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:

```c
#include <stdio.h>

// Тут вам нужно написать
// функцию cube

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

После того как вы напишите 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:

```c
#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 напишите 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 `print_even`	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 (к удалению)

Напищите 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	*
	**
	***
	****
	*****
3	*
	**
	***
1	*

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 `sum_of_digits`	выход функции `sum_of_digits`
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 `print_binary`	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 `trib`	выход функции `trib`
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 `count_even`	выход функции `count_even`
`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. Оставить последнюю цифру (к удалению)

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 `last_digits`	массив после выполнения `last_digits`
`array: 12 61 426 17 115` `size: 5`	2 1 6 7 5
`array: 5 10` `size: 2`	5 0

Задача 10. 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 `factorials`	массив после выполнения `factorials`
`array: 3 4 5 6 7` `size: 5`	6 24 120 720 5040
`array: 10 11 12` `size: 3`	3628800 39916800 479001600

Задача 11. 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 `reverse`	массив после выполнения `reverse`
`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 12. 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 `sort`	массив после выполнения `sort`
`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 13. 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. Получившиеся 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 приходящее 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).

Задача 14. Присвоение матриц

Написать функцию `void assign(float A[MAX][MAX], float B[MAX][MAX], int n)`, которая присваивает элементам матрицы A соответствующие элементы матрицы `B` (`A = B`). Где `MAX` это `#define`-константа.

Задача 15. Умножение матриц

Написать функцию `void multiply(float A[MAX][MAX], float B[MAX][MAX], float C[MAX][MAX], int n)`, которая перемножает матрицы `A` и `B` размера n на n (строка на столбец), а результат записывает в матрицу `C`. Формула: $C_{ij} = \sum_k A_{ik} \cdot B_{kj}$.

Проверьте ваш код на следующих тестах:

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} \cdot \begin{pmatrix} 5 & 2 & 9 \\ -4 & 2 & 11 \\ 7 & 1 & -5 \end{pmatrix} = \begin{pmatrix} 21 & 30 & 130 \\ -6 & 21 & 82 \\ 48 & 12 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 2 & 9 \\ -4 & 2 & 11 \\ 7 & 1 & -5 \end{pmatrix} \cdot \begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} = \begin{pmatrix} 55 & 60 & 70 \\ -4 & -1 & 64 \\ 40 & 52 & -13 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 7 & 7 \\ 3 & 8 & 1 \\ 6 & 1 & 2 \end{pmatrix}$$

В файлах `matA.txt` и `matB.txt` лежат матрицы 10 на 10. Считайте эти матрицы с помощью метода перенаправления потока из файла `combinedAB.txt`, перемножьте (A на B) и запишите результат в другой файл. В результате должно получиться:

$$\begin{pmatrix} 259 & -15 & 237 & 257 & 231 & 67 & 237 & -64 & 152 & 363 \\ 555 & 233 & 539 & 188 & 356 & 325 & 423 & -47 & 123 & 387 \\ 497 & 512 & 572 & 95 & 619 & 155 & 414 & 207 & 203 & 217 \\ 455 & 280 & 675 & 354 & 664 & 346 & 483 & 177 & 168 & 404 \\ 264 & 182 & 272 & 290 & 474 & -33 & 234 & 99 & 379 & 156 \\ 272 & 180 & 469 & 286 & 326 & 282 & 325 & 215 & 195 & 231 \\ 421 & 363 & 475 & 506 & 359 & 481 & 468 & 101 & 325 & 328 \\ 384 & 218 & 567 & 395 & 475 & 488 & 361 & 168 & 291 & 298 \\ 387 & 297 & 480 & 170 & 318 & 423 & 483 & 10 & -17 & 406 \\ 193 & 241 & 486 & 38 & 403 & 146 & 286 & 326 & 212 & 172 \end{pmatrix}$$

Задача 16. Матрица в степени

Написать функцию `void power(int A[MAX][MAX], int C[MAX][MAX], int n, int k)`, которая вычисляет A^k, т.е. возводит матрицу A размера n на n в k-ю степень, а результат записывает в матрицу C. Используйте функции `multiply` и `assign`. Псевдокод простейшей реализации такой функции:

```
float B[MAX][MAX]

B = A   // конечно нельзя приравнивать массивы, но можно
C = A   // использовать вашу функцию assign

for ( k - 1 раз )
{
        C = A*B
        B = C
}
```

Проверьте ваш код на следующих тестах:

$$\begin{pmatrix} 7 & 7 & 2 \\ 1 & 8 & 3 \\ 2 & 1 & 6 \end{pmatrix}^4 = \begin{pmatrix} 7116 & 15654 & 9549 \\ 4002 & 8955 & 6135 \\ 3369 & 6165 & 4350 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}^{70} = \begin{pmatrix} 145547525 & 109870576 & 82938844 \\ 192809420 & 145547525 & 109870576 \\ 109870576 & 82938844 & 62608681 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}^{1000} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Подумайте, как можно возвести матрицу в k-ю степень более эффективно. Рассмотрите случай, когда `k` является степенью двойки и когда не является.

Необязательные задачи

Задача 1. Метод Гаусса

Написать программу, которая бы решала линейную систему уравнений $Ax = b$ методом Гаусса.
Главная функция этой программы должна иметь вид:
`void solve_linear_system(int n, float A[MAX][MAX], float b[], float x[]).`
Программа должна считывать матрицу A размера n на n и столбец b из файла и записывать результат решения x в новый файл. Предполагайте, что детерминант матрицы не равен нулю. Примерный вид вашей программы:

```c
#include <stdio.h>
#define MAX 200

// Возможно понадобится вспомогательная функция для перестановки строк матрицы A
void swap_rows(float A[MAX][MAX], int n, int k, int m)
{
    // Ваш код
}

void solve_linear_system(float A[MAX][MAX], int n, float b[], float x[])
{
    // Ваш код
}

int main()
{
    int n;
    float A[MAX][MAX];
    float b[MAX];
    float x[MAX];

    // Считываем n, A и b
    solve_linear_system(n, A, b, x);
    // Печатаем x
}
```

Проверьте вашу программу на следующих тестах (считывайте из файла методом перенаправления потока):

1. Следующая система:
$$\begin{cases} x_1 + x_2 - x_3 = 9 \\ x_2 + 3x_3 = 3 \\ -x_1 - 2x_3 = 2 \end{cases}$$
Вход для программы должен выглядеть следующим образом:
```
 3
 1 1 -1
 0 1  3
-1 0 -2
9
3
2
```
Решение этой системы: $x = \left(\frac{2}{3}, 7, -\frac{4}{3}\right) \approx (0.67, 7.00, -1.33)$

2. Системы из файлов `system1.txt`, `system2.txt`, `system3.txt`. Решения в файлах с именам `x1.txt`, `x2.txt`, `x3.txt` соответственно. Считывайте из файлов методом перенаправления потока.