Семинар #1: Работа с командной строкой. Основы C.

Часть 1: Работа с командной строкой:

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

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> ./a.out	запустить исполняемый файл 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)

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 на работу с командной строкой:

1. Откройте терминал и узнайте в какой папке вы находитесь. Для этого напечатайте `pwd` и нажмите `Enter`.
2. Перейдите в папку `/home-local/student`. Для этого введите команду:

   ```
   cd /home-local/student
   ```

3. С помощью команды `pwd` проверьте, что вы действительно находитесь в нужной папке.
4. С помощью команды `ls` просмотрите всё содержимое папки `/home-local/student`.
5. Создайте вашу папку, в которой вы будете работать в течении семестра. Используйте команду:

   ```
   mkdir <имя новой папки>
   ```

   За место <имя новой папки> подставьте название вашей папки (треугольные скобки писать не нужно). Желательно, чтобы название содержало только латинские символы без пробелов.
6. С помощью команды `ls` убедитесь, что ваша папка создалась.
7. Перейдите в вашу созданную папку командой `cd <имя папки>`.
8. Перейдите в эту папку с помощью файлового менеджера(проводника) и создайте там файл `test.txt`.
9. Откройте этот файл с помощью любого текстового редактора и напишите в нём что-нибудь.
10. В командной строке введите команду `ls`, чтобы убедится что в папке содержится единственный файл `test.txt`. Если этот файл не отображается с помощью `ls`, то убедитесь, что вы находитесь в нужной папке, используя команду `pwd`.
11. Скопируйте этот файл в эту же директорию командой

    ```
    cp test.txt <имя копии>
    ```

12. Убедитесь, что копия создалась. Для этого используёте `ls` или файловый менеджер.
13. Используя командную строку, создайте новую папку и скопируйте туда файл `test.txt`
14. Перейдите в созданную вами новую папку: `cd <имя папки>`.
15. Используйте `ls`, чтобы просмотреть содержимое этой папки.
16. Выйдите из этой папки (перейдите выше командой: `cd ..`)
17. Переименуйте файл `test.txt`. Для этого используёте команду `mv test.txt <новое имя>`.
18. Удалить созданные файлы и папки в терминале с помощью `rm`. Обратите внимание, что обычные файлы и папки удаляются по-разному.
19. В файловом менеджере создайте файл `test.c` и напишите в нём текст программы *hello world*.
20. В терминале проверьте, что этот файл существует командой `ls`.
21. Скомпилируйте этот файл следующей командой:

    ```
    gcc -o hello test.c
    ```

    После этого в папке создастся новый файл по имени `hello` (это имя мы указали в опции `-o`).
22. Запустите файл `hello` напечатав команду:

    ```
    /home-local/student/<ваша папка>/hello
    ```

    Или просто

    ```
    ./hello
    ```

23. Можно объединить команды компиляции и запуска:

    ```
    gcc -o hello test.c && ./hello
    ```

    Измените программу так, чтобы она печатала `Hello MIPT!`, скомпилируйте и запустите программу.
    Примечание: каждый раз вводить эту команду не нужно, можно просто нажать клавишу вверх.

Часть 2: Основы C

Hello World!

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.

Задание на основы `printf`

1. Скомпилируйте программу, используя `gcc` и запустите.
2. В строке функции `printf()` можно использовать некоторые специальные символы `\n`, `\t` и `\b`. Добавьте эти символы в строку функции `printf` (в произвольное место) и выясните, что они делают.
3. Напишите программу, которая будет выводить на экран:

   ```
   First
       Second
           Third
   ```

   Используйте 1 вызов функции `printf`. Для отступов используйте пробелы или знаки табуляции(`\t`).

Целочисленные переменные `int`:

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);
}
```

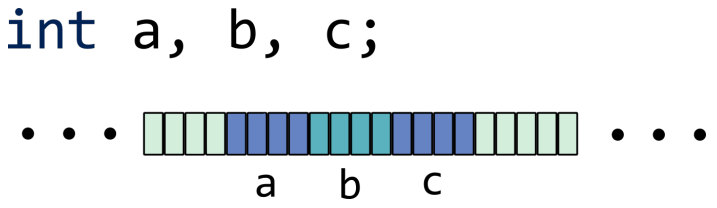
- `int a` - Объявляем, что у нас есть переменная a, которая будет хранить целые числа.
- `int b = 5` - Объявляем, что есть переменная b, которая будет хранить целые числа и присваиваем ей 5.
- `a = 3` - Присваиваем переменной a число 3.
- `res = a * b + (b / a)` - Сохраняем в переменной res результат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

Задание на целочисленные переменные:

1. Создайте переменные `a` и `b` и присвойте им значения `a = 26`, а `b = 7`. Затем:
   - Напечатайте на экран число `a`. Вот так:
     ```
     printf("%i\n", a);
     ```
   - Напечатайте на экран 2 числа `a` и `b`, разделённые пробелом. Вот так:
     ```
     printf("%i %i\n", a, b);
     ```
   - Напечатайте на экран 2 числа `a` и `b` в следующем формате `(26, 7)`.
     ```
     printf("(%i, %i)\n", a, b);
     ```
   - Напечатайте на экран 2 числа `a` и `b` в следующем формате `[26:7]`.
   - Напечатайте на экран 2 числа `a` и `b` в следующем формате `(A = 26 and B = 7)`.
   - Напечатайте на экран сумму чисел `a` и `b`.
   - Напечатайте на экран произведение чисел `a` и `b`.
   - Напечатайте на экран результат целочисленного деления `a` на `b`. Используйте оператор: `a / b`. В результате этой операции должно получиться целое число `3` (так как это операция деления нацело).
   - Напечатайте на экран остаток деления `a` на `b`. Используйте оператор: `a % b`. В результате этой операции должно получиться целое число `5`.
2. 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.



Задание:

1. Напечатать размер переменной типа `int` в байтах. Для этого используйте оператор `sizeof`:
   ```
   printf("%i\n", sizeof(a));
   ```
2. Напечатать адреса переменных типа `int`. Для этого используйте оператор `&`. Адреса памяти обычно хранятся не в переменных типа `int`, а в больших по размеру переменных. Поэтому для их печати нужно использовать не `%i`, а `%lli` или `%p`.:
   ```
   printf("%lli %lli\n", &a, &b);
   ```
   Убедитесь, что переменные `a` и `b` лежат в памяти вплотную друг к другу.

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 на экран их произведение:

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

Примечание: При задании формата в `scanf` не нужно ставить пробелы и символы переноса строки. Т.е. не нужно писать так:

```
scanf("%i\n", &a);  // будет ожидать ввода ещё одного числа
```

Задание на считывание:

1. Написать программу, которая считывает целое число и печатает на экран квадрат этого числа.
2. 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); // <-- не забывайте тут амперсанд &
   ```

3. Считать 2 целых числа и напечатать остаток деления первого на второе.
4. Считать целое число и напечатать его последнюю цифру. Используйте оператор остатка.
5. На вход подаётся прошедшее время в формате `hh:mm`, например, `05:14`. Нужно напечатать, общее количество минут (`314`). Создайте 2 переменные `hours` и `minutes` и считать значения этих переменных с помощью `scanf`. Вот так:

   ```
   scanf("%i:%i", &hours, &minutes); // <-- не забывайте тут амперсанд &
   ```

Операторы инкремента:

Для удобства в языке C введены следующие операторы:

```
+=    -=    *=    /=     ++   --    и другие
```

Например оператор присваивания сложения `+=` увеличивает левый аргумент на величину правого.
Оператор `++` увеличивает значение аргумента на 1.

```
#include <stdio.h>
int main() {
    int a = 100;
    a = a + 5;  // увеличиваем a на 5
    a += 5;     // увеличиваем a на 5 ( то же самое )

    a++;        // увеличиваем a на 1
    ++a;        // увеличиваем a на 1
}
```

Чему будет равно значение переменной a после выполнение данного кода?

Логические операторы:

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

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

Пример программы, которая считывает возраст человека и печатает `Yes`, если возраст больше или равен 18:

```c
#include <stdio.h>
int main() {
    int age;
    scanf("%i", &age);
    if (age >= 18) {
        printf("Yes\n");
    }
}
```

Пример программы, которая считывает число `n` и печатает `Yes`, если число двузначно и `No` иначе:

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%i", &n);
    if (n >= 10 && n < 100) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }
}
```

Пример программы, которая принимает на вход число и печатает `Positive`, если число положительное, `Negative`, если число отрицательное и `Zero`, если число равно нулю:

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%i", &n);
    if (n > 0) {
        printf("Positive\n");
    }
    else if (n == 0){
        printf("Zero\n");
    }
    else {
        printf("Negative\n");
    }
}
```

Задание на логические операторы:

1. Написать программу, которая считывает число и печатает `Yes`, если число равно `42` и `No` иначе. Обратите внимание, что для сравнения чисел нужно использовать оператор `==` ("два равна").
2. Написать программу, которая принимает на вход число и печатает `Yes`, если число принадлежит множеству $(-\infty, -12] \cup (97, +\infty)$ и `No` иначе.
3. Написать программу, которая принимает на вход число и печатает `Even`, если число четное и `Odd`, если число нечетное. Подсказка: число чётное, если остаток от делениея на 2 равен 0.
4. Написать программу, которая принимает на вход два числа и печатает `First`, если первое число больше второго, `Second`, если второе больше первого и `Equal`, если числа равны.
5. Написать программу, которая принимает на вход два числа и печатает большее из этих двух чисел.
6. Написать программу, которая принимает на вход три числа и печатает `Unique`, если все числа различны и `Not Unique`, если хотя бы 2 числа равны.

