SDN150S – SOFTWARE DESIGN 1

Strings & Formatted Input/Output

DEPARTMENT OF ELECTRICAL ELECTRONIC AND COMPUTER ENGINEERING
CAPE PENINSULA UNIVERSITY OF TECHNOLOGY

Cape Peninsula University of Technology

STRINGS (1 OF 7)

- Strings are actually one-dimensional array of characters terminated by a null character '\0'.
- Thus a null-terminated string contains the characters that comprise the string followed by a null.
- The following declaration and initialization create a string consisting of the word "Hello".
- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello.".

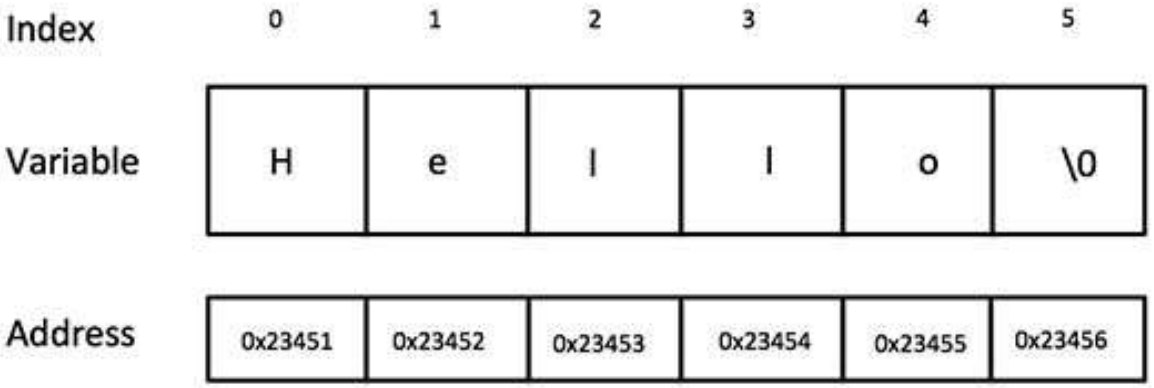
```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

STRINGS (2 OF 7)

- If you follow the rule of array initialization, then you can write the above statement as follows:

```
char greeting[] = "Hello";
```

- Following is the memory presentation of the above defined string in C/C++:



STRINGS (3 OF 7)

- Actually, you do not place the null character at the end of a string constant.
- The C compiler automatically places the '\0' at the end of the string when it initializes the array.
- Let us try to print the above mentioned string:

```
#include <stdio.h>
int main ()
{
    char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
    printf("Greeting message: %s\n", greeting );
    return 0;
}
```

Output

```
Greeting message: Hello
```

STRINGS (4 OF 7)

- C supports a wide range of functions that manipulate null-terminated strings. Such as:

No.	Description
1	**strcpy(s1, s2); -** Copies string s2 into string s1.
2	**strcat(s1, s2); -** Concatenates string s2 onto the end of string s1.
3	**strlen(s1); -** Returns the length of string s1.
4	**strcmp(s1, s2); -** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch); -** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2); -** Returns a pointer to the first occurrence of string s2 in string s1.

STRINGS (5 OF 7)

- The following example uses some of the above-mentioned functions:

```c
#include <stdio.h>
#include <string.h>
int main ()
{
    char str1[12] = "Hello";
    char str2[12] = "World";
    char str3[12];
    int len ;
    /* copy str1 into str3 */
    strcpy(str3, str1);
    printf("strcpy( str3, str1) : %s\n", str3 );
    /* concatenates str1 and str2 */
    strcat( str1, str2);
    printf("strcat( str1, str2): %s\n", str1 );
    /* total lenghth of str1 after concatenation */
    len = strlen(str1);
    printf("strlen(str1) : %d\n", len );
    return 0;
}
```

OUTPUT

