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 №19.12
по дисциплине: «Структуры. Функции для работы со структурами»

Выполнил/а: ст. группы ВТ-231
Кисиль Николай Владимирович

Проверили:
Черников Сергей Викторович
Новожен Никита Викторович

Белгород, 2023 г.

Цель работы: получение навыков написания функций для решения задач со структурами.

Содержание работы

Задача 1: Опишем структуру Point

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

a) Объявите структуру Point с инициализацией

```
int main() {
    Point point = {1, 2};

    return 0;
}
```

b) Реализуйте функцию ввода структуры Point

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

c) Реализуйте функцию вывода структуры Point.

```
void outputPoint(Point *p) {
    printf("%.3f %.3f\n", p->x, p->y);
};
```

d) Создайте две точки p1 и p2. Проведите их инициализацию в коде. Выполните присваивание точки p2 точке p1.

```
int main() {
    Point p1 = {1, 2};
    Point p2 = p1;

    return 0;
}
```

e) Создайте массив структур размера N=3. Реализуйте функции для его ввода inputPoints и вывода outputPoints.

```
void inputPoints(Point *p, int n) {
    for (int i = 0; i < n; i++) {
        inputPoint(&p[i]);
    }
}

void outputPoints(Point *p, int n) {
    for (int i = 0; i < n; i++) {
        outputPoint(&p[i]);
    }
}
```

f) Реализуйте функцию, которая принимает на вход две структуры типа Point и возвращает точку, находящуюся посередине между точками p1 и p2.

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

g) Реализуйте функцию isEqualPoint, которая возвращает значение ’истина’, если точки совпадают (с погрешностью не более DBL_EPSILON, определённой в)

```
int isEqualPoint(Point p1, Point p2) {
    int is_equal = 1;
    if (fabs(p1.x - p2.x) > DBL_EPSILON || fabs(p1.y - p2.y) >
DBL_EPSILON) {
        is_equal = 0;
    }
    return is_equal;
}
```

h) Реализуйте функцию, которая возвращает значение ’истина’, если точка p3 лежит ровно посередине между точками p1 и p2.

```
int isBetween(Point p1, Point p2, Point p3) {
    Point middle = getMiddlePoint(p1, p2);
    return isEqualPoint(middle, p3);
}
```

i) Реализуйте функцию swapCoordinates которая меняет значения координат x и y структуры типа Point

```
void swapCoordinates(Point *p) {
    SWAP(double, p->x, p->y);
}
```

j) Реализуйте функцию swapPoints которая обменивает две точки

```
void swapPoints(Point *p1, Point *p2) {
    SWAP(double, p1->x, p2->x);
    SWAP(double, p1->y, p2->y);
}
```

k) Напишите фрагмент кода, в котором выделяется память под массив структур размера N = 3, после чего укажите инструкцию освобождения памяти.

```
int main() {
    int N = 3;

    Point *points = (Point *)malloc(N * sizeof(Point));

    free(points);
    return 0;
}
```

l) Реализуйте функцию, которая находит расстояние между двумя точками

```
double getDistance(Point p1, Point p2) {
    double distance = sqrt(pow((p2.x - p1.x), 2) + pow((p2.y - p1.y), 2));
    return distance;
}
```

m) Опишите функцию-компоратор для qsort, которая сортирует массив точек размера N = 3 по увеличению координаты x, а при их равенстве – по координате y.

```
int comparePoint(const void *a, const void *b) {

    const Point *p1 = (const Point *) a;
    const Point *p2 = (const Point *) b;

    if (p1->x < p2->x) {
        return -1;
    } else if (p1->x > p2->x) {
        return 1;
    } else {
        return (p1->y < p2->y) ? -1 : p1->y > p2->y;
    }
}
```

n) Опишите функцию-компоратор для qsort, которая сортирует массив точек размера N = 3 по увеличению расстояния до начала координат

```
double distance(const Point *p) {
    return sqrt(p->x * p->x + p->y * p->y);
}

int comparePointsByDistance(const void *a, const void *b) {

    const Point *p1 = (const Point *) a;
    const Point *p2 = (const Point *) b;

    double distanceA = distance(p1);
    double distanceB = distance(p2);

    return distanceA < distanceB ? -1 : distanceA > distanceB;

}
```

