наполнения ёмкости.

*void GetBottle()* – функция получения данных о ёмкости.

*void PrintBottle()* – функция вывода данных о ёмкости.

*Bottle operator+ (Bottle capacity2)* – оператор сложения двух бутылок. Добавляет к первой бутылке воду из второй.

*Bottle operator- (Bottle capacity2) –* оператор вычитания двух бутылок. Переливает из первой бутылки воду во вторую.

*void operator> (Bottle capacity2)* – оператор сравнения объёмов двух бутылок. Выводит сообщение о верности высказывания.

*void operator< (Bottle capacity2)* - оператор сравнения объёмов двух бутылок. Выводит сообщение о верности высказывания.

# Описание функций

*int DataValidation(Bottle a, Bottle b) –* функция проверки данных на верность. Если объём бутылки меньше нуля или процент заполнения бутылки водой больше 1 или меньше 0, то функция возвращает значение 1. Если всё верно, то функция возвращает значение 0.

*void VolumeComparison(Bottle a, Bottle b)* – функция сравнения статических объёмов двух бутылок.

*Void FullnessComparison(Bottle a, Bottle b)* – функция сравнения бутылок по объёму их заполнения водой.

*void Comparison(Bottle a, Bottle b)* – функция сравнения бутылок по статическому объёму и по объёму их заполнения водой.

*Bottle Operation(Bottle a, Bottle b, char c)* – функция выполнения арифметических операций над ёмкостями.

*Bottle Verification(Bottle a, Bottle b, char c)* – функция выполнения операторов сравнения над бутылками.

*Bottle PrintResult(Bottle a, Bottle b, char c, Bottle result)* – функция вывода результата после выполнения арифметической операции.

*int menu()* – функция вывода диалогового меню и последующей работы с введёнными значениями.

# 3. Тесты и результаты их выполнения

Были проведены тесты, на вводе которых написано действие, которое можно выполнять в программе и объекты типа Bottle с операциями над ними.

Таблица 1 – test\_01.txt и результаты его выполнения

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 10 0.5 + 10 0.5 | 10 0.5 + 10 0.5 = 10 1 |
| 1 11 0.5 - 11 0.5 | 11 0.5 - 11 0.5 = 11 0 |
| 1 20 0.75 + 4 0.25 | 20 0.75 + 4 0.25 = 20 0.8 |
| 1 100 0.1 - 100 0.01 | 100 0.1 - 100 0.01 = 100 0 |
| 1 -100 0.1 - 100 0.01 | Invalid data was entered! |
| 1 18 0.5 | 12 0.75 | Volume1 > Volume2  Fullness1 == Fullness2 |
| 2 15 0.33 > 14 0.9 | The statement is true! |
| 2 -14 0.76 < 78 0.1 | Invalid data was entered! |
| 1 0.25 0.5 - 12 0.6 | 0.25 0.5 - 12 0.6 = 0.25 0 |
| 1 1000 0.1 - 10 0.9 | 1000 0.1 - 10 0.9 = 1000 0.099 |
| 1 0.65654 0.55646 | 0.66 0.3 | Volume1 < Volume2 Fullness1 > Fullness2 |
| 1 60 0.1 | 120 0.05 | Volume1 < Volume2  Fullness1 == Fullness2 |
| 1 45342432 0.65 | 45342432 0.7 | Volume1 == Volume2  Fullness1 < Fullness2 |
| 2 54 0.4 > 54 0.5 | The statement is wrong! |
| 2 76 0.1 > 54 0.9 | The statement is true! |
| 2 56 0.9 < 100 0.9 | The statement is true! |
| 3 | Exit completed! |

