ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 6

Дисциплина:

Объектно-ориентированное программирование

Тема:

Массивы и строки.

Выполнил(а): студент(ка) группы 211-7210
Салов Д.К.
(Фамилия И.О.)

Дата, подпись 24.05.22
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание)
(Оценка)
Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____

_____

_____

Москва

2021

Цель: Получить практические навыки в создании классов и их последующем использовании.

Задания:

5. Пополните класс time, рассмотренный в упражнении 3, перегруженными операциями увеличения (++) и уменьшения (--), которые работают в обеих, префиксной и постфиксной, формах записи и возвращают значение. Дополните функцию main(), чтобы протестировать эти операции.

6. Добавьте в класс time из упражнения 5 возможность вычитать значения времени, используя перегруженную операцию -, и умножать эти значения, используя тип float и перегруженную операцию *.

7. Модифицируйте класс fraction в четырехфункциональном дробном калькуляторе из упражнения 11 лабораторной работы №4 так, чтобы он использовал перегруженные операции сложения, вычитания, умножения и деления. (Вспомните правила арифметики с дробями в упражнении 12 лабораторной работы №2 «Циклы и ветвления».) Также перегрузите операции сравнения == и != и используйте их для выхода из цикла, когда пользователь вводит 0/1, 0 и 1 значения двух частей дроби. Вы можете модифицировать и функцию lowterms() так, чтобы она возвращала значение ее аргумента, уменьшенное до несократимой дроби. Это будет полезным в арифметических функциях, которые могут *быть выполнены* сразу *после получения ответа.*

8. Модифицируйте класс bMoney из упражнения 12 лабораторной работы №5 «Массивы и строки», включив арифметические операции, выполненные с помощью перегруженных операций:

bMoney = bMoney + bMoney bMoney = bMoney - bMoney

bMoney = bMoney * long double (цена за единицу времени, затраченного на изделие) long

double = bMoney / bMoney (общая цена, деленная на цену за изделие) bMoney = bMoney / long double (общая цена, деленная на количество изделий) Заметим, что операция / перегружена дважды. Компилятор может различить оба варианта, так как их аргументы разные. Помним, что легче выполнять арифметические операции с объектами класса bMoney, выполняя те же операции с его long double данными.

Убедитесь, что программа main() запросит ввод пользователем двух денежных строк и числа с плавающей точкой. Затем она выполнит все пять операций и выведет результаты. Это должно происходить в цикле, так, чтобы пользователь мог ввести еще числа, если это понадобится.

Некоторые операции с деньгами не имеют смысла: bMoney*bMoney не представляет ничего реального, так как нет такой вещи, как денежный квадрат; вы не можете прибавить bMoney к long double (что же будет, если рубли сложить с изделиями?). Чтобы сделать это невозможным, скомпилируйте такие неправильные операции, не включая операции преобразования для bMoney в long double или long double в bMoney. Если вы это сделаете и запишете затем выражение типа: bmon2 = bmonl + widgets; //это не имеет смысла

то компилятор будет автоматически преобразовывать widgets в ЬМопеу и выполнять

сложение. Без них компилятор будет отмечать такие преобразования как ошибки, что позволит легче найти концептуальные ошибки. Также сделайте конструкторы преобразований явными.

9.

```
// arrover3.cpp

// creates safe array (index values are checked before access)

// uses overloaded [] operator for both put and get



#include <iostream> using namespace std;

#include <process.h>           //for exit() const int LIMIT = 100;         //array size

////////////////////////////////////////////////////////////// class safearay     {    private:        int arr[LIMIT];    public:        int& operator [](int n)  //note: return by reference

        {

        if( n< 0 || n>=LIMIT )

          { cout << "\nIndex out of bounds"; exit(1); }          return arr[n];

        }

  };

////////////////////////////////////////////////////////////// int main()    {

  safearay sa1;
```

```
  for(int j=0; j<LIMIT; j++)  //insert elements        sa1[j] = j*10;          //*left* side of equal sign


  for(j=0; j<LIMIT; j++)      //display elements

    {       int temp = sa1[j];       //*right* side of equal sign

cout << "Element " << j << " is " << temp << endl;

    }    return 0;

  }
```

Дополните класс safearray из программы ARR0VER3 так, чтобы пользователь мог определять и верхнюю, и нижнюю границы массива (например, индексы, начинающиеся с 100 и заканчивающиеся 200). Имеем перегруженную операцию доступа к членам массива, проверяющую индексы каждый раз, когда к массиву нужен доступ, для проверки того, что мы не вышли за пределы массива. Вам понадобится конструктор с двумя аргументами, который определяет верхнюю и нижнюю границы. Так как мы еще не изучили, как выделять память динамически, то данные класса все еще будут размещаться в массиве, состоящем из 100 элементов, но вообще вы можете преобразовывать индексы массива safearray в индексы реального массива целых чисел произвольным образом. Например, если пользователь определил диапазон от 100 до 175, то вы можете преобразовать его в диапазон от агг[0] до агг[75].

