Семинар #5: Структуры. Классные задачи.

Часть 1: Основы структур

Структуры служат для объединения нескольких типов в один. В примере ниже был создан новый тип под названием `struct point`. Переменные этого типа будут содержать внутри себя 2 значения типа `float`.

```
#include <stdio.h>

struct point {
    float x, y;
}; // <--------------------------- Не забудьте тут точку с запятой!

int main() {
    struct point a = {2.1, 4.3};
    a.x = 7.8;
    printf("(%f, %f)", a.x, a.y);
}
```

Операции со структурами

1. При создании структуры её элементы можно инициализировать с помощью фигурных скобочек.

   ```
   struct point a = {2.1, 4.3};
   ```

   Однако нельзя таким образом присваивать

   ```
   a = {5.6, 7.8}; // Ошибка, так можно только инициализировать
   ```

2. Доступ к элементу структуры осуществляется с помощью оператора точка

   ```
   a.x = 5.6;
   a.y = 7.8;
   ```

3. Структуры можно присваивать друг другу. При этом происходит побайтовое копирование содержимого одной структуры в другую.

   ```
   struct point b;
   b = a;
   ```

Массив структур

Структуры, как и обычные переменные, можно хранить в массивах. В примере ниже создан массив под названием `array`, содержащий в себе 2 точки.

```
#include <stdio.h>
struct point {
    float x, y;
};
int main() {
    struct point array[2] = {{2.1, 4.3}, {7.0, 3.1}};
    array[1].x = 1.8;
    printf("(%f, %f)", array[0].x, array[0].y);
}
```

Задачи

- Описать структуру `struct date`, с полями: `day`, `month` и `year`.
- Объявить и инициализировать переменную `a` типа `struct date` в функции `main`.
- Объявить и инициализировать массив дат под названием `holidays` следующими значениями `31.12.2021`, `8.3.2022` и `9.5.2022`.
- Напечатать содержимое массива `holidays` на экран с помощью цикла в следующем виде:

```
31.12.2021
08.03.2022
09.05.2022
```

Передача структуры в функцию

Структуры можно передавать в функции и возвращать из функций также как и обычные переменных. При передаче в функцию происходит полное копирование структуры и функция работает уже с копией структуры. При возвращении из функции также происходит копирование. Для того чтобы избежать лишние копирования нужно передавать структуру по указателю (об этом – в части 3).

```c
#include <stdio.h>

struct point {
    float x, y;
};
void print_point(struct point a) {
    printf("(%f, %f)", a.x, a.y);
}
struct point add_points(struct point a, struct point b) {
    struct point result;
    result.x = a.x + b.x;
    result.y = a.y + b.y;
    return result;
}
int main() {
    struct point a = {2.1, 4.3}, b = {6.7, 8.9};
    struct point c = add_points(a, b);
    print_point(c);
}
```

Задачи

- Написать функцию `void print_date(struct date a)` для печати этой структуры в формате `DD.MM.YYYY`. Используйте модификатор `%02d`. Вызовите эту функцию из main, чтобы напечатать все содержимое `holidays`.
- Написать функцию `struct date pushkin_birthday()` которая создаёт дату, соответствующую дню рождения А. С. Пушкина (6 июня 1799 года) и возвращает её. Протестируйте эту функцию в `main`.
- Написать функцию `struct date create_date(int day, int month, int year)` которая создаёт дату, по трём переданным в функцию числам и возвращает её. Протестируйте эту функцию в `main`.
- Написать функцию `struct date next_day(struct date a)` которая увеличивает значение даты на один день и возвращает эту структуру. Для простоты не учитываем високосные года и считаем, что в феврале всегда 28 дней. Вызовите эту функцию из `main`, чтобы увеличить значение даты `holidays[0]` на 1 день.

Часть 2: Структуры содержащие более сложные типы данных

Структуры могут содержать в себе не только базовые типы данных, но и более сложные типы, такие как массивы (в том числе строки), указатели, а также другие структуры.
Пример программы, в которой описывается структура для удобной работы с объектами Книга (`struct book`).

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

struct book {
    char title[50];
    int pages;
    float price;
};
void print_book(struct book b) {
    printf("Book info:\n");
    printf("Title: %s\nPages: %d\nPrice: %g\n\n", b.title, b.pages, b.price);
}
int main() {
    // Создаём книгу и печатаем её:
    struct book a = {"The Martian", 10, 550.0};
    print_book(a);

    // Меняем количество страниц книги и её название и снова печатаем её
    a.pages = 369;
    strcpy(a.title, "The Catcher in the Rye");
    print_book(a);