Таблица 2 – test\_02.txt и результаты его выполнения

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 78 0.6 + 102932 0.01 | 78 0.6 + 102932 0.01 = 78 1 |
| 1 15 0.1 - 90 0.1 | 15 0.1 - 90 0.1 = 15 0 |
| 1 34 0.65 - 100909 0.98 | 34 0.65 - 100909 0.98 = 34 0 |
| 1 45.78 0.5 - 34 0.76 | 45.78 0.5 - 34 0.76 = 45.78 0.321756 |
| 1 22 0.65 + 43 1 | 22 0.65 + 43 1 = 22 1 |
| 1 54 0.8 | 54 0.9 | Volume1 == Volume2  Fullness1 < Fullness2 |
| 1 65 0.08 | 64 0.9 | Volume1 > Volume2 Fullness1 < Fullness2 |
| 1 0.434 0.56 | 0.67 0.78 | Volume1 < Volume2  Fullness1 < Fullness2 |
| 2 76 1 > 75 1 | The statement is true! |
| 2 89 0.8 < 90 0.9 | The statement is true! |
| 2 98 0.012 < 15 1 | The statement is wrong! |
| 2 67 0.87 > 78 0.98 | The statement is wrong! |
| 1 10 0.1 - 5 0.1 | 10 0.1 - 5 0.1 = 10 0 |
| 1 67 0.8 + 12 0.9 | 67 0.8 + 12 0.9 = 67 0.961194 |
| 1 32 0.8 - 54 0.6 | 32 0.8 - 54 0.6 = 32 0.125 |
| 2 54 0.78 > 34 0.1 | The statement is true! |
| 3 | Exit completed! |

Таблица 3 – test\_02.txt и результаты его выполнения

|  |  |
| --- | --- |
| Ввод | Результаты |
| 1 100 1.2 + 0.9 0.9 | Invalid data was entered! |
| 1 98 0.5 ' 0.7 0.8 | Entered invalid operation |
| 2 78 0.7 > 0.8 0.1 | The statement is true! |
| 2 0.7 0.443242 < 0.6 0.434243243 | The statement is wrong! |
| 2 76 0.5 < 77 0.5 | The statement is true! |
| 2 56 0.6544 > 0.43242 0.2 | The statement is true! |
| 1 57 0.97 + 57 0.03 | 57 0.97 + 57 0.03 = 57 1 |
| 1 98 0.87 - 1 0.1 | 98 0.87 - 1 0.1 = 98 0.860816 |
| 1 10000000 0.27 - 1001223010 0.78 | 1e+07 0.27 - 1.00122e+09 0.78 = 1e+07 0 |
| 1 88433343 0.7564 - 1567 0.34 | 8.84333e+07 0.7564 - 1567 0.34 = 8.84333e+07 0.75638 |
| 1 9898.876 0.45 + 14 0.5 | 9898.88 0.45 + 14 0.5 = 9898.88 0.450707 |
| 1 986 0.87874 - 467 0.9 | 986 0.87874 - 467 0.9 = 986 0.831377 |
| 1 45 0.15 + 56 0.789 | 45 0.15 + 56 0.789 = 45 1 |
| 1 43 0.4 | 45 0.67 | Volume1 < Volume2  Fullness1 < Fullness2 |
| 1 98989 0.89 | 98989 0.9 | Volume1 == Volume2 Fullness1 < Fullness2 |
| 1 7843043 0.65656 | 5435453 0.8978784 | Volume1 > Volume2 Fullness1 > Fullness2 |
| 3 | Exit completed! |

# 4. Листинг программы

**CMakeLists.txt**

cmake\_minimum\_required (VERSION 3.2)

project (OOP)

add\_executable (oop\_exercise\_02 main.cpp menu.cpp functions.cpp bottle.cpp)

set\_target\_properties(oop\_exercise\_02 PROPERTIES CXX\_STANDART 14

CXX\_STANDART\_REQUIRED ON)

#pragma once

#include <iostream>

using namespace std;

**bottle.h**

class Bottle{

    private:

    long double Volume; // Объём

    long double FillingPer; // Процент наполнения

    public:

    Bottle(): Volume(0), FillingPer(0) {}

    Bottle(long double v): Volume(v), FillingPer(0) {}

    Bottle(long double v, long double f): Volume(v), FillingPer(f) {}

    void GetBottle(); // Функция получения данных о бутылке

    void PrintBottle(); // Функция вывода данных о бутылке

// Функция проверки правильности ввода данных

    friend int DataValidation(Bottle a, Bottle b);

// Функция сравнения объёмов двух бутылок

    friend void VolumeComparison(Bottle a, Bottle b);

// Функция сравнения объёмов бутылок, которые заполнены водой

    friend void FullnessComparison(Bottle a, Bottle b);

    Bottle operator+ (Bottle capacity2); // Оператор сложения двух бутылок

    Bottle operator- (Bottle capacity2); // Оператор вычитания двух бутылок

// Оператор сравнения статического объёма бутылок

    void operator> (Bottle capacity2);

// Оператор сравнения статического объёма бутылок

    void operator< (Bottle capacity2);

