Семинар #4: Часть 2: Структуры. Домашнее задание.

Задача 1. Треугольник

Для описания точек и треугольников на плоскости были определены структуры `Point` и `Triangle`:

```
#include <stdio.h>

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

struct triangle
{
    Point a;
    Point b;
    Point c;
};
typedef struct triangle Triangle;

// Тут нужно написать все необходимые функции

int main()
{
    Triangle t = {{1.00, 0.00}, {0.50, 2.00}, {0.00, 1.50}};
    printf("Perimeter = %.2f\n", get_triangle_perimeter(&t));

    Point d = {1.0, 1.0};
    print_triangle(&t);
    move_triangle(&t, d);
    print_triangle(&t);
}
```

Напишите следующие функции для работы с этими структурами:

- Функцию `print_point`, которая будет принимать точку и печатать её в формате `(1.23, 4.56)`. То есть в круглых скобках, через запятую и с двумя знаками после запятой.
- Функцию `print_triangle`, которая будет принимать на вход треугольник и печатать координаты треугольника в следующем формате: `{(1.00, 0.00), (0.50, 2.00), (0.00, 1.50)}`.
- Функцию `distance`, которая будет принимать на вход 2 точки и возвращать расстояние между ними.
- Функцию `get_triangle_perimeter`, которая будет принимать треугольник по константному указателю и возвращать его периметр.
- Функцию `moved_triangle`, которая будет принимать на вход треугольник по константному указателю и одну точку (она будет играть роль вектора-перемещения). Функция должна возвращать новый треугольник, у которого все координаты будут передвинуты на вектор-перемещение.
- Функцию `move_triangle`, которая будет принимать на вход треугольник по указателю и одну точку (она будет играть роль вектора-перемещения). Функция должна менять передаваемый ей треугольник.

Задача 2. Одна строка

Решения всех подзадач этой задачи – одна строка. Вам нужно создать файл в формате `.txt` и, используя любой текстовый редактор, записать в него ответы на все подзадачи. После этого, файл нужно поместить в ваш репозиторий на github.

1. В следующей программе создаётся структура `Book`:

```
struct book
{
    char title[50];
    int pages;
    float price;
};
typedef struct book Book;

int main()
{
    Book b = {"Fahrenheit 451", 400, 700.0};

}
```

   (a) Создайте указатель `pb` и сделайте так, чтобы он указывал на структуру `b`.

   (b) Создайте указатель `pprice` и сделайте так, чтобы он указывал на поле `price` структуры `b`.

   (c) Создайте указатель `pc` и сделайте так, чтобы он указывал символ `'t'` поля `title` структуры `b`.

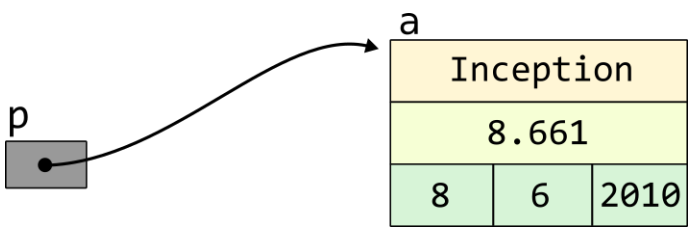
2. В следующей программе была создана структура `a` типа `Movie` и указатель на неё.

```
#include <stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
typedef struct date Date;

struct movie
{
    char title[50];
    float rating;
    Date release_date;
};
typedef struct movie Movie;

int main()
{
    Movie a = {"Inception", 8.7, {8, 6, 2010}};
    Movie* p = &a;

}
```



   (a) Увеличьте на 1 значение поля `rating`, используя только указатель `p`.

