

# Задания для подготовки к КР1. Информатика МФТИ 1 курс.

## Типы и их спецификаторы в printf/scanf:

тип	размер (байт)	диапазон значений ( $2^{\#bits}$ )	спецификатор
char	1	от -128 до 127	%hhd
short	2	от -32768 до 32767	%hd
int	4	примерно от -2-х миллиардов до 2-х миллиардов	%d
long	4 или 8	такой же как у int или long long в зависимости от системы	%ld
long long	8	примерно от $-10^{19}$ до $10^{19}$	%lld
unsigned char	1	от 0 до 255	%hhu
unsigned short	2	от 0 до 65535	%hu
unsigned int	4	примерно от 0 до 4-х миллиардов	%u
unsigned long	4 или 8	такой же как у unsigned int или unsigned long long	%lu
unsigned long long	8	от 0 до $2^{64} \approx 2 * 10^{19}$	%llu
16-ричная система	-	-	%x
указатель	8	$2^{64} \approx 2 * 10^{19}$	%p
float	4	6 значащих цифр, степень - от $10^{-38}$ до $10^{38}$	%f
double	8	15 значащих цифр, степень - от $10^{-308}$ до $10^{308}$	%lf
long double	10 или 12	зависит от системы	%Lf
печать без 0 на конце	-	-	%g
char (как символ)	1	от -128 до 128	%c
строка = массив char-ов	размер массива	-	%s

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

## Переменные и операторы:

- **Остаток (0.5 балла):** Ниже - пример программы, которая считывает 2 числа и печатает сумму этих чисел:

```
#include <stdio.h>
int main()
{
    int x, y;
    scanf("%d%d", &x, &y);
    printf("%d\n", x + y);
}
```

Написать программу, которая будет считывать 2 числа и печатать остаток деления первого числа на второе.

- **Значение, адрес и размер (0.5 балла):** Пример программы, которая создаёт и инициализирует переменную `a` типа `int` и печатает её значение, размер и адрес.

```
#include <stdio.h>
int main()
{
    int a = 753;
    printf("int: Value = %d; Size = %d; Address = %p\n", a, sizeof(a), &a);
}
```

Напишите одну программу, которая делает то же самое для следующих типов: `char`, `short`, `int`, `long`, `long long`, `float`, `double`, `char*`, `int*`, `float*`, `double*`, `int array[80]`. Программа должна создать переменные всех этих типов и напечатать значения, адреса и размеры каждой переменной.

## Оператор if:

Пример программы которая проверяет делится ли число на 2 при этом не лежит на отрезке [15, 100]:

```
#include <stdio.h>
int main()
{
    int a;
    scanf("%d", &a)
    if (a % 2 == 0 && (a < 15 || a > 100))
        printf("YES\n");
    else
        printf("NO\n");
}
```

- **Единичный круг (0.5 балла):** Написать программу, которая проверяет лежит ли точка, задаваемая двумя целыми числами  $x$  и  $y$  внутри круга радиуса  $R$ . Числа  $x$ ,  $y$  и  $R$  считать с помощью `scanf`. Программа должна печатать YES или NO.
- **Гипербола (0.5 балла):** Написать программу, которая проверяет принадлежат ли точка, задаваемая двумя вещественными числами, области  $\{y > \frac{1}{x}, x > 0\}$ . Использовать тип `double` (спецификатор для `double` - `%lf`).

## Циклы while и for. Операторы break и continue:

- **mod 7 (0.5 балла):** Написать программу, которая печатает все числа делящиеся на 7 в интервале от 700 до 1000, используя цикл `for`. (и не используя оператор `if`).
- **break (0.5 балла):** 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`.