Задача 2: Опишем структуру Line, которая задаёт линию на плоскости уравнением $ax + by + c = 0$

```
struct Line {
    double a;
    double b;
    double c;
};

typedef struct Line Line;
```

a) Реализуйте функцию inputLine ввода структуры Line

```
void inputLine(Line *line) {
    scanf("%lf %lf %lf", &line->a, &line->b, &line->c);
}
```

b) Инициализируйте структуру типа Line при объявлении

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    Line line = {1, 2, 3};
}
```

c) Реализуйте функцию getLine которая возвращает прямую по координатам точек

```
Line getLineByPoints(Point p1, Point p2) {
    Line line;

    line.a = p2.y - p1.y;
    line.b = p1.x - p2.x;
    line.c = p2.x * p1.y - p1.x * p2.y;

    return line;
}
```

d) Напишите код для создания линии из точек, без явного создания структур p1 и p2.

```
Line getLine(double x1, double y1, double x2, double y2) {
    Line line;

    line.a = y2 - y1;
    line.b = x1 - x2;
    line.c = x2 * y1 - x1 * y2;

    return line;
}
```

e) Реализуйте функцию outputLineEquation вывода уравнения прямой Line

```
void outputLineEquation(Line Line) {
    printf("%.2lfx%+.2lfy%+.2lf = 0", Line.a, Line.b, Line.c);
}
```

f) Реализуйте функцию isParallel, которая возвращает значение ’истина’ если прямые Line1 и Line2 параллельны, ’ложь’ – в противном случае.

```
int isParallel(Line l1, Line l2) {
    return (l1.a * l2.b) == (l1.b * l2.a);
}
```

g) Реализуйте функцию isPerpendicular, которая возвращает значение ’истина’ если прямые l1 и l2 перпендикулярны, ’ложь’ – в противном случае

```
int isPerpendicular(Line l1, Line l2) {
    return (l1.a * l2.a + l1.b * l2.b) == -1.0;
}
```

h) Определите, есть ли среди данных n прямых на плоскости (n – const) параллельные

```
int hasParallelLines(Line *lines, size_t n) {
    for (size_t i = 0; i < n - 1; i++) {
        for (size_t j = i + 1; j < n; j++) {
            if (isParallel(lines[i], lines[j])) {
                return 1;
            }
        }
    }
    return 0;
}
```

i) Реализуйте функцию printIntersectionPoint, которая выводит точку пересечения прямых l1 и l2. Если точек пересечения нет – проинформируйте пользователя

```
void printIntersectionPoint(Line l1, Line l2) {
    double det = l1.a * l2.b - l2.a * l1.b;

    if (det != 0) {
        double x = (l1.b * l2.c - l2.b * l1.c) / det;
        double y = (l2.a * l1.c - l1.a * l2.c) / det;

        printf("%lf %lf", x, y);
    } else {
        printf("Точек пересечения нет.");
    }
}
```

Задача 3: Опишите структуру Circle, которая задаёт окружность посредством центра окружности center(x0, y0), и радиуса r.

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

typedef struct Circle Circle;
```

a) Объявите с инициализацией структуру типа Circle

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    Point center = {1, 2};
    Circle circle = {center, 3};

    return 0;
}
```

b) Объявите с инициализацией массив из двух структур типа Circle

```
int main() {
    SetConsoleOutputCP(CP_UTF8);

    Circle circles[2] = {
            1, 2, 3,
            4, 5, 6
    };

    return 0;
}
```

c) Реализуйте функцию inputCircle ввода структуры Circle

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

d) Реализуйте функцию inputCircles ввода массива структур Circle.

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

e) Реализуйте функцию outputCircle вывода структуры Circle

```
void outputCircle(Circle a) {
    printf("%lf %lf %lf", a.center.x, a.center.y, a.r);
}
```

f) Реализуйте функцию outputCircles вывода массива структур Circle

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

g) Реализуйте функцию hasOneOuterIntersection, которая возвращает значение ’истина’, если окружность c1 касается внешним образом окружности c2.

```
int hasOneOuterIntersection(Circle c1, Circle c2) {
    double centersDistance = getDistance(c1.center, c2.center);
    return centersDistance == c1.r + c2.r;
}
```

h) Вводится массив из n окружностей (n вводится с клавиатуры). Реализуйте функцию, которая возвращает окружность, в которой лежит наибольшее количество окружностей. Если таких несколько – вернуть окружность с наименьшим радиусом.