10. Создайте класс Polar, который предназначен для хранения полярных координат

(радиуса и угла). Перегрузите операцию + для выполнения сложения для объектов класса Polar. Сложение двух объектов выполняется путем сложения координат X объектов, а затем координат Y. Результат будет координатами новой точки. Таким образом, вам нужно будет преобразовать полярные координаты к прямоугольным, сложить их, а затем обратно преобразовать прямоугольные координаты результата к полярным.

11. Помните структуру sterling? Мы встречались с ней в упражнении 10 лабораторной работы №1 «Основы программирования на C++», в упражнении 11 лабораторной работы №3 и в других местах. Преобразуйте ее в класс, имеющий переменные для фунтов (типа long), шиллингов (типа int) и пенсов (типа int). Создайте в классе

следующие функции: конструктор без аргументов;

конструктор с одним аргументом типа double (для преобразования от десятичных фунтов); конструктор с тремя аргументами: фунтами, шиллингами и пенсами; метод

getSterLing() для получения от пользователя значений количества фунтов, шиллингов и пенсов в формате £9.19.11;

метод putSterling() для вывода значений количества фунтов, шиллингов и пенсов в

формате £9.19.11; метод для сложения (sterling + sterling), используя перегруженную операцию +; метод вычитания (sterling - sterling), используя перегруженную операцию -; метод умножения (sterling * double), используя перегруженную операцию *; метод деления (sterling / sterling), используя перегруженную операцию /; метод деления (sterling / double), используя перегруженную операцию /;

операцию double (для преобразования к типу double)

Выполнять вычисления вы можете, например, складывая отдельно данные объекта: сложить сначала пенсы, затем шиллинги и т. д. Однако легче использовать операцию преобразования для преобразования объекта класса sterling к типу double, выполнить вычисления с типами double, а затем преобразовать обратно к типу sterling.

12. Напишите программу, объединяющую в себе классы bMoney из упражнения 8 и sterling из упражнения 11. Напишите операцию преобразования для преобразования между классами bMoney и sterling, предполагая, что один фунт (£1.0.0) равен пятидесяти долларам ($50.00). Это приблизительный курс обмена для XIX века, когда Британская империя еще использовала меру фунты-шиллинги-пенсы. Напишите программу main(), которая позволит пользователю вводить суммы в каждой из валют и преобразовывать их в другую валюту с выводом результата. Минимизируйте количество изменений в существующих классах bMoney и sterling.

