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 №4е

по дисциплине: Основы программирования

тема: «Структуры. Функции для работы со структурами»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Белгород 2021 г.

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

Содержание отчета:
- Тема лабораторной работы.
- Цель лабораторной работы.
- Решения задач:
  - Текст задания.
  - Исходный код.
  - Решение задач со звёздочкой требуется для получения максимального балла.

Требования к лабораторной работе:
- Каждая функция в качестве комментария, должна содержать спецификацию.
- Указание спецификаций отдельно от кода не требуется.

Решение задач:

Задача №1. Опишем структуру `point`:

Пункт (a). Объявите структуру `point` с инициализацией.

Код программы:

```c
#include <stdio.h>

// Структура Point хранит в себе координаты точки (x; y)
struct point {
  double x; // Координаты по ox
  double y; // Координаты по oy
};

typedef struct point point;

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

  return 0;
}
```

Пункт (b). Реализуйте функцию ввода структуры `point`. Заголовок: `void inputPoint(point *p)`.

Код программы:

```c
// Предназначен для ввода координат точки (x; y)
void inputPoint(point *p) {
  scanf("%lf %lf", &p->x, &p->y);
}
```

Пункт (c). Реализуйте функцию вывода структуры `point`. Выводите данные в следующем формате (1.450; 1.850) с тремя знаками после запятой. Заголовок: `void outputPoint(point p)`.

Код программы:

```c
// Выводит координату точки p в формате (x; y)
void outputPoint(point p) {
  printf("(%.3lf; %.3lf)", p.x, p.y);
}
```

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

Код программы:

```c
int main() {
  point p1 = {1.45, 1.86};
  point p2 = p1;

  outputPoint(p2);

  return 0;
}
```

Пункт (e). Создайте массив структур размера `N=3`. Реализуйте функции для его ввода `inputPoints` и вывода `outputPoint`.
Заголовок функции ввода: `void inputPoints(point *p, int n)`.
Заголовок функции вывода: `void outputPoints(point *p, int n)`.

Код программы:

```c
// Предназначен для ввода массива p размера size с координатами точек
void inputPoints(point *p, int size) {
  for (int i = 0; i < size; i++) {
    inputPoint(&p[i]);
  }
}

// Выводит массив p размера size содержащий координаты точек
void outputPoints(point *p, int size) {
  for (int i = 0; i < size; i++) {
    outputPoint(p[i]);
    printf(" ");
  }
  printf("\n");
}
```

Пункт (f). Реализуйте функцию, которая принимает на вход две структуры типа point и возвращает точку, находящуюся посередине между точками `p1` и `p2`.
Заголовок: `point getMiddlePoint(point p1, point p2)`.

Код программы:

```c
// Возвращает координату точки, находящейся между точкой p1 и p2
point getMiddlePoint(point p1, point p2) {
  point pm = {(p1.x + p2.x) / 2, (p1.y + p2.y) / 2};
  return pm;
}
```

Пункт (g). Реализуйте функцию `isEqualPoint`, которая возвращает значение ’истина’, если точки совпадают (с погрешностью не более `DBL_EPSILON`, определённой в `<float.h>`)
Заголовок: `int isEqualPoint(point p1, point p2)`.

Код программы:

```c
// Возвращает 1 ("Истина"), если точка p1 равна p2, 0 - ином случае
int isEqualPoint(point p1, point p2) {
  if (dabs(p1.x - p2.x) < DBL_EPSILON && dabs(p1.y - p2.y) < DBL_EPSILON)
    return 1;
  return 0;
}
```

Пункт (h). Реализуйте функцию, которая возвращает значение ’истина’, если точка `p3` лежит ровно посередине между точками `p1` и `p2`.
Заголовок: `int isPointBetween(point p1, point p2, point p3)`.

Код программы:

```c
// Возвращает 1 ("Истина"), если точка p3, находится между p1 и p2,
// 0 - ином случае
int isPointBetween(point p1, point p2, point p3) {
  point pm = getMiddlePoint(p1, p2);
  if (isEqualPoint(pm, p3))
    return 1;
  return 0;
}
```

Пункт (i). Реализуйте функцию `swapCoordinates` которая меняет значения координат `x` и `y` структуры типа `point`.
Заголовок: `void swapCoordinates(point *p)`.

