Семинар #1: Основы. Классные задачи.

Работа с терминалом:

Основные команды:

`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
`cp <путь до файла> <путь до копии>`	скопировать файл
`mv <путь до файла> <новый путь>`	переместить или переименовать файл
`rm <путь до файла>`	удалить файл
`rm -r <путь до папки>`	удалить папку
`nano <имя файла>`	nano - это простейший текстовый редактор открывает соответствующий файл, если его нет, то создаёт его Ctrl-O - сохранить; Ctrl-X - выйти при сохранении или выходе у вас попросят ввести имя нового файла нужно ввести и нажать Enter

Сокращение директорий:

`/`	корневая директория
`.`	текущая директория
`..`	директория, которая содержит текущую
`~`	директория пользователя (/home-local/student)

Горячие клавиши:

Tab	автозаполнение
2 раза Tab	показать возможные варианты
стрелка вверх	перейти к предыдущей команде
Ctrl-C	выход из программы, например той, которая зависла
Ctrl-R	поиск по всем предыдущим командам

Задание 1:

1. Откройте терминал и перейдите в папку `/home-local/student`
2. Создайте вашу папку, в которой вы будете работать в течении семестра.
3. Перейдите в эту папку и создайте там файл `test.txt` с помощью `nano`
4. Откройте этот файл с помощью `nano` и напишите в нём что-либо (на ваше усмотрение)
5. Скопируйте этот файл в эту же директорию, но под другим именем.
6. Создайте новую директорию и скопируйте туда файл `test.txt`
7. Переименуйте файл `test.txt`
8. Перейдите в новую созданную вами папку
9. Откройте файл из этой папку в `nano` и измените его.
10. Выйдите из этой папки (перейдите выше: `cd ..`)

11. Зайдите в вашу папку в файловом менеджере (“проводнике”) и проверьте всё.
12. Удалить созданные файлы в терминале с помощью `rm`

Компиляция программы:

Простейшая программа на языке C выглядит следующим образом:

```
#include <stdio.h>
int main()
{
    printf("Hello world!");
}
```

Эта программа печатает на экран строку "Hello world!".

- `#include <stdio.h>` - включаем библиотеку stdio (standard input/output), которая содержит `printf`.
- `int main() { ... }` - основная функция программы, с неё начинается исполнение любой программы.
- `printf("Hello world!");` - печатаем на экран.

Любая программа на языке C должна содержать особую функцию под названием main. По аналогии с обычными математическими функциями, функции в языке C могут принимать и возвращать значения. Принимаемые значения указываются в круглых скобках(в данном случае там ничего нет так как функция ничего не принимает) а тип возвращаемого значения указывается перед функцией (для функции main это всегда тип int, т.е. Integer - т.е. целое число). В фигурных скобках описываются операции, которые совершает функция.

Компилятор gcc:

`gcc <имя файла исходного кода>`	`скомпилировать программу и создать исполняемый файл a.out`
	`файл исходного кода должен иметь расширение .c`
`<путь до исполняемого файла>`	`запустить исполняемый файл`
`./a.out`	`например, запустить файл a.out в текущей директории`
	`. - текущая директория; a.out - имя файла`
`gcc -o <исполняемый файл> <.c файл>`	`скомпилировать программу и создать исполняемый файл`
`gcc -std=c11 <.c файл>`	`использовать стандарт языка C 2011-го года`
`gcc -lm <.c файл>`	`подключить математическую библиотеку (если вы используете math.h)`

Таким образом, чтобы скомпилировать и запустить файл `hello.c` в исполняемый файл `a.out` нужно написать:

```
gcc hello.c
./a.out
```

