

## Справочная информация:

### Типы и их спецификаторы в printf/scanf:

Типы данных в C. Приведенны размеры для 64-х битных систем.

Спецификатор	Тип	Размер (байт)	Спецификатор	Тип	Размер (байт)
%h	short	2	%c	char	1
%uh	unsigned short	2	%c	unsigned char	1
%d или %i	int	4	%f	float	4
%u	unsigned int	4	%lf	double	8
%ld	long	8 (или 4)	%Lf	long double	обычно 10
%lu	unsigned long	8 (или 4)	%g	%f, без нулей на конце	
%lld	long long	8	%p	указатель (<имя типа>*)	8
%llu	unsigned long long	8	%s	Строка	

Многие функции в языке C возвращают особый тип `size_t`. Часто это просто unsigned long:

```
typedef unsigned long size_t;
```

### Функции:

Пример функции вычисляющей  $\cos(|x| + |y|)$ :

```
#include <math.h>
float cos_abs(float x, float y)
{
    float result = cos(fabs(x) + fabs(y));
    return result;
}
```

### Массивы:

```
#include <stdio.h>

// Функция, которая вычисляет сумму элементов массива
// В функцию передаём n - размер массива и сам массив input_array
float get_sum(int n, float input_array[])
{
    float sum = 0.0;
    for (int i = 0; i < n; ++i)
        sum += input_array[i];
    return sum;
}

int main()
{
    int n;          // Объявляем n - это число будет хранить число элементов в массиве
    float arr[100]; // Объявляем массив вещественных чисел типа float
    scanf("%d", &n); // Считываем n
    for (int i = 0; i < n; ++i) // Считываем элементы массива
    {
        scanf("%f", &arr[i]);
    }
    printf("%f\n", get_sum(n, arr)); // Вычисляем и печатаем сумму
}
```

## Двумерный массивы:

```
#include <stdio.h>

#define SIZE 100

// Функция, которая вычисляет сумму 2-х матриц и записывает результат в третью
// Помните, что массивы могут меняться внутри функции
void sum(int n, int m, int m1[SIZE][SIZE], int m2[SIZE][SIZE], int result[SIZE][SIZE])
{
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            result[i][j] = m1[i][j] + m2[i][j];
}

int main()
{
    int n, m;           // Объявляем n и m - размеры массива
    int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE]; // Объявляем 3 массива
    scanf("%d%d", &n, &m); // Считываем n и m

    for (int i = 0; i < n; ++i) // Считываем элементы массива A
        for (int j = 0; j < m; ++j)
            scanf("%d", &A[i][j]);

    for (int i = 0; i < n; ++i) // Считываем элементы массива B
        for (int j = 0; j < m; ++j)
            scanf("%d", &A[i][j]);

    sum(n, m, A, B, C); // Вычисляем сумму A и B и записываем результат в C

    for (int i = 0; i < n; ++i) // Печатаем C
    {
        for (int j = 0; j < m; ++j)
            printf("%d ", C[i][j]);
        printf("\n");
    }
}
```

## Оператор switch:

```
int a;
scanf("%d", &a);
switch ( c )
{
    case 1:
        printf("One\n")
        break;
    case 2:
        printf("Two\n")
        break;
    case 3:
        printf("Thee\n")
        break;
    default:
        printf("Other\n")
}
```

## Задачи:

Функции не должны ничего считывать и печатать. Во всех задачах с функциями нужно не только написать функцию, но и вызвать её из функции `main()`.

