# Chapter-1

# Introduction to ‘C' Language

Different programming languages develop for the Computer. These are designed for specific purpose. Machine language was first language to be developed as Computer programming language. It is very hard to write a Computer program in this language because it has very large Instruction set. It can directly run hardware of Computer System.

FORTRAN was developed for solution of scientific and mathematical problems and COBOL was developed to solve commercial problems. 'C' language was developed for the need of the current scenario. This language can be used in all types of works.

In 1972, 'C' language was developed by Dennis Ritchie in the Bell Telephone Laboratories Industry (Now, it is AT & T Bell Laboratories) as the successor of BCPL (Basic Combined Programming Language) and 'B'. This language can be used with both Unix and Dos Operating systems. It requires different compiler program. The new version 'C++' of ANSI 'C' and TURBO 'C' was developed in 1980.

## 1.1 Charecterstic of ‘C' language

1. 'C' language is a flexible language so that it is easy to make different kinds of programs.

2. The program is written in this language, can run speedily.

3. 'C' language is a high level language but it also has the characteristics of low level language.

4. The program is written in this language can store in less memory.

5. 'C' language program can be arranged in blocks, so that it can easy to make large program.

6. Pointers are used to make the program in 'C' language.

7. Bit operators (0 or 1) are presented in the 'C' language, which makes easy to work with Bytes.

## 1.2 Structure of Program

The program of 'C' language is the group of different functions. Every function is a group of statements and it performs a special task. The first function called by the compiler is main( ) function. It must be present in every 'C' language program. It is very important to know the structure of the program, where, we have to write the statements in the program. The structure of the program is as following :

1. Pre-Processor directive or/ and Header file
2. Global declarations of variables and Function Prototype
3. main( )
4. {
5. Local declarations of variables and Function Prototype of main function
6. Single or compound statements
7. }
8. Header of other function with its arguments
9. {
10. Local variables of function
11. Single or compound statements
12. }

### Description

**Line No. 1:** The header file selects according to the library functions have to be used in which contain information that must be included in the program when it complied. If we want to include maths library function (for example sin(), cos() and sqrt() ) in the program then we have to write following statement.

#include<math h>

A program can contain one or more header file and write in any order.

### Pre-Processor Directive

The pre-processor directives are executed before the execution of the program by the compiler. They give the directions to the compiler. Some pre-processor directives

are #include, #define etc.

The #include pre-processor directive is used to include header files in the program and the #define is used to declare symbolic constants in the program.

Example :

```
#define A 10
```

A is a constant and its value is 10.

**Line No. 2 :** The variables are declared in this line. These variables can be used anywhere in the program. It can be used in function only if the same name variable is not declared in the function.

Example :

```
int A;

float B, C;
```

Here, the function prototypes are also be declared. And these functions can be called in any part of the program.

**Line No. 3 :** It is main( ) function. Every 'C' program must contain main() function because the first function called by compiler is main() function.

**Line No. 4 :** Start the boundary of the main() function with open curly bracket ( { ).

**Line No. 5 :** Variables are declared after the curly bracket, can be used within main() function.

These are also called local variables. These will be declared as global variables (Line No. 2). Here, the function prototypes are also be declared. And these functions can be called within the main() function.

**Line No. 6 :** Definition of main() function is written here. It includes single or compound statements.

**Line No. 7 :** End of the main() function boundary with close curly bracket ( { ).

**Line No. 8, 9, 10, 11, and 12 :** These lines are similar to line 3, 4, 5, 6, and 7. The function name and its arguments are written at line no. 8 that is also called header of the function.

These lines are optional and these lines write when a user defines a function in the program.

The comments are used for documentation. The compiler does not execute the comments. These are written between /* and */.

**Program 1 :** Write a 'C' language program to findout the area of the rectangle.

```
#include<stdio.h>

/* To calculate the of Rectangle.*/

main()

{

        float a,b,area;

        printf("Enter Side of the Rectangle :\n");

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

        area = a * b;

        printf("\nThe area of the Rectangle = %f",area);

}

        Enter Side of the Rectangle : 12 14

        The area of the Rectangle =  168
```

**Program 2 :** Write a 'C' language program to calculate the given formula.

```
#include<stdio.h>

main()

{

        float m,v,C;

        printf("Enter The value of m & v :\n");

        scanf("%f %f",&m,&v);

        C =0.5 * m * v * v;

        printf("\nThe value of C = %f",C);

}
```

Enter The value of m & v : 2.5 1.7

The value of C = 3.612500

## 1.3 Character set of 'C'

The 'C' language includes lower and upper case letters, digits, and special symbols.

Alphabets A, B, C, . . . . . Z

a, b, c, . . . . . z

Digits 0, 1, 2, 3, . . . . . 9

Some of the special characters are shown in the table 1.1

**Table - 1.1**

| Symbol | Meaning | Symbol | Meaning |
|---|---|---|---|
| # | Hash | ? | Question Mark |
| < | Less than | : | Colon |
| > | Greater than | ; | Semicolon |
| = | Equal to | ^ | Caret Sign |
| + | Plus | & | Ampersand |
| - | Minus | ~ | Tiled |
| * | Asterisk | $ | Dollar sign |
| % | Percentage | , | Comma |
| / | slash | \ | Back slash |
| @ | At the rate | ' | single quotes |
| " | Double quote | ! | Exclamation sign |
| \| | Filter sign | | |

## 1.4 Identifier

Identifiers are the name given to various program elements such as variable name,

function name, and symbolic constant name etc. by the user.

The rules for writing identifier are following :

1. The first letter of identifier must be a letter.

2. After the first letter, the identifier can include both uppercase and lowercase letters, all digits and underscore ( _ ).

3. The maximum size of the identifier is 31 characters.

4. Keyword ( or reserve word) cannot be used as identifier.

5. 'C' language is case sensitive language, lowercase letters are different from uppercase letters. Mostly, should use lowercase letters.

**1.5 Constant :**

The value of the identifier does not change during the execution of the program. Constants are three types.

