МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Михеева Кристина Олеговна, группа М8О-207Б-20*

Преподаватель *Дорохов Евгений Павлович*

**Условие:**

**Вариант 17:** 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.

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Реализовать над объектами реализовать в виде перегрузки операторов.

Реализовать пользовательский литерал для работы с константами объектов созданного класса.

**Описание программы :**

Исходный код лежит в 3 файлах:

1. main.cpp: основная программа, взаимодействие с пользователем посредством команд из меню

2. Budget.h: описание класса бюджета

3. Budget.cpp: описание класса точки

**Дневник отладки:**

В данной лабораторной работы возникли небольшие проблемы с перегрузкой операторов, и через некоторое время были устранены.

**Выводы:**

В данной лабораторной работе я познакомилась с двумя очень интересными вещами.Во-первых , это было использование перегрузки операторов. Данные операторы используются в возможности одновременного существования в одной области видимости нескольких различных вариантов применения операторов, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

Во-вторых, использование пользовательских литер. Оказывается, что это очень удобная и практичная вещь. Ведь когда мы вычисляем какие-то значения без использования вспомогательных функций, а попросту переопределением специального оператора.

**Листинг:**

**Budget.h**

#ifndef BUDGET\_H

#define BUDGET\_H

#include <iostream>

class Budget {

public:

Budget();

Budget(double a, double b);

void Rouding();

void Difference();

void Multiplier();

void Division();

void Compare();

Budget operator + (const Budget &object);

Budget operator - (const Budget &object);

bool operator == (const Budget &object);

bool operator > (const Budget &other);

bool operator < (const Budget &other);

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

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

~Budget();

private:

double personal;

double credit;

};

#endif

**Budget.срр**

#include "Budget.h"

#include <cmath>

Budget::Budget() {

personal = 0.0;

credit = 0.0;

std:: cout << "Start: " << std:: endl;

}

Budget::Budget(double a, double b) {

if (a > 0.0 && b > 0.0){

personal = a;

credit = b;

}

else {

std:: cout << "Please enter positive numbers!" << std:: endl;

}

std:: cout << "The budget according to your parameters has been created" << std:: endl;

}

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

std:: cout << "Please enter your budget data: " << std:: endl;

is >> object.personal >> object.credit;

if ((object.personal <= 0.0) || (object.credit <= 0.0)) {

std:: cout << "Invalind input. Enter again!" << std:: endl;

is >> object.personal >> object.credit;

}

std:: cout << "The budget has been created via istream" << std:: endl;

return is;

}

Budget Budget::operator + (const Budget &other) {

this->personal += other.personal;

this->credit += other.credit;

return \*this;

}

Budget Budget::operator - (const Budget &other) {

this->personal -= other.personal;

this->credit -= other.credit;

return \*this;

}

void Budget::Rouding() {

this->personal = round (this->personal \* 100.0) / 100.0;

this->credit = round (this->credit \* 100.0) / 100.0;

std:: cout << "Rouding: " << this->personal << " " << this->credit << std:: endl;

}

bool Budget::operator < (const Budget &other) {

if ((personal < other.personal) || (personal == other.personal && credit < other.credit)) {

return true;

}

else {

return false;

}

}

bool Budget::operator > (const Budget &other) {

if ((personal > other.personal) || (personal == other.personal && credit > other.credit)) {

return true;

}

else {

return false;

}

}

void Budget::Difference() {

double differ = personal - credit;

std:: cout << "Difference: " << differ << std:: endl;

}

void Budget::Multiplier(){

double mult = personal \* credit;

std:: cout << "Multiplier: " << mult << std:: endl;

}

void Budget::Division(){

double div = personal / credit;

std:: cout << " Division: " << div << std:: endl;

}

void Budget::Compare(){

if (personal > credit){

std:: cout << "The personal is more that credit!" << std:: endl;

}

else if (personal == credit) {

std:: cout << "Budget are equal!" << std:: endl;

}

else {

std:: cout << "The credit is more that personal!" << std:: endl;

}

}

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

os << object.personal << ";" << object.credit<< std:: endl;

return os;

}

bool Budget::operator == (const Budget &object) {

if (personal == object.personal && credit == object.credit) {

return true;

}

return false;

}

Budget::~Budget() {

std:: cout << "FROM DESTRUCTOR: Your budget has been deleted" << std:: endl;

}

**main.cpp**

#include "Budget.h"

unsigned long long operator "" \_todollars(unsigned long long sum) {

unsigned long long dollars = sum / 70;

return dollars;

}

int main(){

Budget a;

std:: cin >> a;

Budget b;

std:: cin >> b;

Budget c = a + b;

std:: cout << c;

Budget d;

std:: cin >> d;

Budget f;

std:: cin >> f;

Budget x = d - f;

std:: cout << x;

return 0;

}