UNIT 2
User Interaction

Q. What is computer and output function?

A computer is a device that takes data as input, process that data and generates to output. While output function is used to get data from the program. Output produced by monitor is called standard output. C language offers printf function to display the output.

Q. Explain the printf() function of C language with an example. (or) What function of C language is used to display output on screen?

printf:

printf is a built-in function in C programming language to show output on screen. Its name comes from "print formatted" that is used to print the formatted output on screen. All data types can be displayed with printf function. To understand the working of printf function, consider the following programming example:

Example:

Program	Output:

#include<stdio.h>
void main()
{
printf("Hello World");
}
	Hello World

In this example, printf function is used to display Hello World on screen. Whatever we write inside the double quotes "and" in the printf() function, gets displayed on screen.

Q. Why format specifiers are important to be specified in I/O (Input/ Output) operations?

Format specifier represents data type field width and format of a value of a variable displayed on the screen. A format specifier always begins with the symbol %. Format specifiers are used for both input and output statements. The general syntax of format specifier is. %

Example:

If we want to display the value of a variable? Let's declare a variable and then check the behaviour of printf.

int age = 35;

Now I want to display the value of this variable age on screen. So, I write the following statement:

printf("age");

But, it does not serve the purpose, because it displays the following output on screen. **age**

It does not display the value stored inside the variable age, instead it just displays whatever was written inside the double quotes of printf. In fact, we need to specify the format of

data that we want to display, using format specifiers. Following table shows format specifiers against different data types in C language.

Data type	Format Specifier
int	% d or % i
float	% f
char	% c

Suppose we want to show int type data, we must specify it inside the printf by using the format specifier % d or %i. In the same way, for float type data we must use %f. It is shown in the following example:

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      float height = 5.8;
      int age = 35;
      printf(" My age is %d and my height is %f ", age, height);
      getch();
}
```

Output:

```
My age is 35 and my height is 5.800000
```

We can observe that while displaying output, first format specifier is replaced with the value of first variable/data after the ending quotation mark i.e. age in the above example, and second format specifier is replaced with the second variable/data.

When we use %f to display a float value, it display 6 digits after the decimal point. If we want to specify the number of digits after decimal point then we can write %.nf where n is the number of digits. In the above example, if we write the following statement.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      float height = 5.8;
      int age = 35;
      printf(" My age is %d and my height is %. 2f ", age, height);
      getch();
}
```

Output:

```
My age is 35 and my height is 5.80
```

Format specifiers are not only used for variables. Actually they are used to display the result of any expression involving variables, constants, or both, as shown in the example below.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    printf("Sum of 23 and 45 is %d", 23 + 45);
    getch();
}
```

Output
Sum of 23 and 45 is 68

Q. Write a note on scanf?

scanf:

scanf is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

Consider the following example.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char grade;
        scanf("%c", &grade);
        getch();
}
```

In this example, %c format specifier is used character type for the input variable. Input entered by user is saved in variable grade.

There are two main parts of scanf function as it can be seen from the above code. First part inside the double quotes is the list of format specifiers and second part is the list of variables with & sign at their left.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int number;
    printf("Enter a number between 0-10:");
    scanf("%d", &number);
    printf("The number you entered is: %d", number);
    getch();
}
```

Output
Enter a number between 0-10: 4
The number you entered is : 4

We can take multiple inputs using a single scanf function e.g. consider the following statement.

scanf("%d%d%f", &a, &b, &c);

It takes input into two integer type variables a and b, and one float type variable c. After each input, user should enter a space or press enter key. After the entire input user must press enter key.

It is a very common mistake to forget & sign in the scanf function. Without & sign, the program gets executed but does not behave as expected.

Q: Write a note on getch()?

getch():

getch() function is used to read a character from user. The character entered by user does not get displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key. To use this function, we need to include the library conio.h in the header section of program.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Enter any key if you want to exit program");
        getch();
}
```

The above program wants user to enter any key and then waits for the user's input before finishing the execution of program.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char key;
        printf("Enter any key:");
        key = getch(); // Gets a character from user into variable key
        getch();
}
```

If we run this program, we notice a difference between reading a character using scanf and reading a character using getch functions. When we read character through scanf, it requires us to press enter for further execution. But in case of getch, it does not wait for enter key to be pressed. Function reads a character and proceeds to the execution of next line.

Q. Write a note on clrscr()?

clrscr();

This function is used to clear the output screen of C language editor.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int number;
    printf("Enter a number between 0-10:");
    scanf("%d", &number);
    printf("The number you entered is: %d", number);
    getch();
}
```