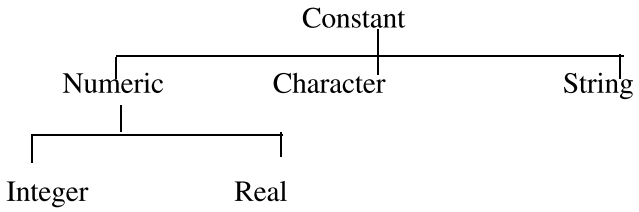



Fig. 1.1 Types of contants

**1. Numeric Constant :**

These are three types

**(a) Integer Constant :**

These are made of numbers and not included decimal points. These are shown in three different number system :

(i) Decimal (Base 10)

(ii) Octal (Base 8)

(iii) Hexa decimal (Base 16)

Numbers and characters used in different number systems are shown in the Table-1.2.

Table-1.2

| Type | Digits / alphabet | Example |
|---|---|---|
| **Decimal** | 0, 1, 2, ..... 9 | 0, 279, 972, 32767 |
| **Octal** | 0, 1, 3, ..... 7 | 02, 027, 07777<br>Octal number starts with 0 (zero). |
| **Hexa decimal** | 0, 1, 2, ... 9 and a, b, c .... (upper and lower case) | 0x1, 0xab, 0xffff<br>Hexa Decimal number starts with 0x or 0X. |

**(b) Real Constant :**

The number which includes decimal points, they are called real number constant.

Example :

0.96, 872.127, 2.0 E-7

0.0693, 1.7676E + 10

**2. Character Constant :**

When a character written in a single quotation then it is called single character constant. It includes alphabets, numbers and special characters.

Example :

'A' '5' 'b' '$'

**3. String Constants :**

When zero, one or more than one characters encloses within the double quotation then it is called string constant.

Examples :

"Home Loan" "ALWAR"

" " "2 * 6 * 7 - 5 + 2"

### 4. Back Slash Character Constant

These are also character constants, which are not used for printing character. These character constant are used for controlling the output. They are written with back slash ( \ ). Some black slash constant and their meaning are shown in the table-1.3.

**Table-1.3**

| Escape Sequence | Meaning |
|---|---|
| \a | bell (alert) |
| \t | horizontal tab |
| \v | vertical tab |
| \n | new line |
| \v | carriage return |
| \' | quotation mark |
| \b | back space |
| \0 | NULL |

## 1.6 Variable

Variable are identifiers, which store the value, and value can be changed during the execution of the program. The same rules are applicable on variable, which are used for writing identifier. Some valid variables are :

employee, result123, roll_num, min

And some invalid variables are :

| | |
|---|---|
| short | Reserve word |
| b' s | Special character |
| 5ort | First letter must be alphabet |
| int eger | blank space |

## 1.7 Data Type

'C' programming language support different types of Data types. They takes different

size in memory to store the data. Generally, 'C' language uses four type of data types.

1. int — To store integer value

2. char — To store one character

3. float — To store single precision floating point

4. double — To store double precision floating point

Some of the datatype, size, keyword and range are shown in Table-1.4.

**Table-1.4**

| Data type | key word | Size (in bits) | Range |
|---|---|---|---|
| Character | char | 8 | 0 to 225 |
| Integer | int | 16 | -32768 to 32767 |
| | short int | 16 | -32768 to +32767 |
| | long int | 32 | -2, 147, 483, 648 to +2,147, 483, 647 |
| | unsigned int | 16 | 0 to 65535 |
| | unsigned long int | 32 | 0 to 4, 294, 967, 295 |
| Float | float | 32 | 3.4E-38 to 3.4E+38 |
| Double | double | 64 | 1.7E-308 to 1.7E+308 |
| | long double | 80 | 3.4E-4932 to 1.7E +4932 |

## 1.8 Declaration

All variables must be declared before they are executed in the program. Variables are declared at the top ( as shown in the program structure) of the program. The general format is :

<data type> <variable name>

Example :

```
int a, b, c;

char  cl;
```

### 1.9 Keyword

The keywords (or reserve words) are the special words that have already defined in the 'C' language. The keywords can be used only for their intended purpose. The keyword cannot be used as identifier (User Defined Words) in program. 'C' language includes 32 keywords. They are shown in table-1.5.

**Table-1.5**

| | | | |
|---|---|---|---|
| auto | float | const | struct |
| break | for | continue | switch |
| case | goto | default | typedef |
| char | if | do | union |
| double | int | short | unsinged |
| else | long | singned | void |
| enum | register | sizeof | volatile |
| extern | return | static | while |

### 1.10 Operatorss

The symbols are used in the program for calculation by the computer. These symbols are called operators. The operator tells which operation to be performed. With the help of variable, constant and operator makes the expression. 'C' language includes Unary, Binary and Ternary operators.

Operators is used in the 'C' language are divided into different categories as follows:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Conditional Operators

5. Increment and Decrement Operator

6. Bitwise Operators

7. Assignment Operators

### 1.10.1 Arithmetic Operator

Arithmetic operators are used for the purpose of numerical calculations. 'C' language includes following Arithmetic operators are shown in table-1.6

**Table-1.6**

| Operator | Purpose |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Remainder (Remaining after division) |

There is no exponentiation operator in 'C'. The library function pow(x,y) is used to calculate exponentiation which is available in the header file <math.h>. It's meaning $X^y$.

### 1.10.2 Arithmetic Expression

Variables and constants are interconnected with the help of operator called expressions. If both operands are integer variable or constant then it is called Integer Expression.

Example : if a = 15 and b = 2 then result of expression will be

| Expression | Result |
|---|---|
| a + b | 17 |
| a - b | 13 |
| a * b | 30 |
| a / b | 7 |
| a % b | 1 |

The actual value of a/b is 7.5. If both operands are integer variable (or constant) then result will be in integer number. So the result of 15/2 will be 7. It is shown in figure 1.2.

Integer division

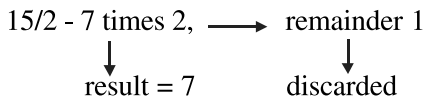



**Figure 1.2**

The modulus operator (%), complement of the division operator in that it provides a means for us to obtain the remainder after integer division. It is shown in figure 1.3.

