

Семинар #3: Функции. Домашнее задание.

Задача 1: Куб

Напишите функцию `cube`, которая будет принимать на вход одно целое число и возвращать куб этого числа. Вызовите эту функцию в функции `main` следующим образом:

```
#include <stdio.h>

// Тут вам нужно написать
// функцию cube

int main()
{
    printf("%i\n", cube(5));
}
```

После того как вы напишете функцию `cube`, скомпилируйте данную программу и запустите она должна напечатать на экран число 125.

Задача 2: Четность

Напишите функцию `is_even`, которая будет принимать на вход одно целое число и возвращать 1, если это число чётное и 0, если число нечётное. Вызовите эту функцию в функции `main` следующим образом:

```
#include <stdio.h>

// Тут вам нужно написать
// функцию is_even

int main()
{
    printf("%i\n", is_even(90));
    printf("%i\n", is_even(91));
}
```

После того как вы напишете функцию `is_even`, скомпилируйте данную программу и запустите она должна напечатать на экран:

```
1
0
```

Задача 3: Печать чётных чисел

Напишите функцию `print_even`, которая будет принимать на вход два целых числа `a` и `b` и печатать на экран все чётные числа, которые находятся на отрезке `[a, b]`.

вход функции <code>print_even</code>	печать на экран
2 15	2 4 6 8 10 12 14
1 15	2 4 6 8 10 12 14
-7 3	-6 -4 -2 0 2

Задача 4: Треугольник из звёздочек

Напишите функцию `triangle`, которая будет принимать на вход одно целое положительно число `n` и будет печатать на экран прямоугольный треугольник из звёздочек (символов `*`). В функции `main` считайте с экраная число и вызовите функцию `triangle`, передав ей считанное число.

ВХОД	ВЫХОД
5	<pre>* ** *** **** *****</pre>
3	<pre>* ** ***</pre>
1	<pre>*</pre>

Задача 5: Сумма цифр числа

- Напишите функцию `sum_of_digits`, которая будет принимать на вход целое число и возвращать сумму всех цифр числа (в десятичной записи). Предполагайте, что на вход функции всегда будут приходить неотрицательные числа. Реализуйте эту функцию с помощью цикла.

вход функции <code>sum_of_digits</code>	выход функции <code>sum_of_digits</code>
123	6
55955	29
4	4
0	0

- Напишите функцию `sum_of_digits_rec`, которая будет принимать на вход целое число и возвращать сумму всех цифр числа (в десятичной записи). Предполагайте, что на вход функции всегда будут приходить неотрицательные числа. Реализуйте эту функцию с помощью рекурсии. Пользоваться уже написаной с помощью цикла функцией `sum_of_digits` в этой функции нельзя.

Задача 6: Бинарное представление числа

Напишите функцию `print_binary`, которая будет принимать на вход целое число и печатать на экран бинарное представление этого числа. Предполагайте, что на вход функции всегда будут приходить неотрицательные числа. Реализуйте эту функцию с помощью рекурсии.

вход функции <code>print_binary</code>	печать на экран
6	110
128	10000000
4823564	10010011001101000001100
0	0

Задача 7: Числа трибоначчи

Числа трибоначчи - это последовательность чисел, задаваемые следующим образом:

```
trib(0) = 0;
trib(1) = 0;
trib(2) = 1;
trib(n) = trib(n - 3) + trib(n - 2) + trib(n - 1);
```

Напишите функцию, `trib`, которая будет принимать на вход целое число `n` и будет возвращать `n`-е число трибоначчи. Убедитесь, что функция работает быстро при вычислении 38-го числа трибоначчи.

вход функции <code>trib</code>	выход функции <code>trib</code>
1	0
5	4
20	35890
35	334745777
38	2082876103

Задача 8: Количество чётных

Напишите функцию `count_even`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна возвращать количество чётных чисел в этом массиве. Протестируйте эту функцию в функции `main`.

вход функции <code>count_even</code>	выход функции <code>count_even</code>
array: 1 2 3 4 5 size: 5	2
array: 10 20 30 40 size: 4	4
array: 10 1 size: 2	1

Задача 9: Отнять единицу

Напишите функцию `dec`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна уменьшать каждый элемент этого массива на 1. Протестируйте эту функцию в функции `main`.

