Rajarata University of Sri Lanka

Department of Physical Sciences

COM1407
Computer Programming

LECTURE 10

INPUT/ OUTPUT FUNCTIONS

T.C IRUGALBANDARA

Objectives

- At the end of this lecture students should be able to;
  - Define the C standard functions for managing input output.
  - Apply taught concepts for writing programs.

Input/ Output Methods

- 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.

Input/ Output Methods (Cont…)

- In order to perform input output functions, there are some standard C functions.
- Some of the input/ output functions are used to format the input/ output and the others for unformatted input/ output.

Input/ Output Methods (Cont…)

- The standard 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.

Input/ Output Methods (Cont…)

Standard File	File Pointer	Device	Purpose
Standard input	stdin	Keyboard	Console input from the user
Standard output	stdout	Screen	Message output to the user
Standard error	stderr	Your screen	System error message output to the user

Input/ Output Methods (Cont…)

- Some standard input output functions are;
  - getchar
  - putchar
  - scanf
  - printf
  - gets
  - puts
- These functions permits the transfer of information between the computer and the standard input/output devices.

The getchar() and 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.
- 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.

The getchar() and putchar() functions (Cont…)

```
#include <stdio.h>
int main( )
{
   char c;
   printf( "Enter a value :");
   c = getchar( );

   printf( "\nYou entered: ");
   putchar( c );

   return 0;
}
```

The getchar() and putchar() functions (Cont…)

```
#include <stdio.h>
int main( )
{
   char response;
   printf( "Do you wish to learn C language (y/n) : ?");
   response =  getchar();
   if (response == 'y')
       printf("\nRead the reference\n");
   else if (response == 'n')
       printf("\nFollow language you like\n");
   else
      printf("\nSorry wrong answer\n");
   return 0;
}
```

The getchar() and putchar() functions (Cont…)

```
#include <stdio.h>
int main( )
{
    int i;
    char let[20] = {'C', ' ', 'P', 'R', 'O', 'G', 'R', 'A', 'M' };
    printf ("Your name is : ");
    for ( i = 0; i<20; i++)
        putchar(let[i]);
    return 0;
}
```

The scanf and printf Functions

- **scanf(format string, arg1, arg2, ..., argn)**
  - Reads the input from the standard input stream **stdin** and scans that input according to the **format** provided.
  - Conversion characters

%c	single character
%d	decimal integer
%e, %f, %g	floating point value
%h	short integer
%i	decimal, hexadecimal or octal integer
%o	octal integer
%s	string of characters
%u	unsigned decimal integer
%x	hexadecimal integer

  - Each variable name must be preceded by an ampersand (&).
  - However, array names should not begin with an &.

The scanf and printf Functions (Cont…)

- **printf(format string, arg1, arg2,…., argn)**
  - function writes the output to the standard output stream **stdout** and produces the output according to the format provided.
  - printf function moves data from the computers memory to the standard output device.
  - 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.

The scanf and printf Functions (Cont…)

```
#include <stdio.h>
int main( ) {
   char str[100];
   int i;

   printf( "Enter a value :");
   scanf("%s %d", str, &i);

   i = i * 2;
   printf( "\nYou entered: %s %d ", str, i);

   return 0; }
```

The gets() and 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).
- Unlike the scanf fucntion, the gets function allows to include spaces and other characters.
- The **int puts(const char *s)** function writes the string 's' and 'a' trailing newline to **stdout**.

The gets() and puts() Functions (Cont…)

```
#include <stdio.h>
int main( ) {
   char str[100];

   printf( "Enter a value :");
   gets( str );

   printf( "\nYou entered: ");
   puts( str );

   return 0;
}
```

Objective Re-cap

- Now you should be able to:
  - Define the C standard functions for managing input output.
  - Apply taught concepts for writing programs.

Q & A

NEXT: WORKING WITH POINTERS