```
int isContainingCircle(Circle c1, Circle c2) {
    double distance = getDistance(c1.center, c2.center);
    return distance + c2.r <= c1.r;
}

Circle maxContainingCircle(Circle * circles, size_t n) {
    int max_count = 0;
    double min_radius = INT_MAX;
    Circle result;

    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if (i != j && isContainingCircle(circles[i], circles[j])) {
                count++;
            }
        }

        if (count > max_count || (count == max_count && circles[i].r <
min_radius)) {
            max_count = count;
            min_radius = circles[i].r;
            result = circles[i];
        }
    }

    return result;
}
```

i) * Вводится массив из n окружностей (n вводится с клавиатуры). Реализуйте функцию сортировки окружностей, по неубыванию количества лежащих в ней окружностей. При равенстве количества последнего показателя, отсортировать по неубыванию радиуса

```
int countEnclosingCircles(const Circle* circles) {
    int count = 0;
    for (int i = 0; i < sizeof(&circles - 1); i++) {
        for (int j = 0; j < sizeof(&circles - 1); j++) {
            if (i != j && isContainingCircle(circles[i], circles[j])) {
                count++;
            }
        };
    }
    return count;
}

int compareCircles(const void * a, const void * b) {

    const Circle* circleA = (const Circle*)a;
    const Circle* circleB = (const Circle*)b;

    int countA = countEnclosingCircles(circleA);
    int countB = countEnclosingCircles(circleB);

    if (countA != countB) {
        return countA - countB;
    } else {
        return (circleA->r > circleB->r) - (circleA->r < circleB->r);
    }
}
```

Задача 4: Опишем структуру Fraction.

```
struct Fraction {
    int numerator;
    int denominator;
};

typedef struct Fraction Fraction;
```

a) Реализуйте функцию inputFraction ввода структуры Fraction

```
void inputFraction(Fraction *f) {
    scanf("%d/%d", &f->numerator, &f->denominator);
}
```

b) Реализуйте функцию inputFractions ввода массива структур Fraction

```
void inputFractions(Fraction *f, size_t n) {
    for (size_t i = 0; i < n; i++) {
        inputFraction(&f[i]);
    }
}
```

c) Реализуйте функцию outputFraction вывода структуры Fraction в формате ’5/7’.

```
void outputFraction(Fraction f) {
    printf("%d/%d", f.numerator, f.denominator);
}
```

d) Реализуйте функцию outputFractions вывода массива структур Fraction

```
void outputFractions(Fraction *f, size_t n) {
    for (size_t i = 0; i < n; i++) {
        outputFraction(f[i]);
    }
}
```

e) Реализуйте функцию gcd возвращающую наибольший общий делитель

```
int gcd(int a, int b) {
    while (a != 0 && b != 0) {
        if (a > b) {
            a %= b;
        } else {
            b %= a;
        }
    }
    return a + b;
}
```

f) Реализуйте функцию lcm возвращающую наименьшее общее кратное

```
int lcm(int a, int b) {
    if (a == 0 || b == 0) {
        return 0;
    }

    return abs(a * b) / gcd(a, b);
}
```

g) Реализуйте функцию simpleFraction для сокращения дроби a

```
void simpleFraction(Fraction *f) {
    int divider = gcd(f->numerator, f->denominator);
    f->numerator = f->numerator / divider;
    f->denominator = f->denominator / divider;
}
```

h) Реализуйте функцию mulFractions умножения двух дробей a и b

```
Fraction mulFractions(Fraction f1, Fraction f2) {
    Fraction result;

    result.numerator = f1.numerator * f2.numerator;
    result.denominator = f1.denominator * f2.denominator;

    simpleFraction(&result);

    return result;
}
```

i) Реализуйте функцию divFractions деления двух дробей a и b.

```
Fraction divFractions(Fraction f1, Fraction f2) {
    Fraction inverted_f2 = {f2.denominator, f2.numerator};
    return mulFractions(f1, inverted_f2);
}
```

j) Реализуйте функцию addFractions сложения двух дробей a и b

```
Fraction addFractions(Fraction f1, Fraction f2) {
    Fraction result;

    simpleFraction(&f1);
    simpleFraction(&f2);

    int denominator = lcm(f1.denominator, f2.denominator);

    result.numerator = f1.numerator * (denominator / f1.denominator) +
f2.numerator * (denominator / f2.denominator);
    result.denominator = denominator;

    return result;
}
```

k) Реализуйте функцию subFractions вычитания двух дробей a и b.