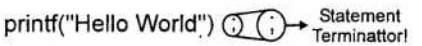
Output

```
Enter a number between 0-10: 4
The number you entered is    : 4
```

Q. What is a Statement Terminator?

Statement Terminator:

A statement terminator is identifier for compiler which identifies end of a line. In C language semi colon (;) is used as statement terminator. If we do not end each statement with a; it results into error.



Q. What are escape sequences? Why do we need them?

Escape Sequence:

Escape sequence are used in printf functioh inside the double quotes (" "). it force printf to change its normal behaviour of showing output. Let's understand the concept of an escape sequence by considering the following example statement:

printf ("My name is \"Ali \" ");

The output of above statement is: **My name is "Ali"**

In the above example \" is an escape sequence. It causes printf to display "on computer screen".

Formation of escape sequence

Escape sequence consist of two characters. The first character is always back slash (\) and the second character varies according to the functionality that we want to achieve. Back slash (\) is called escape character which is associated with each escape sequence to notify about escape. Escape character and character next to it are not displayed on screen, but they perform specific task assigned to them. The following escape sequences are also commonly used in C languages:

Sequence	Purpose	Sequence	Purpose
\'	Display Single Quote (')	\a	Generates an alert sound
\\	Display Back slash(\)	\b	Removes previous char
\t	Display 8 spaces	\n	Move on next line

New line (\n)

After escape character, n specifies movement of the cursor to start of the next line. This escape sequence is used to print the output on multiple lines.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("My name is Ali. \n");
        printf("I live in Lahore");
        getch();
}
```

Output

```
My name is Ali.
I live in Lahore
```

In the absence of an escape sequence, even if we have multiple printf statement, their output is displayed on a single line. Following example illustrates their point.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("My name is");
        printf("Ahmad");
        getch();
}
```

Output

```
My name is Ahmad
```

Tab (\t)

Escape sequence \t specifies the I/O function of moving to the next tab stop horizontally. A tab stop is collection of 8 spaces. Using \t takes cursor to the next tab stop. This escape sequence is used when user presents data with more spaces.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Name: \tAli\nFname: \tHammad\nMarks: \t1000");
        getch();
}
```

Output

```
Name:   Ali
Fname: Hammad
Marks:  1000
```

Q. What are Operators?

Operators:

Operators are the mathematical symbols that perform some operation on operands. Operands are the values or variables. There are many operators used in C language, some of these are:

- Assignment operator
- Arithmetic operator
- Logical operator
- Relational operator

Q. Explain Assignment Operator.

Assignment Operator:

Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable. Equal sign (=) is used as assignment operator in C. Consider the following example: **int sum = 5;**

Values 5 is assign to a variable named **sum** after executing this line of code. Let's have a look at another example:

```
int sum = 6;
int var  = sum;
```

First, value 6 is assign to variable sum. In the next line, the value of sum is assigned to variable var.

Example: Write a program that swaps the values of two integer variables.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int a = 2, b = 3, temp;
        temp = a;
        a = b;
        b = temp;
        printf("Value of a after swapping: %d\n:", a);
        printf("Value of b after swapping: %d\n:", b);
        getch();
}
```

Q. Which operators are used for arithmetic operations?

Arithmetic Operators:

Arithmetic Operators are used to perform arithmetic operations on data. Following table represents arithmetic operators with their description.

Operator	Name	Description
/	Division Operator	It is used to divide the value on left side by the value on right side.
*	Multiplication Operator	It is used to multiply two values.
+	Addition Operator	It is used to add two values.
-	Subtraction Operator	It is used to subtract the value on right side from the value on left side.
%	Modulus Operator	It gives remainder value after dividing the left operand by right operand.

Division

Division operator (/) divides the value of left operand by the value of right operand.

e.g. **float result = 3.0 / 2.0;**

After the statement, the variable result contains the value 1.5.