## Математическая библиотека math.h:

Ниже - пример функции вычисляющей  $\cos(|x| + |y|)$ . ( $x$  и  $y$  задаются в градусах).

```
#include <stdio.h>
#include <math.h>

double func(double x, double y)
{
    return cos(3.14159265/180*(fabs(x) + fabs(y)));
}

int main()
{
    double x, y;
    scanf("%lf%lf", &x, &y);
    printf("%g\n", func(x, y)); // вызываем функцию
}
```

Чтобы подключить математическую библиотеку нужно использовать опцию `-lm` при компиляции в `gcc`:  
\$ gcc -std=c99 -lm <имя файла>

- **Математическая функция (0.5 балла):** Написать функцию, которая вычисляет выражение  $\sqrt{|\sin(x)|^7}$ . Использовать числа двойной точности `double`.  $x$  подаётся на вход в градусах. Вызвать эту функцию из `main`.
- **Сравнение вещественных чисел (0.5 балла):** Написать функцию `int is_equal(double a double b, double eps)`, которая сравнивает 2 числа  $a$  и  $b$  с точностью `eps`. Вызвать эту функцию из `main`.

## Массивы:

Пример программы, которая показывает как работать с массивами:

```
#include <stdio.h>
int main()
{
    int n;
    int array[1000]; // Выделяем память под 1000 элементов ( с запасом )
    // Считываем размер массива и его элементы:
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        scanf("%d", &array[i]);

    // Возводим каждый элемент в квадрат:
    for (int i = 0; i < n; i++)
        array[i] *= array[i];

    // Печатаем:
    for (int i = 0; i < n; i++)
        printf("%d ", array[i]);
}
```

- **Нормализация (0.5 балла):** 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.
- **Среднее и дисперсия (0.5 балла):** На вход программе подаётся целое число  $n$  и  $n$  вещественных чисел типа `double`  $x_i$ . Нужно найти среднее этих чисел  $\mu$  и дисперсию  $D$ :

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

## Массивы и функции:

Пример функции, которая принимает на вход массив:

```
#include <stdio.h>
int sum(int n, int array[])
{
    int result = 0;
    for (int i = 0; i < n; i++)
        result += array[i];
    return result;
}
int main()
{
    int numbers[6] = {4, 8, 15, 16, 23, 42};
    printf("Sum = %d\n", sum(6, numbers));
}
```

Помните что, в отличии от обычных переменных, массив внутри функции может меняться.

- **Делимость (0.5 балла):** На вход функции подаётся целое число  $n$  - размер массива и сам массив. Функция должна вернуть число 1 если все числа массива делятся на 7. Иначе, функция должна вернуть число 0. Проверить работу этой функции, вызвав её из функции `main`.

## Сортировка вставками:

```
#include <stdio.h>
// Сортировка вставками:
void selection_sort(int n, int array[])
{
    for (int i = 0; i < n; i++)
    {
        // Находим минимальный элемент
        int min_index = i;
        for (int j = i + 1; j < n; j++)
            if (array[j] < array[min_index])
                min_index = j;

        // Меняем первый и минимальный
        int temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }
}

int main()
{
    int numbers[10] = {54, 76, 83, 26, 17, 95, 43, 6, 54, 61};
    selection_sort(10, numbers);
    for (int i = 0; i < 10; i++)
        printf("%d ", numbers[i]);
}
```

- **Сортировка по последней цифре (0.5 балла):** Измените функцию `selections_sort` в примере выше так, чтобы она сортировала по последней цифре числа. То есть в результате должно получиться: 61, 83, 43, 54, 54, 95, 76, 26, 6, 17 (числа с одинаковой последней цифрой могут идти в любом порядке). Подсказка: нужно поменять всего 1 строчку.

## Структуры:

```
#include <stdio.h>
struct point
{
    float x, y;
};
typedef struct point Point;

int main()
{
    Point points[5] = {{5.2, 3.1}, {7.3, -3.4}, {-1.2, 6.4}, {3.8, 8.7}, {-0.6, 2.1}};
    for (int i = 0; i < 5; i++)
        printf("%g %g\n", points[i].x, points[i].y);
}
```

- **Структура Triangle (0.5 балла):** Определите структуру, которая будет описывать треугольник на плоскости.
- **Сортировка структур (1 балл):** Измените функцию `selections_sort` в примере выше так, чтобы она принимала на вход массив структур `Point` и сортировала их по расстоянию от начала координат. Используйте эту функцию, чтобы отсортировать массив `points` в функции `main`.