```
strcpy( str3, str1) : Hello
strcat( str1, str2): HelloWorld
strlen(str1) : 10
```

FORMATTED INPUT/OUTPUT (1 OF 6)

- When we say ***INPUT***, it means to feed some data into a program. An input can be given in the form of a file or from the command line.
- C programming provides a set of built-in functions to read the given input and feed it to the program as per requirement.
- When we say ***OUTPUT***, it means to display some data on screen, printer, or in any file.
- C programming provides a set of built-in functions to output the data on the computer screen as well as to save it in text or binary files.
- C programming treats all the devices as files. So devices such as the display are addressed in the same way as files and the following three files are automatically opened when a program executes to provide access to the keyboard and screen.

FORMATTED INPUT/OUTPUT (2 OF 6)

- The file pointers are the means to access the file for reading and writing purpose. The table below shows the standard file and file pointers used in C programming.

Standard File	File Pointer	Device
Standard input	stdin	Keyboard
Standard output	stdout	Screen
Standard error	stderr	Your Screen

GETCHAR() & PUTCHAR() FUNCTIONS:

- The int getchar(void) function reads the next available character from the screen and returns it as an integer. This function reads only single character at a time. You can use this method in the loop in case you want to read more than one character from the screen.

FORMATTED INPUT/OUTPUT (3 OF 6)

- The int putchar(int c) function puts the passed character on the screen and returns the same character. This function puts only single character at a time. You can use this method in the loop in case you want to display more than one character on the screen. For example:

```
#include <stdio.h>
int main( )
    {int c;
    printf( "Enter name: ");
    c = getchar( );
    printf( "\nYou entered: ");
    putchar( c );
    return 0;}
```

OUTPUT

Enter name: Edmond

You entered: E

"Notice that only the first character of the input name was displayed"

FORMATTED INPUT/OUTPUT (4 OF 6)

GETS() & PUTS() FUNCTIONS:

- The char *gets(char *s) function reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF (End of File). The int puts(const char *s) function writes the string 's' and 'a' trailing newline to stdout.

```
#include <stdio.h>
int main( )
    {char str[100];
    printf( "Enter name: ");
    gets( str );
    printf( "\nYou entered: ");
    puts( str );
    return 0;}
```

OUTPUT
Enter name: Edmond
You entered: Edmond

"Notice that the string of the input name was displayed, however with warnings about the use of gets"

FORMATTED INPUT/OUTPUT (5 OF 6)

SCANF() & PRINTF() FUNCTIONS:

- The int scanf function reads the input from the standard input stream stdin and scans that input according to the format provided.
- The int printf function writes the output to the standard output stream stdout and produces the output according to the format provided.
- The format can be a simple constant string, but you can specify %s, %d, %c, %f, etc., to print or read strings, integer, character, or float, respectively.
- There are many other formatting options available which can be used based on requirements.

FORMATTED INPUT/OUTPUT (6 OF 6)

- When the code in the example below is compiled and executed, it prompts for you to input some text (name and age), then program proceeds and reads the input and displays it as following output:

```
#include <stdio.h>
int main( )
    {char str[100];
    int i;
    printf( "Enter name and age: ");
    scanf("%s %d", str, &i);
    printf( "\nYou entered: %s %d ", str, i);
    return 0;}
```

OUTPUT
Enter name: Edmond 23
You entered: Edmond 23

- Here, it should be noted that scanf() expects input in the same format as you provided %s and %d.
- Secondly, while reading a string, scanf() stops reading as soon as it encounters a space, so if you entered “Edmond James" are two strings for scanf(), and output will be zero.

CLASS EXERCISE

1. Write a C program that counts the number of occurrences of a specific character in a given string.
2. Write a C program to count the number of vowels in a string given by the user.
3. Write a C program that reads a string from the user and reverses it using the **'strlen()'** function from the 'string.h' library.
4. Write a C program that reads a string from the user and converts it to uppercases using the **'toupper()'** function from the 'ctype.h' library.