Код программы:

```c
// Обменивает значение x и y в структуре point
void swapCoordinates(point *p) {
  double t = p->x;
  p->x = p->y;
  p->y = t;
}
```

Пункт (j). Реализуйте функцию `swapPoints` которая обменивает две точки.
Заголовок: `void swapPoints(point *p1, point *p2)`.

Код программы:

```c
// Обменивает координаты p1 и p2
void swapPoints(point *p1, point *p2) {
  point t = *p1;
  *p1 = *p2;
  *p2 = t;
}
```

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

Код программы:

```c
#include <malloc.h>

#define N 3

int main() {
  point *p = (point*)malloc(sizeof(point)*N);

  free(p);

  return 0;
}
```

Пункт (l). Реализуйте функцию, которая находит расстояние между двумя точками.
Заголовок: `double getDistance(point p1, point p2)`.

Код программы:

```c
// Возвращает расстояние между точками p1 и p2
double getDistance(point p1, point p2) {
  return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}
```

Пункт (m). Реализуйте функцию, которая сортирует массив точек размера `N = 3` по увеличению координаты `x`, а при их равенстве – по координате `y`.

Код программы:

```c
#include <stdio.h>

#define N 3

int cmp_distance(const void *a, const void *b) {
  point *p1 = (point*)a;
  point *p2 = (point*)b;
  if (isEqualPoint(*p1, *p2))
    return 0;
  else if (p1->x < p2->x || p1->x == p2->x && p1->y < p2->y)
    return -1;
  return 1;
}

int main() {
  point p[N] = {
    {1, 3},
    {2, 2},
    {2, 1}
  };

  qsort(p, N, sizeof(point), cmp_distance);

  outputPoints(p, 4);

  return 0;
}
```

увеличению расстояния до начала координат.

Код программы:

```c
int cmp_distance(const void *a, const void *b) {
  point zeroPoint = {0, 0};
  point *p1 = (point*)a;
  point *p2 = (point*)b;
  double d1 = getDistance(*p1, zeroPoint);
  double d2 = getDistance(*p2, zeroPoint);
  if (dabs(d1 - d2) < DBL_EPSILON)
    return 0;
  else if (d1 < d2)
    return -1;
  return 1;
}

int main() {
  point p[N] = {
    {0, 0},
    {2, 2},
    {1, 1}
  };

  qsort(p, N, sizeof(point), cmp_distance);

  outputPoints(p, 5);

  return 0;
}
```

Задача 2. Опишем структуру `line`, которая задаёт линию на плоскости уравнением

$$ax + by + c = 0$$

Пункт (a). Реализуйте функцию `inputLine` ввода структуры `line`.
Заголовок: `void inputLine(line *line)`.

Код программы:

```
// Осуществляет ввод структуры line, (коэффициенты: a, b, c)
void inputLine(line *line) {
  scanf("%lf %lf %lf", &line->a, &line->b, &line->c);
}
```

Пункт (b). Инициализируйте структуру типа `line` при объявлении.

Код программы:

```
int main() {
  line line = {1, 2, 3};

  return 0;
}
```

Пункт (c). Реализуйте функцию `getLine` которая возвращает прямую по координатам точек.
Заголовок: `line getLineByPoints(point p1, point p2).`

Код программы:

```
// Возвращает коэффициенты прямой a, b и c в структуре line, построенной по
// точкам p1 И p2
line getLineByPoints(point p1, point p2) {
  line line;
  line.a = p1.y - p2.y;
  line.b = p2.x - p1.x;
  line.c = p1.x * p2.y - p2.x * p1.y;
  return line;
}
```

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

Код программы:

```
// Возвращает коэффициенты прямой a, b и c в структуре line, построенной по
// координатам (x1, y1) и (x2, y2)
line getLine(double x1, double y1, double x2, double y2) {
  line line;
  line.a = y1 - y2;
  line.b = x2 - x1;
  line.c = x1 * y2 - x2 * y1;
  return line;
}
```

Пункт (e). Реализуйте функцию `outputLineEquation` вывода уравнения прямой `line`.
Заголовок: `void outputLineEquation(line line)`.

Код программы:

```
// Выводит структуру line в формате: ax + bx + c = 0
void outputLineEquation(line line) {
  printf("%+.2lfx%+.2lfy%+.2lf = 0", line.a, line.b, line.c);
}
```

Пункт (f). Реализуйте функцию `isParallel`, которая возвращает значение ’истина’ если прямые `line1` и `line2` параллельны, ’ложь’ – в противном случае.
Заголовок: `int isParallel(line l1, line l2)`.

