Notes on "C" Programming

Basic Model of Computation

To solve any kind of problem we need to write a step-by-step process of solution using simple instruction to obtain the result. There would be numbers of methods to solve a problem and solution may differ from person to person.

Example: If you have to inform a person with a distance of 10 KM then how will you do it?

Person 1: Tele Calling to him

Person 2: Messaging to him

Person 3: Physical visit to him

And so on…..

The Basic Steps of Computation:

a. Formulating the Problem and decide the data-types to be input.
b. Identifying the step of computation that is necessary for getting the solution.
c. Identifying the decision point, under what circumstances a particular operation to be performed and when not to be performed.
d. Knowing the expected results and verifying with the actual values.

Procedure for Problem Solving

To solve a problem first we have to break down the problem into small parts which can be solved step by step to arrive at the final solution. It need not involve the computer. Every problem is unique by itself, but some steps are common. They are as follows…

1. Spend some time to understand the problem thoroughly. In this step we do not need a computer. We need to try to answer what is expected & how to get it.
2. Construct a list of variables that are needed to get the solution of the problem.
3. Decide the layout for the output presentation.
4. Select the programming technique which is best suited to solve the problem. Then carryout a coding using a programming language.
5. Test the Program. In this step we test the each part of the program whether it is functioning properly or not.
6. Validation of the program. This step guards the program against wrong data processing. Here we check the program by input wrong data to it. If wrong data is processed then there is an error else program is correct.

Algorithms

It is a set of instruction which describe the steps to be followed to carry out an activity. If it is written in the computer programming language then it is called a program. It tells us how to go about for getting the solution of a problem.

Example:

Making a cup of Tea

1. Begin
2. Take the suspense
3. Turn on The Gas
4. Put ignition to the Gas
5. Add some water in the suspense
6. Add some tea powder & Sugar

7. Boil it upto 5 min
8. Turn off the Gas
9. Filter the Tea
10. Serve it into the cup
11. End

Identify whether a number is odd or even

1. Begin
2. Input a Number (Read a No)
3. Divide the Number by 2
4. If the remainder is 0, then No is even
5. Else The No is Odd
6. End

Calculate the Total of serial no upto 9

1. Begin
2. Read a number 9
3. Take a variable x and assign 1 to it
4. Take another variable as sum
5. Add the value of x to sum
6. Add 1 to x
7. check if x>=9 then display the value of sum
8. else repeat the step 5 & 6
9. end

Flow Chart

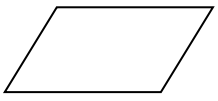
A Flow chart is a graphical representation of the sequence of operations for the solution of a problem. Flow chart shows the sequence of instruction written in a single program or subroutine.

It uses boxes and lines of different shapes to denote different types of instructions. The instructions are written in these boxes in clear and concise manner. These boxes are connected with solid lines with arrow head, these lines are called flow line. It denotes the flow or actual sequence of the program. Though the flow chart shows a program in a graphical form, logical errors can be detected very easily.

Once a flowchart is ready, the making a programming code in any programming language becomes very easy.

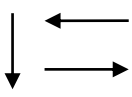
Flow Chart Symbols

Terminal: - It is used to indicate the Starting (Begin) or Ending (End) of a Program. It is the first symbol & also the last symbol in the flowchart.

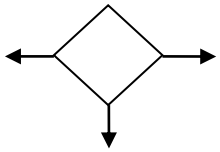


Input / Output: - It is used to indicate the operation of input and output of the program. If there is a program input from a Disk, CD, Pen Drive or any other input devices then that step will be indicated in this box. Also an output to display on the monitor is also represented here.

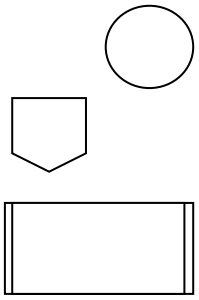
Processing: - it is used to represent the process of a program. An arithmetic instruction, Data movement in a program, addition, subtraction, multiplication, division type operational instructions are represented here.



Flow Lines: - A Line with arrow head is called flow line which is used to indicate the sequential flow of instruction in a program. These flow lines indicates a data flow of left to right, right to left and top to bottom only. Bottom to top flow line is not used in flow chart.



Decision box: -It is also called as Diamond box. It is used to indicate a point at which a decision has to be made and to branch to one of the alternatives.



Connectors: - If a flow chart becomes long and the flow lines got crisscross the it need another area or paper to make a prefect flowchart. To join these two area or paper we need connector. A circle connector is used to connect two part of a program written in a single paper. A pentagon shape is used to connect two part of a program written in two different papers.

Predefined Process (Function):- Used to represent a group of statements performing one processing task.

Algorithm to Identify smallest number among 7 numbers

12,5,8,14,17,34,3

1. Start
2. Take 7 variables a,b,c,d,e,f,g.
3. Read the 7 variables
4. Compare a < b, if true
5. Compare a < c if true
6. Compare a < d if true
7. Compare a < e if true
8. Compare a < f if true
9. Compare a < g if true
10. a is smallest
11. else compare b < c if true
12. Compare b < d if true
13. Compare b < e if true
14. Compare b < f if true
15. Compare b < g if true
16. b is smallest
17. else compare c < d if true

so on …..

