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: Перегрузка операторов в C++

Студент: Пивницкий Даниэль Сергеевич
Группа: 80-206
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

a. Ознакомиться с теоретическим материалом.

b. Создать класс **Money** для работы с денежными суммами. Сумма денег должна быть представлено двумя полями: типа unsigned long long для рублей и типа unsigned char – для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.
Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с константами типа **Money**.

c. Настроить CMake файл для сборки программы.

d. Подготовить наборы тестовых данных.

e. Загрузить файлы лабораторной работы в репозиторий GitHub.

f. Подготовить отчет по лабораторной работе.

2. Описание программы

Данная программа создает класс Money для работы с денежными суммами. Сумма денег представлена двумя полями: типа unsigned long long для рублей и типа unsigned short int – для копеек. Дробная часть (копейки) при выводе на экран отделена от целой части запятой. Реализовано сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и перегружены операторы сравнения.

В программе реализованы следующие функции:

1. Вывести количество рублей
2. Сложить 2 суммы
3. Вычесть из первой суммы вторую
4. Разделить первую сумму на вторую
5. Разделить первую сумму на дробное число
6. Умножить первую сумму на дробное число
7. Оператор <
8. Оператор >
9. Оператор <=
10. Оператор >=
11. Оператор ==
12. Оператор !=

При выполнении первой функции программа выводит количество денег в формате <рубли>,<копейки>. Рубли и копейки разделяются в разные переменные класса Money с помощью функции shiftUp().
При выполнении второй функции программа просит ввести вторую сумму, затем складывает с первой, записывает в класс a2 для последующего использования функции shiftUp() и вывода.
Третья и четвертая функции работают аналогично второй.
Пятая и шестая функции берут на вход дробное число типа float, проводят соответствующие операции и выводят сумму аналогично пунктам 2-4.

Перегрузка операторов 7-12 — операторы сравнения. Они берут на вход вторую сумму, которая будет сравниваться с первоначальной, затем выводится результат true/false (true в случае верного сравнения, false – в противном случае).

3. Набор тестов

test_1

Входные данные:

1000

500

2

test_2

Входные данные:

1000.25

50.64

0

test_3

Входные данные:

500

500

3

test_4

Входные данные:

1234.5

123

4

4. Результаты выполнения тестов

test_1

Выводимые данные:

1000,00

500,00

1500,00

500,00

2000,00

<: false

>: true

<=: false

>=: true

==: false

!=: true

$:321350

test_2

Выводимые данные:

cannot be divided by zero

test_3

Выходные данные:

500,00

500,00

1000,00

0,00

1,00

166,66

1500,00

<: false

>: false

<=: true

>=: true

==: true

!=: false

$:321350

test_4

Выходные данные:

1234,50

123,00

1357,50

1111,50

10,68

308,50

4936,00

<: false

>: true

<=: false

>=: true

==: false

!=: true

$:321350

5. Листинг программы

```
//
//  main.cpp
//  lab2
//  Variant 8
//  М8о-206Б-19
//  Created by Daniel Pivnitskiy on 10.10.2020.
//  github.com/SLAST1
//  Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
/*
Создать класс Money для работы с денежными суммами.
Сумма денег должна быть представлено двумя полями: типа unsigned long long для рублей и типа unsigned char – для копеек.
Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой.
Реализовать сложение сумм, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.
Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с константами типа Money.
*/

#include <iostream>
#include "money.h"

#define UNUSED(variable) (void)variable

int main(int argc, char** argv) {
```

```
Money a1;
Money a2;
float arg;

std::cout << "First summ:" << std::endl;
//a1.get(std::cin);
std::cin >> a1;
std::cout << "Second summ:" << std::endl;
//a2.get(std::cin);
std::cin >> a2;
std::cout << "Number to div and multiply" << std::endl;
std::cin >> arg;
if(arg == 0){
    std::cout << "cannot be divided by zero" << std::endl;
    return 0;
}
std::cout << std::endl;

//a1.show(std::cout);
std::cout << a1;
//a2.show(std::cout);
std::cout << a2;

//(a1 + a2).show(std::cout);
std::cout << a1 + a2;

//(a1 - a2).show(std::cout);
std::cout << a1 - a2;
```

```
std::cout << a1 / a2 << std::endl;

//(a1 / arg).show(std::cout);
std::cout << a1 / arg;

//(a1 * arg).show(std::cout);
std::cout << a1 * arg << std::endl;

if(a1 < a2) std::cout << "<:" << std::endl;
else std::cout << "<: false" << std::endl;

if(a1 > a2) std::cout << ">: true" << std::endl;
else std::cout << ">: false" << std::endl;

if(a1 <= a2) std::cout << "<=: true" << std::endl;
else std::cout << "<=: false" << std::endl;

if(a1 >= a2) std::cout << ">=: true" << std::endl;
else std::cout << ">=: false" << std::endl;

if(a1 == a2) std::cout << "==: true" << std::endl;
else std::cout << "==: false" << std::endl;

if(a1 != a2) std::cout << "!=: true" << std::endl;
else std::cout << "!=: false" << std::endl;
long double sum;
std::cout << "insert summ (+50 cops): ";
std::cin >> sum;
```

```
    std::cout << sum + 50.00_toCop << std::endl;

    UNUSED(argc);

    UNUSED(argv);

    return 0;

}
```

6. Выводы

Проделав данную работу я изучил перегрузку операторов и пользовательские литералы. Сделал вывод, что перегрузка операторов необходима для переопределения того, что должен делать данный оператор, если это необходимо. Например, если класс реализован двумя переменными, и их необходимо сравнить как одно целое, то необходимо применить перегрузку операторов. Так же пользовательские литералы необходимы для просчета константных переменных где это возможно, так как расчеты производятся на этапе компиляции.

7. Список литературы

1.Руководство по написанию кода на C++ [Электронный ресурс]. URL:https://metanit.com/cpp/tutorial/
Дата обращения: 10.09.2019

2.Документация по C++ [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp
Дата обращения 12.09.2019