Код программы:

```
// Возвращает 1 ("истина"), если прямая l1 параллельна l2,
// 0 ("Ложь") - в ином случае
int isParallel(line l1, line l2) {
  if (dabs(l1.a / l2.a - l1.b / l2.b) < DBL_EPSILON)
    return 1;
  return 0;
}
```

Пункт (g). Реализуйте функцию `isPerpendicular`, которая возвращает значение ’истина’ если прямые `line1` и `line2` перпендикулярны, ’ложь’ – в противном случае.
Заголовок: `int isPerpendicular(line l1, line l2)`.

Код программы:

```
// Возвращает 1 ("истина"), если прямая l1 перпендикулярна l2,
// 0 ("Ложь") - в ином случае
int isPerpendicular(line l1, line l2) {
  if (dabs(l1.a * l2.a + l1.b * l2.b) < DBL_EPSILON)
    return 1;
  return 0;
}
```

Пункт (h). Определите, есть ли среди данных `n` прямых на плоскости (`n` – `const`) параллельные.
Заголовок: `int hasParallelLines(line *lines, size_t n)`.

Код программы:

```
// Возвращает 1 ('Истина'), если в массиве lines размера n структуры line есть
// две параллельные прямые
int hasParallelLines(line *lines, size_t n) {
  for (size_t i = 0; i < n - 1; i++)
    if (isParallel(lines[i], lines[i + 1]))
      return 1;
  return 0;
}
```

Пункт (i). Реализуйте функцию `printIntersectionPoint`, которая выводит точку пересечения прямых `line1` и `line2`. Если точек пересечения нет – проинформируйте пользователя.
Заголовок: `void printIntersectionPoint(line l1, line l2)`.

Код программы:

```
// Выводит координаты точки пересечения двух прямых l1 и l2
void printIntersectionPoint(line l1, line l2) {
  point p;
  p.x = (l1.b * l2.c - l2.b * l1.c) / (l1.a * l2.b - l2.a * l1.b);
  p.y = (l2.a * l1.c - l1.a * l2.c) / (l1.a * l2.b - l2.a * l1.b);
  outputPoint(p);
}
```

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

Пункт(a). Объявите с инициализацией структуру типа `circle`.

Код программы:

```
int main() {
  circle circle = {{0, 0}, 4};

  return 0;
}
```

Пункт(b). Объявите с инициализацией массив из двух структур типа `circle`.

Код программы:

```
int main() {
  circle circle[2] = {
    {{0, 0}, 4},
    {{2, 5}, 2}
  };

  return 0;
}
```

Пункт(b). Объявите с инициализацией массив из двух структур типа `circle`.

Код программы:

```
int main() {
  circle circle[2] = {
    {{0, 0}, 4},
    {{2, 5}, 2}
  };

  return 0;
}
```

Пункт(c). Реализуйте функцию `inputCircle` ввода структуры `circle`.
Заголовок: `void inputCircle(circle *a)`.

Код программы:

```
// Ввод структуры circle (координаты центра окружности и радиуса)
void inputCircle(circle *a) {
  scanf("%lf %lf %lf", &a->center.x, &a->center.y, &a->r);
}
```

Пункт(d). Реализуйте функцию `inputCircles` ввода массива структур `circle`.
Заголовок: `void inputCircles(circle *a, size_t n)`.

Код программы:

```
// Ввод массива a размера n структуры circle
void inputCircles(circle *a, size_t n) {
  for (size_t i = 0; i < n; ++i)
    inputCircle(&a[i]);
}
```

Пункт(e). Реализуйте функцию `outputCircle` вывода структуры `circle`.
Заголовок: `void outputCircle(circle a)`.

Код программы:

```
// Вывод структуры circle (координаты центра окружности и радиуса)
void outputCircle(circle a) {
  printf("C");
  outputPoint(a.center);
  printf(", r = %.3lf", a.r);
}
```

Пункт(f). Реализуйте функцию `outputCircles` вывода массива структур `circle`. Заголовок: `void outputCircles(circle *a, size_t n)`.

Код программы:

```
// Вывод массива a размера n структуры circle
void outputCircles(circle *a, size_t n) {
  for (size_t i = 0; i < n; i++) {
    outputCircle(a[i]);
    printf("\n");
  }
}
```

Пункт(g). Реализуйте функцию `hasOneOuterIntersection`, которая возвращает значение ’истина’, если окружность `c1` касается внешним образом окружности `c2`. Заголовок: `int hasOneOuterIntersection(circle c1, circle c2)`.