30. compare f < g
31. f is smallest
32. Else g is smallest

Algorithm to Check multiple of a number

1. Start
2. Take variables x, n & count
3. Read the variable x, n as 0 & count as 0
4. Check x > n, if true
5. Add 1 to n
6. Divide x by n
7. Check x is divisible by n, if true
8. Add 1 to count

9. Repeat the step 4, 5, 6 & 7
10. If false, repeat the step 4, 5, 6 & 7
11. If false, display Count
12. end

Algorithm to check whether No is Prime or Not

1. Start
2. Take variables x, n & count
3. Read the variable x, n as 0 & count as 0
4. Check x > n, if true
5. Add 1 to n
6. Divide x by n
7. Check x is divisible by n, if true
8. Add 1 to count
9. Repeat the step 4, 5, 6 & 7
10. If false, repeat the step 4, 5, 6 & 7
11. If false,
12. Check count<= 2, if true
13. The No is Prime
14. If false, The no is Not Prime
15. end

Symbols Used in C Programming

;	Semi Colon
.	Dot
,	Coma
:	Colon
“	Double Quotes
‘	Single Quotes
!	Not
%	Modulo
^	zor / caret / circumflex

&	Ampersand
( )	Parenthesis
{ }	Curly Braces
[ ]	Square Braces
$	Dollar
#	Hash
/	Forward Slash
\	Back Slash
*	Asterisk
|	Bar
_	Underscore
-	Hyphen
~	Tilde
`	acute, backtick, grave, grave accent, left quote, open quote, or a push

What is a variable?

In C Programming Variable is a storage location in Main memory (RAM) of computer associated by the "C" Compiler to manipulate different types of data. Variable holds data that can be modified during program execution.

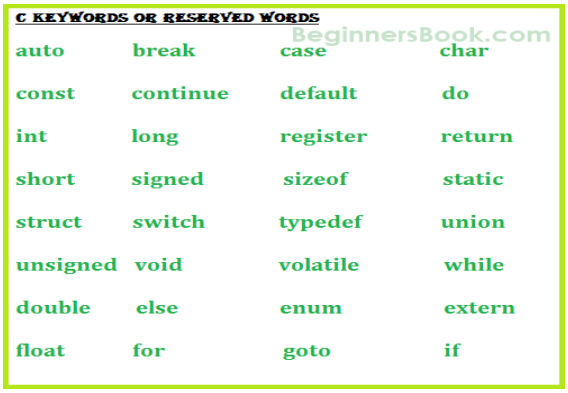
Assigning Name to a variable

- We can use A to Z, a to z, 0 to 9 and underscore ("_") to give name to a variable.
- No other special characters should be used.
- The first character of the variable should always a letter or underscore. Never use number as the first character.
- Variable name in "C" Language is case sensitive. It means "num", "Num" & "NUM" are three different variable name.
- No space should be used within the variable name.

Invalid variable name: 2num, 2_sum, 3num_, account no
Valid variable name: Num, sum, num_1 & _av etc.

Keywords / Reserved words

When C Compiler is designed at that time some words are reserved to perform specific task, those words are called "Reserved Keywords". These reserved keywords should not be used as any variable or function name. If we do so then error occurred.



Data Type in “C”

The kind of data that the variable may hold in a programming language is called Data Type. There is various types of data that is to be manipulated in C programming such as Integer, Floating Point, Character, String, Real etc.

Integer Data Type (int)

In this kind of data type we store an integer number like 1, 2, 56, 237, 32767 etc.
The keyword used to denote this kind of Data Type is “int”
Data Range: -32768 to 32767
Default value of Integer variable is 0

Declaration of Integer variable

Syntax

```
Datatype    variableName;
Datatype    variableName = value;
Datatype    variableName1 = value1, variableName2 = value2, variableName3 = value3;
```

Example

```
int    num1;                          (Only Variable is declared)
int    num2 = 326;                    (variable is declared & value 326 is assigned to the variable)
int    sum = 0, num3 = 124, num4 = 332; (Multiple variable declared and assigned with its value)
```

Floating Point Data Type

In this kind of data type we store a floating point number like 1.23, 2.56, 56.0001, 237.00 etc.
The keyword used to denote this kind of Data Type is “float”
Data Range: 3.4×10^{-38} to 3.4×10^{38}
Default value of floating point variable is 0.00000

Declaration of Float variable

Syntax

```
Datatype    variableName;
Datatype   variableName = decimalvalue;
```

Example

```
float    num1;                  (Only Variable is declared)
float    num2 = 326.34;         (variable is declared & value 326.34 is assigned to the variable)
float    average = 0.00;        (variable is declared & value 0.000000 is assigned to the variable)
```

Character Data Type

In this kind of data type we store a character like a, b, c, D, E, F, 1, 6, $, &, % etc.
The keyword used to denote this kind of Data Type is “char”
Data Range: -128 to 127 (ASCII Codes data)
Default value of character variable is null

Declaration of character variable

Syntax

```
Datatype    variableName;
Datatype    variableName = ‘value’;
```

Example

```
char    Alphabet;              (Only Variable is declared)
char    First_Alphabet = ‘A’;  (variable is declared & value ‘A’ is assigned to the variable)
char    Symbol_Dollar = ‘$’;   (variable is declared & value ‘$’ is assigned to the variable)
```

String Data Type

In this kind of data type we store multiple characters like Ram, Unitty2019, Akash Kumar, 25A@ etc.
The keyword used to denote this kind of Data Type is “string”
Data Range: as per the available memory
Default value of string variable is null

Declaration of String variable

Syntax

datatype variableName[no of character to store];
datatype variableName [no of character to store]= {“value”};

Example

char Name[20]; (Only Variable is declared)
char First_Name[15] = {“Akash Kumar”}; (variable is declared & value “Akash Kumar” is assigned to the variable)
char Last_Name[15] = {“Bandichhor”}; (variable is declared & value “Bandichhor” is assigned to the variable)
string Fruit[10] = {“Apple”}

Table 3.3 Typical Range and Size of Basic Data Types

Type	Range		Bytes	Represents
	From	*To*		
char/short	−128	127	1	characters
unsigned char	0	255	1	characters
int	−32,768	32,767	2	whole numbers
unsigned int	0	65,535	2	whole numbers
long	−2,147,438,648	2,147,438,647	4	whole numbers
unsigned long	0	4,294,967,295	4	whole numbers
float	3.4×10^{-38}	3.4×10^{38}	4	fractional numbers
double	1.7×10^{-308}	1.7×10^{308}	8	fractional numbers
long double	3.4×10^{-4932}	3.4×10^{4932}	10	fractional numbers

