SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 23ITT101 PROBLEM SOLVING AND C PROGRAMMING

I YEAR /I SEMESTER

Unit 2- C-Programming Basics

Topic 6:Managing Input and Output operations

Brain Storming

1. How to perform IO operations in C?

IO statements

- In C Language input and output function are available as C compiler function or C library provided with each C compiler implementation.
- These all functions are collectively known as **Standard I/O Library function.**
- Here I/O stands for Input and Output used for different inputting and outputting statements.

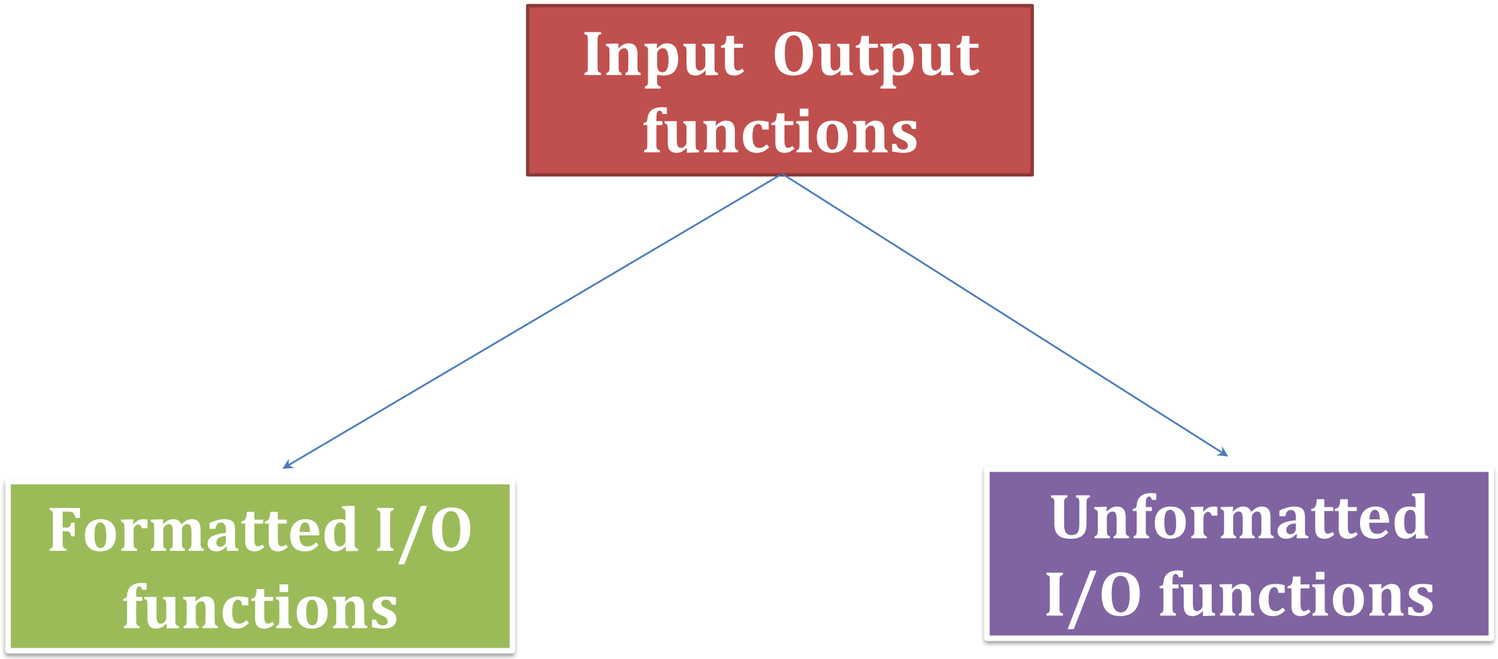
Conti...

stdin: This file is used to receive the input (usually is keyborad but can also take input from the disk file).

