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* address_of_x, float* address_of
{
    float sum = *address_of_x + *address_of_y;
    *address_of_x = *address_of_x / sum;
    *address_of_y = *address_of_y / 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
```

Задачи:

1. **Работа с указателями**
   (a) Объявить переменную типа float и инициализировать её какими-либо значениями
   (b) Напечатать значение и адрес переменной, используя эту переменную (чтобы напечатать адрес используйте спецификатор %p)
   (c) Объявить указатель типа float* и присвоить ему адрес переменной
   (d) Напечатать значение и адрес переменной, используя только указатель
   (e) Изменить значение переменной используя только указатель и напечатать это значение
2. **Modify1:** Написать функцию `void add10(int* p)`, которая добавляет 10 к переменной типа int. Используйте эту функцию в функции main() следующим образом:

   ```
   int a = 50;
   add10(&a);
   printf("%d\n", a);
   ```

3. **Cube:** Написать функцию `double cube(double p)`, которая возвращает куб числа. Используйте эту функцию в функции main() чтобы возвести в куб переменную типа double.
4. **Modify2:** Написать функцию `void cube(double* p)`, которая возводит значение переменной типа double в куб, используя указатель на эту переменную. Используйте эту функцию в функции main() чтобы возвести в куб переменную типа double.
5. **Swap:** Написать функцию `swap`, которая меняет значения 2-х переменных типа int местами. Используйте эту функцию в функции main().

Справочная информация по структурам:

```c
// Описываем структуру книги
struct book
{
    char title[50];
    int pages;
    float price;
}; // <-- НЕ ЗАБУДЬТЕ ТОЧКУ С ЗАПЯТОЙ
// Чтобы писать Book вместо struct book
typedef struct book Book;

// Функция для печати информации о книге
// Происходит передача по значению
// Используется оператор . точка()
void print_book_info(Book b)
{
    printf("\nBook info:\n");
    printf("Title: %s\n", b.title);
    printf("Pages: %d\n", b.pages);
    printf("Price: %f\n", b.price);
}

// Функция, которая изменяет цену книги
// Происходит передача через указатель
// Используется оператор -> стрелочка()
void change_price(Book*pb,float new_price)
{
    (*pb).price = new_price;
}
```

```c
int main()
{
 // Создадим переменную a типа Book
 // Её поля не заданы, может быть мусор
 // Но их можно будет задать позднее
 Book a;
 // Создадим переменную b типа Book
 // и сразу её инициализируем
 Book b = {"The Martian", 10, 550.0};
 // К полям структуры можно получить
    доступ
 // с помощью оператора . точка()
 b.pages = 369;
 // Массив книг, может содержать до 100
 // Сейчас там 3 книги, остальное -- мусор
 Book scifi_books[100] = {
     {"The Dark Tower", 300, 500.0},
     {"Fahrenheit 451", 400, 700.0},
     {"Day of the Triffids", 304, 450.0}
 };
 // Используем функцию print_book_info()
 print_book_info(scifi_books[2]);

 // Используем функцию change_price()
 // Обратите внимание на амперсанд
 change_price(&scifi_books[0], 2000.0);

 // Конечно, можно было сделать и так:
 scifi_books[0].price = 2000.0;
}
```

Задачи:

1. **Структура даты:** 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`.
    - Написать функцию `void scan_date(struct date* px)` для считывания этой структуры в формате DD.MM.YYYY. Вызовите эту функцию из main, чтобы считать переменную `a` из стандартного входа. Напечатайте эту переменную с помощью `print_date`.
    - Используйте `typedef`, чтобы сделать имя типа короче.

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 - 8.37, release_date - {3, 10, 2019}`.
- **Доступ с полю структуры:** В новой строке изменить рейтинг и месяц выхода фильма. Используйте оператор точка (.).
- **Указатель на структуру:** Создать указатель `Movie*` и присвоить ему адрес созданной переменной. Изменить поле running_time, используя только указатель. Используйте оператор точка (.).
- **Печать:** Написать функцию `print_movie(Movie m)` и вызвать её в функции `main()`.
- **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`.
- **Массив структур:** Объявить и инициализировать массив, содержащий 10 различных фильмов.
- **Печать массива структур:** Написать функцию `print_movie_array(int n, Movie m[])`, который бы печатал массив структур `Movie` и вызвать её в функции `main()`.
- **Средний рейтинг:** Написать функцию, которая по массиву фильмов находит средний рейтинг.
- **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`.
- **Считывание из файла:** Создайте файл `movies.txt`, который будет хранить информацию о фильмах. Запишите туда 10 фильмов (используйте текстовый редактор). Как считывать:

```
FILE* input_file = fopen("movies.txt", "r");
fscanf(input_file, /*  также как и в scanf  */);
fclose(input_file);
```

  Напишите программу, которая будет считывать фильмы из файла, записывать их в массив, сортировать и записывать в новый файл.