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* p = &a; // Указатель, который будет хранить адрес переменной a
```

Чтобы доступиться к переменной по указателю нужно поставить символ * перед указателем:

```
*p = *p + 10;
printf("%d", a);  // Напечатает 52
printf("%d", *p); // 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* x, float* y)
{
    float sum = *x + *y;
    *x = *x / sum;
    *y = *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().
6. **Умножение массива на число:** Написать функцию **mult_array**, которая принимает на вход массив и некоторое число и умножает все элементы массива на это число.

Справочная информация по структурам:

```
// Описываем структуру книги
struct book
{
    char title[50];
    int pages;
    float price;
};
// 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;
}
```

```
int main()
{
    // Создадим переменную типа Book по имени a
    // Её поля не заданы, там может быть мусор
    // Но их можно будет задать позднее
    Book a;

    // Создадим переменную типа Book по имени b
    // 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 книг
    // Сейчас там 3 книги, остальное -- мусор
    Book scifi_books[100] = {
        {"The Dark Tower", 300, 500.0},
        {"Fahrenheit 451", 400, 700.0},
        {"The 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. **Date:** Описать структуру Date, с полями: day, month и year. Написать функцию void print_date(Date x) для печати этой структуры в формате DD.MM.YYYY
2. **Movie:**
   (a) Описать структуру Movie с полями:
   - title – название фильма
   - running_time – длительность в минутах
   - rating – оценка на Кинопоиске
   - release_date – дата выхода

   (b) Объявить переменную типа Movie в функции main и инициализировать её следующими значениями: title – “Blade Runner 2049”, running_time – 163, rating – 7.98, release_date – {3, 10, 2017}.
   (c) Отдельными командами изменить рейтинг и месяц выхода фильма.
   (d) Создать указатель Movie* и присвоить ему адрес созданой переменной. Изменить поле running_time, используя только указатель.
   (e) Написать функцию print_movie_info(Movie m) и вызвать её в функции main().
   (f) Написать функцию change_rating(Movie* pm, float new_rating) и вызвать её в функции main().
   (g) Объявить и инициализировать массив, содержащий 4 различных фильма.
   (h) Написать функцию, которая по массиву фильмов находит средний рейтинг.
   (i) Написать функцию, которая принимает на вход массив фильмов и возвращает указатель на фильм с самым высоким рейтингом.