Input/ Output Functions and statements

C language has a collection of functions that can be used in a program with required number of arguments written in parantheses. Input /Output functions are also included in the user program by using the header file **<stdio.h>** which stands for standard input-output header.

Formatted Input/ Output Functions :

scanf( ) Function:

scanf() function is used to read/input values of variables using the standared input device (keyboard). It has the following form

scanf ("format string", &v1, &v2, . . . , &vn);

where v1, v2, . . . , vn are variables whose values are to be read from the keyboard.

"format string" is the control string which represents the format specification.

The symbol & (ampersand) represents the memory address where the variable value is to be stored.

For eg. scanf("%d%d", &a, &b); to read value of int variables a and b.

printf( ) Function:

printf() function is used to print/display values of variables using the standared output device (monitor). It has the following form

printf ("format string", v1, v2, . . . , vn);

where v1, v2, . . . , vn are variables whose values are to be display in the monitor.

"format string" is the control string which represents the format specification.

For eg. printf (“%f”, s);
printf (“\n sum=%6.2f”, s);
printf (“\n %d factorial is %d”, k, kfact);

Q. What is format specifier? Explain with examples.

Solution: A format specifier is a special character sequence that begins with a percent sign (%) and indicates how to write a single data item.
The format specifier in printf() determines the interpretation of a variable’s type, the width of the field, the number of decimal places printed and the justification. Format specification determines how the variables to be output should be displaed on the screen, is provided in the control string.

The format specifiers in scanf() direct scanf to read and convert characters from the input field into specific types of values, and then store them in the locations given by the address arguments.
Some commonly used format specifier with printf() and scanf() are listed below:

Format Specifiers

Format	printf() format specifier	scanf() format specifier
Single character	%c	%c
String	%s	%s
Signed decimal integer	%d or %i	%d or %i
Floating point (decimal notation)	%f	%f or %e
Floating point (exponential notation)	%e	%f or %e
Unsigned decimal integer	%u	%u
Unsigned hexadecimal integer (used with “ABCDEF” that have decimal values 10 to 15	%x	%x
Unsigned octal integer	%o	%o

Some commonly used format specifiers for qualified numbers with printf() are listed below:

Format	printf() format specifier
Decimal long integer	%ld, %li

Decimal unsigned long integer	%lu
Decimal short integer	%hd, %hi
Decimal unsigned short integer	%hu
Signed, double	%le, %lf, %lg
Signed, long double	%Le, %Lf, %Lg
Octal long integer	%lo
Hexadecimal long integer	%lx

To illustrate with a few examples: the call

```
printf("%c %d %f %e %s %d%%\n", '1',23.14,56000000, "eight",9);
```

Would print

```
1 2 3.140000 5.600000e+07 eight 9%
```

The call

```
printf("%d %o %x\n",100, 100, 100);
```

would print

```
100 144 64
```

The following are example of scanf() statements and their inputs are supplied as given:

```
scanf("%d%c",&i,&c);
```

Input: 5 x

```
scanf("%d%f%f",&x,&y,&z);
```

Input: 12 13.16 24.19

Q. How can the maximum field width for a data item be specified within a "scanf" function?

Solution: The maximum field width for a data item within a "scanf" function can be specified using an unsigned integer indicating the field width is placed within the control string, between the percent sign (%) and the conversion character. For example,

```
scanf("%3d",&a);
```

here the maximum field width for the integer variable a is specified as 3.

The number of characters in the actual data item cannot exceed the specified field width. Any characters that extend beyond the specified field width will not be read. Such leftover characters may be incorrectly interpreted as the components of the next data item.

Q. What happens if an input data item contains more characters than the maximum allowable field width in a "scanf" function? What if the data item contains fewer characters?

Escape Sequence:

An escape sequence provides special formatting control. An escape sequence consists of a backslash (\) followed by a single character. Some character such as line feed, form feed, vertical tab, alert etc., cannot be typed through the keyboard. Such invisible characters can be made understood to the 'C' compiler through the use of execution characters or escape sequences. The terms execution characters and escape sequences are used inter-changeably.

Following program segment shows the usage of '\n' and a new escape sequence '\t', called 'tab'. The newline character, '\n' when inserted in a printf()'s format string, would take the cursor to the beginning of the next line. This is also demonstrated in the following program:

```
main( )
{
        printf("You\tmust\tbe\tcrazy\nto\tlove\tthis\tbook");
}
```

And the run output on the 80 columns printer or on screen. The numerals 0, 1, 2, 3, 4, ….. show the position where the prints will take place.

```
          1         2         3         4
01234567012345678901234567890123456789
You     must    be        crazy
to      love            this      book
```

Note that '\t' moves the cursor to the next tab stop.