Код:

```
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>

using namespace std;



//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++5  //works
class Time
{
private: int hrs, mins, secs; public:
	Time() : hrs(0), mins(0), secs(0) //no-arg constructor
	{ }
	Time(int h, int m, int s) : hrs(h), mins(m), secs(s) //3-arg constructor
	{ }
	void display()
	{
		cout << hrs << ":" << mins << ":" << secs;
	}
	Time operator + (Time t2) //add two times
	{
		int s = secs + t2.secs; //add seconds
		int m = mins + t2.mins; //add minutes
		int h = hrs + t2.hrs; //add hours
		if (s > 59) //if secs overflow,
		{
			s -= 60;
			m++;
		} // carry a minute
		if (m > 59) //if mins overflow,
```

```
		{
			m -= 60;
			h++;
		} // carry an hour
		return Time(h, m, s); //return temp value
	}
	Time operator++() {
		return Time(++hrs, ++mins, ++secs);
	}
	Time operator--(int) {
		return Time(hrs--, mins--, secs--);
	}
 };

void fun5(){
	Time time1(11, 22, 33);
	Time time2(1, 2, 3);
	Time time3;
	time3 = time1 + time2;
	time3.display();
	++time3;
	time3.display();
	time3--;
	time3.display();
	cout << endl;
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6 //names from
fun5 //works

class timeFrame_6 {
private:
	int hrs, mins, secs;
public:
	timeFrame_6() : hrs(0), mins(0), secs(0) {}
	timeFrame_6(int h, int m, int s) : hrs(h), mins(m), secs(s) {}
	void display() {
		cout << hrs << ":" << mins << ":" << secs << "\n";
	}
	timeFrame_6 operator + (timeFrame_6 t2) {
		int s = secs + t2.secs;
		int m = mins + t2.mins;
		int h = hrs + t2.hrs;
		if (s > 59) {
			s -= 60;
			m++;
		}
		if (m > 59) {
			m -= 60;
			h++;
		}
		return timeFrame_6(h, m, s);
	}
	timeFrame_6 operator - (timeFrame_6 t2) {
		int s = secs - t2.secs;
		int m = mins - t2.mins;
		int h = hrs - t2.hrs;
		if (s > 59) {
			s -= 60;
			m++;
		}
		if (m > 59) {
			m -= 60;
			h++;
		}
```

```
		return timeFrame_6(h, m, s);
	}
	timeFrame_6 operator * (timeFrame_6 t2) {
		int s = secs * t2.secs;
		int m = mins * t2.mins;
		int h = hrs * t2.hrs;
		if (s > 59) {
			s = s / 60;
			m += s;
		}
		if (m > 59) {
			m = m / 60;
			h += m;
		}
		return timeFrame_6(h, m, s);
	}
	timeFrame_6 operator++() {
		return timeFrame_6(++hrs, ++mins, ++secs);
	}
	timeFrame_6 operator--(int) {
		return timeFrame_6(hrs--, mins--, secs--);
	}
};

void fun6() {
	timeFrame_6 time1(5, 59, 59);
	timeFrame_6 time2(4, 30, 30);
	timeFrame_6 time3;
	time3 = time1 + time2;
	time3.display();
	++time3;
	time3.display();
	time3--;
	time3.display();
	time3 = time1 - time2;
	time3.display();
	time3 = time1 * time2;
	time3.display();
	cout << endl;
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++7 //works
class Fraction {
private:
	int numerator;
	int denominator;
public:
	Fraction() : numerator(), denominator() {}
	Fraction(int n, int d) : numerator(n), denominator(d) {}
	void getValueFromUser()
	{
		cout << "Enter fraction: \n";
		cin >> numerator >> denominator;
	}
	Fraction operator + (Fraction val)
	{
		numerator = numerator * val.denominator + denominator * val.numerator;
		denominator = denominator * val.denominator;
		return Fraction(numerator, denominator);
	}
	Fraction operator - (Fraction val)
	{
		numerator = numerator * val.denominator - denominator * val.numerator;
		denominator = denominator * val.denominator;
```

```
		return Fraction(numerator, denominator);
	}
	Fraction operator * (Fraction val)
	{
		numerator = numerator * val.numerator;
		denominator = denominator * val.denominator;
		return Fraction(numerator, denominator);
	}
	Fraction operator / (Fraction val)
	{
		numerator = numerator * val.denominator;
		denominator = denominator * val.numerator;
		return Fraction(numerator, denominator);
	}
	void lowterms() {
		long tnum, tden, temp, gcd;
		tnum = labs(numerator);
		tden = labs(denominator);
		if (tden == 0) {
			std::cout << "Error!\n";
			exit(1);
		}
		else if (tnum == 0) {
			numerator = 0;
			denominator = 1;
		}
		while (tnum != 0) {
			if (tnum < tden) {
				temp = tnum;
				tnum = tden;
				tden = temp;
			}
			tnum = tnum - tden;
		}
		gcd = tden;
		numerator = numerator / gcd;
		denominator = denominator / gcd;
	}
	bool operator != (Fraction val) {
		return numerator != val.numerator;
	}
	bool operator == (Fraction val) {
		return denominator == val.denominator;
	}
	void getOutput()const {
		cout << "Answer is: " << numerator << "/" << denominator << "\n";
	}
};


void fun7()
{
	char sign;
	char ch = 'y';
	Fraction val1, val2, result;
	Fraction stop(0, 1);
	do {
		val1.getValueFromUser();
		val2.getValueFromUser();