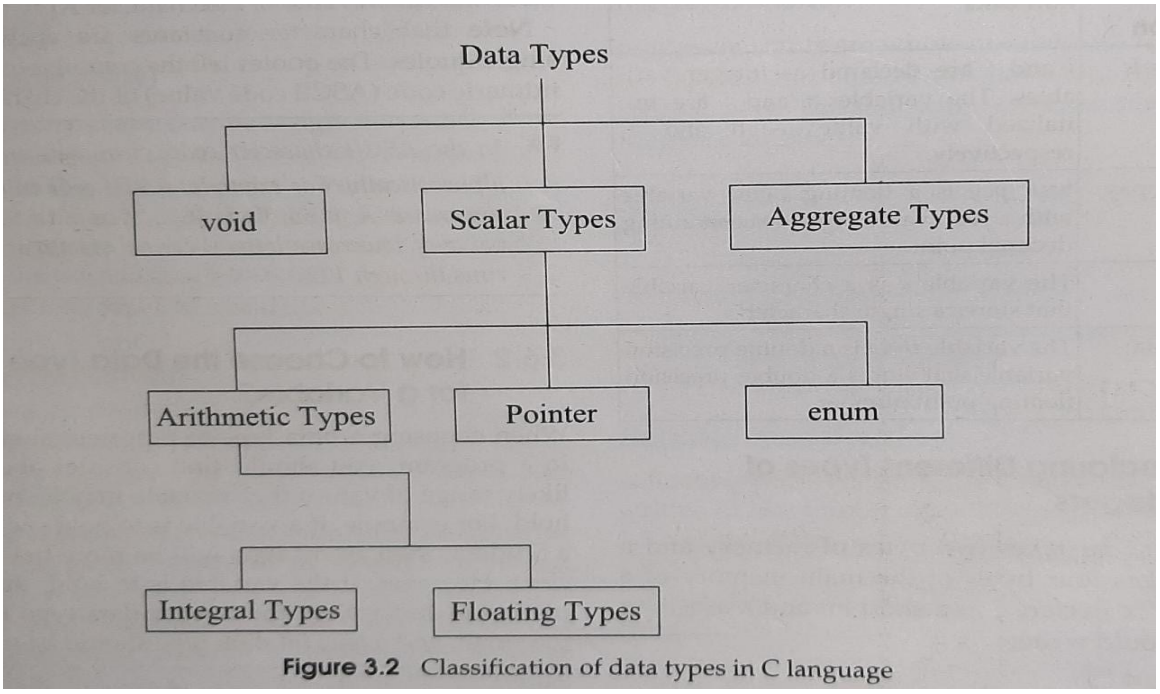


Figure 3.2 Classification of data types in C language

Declaration of Different Variables & Assigning its Values

- ✓ When we create a variable with a name than that is called "***declaration of variable"***.
- ✓ When we assign the first value to the variable than that is called "***Initialization of the variable***".
- ✓ If we do not assign any value to a variable then the default value will be "0" in case of integer & float and "null" in case of char & string variable. But it will not initiate with a value and program may not work.

Declaring Integer variable

```
DataType    VariableName;
int     num1;                   (a variable is created with a name "num1" with default garbage value 0 or 1)
int     num2, num3, num4;       ( a group of variables are created named
                                        num2 num3 & num4 with default value 0)
int      num5 = 24;             (a variable is created with a name "num5" with assigned value 24)
int     num6 = 5, num7 = 84;    ( a group of variables are created named
                                    num6 with assigned value 5 & num 7 with assigned value 84)
```

Assigning a variable separately

```
int    num8;                    (a variable is created with a name "num8" with default value 0)
num8 = 75;                      (num8 is assigned with a value 75 here)
```

Declaring Floating Point variable

```
DataType    VariableName;
float     num1;                 (a variable is created with a name "num1" with default value 0.00)
float     num2, num3, num4;       ( a group of variables are created named
                                        num2 num3 & num4 with default value 0.00)
float      num5 = 24.45;        (a variable is created with a name "num5" with assigned value 24.45)
float     num6 = 5.00, num7 = 84.67;    ( a group of variables are created named
                                        num6 with assigned value 5.00 & num 7 with assigned value 84.67)
```

Assigning a variable separately

```
float    num8;                  (a variable is created with a name "num8" with default value 0.00)
num8 = 75.34;                   (num8 is assigned with a value 75.34 here)
```

Declaring character variable

```
DataType    VariableName;
char     chr1;                  (a variable is created with a name "chr1" with default value 'null')
char     chr2, chr3, chr4;      ( a group of variables are created named
                                        chr2 chr3 & chr4 with default value 'null')
char      chr5 = 'y';           (a variable is created with a name "chr5" with assigned value 'y')
char     chr6 = 'n', chr7 = '$';    ( a group of variables are created named
                                    chr6 with assigned value 'n' & chr7 with assigned value '$')
```

Assigning a variable separately

```
char    chr8;                   (a variable is created with a name "chr8" with default value 'null')
```

chr8 = ‘P’; (chr8 is assigned with a value ‘P’ here)

Declaring string variable

DataType VariableName;

string str1; (a variable is created with a name “str1” with default value “null”)

string str2, str3, str4; (a group of variables are created named str2 str3 & str4 with default value “null”)

string str5 = “Unitty Academy”; (a variable is created with a name “str5” with assigned value “Unitty Academy”)

string m_brand1 = “Apple”, m_brand2 = “Samsung”; (a group of variables are created named m_brand1 with assigned value “Apple” & m_brand2 with assigned value “Samsung”)

Assigning a variable separately

string country; (a variable is created with a name “country” with default value “null”)

country = “India”; (country is assigned with a value “India” here)

Static Variable in C

A Static variable is initialized only once, when the program is compiled. It is never initialized again. If it is not initialized by the user then it initializes the value “0” (zero) to the variable. The static variable exists in the memory until the program termination.