или так:

```
gcc hello.c && ./a.out
```

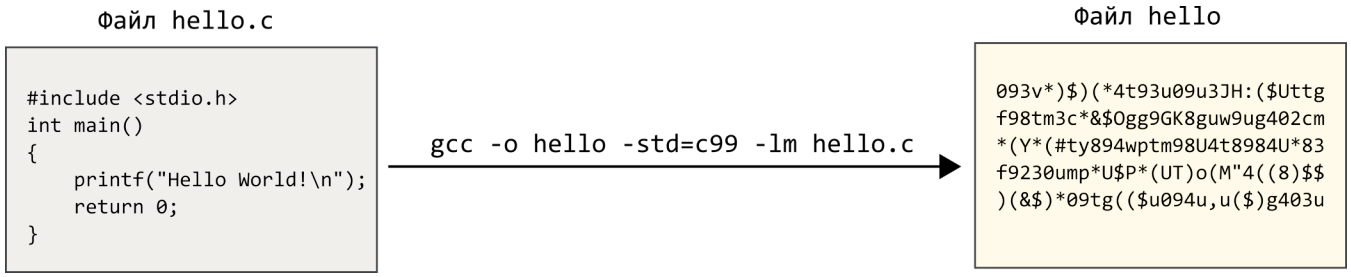
А, чтобы скомпилировать и запустить файл `hello.c` в исполняемый файл `hello` в нужном стандарте и с подключённой математической библиотекой нужно написать:

```
gcc -o hello -std=c11 -lm hello.c
./hello
```

или так:

```
gcc -o hello -std=c11 -lm hello.c  &&  ./hello
```

Помните, что постоянно набирать эту команду не надо, можно просто использовать стрелку вверх.



Задание 2:

1. Скомпилируйте программу `hello.c` и запустите файл `a.out`.
2. Скомпилируйте программу `hello.c` с опцией `-o` и запустите файл `hello`.
3. В строке функции `printf()` можно использовать некоторые специальные символы \n, \t и \b. Добавьте эти символы в строку функции printf и выясните, что они делают.

Переменные `int` (целое число) и их печать - printf:

В переменных `int` можно хранить целые числа от -2^{31} до $2^{31}-1$. (2^{31} примерно равно двум миллиардам)

```
#include <stdio.h>
int main()
{
    int a;
    int b = 5;
    a = 3;
    int res = a * b + (b / a);
    printf("Result = %i\n", res);
}
```

- `int a` - Объявляем, что у нас есть переменная a, которая будет хранить целые числа (integer - целое).
- `int b = 5` - Объявляем, что есть переменная b, которая будет хранить целые числа и присваиваем ей 5.
- `a = 3` - Присваиваем переменной a число 3.
- `res = a * b + (b / a)` - Сохраняем в переменной res результат вычислений.
- `printf("Result = %i \n ", res)` Печатаем, за место спецификатора `%i` (сокращение от `int`) подставится значение переменной.

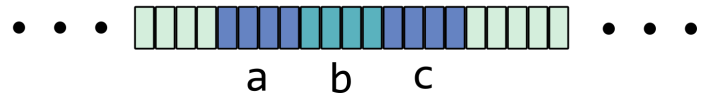
Задание 2:

1. Пусть `a = 451`. Напечатать `a` с помощью следующих спецификаторов:
   - `%i`
   - `%d`
   - `%7d`
   - `%07d`
   - `%x`
   - `%X`
   - `%o`
2. Пусть `a = 436596`, а `b = 7361`. Найти и напечатать остаток деления `a` на `b`. Остаток вычисляется с помощью оператора `a % b`.
3. Пусть `a = 2147483647` (максимальное возможное значение для `int`). Напечатайте значение `a + 1` и `2a`.

Адрес и размер переменной:

- 1 бит - минимальная единица измерения памяти. В 1 бите может хранится либо `0` либо `1`.
- Вся память делится на ячейки, размером в 8 бит = 1 байт.
- Все эти ячейки занумерованы, номер ячейки называется адресом.
- Все переменные содержатся в памяти. Адрес переменной - это адрес первого байта переменной.
- Чтобы найти адрес переменной, нужно перед ней поставить `&`, например, `&a`
- Чтобы найти размер переменной в байтах: `sizeof(a)`
- Например, переменная типа `int` имеет размер 4 байта = 32 бита. Значит в ней может хранится максимум 2^{32} значений.

```
int a, b, c;
```



Задание 3:

1. Создать целочисленные переменные типов `int`, `short` и `char`. Напечатать их адрес и размер.