Цикл `while`:

Пример:

Пример программы, которая печатает числа от 0 до 9, разделённые пробелом:

```c
#include <stdio.h>
int main() {
    int i = 0;
    while (i < 10) {
        printf("%i ", i);
        i += 1;
    }
}
```

Задачи:

Измените программу выше так чтобы:

1. Программа печатала числа от 0 до 20
2. Программа печатала числа от 5 до 15
3. Программа печатала числа от 5 до 15, разделённые не пробелом, а запятой:

   ```
   5,6,7,8,9,10,11,12,13,14,15,
   ```

4. Программа печатала числа, разделённые символом `+`
5. Программа печатала числа, разделённые символом переноса строки `\n`. (каждое число в новой строке)
6. Программа печатала квадраты этих чисел
7. Программа печатала только чётные числа от 0 до 100
8. Программа печатала только числа, делящиеся на 7 (от 0 до 100)
9. Программа должна считывать число `n` и печатать все числа от `0` до `n` через пробел.
10. Программа должна считывать число `n` и печатать все квадраты чисел от `0` до `n` через перенос строки.
11. Программа должна считывать число `n` и для каждого числа из диапазона от `0` до `n` программа должна печатать `Foo`, если число делится на `3` и `Bar`, если число делится на `5`.
12. Программа должна считывать числа `a`, `b`, `c`, и печатать все числа, делящиеся на `c` на отрезке от `a` до `b` через пробел.

Пример:

Пример программы, которая вычисляет сумму чисел от `1` до `n`.

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int sum = 0;
    int i = 1;
    while (i <= n) {
        sum += i;
        i += 1;
    }
    printf("%d\n", sum);
}
```

Задачи:

Измените программу выше так чтобы:

1. Программа находила произведение всех чисел от `1` до `n`.
2. Программа находила сумму всех нечётных чисел от `1` до `n`.
3. Программа находила сумму квадратов всех чисел от `1` до `n`.
4. Программа вычисляла следующее выражение:

$$1^2 - 2^2 + 3^2 - 4^2 + 5^2 - 6^2 + ... + (-1)^{n+1} n^2$$

Пример:

Пример программы, которая считывает числа последовательно и печатает квадраты этих чисел. Если попадётся отрицательное число, то программа закончится.

```
#include <stdio.h>
int main() {
    while (1) {
        int a;
        scanf("%i", &a);
        if (a < 0) {
                break;
        }
        printf("%d\n", a * a);
    }
}
```

Задачи:

Измените программу выше так чтобы:

1. Программа выводила кубы чисел, пока не встретит отрицательное число.
2. Программа печатала `Odd`, если число нечётное и `Even`, если число чётное, пока не встретит отрицательное число
3. Для каждого введённого числа `a` программа должна печатать последовательность чисел от `1` до `a` через пробел. Для этого вам нужно использовать ещё один цикл `while` внутри цикла `while`.
4. Для каждого введённого числа `a` программа должна печатать сумму последовательности чисел от `1` до `a`.

Цикл `for`:

Пример:

Пример программы с циклом `for`, которая печатает числа от `0` до `9`.

```
#include <stdio.h>
int main() {
    for (int i = 0; i < 10; ++i) {
        printf("%i ", i);
    }
}
```

Для компиляции этой программы возможно потребуется указать опцию компилятора `-std=c11`. Вот так:

```
gcc -o prog -std=c11 <файл исходного кода>
```

Задачи:

Измените программу выше так чтобы:

1. Программа печатала числа от 0 до 20
2. Программа печатала числа от 5 до 15
3. Программа печатала числа от 5 до 15, разделённые не пробелом, а запятой:

   ```
   5,6,7,8,9,10,11,12,13,14,15,
   ```

4. Программа печатала числа, разделённые символом переноса строки `\n`. (каждое число в новой строке)
5. Программа печатала только числа, делящиеся на `7` (от `0` до `100`)
6. Программа считывала `n` и печатала все числа от `1` до `n` и их квадраты в следующем виде:

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

7. Программа считывала `n` и печатала `n` символов звёздочка `*`. Например, если ввести `7`, то программа должна напечатать `*******`.

Пример:

Что напечатает данная программа?

```
#include <stdio.h>
int main() {
    int n;
    scanf("%i", &n);
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("(%i %i) ", i, j);
        }
        printf("\n");
    }
}
```

Задачи:

Измените программу выше так чтобы:

1. Программа печатала таблицу умножения.
2. Программа считывала `n` и печатала квадрат из звёздочек размером `nxn`.