    // Пример работы с массивом структур
    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(scifi_books[2]);
}
```

Задачи:

- Описать структуру `struct movie` с полями:
  - `title` – название фильма (строка длиной не более 50 символов).
  - `running_time` – длительность в минутах (`int`)
  - `rating` – оценка на Кинопоиске (`float`)
  - `release_date` – дата выхода (используйте структуру `Date`).
- Объявить переменную типа `struct movie` в функции `main` и инициализировать её следующими значениями: `title - ‘‘Joker’’, running_time - 122, rating - 7.98, release_date - {3, 10, 2019}`.
- В новых строках изменить рейтинг и месяц выхода фильма. Используйте оператор точка.
- Написать функцию `void print_movie(struct movie m)` и вызвать её в функции `main()`.
- Написать функцию `struct movie get_titanic()` которая будет возвращать структуру с полями `title - ‘‘Titanic’’, running_time - 194, rating - 8.4, release_date - {1, 11, 1997}`. Вызовите эту функцию из `main` и напечатайте результат возвращаемого значения.
- Объявить и инициализировать массив, содержащий 10 различных фильмов. Решение этой задачи есть ниже и в файле `movie_array_init.txt`. Просто скопируйте код.

- В новой строке изменить день выхода фильма `Pulp Fiction` с 19-го мая на 21-е мая.
- Написать функцию `print_movie_array(struct movie array[], int size)`, которая бы печатала массив структур `Movie` и вызвать её в функции `main()`.
- Написать функцию, которая по массиву фильмов находит средний рейтинг. Протестируйте её в `main`.
- **Сортировка структур:** Одна из простейших сортировок - это сортировка выбором:

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

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

  Видоизмените эту сортировку так, чтобы она сортировала фильмы по рейтингу (от большего к меньшему). Помните, что структуры можно присваивать целиком, а не поэлементно.

- **Сортировка по алфавиту:** Отсортируйте структуры по их названию в алфавитном порядке. Используйте функцию `strcmp` из `string.h`. Функция `strcmp(a, b)` возвращает `0`, если строки равны, отрицательное число если строка `a` меньше, чем строка `b` и положительное число, если строка `a` больше, чем `b`.

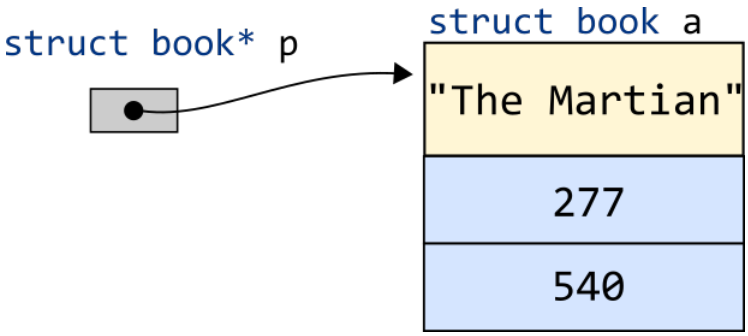
```
struct movie array[10] = {{"The Godfather", 175, 8.735, {14, 3, 1972}},
                          {"The Shawshank Redemption", 142, 9.112, {10, 9, 1994}},
                          {"Fight Club", 175, 8.651, {10, 9, 1999}},
                          {"The Matrix", 131, 8.491, {24, 3, 1999}},
                          {"Pulp Fiction", 154, 8.620, {19, 5, 1994}},
                          {"Citizen Kane", 119, 7.826, {1, 5, 1941}},
                          {"A Clockwork Orange", 137, 7.959, {19, 12, 1971}},
                          {"2001: A Space Odyssey", 149, 7.988, {2, 4, 1968}},
                          {"Finding Nemo", 175, 7.862, {30, 05, 2003}},
                          {"Vzlomat blogerov", 90, 1.029, {10, 11, 2016}}};
```

Часть 3: Указатели на структуры:

Указатель на структуру хранит адрес первого байта структуры. Для доступа к полям структуры по указателю нужно сначала этот указатель разыменовать, а потом использовать: `(*p).price`. Для удобства был введён оператор стрелочка `->`, который делает то же самое: `p->price`.