Syntax:

static data_type var_name = var_value;

static int count = 0;

some interesting facts about static variables in C.

1) A static int variable remains in memory while the program is running. A normal or auto variable is destroyed when a function call and its role is over.
   For example below program prints “1 2”

Difference between Normal variable & Static Variable

Normal Variable or Auto variable	Static Variable
1. The normal variable is declared again and again when a function is run. 2. The normal variable store a garbage value between “0” and “1” randomly, if we do not initialize it, so result differs when run in different computer. 3. The value of normal variable ended when the function is ended. 4. The value of this variable is used only within its function. 5. It is a local variable.	1. The Static variable Declare only once in a program. 2. The Static variable stores “0”, if we do not initialize it, so it runs without any error. 3. The value of static variable exists until the program is ended. 4. The value of static variable is used anywhere in the program. 5. It is a global variable.

Program Example

```
#include <stdio.h>
int fun()
{
static int count = 0;
count = count + 1;
return count;
}
```

```
int main()
{
printf("%d ", fun());
printf("%d ", fun());
return 0;
}
```

Output:

1 2

Second example with normal variable

```
#include<stdio.h>
int fun()
{
int count = 0;
count++;
return count;
}

int main()
{
printf("%d ", fun());
printf("%d ", fun());
return 0;
}
```

Output

1, 1

How to Choose correct Data type for the program

For selection of data type we need to concern on the following points…

Estimate the output value of the program that it falls under which data range. (see data range table)

a. If the output value may positive then take unsigned data type
b. Any variable that may contain decimal value or the output may the decimal then take that variable as float data type.
c. If storing value is a single character than take char data type.
d. If storing value is multiple characters than take string data type.
e. If output may null then we can choose a void data type.

Constant /or/ Literal

In a program a name may be assigned to store a data. If that data remains the same throughout the program execution, then it is called a “constant” or “literal”. A value written as constant for a program never change but if it is written as variable then it may change.

There are three types of Constants

1. Character Constant
2. String Constant
3. Numeric Constant

Character constant

Character constant is a single character, single digit or a single symbol enclosed with a pair of single quotation mark ‘ ’ .

Example: ‘A’ ‘c’ ‘2’ ‘#’ etc

String constant

String constant is a sequence of alpha numeric characters enclosed with a pair of double quotation mark " "

Example: "apple" "Table No 21" "*123#" etc

Numeric constant

Numeric constant has a constant value in numbers. The value may positive or negative number. There are four types of numeric constant.

a. Integer constant
b. Floating point constant
c. Octal Constant
d. Hex constant

Integer Constant

Integer constant are the whole no with positive or negative value. It may short or long data type.

i.e 5 124 -65 0 -789 or 34456567787 etc

Floating-Point Constant

Floating-Point constant are the fractional number. It has real value with positive or negative value. It may be written in two forms as fractional or exponential.

Fractional i.e. 5.23 124.43 -65.21 0.26 etc
Exponential i.e. 3.7E12 4.2E-15 etc.

Octal Constant

Octal constant are the integer number with a base of 8. The digit allowed in the system are 0 to 7. It may positive or negative. It is written as preceding the digit "0".

Octal No i.e. 005 0124 -098 026 etc

Hex Constant

Hex constant are the Hexa-decimal number with a base of 16. The digit allowed in the system are 0 to 9 and A to F. It may positive or negative. It is represented as preceding the "0x".

Hexa-Decimal No i.e. 0x5 0x124 -0x98 0x26 etc

Symbolic Constant

A symbolic constant is represented as a name of a symbol.

Symbolic Constant i.e. pi etc.

Declaring a constant

Syntax

#define constantName constantValue

Example

```
#define organization "unitty academy"          (String Constant)
#define vowel2 'e'                              (character Constant)
#define num1 24                                 (integer constant)
#define increment 2400.00                       (floating point constant)
#define oct1 035                                (octal constant)
#define hexa1 0x54                              (Hexa constant)
```

#define pi 3.14 (Symbolic constant)

Writing comment in a program

Comment in "C" program is a statement which is escaped by the compiler for execution. The "C" compiler never executes these lines. Comments are of two types.

a. Single line comment (Double forward slash "//" is used before the comment text)
b. Multi-Line Comments (a forward slash with asterisk "/*" is used before the text & an asterisk with forward slash "*/" is used after the comment text.

How to write comments in a program

```
// This is a single line comment
// This line also skipped by compiler.

/*  Programmer Name: Abinash Kumar
        This program is developed to identify the prime number
        Through user entry  */
```

Statement in "C"

Statement is a line of instruction which display the desired text of user or output value of a program. It is written in a parenthesis with double quotation following the command "printf".

Syntex: - printf("Write the text to display");
Example: - printf("My Institute Name is Unitty Academy, Sinapali");

Runtime Library in "C"

A runtime library is a file that contain one or more pre-written programs to perform specific & commonly used function. It is used in high level languages like "C", "C++" etc. Each runtime library file contains separate code to perform separate task by using its predefined functions. These files are also called header file.

Example: -

stdio.h	This file contains the Basic input/output functions
conio.h	This file contains console input/output functions
math.h	This file contains all mathematical functions
geomatry.h	This file contains Geometrical functions
graphics.h	This file contains functions to perform 2D & 3D graphics
etc…	

Functions in "C"

A function is a sub-program that contains instruction or statement to perform a specific computation on its variable.

Function is defined as the action carries out by a program or subroutine. In "C" Compiler the most important and useful function is "main()" function. A function in "C" is similar to a subroutine or procedure in other procedural language like PASCAL. When "C" compiler run a program at first the "main()"function is invoked (called) then other functions.

