Chapter 1: Tutorial Introduction

Getting Started:

- The only way to learn a new programming language is by writing programs in that language.
- We write the hello world program and compile it with gcc.
- C programs consist of functions and variables.
    - Functions contain statements that specify the operations to be done.
        - main is a special function name that is reserved for beginning program execution.
        - Every program must have a main function.
            - main will usually call other functions to help do its job.
            - Some functions will be programmer defined some will be from libraries.
    - Variables store the values to be used in the computations.
- A sequence of characters in double quotes is a character string or string constant. More modern terminology is string literal.
- Escape sequences are used to represent hard to type or invisible characters
    - \n, \t etc.

Exercises:

- Run the hello world program and experiment with leaving portions out. See what the error messages are.
    - If we leave out the quotes around hello world we get undeclared identifier errors for Hello and World. We also get that there is a stray .

    - If we leave out the printf and leave the parenthesis portion we don't get a compiler error and when we run it nothing happens.
    - If we leave out the directive then we get incompatible implicit declaration of built in function printf.
- Experiment to see what happens when printf's argument string contains \c where c is some character not listed previously.
    - We get a warning from gcc stating unknown control sequence.

Variables and Arithmetic Expressions:

- C provides basic data types for arithmetic.
    - int, float, char, short, long, double.
    - Sizes of these data types are machine dependent.
- Assignment to a variable has the following form:
    - int x = 0;
- Integer division truncates any fractional part of the result.

The For Statement:

- The for statement is the standard for statement you already know.

Symbolic Constants:

- It's bad practice to bury magic numbers in a program.
    - They convey little information to anyone reading the program later.
    - They are difficult to change in a systematic way.
- We may define constants with the following syntax:
    - `#define name replacement text`
    - Any occurrence of *name* in the program will be replaced with the *replacement text*.

Character Input and Output:

- Text input or output, regardless of where it comes from or its destination, is treated as streams of characters.
- A *text stream* is a sequence of characters divided into lines.
    - Each line consists of zero or more characters followed by a newline char.
    - The library makes sure that each input or output stream conforms to this mode.
- The standard C library provides several functions for reading or writing one character at a time.
    - *getchar()* reads the next input character from a text stream and returns its value.
        - `c = getchar()` assigns the value fo the next character input to c.
    - *putchar()* is the reverse, it prints a character when called.
        - `printchar(c)` prints the integer c as a character.

- We could use the following snippet to copy one char stream to another:

```
main() {
  int c;

  c = getchar();
  while(c != EOF){
      putchar(c);
      c = getchar();
  }
}
```

- This code reads a character and assigns it to c. Then it checks if c is the EOF character. If not it prints the character c and gets the next character. repeating until the while loop condition is not met.

- A more concise version of the same snippet:

  ```
  main() {
    int c;
    while((c = getchar()) != EOF)
        putchar(c);
  }
  ```

- This version combines the assignment with the while condition. It is perfectly reasonable to do this in C.
  - The benefit of this is shrinking the code and making only one call to the *getchar()* function.
  - != has higher precedence that =, so we need the added parenthesis.

Exercises:

- I assigned an integer variable EOF and printed it. The value was -1.

Character Counting:

- We have a program to count the characters until the EOF marker is reached.
- Introducing the *increment by one* operator which is ++.
  - It matters if you use ++v or v++.

Line Counting:

- We can also write a program to count lines of input.
  - To do this we just have to count the number of new lines escapes.
- We us the state variable constant to determine if we are in the middle of a word or not.
- When we get to larger programs we are going to want to write more

comments explaining whats going on.

Arrays

- An array subscript always starts at 0.
  - We have to adjust the count in for loops to reflect that
  - We can use an expression to identify the subscript.

Functions

- In C a function provides a convenient way to encapsulate some computations in a way that can be used without worrying about implementation.
  - if a function is properly designed it is possible to ignore how something is done and just go with what is being done.
- We have used some functions already:
  - printf()
  - getchar()
  - putchar()
- We can write a function that will raise a number to a power.

Arguments - Call By Value

- In C all function arguments are passed "by value".
  - This means that the called function is given the value of its arguments in temporary variables rather than the originals.
    - In fact we push the values onto the stack before calling the function.
- Side effects of this property:
  - A function can not directly alter he value of a variable in the calling function.

        - It may only alter its' copy of that value locally.
    - This property leads to fewer extraneous variables because the parameters can be treated as initialized local variables.
    - Example:
        - We could use a passed variable as a counter without needing to initialize a new one.
- When we need to modify a variable in a calling routine we must provide the address of the variable to be set.
    - We will use a pointer for this operation.
- When using an array the address of the first entry is is passed to the function.
    - There is no copy of the array elements.

Character Arrays

- The most common type of array in C is the character array.

External Variables and Scope

- The variables in main are private or local to main.
    - Because they are declared in main other functions can have no direct access to them.
    - This is the case for all functions.
- Each variable in a function comes into existence only when the function is called and disappears when the function is exited.
    - In K and R they refer to local variables as automatic variables.
- It is possible to define variables that are external to all functions.
    - These variables are global variables.
    - An external variable must be defined exactly once outside of any function.
        - storage is set aside for it.

- The variable must be declared in each function that wants to access it.
    - States the type of the variable.
- In certain circumstances the extern declaration can be omitted.
- If the definition of an external variable occurs in the source file before its use in a particular function then there is no need for an extern declaration in the function.
    - The common practice is to place definitions of all external variables at the beginning of the source file and omit the extern declarations.