МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №12

по дисциплине: Основы программирования
тема: «Структуры. Функции для работы со структурами»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Код-ревьер: ст. группы ПВ-223
Голуцкий Георгий Юрьевич

Белгород 2022 г.

Лабораторная работа № 12

Вариант №2

Содержание отчёта:

- Тема лабораторной работы.
- Цель лабораторной работы.
- Решения задач.
  - Текст задания.
  - Исходный код.
  - Решения задач с двумя звёздочками являются необязательными.
- Вывод по работе.

Тема лабораторной работы: Структуры. Функции для работы со структурами
Цель лабораторной работы: получение навыков написания функций для решения задач со структурами.
Решения задач:

1. Задача №1

1. Описать структуру Point в соответствии с условием задания.
2. (a) Объявите переменную типа Point с инициализацией.
3. (b) Реализуйте функцию ввода структуры Point.
4. (c) Реализуйте функцию вывода структуры Point.
5. (d) Объявите с инициализацией точку p1. Объявите точку p2 и инициализируйте точку p2 координатами точки p1.
6. (e) Создайте массив структур размера N=3. Реализуйте функции для его ввода inputPoints и вывода outputPoints.
7. (f) Реализуйте функцию, которая принимает на вход две структуры типа Point и возвращает точку, находящуюся посередине между точками p1 и p2.
8. (g) Реализуйте функцию arePointsEqual, которая возвращает значение ’истина’, если точки совпадают.
9. (h) Реализуйте функцию, которая возвращает значение ’истина’, если точка p3 лежит ровно посередине между точками p1 и p2.
10. (i) Реализуйте функцию swapCoordinates которая меняет значения координат x и y структуры типа Point.
11. (j) Реализуйте функцию swapPoints которая обменивает две точки.
12. (k) Напишите фрагмент кода, в котором выделяется память под массив структур размера N = 3, после чего укажите инструкцию освобождения памяти.
13. (l) Реализуйте функцию, которая находит расстояние между двумя точками.
14. (m) ** Опишите функцию-компоратор для qsort, которая сортирует массив точек размера N = 3 по увеличению координаты x, а при их равенстве – по координате y.
15. (n) ** Опишите функцию-компоратор для qsort, которая сортирует массив точек размера N = 3 по увеличению расстояния до начала координат.

main.c

```c
#include "../libs/alg/lab12/point/point.h"

#define N 3

int main() {
    Point p1 = {};
    inputPoint(&p1);

    outputPoint(p1);
    printf("\n");

    Point p2 = p1;
    //Point p[N];
    Point *p = (Point*) malloc(sizeof(Point) * N);
    inputPoints(p, N);

    qsort(p, N, sizeof(p[0]), coordinatesComparator);

    outputPoints(p, N);

    qsort(p, N, sizeof(p[0]), distanceComparator);

    outputPoints(p, N);

    free(p);

    return 0;
}
```

point.h

```c
#ifndef PROGRAMMING_AND_ALGORITHMIZATION_BASICS_POINT_H
#define PROGRAMMING_AND_ALGORITHMIZATION_BASICS_POINT_H

#include <stdio.h>
#include "../../alg.h"

#define EPS 0.0000001
#define COORDINATES_START (Point) {0, 0}

typedef struct Point {
    double x;
    double y;
} Point;

// вводит x и y в точку по адресу p
void inputPoint(Point *p);

// выводит x и y структуры p
void outputPoint(Point p);

// вводит x и y в массив структур p размером arraySize
void inputPoints(Point *p, size_t arraySize);

// выводит x и y структур в массиве p размером arraySize
void outputPoints(Point *p, size_t arraySize);
```

```c
// возвращает точку находящуюся посередине точек p1 и p2
Point getMiddlePoint(Point p1, Point p2);

// возвращает "истина", если точки p1 и p2 равны, иначе - "ложь"
int arePointsEqual(Point p1, Point p2);

// возвращает "истина", если точка p3 находится посередине между точками p1 и p2,
иначе - "ложь"
int isPointBetween(Point p1, Point p2, Point p3);

// обменивает значения по адресам a и b размером n
void swap(void *a, void *b, size_t n);

// меняет местами координаты x и y точки по адресу p
void swapCoordinates(Point *p);

// меняет точки по адресам p1 и p2
void swapPoints(Point *p1, Point *p2);

// возвращает расстояние между точкой p1 и p2
double getDistance(Point p1, Point p2);

// возвращает
// 0,  если точки p1 и p2 равны, иначе
// 1,  если координаты x точки p1 больше координаты x точки p2
//     или координаты x точек p1 и p2 равны, а координата y точки p1
//     больше чем y точки p2,
// -1  во всех остальных случаях
int coordinatesComparator(const void *p1, const void *p2);

// возвращает
// 0,  если расстояние точек p1 и p2 до начала координат равно, иначе
// 1,  если расстояние точки p1 до центра координат больше расстояния
//     точки p2 до центра координат
// -1  во всех остальных случаях
int distanceComparator(const void *p1, const void *p2);

#endif //PROGRAMMING_AND_ALGORITHMIZATION_BASICS_POINT_H
```