Delimiter in "C"

After writing a function name like main(), there are a pair of curly braces to denote the beginning and end of the function body. That braces are called the delimiter. The Opening curly braces denote the initiation of the function body & the Closing curly braces denotes the end of the function body.

Structure of a Program in "C"

Library Files
Main function
Delimiter Start
Program Statement 1
Program Statement 2
.
.
.
Delimiter end
(Program End)

Example of a sample "C" Program.

```
#include<stdio.h>

main()
{
printf("/n I learn computer training at Unitty Academy, Sinapali");
printf("/n The sum of 5 & 6 is %d", 5+6);

}
```

Argument List (it may a function, variable, value or all)
Control String

Format Specifier in "C"

The printf() function is the most important function in "C" to display the formatted text or output. To format the output data it needs some format specifiers according to the "Data Type" used for the concern variable.

Format specifier is a single character followed by "%" sign which specify the text or number to display in the given format.

%c is used to display a single character
%d is used to display a decimal integer
%f is used to display a floating point no
%h is used to display a short integer
%o is used to display an octal no
%s is used to display a string
%x is used to display a hexa-decimal no

Example of a sample program to display different variables declaration with its output value using format specifier.

```
#include<stdio.h>
main()
{
// This program will display the output according to the format specifiers used

float salary = 20000.00;
int num1 = 4540;
```

```
short int num2 = 125;


printf("\n The Salary is Rs %f", salary);
printf("\nThe intiger no is %d ", num1);
printf("\n The short intiger no 2 is  %h", num2);
printf("\nThe octal no of 125 is  %o", num2);
printf("\nThe hexa-decimal no of 4540 is  %x", num1);
printf("\n%f is The  Salary", salary);
}
```

printf("\n The Salary is Rs %f", salary);

in this statement \n is used to create a line break.

Important Note: -

- Where the format specifiers are took place at that position the value of the concern variable will displayed.
- We can use multiple format specifiers in the control string to use more than 1 variable in the argument list respectively for correct output. Example
  - printf("\n The integer no is %d \n The octal no of 125 is %o", num1, num2);
    - Output 1 is The integer no is 4540
    - Output 2 is The octal no of 125 is 175

Operator in "C"

The operators are the sign, symbol or word, which operate two or more variable or value in a program. We need operators to perform arithmetic operation like addition, subtraction, multiplication, division and so on.

There are 7 types of operators in "C" Compiler.

1. Arithmetic Operator
2. Relational Operator
3. Logical Operator
4. Assignment Operator
5. Pointer Operator
6. Special Operator
7. Bitwise Operator

Arithmetic Operator in "C"

An arithmetic operator is a symbol which performs arithmetic operation like addition, subtraction, multiplication, division & Remainder.

Operator	Symbol used	Form of operation	Operation done
Addition	+ (Plus)	x + y	Add y to x
Subtraction	- (Minus)	x - y	Subtraction y from x
Multiplication	* (Multiplication)	x * y	Multiply x times to y
Division	/ (Division)	x / y	Divide x by y
Remainder	% (Modulo)	x % y	Divide x by y and output the remainder

Example Arithmetic Operator used in "C" Program

```
#include<stdio.h>
main()
{
// This program will calculate the drawn salary of an employee
float salary = 5000.00, incentive = 1000, total_sal = 0, penalty = 0, drawn_sal = 0;
int month = 4, ua_leave = 2;

total_sal = (salary * month)+ incentive;
penalty = (salary / 30) * ua_leave;
drawn_sal = total_sal - penalty;

printf("Dear Employee, Your Monthly Salary is Rs %f. Your Incentive amount is Rs %f. Your Total Salary is Rs %f. But You took un-authorized leave for %d days, So your penalty amount is Rs %f. Now your drawn  salary for %d month is Rs %f",salary,incentive,total_sal,ua_leave,penalty,month,drawn_sal);
}

/* Output will be
 Dear Employee, Your Monthly Salary is Rs 5000.000000. Your Incentive amount is Rs 1000.000000. Your Total Salary is Rs 21000.0  00000. But You took un-authorized leave for 2 days, So your penalty amount is Rs 333.333344. Now your drawn  salary for 4 month is Rs 20666.666016 .
```

How to use "dev C++" for "C" Program

Download Dev C++ → Right Click on it → Choose Run as administrator →Click Yes → Click Next → Click Next → Click Finish. (Now Dev C++ is installed in your computer and a shortcut icon 

will be shown on the desktop)

How to save a program code for "C"

Double Click on the shortcut icon of "Dev C++" → Choose File Menu →New →Source File →type your programming code → Go to File menu → choose save → Give a name to your program file →then click save as type dropdown →choose "C Source Files"→ Click "Save"

How to compile & run a program code in "Dev C++"

Go to Execute menu → choose compile →correct the error in the program if it shows →again compile it →if there is no error now → Click save button →Go to Execute menu →choose run.

Increment & Decrement Operator

The increment operator in "C" Program is represented by "++" sign and the decrement operator is represented by "- -" sign. Increment operator increases the value of a variable by adding 1. Decrement operator decreases the value of a variable by subtracting 1.

Example:

X = 5;	here the value of x is 5
++x;	here the value of x will be 6
Y = 8;	here the value of Y is 8
--Y	here the value of Y will be 7
Again if we write	
--x	here the value will be 5

++Y here the value will be 8

Note: We can use the following procedure to add or subtract 1 to a variable.

++a or a = a+1 or a+=1

--b or b=b-1 or a-=1

Pre-Increment & Post-Increment Operator

Pre-Increment operator is represented with a "++" sign before the variable name.

Pre increment operator adds the value 1 to the concern variable and then assign the new value to the next variable.