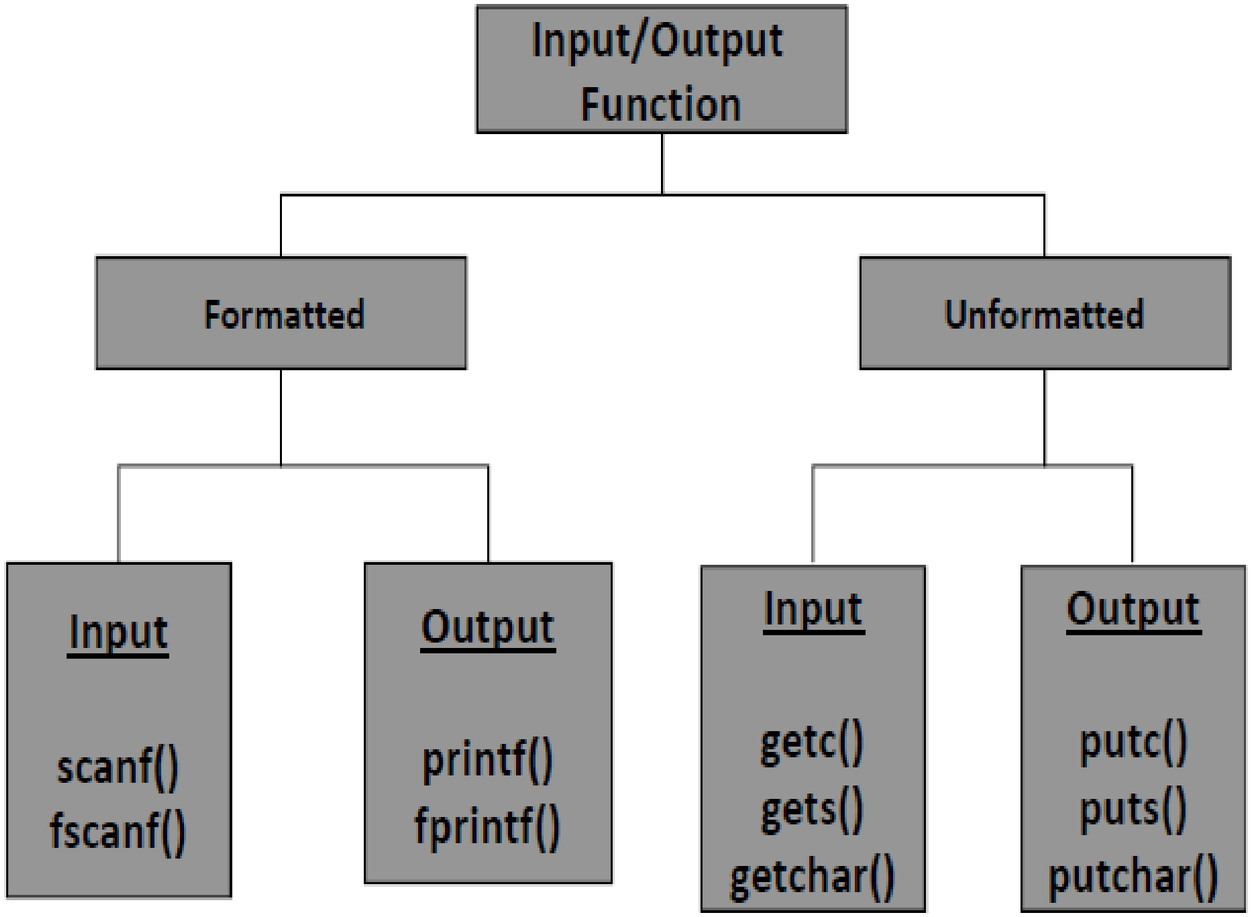
stdout: This file is used to send or direct the output (usually is a monitor but can also send the output to a disk file or any other device).

stderr: This file is used to display or store error messages.

Conti...


Input Output functions
Formatted I/O functions
Unformatted I/O functions

Conti...


Input/Output Function
Formatted
Unformatted
Input
scanf()
fscanf()
Output
printf()
fprintf()
Input
getc()
gets()
getchar()
Output
putc()
puts()
putchar()

Unformatted input/output functions

- Unformatted console input/output functions are used to read a ***single*** input from the user at console and it also allows us to display the value in the output to the user at the console.
- **we do not have to use any format specifiers in them**, to read or display a value. Hence, these functions are named **unformatted** console I/O functions.