Код программы:

```
// Возвращает 1 ('Истина'), если окружность c1 касается внешней стороны
// окружности c2, 0 ('Ложь')- в ином случае
int hasOneOuterIntersection(circle c1, circle c2) {
  if (dabs(c1.r + c2.r - getDistance(c1.center, c2.center)) < DBL_EPSILON)
    return 1;
  return 0;
}
```

Пункт(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.

Код программы:

```
// Возвращает количество окружностей внутри i-ой окружности массива c размера n
int countInnerCircles(circle *c, size_t n, size_t i) {
  int count = 0;
  for (size_t j = i + 1; j < n; j++)
    if (getDistance(c[i].center, c[j].center) + c[j].r <= c[i].r)
      count += 1;
  return count;
}

// Выводит значения окружности массива c размера n, внутри которой
// располагается набольшее число окружностей
void printCircleWithMostInnerCircles(circle *c, size_t n) {
  int maxInnerCircle = 0;
  size_t maxIndex = 0;
  for (size_t i = 0; i < n; i++) {
    int countI = countInnerCircles(c, n, i);
    if (countI > maxInnerCircle) {
      maxInnerCircle = countI;
      maxIndex = i;
    } else if (countI == maxInnerCircle && c[i].r < c[maxIndex].r)
      maxIndex = i;
  }
  outputCircle(c[maxIndex]);
}

int main() {
  size_t n;
  scanf("%zd", &n);
  circle circle[n];
  inputCircles(circle, n);

  printCircleWithMostInnerCircles(circle, n);

  return 0;
}
```

**Пункт(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.

Код программы:

```
// Обменивает значения a и b структуры circle
void swapCircles(circle *a, circle *b) {
  circle t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс окружности массива с размера n, внутри которой
// располагается набольшее число окружностей
size_t indexCircleWithMostInnerCircles(circle *c, size_t n) {
  int maxInnerCircle = 0;
  size_t maxIndex = 0;
  for (size_t i = 0; i < n; i++) {
    int countI = countInnerCircles(c, n, i);
    if (countI > maxInnerCircle) {
      maxInnerCircle = countI;
      maxIndex = i;
    } else if (countI == maxInnerCircle && c[i].r < c[maxIndex].r)
      maxIndex = i;
  }
  return maxIndex;
}

// Сортирует массив по неубыванию количества лежащих в ней окружностей, в
// случае, при равенстве - сортирует по неубыванию радиуса
void sortCircleWithMostInnerCircles(circle *c, size_t n) {
  for (size_t i = 0; i < n - 1; i++) {
    swapCircles(&c[i], &c[indexCircleWithMostInnerCircles(c, n)]);
  }
}
```

Задание 4. Опишем структуру `fraction`:

Пункт (a). Реализуйте функцию `inputFraction` ввода структуры `fraction`. Пример ввода, который должен обрабатываться программой: ’5/7’, ’2 / 17’.
Заголовок: `void inputFraction(fraction *f)`.

Код программы:

```c
// Осуществляет ввод структуры fraction
void inputFraction(fraction *f) {
  scanf("%d %d", &f->numerator, &f->denumerator);
}
```

Пункт (b). Реализуйте функцию `inputFractions` ввода массива структур `fraction`.
Заголовок: `void inputFractions(fraction *f, size_t n)`.

Код программы:

```c
// Осуществляет ввод массива структуры fraction
void inputFractions(fraction *f, size_t n) {
  for (size_t i = 0; i < n; i++)
    inputFraction(&f[i]);
}
```

Пункт (c). Реализуйте функцию `outputFraction` вывода структуры `fraction` в формате ’5/7’.
Заголовок: `void outputFraction(fraction f)`.

Код программы:

```c
// Выводит структуру fraction
void outputFraction(fraction f) {
  printf("%d/%d", f.numerator, f.denumerator);
}
```

Пункт (d). Реализуйте функцию `outputFractions` вывода массива структур `fraction`.
Заголовок: `void outputFractions(fraction *f, size_t n)`.