If both the operands are of type int, then result of division is also of type int. Remainder is truncated to give the integer answer. Consider the following line of code.

float result = 3 / 2;

As both values are of type int so answer is also an integer which is 1. When the value 1 is assigned to the variable result of type float, then this 1 is converted to float, so value 1.0 is stored in variable result. If we want to get the precise answer then one of the operands must be of floating type. Consider the following line of code:

float result = 3.0 / 2;

In the above example, the value stored in variable result is 1.5.

Example: Write a program takes as input the price of a box of chocolates and the total number of chocolates in the box. The program finds and displays the price of one chocolate.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        float box_price, num_of_chocolates, unit_price;
        printf("Please enter the price of whole box of chocolates:");
        scanf("%f ", &box_price);
        printf("Please enter the number of chocolates in the box:");
        scanf("%f ", &num_of_chocolates);
        unit_price = box_price / num_of_chocolates;
        printf("The price of a single chocolates is: %f ", unit_price);
        getch();
}
```

Output

```
Please enter the price of whole box of chocolates: 150
Please enter the number of chocolates in the box: 50
The price of a single chocolates is: 3.000000
```

Multiplication

Multiplication operator (*) is a binary operator which performs the product of two numbers. e.g. **int multiply = 5 * 5;**

After the execution of statement, the variable multiply contains value 25.

Example: Write a program that takes as input the length and width of a rectangle. Program calculates and displays the area of rectangle on screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        float length, width, area;
        printf("Please enter the length of rectangle:");
        scanf("%f ", &length);
        printf("Please enter the width of rectangle:");
        scanf("%f ", &width);
        area = length * width;
        printf("Area of rectangle is : %f ", area);
        getch();
}
```

Output

```
Please enter the length of rectangle: 6.5
Please enter the width of rectangle: 3
Area of rectangle is: 19.500000
```

Addition

Addition operator (+) calculates the sum of two operands.

e.g. **int add = 10 + 10;**

Resultant values in variable add is 20.

Example: Write a program that takes marks of two subjects from user and displays the sum of marks on console.

Program

```
#include <stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int sum, math, science;
        printf("Enter marks of Mathematics:");
        scanf("%d", &math);
        printf("Enter marks of Science:");
        scanf("%d", &science);
        sum = math + science;
        printf("Sum of marks is : %d", sum);
        getch();
}
```

Output

```
Enter marks of Mathematics: 90
Enter marks of Science: 80
Sum of marks is: 170
```

The statement a = a + 1; is used to increase the value of variable a by 1. In C language, this statement can also be written as a++; or ++a; Similarly, a - - ; or - - a; used to decrease the value of a by 1.

Subtraction

Subtraction operator (-) subtracts right operand from the left operand.

e.g. **int result = 20 – 15;**

After performing subtraction, value 5 is assigned to the variable result.

Modulus operator

Modulus operator (%) performs divisions of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

e.g. **int remaining = 14 % 3;**

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable remaining is 2.

Example: Write a program that finds and displays the right most digit of an input number.

Program

```
#include <stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int num, digit;
        printf("Enter a number:");
        scanf("%d", &num);
        digit = num % 10;
        printf("Right most digit of number you entered is: %d", digit);
        getch();
}
```

Output

```
Enter a number: 789
Right most digit of number you entered is: 9
```

While writing arithmetic statement is C language, a common mistake is to follow the usual algebraic rules e.g. writing **6 * y** as 6y, and writing **x * x * x** as x^3 etc. It results in a compiler error.

Q. What are relational operators? Describe with an example.

Relational Operators:

Relational operators compare two values to determine the relationship between values. Relational operators identify either the values are equal, not equal, greater than or less than one another. C language allows us to perform relational operators on numeric and char type data. The basic relational operators with their description are given below:

Relational Operator	Description
= =	Equal to
! =	Not equal
>	Greater than
<	Less than
> =	Greater than equal to
< =	Less than equal to

Relational operators perform operations on two operands and return the result in Boolean expression (true or false). A true value is represented by 1, whereas a false value is represented by a 0. This concept is further shown in following table:

Relational Expression	Explanation	Result
5 == 5	5 is equal to 5?	**True**
5 ! = 7	5 is not equal 7?	**True**
5 > 7	5 is greater than 7?	**False**
5 < 7	5 is less than 7?	**True**
5 > = 5	5 is greater than or equal to 5?	**True**
5 < = 4	5 is less than or equal to 4?	**False**

Q. What is the difference between == operator and = operator?

Assignment operator (=) and equal to operator (==)

In C language, == operator is used to check for equality of two expressions, whereas = operator assigns the result of expression on right side to the variable on left side. Double equal operator (==) checks whether right and left operands are equal or not. Single equal operator (=) assigns right operand to the variable on left side.

