CSI 333 – Lecture 3
Introduction to C: Part II
Another Program Example

Program Example 4: See Handout 3.1.

Observations on Example 4:

- Function `getchar` (from `stdio.h`) is used to obtain the next character from the input. (There are no parameters to `getchar`.)
- The function `getchar` returns a value of type `int`. This is needed to handle EOF.
- Reading a character and checking for EOF are done in the same statement. (This is common in C; more discussion below.)
- When the character is a digit, the expression `c - ’0’` gives the correct digit value (as an `int`).
- Note the use of `break` statements within the `switch` statement.
Any assignment statement in C is an expression; the value of the expression is the value of the right side of the statement. So, an assignment and a test can be combined.

Example:

```c
while ((c = getchar()) != EOF) {
    //
    //
    //
}
```

Important Note

The parentheses around “c = getchar()” ensure that the assignment is done before the test for EOF.
Watch out!!

The following does not cause a syntax error, but it is an infinite loop.

```c
while (x = 1) {
    .
    .
}
```

The programmer wrote “x = 1” instead of the correct comparison “x == 1”.

**Note:** Using the “-Wall” option to gcc will produce a warning for the above type of error.
Remark: Some assignment statements can be shortened. For example

\[ x = x + z; \]

can also be written as

\[ x += z; \]

Note that

\[ x *= y + 1; \]

is equivalent to

\[ x = x * (y + 1); \]
Remark: As shown in Handout 3.1, multiple assignments can be done using one statement. For example,

\[ p = q = 0x27; \]

sets both \( p \) and \( q \) to the value \( 27_{16} \).

Arrays in C: Similar to Java. (See the program in Handout 3.1.)

Program Example 5: See Handout 3.2.
Observations on Example 5:

- The program uses functions `srand` and `rand` from the C standard library (header: `<stdlib.h>`).
  - Function `srand` accepts an `int` parameter which serves as the “seed” for the random number generator.
  - Each call to function `rand` (which has no parameters) returns a random non-negative integer.
  - The integer returned by `rand` is converted into an integer in the range 1 to 6 by the program.
- The frequency array stores the number of occurrences of each possible value over 6000 trials.
A C program consists of a collection of (non-nested) functions.

Execution begins with the function named `main`.

Each function has a **return** type; the special type `void` indicates that the function does not return a value.

Functions can have parameters; the keyword `void` is used to indicate that a function does not use any parameters.

Parameters are passed by value ("call-by-value") in general; pointers must be passed in order to achieve "call-by-reference".

**Exception:** Array parameters are always passed by reference.
Program Example 6:  See Handout 3.3.

Observations on Example 6:

- The program has four functions: main, print_array, increment and stat_values.

- Note the use of function prototypes.

- For array parameters in function headers, sizes are not specified.

- To the function stat_values, parameter n is passed by value; parameters *min and *max are pointers used to achieve call-by-reference.

- Note how the reference parameters are used within the body of function stat_values.
Note the actual parameters &min and &max used in the call to stat_values in main. (Recall that the function scanf uses this form of parameters.)

The function increment changes the values stored in the array. (This shows that arrays are always passed by reference.)

Note the typecast used to compute the average in stat_values.

An important rule in C

If you want a function to change the value of a parameter (which is not an array), you must to pass the address of the parameter to the function.
Remarks:

- In Java and C++, local variables can be defined anywhere (e.g. indices of for loops).

- In C, all local variables must be declared before the first executable statement of a function. Within any function, declarations are not permitted after the first executable statement.

Example: The following code is not valid in C.

```c
for (int i = 0; i < m; i++) {
    
    
}
```
Structure of a C Program:

1. `#include` statements for standard libraries (e.g. `stdio.h`, `stdlib.h`).
2. Definitions of symbolic constants (using `#define`).
3. Function prototypes.
4. Code for each of the functions.
Facts to Remember:

- Each assignment statement is an expression; so an assignment statement can be combined with a test.

- For any function, all parameters (except arrays) are passed by value.

- For a function to change the value of a parameter (i.e., to pass a parameter by reference), you must pass the address of that parameter.

- In any function, all declarations must appear before the first executable statement.