```
Fraction subFractions(Fraction f1, Fraction f2) {
    Fraction new_f2;
    new_f2.numerator = -f2.numerator;
    new_f2.denominator = f2.denominator;

    return addFractions(f1, new_f2);
}
```

l) Реализуйте функцию для поиска суммы n дробей

```
Fraction sumFractions(Fraction *f, size_t n) {
    Fraction result = {0, 1};
    for (size_t i = 0; i < n; i++) {
        result = addFractions(result, f[i]);
    }

    return result;
}
```

Задача 5: * Дан массив записей. Каждая запись содержит сведения о студенте группы: фамилию и оценки по 5 предметам. Удалить записи о студентах, имеющих более одной неудовлетворительной оценки. Вывести фамилии оставшихся студентов

Код:

```c
#include <stdio.h>
#define N_MARKS 5

struct Student {
    char surname[20];
    int marks[N_MARKS];
};

typedef struct Student Student;

void inputStudent(Student *s) {
    scanf("%s", s->surname);
    for (size_t i = 0; i < N_MARKS; i++) {
        scanf("%d", &s->marks[i]);
    }
}

void inputStudents(Student *s, size_t n) {
    for (size_t i = 0; i < n; i++) {
        inputStudent(&s[i]);
    }
}

int isGoodStudent(Student s) {
    int is_good_student = 1;
    for (size_t i = 0; i < N_MARKS; i++) {
        if (s.marks[i] <= 2) {
            is_good_student = 0;
        }
    }
    return is_good_student;
}

void deleteBadStudent(Student *s, size_t n) {
    for (size_t i = 0; i < n; i++) {
        if (isGoodStudent(s[i]) == 1) {
            printf("%s ", s[i].surname);
        }
    }
}

int main() {
    int n;
    scanf("%d", &n);
    Student s[n];

    inputStudents(s, n);
    deleteBadStudent(s, n);
}
```

Задача 6: * Дан массив, каждый элемент которого представляет собой временную отметку в рамках одного дня (запись из трех полей: часы, минуты и секунды). Упорядочить отметки в хронологическом порядке.

Код:

```c
#include <stdio.h>
#include <stdlib.h>

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

typedef struct Time Time;

void inputTime(Time *t) {
    scanf("%02d:%02d:%02d", &t->hours, &t->minutes, &t->seconds);
}

void inputTimes(Time *t, size_t n) {
    for (size_t i = 0; i < n; i++) {
        inputTime(&t[i]);
    }
}

void outputTime(Time t) {
    printf("%02d:%02d:%02d\n", t.hours, t.minutes, t.seconds);
}

void outputTimes(Time *t, size_t n) {
    for (size_t i = 0; i < n; i++) {
        outputTime(t[i]);
    }
}

int compareTime(const void *a, const void *b) {

    const Time *t1 = (const Time *)a;
    const Time *t2 = (const Time *)b;

    if (t1->hours != t2->hours) {
        return t1->hours - t2->hours;
    } else if (t1->minutes != t2->minutes) {
        return t1->minutes - t2->minutes;
    } else {
        return t1->seconds - t2->seconds;
    }
}


int main() {
    int n;
    scanf("%d", &n);

    Time times[n];
    inputTimes(times, n);
```

```
    qsort(times, n, sizeof(Time), compareTime);

    outputTimes(times, n);
    return 0;
}
```

Задача 7: * Определить время, прошедшее от t1 до t2. Время предоставлено записью из трех полей: часы, минуты, секунды.

Код:

```c
#include <stdio.h>

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

typedef struct Time Time;

void inputTime(Time *t) {
    scanf("%02d:%02d:%02d", &t->hours, &t->minutes, &t->seconds);
}

void outputTime(Time t) {
    printf("%02d:%02d:%02d\n", t.hours, t.minutes, t.seconds);
}


Time getTimeDifference(Time t1, Time t2) {
    Time result;

    int second_diff = t2.seconds - t1.seconds;
    if (second_diff < 0) {
        second_diff += 60;
        t2.minutes--;
    }

    result.seconds = second_diff;

    int minutes_diff = t2.minutes - t1.minutes;
    if (minutes_diff < 0) {
        minutes_diff += 60;
        t2.hours--;
    }

    result.minutes = minutes_diff;

    int hours_diff = t2.hours - t1.hours;
    if (hours_diff < 0) {
        hours_diff += 24;
    }

    result.hours = hours_diff;

    return result;
}


int main() {
    Time t1, t2;
    inputTime(&t1);
    inputTime(&t2);
    outputTime(getTimeDifference(t1, t2));
    return 0;
}
```

Вывод: получили навыки написания структур для решения задач.