- The newline character is an 'escape sequence'. It is so called because the backslash symbol (\) is named as an 'escape' character.

A 80 column monitor screen has 10 tab stops. Which means, that the screen is divided into 10 zones of 8 columns each. Pressing a tab takes the cursor to the beginning of next printing zone.

- The cursor jumps over eight columns when it encounters the '\t' character.

Thus, tab pressing is another useful technique for lining up columns for the output. The '\n' character causes a new line to begin following "crazy". The tab and the newline are most used for escape sequences. Table 2 shows complete list of these escape sequences.

Table 2 Complete list of escape sequence

Esc.Seq	Purpose	Esc.Seq	Purpose
\n	New line	\t	Tab
\b	Backspace	\r	Carriage return
\f	Form feed	\a	Alert
\’	Single quote	\”	Double quote
\\	Backslash		

The escape sequence ‘\b’ moves the cursor one position to the left of its current position. ‘\r’ takes the cursor to the beginning of the line in which it is currently placed. ‘\a’ alerts the user by sounding the speaker inside the computer. Form feed(\f) advances the continuous stationery paper inside the printer to the top of the next page. Characters which are used as delimiters such as the single quote, double quote, etc., and the backslash can be printed by preceding them with the backslash. Thus, the statement:

printf(“He said, \”Let’s do it!\” ”);

Will be printed as:

He said, “Let’s do it!”

Q. What do you mean by Escape Sequence? How many types of Escape Sequences are available in the programming language ‘C’?

Assignment Statement:

An assignment statement is used to assign value to a variable. It has the following form.

Variable = value or expression

Eg. m=24;

s=s+x;

Q. Discuss the difference between assignment and equality.

Solution: Assignment means the process of assigning the value of a variable/expression from the right side of assignment operator to the left side variable. The assignment operator in C is =.

Equality means the process of checking the value of the left side variable/expression with the right side variable. The equality operator in C is = =.

Q. Write a C program to find the sum and product of given two numbers.

Solution:

```
/*  program to find sum and product  */

#include<stdio.h>
Main()
{
        int a, b, sum, product;
        scanf("%d %d",, &a, &b);
        sum=a+b;
        product=a*b;
        printf("%d %d", sum, product);
}
```

tells the compiler "to search for a file name stdio.h and place its contents at this point in the program.

ampersand symbol and before each variable name is an operator that specifies the variable names address.

conversion specification

Q. Write a user-friendly program in C to find the sum and product of two numbers.

Solution: The program in C to find the sum and product of two numbers is given below:

```
/* Program to find the sum and product of two numbers */
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
        int a,b, sum, product;
        clrscr();
        printf("Enter any two integer numbers:\n");
        scanf("%d%d",&a,&b);
        printf("%d%4d\n",a,b);
        sum=a+b;
        product=a*b;
        printf("The sum of %d and %d is = %d\n",a,b,sum);
        printf("The product of %d and %d is = %d\n",a,b,product);
        printf("Press any key to exit...\n");
        getch();
}
```

The output of the above program will be:

```
Enter any two integer numbers:
4   5
The sum of 4 and 5 is = 9
The product of 4 and 5 is = 20
Press any key to exit...
```

Q. Temperature of a city in Fahrenheit degrees is input through the keyboard. Write a program to convert this temperature into Centigrade degrees.

Solution:

```
/* Conversion of temperature form Fahrenheit to Centigrade */
#include<stdio.h>
#include<conio.h>
main()
{
        float fr,cent;
        clrscr();
        printf("\nEnter the temperature(F): ");
        scanf("%f",&fr);
        cent=5.0/9.0*(fr-32);
```

```
        printf("\nTemperature in centigrade=%.3f",cent);
        printf("\n\n\nPress any key to exit...");
        getch();
        return;
}
```

Q. Two numbers are input through the keyboard into two locations C and D. Write a program to interchange the contents of C and D.

Solution:

```
/* Interchanging of contents of two variable c & d  */
#include<stdio.h>
#include<conio.h>
main()
{
        int c, d, e;
        clrscr();
        printf("\n Enter the number at location C: ");
        scanf("%d",&c);
        printf("\n Enter the number at location D: ");
        scanf("%d",&d);
        /*Interchanging the contents of two variable using a third variable
        as temporary store */
        e=c;
        c=d;
        d=e;
        printf("\n New Number at location C=%d",c);
        printf("\n New Number at location D=%d",d);
        printf("\n\n\n Press any key to exit...");
        getch();
        return;
}
```

Q. Write a program to calculate the area & perimeter of the rectangle and the area & circumference of the circle. The length & breadth of a rectangle and radius of a circle are input through the keyboard.