```c
#include <stdio.h>

struct book {
    char title[50];
    int pages;
    float price;
};
int main() {
    struct book a = {"The Martian",277,540};
    struct book* p = &a;
    // Три способа доступа к полю:
    a.price += 10;
    (*p).price += 10;
    p->price += 10;
}
```

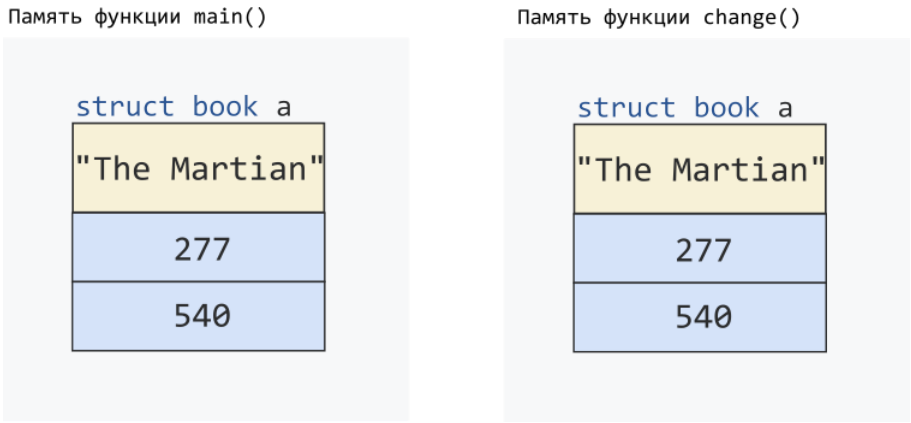


- Измените первую букву поля `title` структуры, используя только указатель `p`.

Передача по значению

При обычной передаче в функцию всё содержимое копируется. Функция работает с копией.

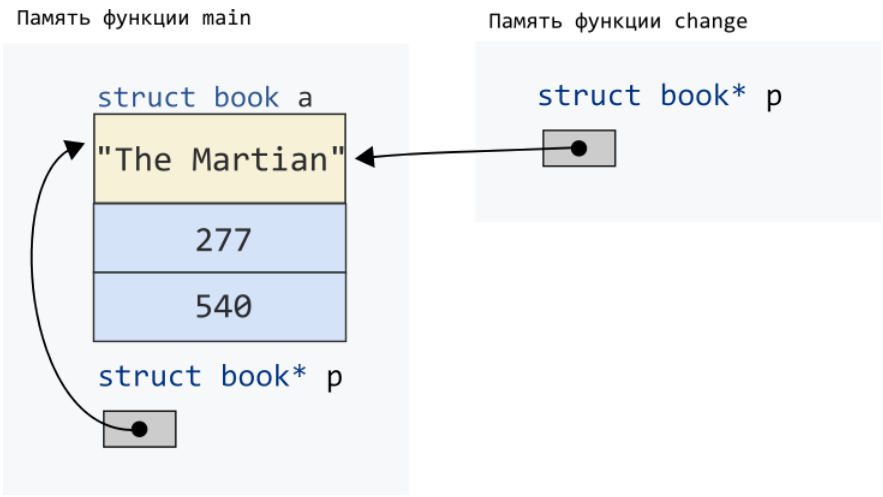
```c
#include <stdio.h>
struct book {
    char title[50];
    int pages;
    float price;
};
void change(struct book a) {
    a.price += 10;
}
int main() {
    struct book a = {"The Martian",277,540};
    change(a);
}
```



Передача по указателю

При передаче в функцию по указателю копируется только указатель.

```
#include <stdio.h>
struct book {
    char title[50];
    int pages;
    float price;
};
void change(struct book* p) {
    p->price += 10;
}
int main() {
    struct book a = {"The Martian",277,540};
    struct book* p = &a;
    change(p);
}
```



Такой способ передачи имее 2 преимущества:

1. Можно менять структуру внутри функции, и изменения будут действительны вне функции
2. Не приходится копировать структуры, поэтому программа работает быстрее.

Передача по указателю на константу

Иногда мы не хотим менять структуру внутри функции, но хотим чтобы ничего не копировалось. Тогда желательно использовать передачу по указателю на константу.

```
#include <stdio.h>
struct book {
    char title[50];
    int pages;
    float price;
};
void print_book_info(const struct book* p) {
    printf("Title: %s\nPages: %d\nPrice: %g\n\n", p->title, p->pages, p->price);
}
int main() {
    struct book a = {"The Martian",277,540};
    change(&a);
}
```

- Создать указатель `struct movie*` и присвоить ему адрес переменной типа `struct movie`. Изменить поле `running_time`, используя только указатель. Используйте либо оператор точка (`.`) и оператор стрелочка (`->`).
- Написать функцию `change_rating(struct movie* pm, float new_rating)` и вызвать её в функции `main`.
- **Поиск лучшего фильма:** Написать функцию, которая принимает на вход массив фильмов и возвращает указатель на фильм с самым высоким рейтингом. Протестируйте в функции `main`.
- **Считывание:** Написать функцию `scan_movie(struct movie* m)` и вызвать её в функции `main`. Функция должна считывать фильм из стандартного входа с помощью `scanf` (каждое поле нужно считать отдельно).

Часть 4: Выравнивание

Пусть есть структура `struct test` и нам нужно узнать её размер если размеры типов `char`, `int` и `double` равны `1`, `4` и `8` байт соответственно.

```
struct test {
    int a;
    char b;
    double c;
};
```

Кажется, что размер этой структур равен сумме размеров состовляющих её элементов, то есть `13`. Но это не так. На самом деле размер этой структуры будет отличаться в зависимости от вычислительной системы, на которой запускается код (как, впрочем, и размеры других типов). Но на большинстве вычислительных систем размер структуры `struct test` будет больше суммы состовляющих её элементов. Это можно проверить с помощью следующего кода:

```
int main() {
    printf("Size of char   = %llu\n", sizeof(char));
    printf("Size of int    = %llu\n", sizeof(int));
    printf("Size of double = %llu\n", sizeof(double));
    printf("Size of test   = %llu\n", sizeof(struct test));
}
```

Причина по которой это происходит заключается в том, что система значительно быстрей работает с данными, если они лежат в памяти по адресам, кратным 4-м или 8-ми. Поэтому компилятор автоматически выравнивает элементы структуры в памяти так, чтобы их адреса были кратны некоторой степени двойки.

Проверьте чему будет равен размер структуры `struct test` в зависимости от последовательности её полей.