# Семинар #5: Структуры. Классные задачи.

## Структуры. Описание, объявление и инициализация:

Пример программы, в которой описывается структура для удобной работы с объектами Книга (`Book`).

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

// Создадим новый составной тип под названием struct book
struct book
{
    char title[50];
    int pages;
    float price;
}; // <------------------------ НЕ ЗАБУДЬТЕ ТУТ ТОЧКУ С ЗАПЯТОЙ

void print_book_info(struct book b)
{
    printf("Book info:\n");
    printf("Title: %s\nPages: %d\nPrice: %g\n\n", b.title, b.pages, b.price);
}

int main()
{
    struct book a = {"The Martian", 10, 550.0};
    print_book_info(a);
    a.pages = 369;
    strcpy(a.title, "The Catcher in the Rye");
    print_book_info(a);

    struct book scifi_books[10] = {{"Dune", 300, 500.0}, {"Fahrenheit 451", 400, 700.0},
                                                         {"Day of the Triffids", 304, 450.0}};
    scifi_books[2].price = 2000.0;
    print_book_info(scifi_books[2]);
}
```

### Задачи:

1. **Структура Дата:** Описать структуру `struct date`, с полями: `day`, `month` и `year`.
   - Объявить и инициализировать переменную `a` даты в функции `main`.
   - Объявить и инициализировать массив дат под названием `holidays` следующими значениями `31.12.2019`, `8.3.2020` и `9.5.2020`.
   - Написать функцию `void print_date(struct date x)` для печати этой структуры в формате DD.MM.YYYY. Используйте модификатор `%02d`. Вызовите эту функцию из main, чтобы напечатать все элементы массива `holidays`.
   - Используйте `typedef`, чтобы сделать имя типа короче.

     ```c
     typedef struct date Date;
     ```

     Измените все имена типов с `struct date` на `Date`.

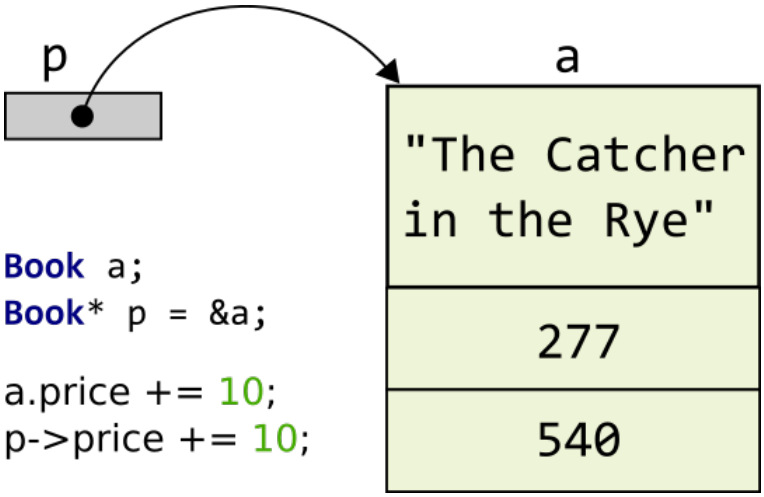
2. **Структура Фильм:**

   - **Описание структуры:** Описать структуру `Movie` с полями:
     - `title` – название фильма
     - `running_time` – длительность в минутах
     - `rating` – оценка на Кинопоиске
     - `release_date` – дата выхода (используйте структуру `Date`).
   - **Инициализация структуры:** Объявить переменную типа `Movie` в функции `main` и инициализировать её следующими значениями:
     `title - ‘‘Joker’’, running_time - 122, rating - 7.98, release_date - {3, 10, 2019}`.
   - **Доступ с полю структуры:** В новой строке изменить рейтинг и месяц выхода фильма. Используйте оператор точка (`.`).
   - **Печать:** Написать функцию `print_movie(Movie m)` и вызвать её в функции `main()`.
   - **Массив структур:** Объявить и инициализировать массив, содержащий 10 различных фильмов.
   - **Печать массива структур:** Написать функцию `print_movie_array(Movie* movies, int n)`, который бы печатал массив структур `Movie` и вызвать её в функции `main()`.
   - **Средний рейтинг:** Написать функцию, которая по массиву фильмов находит средний рейтинг.

# Указатели на структуры:

```c
#include <stdio.h>
#include <string.h>
struct book
{
    char title[50];
    int pages;
    float price;
};
typedef struct book Book;

void change_price(Book* p, float new_price) // Передача по указателю
{
        (*p).price = new_price;
}

void print_book_info(const Book* p) // Передача по константному указателю
{
    printf("Book info:\n");
    printf("Title: %s\nPages: %d\nPrice: %g\n\n", p->title, p->pages, p->price);
}

int main()
{
    Book a = {"The Martian", 100, 550.0};
    Book* p = &a;