Код программы:

```c
// Выводит массив структуры fraction
void outputFractions(fraction *f, size_t n) {
  for (size_t i = 0; i < n; i++) {
    outputFraction(f[i]);
    if (i < n - 1)
      printf(", ");
  }
  printf("\n");
}
```

Пункт (e). Реализуйте функцию `gcd` возвращающую наибольший общий делитель.
Заголовок: `int gcd(int a, int b)`.

Код программы:

```c
// Возвращает наибольший общий делитель чисел a и b
int gcd(int a, int b) {
  sort2(&a, &b);
  if (b % a == 0)
    return a;
  int gcd = 1;
  for (int i = 2; i <= (int)sqrt(b); i ++)
    if (a % i == 0 && b % i == 0)
      gcd = i;
  return gcd;
}
```

Пункт (f). Реализуйте функцию `lcm` возвращающую наименьшее общее кратное.
Заголовок: `int lcm(int a, int b)`.

Код программы:

```
// Возвращает наименьшее общее кратное чисел a и b
int lcm(int a, int b) {
  sort2(&a, &b);
  int d = gcd(a, b);
  if (d == 1)
    return a * b;
  else if (d == a)
    return b;
  return b * a / d;
}
```

Пункт (g). Реализуйте функцию `simpleFraction` для сокращения дроби `a`.
Заголовок: `void simpleFraction(fraction *f)`.

Код программы:

```
// Возвращает упрощенную дробь f структуры fraction
void simpleFraction(fraction *f) {
  assert(f->denumerator != 0);
  int d = gcd(f->numerator, f->denumerator);
  f->numerator /= d;
  f->denumerator /= d;
}
```

Пункт (h). Реализуйте функцию `mulFractions` умножения двух дробей `a` и `b`.
Заголовок: `fraction mulFractions(fraction f1, fraction f2)`.

Код программы:

```
// Возвращает дробь, которая является результатом произведения дроби f1 и f2
fraction mulFractions(fraction f1, fraction f2) {
  assert(f1.denumerator != 0 && f2.denumerator != 0);
  fraction f3;
  fraction n1 = (fraction){f1.numerator, f2.denumerator};
  fraction n2 = (fraction){f2.numerator, f1.denumerator};
  simpleFraction(&n1);
  simpleFraction(&n2);
  f3.numerator = n1.numerator * n2.numerator;
  f3.denumerator = n1.denumerator * n2.denumerator;
  return f3;
}
```

Пункт (i). Реализуйте функцию `divFractions` деления двух дробей `a` и `b`.
Заголовок: `fraction divFraction(fraction f1, fraction f2)`.

Код программы:

```
// Возвращает дробь, которая является результатом делением дроби f1 на f2
fraction divFraction(fraction f1, fraction f2) {
  assert(f1.denumerator != 0 && f2.denumerator != 0);
  return mulFractions(f1, (fraction){f2.denumerator, f2.numerator});
}
```

Пункт (j). Реализуйте функцию `addFractions` сложения двух дробей `a` и `b`.
Заголовок: `fraction addFraction(fraction f1, fraction f2)`.

Код программы:

```
// Возвращает дробь, которая является суммой дроби f1 и f2
fraction addFraction(fraction f1, fraction f2) {
  assert(f1.denumerator != 0 && f2.denumerator != 0);
  fraction f3;
  int d = lcm(f1.denumerator, f2.denumerator);
  f3.numerator = f1.numerator * (d / f1.denumerator) +
                 f2.numerator * (d / f2.denumerator);
  f3.denumerator = d;
  simpleFraction(&f3);
  return f3;
}
```

Пункт (k). Реализуйте функцию `subFractions` вычитания двух дробей `a` и `b`.
Заголовок: `fraction subFraction(fraction f1, fraction f2)`.

Код программы:

```
// Возвращает дробь, которая является разницей f1 и f2
fraction subFraction(fraction f1, fraction f2) {
  return addFraction(f1, (fraction){-f2.numerator, f2.denumerator});
}
```

Пункт (l). Реализуйте функцию для поиска суммы `n` дробей.
Заголовок: `fraction sumFractions(fraction *f, size_t n)`.

