CSI 333 – Lecture 2
Introduction to C: Part I
Remark: Skim Chapters 1 through 6 of Deitel & Deitel. You will notice the following:

- C is (more or less) a subset of Java. (So, you are not really learning a “new” language.)
- The differences between C and Java are not difficult to master.

Why learn C now? C is usually preferred over Java in implementing systems software because:

- C compilers generating good quality machine code are readily available.
- Size of executable created from a C source is typically smaller.
Parts common to C and Java:

- Primitive data types (int, char, float and double).
- Arrays.
- Arithmetic and logical operators and expressions.
- Pre and post increment/decrement operators.
- Assignment statement.
- Control statements (if, switch, etc.)
- Loops (for, while, etc.).
- Functions and return statement.

Note: The above list is not exhaustive.

Main Difference

C is not an object-oriented language.
A Simple C Program (Example 1)

```c
#include <stdio.h>
int main(void) {
    printf("Hello World!\n");
    return 0;
} /* End of main. */
```

**Notes about the program:**

- Every C program must have a function named `main`. In the above example,
  - The `main` function returns a value of type `int`.
  - The function does not have any parameters.
Notes on Example 1 (continued)

- `<stdio.h>`: Header file needed to use I/O functions (such as `printf`).
- Each C program can use three standard I/O devices: `stdin`, `stdout` and `stderr`.
- The function `printf` writes to `stdout`.
- The symbol `'\n'` denotes the newline character.
- The `return` statement is followed by a value. (In general, the value may be an expression.)
- Comments are enclosed within `"/\*"` and `"*/"`. (Most C compilers will also accept `"//"`.)
Compiling and Executing C programs

1. Suppose file `hello.c` contains the program of Example 1.
2. Compile the program using the following Unix command:
   ```bash
   gcc -c hello.c
   ```
3. If the compilation produces syntax errors, edit the file to correct the errors and compile again. (Repeat steps 2 and 3 until all syntax errors have been fixed.)
4. Create the executable version of the program using the following Unix command:
   ```bash
   gcc hello.c
   ```
   The executable version is created in a file called `a.out`.
5. Execute the program using the following Unix command:
   ```bash
   a.out
   ```
Compilation Tip

To compile a C program, it is better to use the following Unix command:

```
gcc -c -g -Wall hello.c
```

This identifies syntax errors and also produces warnings that are useful in identifying potential problems. The “-g” part is useful for debugging (to be seen later).

- The command

```
gcc hello.c -o prog.out
```

produces the executable version in prog.out instead of a.out.

Program Example 2: See Handout 2.1.
Function `scanf` (from `<stdio.h>`) is used to read values from `stdin`.

The format "%d" indicates that the value to be read is a decimal integer.

**Very Important:** The `scanf` function needs addresses of variables (i.e., pointers) to read values. (The addresses of variables `x` and `y` are denoted by `&x` and `&y` respectively.)

---

**Bus Error**

Leaving out the ‘&’ in a call to `scanf` often causes a “bus error” when the program is executed.

- The function `printf` also uses formats.
- In general, values to be printed may be expressions.
The primitive data types provided in C are:

- char, int, float, double.

The int data type can also use the modifiers short, long or unsigned.

Example:

```c
long int val;
unsigned long p;
short x;
```

The number of bytes needed for the above data types is machine dependent.

C does not have the byte or boolean types of Java.

In C, the value 0 represents false and any nonzero value represents true.
Example:

```c
int x, y;
.
.
if (x-y) { /* Here, the value of x-y is nonzero. */
.
.
}
else { /* Here, the value of x-y is zero. */
.
.
}
```
Comments using "/*" and "*/" can't be nested. For example, the following causes a syntax error. (Why?)

```c
/* -- A line of comment
   /* -- Invalid nesting -- */
*/
```

**Caution**

Commenting out sections of C code must be done with care. The safest way to comment out a section of C code is:

```c
#if 0
    --- code to be commented out --
#endif
```

**Note:** The "#if" and "#endif" are part of the macro facility supported by C. (More on macros later.)
Symbolic constants can be introduced through the `#define` statement. For example,

```c
#define SIZE 100
#define MAX_LEN 20

int values[SIZE];
char str[MAX_LEN];
```

Note: The `#define` statement is also part of the macro facility supported by C.

Program Example 3: See Handout 2.2.
Note the definition of symbolic constants using `#define`.

Formats `"%d"` and `"%f"` are used in the call to function `printf`:

- Format `"%4d"` means that the value to be printed is a decimal integer and the printing width must be at least 4 characters.

- Format `"%6.1f"` means that the value to be printed is of type `float` and the printing width must be at least 6 characters with one character after the decimal point.
Notes on printf

- Prints output to stdout.
- The first parameter to printf is a string specifying format for printing values. (e.g. \%d for int, \%f for float, etc.)
- Values to be printed appear after the format string separated by commas.
- Value to be printed may be an expression. (The expression will be evaluated and its value will be printed.)
Notes on scanf

- Reads input from stdin.
- The first parameter to scanf is a string specifying format for reading values.
- Addresses of (i.e., pointers to) variables whose values are to be read follow the format string.

Recall: Leaving out the ‘&’ in a call to scanf often causes “bus error” at run time.

Remark: For a complete list of format specifications (to be used with printf or scanf), see Chapter 9 of Deitel & Deitel.
Suggested Exercises

- Study the handouts discussed in this lecture. Make sure that you understand how to use the standard I/O functions `scanf` and `printf`.

- Go through the program examples in Chapters 1 through 5 of Deitel & Deitel. (This will allow you to become familiar with the syntax of C. You will also notice the similarities between C and Java.)

- Try writing your favorite short pieces of code (e.g. sequential search, simple sorting algorithms) in C.

- Compile and execute some C programs in Unix.