1. **Остаток:** Написать программу, которая считывает 2 числа  $a$  и  $b$  и печатает остаток деления  $a$  на  $b$ .  $0 \leq a, b \leq 2^{32} - 1$ . Использовать тип `unsigned int`.
2. **Остаток + if:** Написать программу, которая считывает 2 числа  $a$  и  $b$  и печатает остаток деления большего числа на меньшее.  $0 \leq a, b \leq 2^{32} - 1$ .
3. **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 типов.
4. **mod 7:** Написать программу, которая печатает все числа делящиеся на 7 в интервале от 700 до 1000, используя цикл `for`.
5. **break:** Написать программу, которая считывает целые числа и печатает их до первого отрицательного. Например, если на вход поступает последовательность 5 0 74 -3 5 31 -7 -10, то программа должна напечатать 5 0 74. Использовать оператор `break`.
6. **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.
7. **Математическая функция:** Написать функцию, которая вычисляет выражение  $\sin(\sqrt{|x|})$ . Использовать числа двойной точности `double`. Функция для вычисления модуля вещественного числа – `fabs()` из библиотеки `math.h`.
8. **Единичный круг:** Написать функцию, которая проверяет принадлежат ли 2 вещественных числа единичному кругу. Использовать тип `double`.
9. **Гипербола:** Написать функцию, которая проверяет принадлежат ли 2 вещественных числа области  $\{y > \frac{1}{x}, x > 0\}$ . Использовать тип `double`. Функция должна возвращать 0 или 1. Тесты:  
( $x = 2.0, y = 4.0$ ) -> 1;  
( $x = 0.0, y = 2.0$ ) -> 0;  
( $x = 0.0, y = 0.0$ ) -> 0;  
( $x = -2.0, y = -5.0$ ) -> 0;
10. **Делимость:** На вход программе подаётся целое число  $n$  и  $n$  целых чисел типа `unsigned long`. Нужно напечатать 1 если все числа делятся на 7, в ином случае нужно напечатать 0.
11. **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.
12. **Среднее и дисперсия:** На вход программе подаётся целое число  $n$  и  $n$  вещественных чисел типа `double`  $x_i$ . Нужно найти среднее этих чисел  $\mu$  и дисперсию  $D$ :

$$\mu = \frac{1}{n} \sum_{i=0}^{n-1} x_i$$

$$D = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \mu)^2$$

13. **Таблица умножения:** Написать программу, которая распечатывает таблицу умножения.
14. **Sqared Matrix:** Написать функцию `void matrix_square(int n, int in[SIZE][SIZE], int out[SIZE][SIZE])`, которая возводит матрицу `in` в квадрат и записывает результат в матрицу `out`. `SIZE` – максимально возможный размер массива, задаётся так:

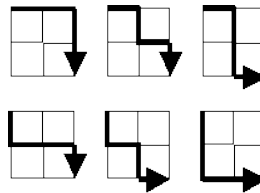
```
#define SIZE 100
```

15. **A\_n:** Написать функцию `void matrix_square(int n, int in[SIZE][SIZE], int out[SIZE][SIZE])`, которая возводит матрицу `in` в `n`-ю степень и записывает результат в матрицу `out`. Использовать функцию, написанную в предыдущей задаче.
16. **Matrix Multiply:** Написать функцию `void matrix_multiply(int k, int m, int n, int m1[SIZE][SIZE], int m2[SIZE][SIZE], int out[SIZE][SIZE])`, которая умножает матрицу `m1` размера  $k \times m$  на матрицу `m2` размера  $m \times n$  и записывает результат в матрицу `out`.
17. **switch:** Написать программу, которая по номеру дня недели (от 1 до 7) печатает название этого дня. Если число не входит в этот диапазон, то программа должна напечатать Error!

input	output
1	Monday
2	Tuesday
7	Sunday
100500	Error!

Что будет, если убрать все операторы `break`?

18. **Сколько дорог:** (Задача с ejudge -> Динамическое программирование) Из верхнего левого угла в правый нижний угол сетки  $2 \times 2$  можно пройти 6 разными путями (без возвратов, т.е. если идти только вниз или вправо). Сколько таких разных путей можно найти в сетке  $N \times M$ ?



Пусть  $C_{i,j}$  – число путей от точки  $(0, 0)$  до точки  $(i, j)$ . Как  $C_{i,j}$  зависит от  $C_{i-1,j}$  и  $C_{i,j-1}$ ?

19. **Черепашка:** В левом верхнем углу прямоугольной таблицы размером  $N \times M$  находится черепашка. В каждой клетке таблицы записано некоторое число. Черепашка может перемещаться вправо или вниз, при этом маршрут черепашки заканчивается в правом нижнем углу таблицы.

Подсчитаем сумму чисел, записанных в клетках, через которую проползла черепашка (включая начальную и конечную клетку). Найдите наибольшее возможное значение этой суммы.

Подсказка: Пусть  $C_{i,j}$  – наибольшее возможное значение суммы, если черепашка доползла до точки  $(i, j)$ . Как  $C_{i,j}$  зависит от  $C_{i-1,j}$  и  $C_{i,j-1}$ ?

Тест:

input	output
3 4	9
1 1 2 1	
2 2 1 1	
2 1 2 1	