

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

### Задача 1. Треугольник

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

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

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

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 `get_triangle_area`, которая будет принимать на вход треугольник по константному указателю и возвращать его площадь. Можно использовать формулу Герона.
- 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. 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	The Cube, 8.286
Need For Speed: 6 6 1 2 7 5 4	Metal Power, 5.900
Sector: 3 1 4 2	Principle Of Chaos 2, 5.200
The Cube: 7 9 8 7 9 8 10 7	Need For Speed, 4.667
Principle Of Chaos 2: 5 4 3 6 5 7	Sector, 2.333
Metal Power: 10 8 5 3 9 6 2 6 7 5 8	

Протестировать программу можно на файле `videogames.txt`.

### Задача 3. Создание указателей

Решения всех подзадач этой части – одна строка. Результат выполнения задания – .txt файл, который содержит все эти строки.

1. В следующей программе создаётся структура Book из семинара про структуры:

```
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 создаётся переменная a типа float и p указатель, который хранит её адрес:

```
int main()
{
    float a = 1.2;
    float* p = &a;
    // Тут нужно написать 1 строку кода
}
```

Создайте указатель pp и сделайте так, чтобы он указывал на указатель p.

3. В следующей программе создаётся структура Book из семинара про структуры и указатель на неё:

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

int main()
{
    Book b = {"Fahrenheit 451", 400, 700.0};
    Book* pb = &b;
    // Тут нужно написать 1 строку кода
}
```

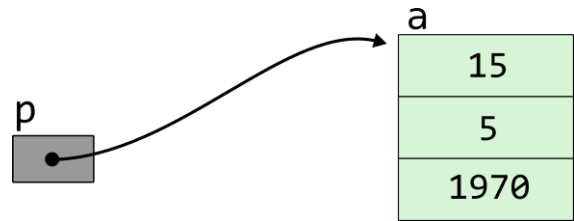
Создайте указатель ppb и сделайте так, чтобы он указывал на указатель pb.

## Задача 4. Использование указателей

Решения всех подзадач этой части – одна строка. Результат выполнения задания – .txt файл, который содержит все эти строки.

1. В следующей программе была создана структура `a` типа `Date` и указатель на эту структуру. Добавьте 1 к значению поля `year`, используя только указатель `p`.

```
#include <stdio.h>
struct date
{
    int day, month, year;
};
typedef struct date Date;
int main()
{
    Date a = {15, 5, 1970};
    Date* p = &a;
    // Тут нужно написать 1 строку кода
    printf("%d %d %d\n",
           a.day, a.month, a.year);
}
```

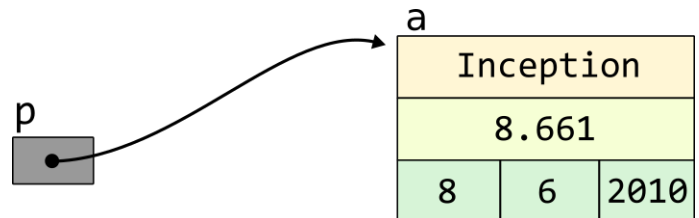


2. В следующей программе была создана структура `a` типа `Movie` и указатель на неё.

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

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

int main()
{
    Movie a = {"Inception", 8.661, {8, 6, 2010}};
    Movie* p = &a;
    // Тут нужно написать 1 строку кода
}
```



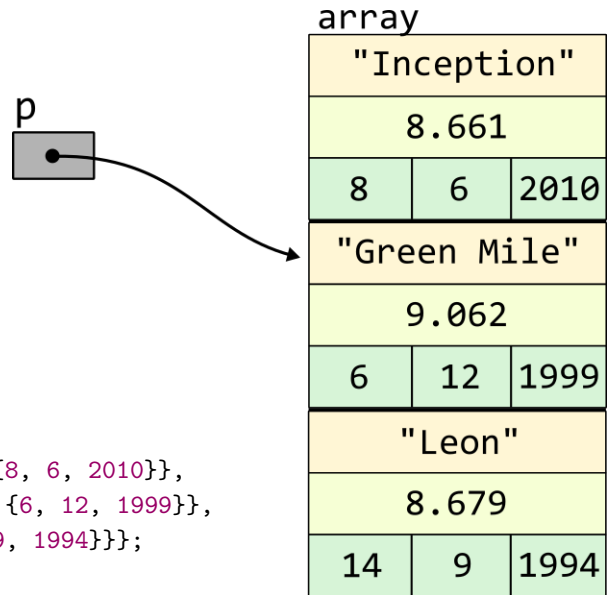
- (a) Увеличьте на 1 значение поля `rating`, используя только указатель `p`.
- (b) Увеличьте на 1 значение поля месяца выхода фильма, используя только указатель `p`.

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

```
#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;

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`. При этом менять `p` нельзя, он должен указывать на `array[1]`.
- (b) Удвойте значение года выхода фильма `Leon`, используя только указатель `p`. При этом менять `p` нельзя, он должен указывать на `array[1]`.

## Задача 5. Передача в функцию по указателю

### Передача в функцию по значению

```
#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);
}
```

Память функции main

a		
Inception		
8.661		
8	6	2010

Память функции change\_rating

m		
Inception		
8.661		
8	6	2010

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.

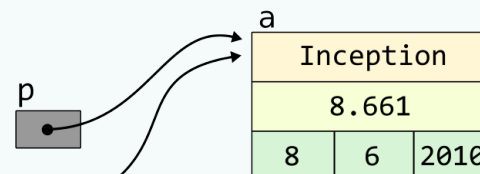
### Передача в функцию по указателю:

```
#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);
}
```

Память функции main



Память функции change\_rating



Теперь в функцию копируется указатель, который содержит адрес структуры `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]);
}
```