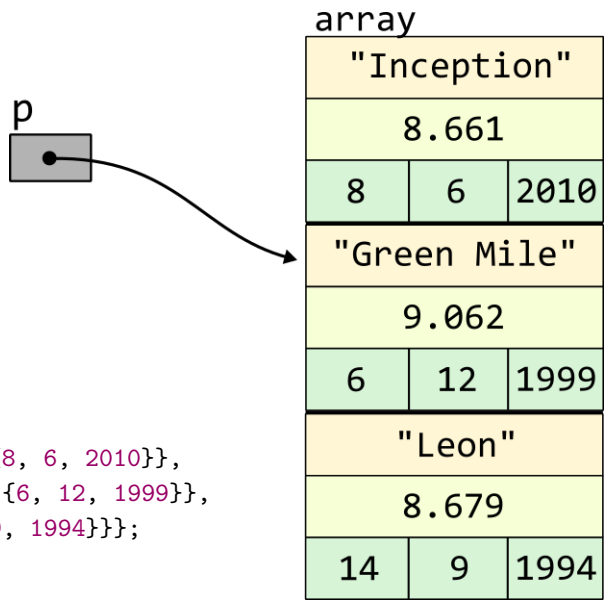
   (b) Увеличьте на 1 значение поля месяца выхода фильма, используя только указатель `p`.

3. В следующей программе был создан массив `array` из структур типа `Movie` и указатель `p`, который указывает на второй элемент массива (`array[1]`).

```c
#include <stdio.h>
struct date
{
    int day;
    int month;
    int year;
};
typedef struct date Date;

struct movie
{
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

int main()
{
    Movie array[3] = {{"Inception", 8.661, {8, 6, 2010}},
                      {"Green Mile", 9.062, {6, 12, 1999}},
                      {"Leon", 8.679, {14, 9, 1994}}};
    Movie* p = &array[1];
}
```



(a) Увеличьте на 1 значение рейтинга фильма `Inception`, используя только указатель `p`.

(b) Удвойте значение года выхода фильма `Leon`, используя только указатель `p`.

Задача 3. Рецензии на компьютерные игры

На вход программе приходит информация о рецензиях компьютерных игр. В первой строке содержится число `n` - количество игр. Далее идут `n` строк. В каждой строке содержится название игры, заканчивающееся двоеточием сразу после идёт целое число `k` – количество оценок, которые эта игра получила, затем идут `k` оценок. Оценка, это число от 1 до 10. Нужно отсортировать все игры по средней оценке и напечатать название игр и их среднюю оценку.

вход	выход
5 Need For Speed: 6 6 1 2 7 5 4 Sector: 3 1 4 2 The Cube: 7 9 8 7 9 8 10 7 Principle Of Chaos 2: 5 4 3 6 5 7 Metal Power: 10 8 5 3 9 6 2 6 7 5 8	The Cube, 8.286 Metal Power, 5.900 Principle Of Chaos 2, 5.200 Need For Speed, 4.667 Sector, 2.333

Для считывания до символа `:` можно использовать `scanf` следующим образом:

```c
char title[50];
char temp;
scanf("%[^:]", title);  // Считываем название до символа :
scanf("%c", &temp);     // Считываем символ :
...                     // Считываем числа
scanf("%c", &temp);     // Считываем перенос на новую строку ( символ \n )
scanf("%[^:]", title);  // Считываем название до символа :
scanf("%c", &temp);     // Считываем символ :
...
```

Протестировать программу можно на файле `videogames.txt`.

Задача 4. Цикл

```
struct node
{
    int value;
    struct node* ptr;
};
typedef struct node Node;
```

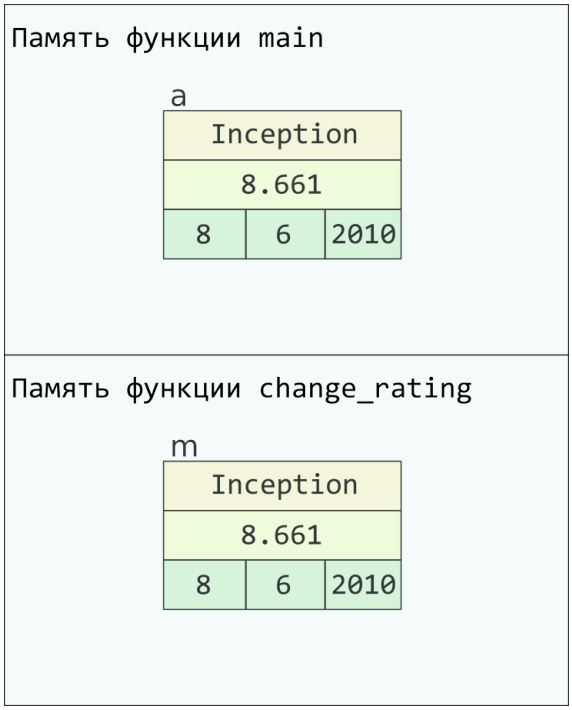
Необязательные задачи (не входят в ДЗ, никак не учитываются)

