Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторной работе № 02**

**по курсу «Объектно-ориентированное программирование»**

**Тема:**

**«Операторы, литералы»**

|  |  |
| --- | --- |
| Студент: | Пшеницын А. А. |
| Группа: | М80-208Б-18 |
| Преподаватель: | Журавлев А. А. |
| Вариант: | 17 |
| Оценка: |  |
| Дата: |  |

Москва

2019

**Цель:**

Цель:

• Изучение механизмов перегрузки операторов;

• Изучение механизмов работы с пользовательскими литералами;

**Задание**

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 в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Budget.

**Код**

**budget.h**

#ifndef \_BUDGET\_H\_

#define \_BUDGET\_H\_

#include <iostream>

#include <cstdlib>

#include <cmath>

typedef struct Budget Budget;

struct Budget{

//private:

double a; //собственная часть

double b; //заемная часть

public:

Budget();

Budget(double ch1, double ch2);

friend std::istream& operator>> (std::istream& is, Budget& bud);

friend std::ostream& operator<< (std::ostream& os, Budget& bud);

friend bool operator== (const Budget& lhs, const Budget& rhs);

friend bool operator!= (const Budget& lhs, const Budget& rhs);

friend bool operator> (const Budget& lhs, const Budget& rhs);

friend bool operator< (const Budget& lhs, const Budget& rhs);

friend Budget& operator+= (Budget& lhs, const Budget& rhs);

friend Budget& operator-= (Budget& lhs, const Budget& rhs);

friend Budget operator+ (const Budget& lhs, const Budget& rhs);

friend Budget operator- (const Budget& lhs, const Budget& rhs);

friend Budget operator\* (const Budget& lhs, const Budget& rhs);

friend Budget operator/ (const Budget& lhs, const Budget& rhs);

Budget& operator= (const Budget& bud){

a = bud.a;

b = bud.b;

return \*this;

}

};

#endif

**budget.cpp**

#include "budget.h"

#include <iostream>

#include <cstdlib>

#include <cmath>

Budget::Budget(){

a = 0;

b = 0;

}

Budget::Budget(double ch1, double ch2){

a = round(ch1 \* 100) / 100;

b = round(ch2 \* 100) / 100;

}

std::istream& operator>> (std::istream& is, Budget& bud){

is >> bud.a >> bud.b;

bud.a = round(bud.a \* 100) / 100;

bud.b = round(bud.b \* 100) / 100;

return is;

}

std::ostream& operator<< (std::ostream& os, Budget& bud){

os << bud.a << " " << bud.b << '\n' << '\n';

return os;

}

bool operator== (const Budget& lhs, const Budget& rhs){

return (lhs.a == rhs.a) && (lhs.b == rhs.b);

}

bool operator!= (const Budget& lhs, const Budget& rhs){

return (lhs.a != rhs.a) || (lhs.b != rhs.b);

}

bool operator> (const Budget& lhs, const Budget& rhs){

return (lhs.a > rhs.a) && (lhs.b > rhs.b);

}

bool operator< (const Budget& lhs, const Budget& rhs){

return (lhs.a < rhs.a) && (lhs.b < rhs.b);

}

Budget& operator+= (Budget& lhs, const Budget& rhs){

lhs.a += rhs.a;

lhs.b += rhs.b;

return lhs;

}

Budget& operator-= (Budget& lhs, const Budget& rhs){

lhs.a = fabs(lhs.a - rhs.a);

lhs.b = fabs(lhs.b - rhs.b);

return lhs;

}

Budget operator+ (const Budget& lhs, const Budget& rhs){

Budget res{};

res = lhs;

res += rhs;

return res;

}

Budget operator- (const Budget& lhs, const Budget& rhs){

Budget res{};

res = lhs;

res -= rhs;

return res;

}

Budget operator\* (const Budget& lhs, const Budget& rhs){

Budget res{};

res.a = round(lhs.a \* rhs.a \* 100) / 100;

res.b = round(lhs.b \* rhs.b \* 100) / 100;

return res;

}

Budget operator/ (const Budget& lhs, const Budget& rhs){

Budget res{};

res.a = round(lhs.a / rhs.a \* 100) / 100;

res.b = round(lhs.b / rhs.b \* 100) / 100;

return res;

}

/\*Budget operator"" \_bud(std::string str){

std::istringstream is(str);

Budget bud;

is >> bud;

return bud;

}\*/

**main.cpp**

#include <cmath>

#include <exception>

#include <stdexcept>

#include <sstream>

#include "budget.h"

Budget operator""\_bud(const char\* str, size\_t size){

std::stringstream is(str);

Budget bud;

is >> bud;

return bud;

}

int main(){

Budget my\_bud{};

std::cin >> my\_bud;

std::cout << my\_bud;

Budget his\_bud{};

std::cin >> his\_bud;

std::cout << his\_bud;

Budget res{};

std::cout << "Addition" << '\n';

res = my\_bud + his\_bud;

std::cout << res;

std::cout << "Subtraction" << '\n';

res = my\_bud - his\_bud;

std::cout << res;

std::cout << "Multiplication" << '\n';

res = my\_bud \* his\_bud;

std::cout << res;

std::cout << "Division" << '\n';

res = my\_bud / his\_bud;

std::cout << res;

if(my\_bud == his\_bud){

std::cout << "Budget1 == Budget2 - YES" << '\n' << '\n';

}

else{

std::cout << "Budget1 == Budget2 - NO" << '\n' << 'n';

}

if(my\_bud != his\_bud){

std::cout << "Budget1 != Budget2 - YES" << '\n' << '\n';

}

else{

std::cout << "Budget1 != Budget2 - NO" << '\n' << '\n';

}

Budget dop = "1 5"\_bud;

std::cout << dop;

}

**Ссылка на репозиторий на GitHub**

<https://github.com/AlexPshen/oop_exercise_06.git>

**Тесты**

**test\_01.txt**

8 9

8 9

res\_01.txt

8 9

8 9

8 9

8 9

Addition

16 18

Subtraction

0 0

Multiplication

64 81

Division

1 1

Budget1 == Budget2 - YES

Budget1 != Budget2 - NO

1 5

**test\_02.txt**

5 8

8 10

res\_01.txt

5 8

5 8

9 10

9 10

Addition

14 18

Subtraction

4 2

Multiplication

45 80

Division

0.56 0.8

Budget1 == Budget2 - NO

nBudget1 != Budget2 - YES

1 5

**Объяснение результатов работы программы**

В данной лабораторной работе были перегружены такие операторы, как >> и << (операторы ввода и вывода соответственно) , оператор проверки на равенство == и неравенство != , операторы +=, -=. +. -, \*, /, а также оператор присваивания.

Также в данной лабораторной работе реализован пользовательский литерал на \_bud.

**Вывод**

В данной лабораторной работе мы познакомились с механизмом перегрузки операторов. Оператор в С++ - это некоторое действие или функция обозначенная специальным символом. Для того что бы распространять эти действия на новые типы данных, при этом сохраняя естественный синтаксис, в С++ была введена возможность перегрузки операторов, что делает код более простым для чтения.