Pre Increment operator Example

int sum = 0, x = 12;

Sum = ++x; (Here value of x is added with 1 and new value becomes 13 then that value transferred to the variable "Sum" and now value of sum will be 13).

Post-Increment operator is represented with a "++" sign after the variable name.

Post- increment operator first assign its existing value to the next variable then add the value 1 to the concern variable and form new value.

int sum = 0, Y = 23;

Sum2 = Y++; (Here value of Y which is 23 is transferred to the next variable "Sum2" and makes the value of Sum2 to 23 then increase the value of Y by adding 1 and form a new value of "Y" as 24).

x = 12, Y = 23;

Sum3 = x++ + Y + Y++; 58

Sum4 = --y + Y-- + ++x; 59

Precedence of Arithmetic Operators

If we have various arithmetic operators to calculate various values then which arithmetic operator is used to evaluate first is called Precedence.

But if we have two operators in the same precedence then they are evaluated Left to Right but in case of increment or decrement operator they are evaluated right to left.

The Highest precedence is evaluated first.

Level	Operators	Description	Associativity
15	() [] -> . ++ --	Function Call Array Subscript Member Selectors Postfix Increment/Decrement	Left to Right
14	++ -- + - ! ~ (type) * & sizeof	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting Dereferencing Address of Find size in bytes	Right to Left
13	* / %	Multiplication Division Modulo	Left to Right
12	+ -	Addition / Subtraction	Left to Right
11	>> <<	Bitwise Right Shift Bitwise Left Shift	Left to Right
10	< <= > >=	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To	Left to Right
9	== !=	Equality Inequality	Left to Right
8	&	Bitwise AND	Left to Right
7	^	Bitwise XOR	Left to Right
6	|	Bitwise OR	Left to Right
5	&&	Logical AND	Left to Right
4	||	Logical OR	Left to Right
3	?:	Conditional Operator	Right to Left
	= += -=		

Relational & Logical Operators

The operator which is used to compare between the value of two variables is called relational. Such as Less than, Greater Than, Greater than equals, not equals etc.

The operator **And (&&)** and **OR (||)** is called Logical Operator.

Relational Operator Table

Operator	Operation (Action)
>	Greater Than
>=	Greater than equal to
<	Less Than
<=	Less Than Equal to
==	Equal to
!=	Not Equal to

Logical Operator Table

Operator	Operation (Action)	
&&	And	It evaluate each and every expression.
||	Or	It evaluates all the expression one after another until it returns true or end of the expression.
!	Not	It reversed the result (makes true if false and make false if true)

These operators output the result either “True” or “False”. True represent “1” or “non zero” & False represent “0”.

```
x = 5;
y = 7;
x > y;                Output  0
y > x;                Output  “1” or “non zero”
result1 = x > y;      Output  0 will stored in the variable result1
result2 = y > x;      Output  “1” or “non zero” will stored in the variable result2
```

Example

```
#include<stdio.h>
main()
{
// This program is the example of Logical & Relational Operators

Int   x= 5, y = 8, z = 0, result1 = 0, result2 = 0, result3 = 0, result4 = 0;

result1 = x<=y;
result2 = x>=y;
```

```
result3 = x!=y;
result4 = x==y;

printf("is x is greater than y %d",x>y);
printf("\n is x is Less than y %d",x<y);
printf("\n is x Less than equals y %d \n is x  greater than equals y %d \n is x  not equals y %d \n is x  equals y
%d",result1, result2, result3, result4);
}
```

Output of the Program

```
 is x is greater than y 0
 is x is Less than y 1
 is x Less than equals y 1
 is x  greater than equals y 0
 is x  not equals y 1
 is x  equals y 0
```

Precedence of Relational & Logical Operator

Operators

Operator	Associativity	Precedence
!	Right to left	Highest
> >= < <=	Left to right	↑
== !=	Left to right	↑
&&	Left to right	↑
||	Left to right	Lowest

Pointer Operator

The Pointer operator are the “address of operator” (&) and the “indirection operator” (*). The “indirection operator” is used to get the content of a file, memory location or cell. The “&” operator returns the address of the variable.

Example:

Int salary = 5000;	Here an integer variable is declared as “salary”
Int *data_in_salary;	Here a pointer variable is declared as “data_in_salary”
data_in_salary = &salary;	Here address of the data in the variable salary is assigned to the pointer variable “data_in_salary”.

Program Example

```
#include<stdio.h>
main()
{
```

```
// This program is the example of Pointer Variable
int salary = 5000;
int *data_in_salary;
data_in_salary = &salary;

printf("\n value of salary is %d",salary);
printf("\n value of data_in_salary is %d",*data_in_salary);
*data_in_salary = 25000;
printf("\n value of salary is %d",salary);

}
```

Output of the Program

value of salary is 5000
value of data_in_salary is 5000
value of salary is 25000

N.B. "When we change the value of a pointer variable (here *data_in_salary) then the value of the addressed variable (here salary) will be changed."

Special Operator

The Programming Language "C" Contains many different operators but some operators do not come under any category, they are called **miscellaneous** or **Special** Operator.

Example:

Conditional Operator: - ?:
Coma Operator: - ,
Sizeof operator: - sizeof

Syntex

Variable_Name = Expression_1 ? Expression_2 : Expression_3

Here if the condition of the expression 1 is comes true than the value of expression 2 will be assigned to the variable else the value of expression 3 will assigned to the variable.