Modulus Operation

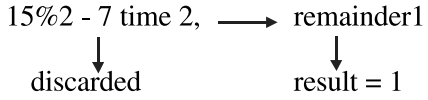



**Figure 1.3**

Other examples of modulus operator are as following :

- 15 % 2 = -1

- 15 % -2 = -1

15 % -2 = 1

If both operands are real numbers then the result of the expression will be a real number. This type of expression are called real expression. If a operand is real number and other operand is integer number the result will be real number and this expression called a mixmode expression.

Example :

if a = 15.5, b = 4.2 then

a + b = 15.5 + 4.2 = 19.700000

a * b = 15.5 * 4.2 = 65.100000

above both operands are real numbers

Example :

if a = 15.5 (real), b = 4 (integer) then

a + b = 15.5 (real) + 4(int) = 19.500000 (real)

a * b = 15.5 (real) * 4 (int) = 62.000000 (real)

above both are Mixmode expressions.

### 1.10.3 Relational Operators

Relational operators are used to compare two values. All operators are needed two operands. If it's value is true after comparison then it return 1 otherwise it returns 0 on false value. In the relational expression, both operands must have same data type. The relational operators used in 'C' language are as follows.

**Table-1.7**

| Operator | Meaning |
|---|---|
| < | less than |
| <= | less than equal to |
| = = | equal to |
| ! = | not equal to |
| > | greater than |
| > = | greater than equal to |

Example :

If p, q and r are integer variable and their values are 1,2 and 3 respectively. Result of the relational expression are given in the table.

**Table 1.8**

| Relational expression | Result | value |
|---|---|---|
| p < q | true | 1 |
| (p + q) > = r | true | 1 |
| (p + 5) = = r + 4 | False | 0 |
| r ! = 5 | true | 1 |
| p * q > q * r | False | 0 |

### 1.10.4 Logical Operator

Logical operators are also used with two operators. These operators may be logical or relational expression. The result of logical operators are true or false. 'C' language includes three logical operators.

**Table-1.9**

| Operators | Meaning |
|---|---|
| && | and |
| \|\| | or |
| ! | not |

**Logical AND (&&)**

The result of the logical operator AND (&&) will be true only if both operands are true otherwise false.

Example :

if i = 9 and j = 8.5 then the result of the following expression will be true because both expressions are true.

$$(i > 6) \&\& (j < 9.5)$$

**Logical OR ( || ) :**

The result of the logical operator OR (||) will be true if one operand is true out of two operators otherwise false.

**Logical NOT ( ! ) :**

Logical NOT is a Unary operator. There fore, it requires one operand. If operand is true then result will be false and if operand is false then result will be true.

### 1.10.5 Boolean Expression

The value of boolean expressions is true or False. A boolean expression is interconnected with boolean variable, constant and relational operators. Two boolean expressions can also be interconnected with logical operator.

Examples of logical operators are as follows :

(i) age >55 && salary <1000

(ii) number <0 && number >100

(iii) ! (status==1)

Example :

Let i=7, f=35 and c = 'W' ,here some of the complex logical expression are shown in the table.

| | | |
|---|---|---|
| **Boolean Expression** | **Result** | **Value** |
| (i >=6) && (c =='W') | True | 1 |
| (f >11) && (i >100) | False | 0 |
| C ! = 'a' \| \| (1 + f) <=10 | True | 1 |
| C = = i \| \| f > 20 | False | 0 |

### 1.10.6 Conditional Operator

This is a ternary operator, so it has three operands. This is write as follows :

variable = exp1 ? exp2 : exp3 ;

The exp1 is a boolean expression and it will evaluate first it, if exp1 is true then exp2 will evaluate and result will be stored in the variable otherwise exp3 will evaluate and result will be stored in the variable. Example :

if a = 5, b = 9

c = a > b ? a : b ;

The value of c will be 9. Because the value of expression a>b will be False and the value of exp3 will be stored in c which is b. So, the value of b will be stored in c.

### 1.10.7 Increment and decrement operators

'C' language has two very useful operators.

(i) Increment Operator - ++

(ii) Decrement Operator - --

These are unary operators. So, they required one operand. The increment operator (++) increase the value of the variable by one and decrement operator (--) decrease the value of the variable by one. These operators are written as follows :

++A or A++

and

--A or A--

If the operator is used before the operand (eg ++a), then the operand would be add or subtract one from the variable before it is utilized for its intended purpose within the program. If the operator follows the operand (eg a++) then the value of the operand would be add or subtract one from the variable after it utilized.

### 1.10.8 Assignment Operators

Assignment operators are used to assign the result of an expression to a variable. This is shown by '=' sign. This operator stores the value of the right hand side expression after calculation into the left hand side variable. The general format are as follows :

variable = Expression

Example :

These are the example of the assignment operator :

area = 2 * P1 * r1

x = y

a = 7.52

Total = A[1] + A[2]

In the above assignment statements, the left hand side must have only one variable (cannot be constant and expression).

'C' language also includes special assignment operator, these are as follows :

+=, *=, %=, -=, /=

Example :

x = x + 4 writes as x += 4

x = x * 5 writes as x *= 5

x = x % 4 writes as  x %= 4

x = x - 5  writes as  x -= 5

x = x / 4  writes as  x /= 4

### 1.10.9 Type Conversion of expression

These are used to change the data type of the result of the expression. This writes as follows :

(Data type) Expression;

(datatype) expression, the value of the expression will be converted into the data type which is written in the bracket.

If a expression have more than one operand then the result of the expression automatically converted into higher precision data type, presented in the expression. This type of casting is called implicit type casting.

