Introduction to Computing

MCS1101B

Lecture 7

Recap

- Pointers
  - Value
  - Name
  - Address
  - Size and type of pointers
- Array and pointers
- Function and pointers

- Functions calling functions
- Recursion
- Passing Array to functions
  - Problems with sending size
- Character Array
- Strings

Character Arrays and Strings

- Character arrays are very useful in storing data
  - Even though they are basically integers underlying, but the range of the values are limited
  - This allows to have some additional functionalities (for convenience, of course)
  - Strings are declared and defined the same way as any other array types
  - Since the values are in range of 0-127 (sometimes more, but still, limited), we have the convenience make some of the characters for special use such as:
    - newline(\n)
    - backspace (\b), etc.
  - In the case of character arrays we use a special character called the null character
    - Represented as '\0' (backslash-zero)
    - Ascii value of this character is 0
    - It prints nothing on the computer screen

Character array and strings

- Character variable
  - char ch1, ch2 = ‘a’;
- Character array
  - char ca1[10];
  - char ca2[3] = {‘S’,‘D’,‘B’};
  - char ca3[5] = {‘S’,‘D’,‘B’};

A string is a character array for which the last valid character is the null character.

- char ca4[10] = {‘S’,‘o’,‘u’,‘m’,‘a’,‘d’,‘i’,‘p’,‘\0’};
- char ca5[10] = “Soumadip”;
  - *Both the above statements are equivalent*
  - This type of initialization makes sure that the null character is automatically appended at the end

You *can't do the following after declaration* though

ca1 = "word1"; // not allowed – why?
ca4 = "word2"; // not allowed – what is the type of ca1 or ca4?

-- More on what can and can't be done, later

String is basically short for “a string of characters”

- A single character in C is written within single quotes e.g. ‘a’, ‘3’, ‘Z’, ‘%’, etc.
- A string is written in C within double quotes, e.g., “a_string”, “with spaces”, “and with $”, etc.

Strings and scanf

- scanf also provides a shortcut for strings format **%s**
  - scanf (“%s”, ch_arr); ⇒ this allows you to read a string from user without spaces
  - scanf ("%[^ ^\n]%*c", ch_arr);
    - This is equivalent to **%s**; reads the characters until space ( ) or the newline character (\n) is encountered
  - scanf ("%[^\n]%*c", ch_arr);
    - reads a string with spaces until a newline(\n); so, it can read strings with spaces

Note: *All the method discussed here will add a ‘\0’ to the end of the scanned characters - making it a string*

String Operations

- Normal assignments do not work on strings (or any arrays for that matter)
- You can define different operation on strings by writing your own functions
  - Compare two strings for equality
  - Copy one string to another
  - Concatenate two strings
  - Check if a input string is integer or float

- Alternatively, you can choose to #include a new header file called string.h and use built-in functions for such operations
  - strlen
    - int strlen(const char *str)
  - strcmp
    - int strncmp(const char *str1, const char *str2, int n)
  - strstr
    - char* strstr(const char *haystack, const char *needle)
  - strcat
    - char* strcat(char *dest, const char *src)
  - Check the link for more.

Preprocessors

- Preprocessor is not a part of the compiler
- It is a step in the compilation process
- a C Preprocessor is just a text substitution tool
- It instructs the compiler to do required pre-processing before the actual compilation
- They are also known as macro

Examples:

- #inlcude <string.h>
- #define SIZE 10
- #define SQUARE(x) ((x)*(x))
- #ifdef <macro>.. #endif
- etc.