Код программы:

```
// Возвращает сумму дробей массива f размера n структуры fraction
fraction sumFractions(fraction *f, size_t n) {
  fraction f3 = f[0];
  for (size_t i = 1; i < n; i++) {
    assert(f->denumerator != 0);
    f3 = addFraction(f3, f[i]);
  }
  return f3;
}
```

Задание 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. Указание: используйте структуру

Реализация:

```
// Ввод данных о студенте s
void inputStudent(student *s) {
  scanf("%s %d %d %d %d %d", s->surname, &s->marks[0], &s->marks[1],
        &s->marks[2], &s->marks[3], &s->marks[4]);
}

// Ввод массива s размера n данных о студентах
void inputStudents(student *s, size_t n) {
  for (int i = 0; i < n; i++)
    inputStudent(&s[i]);
}

// Вывод имени студента
void outputStudent(student s) {
  printf("%s", s.surname);
}

// Выводит 1 ('Истина'), если оценка меньше или равно двум, иначе 0
int isBad(int a) {
  return a <= 2 ? 1 : 0;
}

// Выводи 1 ('Истина'), если количество "Плохих" оценок у стедента < 2, иначе 0
int isGoodStudent(student s) {
  int countBad = 0;
  for (size_t i = 0; i < 5; ++i)
    if (isBad(s.marks[i]))
      countBad++;
  if (countBad < 2)
    return 1;
  return 0;
}

// Выводит имена студентов прошедших аттестацию
void outputGoodStudent(student *s, size_t n) {
  for (int i = 0; i < n; ++i)
    if (isGoodStudent(s[i]))
      outputStudent(s[i]);
}

int main() {
  size_t n;
  scanf("%zd", &n);
  student s[n];
  inputStudents(s, n);

  outputGoodStudent(s, n);

  return 0;
}
```

Задание 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. Сравнение времени `t1` с `t2` оформить подпрограммой.

Реализация:

```c
// Структура time хранит в себе:
// hh - Часы
// mm - Минуты
// ss - Секунды
struct time {
  int hh;
  int mm;
  int ss;
};

typedef struct time time;


// Вводит данные в структуру time
void inputTime(time *t) {
  scanf("%d %d %d", &t->hh, &t->mm, &t->ss);
  assert(t->hh <= 24 && t->mm <= 60 && t->ss <= 60);
}

// Вводит массив t размера n структуры time
void inputTimes(time *t, size_t n) {
  for (size_t i = 0; i < n; i++)
    inputTime(&t[i]);
}

// Выводит структуру time в формате hh:mm:ss
void outputTime(time t) {
  printf("%d:%d:%d\n", t.hh, t.mm, t.ss);
}

// Выводит массив t размера n структуры time
void outputTimes(time *t, size_t n) {
  for (size_t i = 0; i < n; i++)
    outputTime(t[i]);
}

#define COUNT_SS_IN_HH 3600
#define COUNT_SS_IN_MM 60

// Возвращает 1, если время t1 < t2, приведенное в секунды
int isBefore(time const t1, time const t2) {
  int ss1 = t1.hh * COUNT_SS_IN_HH + t1.mm * COUNT_SS_IN_MM + t1.ss;
  int ss2 = t2.hh * COUNT_SS_IN_HH + t2.mm * COUNT_SS_IN_MM + t2.ss;
  if (ss1 < ss2)
    return 1;
  return 0;
}

int cmp_sortTime(const void *a, const void *b) {
  time *t1 = (time *) a;
  time *t2 = (time *) b;
  if (t1->hh == t2->hh && t1->mm == t2->mm && t1->ss == t2->ss)
    return 0;
  else if (isBefore(*t1, *t2))
    return -1;
  return 1;
}
```

```
int main() {
  size_t n;
  scanf("%zd", &n);
  time t[n];
  inputTimes(t, n);

  qsort(t, n, sizeof(time), cmp_sortTime);

  outputTimes(t, n);

  return 0;
}
```

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

Реализация:

```
// Возвращает время прошедшее в промежутке от t1 до t2
time timePassed(time const t1, time const t2) {
  int ss1 = t1.hh * COUNT_SS_IN_HH + t1.mm * COUNT_SS_IN_MM + t1.ss;
  int ss2 = t2.hh * COUNT_SS_IN_HH + t2.mm * COUNT_SS_IN_MM + t2.ss;
  int ss3 = ss2 - ss1;
  time t3;
  t3.hh = ss3 / COUNT_SS_IN_HH;
  ss3 -= t3.hh * COUNT_SS_IN_HH;
  t3.mm = ss3 / COUNT_SS_IN_MM;
  ss3 -= t3.mm * COUNT_SS_IN_MM;
  t3.ss = ss3 % COUNT_SS_IN_MM;
  return t3;
}
```