Example :

If a = 7 (integer) b = 9.5 (float) then the result of 2a + b will be 23.500000

**Program 3**

```
/* Program to convert a centimeter into meter-centimeter with integer Arithmetic*/

#include<stdio.h>

main()

{

    int cent,meter;

    printf("Enter the value = ");

    scanf("%d",&cent);

    meter = cent / 100;

    cent = cent % 100;

    printf("\nMeters = %d and Centimeter = %d",meter,cent);
```

```
}
```

Enter the value = 1056

Meters = 10 and Centimeter = 56

**Program 4**

```
/* Program to explain explicit and implicit conversion */

#include<stdio.h>

main()

{

        int A=7,B=8;

        float X,Y=7.5;

        X = A + Y; /*Implicit Type Conversion*/

        printf("X = %f\n",X);

        X = (float)A/B; /*Explicit Type Convesion*/

        printf("X = %f\n",X);

}
```

Output:

X = 14.500000

X = 0.875000

### 1.11 Precedence of Operators

If a expression has more than one operator then it is necessary to know how they will execute in a sequence. It is predefined which operator will execute first. The precedence of the operator are predefined. The sequence in which same group operators in an expression are executed is determined by the associativity of the operator.

**Table-1.10 :Precedence of Operators and associativity rules**

| Precedence group | Operator | Associativity |
|---|---|---|
| Function,array, | | |
| struct member and pointer | () [ ] .→ | L→R |
| Unary Operator | - + + — ! ~ * & sizeof(type) | R→L |
| Arithmetic multiplication, | | |
| division and remainder | * / % | L→R |
| Arithmetic add and subtract | + - | L→R |
| Bitwise shift operator | << >> | L→R |
| Relational Operator | < <= > >= | L→R |
| Equality Operators | == != | L→R |
| Bitwise AND | & | L→R |
| Bitwise XOR | ^ | L→R |
| Bitwise OR | \| | L→R |
| Logical AND | & & | L→R |
| Logical OR | \| \| | L→R |
| Conditional Operators | ? : | R→L |
| Assignment Operators | = += -= *= /= %= &= ^= \|= <<= >>= | R→L |
| Comma Operators | , | L→R |

Example :

If X = 7 y = 3.0 Z = 2 A = 2.5 B = 7 then solve the following expression

X + Y / (Z * A + B / Z)

Solution :

X + Y / (Z * A + B/Z)

= 7 + 3.0 / (2 * 2.5 + 7 / 2)

= 7 + 3.0 / (5.0 + 7/2)

= 7 + 3.0 / (5.0 + 3)

= 7 + 3.0 / 8.0

= 7 + 0.375

= 7.375

**Program 5**

```
/* Program to calculate a expression */

#include<stdio.h>

main()

{

        float A,B=2.5,D=9.25;

        int i = 5,j=10;

        A = (float)i/j + (B*2)/(i + j) + D;

        printf("Result = %f\n",A);

 }
```

Output:

A = 10.083333

### 1.12 Input-Output Function

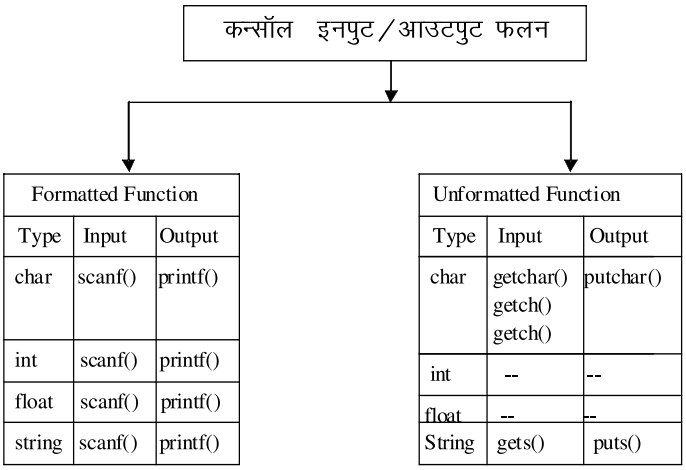
Generally, a program do three work read data, process on data and gives output. Program uses functions to take input and to produce output. These functions are scanf( ), printf( ), getchar( ), putchar( ). The input/output are of two types formatted and unformatted. The function printf( ) and scanf( ) are formatted and putchar( ) and getchar( ) are unformatted I/O statement.

### 1.12.1 Console Input/Output

Console input/output functions are divided into two categories.

1. Unformatted console input/output functions
2. Formatted console input/output functions

The main difference between these two type of I/O statement is, the formatted console input takes the input from keyboard and print the output on the VDU according to the requirement of the user. But unformatted statement takes the data and print in normal form. For example we wants to print Total_item and sale_item on VDU. Where does it print on VDU and how much space is required between two data? This can be done using formatted console input/output statements. Both types of functions are shown in the figure given below.




| Formatted Function | | |
|---|---|---|
| Type | Input | Output |
| char | scanf() | printf() |
| int | scanf() | printf() |
| float | scanf() | printf() |
| string | scanf() | printf() |

| Unformatted Function | | |
|---|---|---|
| Type | Input | Output |
| char | getchar()<br>getch()<br>getch() | putchar() |
| int | -- | -- |
| float | -- | -- |
| String | gets() | puts() |

Fig. 1.4

### 1.12.2 Output Function printf()

The output function sends the values to output device or output file from the memory. The printf() function in the 'C' language sends data to output device from the memory. The printf() function write as follows :

printf("Control string", variable1,variable2,variable3);

The variable1,variable2,variable3 are variable name. The values of these variables will send to the output device. The control string writes between double quote (" "). The control string writes according to the data types to be printed. For example %d used for integer data type. The formats of the control string are shown in table-1.11.

**Table-1.11**

| Format | Data type of variable |
|---|---|
| %d | int (Decimal) |
| %x | int (Hexa Decimal) |
| %o | int (Octal) |
| %h | short int |
| %u | unsigned int |
| %ld | long int |
| %f | float |
| %e | float (Exponent) |
| %lf | double float |
| %Lf | long double |
| %c | character |
| %s | string |

Example :

A = 20, B = 30

printf("%d %d",A,B)

Output : 20 30

A and B are integer type variables and according to control string it print the value of A and B variables. The printf( ) function also prints a same string which is written in the double quotation.

Example :

printf ("C is a good programming language.")

Output : C is a good programming language.

### 1.12.3 Formatted printf( ) function

When printf( ) function print the value of the variables in the format form then we

have to place some extra character in the control string.

**Output of integer numbers**

The format specification for printing an integer number is :

%wd

Where w is an integer value that specifies the total number of column for the output.