Unformatted IO-Conti…

Functions	Description
getch()	Reads a *single* character from the user at the console, ***without*** *echoing it*.
getche()	Reads a *single* character from the user at the console, *and echoing it*.
getchar()	Reads a *single* character from the user at the console, *and echoing it*, but needs an **Enter** key to be pressed at the end.
gets()	Reads a *single* string entered by the user at the console.
puts()	Displays a *single* string's value at the console.
putch()	Displays a *single* character value at the console. alphanumeric chars
putchar()	Displays a *single* character value at the console. chars

Example: getch(), getche(), getchar();

```
#include<stdio.h>
#include<conio.h>
int main()
{
int ch;
printf("Running getch(), enter a character : ");
/* getch() does not echo the pressed character key on the screen */
ch = getch(); printf("\n");
printf("The character value you have just entered : %c", ch);
printf("\n");
printf("Running getche(), enter a character : ");
```

```
/*getche() function does echo the pressed character key on the screen */
ch = getche();
printf("\n");
printf("The character value you have just entered : %c", ch);
printf("\n");
printf("Running a getchar(), enter a character : ");
/*getchar() function echos the pressed character key and also requires the ENTER
key at the end */
ch = getchar();
printf("The character value you have just entered : %c", ch);
 return 0;
}
```

OUTPUT

```
Running getch(), enter a character :
The character value you have just entered : a
Running getche(), enter a character : g
The character value you have just entered : g
Running a getchar(), enter a character : w
The character value you have just entered : w
```

Program Analysis

- On pressing the character key **'a'**, the **getch()** stores the character in **ch** but *does not echo it on the screen*.
- On pressing the character key **'g'**, the **getche()** stores the character in **ch** and *echos it on the screen.*
- On pressing the character key **'w'**, the **getchar()** *echos it on the screen* but waits for the **Enter** key to be pressed, to store the character in **ch** variable.

Gets ( ) and puts( )

```c
/* Using puts() function to display a string*/

#include<stdio.h>

int main()
{
char country[20];
printf("Enter a country you want to visit : ");
gets(country);
printf("The country you want to visit is : ");
puts(country);

return 0;
}
```

```
Enter a country you want to visit : New Zealand
The country you want to visit is :  New Zealand
```

```c
#include<stdio.h>
#include<conio.h>

int main()
{
int ch = 'a';

printf("The character value printed using putch() : ");
putch(ch);

printf("\n");
printf("Another character value printed using putch() : ")
putch('B')

printf("\n");
printf("The character value printed using putchar() : ");
putchar(ch);

printf("\n");
printf("Another character value printed using putchar() : ")
putch('D');

return 0;
}
```

```
The character value printed using putch() :a
Another character value printed using putch() : B
The character value printed using putchar() : a
Another character value printed using putchar() : D
```

Formatted input/output functions

✓Formatted console input/output functions are used to take one or more inputs from the user at console,

✓It also allows us to display one or multiple values in the output to the user at the console

✓scanf (“control string”, arg1, arg2, ... argn);

✓ The control string specifies the field format in which the data is to be entered and the arguments arg1,arg2, ..., argn specify the address of locations where the data is stored.

✓ Control string and arguments are separated by commas.

✓Control string (also known as format string) contains field specifications, which direct the interpretation of input data.

✓It may include:

➢ Field (or format) specifications, consisting of the conversion character %, a data type character (or type specifier), and an optional number, specifying the field width.

➢ Blanks, tabs, or newlines;

scanf (“%2d %5d”, &num1, &num2);

✓The percentage sign (%) indicates that a conversion specification follows.
✓ 2 is an integer number that specifies the field width of the number to be read and d, known as data type character, indicates that the number to be read is in integer mode.

✓ scanf (“%d %c %f %s”, &count, &code,
&ratio, name);

will read the data
✓15 p 1.575 coffee

The commonly used format specifiers

Format specifier	Description
%d or %i	It is used to print the signed integer value where signed integer means that the variable can hold both positive and negative values.
%u	It is used to print the unsigned integer value where the unsigned integer means that the variable can hold only positive value.
%o	It is used to print the octal unsigned integer where octal integer value always starts with a 0 value.
%x	It is used to print the hexadecimal unsigned integer where the hexadecimal integer value always starts with a 0x value. In this, alphabetical characters are printed in small letters such as a, b, c, etc.
%X	It is used to print the hexadecimal unsigned integer, but %X prints the alphabetical characters in uppercase such as A, B, C, etc.
%f	It is used for printing the decimal floating-point values. By default, it prints the 6 values after '.'.
%e/%E	It is used for scientific notation. It is also known as Mantissa or Exponent.
%g	It is used to print the decimal floating-point values, and it uses the fixed precision, i.e., the value after the decimal in input would be exactly the same as the value in the output.
%p	It is used to print the address in a hexadecimal form.
%c	It is used to print the unsigned character.
%s	It is used to print the strings.
%ld	It is used to print the long-signed integer value

printf(“control string”, arg1, arg2, ..., argn);

Control string consists of following three types of items:

1. Characters that will be printed on the screen as they appear.
2. Format specifications that define the output format for display of each item.
3. Escape sequence characters such as \n, \t, and \b.