Solution:

```
/* Calculation of perimeter & area of rectangle and circle */
#include<stdio.h>
#include<conio.h>
main()
{
        int l,b,r,area1,perimeter;
        float area2,circum;
        clrscr();
        printf("\nEnter Length & Breadth of a Rectangle: ");
        scanf("%d%d",&l,&b);
        area1=l*b; /* Area of a rectangle */
        perimeter=2*l+2*b; /* Perimeter of  a rectangle */
        printf("\nArea of the Rectangle is=%d",area1);
        printf("\nPerimeter of the Rectangle is=%d",perimeter);
        printf("\n\nEnter Radius of a circle: ");
        scanf("%d",&r);
        area2=3.14*r*r; /*Area of Circle */
        circum=2*3.14*r; /*Circumference of a circle */
        printf("\nArea of the Circle is=%.2f",area2);
        printf("\nCircumference of a Circle is=%.2f",circum);
        printf("\n\n\nPress any key to exit...");
        getch();
        return;
}
```

Character Input/Output Functions:

getchar( ) Function

getchar( ) function is used to read one charater at a time from the standard input device(keyboard). It has the following form:

ch= getchar ( );

where ch is a **char** variable.

Eg.
```
char c;
c=getchar ( );
```

When this function is executed, the computer will wait for a key to be pressed and assigns the value to the variable when the entre key is pressed.

putchar( ) Function

putchar( ) function is used to display one charater at a time from the monitor screen. It has the following form:

putchar (ch);

where ch is a **char** variable.

Eg. char c='M';
putchar (c);

When this function is executed, the computer will display the value of the **char** variable (i.e. letter M) on the monitor screen.

getch( ) Function

getch( ) function is used to read a charater from the keyboard and it does not expect the enter key press. It has the following form:

ch= getch( );

where ch is a **char** variable.

Eg. char c;
c=getch( );

When this function is executed, the computer will wait for a key to be pressed from the key board. As soon as a key is pressed, the control is transferred to the next line of the program and the value is assigned to the **char** variable. It is to be noted that the key(lettrer) pressed will not be display on the monitor screen.

putch( ) Function

putch( ) function is used to display a charater on the monitor screen. It has the following form:

putch(ch);

where ch is a **char** variable.

Eg. char c='M';
putch(c);

When this function is executed, the computer will display the value of the **char** variable 'M' on the monitor screen.

getche( ) Function

getche( ) function is used to read a charater from the keyboard without expecting the entre key press. However any key pressed by the user will be display on the monitor. It has the following form:

ch= getche( );

where ch is a **char** variable.

Eg. char c;
c=getche( );

When this function is executed, the computer will wait for a key to be pressed from the key board. The value is assigned to the **char** variable.

Noted that getche() is similar to getch() expect that getche() display the key pressed from the keyboard on the monitor screen. The ‘e’ at the end of getche() stands for echo. The header file **<conio.h>** (console input output header) is included to ues these functions in a program.

gets( ) Function

gets( ) function is used to read a string of characters including white space. Note that a string containing white spaces cannot be read using scanf() with “%s” format specifer. It has the following form:

gets(st);

where st is a **char string**variable.

Eg. char st[20];
gets(st);

When this function is executed, the computer waits for the string value.

puts( ) Function

puts( ) function is used to display a charater string on the monitor screen. It has the following form:

puts(str);

where str is a string(array of characters).

Eg. char str[20]=’Computer World’;

puts(str);

When this function is executed, the computer will display the string (i.e. Computer World) on the monitor screen.

clrscr( ) Function

clrscr( ) function is used to clear the monitor screen. It has the following form:

clrscr( );

When this function is executed, the previously displayed text/error messages will be cleared. This is used when a new program is run. It is to be noted that the header file **<conio.h>** (console input output header) is included to use this function in a program.

Q. Explain the sizeof( ) operator with a suitable example.

Solution:

The sizeof Operator: The sizeof operator is used to return the size (in bytes) of its operand. Is is used in the form:

sizeof(b)

Where b is a variable or a type.