Example :

```
int A=1476

main()

{

    printf("%7d",A);

}
```

The output of the above program is :


This output will be written in seven columns but number will arrange in four columns. So, first three columns will be empty.

**Output of real numbers**

The format specification for printing a real number is :

%w.pf

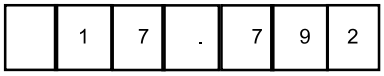
Where w and p are two integer value and w specifies the total number of column for the output and p is the total number of column after decimal point.

Example

```
float x = 17.7927 ;

main( )

{

    printf("7.3f", x) ;
```

}

The output of the above program is :




This output will be written in seven columns. It will use three columns after decimal point, one for decimal point and two for number written before the decimal point. One column will be blank because it uses 6 column. Which is shown in the output.

**Output of strings**

The format specification for printing a string is :

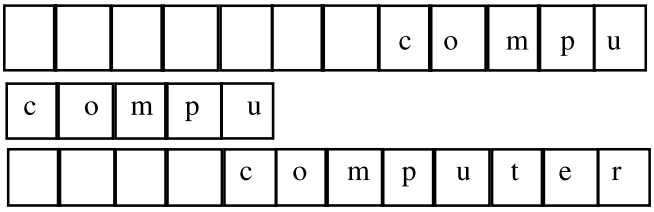
%w.ps

Where w and p are two integer values and w specifies the total number of column for the output and p is the total number of characters to be printed.

Example :

```
char  str[9] = "computer" :

main( )

{

        printf("\n%12.5s", str) ;

        printf("\n%.5s", str) ;

        printfl("\n%12s", str) ;

}
```

The output of the above program is :




First printf() statement will write output in 12 column and only five characters of the string will be printed. Second printf( ) statement print only five character in first five column and third printf() statement print the "computer" string in 8 column out of 12 column and four column will be empty. Which is shown in the output.

The format specification for printing a single character is :

%wc

This character leaves blank (w-1) column and print in the $w^{th}$ column.

### 1.12.4 Input function scanf( )

The scanf( ) function used to store data in the variable of the program from Input device (keyboard). The printf() function write as follows :

scanf(" Control String ",&variable1,&variable2,&variable3);

The &variable1,&variable2, and &variable3 are the address of the variables in the scanf where receive data from keyboard will stored. The ampersand sign (&) shows the address of the variable not value. The percentage sign (%) written with English alphabet in the control string as same in the printf() function. These are shown in the table-1.11.

Example :

scanf("%d%d%d", &NUM1, &NUM2, &NUM3) ;

NUM1, NUM2 and NUM3 are integer variable. All variable have to write different control string for different variable.

**Program 6 :** Write a 'C' language program to read three real numbers and findout the smallest number.

```
#include<stdio.h>

/* C program to read three float number and find smallest number.*/

main()

{

    float a,b,c,small;       /*Declaration of float numbers*/

    printf("Enter three float Numbers :\n");

    scanf("%f %f %f",&a,&b,&c);    /*read three float numbers*/

    if(a < b)

      if(a < c)

            small = a;
```

```
        else
                small = c;
    else
        if(b < c)
                small = b;
        else
            small = c;
    printf("\nThe Smallest Number = %f",small);
}
```

Input :Enter three float Numbers :

78.23 78.96 85.52

Output:The Smallest Numbers : 85.52

Input :Enter three float Numbers :

23.23 23.15 23.16

Output:The Smallest Numbers : 23.15

**getchar( ) function**

This function takes a character from the input device and gives to the computer. The general format is :

Variable Name = getchar( );

Example :

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

In the above example, the c variable define as char type. When second statement

c=getchar( ) is executed then computer waits for pressing a key on the keyboard(if data is not in a buffer) and store in the variable c. So, c = getchar( ) and scanf(" %c", &c) both are equavlent statement.

**putchar( ) Function**

This function puts a character from the memory to the output device. The general format is :

putchar(Name of char Variable);

The character variable defined previously which will print on the monitor.

Example :

```
char a;
. . . . .
. . . . .
putchar(a)
```

In the above example, a variable defines as char type through the first statement. And second statement putchar(a) send a charater to the output unit. The type of variable must be char. So, putchar(a) and printf("%c", c) both are equivalent statement.

**Program 7 :** Write a 'C' language program to read a line and print it in the uppercase.

```
#include<stdio.h>
/* C program to read a line and print it into upper case.*/
main()
{
    char c[80];/*Declaration of char array or string*/
    int i;
    printf("Enter A line:\n");
    for(i=0;(c[i]=getchar()) != '\n';i++);/*This statement read a
                                            line*/
```

```
    c[i]='\0'; /* Store NULL char at end*/
    printf("The UpperCase line is:\n");
    for(i=0;c[i]!='\0';i++)
    {
      c[i] = toupper(c[i]);/*Convert into the UpperCase Letter*/
      putchar(c[i]);  /*To print a Character on the screen*/
    }
}
```

Input :Enter a line :

The string is a combination of character.

Output:The UpperCase line is:

- THE STRING IS A COMBINATION OF CHARACTER.

### 1.13 Control Statements

In the 'C' language, a statements executes in the sequential fashion in which sequence they are written in the program. Each statement is executed once and once only. But a logical condition decides which statement will execute and which will not execute in the program. The if and switch statements can fulfil the above requirement of 'C' language.

Some of the statements will be executed until the logical condition is true. It is called looping.

The control statements execute the program statement in the predefine sequence. These are the following control statements in the 'C' language :

1. Decision Making Statement

   (a) if Statement (b) if else statement (c) switch statement

2. Loop statement

   (a) for loop (b) while loop (c) do while loop

3. Other control statement

(a) break (b) continue (c) goto

## 1.14 if statement

It is a powerful control statement which first check the condition than after it execute other statements according to result of the condition. The if statement are following four types.

(i) Simple if statement

(ii) if else statement

(iii) Nested if else statement

(iv) else if stairs

### 1.14.1 Simple if statement

First, it checks the condition in the statement. If the condition is True then it executes the group of statement otherwise it does not. One or more statement can be in the group of statement. The general format of the if statement is as following :

```
if (Condition)
{
   Group of statement
}
```

If control statement has one statement in the group of statement then there is no need to close within curly bracket. As shown in the example.

