Lecture 13

Strings

CSE115: Computing Concepts

Fundamentals of Characters and Strings

- Characters in C consist of any printable or nonprintable character in the computer's character set including lowercase letters, uppercase letters, decimal digits, special characters and escape sequences.
- A character is usually stored in the computer as an 8-bits (1 byte) integer.
- The integer value stored for a character depends on the character set used by the computer on which the program is running.

- There are two commonly used character sets:
  - ASCII (American Standard Code for Information Interchange)
  - EBCDIC (Extended Binary Coded Decimal Interchange Code)

Difference Between an Integer Digit and a Character Digit

- **char num = 1** and **char num = ‘1’** are not the same.
- **char num = 1** is represented in the computer as 00000001.
- **char num = ‘1’** on the other hand is number 49 according to the ASCII character set. Therefore, it is represented in the computer as 00110001.

Example: ASCII Character

```
#include <stdio.h>

void main(void)
{
  char my_A = 'A';
  char my_Z = 'Z';
  char my_a = 'a';
  char my_z = 'z';

  printf("\nASCII value for A is %d", my_A);
  printf("\nASCII value for Z is %d",my_Z);
  printf("\nASCII value for a is %d", my_a);
  printf("\nASCII value for z is %d",my_z);

  printf("\n");
  printf("\n65 in ASCII represents %c",65);
  printf("\n90 in ASCII represents %c",90);
  printf("\n97 in ASCII represents %c",97);
  printf("\n122 in ASCII represents %c",122);
}
```

Sample Output

ASCII value for A is 65

ASCII value for Z is 90

ASCII value for a is 97

ASCII value for z is 122

65 in ASCII represents A

90 in ASCII represents Z

97 in ASCII represents a

122 in ASCII represents z

Example

```
#include <stdio.h>

int main()
{
  char ch;
  printf("enter a character: ");
  scanf("%c", &ch);
  if (ch >= 'A' && ch <= 'Z')
        printf("\ncapital letter\n");
  return 0;
}
```

equivalent to

```
#include <stdio.h>

int main()
{
  char ch;
  printf("enter a character: ");
  scanf("%c", &ch);
  if (ch >= 65 && ch <= 90)
        printf("\ncapital letter\n");
  return 0;
}
```

String Declaration and Initialization

- A string in C is an **array of characters** ending with the null character (`'\0'`). It is written inside a double quotation mark (" ")
- A string can be assigned (in a declaration) to a char array:
  - **`char color[6] = "green";`**

`color`

0	1	2	3	4	5
'g'	'r'	'e'	'e'	'n'	'\0'

String Declaration and Initialization

- A string can also be defined by specifying the individual characters:
  - **`char color[ ] = {'g', 'r', 'e', 'e', 'n', '\0'};`**

`color`

0	1	2	3	4	5
'g'	'r'	'e'	'e'	'n'	'\0'

String Declaration and Initialization

- Notice that even though there are only five characters in the word ‘green’, six characters are stored in the computer. The last character, the character ‘\0’, is the NULL character which indicates the end of the string.
- Therefore, if an array of characters is to be used to store a string, the array must be large enough to store the string and its terminating NULL character.

String Declaration and Initialization

- If we happen to declare a string like this:

  char my_drink[3] = “tea”;

- We will get the following syntax error:

  error C2117: 'tea' : array bounds overflow

- Instead, we need to at least declare the array with (the size of the string + 1) to accommodate the null terminating character ‘\0’.

  char my_drink[4] = “tea”;

String Declaration and Initialization

- We can initialize string variables at compile time such as;
  - **`char name[10] = "Arris";`**
  - This initialization creates the following spaces in storage :

name

0	1	2	3	4	5	6	7	8	9
'A'	'r'	'r'	'i'	's'	'\0'	'\0'	'\0'	'\0'	'\0'

Example: String and '\0'

```
#include <stdio.h>
void main()
{
  char sentence[] = "I love Bangladesh";
  int i, count = 0;

  for (i = 0; sentence[i] != '\0'; i++)
       count++;
  printf("%s has %d characters including the whitespace",
       sentence, count);
}
```

Example: String and '\0'

```
#include <stdio.h>
void main()
{
  char sentence[] = "I love Bangladesh";
  int i, count = 0;

  for (i = 0; sentence[i] != '\0'; i++)
       count++;
  printf("%s has %d characters including the whitespace",
       sentence, count);
}
```

Sample output:

I love Bangladesh has 15 characters including the whitespace

String Input/Output Functions

- Standard Functions Input
  - scanf( )
  - gets( )
- Standard Functions Output
  - printf( )
  - puts( )
- Use scanf function together with the format specifier %s for interactive input string. (no whitespace character)
- If the string to be read as an input has embedded whitespace characters, use standard *gets* function.

Example: **gets**, **puts**, **scanf** and **printf**

```
#include <stdio.h>

int main()
{
  char string1[50];
  char string2[50];

  printf("Enter a string less than 50 characters with
       spaces: \n");
  gets(string1);

  printf("\nYou have entered: ");
  puts(string1);

  printf("\nTry entering a string less than 50
       characters, with spaces: \n");
  scanf("%s", string2);

  printf("\nYou have entered: %s\n", string2);
  return 0;
}
```

Example: **`gets`**, **`puts`**, **`scanf`** and **`printf`**

Sample output

```
Enter a string less than 50 characters with spaces:
hello world

You have entered: hello world

Try entering a string less than 50 characters, with spaces:
hello world

You have entered: hello
```

String Conversion Functions

- These functions convert strings of digits to integer and floating-point values.
- To use these functions, the general utilities library **`<stdlib.h>`**, needs to be included.
- Example:
  - **atoi**: string to int
  - **atof**: string to double

Example

```
/*1.  Converting a String Into an int Using
 atoi. */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char str1[ ] = "124z3yu87";
    char str2[ ] = "-3.4";
    char str3[ ] = "e24";
    int i1 = atoi(str1), i2 = atoi(str2),
 i3 = atoi(str3);
    printf("i1: %d\n", i1);
    printf("i2: %d\n", i2);
    printf("i3: %d\n", i3);
    return 0;
}
```

Example

```
/*1.  Converting a String Into an int Using
 atoi. */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char str1[ ] = "124z3yu87";
    char str2[ ] = "-3.4";
    char str3[ ] = "e24";
    int i1 = atoi(str1), i2 = atoi(str2),
 i3 = atoi(str3);
    printf("i1: %d\n", i1);
    printf("i2: %d\n", i2);
    printf("i3: %d\n", i3);
    return 0;
}
```

Output:

i1: 124
i2: -3
i3: 0