		cout << "Enter a sign: \n";
		cin >> sign;
		switch (sign) {
		case '+': {result = val1 + val2; break; }
		case '-': {result = val1 - val2; break; }
		case '*': {result = val1 * val2; break; }
```

```
                case '/': {result = val1 / val2; break; }
                }
                result.lowterms();
                result.getOutput();
                cout << "Continue? (y/n)\n";
                cin >> ch;
        } while (ch != 'n');
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++8 //wtf with
exercise? //works
class bMoney
{
private:
        string strMon;
        long double capital;

public:
        bMoney() : capital(0)
        {}
        bMoney(long double num)
        {
                capital = num;
        }
        operator long double() const
        {
                long double num = capital;
                return num;
        }
        void mstold()
        {
                cout << "Enter the amount in decimal point. End with a dollar sign" << endl;
                getline(cin, strMon, '$');

                int wlen = strMon.length();
                int n = 0;
                string num;

                for (int j = 0; j < wlen; j++)
                        if (strMon[j] != ',' && strMon[j] != '$')
                                num.push_back(strMon[j]); ;
                capital = stold(num);
        }
        bMoney operator + (bMoney mon2)
        {
                return capital + mon2.capital;
        }
        bMoney operator - (bMoney mon2)
        {
                return capital - mon2.capital;
        }
        bMoney operator * (bMoney mon2)
        {
                return capital * mon2;
        }
        bMoney operator / (bMoney mon2)
        {
                return capital / mon2.capital;
        }
        bMoney operator / (long double mon2)
        {
                return capital / mon2;
        }
        void output()const
        {
```

```
		cout << setiosflags(ios::fixed)
			<< setiosflags(ios::showpoint)
			<< setprecision(2)
			<< "\n" << capital << '$' << endl;
	}
};

void fun8()
{
	bMoney val1, val2, result;
	long double amount = 0;
	do {
		val1.mstold();
		val2.mstold();
		result = val1 + val2;
		result.output();
		result = val1 - val2;
		result.output();
		amount = val2;
		result = val1 * val2;
		result.output();
		result = val1 / val2;
		result.output();
		result = val1 / amount;
		result.output();
		cout << "Continue'? (y/n)\n";
		cin.get();
	} while (std::cin.get() != 'n');
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++9 //works
class safearray {
private:
	int start;
	int end;
	int array[100];
public:
	safearray(int s, int l) : start(s), end(l) {}
	int& operator[](int n) {
		if (end - start > 100 || start > end) {
			cout << "\nIndex out of bounds";
			exit(1);
		}
		return array[n - start];
	}
	void getOutput() {
		for (int i = start; i < end; i++)
			array[i] = i;
		for (int i = start; i < end; i++)
			std::cout << "Element " << i << " is " << array[i] << "\n";
	}
};

void fun9()
{
	safearray value(100, 175);
	value.getOutput();
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++10 //wrode
works
class Polar {
private:
	double angle;
```

```
	double radius;
public:
	Polar() : angle(), radius() {}
	Polar(double a, double b) : angle(a), radius(b) {}
	void input() {
		cout << "Enter angle: \n";
		cin >> angle;
		if (angle < 0)
			if (angle > 360)
				exit(1);
		cout << "Enter radius: \n";
		cin >> radius;
	}
	void output() {
		cout << "The result: \n"
			<< "angle: " << angle
			<< "\nradius: " << radius;
	}

	Polar operator + (Polar coordinate2)
	{ //magic
		double x1 = radius * cos(angle);
		double y1 = radius * sin(angle);
		double x2 = coordinate2.radius * cos(coordinate2.angle);
		double y2 = coordinate2.radius * sin(coordinate2.angle);
		double X = x1 + x2;
		double Y = y1 + y2;
		double rad = sqrt(X * X + Y * Y);
		double ang = atan(Y / X);
		Polar buf;
		buf.radius = rad;
		buf.angle = ang;
		return buf;
	}
};

void fun10()
{
	Polar coordinate1, coordinate2, result_coordinate;
	coordinate1.input();
	coordinate2.input();
	result_coordinate = coordinate1 + coordinate2;
	result_coordinate.output();
}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++11 //bad code
//works
#include <iostream>
class Sterling {
private:
	long pounds;
	int shilling;
	int pens;
public:
	Sterling() : pounds(), shilling(), pens() {}
	Sterling(long x, int y, int z) : pounds(x), shilling(y), pens(z) {}

	void getSterling() {
		cout << "Enter pounds: \n";
		cin >> pounds;
		cout << "Enter shillings: \n";