(a)
```
if (a > 10)
    printf("a is greater than 10");
```

(b)
```
if (a>10|| b = = 5)
    printf("a is greater than10 or equal to 5");
```

(c)
```
if (n%2 ==0)
    printf("N is a even number");
```

### 1.14.2 if-else statement

The control flow is bi-directional in this statement. First, it checks the condition. If the condition is true then it executes first group of statements. If it is false then it executes second group of statement. if-else statement will be written as follows :

```
if (Condition 1)
{
        Statement Group 1
}
else
{
        Statement Group 2
}
```

Examples of if else statements are as follows:

(a)
```
if (a > b)
        printf("a is greater than b");
else
        printf("b is greater than a");
```

(b)
```
if (n%2 = = 0)
        printf("N is Even Number") ;
else
        printf("N is odd Number") ;
```

(c)
```
if (Basic > 50000)
{
        HRA = BASIC * 0.15 ;
        DA = BAISC * 0.61 ;
```

```
}
else
{
        HRA = BASIC * 0.10
        DA  = BASIC * 0.61 ;
}
```

The example (a) and (b) include the single statement so, there is no need to close with in curly bracket. But in the example (c) close the curly bracket due to two statements.

**Program 8 :** Write a 'C' language program to find the given year is leap year or not.

```
#include<stdio.h>
#include<conio.h>
/*It is C program to check for leap Year*/
main()
{
    int year,leap;
    printf("\nEnter the year :");
    scanf("%d",&year);
    if(year % 100 == 0) /* Checking for century*/
        if(year % 400 == 0)/* Century is leap year*/
            printf("It is century and leap year");
        else
            printf("It is century but not leap year");
    else if(year % 4 == 0) /* Checking for leap year*/
            printf("It is leap year");
         else
```

```
            printf("It is not leap year");
    getch();
}
```

Input/Output:

Enter the year :2000

It is century and leap year

Enter the year :2001

It is not leap year

Enter the year :2004

It is leap year

Enter the year :1900

It is century but not leap year

### 1.14.3 Nested if else statement :

When an if else statement written within another if-else statement then it is called nested if else.

```
if (Condition 1)
{
   if (Condition 2)
   {
     Statement group1
   }
   else
   {
     Statement group 2
   }
```

```
   Statement group 3
}
else
{
  Statement group 4
}
```

First, it check the condition in the above control statement. If it is False then it starts to execute statement group 4. It left all the statements groups. If the condition 1 is true then condition 2 will be checked. If condition 2 is true then it execute statement group 1 otherwise on false it execute statement group 2. Then after control flow execute the statement group 3 and leave the if else statement and it will not execute statement group 4.

Example :

```
if ( x > = 40)
{
     if (x > = 65)
     {
          printf("FIRST DIVISION") ;
     }
     else
     {
          printf("SECOND DIVISION");
     }
}
else
{
```

```
        printf("FAIL") ;
    }
```

### 1.14.4 else if stairs

This type helps to take decision on multi-way statement. This statement is used when the multiple condition is checked one by one. This is written as follows :

```
if (Condition 1)
{
    Statement Group 1
}
else if (Condition 2)
    {
        Statement Group 2
    }
    else if (Condition 3)
        {
            Statement Group 3
        }
        else
        {
            Statement Group 4
        }
```

### 1.14.5 switch statement

This is a multi-way conditional statement. It also have a condition similar to if else statement. Different statements will be executed according to expression or variable in the switch statement. The value of expression or variable must be integer or character. The format is as follows :

```
switch (Expression or variable)
{
        case 1      :    Statement group 1
                         break ;
        case 2      :    Statement group 2
                         break :
        case n      :    Statement group n
                         break ;
        default     :    Statement group
}
```

Here switch, case, break and default are keywords. First of all, switch statement checks the value of expression or variable. It compares with the constant written with case keyword. If this value find equal to any case constant then it executes the statement group related to the case statement and it will be out from the switch statement after executing break statement otherwise it will execute all case statements. If the value of expression or variable does not match with any case value then it executes the default statement group. If default statement does not present in the switch statement then the control flow come out from the switch statement without executing any statement group if any case value does not match.

Example :

```
        switch (N) /* N is an integer value * /
        {
        case 1 : printf("ONE") ;
                         break ;
        case 2 : printf("TWO") ;
                         break ;
        case 3 : printf("THREE") ;
```

```
                    break ;
case 4 : printf("FOUR") ;
                    break ;
case 5 : printf("FIVE") ;
                    break ;
default : printf("Enter Between 1-5")
}
```

In the above example, it will print the numbers in words. The value of N compares with constant (1, 2, 3, 4 and 5). If there is any value matches with case constant then it prints a number in word otherwise it executes the default statement and control flow come out from the switch statement.

**Program 9 :** Write a 'C' program to print the grade of the students. Grades are as follows:

| | |
|---|---|
| 100 - 90 | A + |
| 89 - 80 | A |
| 79 - 70 | B + |
| 69 - 60 | B |
| 56 - 50 | C |
| 50 - 0 | Fail (using switch statement) |

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
/* It is C program to Use switch Statement.*/
main()
{
    int marks,Rollno;
    char Name[20],grade[5];
```

