Семинар #2: Функции. Классные задачи.

Работа с терминалом:

Основные команды терминала:

`pwd`	напечатать имя текущей директории
`ls`	напечатать все файлы и папки текущей директории
`ls -l`	то же, что и ls, но больше информации о файлах
`cd <имя папки>`	перейти в соответствующую папку например: cd /home-local/student
`mkdir <имя новой папки>`	создать новую папку
`rm <путь до файла>`	удалить файл
`gcc <имя файла исходного кода>`	скомпилировать и сохранить в исполняемый файл a.out
`./a.out`	запустить исполняемый файл a.out
`gcc <имя исходного файла> && ./a.out`	скомпилировать и запустить

Задание 1:

1. Перейдите в терминале в вашу рабочую папку
2. Создайте в ней файл `function.c` с содержимым:

```c
#include <stdio.h>
void hello()
{
        printf("Hello\n");
}
int main()
{
        for (int i = 0; i < 10; i++)
                hello();
}
```

   Можно создать с помощью `nano` или `gedit`.

3. Скомпилируйте этот файл и запустите.

Функции без возвращаемого значения:

Пример программы, которая печатает натуральные числа от `1` до `n`:

```c
#include <stdio.h>
void print_numbers(int n)
{
        for (int i = 0; i < n; i++)
                printf("%d ", i);
}
int main()
{
        // Тут нужно дописать
}
```

1. Используйте функцию `print_numbers`, чтобы вывести на экран все числа от `0` до `200`.

2. Напишите функцию `print_even_numbers(int a, int b)`, которая будет печатать все четные числа от `a` до `b`. Вызовите эту функцию из функции `main`.

3. Напишите функцию `print_many_times(int a, int b)`, которая будет печатать число `a` `b` раз. Вызовите эту функцию из функции `main`.

4. Напишите функцию `print_rectangle(int a, int b)`, которая будет печатать прямоугольник из звёздочек `*`. Например, если эта функция будет вызвана с аргументами `4` и `3`, то функция должна напечатать:

   ```
   ****
   ****
   ****
   ```

   Вызовите эту функцию из функции `main` с различными аргументами.

5. Напишите функцию `void multi(int type, int a, int b)`, которая, в зависимости от переменной `type`, должна делать различные вещи. При `type == 1`, она должна вызывать функцию `print_even_numbers`. При `type == 2`, она должна вызывать функцию `print_many_times`. При `type == 3`, она должна вызывать функцию `print_rectangle`. При ином другом значении `type`, она должна просто печатать `Error!`. Протестируйте вашу функцию.

Функции с возвращаемым значением:

Пример функции, которая принимает на вход 2 числа и возвращает максимальное из них:

```c
#include <stdio.h>
// Функция возвращает значение типа int
int max(int a, int b)
{
        if (a > b)
                return a;
        else
                return b;
}
int main()
{
        printf("%d\n", max(10, 40));  // На место max(10, 40) подставится результат вычисления
        printf("%d\n", max(5, -14));
}
```

1. **Минимум:** Напишите функцию `int min(int a, int b)` и протестируйте её.

2. **Минимум из трёх:** Напишите функцию `int min(int a, int b, int c)` и протестируйте её. Подсказка: можно использовать функцию из предыдущей задачи.

3. **Doubler:** Написать функцию `doubler`, которая принимает число и возвращает это число, умноженное на 2. Вызвать эту функцию из функции main:

   ```c
   // Тут нужно написать функцию doubler
   int main()
   {
           int a;
           scanf("%d", &a);
           printf("%d\n", doubler(a));
   }
   ```

4. **Sum:** Напишите функцию `int sum(int n)`, которая вычисляет сумму первых n натуральных чисел. Вызовите эту функцию из main и протестируйте на следующих значениях: `10` (ответ - `55`), `100` (ответ - `5050`) и `1234` (ответ - `761995`).

5. **Среднее геометрическое** Пример программы, которая вычисляет среднее геометрическое (при компиляции не забудьте флаг `-lm`)

```c
#include <stdio.h>
#include <math.h>
float geometric_mean(float a, float b)
{
        return sqrt(a * b);
}
int main()
{
        float x = geometric_mean(10, 45);
        printf("%f\n", x);
}
```

   Видоизмените эту программу так, чтобы она считывала 2 вещественных числа и печатала среднее геометрическое.

6. **Is prime?:** Написать функцию `int is_prime(int n)`, которая будет проверять является ли число n простым и возвращать `1` если число `n` простое либо `0` если число `n` не является простым.

7. **Print primes:** Написать функцию `void print_primes(int a, int b)`, которая будет печатать все простые числа из отрезка [a, b]. Используйте функцию `is_prime` из предыдущей задачи! Вызвать эту функцию в функции `main`.

Рекурсия:

Что напечатает следующая программа?

```c
#include <stdio.h>
void hello(int n)
{
        if (n < 0)
                return;
        printf("Hello!\n");
        hello(n - 1);
}
int main()
{
        hello(10);
}
```

Что произойдёт если убрать условие `if (n < 0) return;`?

1. **Sum recursive:** Напишите функцию `int sumrec(int n)`, которая рекурсивно вычисляет сумму первых n натуральных чисел. Вызовите эту функцию из main.

2. **Counter:** Написана рекурсивная функция `void counter(int n)`, которая печатает числа от n до 1.

```c
#include <stdio.h>
void counter(int n) {
        if (n > 0) {
                printf("%d ", n);
                counter(n-1);
        }
}
int main() {
        counter(42);
}
```

   Немного измените эту функцию так, чтобы она печатала числа от 1 до n. (в нормальном порядке).

Указатели:

```c
#include <stdio.h>
int main()
{
        int a = 100;
        int* address = &a;   // адрес переменной a, хранится в переменной address

        // Чтобы по адресу получить саму переменную, нужно поставить * перед
        // *address это то же самое, что и a
        *address = 321;

        printf("%d\n", a);
}
```

1. **Pointer:** Создайте переменную `b` типа `float` и присвойте ей какое-либо значение. Создайте переменную `p` типа указатель на `float` (`p` - это сокращение от pointer - указатель) и присвойте ей значение – адрес переменной `b`. Измените переменную `b`, используя только переменную `p`.
2. **Pointer to pointer:** Создайте переменную `pp` и присвойте ей значение – адрес переменной `p`. Измените переменную `b`, используя только переменную `pp`.

Передача по указателю:

```c
#include <stdio.h>
void add10_wrong(int a) // По значению
{
        a += 10;
}
void add10_right(int* address_of_a) // По указателю
{
        *address_of_a += 10;
}
int main()
{
        int a = 80;
        add10_wrong(a);
        printf("%d\n", a);

        add10_right(&a);
        printf("%d\n", a);
}
```

1. **Modify:** Написать функцию `void doublerp(int* address)`, которая удваивает число, поступающее на вход, используя указатель на эту переменную. Протестируйте эту функцию в функции `main`.
2. **Swap:** Написать функцию `swap`, которая меняет значения 2-х переменных типа int местами. Используйте эту функцию в функции `main()`.
3. **Quadratic:** Написать функцию
   `int solve_quadratic(double a, double b, double c, double* px1, double* px2)`,
   которая будет решать квадратное уравнение. Результат решения функция должна записывать по адресам `px1` и `px2`. Функция должна возвращать целое число - количество корней данного уравнения. Протестировать вашу функцию.