    (*p).pages += 10;
    p->price = 400.0;
    strcpy(p->title, "The Catcher in the Rye");
    change_price(&a, 700);
    print_book_info(&a);
}
```

## Указатели на структуры. Задачи:




- **Указатель на структуру:** Создать указатель `Movie*` и присвоить ему адрес переменной типа `Movie`. Изменить поле running_time, используя только указатель. Используйте либо оператор точка (`.`) либо оператор стрелочка (`->`).
- **Передача по адресу:** Написать функцию `change_rating(Movie* pm, float new_rating)` и вызвать её в функции `main`.
- **Считывание:** Написать функцию `scan_movie(Movie* m)` и вызвать её в функции `main`. Функция
- **Поиск лучшего фильма:** Написать функцию, которая принимает на вход массив фильмов и возвращает указатель на фильм с самым высоким рейтингом.
- **Сортировка структур:** Одна из простейших сортировок - это сортировка выбором:

```
void selection_sort(int n, int arr[])
{
        for (int j = 0; j < n; j++)
        {
                // Находим индекс минимального элемента на отрезке [j:n-1]
                int min_index = j;
                for (int i = j+1; i < n; i++)
                        if (arr[i] < arr[min_index])
                                min_index = i;

                // Меняем местами элемент номер j и минимальный элемент
                int temp = arr[j];
                arr[j] = arr[min_index];
                arr[min_index] = temp;
        }
}
```

  Видоизмените эту сортировку так, чтобы она сортировала фильмы по рейтингу (от большего к меньшему).

- **Сортировка по алфавиту:** Отсортируйте структуры по их названию в алфавитном порядке. Используйте функцию `strcmp` из `string.h`. Функция `strcmp(a, b)` возращает `0`, если строки равны, отрицательное число если строка `a` меньше, чем строка `b` и положительное число, если строка `a` больше, чем `b`.
- **Считывание из файла:** Создайте файл `movies.txt`, который будет хранить информацию о фильмах. Запишите туда 10 фильмов (используйте текстовый редактор). Разделяйте поля с использованием точки с запятой и считывайте строки с помощью спецификатора `%[^;]`. Напишите программу, которая будет считывать фильмы из файла, записывать их в массив, сортировать и записывать в новый файл.

## Справочная информация по указателям:

Каждая переменная в языке C хранится где-то в памяти и имеет адрес. Адрес переменной это просто номер первого байта соответствующей области памяти. Чтобы получить адрес переменной нужно перед переменной поставить &(амперсанд). Указатель это переменная, которая хранит адреса переменных. Тип указателя такой: <тип переменной>*. Пример:

```
int a = 42;  // Переменная, которая хранит число 42
int* address_of_a = &a; // Указатель, который будет хранить адрес переменной a
```

Чтобы доступиться к переменной по указателю нужно поставить символ * перед указателем
a и *address_of_a это абсолютно одно и то же. a == *address_of_a.

```
*address_of_a = *address_of_a + 10;
printf("%d", a);  // Напечатает 52
printf("%d", *address_of_a); // Напечатает 52
```

Указатели часто используются чтобы изменять передаваемые значения в функциях:

```
// Неправильно:
void normalize(float x, float y)
{
    float sum = x + y;
    x = x / sum;
    y = y / sum;
    // Изменятся x и y - копии a и b
}
// ...
float a = 20.0, b = 80.0;
normalize(a, b);
// a и b не изменятся: a=20.0, b=80.0
```

```
// Правильно:
void normalize(float* px, float* py)
{
    float sum = *px + *py;
    *px = *px / sum;
    *py = *py / sum;
    // Изменятся переменные a и b
}
// ...
float a = 20.0, b = 80.0;
normalize(&a, &b);
// a и b изменятся:a=0.2, b=0.8
```

## Типы передачи в функцию:

1. По значению. То что передаётся в функцию копируется. При изменении копий, оригинал не меняется.

   ```
   void func(int a)
   ```

2. По указателю. В функцию копируется адрес переменной. Используя этот адрес, можно изменить оригинал.

   ```
   void func(int* p)
   ```

3. По постоянному указателю. В функцию копируется адрес переменной, но изменять оригинал с помощью этого указателя запрещено. Используется для того чтобы передать в функцию переменую большого размера (например структуру) и если вы не хотите изменять её внутри функции. Помогает избежать лишнего копирования.

   ```
   void func(const int* p)
   ```