point.c

```c
#include "point.h"

void inputPoint(Point *p) {
    scanf("%lf %lf", &p->x, &p->y);
}

void outputPoint(Point p) {
    printf("(%.3lf %.3lf)", p.x, p.y);
}

void inputPoints(Point *p, size_t arraySize) {
    for (size_t i = 0; i < arraySize; i++)
        inputPoint(p + i);
}

void outputPoints(Point *p, size_t arraySize) {
    for (size_t i = 0; i < arraySize; i++) {
```

```
        outputPoint(p[i]);
        printf("\n");
    }
}

Point getMiddlePoint(Point p1, Point p2) {
    return (Point) {(p1.x + p2.x) / 2, (p1.y + p2.y) / 2};
}

int arePointsEqual(Point p1, Point p2) {
    return fcompare(p1.x, p2.x) && fcompare(p1.y, p2.y);
}

int isPointBetween(Point p1, Point p2, Point p3) {
    return arePointsEqual(getMiddlePoint(p1, p2), p3);
}

void swap(void *a, void *b, size_t n) {
    char *pA = (char *) a;
    char *pB = (char *) b;

    for (size_t i = 0; i < n; i++, pA++, pB++) {
        char t = *pA;
        *pA = *pB;
        *pB = t;
    }
}

void swapCoordinates(Point *p) {
    swap(&p->x, &p->y, sizeof(p->x));
}

void swapPoints(Point *p1, Point *p2) {
    swap(p1, p2, sizeof(*p1));
}

double getDistance(Point p1, Point p2) {
    double distX = p1.x - p2.x;
    double distY = p1.y - p2.y;

    return sqrt(distX * distX + distY * distY);
}

int coordinatesComparator(const void *p1, const void *p2) {
    Point *a = (Point *) p1;
    Point *b = (Point *) p2;

    if (a->x != b->x)
        return (a->x - b->x) > 0 ? 1 : -1;

    if (a->y != b->y)
        return (a->y - b->y) > 0 ? 1 : -1;

    return 0;
}

int distanceComparator(const void *p1, const void *p2) {
    Point *a = (Point *) p1;
```

```
    Point *b = (Point *) p2;

    double distA = getDistance(*a, COORDINATES_START);
    double distB = getDistance(*b, COORDINATES_START);

    if (distA == distB)
        return 0;

    return (distA - distB) > 0 ? 1 : -1;
}
```

2. Задача №2

1. Опишите структуру в соответствии с условием задания.
2. (a) Объявите с инициализацией переменную типа Circle.
3. (b) Объявите с инициализацией массив из двух структур типа Circle.
4. (c) Реализуйте функцию inputCircle ввода структуры Circle.
5. (d) Реализуйте функцию inputCircles ввода массив структур Circle.
6. (e) Реализуйте функцию outputCircle вывода структуры Circle.
7. (f) Реализуйте функцию outputCircles вывода массива структур Circle.
8. (g) Реализуйте функцию hasOneOuterIntersection, которая возвращает значение ’истина’, если окружность c1 касается внешним образом окружности c2. Заголовок: int hasOneOuterIntersection(Circle c1, Circle c2).

main.c

```
#include "../libs/alg/lab12/circle/circle.h"

#define N 2

int main() {
    Circle c = {(Point) {3, 4}, 5};

    Circle cArray[2] = {(Circle) {(Point) {6, -12.44}, 98},
                        (Circle) {(Point) {42, -42}, 42.42}};
}
```

point.h

```
#ifndef PROGRAMMING_AND_ALGORITHMIZATION_BASICS_CIRCLE_H
#define PROGRAMMING_AND_ALGORITHMIZATION_BASICS_CIRCLE_H

#include "../point/point.h"

typedef struct Circle {
    Point center;
    double r;
} Circle;

// вводит центр и радиус окружности по адресу a
void inputCircle(Circle *a);

// вводит центр и радиус в массив окружностей a размером n
void inputCircles(Circle *a, size_t n);

// выводит центр и радиус окружности a
```

```c
void outputCircle(Circle a);

// вводит центры и радиусы массива окружностей a размером n
void outputCircles(Circle *a, size_t n);

// возвращает "истина", если c1 и c2 касаются внешним образом,
// иначе - "ложь"
bool hasOneOuterIntersection(Circle c1, Circle c2);

#endif //PROGRAMMING_AND_ALGORITHMIZATION_BASICS_CIRCLE_H
```

point.c

```c
#include "circle.h"

void inputCircle(Circle *a) {
    inputPoint(&a->center);
    scanf("%lf", &a->r);
}

void inputCircles(Circle *a, size_t n) {
    for (size_t i = 0; i < n; i++) {
        inputCircle(a + i);
    }
}

void outputCircle(Circle a) {
    outputPoint(a.center);
    printf(" r = %.3lf", a.r);
}

void outputCircles(Circle *a, size_t n) {
    for (size_t i = 0; i < n; i++) {
        outputCircle(a[i]);
        printf("\n");
    }
}

bool hasOneOuterIntersection(Circle c1, Circle c2) {
    return fcompare(c1.r + c2.r, getDistance(c1.center, c2.center));
}
```

Вывод: в ходе выполнения лабораторной работы получены навыки написания функций для решения задач со структурами.