```
        clrscr();
        printf("Enter The Name of a Student     : ");
        gets(Name); /*Library function to read a string*/
        printf("Enter The Roll No. of a Student : ");
        scanf("%d",&Rollno);
        printf("Enter The Marks Obtained        : ");
        scanf("%d",&marks);
        switch(marks/10)
        {
            case 10:
            case  9: strcpy(grade,"A+");break;
            case  8: strcpy(grade,"A");break;
            case  7: strcpy(grade,"B+");break;
            case  6: strcpy(grade,"B");break;
            case  5: strcpy(grade,"C");break;
            default: strcpy(grade,"Fail");
        }
        printf("\nName of a Student          : %s",Name);
        printf("\nThe Roll No. of a Student : %d",Rollno);
        printf("\nThe Marks Obtained         : %d",marks);
        printf("\nThe Grade Obtained         : %s",grade);
getch();
}
```

Result:

Input:

Enter The Name of a Student : Sunil Methi

Enter The Roll No. of a Student : 8542

Enter The Marks Obtained : 85

Output:

Name of a Student : Sunil Methi

The Roll No. of a Student : 8542

The Marks Obtained : 85

The Grade Obtained : A

Input:

Enter The Name of a Student : Karishma Methi

Enter The Roll No. of a Student : 8645

Enter The Marks Obtained : 45

Output:

Name of a Student : Karishma Methi

The Roll No. of a Student : 8645

The Marks Obtained : 45

The Grade Obtained : Fail

## 1.15 Looping

Besides of decision-making statements, some statements are required to execute more than once in the program. When some statements execute again and again in the same order then this processing is called looping. for, while and do-while are three loop statement in the 'C' language. These all statements have two parts:

(i) Looping body

(ii) Condition

It checks the control statement in the loop and if this condition is true then loop

will be executed otherwise control flow will come out from the loop.

### 1.15.1 for loop

The general format of for statement is :

```
for (exp1 ; exp2 ; exp3)
{
        Loop body
}
```

Where exp1 : This tell the initial value of the counter variable and the initializing expression is evaluated once only, at the beginning of the for loop.

exp2 : This is a boolean expression. If it is true then loop will executing continuously.

exp3 : Increment/ decrement statement is used to increase/decrease the value of counter variable.

Example :

(a)
```
for (i = 0 ; i < 10 ; i + +)
        printf("%d", i) ;
```

(b)
```
for (j = 1 ; j < = 20 ; j + = 2)
{
        printf("%d\n",j) ;
        SUM = SUM + j ;
{
printf("SUM = %d", SUM);
```

(c)
```
for (j = 1, i = 10; j < i ; j ++, i - -)
        printf("%d %d", j, i);
```

**Program 10 :** Write a program to add 10 numbers.

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

```
#include<conio.h>
main()
{
    int N,sum=0,i;/*Sum initialize by zero at declaration*/
    for(i=1; i<=10;i++)
    {
        printf("Enter %d  Number :",i);
        scanf("%d",&N);
        sum += N; /*short hand assignment operator.*/
    }
    printf("\nSum of ten integer Number = %d",sum);
    getch();
}
```

Input/Output:

Enter 1 Number :23

Enter 2 Number :47

Enter 3 Number :98

Enter 4 Number :45

Enter 5 Number :85

Enter 6 Number :93

Enter 7 Number :123

Enter 8 Number :243

Enter 9 Number :24

Enter 10 Number :45

Sum of ten integer Number = 826

### 1.15.2 while loop

It is not known in advance, how many times a loop will be executed and it is depends on a condition. When the condition is true then the statements will be executed continuously. It written as follows :

```
while (Condition)
{
        Loop body
}
```

Example :

```
(a)  while (i < 10) / * initial value of i is one * /
     {
     printf("%d", i) ;
     i + + ;
     }
(b)  while (j < = 20) / * initial value of j is one * /
     {
          printf("%d", j) ;
          SUM = SUM + j ;
          j + = 2 ;
     }
     printf("SUM = %d", SUM) ;
```

**Program 11 :** Write a program to add the digit of the given number.

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

```
/*It is C program to Add the digit of a given number*/
/*e.g. 12345 ,sum = 1 + 2 + 3 + 4 + 5 */
main()
{
    int m,N,sum=0;/* Sum initialize by zero at declaration*/
    printf("Enter a Number :");
    scanf("%d",&N);
    while(N != 0)
    {
        m = N % 10;
        sum = sum + m;
        N = N / 10;
    }
    printf("\nSum of digits = %d",sum);
    getch();
}
```

Input/Output:

```
Enter a Number :24563
Sum of digits = 20
```

### 1.15.3 do while loop

The do-while loop is similar to the while loop. First it check the condition in the while loop then after it executes the loop body. Whenever in the do-while loop, first, it executes the loop body then after it checks the condition. So, the do-while loop statement is always executed at least once. The general format of the do - while loop is :

```
do
{
```

```
                    Loop body
                } while (Condition);
```

Example :

```
(a)  do
     {
          printf("%d", i) ; / * /initial value of i is one */
          i + + ;
     } while (i < 10) ;
(b)  do  / * initial value of j is one * /
     {
          printf("%d", j) ;
          SUM + = j ;
          j + = 2 ;
     }    while (j < = 20) ;
```

**Program 12 :** Write a 'C' program to reverse a given number.

```
#include<stdio.h>
#include<conio.h>
/*It is C program to reverse a given number*/
/*e.g. 12345 print as 54321*/
main()
{
     int m,N,rev=0;
     printf("Enter a Number :");
     scanf("%d",&N);
```

```
    do
    {
        m = N % 10;
        rev = rev * 10 + m;
        N = N / 10;
    }while(N != 0);
    printf("\nReverse Number is = %d",rev);
    getch();
    return 0;
}
```

Input/Output:

Enter a Number :32512

Reverse Number is = 21523

### 1.15.4 Nested Loop

When a loop run in side the another loop then it is called nested loop. For example : for loop inside the other for loop. It must be remembered that the inside loop must close first and after this outer loop will close. As shown in the figure.

```
for (          )
{
        for (        )
        {
        {
}
```

Example :

(a) for (i = 0 ; i < 10 ; i ++)

```
        for (j = i; j < 10 ; j ++)
            printf("%d", i) ;
    printf statement print 55 integers.
(b) for (i = 1 ; i < 10; i ++)
    {   j = 1 ;
        while (j < = 10)
        {
            printf("%d", i*j) ;
            j++ ;
        }
        printf("\n") ;
    }
```

The above loop will print 1 to 10 tables.

### 1.15.5 break statement

Generally, break statement is used with loops or switch statement. It is used to terminate a loop. If break statement executes in nested loop then the control flow will come out from the loop where the break statement is be executed.

Example :