Program Example: 1

```
#include<stdio.h>
main()
{
// This program is the example of special operator (? :)
int target = 5000, incentive = 1000, sales = 6500, x = 0, y = 0, z = 0;

x = sales > target ? incentive : 0;
y = sales < target ? incentive : 0;
z = sales == target ? incentive : 0;

printf(" \n U Get the amount against incentive Rs %d",x);
```

```
printf(" \n U Get the amount against incentive Rs %d",y);
printf(" \n U Get the amount against incentive Rs %d",z);
}
```

Output of Program

```
U Get the amount against incentive Rs 1000
 U Get the amount against incentive Rs 0
 U Get the amount against incentive Rs 0
```

Program Example: 2

```
#include<stdio.h>
main()
{
// This program is the example of special operator (? :)
int target = 5000, incentive = 1000, sales = 6500;
char = *msg1, *msg2, *msg3;
msg1 = "Bravo, Well don boy, keep it up, promotion is near by u";
msg2 = "hey boy, u do worse duty, be seriour and sale perfectly";
msg3 = "neutral duty, do some more";
printf(" \n %s",sales > 5000 ? msg1 : msg2);
printf(" \n %s",sales < 5000 ? msg1 : msg2);
printf(" \n %s",sales == 5000 ? msg3 : msg2);
}
```

Coma Operator (Special Operator Category)

Coma operator (,) is used to build a compound expression by putting several expression inside a set of parenthesis. The expression is evaluated from "Left to Right" & the last evaluated value is the final value.

Example:

```
#include<stdio.h>
#include<conio.h>
main ()
{
   // This program is the example of coma Operators
char chr1 = 'z', chr2;
printf (" \n The coma operator expression value is %c", (chr2 = chr1, chr1 = getchar ()));
printf (" \n The Original value of C is %c \n", chr2);
}
```

Output of the Program

```
f                          (User entry done here)
 The coma operator expression value is f
 The Original value of C is z
```

Sizeof operator

The "sizeof" operator returns the memory size of a variable.

Syntex

sizeof(variable_Name);

Example

sizeof(str1);

Program Example

```
#include<stdio.h>
main()
{
// This program will calculate the size of the variable
float salary = 5000.00, total_sal = 0, incentive = 1000, penalty = 0, drawn_sal = 0;
int month = 4, ua_leave = 2;
char grade1 = 'A', grade2 = 'b', alpha [26];

printf(" \n Memory size of float variable salary  is %d byte", sizeof(salary));
printf(" \n Memory size of char variable grade1 is %d byte", sizeof(grade1));
printf(" \n Memory size of integer variable month is %d byte", sizeof(month));
printf(" \n Memory size of array variable alpha is %d byte", sizeof(alpha));
}
```

Output of the Program

Memory size of float variable salary is 4 byte
Memory size of char variable grade1 is 1 byte
Memory size of integer variable month is 4 byte
Memory size of array variable alpha is 26 byte

Bitwise Operator

To operate the operand in machine level, we need bitwise operators. A bit is a smallest possible unit of data storage and it can have only one of the two possible values “0” or “1”.

The bitwise operator is operated using two shift operations.

a. Left Shift Operation
b. Right Shift Operation

Left Shift Operation

The left shift operation shifts the bits to the left for a specified no of bit operation. The sign used to represent left shift operation is **double less than** “<<”.

Syntax

Variable_name << No of bits to shift

Example

```
int  num1 = 7, num2 = 12;
num1 << 1          (The new value will be 15)
num2 << 2          (The new value will be  48)
```

BIT Range for Left Shift Operation							
128	64	32	16	8	4	2	1
Left Shift 1 of 7							
0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	1
Left Shift 2 of 12							

0	0	0	0	1	1	0	0
0	0	1	1	0	0	0	0

Original Bit of 7
After left shift to 1

Original Bit of 12
After left shift to 2

Right Shift Operation

The right shift operation shifts the bits to the right for a specified no of bit operation. The sign used to represent right shift operation is **double greater than** ">>".

Syntax

Variable_name >> No of bits to shift

Example

```
int  num1 = 7, num2 = 12;
num1 >> 1          (The new value will be 3)
num2 >> 2          (The new value will be 3)
```

BIT Range for Right Shift Operation							
128	64	32	16	8	4	2	1
Right Shift 1 of 7							
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
Right Shift 2 of 12							
0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1

Original Bit of 7
After right shift to 1

Original Bit of 12
After right shift to 2

Bitwise Operators & Its Meaning

Bitwise Operator	Meaning
&	Bitwise AND
||	Bitwise Or
^	Bitwise xor
~	1's Complement
>>	Right Shift
<<	Left Shift

Program Example

Expression

An expression in programming language "C" is a combination of operators, Numbers & Names that work together to carry out the result.

Types of Expression

There are four types of expression in "C" Language.

a. Constant Expression Ex. 5+6*34/3.5;
b. Integer Expression Ex. J*K+5-a;
c. Float Expression Ex x/y+3.56;
d. Pointer Expression Ex P; &j; P+j;

Type Conversion

The Programming Language "C" Permits us to mix the different types of variable & constant for calculation. Sometimes it is required to convert a variable to another type, at that time we need "typecast". The compiler performs these calculations without intimating the user is called "implicit conversion". The programmer may instruct the compiler to convert one type of variable to another type is called "Explicit Conversion".

Example of Explicit Conversion (Type Conversion)

Syntax

(Typecast_Datatype)Variable_Name
(Float)x;
(int)Salary;

Printf("The Converted value of x from integer to float is %f", (float)x);

Program Example

```
#include<stdio.h>
#include<conio.h>
main ()
{

  // This program is the example Typecast

int x = 5, y = 7, sum =0;
float a = 5, b = 9;
float sum2 = 0, sum3 = 0;
sum = x + y;
sum2 = y/x;
```

```
sum3 = a/b;

printf (" \n The value of X is %d and \n the value of Y is %d", x,y );
printf (" \n The Total of X & Y before Typecast %d", sum );
printf (" \n The Total of X & Y after Typecast to float %f", (float)sum);
printf (" \n The division of X & Y before Typecast %f", sum2);
//The below line is example of implicit conversion
printf (" \n The Division of X & Y after Typecast to integer %d", (int)sum2);
printf (" \n The division of a & b before Typecast %f", sum3);
// implicit conversion ends here (autometic converted by the compiler witout informing the user)
printf (" \n The Division of a & b after Typecast to integer %d", (int)sum3);
}
```

Output of the Program

```
The value of X is 5 and
 the value of Y is 7
 The Total of X & Y before Typecast 12
 The Total of X & Y after Typecast to float 12.000000
 The division of X & Y before Typecast 1.000000
 The Division of X & Y after Typecast to integer 1
 The division of a & b before Typecast 0.555556
 The Division of a & b after Typecast to integer 0
```

Input / Output Statement

"C" Program allows us to read data from standard input device like Keyboard & Write data to the standard output device like Monitor (VDU). For this purpose there is some input & output statement is used.