    ~Bottle(){}

};

// Пользовательский литерал для работы с бутылками

Bottle operator"" \_bot(long double vol);

**bottle.cpp**

#include <iostream>

#include "bottle.h"

using namespace std;

void Bottle::GetBottle(){  // Функция получения данных о бутылке

    cin >> Volume >> FillingPer;

}

void Bottle::PrintBottle(){  // Функция вывода данных о бутылке

    cout << Volume << " " << FillingPer;

}

Bottle Bottle::operator+ (Bottle capacity2){ // Оператор сложения двух бутылок

    Bottle t;

    t.Volume = Volume;

    t.FillingPer = FillingPer;

    t.FillingPer = (t.Volume \* t.FillingPer + capacity2.Volume \*

capacity2.FillingPer) / Volume;

    if(t.FillingPer > 1){

        t.FillingPer = 1;

    }

    return t;

}

Bottle Bottle::operator- (Bottle capacity2){  // Оператор вычитания двух бутылок

    Bottle t;

    t.Volume = Volume;

    t.FillingPer = FillingPer;

    t.FillingPer = (t.Volume \* t.FillingPer - (capacity2.Volume \*

(1 - capacity2.FillingPer))) / Volume;

    if (t.FillingPer < 0){

        t.FillingPer = 0;

    }

    return t;

}

// Оператор сравнения статического объёма бутылок

void Bottle::Bottle::operator>(Bottle capacity2){

    if(Volume > capacity2.Volume){

        cout << "The statement is true!\n\n";

    }

    else{

        cout << "The statement is wrong!\n\n";

    }

}

// Оператор сравнения статического объёма бутылок

void Bottle::Bottle::operator<(Bottle capacity2){

    if(Volume < capacity2.Volume){

        cout << "The statement is true!\n\n";

    }

    else{

        cout << "The statement is wrong!\n\n";

    }

}

// Пользовательский литерал для работы с бутылками

Bottle operator"" \_bot(long double vol){

    return Bottle(vol, 1);

}

**functions.h**

#pragma once

#include "bottle.h"

// Функция проверки правильности ввода данных

int DataValidation(Bottle a, Bottle b);

// Функция сравнения объёмов двух бутылок

void VolumeComparison(Bottle a, Bottle b);

// Функция сравнения объёмов бутылок, которые заполнены водой

void FullnessComparison(Bottle a, Bottle b);

// Функция сравнения бутылок по статическому объему и объёму заполненности водой

void Comparison(Bottle a, Bottle b);

// Функция выполнения арифметических операций над бутылками

Bottle Operation(Bottle a, Bottle b, char c);

// Функция выполнения операторов сравнения бутылок

void Verification(Bottle a, Bottle b, char c);

// Функция вывода результата выполнения арифметической операции

void PrintResult(Bottle a, Bottle b, char c, Bottle result);

**functions.cpp**

#include <iostream>

#include "bottle.h"

#include "functions.h"

using namespace std;

// Функция проверки правильности ввода данных

int DataValidation(Bottle a, Bottle b){

    if(a.Volume < 0 | a.FillingPer < 0 | a.FillingPer > 1 | b.Volume < 0 | b.FillingPer < 0 | b.FillingPer > 1){

        cout << "Invalid data was entered!\n\n";

        return 1;

    }

    return 0;

}

// Функция сравнения объёмов двух бутылок

void VolumeComparison(Bottle a, Bottle b){

    if(a.Volume > b.Volume){

        cout << "Volume1 > Volume2\n";

    }

    else if(a.Volume < b.Volume){

        cout << "Volume1 < Volume2\n";

    }

    else{

        cout << "Volume1 == Volume2\n";

    }

}

// Функция сравнения объёмов бутылок, которые заполнены водой

void FullnessComparison(Bottle a, Bottle b){

    if(a.Volume \* a.FillingPer >  b.Volume \* b.FillingPer){

        cout << "Fullness1 > Fullness2\n\n";

    }

    else if(a.Volume \* a.FillingPer <  b.Volume \* b.FillingPer){

        cout << "Fullness1 < Fullness2\n\n";

    }

    else{

        cout <<"Fullness1 == Fullness2\n\n";

    }

}

// Функция сравнения бутылок по статическому объему и объёму заполненности водой

void Comparison(Bottle a, Bottle b){

    VolumeComparison(a, b);

    FullnessComparison(a, b);

