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 № 4

Дисциплина:

Объектно-ориентированное программирование

Тема:

Функции.

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

Дата, подпись 16.03.22
(Дата) (Подпись)

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

Замечания: _____

_____

_____

Москва

2021

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

Задания:

5. Напишите функцию с именем hms_to_secs(), имеющую три аргумента типа int: часы, минуты и секунды. Функция должна возвращать эквивалент переданного ей временного значения в секундах (типа long). Создайте программу, которая будет циклически запрашивать у пользователя ввод значения часов, минут и секунд и выводить результат работы функции на экран.

6. Модифицируйте программу, описанную в упражнении 11 предыдущей лабораторной работы «Структуры», складывающую два структурных значения типа time. Теперь программа должна включать в себя две функции. Первая, time_to_secs(), принимает в качестве аргумента значение типа time и возвращает эквивалентное значение в секундах типа long. Вторая, secs_to_time(), в качестве аргумента принимает число секунд, имеющее тип long, а возвращает эквивалентное значение типа time.

7. Взяв в качестве основы функцию power() из упражнения 2, работающую только со значением типа double, создайте перегруженные функции с этим же именем, принимающими в качестве аргумента значения типа char, int, long и float. Напишите программу, вызывающую функцию power() со всеми возможными типами аргументов.

8. Напишите функцию с именем swap(), обменивающую значениями два своих аргумента типа int (обратите внимание, что изменяться должны значения переменных из вызывающей программы, а не локальных переменных функции). Выберите способ передачи аргументов. Напишите вызывающую программу main(), использующую данную функцию.

9. Переработайте программу из упражнения 8 так, чтобы функция swap() принимала в

качестве аргументов значения типа time (см. упражнение 6).

10. Напишите функцию, которая при каждом вызове будет выводить на экран количество раз, которое она вызывалась ранее. Напишите программу, которая будет вызывать данную функцию не менее 10 раз. Попробуйте реализовать данную функцию двумя различными способами: с использованием глобальной переменной и статической локальной переменной для хранения числа вызовов функции. Какой из способов предпочтительней? Почему для решения задачи нельзя использовать обычную локальную переменную?

11. Напишите программу, использующую структуру sterling, которая описана в упражнении 10 лабораторной работы 2 «Структуры». Программа должна получать от пользователя значения двух денежных сумм, выраженных в фунтах, шиллингах и пенсах, складывать эти значения и выводить результат на экран в том же формате. Необходимо разработать три функции. Первая из них должна получать от пользователя число фунтов, шиллингов и пенсов и возвращать соответствующее значение типа sterling. Вторая функция должна принимать в качестве аргументов два значения типа sterling, складывать их и возвращать значение, также имеющее тип sterling. Третья функция должна принимать аргумент типа sterling и выводить его значение на экран.

12. Модифицируйте калькулятор, созданный в упражнении 12 лабораторной работы 2 «Структуры», так, чтобы каждая арифметическая операция выполнялась с помощью функции. Функции могут называться fadd(), fsub(), fmul() и fdiv(). Каждая из функций должна принимать два структурных аргумента типа fraction и возвращать значение того же типа.