Задача 1. Передача в функцию по указателю

Передача в функцию по значению

```c
#include <stdio.h>
struct movie
{
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void change_rating(Movie m)
{
    m.rating += 1;
}
int main()
{
    Movie a = {"Inception", 8.661,
                {8, 6, 2010}};
    change_rating(a);
}
```

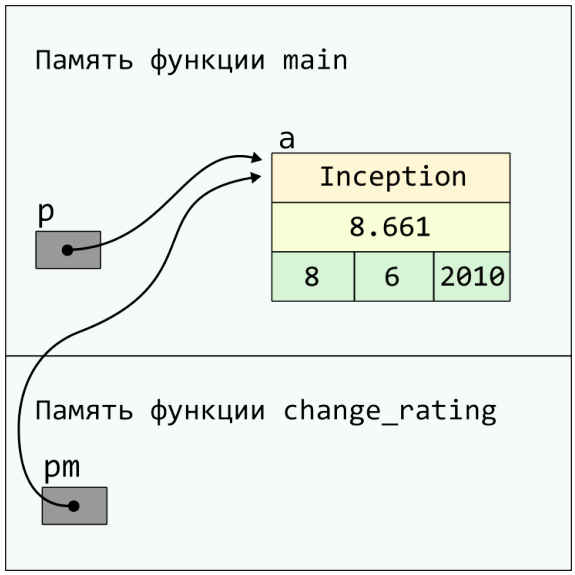


Всё, что передаётся в функцию, копируется (кроме массивов). Поэтому функция `change_rating` будет менять поле `rating` у копии структуры `a`, а изначальная структура не изменится.

Передача в функцию по указателю:

```c
#include <stdio.h>
struct movie
{
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void change_rating(Movie* pm)
{
    pm->rating += 1;
}
int main()
{
    Movie a = {"Inception", 8.661,
                {8, 6, 2010}};
    Movie* p = &a;
    change_rating(&a);
}
```



Теперь в функцию копируется указатель, который содержит адрес стурктуры `a`. Используя этот указатель, мы можем изменить изначальную структуру. Более того, так как указатель занимает меньше памяти, его копирования в функцию происходит быстрее, чем копирование всей структуры.

Подзадачи:

1. Напишите функцию `void increase_rating(Movie* p)`, которая будет принимать указатель типа `Movie*` и увеличивать рейтинг фильма, на которой указывает `p`, на 1.
2. Напишите функцию `void change_year_of_movies(Movie* p, int size)`, которая принимает на вход указатель на первый элемент массива структур типа `Movie` и размер этого массива. Функция должна увеличивать год выхода всех фильмов на `1`. Протестируйте функции, вызвав их из функции `main` с помощью следующего кода:

```
#include <stdio.h>
struct date
{
    int day, month, year;
};
typedef struct date Date;
struct movie
{
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void print_date(const Date* pd)
{
    printf("%02d.%02d.%04d", pd->day, pd->month, pd->year);
}
void print_movie(const Movie* pm)
{
    printf("Title: %s\nRating: %.2f\nDate: ", pm->title, pm->rating);
    print_date(&pm->release_date);
    printf("\n");
}
// Тут вам нужно написать функции increase_rating и change_year_of_movies

int main()
{
    Movie a[3] = {{"Inception", 8.661, {8, 6, 2010}},
                  {"Green Mile", 9.062, {6, 12, 1999}},
                  {"Leon", 8.679, {14, 9, 1994}}};
    increase_rating()
    change_year_of_movies(a, 3);
    for (int i = 0; i < 3; ++i)
        print_movie(&a[i]);
}
```