We can also use printf function to show the result of a relational expression, e.g. Consider the following examples:

printf("%d", 5 == 5); // This statement displays 1

printf(" %d" , 5 > 7); // This statement displays 0

Q. What are logical operators? Describe with an example.

Logical Operators:

Logical operators perform operations on Boolean expression and produce a Boolean expression as a result.

The result of a relational operation is a Boolean expression. Logical operators can be performed to evaluate more than one relational expression. Following table shows the basic logical operators and their description:

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

AND operator (&&):

AND operator && takes two Boolean expressions as operands and produces the result true if both of its operands are true. It returns false if any of the operands is false. The truth table for AND operator is shown below:

Expression 1	Operator	Expression 2	Result
False	&&	False	**False**
False	&&	True	**False**
True	&&	False	**False**
True	&&	True	**True**

OR operator (||):

OR operator accepts Boolean expression and returns true if at least one of the operands is true. The truth table for OR operator is shown below:

Expression 1	Operator	Expression 2	Result
False	||	False	**False**
False	||	True	**True**
True	||	False	**True**
True	||	True	**True**

NOT operator (!):

NOT operator negates or reverses the value of Boolean expression. It makes it true, if it is false and false if it is true. The truth table for Not operator is given below:

Expression	Operator	Result
True	!	**False**
False	!	**True**

Examples of Logical Operators:

Following table shows the concept of logical operators with the help of examples.

Logical Expression	Explanation	Result
3 < 4 && 7 > 8	3 is less than 4 AND 7 is greater than 8 ?	**False**
3 == 4 || 3 > 1	3 is equal to 4 OR 3 is greater than 1 ?	**True**
!(4 > 2 || 2 == 2)	Not (4 is greater than 2 OR 2 is equal to 2) ?	**False**
6 < = 6 && !(1 > 2)	6 is less than or equal to 6 AND NOT (1 is greater than 2) ?	**True**
8 > 9 || !(1 <= 0)	8 is greater than 9 OR NOT (1 is less than or equal to 0) ?	**True**

C language performs **short-circuit evaluation**. It means that:

1: While evaluating an **AND** operator, if sub expression at left side of the operator is **false** then the result is immediately declared as false without evaluating complete expression.

2: While evaluating an **OR** operator, if sub expression at left side of the operator is **true** then the result is immediately declared as true without evaluating complete expression.

Q. What is the difference between unary operators and binary operators?

Unary vs Binary Operators:

All the operators discussed can be divided into two basic types, based on the number of operands on which the operator can be applied.

Unary Operators:

Unary operators are applied over one operand only e.g. logical NOT (!) operator has only one operand. Sign operator (-) is another example of a unary operator e.g. -5.

Binary Operators:

Binary operators require two operands to perform the operation e.g. all the arithmetic operators, and relational operators are binary operators. The logical operators && and || are also binary operators.

Ternary Operator:

The operator offers three operands in C programming Language is called ternary operator.

Q. What is meant by precedence of operators? Which operator has the highest precedence in C language?

Operator's Precedence:

If there are multiple operators in an expression, the question arises that which operator is evaluated first. To solve this issue, precedence has been given to each operator. An operator with higher precedence is evaluated before the operator with lower precedence. In case of equal precedence, the operator at left side is evaluated before the operator at right side.

Operator	Precedence
( )	1
!	2
*, /, %	3
+, -	4
>, <, >=, <=	5
==, !=	6
&&	7
||	8
=	9

Example:

```
result = 18 / 2 * 3 + 7 % 3 + (5 * 4);        // evaluate ( )
result = 18 / 2 * 3 + 7 % 3 + 20;             // evaluate /
result = 9 * 3 + 7 % 3 + 20;                  // evaluate *
result = 27 + 7 % 3 + 20;                     // evaluate %
result = 27 + 1 + 20;                         // evaluate +
result = 28 + 20;                             // evaluate +
result = 48;
```