Справочная информация:

Спецификатор	Тип	Размер (байт)
%d или %i	int	4
%u	unsigned int	4
%l	long	8
%ul	unsigned long	8
%ll	long long	8
%ull	unsigned long long	8

Спецификатор	Тип	Размер (байт)
%f	float	4
%f	double	8
%p	указатель (<имя типа>*)	8
%c	char	1
%s	Строка	

Задачи:

Функции не должны ничего считывать и печатать.

1. **Произведение чисел:** Написать программу, которая считывает 2 числа a и b и печатает их произведение. $0 \leq a, b \leq 2^{32} - 1$. Обратите внимание на диапазон значений типа.

2. **mod 7:** Написать программу, которая печатает все числа делящиеся на 7 в интервале от 700 до 1000, используя цикл for.

3. **Часть года:** Написать функцию на вход которой подаётся целое число – число дней прошедших с начала года. Она должна возвращать вещественное число типа float – доля прошедшего года(от 0 до 1). В году 365 дней.

4. **Математическая функция:** Написать функцию, которая вычисляет выражение $\sin(\sqrt{x})$ от положительного числа x.

5. **Нормализация:** На вход программе подаётся целое число n и n вещественных чисел типа float. Нужно эти числа нормировать (то есть разделить на их сумму) и напечатать.

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 просто 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 int 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 int* 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().
3. **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() следующим образом:

   ```
   double x = 0.99;
   while (x)
   {
       cube(&x);
       printf("%.200f\n", x);
   }
   ```

4. **Swap:** Написать функцию **swap**, которая меняет значения 2-х переменных типа int местами. Используйте эту функцию в функции main().
5. **Sqared Matrix:** Написать функцию **void matrix_square(int n, int arr[SIZE][SIZE])**, которая возводит двумерную матрицу в квадрат. SIZE – максимально возможный размер массива, задаётся так:

   ```
   #define SIZE 100
   ```