Example:

The following program demonstrates the use of the sizeof operator to display the size of the different data types in bytes.

```
#include<stdio.h>
main()
{
    printf("\nThe Size of char is %d ",sizeof(char));
    printf("\nThe Size of int is %d ",sizeof(int));
    printf("\nThe Size of float is %d ",sizeof(float));
    printf("\nThe Size of double is %d ",sizeof(double));
    return;
}
```

RUN:

```
The Size of char is 1
```

The Size of int is 2
The Size of float is 4
The Size of double is 8

Q. Write a C program for calculating volume and surface area of a sphere given diameter of the sphere.

Solution:

```
/* Calculate the volume & surface area of sphere given diameter of the sphere */
#include<stdio.h>
#define PI  3.14
main( )
{
        float radius, volume, area;
        printf("\nEnter the radius of sphere :  ");
        scanf("%f", &radius);
        volume = (4*PI*radius*radius*radius)/3;
        area=4*PI*radius*radius;
        printf("Volume of sphere = %f", volume);
        printf("\n Surface area of sphere = %f",area);
}
```

Q. A student obtained marks in five different subjects are input through the keyboard. Write a program to find out the aggregate marks and percentage marks obtained by the student. Assume that the maximum marks that can be obtained by a student in each subject is 100.

Solution:

```
/* Calculation of aggregate & percentage marks */
#include<stdio.h>
```

```
#include<conio.h>
main()
{
        int m1,m2,m3,m4,m5,aggr;
        float per;
        clrscr();
        printf("\nEnter marks in 5 subjects: ");
        scanf("%d%d%d%d%d",&m1,&m2,&m3,&m4,&m5);
        aggr=m1+m2+m3+m4+m5;
        per=(float)aggr/5;
        printf("\nAggregate Marks=%d",aggr);
        printf("\nPercentage Marks=%.2f",per);
        printf("\n\n\nPress any key to exit...");
        getch();
        return;
}
```

Q. Define the following:

a. comment b. printf( ) c. delimiters

Solution:

a. Comment: Comment is an entry in a computer program for the purpose of documentation or explanation. The line starting with the characters /* and ending with the characters / is the comment. Comments are important because they help in documenting a program. Comments should be written so that the logic of a program is understand easily. We may type as many lines as we wish between these two sets of characters i.e. (/*, */) for writing comments. Note that the compiler ignores comments, and does not take any action.

b. printf( ): The printf( ) function is one of the most important and useful functions to display the formatted output data items on the standard output device such as a monitor.

c. delimiters: Following the function definition, there are braces ({ }) to indicate the beginning and end of the body of the function. The opening

brace ({) means a block of code that forms a distinct unit is about to begin. The closing brace(}) terminates that block of programming code. These set of braces are the delimiters for indicating the beginning and end of the program segment.

Q. What are the commonly used input/output function in C? How are they accessed?

Solution: There are a couple of function that provide basic I/O facilities.
Pobably the most common are: `getchar()` and `putchar()`. They are defined and used as follows:

`int getchar(void)` -- reads a char from `stdin`

`int putchar(char ch)` -- writes a char to `stdout`, returns character written.

```
   int ch;

            ch = getchar();
            (void) putchar((char) ch);
```

Related Functions:

```
  int getc(FILE *stream),
int putc(char ch,FILE *stream)
```

Formatted I/O

We have seen examples of how C uses formatted I/O already. Let's look at this in more detail.

Printf :

The function is defined as follows:

`int printf(char *format, arg list ...)` --
prints to `stdout` the list of arguments according specified format string. Returns number of characters printed.

The **format string** has 2 types of object:

ordinary characters -- these are copied to output.

conversion specifications -- denoted by % and listed in the following Table :

Table: Printf/scanf format characters

Format Spec (%)	Type	Result
c	char	single character
i,d	int	decimal number
o	int	octal number
x,X	int	hexadecimal number
		lower/uppercase notation
u	int	unsigned int
s	char *	print string

		terminated by \0
f	double/float	format -m.ddd...
e,E	"	Scientific Format
		-1.23e002
g,G	"	e or f whichever
		is most compact
%	-	print % character

Between % and format char we can put:
- (minus sign)
-- left justify.
integer number
-- field width.
m.d
-- m = field width, d = precision of number of digits after decimal point or number of chars from a string.
So, `printf("%-2.3f\n",17.23478);`
The output on the screen is:

```
          17.235
```

and, `printf("VAT=17.5%%\n");`
...outputs:

```
          VAT=17.5%
```

scanf :
This function is defined as follows:
`int scanf(char *format, args....)` -- reads from stdin and puts input in address of variables specified in `args` list. Returns number of chars read.
Format control string similar to `printf`
Note: The ADDRESS of variable or a pointer to one is required by `scanf`.

```
scanf(``%d'',&i);
```

We can just give the name of an array or string to scanf since this corresponds to the start address of the array/string.

```
              char string[80];
            scanf(``%s'',string);
```

&&&&&&&&&&