}

// Функция выполнения арифметических операций над бутылками

Bottle Operation(Bottle a, Bottle b, char c){

    Bottle res;

    if(c == '+'){

        res = a + b;

    }

    else if(c == '-'){

        res = a - b;

    }

    else{

        cout << "Wrong operation was selected!\n";

    }

    return res;

}

// Функция выполнения операторов сравнения бутылок

void Verification(Bottle a, Bottle b, char c){

    if(c == '>'){

        a > b;

    }

    else if(c == '<'){

        a < b;

    }

}

// Функция вывода результата выполнения арифметической операции

void PrintResult(Bottle a, Bottle b, char c, Bottle result){

    cout << "Solution (Volume, Filling Persent): ";

    a.PrintBottle();

    cout << " " << c << " ";

    b.PrintBottle();

    cout << " = ";

    result.PrintBottle();

    cout <<"\n\n";

}

**menu.h**

#pragma once

int menu(); // Функция вывода меню действий

**menu.cpp**

#include <iostream>

#include "menu.h"

#include "bottle.h"

#include "functions.h"

using namespace std;

int menu() // Функция вывода меню действий

{

    while(true){

        int variant;

        cout << "Choose an action:\n";

        cout << "1 - Perform operation\n";

        cout << "2 - Compare bottle volumes\n";

        cout << "3 - Exit\n";

        cin >> variant;

        if(variant == 1){

            cout << "Work with bottles (a, b - bottles):\n";

            cout << "a + b - Addition of filling volume\n";

            cout << "a - b - Substraction of filling volume\n";

            cout << "a | b - Comparing bottles\n";

            char c;

            Bottle a, b;

            Bottle result;

            a.GetBottle();

            cin >> c;

            b.GetBottle();

            if(DataValidation(a, b) == 1){

                continue;

            }

            if(c != '+' && c != '-'){

                if(c == '|'){

                    Comparison(a, b);

                }

                else{

                    cout << "Entered invalid operation\n\n";

                }

            }

            else{

                PrintResult(a, b, c, Operation(a, b, c));

            }

        }

        else if (variant == 2){

            cout << "Compare volumes of bottles\n";

            cout << "Comparison operations: >, <\n";

            char c;

            Bottle a, b;

            a.GetBottle();

            cin >> c;

            b.GetBottle();

            if(DataValidation(a, b) == 1){

                cout << "\n";

                continue;

            }

            Verification(a, b, c);

        }

        else if (variant == 3){

            cout << "Exit completed!";

            return 0;

        }

        else{

            cout << "Entered action does not exist\n\n";

        }

   }

   return 0;

}

**main.cpp**

#include <iostream>

#include "menu.h"

using namespace std;

int main(){

    menu();

}

Ссылка на github репозиторий: <https://github.com/HarryLiker/oop_exercise_02>

# 5. Выводы

Была написана программа реализации класса Bottle для работы с ёмкостями. Ёмкости можно складывать, вычитать и сравнивать. Был создан пользовательский литерал \_bot для работы с константами типа Bottle. Пользовательский литерал является некоторым выражением, которое создаёт объект. Если представлять арифметические операции над бутылками в реальном мире, то сложение первой и второй бутылки выглядело бы как переливание воды из второй бутылки в первую. Вычитание второй бутылки из первой выглядело бы как зачерпывание воды из первой бутылки второй.

Были проведены тесты, после которых выводился ожидаемый результат. Были получены навыки работы с классами, созданием конструкторов и деструктора, работы с дружественными для класса функциями и работы с пользовательскими литералами.

# Список используемых источников

1. Р. Лафоре, В.Е. Зайцев «Объектно-ориентированное программирование в C++» - Москва: «Питер», 2004.

2. Информация о работе с дружественными функциями - [Электронный ресурс]. - URL: <https://ravesli.com/urok-126-druzhestvennye-funktsii-i-klassy>

(дата обращения: 08.10.2020).

3. Информация о пользовательских литералах - [Электронный ресурс]. - URL: <https://habr.com/ru/post/140357> (дата обращения: 08.10.2020).

4. Информация о пользовательских литералах - [Электронный ресурс]. - URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/user-defined-literals-cpp?view=vs-2019> (дата обращения: 10.10.2020).

5. Информация о пользовательских литералах - [Электронный ресурс]. - URL: <https://coderoad.ru/237804/> (дата обращения: 11.10.2020).