## Структуры и функции:

Структуры передаются в функции и возвращаются из функций по тем же правилам, что и обычные переменные:

```
#include <stdio.h>
struct point
{
    float x, y;
};
typedef struct point Point;

// Функция, которая будет находить точку, лежащую посередине между a и b
Point midpoint(Point a, Point b)
{
    Point result;
    result.x = (a.x + b.x) / 2;
    result.y = (a.y + b.y) / 2;
    return result;
}
int main()
{
    Point points[2] = {{5.2, 3.1}, {7.3, -3.4}};
    Point m = midpoint(points[0], points[1]);
    printf("%g %g\n", m.x, m.y);
}
```

- **Расстояние (0.5 балла):** Написать функцию `float distance(Point a, Point b)`, которая вычисляет расстояние между двумя точками. Вызовите эту функцию из `main`.
- **Центр масс (0.5 балла):** Написать функцию `Point center_of_mass(Triangle t)`, которая вычисляет центр масс треугольника. Вызовите эту функцию из `main`. Центр масс треугольника вычисляется просто как среднее арифметическое по всем координатам.

## Двумерные массивы:

Пример программы, вычисляющей сумму двух двумерных массивов:

```
#include <stdio.h>
#define MAX 100

// Функция, которая вычисляет сумму двух матриц и записывает результат в третью
// В отличие от одномерных массивов, тут обязательно нужно указывать максимальный размер
void sum(int n, int A[MAX][MAX], int B[MAX][MAX], int result[MAX][MAX])
{
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            result[i][j] = A[i][j] + B[i][j];
}
int main()
{
    int n;
    int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];
    scanf("%d", &n);

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            scanf("%d", &A[i][j]);
```

```

for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
        scanf("%d", &B[i][j]);

sum(n, 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 < n; ++j)
        printf("%d ", C[i][j]);
    printf("\n");
}
}

```

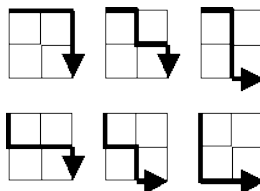
- **Перемножение матриц (1 балл):** Написать функцию `void mult(int n, int A[MAX][MAX], int B[MAX][MAX], int result[MAX][MAX])`, которая вычисляет произведение двух квадратных матриц (строка на столбец) и записывает результат в третью. Вызовите эту функцию из `main`.

## Динамическое программирование:

- **Кузнечик (0.5 балла):** Кузнечик находится в начале координат ( $x = 0$ ) и может прыгать по целым числам вправо либо с шагом 1 либо с шагом 2. Найдите число различных маршрутов кузнечика, приводящих его в точку с координатой  $n$ .

**Подсказка:** Создайте массив `int ways[]` и в каждой ячейке  $i$  храните число различных маршрутов до точки  $i$ . Очевидно, что `ways[1] = 1`, `ways[2] = 2`. До точки 3 можно добраться 3-мя различными способами ( $\{1,1,1\}$ ,  $\{1, 2\}$ ,  $\{2,1\}$ ), поэтому `ways[3] = 3`. Как найти `ways[k]`, зная предыдущие значения массива?

- **Кузнечик 2 (0.5 балла):** Пусть теперь кузнечик может прыгать на  $+1 +2 +3$ . Найдите число различных маршрутов кузнечика, приводящих его в точку с координатой  $n$ .
- **Сколько дорог (1 балл):** (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  $N \times M$ ?



**Подсказка:** Составьте табличку из решений следующих подзадач:  $C_{i,j}$  – число путей от точки  $(0, 0)$  до точки  $(i, j)$ . Чему равно  $C_{0,j}$  и  $C_{i,0}$ ? Как  $C_{i,j}$  зависит от  $C_{i-1,j}$  и  $C_{i,j-1}$ ?

- **Черепашка (1 балл):** 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:** Составьте табличку из решений следующих подзадач:  $C_{i,j}$  – наибольшее возможное значение суммы, если черепашка доползла до точки  $(i, j)$ . Чему равно  $C_{0,j}$  и  $C_{i,0}$ ? 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	
7 7	
1 1 10 1 1 1 1	
1 1 1 1 1 70 1	
5 5 5 1 1 1 1	
3 5 20 20 1 1 10	100
8 5 6 20 1 1 1	
7 5 7 5 7 1 1	
1 4 5 11 4 1 1	