		cin >> shilling;
		cout << "Enter pens: \n";
		cin >> pens;
```

```
	}
	void putSterling() const {
		cout << "Output is: " << pounds << "." << shilling << "." << pens << "\n";
	}
	Sterling operator + (Sterling value) {
		int sumpens = (pounds * 240 + shilling * 12 + pens) + (value.pounds * 240 +
			value.shilling * 12 + value.pens);
		long x = sumpens / 240;
		int y = sumpens % (20 * 12) / 12;
		int z = sumpens % (20 * 12) % 12;
		return Sterling(x, y, z);
	}
	Sterling operator - (Sterling value) {
		int sumpens = (pounds * 240 + shilling * 12 + pens) - (value.pounds * 240 +
			value.shilling * 12 + value.pens);
		long x = sumpens / 240;
		int y = sumpens % (20 * 12) / 12;
		int z = sumpens % (20 * 12) % 12;
		return Sterling(x, y, z);
	}
	Sterling operator * (double value) {
		int sumpens = (pounds * 240 + shilling * 12 + pens) * (value);
		long x = sumpens / 240;
		int y = sumpens % (20 * 12) / 12;
		int z = sumpens % (20 * 12) % 12;
		return Sterling(x, y, z);
	}
	Sterling operator / (Sterling value) {
		int sumpens = (pounds * 240 + shilling * 12 + pens) / (value.pounds * 240 +
value.shilling * 12 + value.pens);
		long x = sumpens / 240;
		int y = sumpens % (20 * 12) / 12;
		int z = sumpens % (20 * 12) % 12;
		return Sterling(x, y, z);
	}
	Sterling operator / (double value) {
		int sumpens = (pounds * 240 + shilling * 12 + pens) / (value);
		long x = sumpens / 240;
		int y = sumpens % (20 * 12) / 12;
		int z = sumpens % (20 * 12) % 12;
		return Sterling(x, y, z);
	}
	operator double() {
		float solution = pens + shilling * 12 + (pounds * 20) * 12;
		double amount = (solution / 2.4) / 100;
		return amount;
	}
};


void fun11()
{
	Sterling value1, value2, solution;
	double decimal;
	cout << "Enter decimal pounds: \n";
	cin >> decimal;
	value1.getSterling();
	value2.getSterling();
	solution = value1 + value2;
	solution.putSterling();
	solution = value1 - value2;
	solution.putSterling();
	solution = value1 * decimal;
	solution.putSterling();
	solution = value1 / value2;
```

```
	solution.putSterling();
	solution = value1 / decimal;
	solution.putSterling();
	decimal = solution;
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++12
class Sterling2;
class bMoney2;
class Sterling2 {
private:
	long pounds;
	int shilling;
	int pens;
public:
	Sterling2() : pounds(), shilling(), pens() {}
	Sterling2(long x, int y, int z) : pounds(x), shilling(y), pens(z) {}
	Sterling2(const bMoney2& mon);
	void getSterling() {
		cout << "Enter pounds: ";
		cin >> pounds;
		cout << "Enter shillings: ";
		cin >> shilling;
		cout << "Enter pens: ";
		cin >> pens;
	}
	void putSterling() const {
		cout << "Output is: " << pounds << "." << shilling << "." << pens << "\n";
	}
	long get_pounds() const {
		return pounds;
	}
	int get_shilling() const {
		return shilling;
	}
	int get_pens() const {
		return pens;
	}
};
class bMoney2 {
private:
	string strMon;
	long double capital;
public:
	bMoney2() : capital() {}
	bMoney2(const Sterling2 ster);
	void mstold() {
		cout << "Enter the amount in decimal point: ";
		getline(cin, strMon, '$');
		int length = strMon.length();
		int n = 0;
		string stro4ka;
		//preobrazuem
		for (int j = 0; j < length; j++)
			if (strMon[j] != ',' && strMon[j] != '$')
				stro4ka.push_back(strMon[j]); ;
		capital = stold(stro4ka);
	}
	void display()const {
		cout << std::setiosflags(std::ios::fixed)
			<< std::setiosflags(std::ios::showpoint)
			<< std::setprecision(2)
			<< "Output is: " << capital << '$';
	}
	long double get_number() const {
		return capital;
```

```
	}
};
Sterling2::Sterling2(const bMoney2& mon) {
	long double dollar = mon.get_number();
	int sumpens = (dollar / 50) * 240;
	pounds = sumpens / 240;
	shilling = sumpens % (20 * 12) / 12;
	pens = sumpens % (20 * 12) % 12;
}
bMoney2::bMoney2(const Sterling2 ster) {
	long x = ster.get_pounds();
	int y = ster.get_shilling();
	int z = ster.get_pens();
	capital = (x * 50) + ((50 / 20) * y) + ((50 / 240) * z);
}

void fun12()
{
	Sterling2 sterling, sterling1;
	bMoney2 dollars, dollars1;
	sterling.getSterling();
	dollars.mstold();
	sterling1 = dollars;
	dollars1 = sterling;
	sterling1.putSterling();
	dollars1.display();
}



int main()
{
	setlocale(LC_ALL, "Russian");
	int nomer;
	cout << "\nВведите номер задачи\n";
	cin >> nomer;
	switch (nomer) {
	case 5:
		fun5();
		break;
	case 6:
		fun6();
		break;
	case 7:
		fun7();
		break;
	case 8:
		fun8();
		break;
	case 9:
		fun9();
		break;
	case 10:
		fun10();
		break;
	case 11:
		fun11();
		break;
	case 12:
		fun12();
		break;
	}

	return 0;
}
```