Считывание переменных - scanf:

Считывание переменных из терминала осуществляется с помощью функции `scanf` из библиотеки `stdio`. В отличии от `printf`, в `scanf` нужно передавать не саму переменную, а её адрес. Это естественно, так как `scanf` должен записать считываемое значение в соответствующие ячейки памяти.
Пример программы, которая считывает переменные a и b и печатает их на экран:

```c
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%i", &a); // <-- не забудьте тут амперсанд &
    scanf("%i", &b); // <-- не забудьте тут амперсанд &
    printf("Multiplication = %i\n", a * b);
}
```

Задание 4:

1. Считать 2 целых числа и напечатать результат целочисленного деления первого на второе. Считать 2 числа с помощью `scanf` можно и одной строкой: `scanf("%i%i", &a, &b)`.
2. Считать 2 целых числа и напечатать остаток деления первого на второе.
3. На вход подаётся прошедшее время в формате `hh:mm`, например, `05:14`. Нужно напечатать, общее количество минут. Создайте 2 переменные `hours` и `minutes` и считать значения этих переменных с помощью `scanf`.
4. Операторы ++. Чему будут равны результаты выполнения следующего кода. Напечатайте `a`, `b` и `c`.

```c
#include <stdio.h>
int main() {
    int a = 753;
    int b = a++;
    int c = ++a;
}
```

Вещественные числа:

Пример программы, которая считывает 2 вещественных числа и вычисляет среднее геометрическое:

```
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b;
    scanf("%f", &a); // <-- не забудьте тут амперсанд & и %f
    scanf("%f", &b); // <-- не забудьте тут амперсанд & и %f
    printf("Geometric average = %f\n", sqrt(a * b));
}
```

В библиотеке `math.h` хранятся математические функции, такие как `sqrt` (корень), `sin`, `cos`, `exp`, `log` (натуральный логарифм), `fabs` (модуль вещ. числа) и другие.

Задание 5:

1. На вход программе подаются 2 положительных вещественных числа - катеты треугольника. Найти гипотенузу.
2. На вход программе подаются 2 положительных вещественных числа `a` и `b`. Вычислить значение выражения `sin(|a - b|) + log(a + b)`.

Логические операторы:

Пример программы, использующие логические операторы:

```
#include <stdio.h>
int main()
{
    int age;
    scanf("%i", &age);
    if (age >= 18 && age < 28)
        printf("Yes\n")
    else
        printf("No\n")
}
```

```
==       равно
!=     не равно
 >       больше
>=  больше и равно
 <       меньше
<=  меньше и равно
```

```
&&   логическое И
||  логическое ИЛИ
 !   логическое НЕ
```

Задание 6:

1. Написать программу, которая принимает на вход число и печатает `Positive`, если число положительное, `Negative`, если число отрицательное и `Zero`, если число равно нулю.
2. Написать программу, которая принимает на вход число и печатает `Yes`, если число принадлежит множеству $(-\infty, -12] \cup (97, +\infty)$.
3. Написать программу, которая принимает на вход число и печатает `Even`, если число четное и `Odd`, если число нечетное. Подсказка: `%`.

Цикл `while`:

Пример: программа, которая вычисляет сумму чисел от `1` до `n`. Тело цикла `while` повторяется до тех пор, пока выполнено условие `n != 0`.

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    int sum = 0;
    while (n != 0)
    {
        sum += n;
        n--;
    }
    printf("%d\n", sum);
}
```

Задание 7:

1. Написать программу, которая будет вычислять произведение всех чисел от `1` до `n`.
2. Написать программу, которая печатает все числа от 1 до n и их квадраты в следующем виде:

   ```
   1 1
   2 4
   3 9
   4 16
   ...
   ```

3. Написать программу, которая будет печатать все числа от `1` до `n`, которые делятся на 7. (Подсказка: в данной задаче можно не использовать `%`).
4. Написать программу, которая будет считывать `n` и печатать `n` звёздочек `*`.
5. Написать программу, которая будет проходить все числа от `1` до `n`. Если число делится на `3`, то программа должна печатать `Fizz`. Если число делится на `5`, то программа должна печатать `Buzz`. Если число делится на `3` и на `5`, то программа должна печатать `FizzBuzz`. Иначе программа должна печатать просто само число.
6. Написать программу, которая будет вычислять квадратный корень вавилонским способом. Считываем вещественное число `a` и нам нужно найти корень из него. Ищем следующим способом:

   $$x_0 = 1$$
   $$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right)$$

   Цикл заканчиваем когда выполнится условие: $|x_n - x_{n-1}| < 10^{-4}$.
7. Переписать предыдущие программы, используя цикл `for`.