Код:

```
#include <iostream>
#include <iomanip>
using namespace std;

struct Time
{
	int hours;
	int minutes;
	int seconds;
};

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6
int hour, mins, secs;

unsigned long hms_to_secs(int h, int m, int s)
{
	unsigned long seconds = h * 60 * 60 + m * 60 + s;
	return seconds;
}
```

```
void fun5() {
        unsigned long seconds;
        char ch;
        do
        {
                cout << "Enter the hours" << endl; cin >> hour;
                cout << "Enter the minutes" << endl; cin >> mins;
                cout << "Enter the seconds" << endl; cin >> secs;
                seconds = hms_to_secs(hour, mins, secs);
                cout << "all seconds: " << seconds << endl;
                cout << "Repeat? (y/n) ";
                cin >> ch;
        } while (ch != 'n');
}
//-------------------------------------------------------------------------------6

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6
Time t1, t2, tsum;
long time_to_secs(Time t) {
        long totalsecs = (t.hours * 3600) + (t.minutes * 60) + t.seconds;
        return totalsecs;
}
void secs_to_time(long totalsecs) {
        tsum.hours = totalsecs / (60 * 60);
        tsum.minutes = totalsecs % (60 * 60) / 60;
        tsum.seconds = totalsecs % (60 * 60) % 60;
}
void fun6(){
        char ch;
        cout << "Enter time 1 (hours:minutes:seconds)" << endl;
        cin >> t1.hours >> ch >> t1.minutes >> ch >> t1.seconds;
        cout << "Enter time 1 (hours:minutes:seconds)" << endl;
        cin >> t2.hours >> ch >> t2.minutes >> ch >> t2.seconds;
        long totalsecs = time_to_secs(t1) + time_to_secs(t2);
        secs_to_time(totalsecs);
        cout << "Total time: " << tsum.hours << ch << tsum.minutes << ch << tsum.seconds << endl;
}
//-------------------------------------------------------------------------------6

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++7
double power(double n, int p = 2);
double power(char n, int p = 2);
double power(int n, int p = 2);
double power(long n, int p = 2);
double power(float n, int p = 2);

void fun7() {
        cout << "result (double): " << power(3.563678456434, 5) << endl;
        cout << "result (char): " << power('r', 5) << endl;
        cout << "result (int): " << power(3, 5) << endl;
        cout << "result (long): " << power(3L, 5) << endl;
        cout << "result (float): " << power(3.3F, 5) << endl;
}
double power(double n, int p){
        double res = n;
        for (int j = 0; j < p; j++)
                res *= n;
        return res;
}
double power(char n, int p){
        char res = n;
        for (int j = 0; j < p; j++)
                res *= n;
        return res;
}
double power(int n, int p){
        int res = n;
        for (int j = 0; j < p; j++)
                res *= n;
        return res;
}
double power(long n, int p){
        long res = n;
        for (int j = 0; j < p; j++)
                res *= n;
```

```
        return res;
}
double power(float n, int p){
        float res = n;
        for (int j = 0; j < p; j++)
                res *= n;
        return res;
}
//-------------------------------------------------------------------------------7

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++8
void swap(int& num1, int& num2){
        int temp = num1;
        num1 = num2;
        num2 = temp;
}
void fun8() {
        int n1, n2;
        cout << "Enter first number" << endl;
        cin >> n1;
        cout << "Enter second number" << endl;
        cin >> n2;
        swap(n1, n2);
        cout << n1 << " " << n2 << endl;
}
//-------------------------------------------------------------------------------8

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++9
void swap(Time num1, Time num2)
{
        Time buf = num1;
        num1 = num2;
        num2 = buf;
}
void fun9() {
        int n1, n2;
        cout << "Enter first number" << endl;
        cin >> n1;
        cout << "Enter second number" << endl;
        cin >> n2;
        swap(n1, n2);
        cout << n1 << " " << n2 << endl;
}
//-------------------------------------------------------------------------------9

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++10
//This is first variant:
/*
int times = 1;
void call() {
        cout << "Was called " << times << " times" << endl;
        times++;
}

void fun10() {
        for (int i = 0; i < 20; i++) {
                call();
        }
}
*/

//This is second variant:
void call() {
        static int times;
        cout << "Was called " << times << " times" << endl;
        times++;
}

void fun10() {
        for (int i = 0; i < 20; i++) {
                call();
        }
}
//-------------------------------------------------------------------------------10

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++11
struct sterling
```

```
{
        int pounds;
        int shilling;
        int pens;
};
sterling s1, s2, sum;
char ch;

sterling input(sterling str)
{
        cout << "Enter (pounds.shilling.pens)" << endl;
        cin >> str.pounds >> ch >> str.shilling >> ch >> str.pens;
        return str;
}

sterling operating(sterling str1, sterling str2)
{
        sterling result;
        int respens = (str1.pounds * 20 * 12 + str1.shilling * 12 + str1.pens);
        respens += (str2.pounds * 20 * 12 + str2.shilling * 12 + str2.pens);
        result.pounds = respens / (20 * 12);
        result.shilling = respens % (20 * 12) / 12;
        result.pens = respens % (20 * 12) % 12;
        return result;
}

void output(sterling sum)
{
        cout << "Result: " << sum.pounds << ch << sum.shilling << ch << sum.pens << endl;
}

void fun11() {
        s1 = input(s1);
        s2 = input(s2);
        sum = operating(s1, s2);
        output(sum);
}
//-------------------------------------------------------------------------------11

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++12
struct fraction
{
        int numerator;
        int denominator;
};
fraction f1, f2, sumf;

fraction fadd(fraction f1, fraction f2)
{
        sumf.numerator = f1.numerator * f2.denominator + f1.denominator * f2.numerator;
        sumf.denominator = f1.denominator * f2.denominator;
        return sumf;
}

fraction fsub(fraction f1, fraction f2)
{
        sumf.numerator = f1.numerator * f2.denominator - f1.denominator * f2.numerator;
        sumf.denominator = f1.denominator * f2.denominator;
        return sumf;
}

fraction fmul(fraction f1, fraction f2)
{
        sumf.numerator = f1.numerator * f2.denominator;
        sumf.denominator = f1.denominator * f2.denominator;
        return sumf;
}

fraction fdiv(fraction f1, fraction f2)
{
        sumf.numerator = f1.numerator * f2.denominator;
        sumf.denominator = f1.denominator * f2.numerator;
        return sumf;
}

void fun12() {
        char ch = '/', zn;
```

```
        do
        {
                cout << "enter first fraction" << endl;
                cin >> f1.numerator >> ch >> f1.denominator;
                cout << "Enter symbol" << endl;
                cin >> zn;
                cout << "enter second fraction" << endl;
                cin >> f2.numerator >> ch >> f2.denominator;
                switch (zn)
                {
                case '+': sumf = fadd(f1, f2); break;
                case '-': sumf = fsub(f1, f2); break;
                case '*': sumf = fmul(f1, f2); break;
                case '/': sumf = fdiv(f1, f2); break;
                }
                cout << sumf.numerator << ch << sumf.denominator << endl;
                cout << "Repeat? (y,n)" << endl;
                cin >> ch;
        } while (ch != 'n');
}

int main(){
        int nomer;
        cout << "\nEnter # of exercise \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;
}
```