вход функции <code>dec</code>	массив после выполнения <code>dec</code>
array: 1 2 3 4 5 size: 5	0 1 2 3 4
array: 10 20 30 40 size: 4	9 19 29 39

Задача 10: Оставить последнюю цифру

Напишите функцию `last_digits`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна заменять каждый элемент массива на его последнюю цифру (в десятичной записи числа). Протестируйте эту функцию в функции `main`.

вход функции <code>last_digits</code>	массив после выполнения <code>last_digits</code>
array: 12 61 426 17 115 size: 5	2 1 6 7 5
array: 5 10 size: 2	5 0

Задача 11: Факториалы

Напишите функцию `factorials`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна заменять каждый элемент этого массива на его факториал. Напишите вспомогательную функцию `fact`, которая будет принимать одно целое число и возвращать факториал этого числа. Протестируйте эту функцию в функции `main`.

вход функции <code>factorials</code>	массив после выполнения <code>factorials</code>
array: 3 4 5 6 7 size: 5	6 24 120 720 5040
array: 10 11 12 size: 3	3628800 39916800 479001600

Задача 12: Обратный массив

Напишите функцию `reverse`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна переворачивать массив задом наперёд. Протестируйте эту функцию в функции `main`.

вход функции <code>reverse</code>	массив после выполнения <code>reverse</code>
array: 10 20 30 40 50 size: 5	50 40 30 20 10
array: 60 20 80 10 size: 4	10 80 20 60

Задача 13: Обратный подмассив

Напишите функцию `reverse_subarray`, которая будет принимать массив и два целых числа `left` и `right`. Эта функция должна переворачивать подмассив, задаваемый числами `left` и `right` задом наперёд. Протестируйте эту функцию в функции `main`.

вход функции <code>reverse_subarray</code>	массив после выполнения <code>reverse_subarray</code>
array: 10 20 30 40 50 60 70 80 90 100 left: 2; right: 6	10 20 60 50 40 30 70 80 90 100
array: 10 20 30 40 50 60 70 80 90 100 left: 4; right: 10	10 20 30 40 100 90 80 70 60 50
array: 10 20 30 40 50 60 70 80 90 100 left: 6; right: 8	10 20 30 40 50 60 80 70 90 100
array: 10 20 30 40 50 60 70 80 90 100 left: 6; right: 7	10 20 30 40 50 60 70 80 90 100

Задача 14: Сортировка

Напишите функцию `sort`, которая будет принимать на вход массив целых чисел и количество элементов этого массива. Эта функция должна сортировать все элементы массива по убыванию. Протестируйте эту функцию в функции `main`.

вход функции <code>sort</code>	массив после выполнения <code>sort</code>
array: 70 20 80 30 50 40 10 60 size: 8	80 70 60 50 40 30 20 10
array: 60 20 80 10 size: 4	80 60 20 10

Задача 15: Необычная рекурсия

Алиса и Боб играют в игру. Сначала Алиса получает некоторое нечётное число. Алиса умножает это число на 3 и прибавляет к результату умножения единицу. Получившееся число Алиса печатает на экран и передаёт Бобу. Боб получает чётное число и делит это число на 2 пока число не станет нечётным. После каждого деления на 2, Боб печатает число на экран. Как только Боб получит нечётное число, отличное от 1, он передаёт её Алисе. Но если Боб получит число 1, то программа заканчивает выполнение.

Например, если мы передадим на вход программе число 13, то она должна напечатать следующее:

```
Alice: 40
Bob: 20
Bob: 10
Bob: 5
Alice: 16
Bob: 8
Bob: 4
Bob: 2
Bob: 1
```

Эту программу очень просто написать с помощью обычного цикла. Но ваша задача заключается в том, чтобы написать её с помощью рекурсии, причём поведение Алисы и Боба должны описываться двумя разными функциями. Напишите следующие функции:

- `void alice(int n)`
- `void bob(int n)`

Эти функции должны изменять проходящее на вход число, печатать его на экран и передавать новое число другой функции. То есть функция `alice` должна вызывать функцию `bob`, а функция `bob` должна вызывать функцию `alice` (но только если новое нечётное число будет отлично от единицы).