✓The control string indicates how many arguments follow and what their types are. The arguments arg1,arg2, ..., argn are the variables whose values are formatted and printed according to the specifications of the control string.

✓The arguments should match in number, order and type with the format specifications.

✓printf(“%d %f %s %c”, a, b, c, d);

The following examples illustrate the output of the number y = 98.7654 under different format specifications:

Format	Output
printf("%7.4f",y)	9 8 . 7 6 5 4
printf("%7.2f",y)	9 8 . 7 7
printf("%-7.2f",y)	9 8 . 7 7
printf"%f",y)	9 8 . 7 6 5 4
printf("%10.2e",y)	9 . 8 8 e + 0 1
printf("%11.4e",-y)	- 9 . 8 7 6 5 e + 0 1
printf("%-10.2e",y)	9 . 8 8 e + 0 1
printf"%e",y)	9 . 8 7 6 5 4 0 e + 0 1

Conti...

Functions	Description
scanf()	This function is used to read *one or multiple* inputs from the user at the console.
printf()	This function is used to display *one or multiple* values in the output to the user at the console.
sscanf()	This function is used to read the characters from a string and stores them in variables.
sprintf()	This function is used to read the values stored in different variables and store these values in a character array.

Why these functions are called formatted console input/output functions?

- While calling any of the **formatted** console input/output functions, we must use a specific **format specifiers** in them, which allow us to read or display any value of a specific primitive data type.

Format specifiers in console *formatted* I/O functions

Format Specifiers	Description
%d or %i	Format specifier used to read or display a *signed* **short** integer value or **short** integer value.
%u	Format specifier used to read or display an *unsigned* **short** integer value.
%ld	Format specifier used to read or display a **long** integer value or *signed* **long** integer value.
%lu	Format specifier used to read or display a *unsigned* **long** integer value.
%c	Format specifier used to read or display a **char**, character value.
%f	Format specifier used to read or display a **float**, floating point value.
%lf	Format specifier used to read or display a **double**, floating point value.
%s	Format specifier used to read or display a **string** value, to be stored in a **char[]** array.

Conti...

Format Specifiers	Description
%e or %E or %g or %G	Scientific notation of floating values
%o	Octal representation of integer
%X or %x	Hexadecimal representation of integer
%n	Prints nothing
%%	Prints % character, printf("10%%\n"); 10%

Example for formatted Functions

sprintf() and sscanf() Function

- sprintf() function is quite similar to printf() function but instead of printing the output on screen, it stores it in the character array.

```c
#include<stdio.h>

void main()
{
 int j=32;
 char cha='p';
 float a=123.2;
 char str[20];
 sprintf(str,"%d %c %f",j,cha,a);
 printf("%sn",str);

}
```

OUTPUT: 32p123.2

Conti…

Explanation

- As said earlier sprintf() doesn't print the output on screen. So it is printed through str using printf(). It just stores the data in string. In the above program, str will store the values of "j", "cha" and "a".
- **sscanf()** is the counter part of sprintf() function. It allows the programmer to store the characters of string in some other variable. These two functions are used very rarely in C.

Example for formatted Functions

fprintf() and fscanf()

- It is used to perform Input and Output operations in file.

```
#include <stdio.h>
main(){
   FILE *fp;
   fp = fopen("file.txt", "w");//opening file
   fprintf(fp, "Hello file by fprintf...\n");//writing data into file
   fclose(fp);//closing file
}
```

The fprintf() function is used to write set of characters into file.

Conti...

```c
#include <stdio.h>
main(){
   FILE *fp;
   char buff[255];//creating char array to store data of file
   fp = fopen("file.txt", "r");
   while(fscanf(fp, "%s", buff)!=EOF){
   printf("%s ", buff );
   }
   fclose(fp);
}
```

The fscanf() function is used to read set of characters from file. It reads a word from the file and returns EOF at the end of file.

Assessment 1

1. Write about formatted IO Statements?

Ans : _____

2. Write about Un-formatted IO Statements?

Ans : _____

References

1. Reema Thareja, “Programming in C”, Oxford University Press, Second Edition, 2016

Thank You