```
(a) for (i = 1 ; i < = 10 ; i + +)
    {
        scanf("%d", &N) ;
        SUM = SUM + N ;
        if (SUM> = 200)
            break ;
    }
```

The loop can terminate in two way first, the value of i reaches to 10 or SUM exceeds 200. When the value of SUM exceed 200 then break statement will execute and it terminate the loop.

### 1.15.6 continue statement

The break and continue, both statements are used to stop execution. But break statement stop the execution of the loop and control flow come out from the loop. The loop does not terminate when continue statement is encounter then the remaining loop statements are skipped and computation proceeds directly start the next pass of the loop. It Is written as:

Example :

```
do
{
    scanf("%d", &N) ;
    it (N < 0)
    {
        printf("Error") ;
        continue ;
    }
    / * other statements * /
}  while (N < = 50) ;
```

In the above example, If N read a negative value then continue statement will be executed and it cause to start new pass of the loop.

**Important Points**

1. The semicolon separates the statements. This is indication of end of statement. The semicolon does not attach after the pre-processor directives.

2. 'C' language is a case sensitive language. It means, uppercase letters are different from lowercase letters.

3. The printf( ) function sends the message to the output devices.

4. The scanf( ) function also calls a function which takes the input from input device.

5. In the 'C' language, all 32 keywords are in lower case letter.

6. 'C' language is a operator rich language. It includes arithmetic, relational, logical, bitwise etc. operators.

7. The precedence and associativility rules are applicable to solve a expression.

8. The main( ) function is also a user define function and it must be presented in the program.

9. scanf(), getchar(), getch(), getche() and gets() are input function.

10. printf(), putchar() and puts() are output function.

11. The decision statements if, if-else and switch are used in the 'C' language.

12. The part of the program can be executed more than once by looping. The for, while and do while loop are used in the 'C' language.

13. The break statement is used to stop execution of switch and loop and control flow comes out from the switch or loop.

14. The continue statement is used to stop execution part of the loop and start new iteration.

## Exercises

### Objective Type Questions

1. if(1)

   printf("TRUE")

   else

   printf("FALSE")

   (a) TRUE (b) FALSE

   (c) satement is wrong (d) None of the above

2. Who was developed the 'C' language?

   (a) Ken Thomson (b) Dennis Rechie

   (c) Martin Recharson (d) Donavon

3. The symbol used to seperate the two statement in the 'C' language.

(a) & (b) !

(c) ; (d) “

4. Which must be presented in the 'C' language program?

(a) Global variable (b) main() function

(c) Pointer (d) Function

5. The meaning and uses are predefined, these words are called.

(a) Variable (b) Constant

(c) Identifier (d) Reserve word

6. Which is not a reserve word?

(a) auto (b) while

(c) switch (d) total

7. The valid constant is :

(a) 4, 78 (b) OX 23 A

(c) 177A (d) 5.7, 632

8. Which is not a valid identifier?

(a) 5XYZ (b) total_subject

(c) Stud_mark (d) XSXY

9. The control string for long double data type in the printf statement.

(a) %d (b) %lf

(c) %ld (d) %Lf

10. The purpose of % (percentage) operator is

(a) Addition (b) Division

(c) Remainder (d) Multiplication

11. If i = 8 and b = --i + 3 then the value of b.

(a) 11 (b) 10

(c) 9 (d) 12

12. Which statement is used to stop the execution of the loop :

(a) break (b) stop

(c) end (d) None of these

13. for (i=0;i<=5;i++)

```
        for(j=i;j<=5;j++)
            printf("India");
```

In the above program, How many times will print India.

(a) 6 (b) 21

(c) 36 (d) 30

14. In which statement a case keyword is used-

(a) switch (b) for

(c) do while (d) while

15. Which loop will first terminate in the nested looping :

(a) Outer loop (b) Inner loop

(c) Both at once (d) None of these

**Very Short Type Questions**

1. What is reserve word?
2.. How many types are arguments are there?
3. How many times main( ) function can be used in the 'C' language?
4. What is use of ++ operator.
5. What do you mean by getchar( ).
6. Which function is used for giving output?
7. In which structure the "default" statement is used?
8. How does nested if else statement written?
9. Where is continue statement used?
10. How many loop structures are presented in the 'C' Language?
11. Explain do while structure with the help of an example.

**Short Type Questions**

1. What is the meaning of algorithm?
2. What is constant?
3. How does declare a variable?
4. Differentiate between string constant and character constant.
5. What is identifier?
6. Explain ternary operator.

7. What is precedence of the operators? Explain.
8. When does it require associativility rules for operators?
9. Why does 'C' language call middle level language?
10. What is nested loop?
11. Explain do-while statement.
12. Write all types of the if statement.
13. How does switch statement writes, write format.
14. Differentiate between continue and break.
15. Write the output of the following program :

(a)
```
#include<stdio.h>
main ()
{
    int i = 1, x = 1 ;
    for (i = 1 ; i < 10, i++)
    {
        printf("%d\n", x+i);
        x = x + 1;
    }
}
```

(b)
```
# include <stdio.h>
# define p 10
{
    int k =1, w = p;
    while (k <=w)
    {
        printf("%d\n",k);
    }
}
```

**Essay Type Questions**

1. Explain the structure of the 'C' language program.
2. What is meaning of program structure?
3. How many type of data types are there in C language?
4. Explain the input/output statements used in the 'C' language.
5. What do you understand by precedence and associativity of the operators?
6. Explain all input function with the help of an example.
7. How does you get the output from the output function? Explain with the help of an example.
8. Convert the following expression into 'C' equivalent expression.

   (i)

   (ii) $a^2 + b^2$

   (iii)

   (iv) $1 + \frac{a}{b + 1/c}$

$\frac{ab}{c} + \frac{b^2}{c} + d$

9. Write a program to read three numbers for finding largest number with using conditional operator.
10. Which statements are used for looping? Explain.
11. Write a 'C' program for printing all odd numbers from 1 to 50.
12. Differentiate while loop and do-while loop with the help of examples.
13. Write a program to findout the sum of the following series.

    $1 + x + x^2 + x^3 \ldots\ldots\ldots\ldots x^n$

**Answer Key**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1. a | 2. b | 3. c | 4. b | 5. d | 6. d | 7. b | 8. a | 9.d |
| 10.c | 11. b | 12. a | 13. b | 14. a | 15. b | | | |