There are many library functions available for input & output devices. These functions are categorized into 3 broad categories.

a. Console I/O Function — it Receive input from keyboard & Write output to Monitor.
b. Disk I/O Function — it read & write to the storage media like Floppy, Hard disk, SD Card etc
c. Port I/O Function — it performs various input & output operation to the different ports like USB, VGA, HDMI, PS2, LAN, Audio port etc.

Console I/O Functions

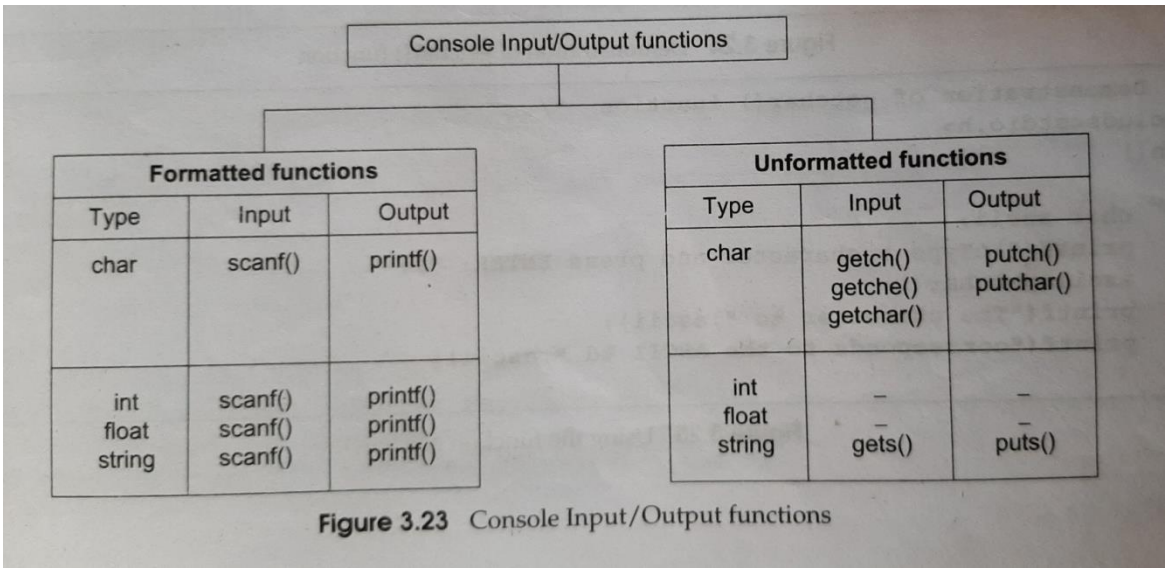


Figure 3.23 Console Input/Output functions

Formatted Console I/O Functions

Output Function printf(): This function displays the formatted output to the standard output device.

Syntax

printf("Message Text Format Specifier", VariableName);

Example

printf("The sum of x + y is %d", sum);

input Function scanf(): This function accept the formatted input from the standard input device "Key Board".

Syntax

scanf("Message Text Format Specifier", addressofVariableName);

Example

scanf("Enter the value for x and y %d", &x, &y);

Program Example

```
#include<stdio.h>
main()
{
// This program will calculate the drawn salary of an employee with user input
float salary = 5000.00, total_sal = 0, incentive = 1000, penalty = 0, drawn_sal = 0;
int month = 0, ua_leave = 0;
printf("Enter no of month ");
scanf("%d", &month);
printf("\n Enter Un Authorised Leave ");
scanf("%d",&ua_leave);
total_sal = (salary * month)+ incentive;
penalty = (salary / 30) * ua_leave;
drawn_sal = total_sal - penalty;
printf("\n You are receving your %d month of sal with %d days of UA_Leave is Rs %f",month, ua_leave,
total_sal);
}
```

Output of the Program

```
Enter no of month 5
 Enter Un Authorised Leave 7
You are receving your 5 month of sal with 7 days of UA_Leave is Rs 26000.000000
```

Unformatted console I/O Function

Character Input: - for Character input as unformatted function we use getch(), getche() & getchar() function.

The getch() & getche() function is very similar as they respond without pressing of enter key of keyboard. The only different is that the getche() function makes the user entry text visible but the getch() function donot show the text entered by the user. If we use these two function in any program than we neet to include the header file "conio.h" in the beginning of the program.
The getchar() reads one character from the key board after the new line character (enter key) is received.

Syntax

```
char x, y, z;
getch(variableName);
getche(variableName);
getchar(variableName);
```

Example

getch(x); reads a single character and store it to the variable x without visible user input.
getche(y); reads a single character and store it to the variable y with visible user input.
getchar(z); reads a single character and store it to the variable z after pressing enter key.

Character Output: -

putc()
putchar()

if we have to print a single character in a program the instead of using the formatted output function printf(), we must use the unformatted putc() or putchar(). For doing this we must enclose the character in a single quotes (' ').

Unformatted String input & Output: - gets() & puts()

gets() stands for "get the string". It reads the string entered by the user till the newline character (enter key press). It can assign the string to an array variable. We must use double quotes ("") to enclose the string.
puts() stands for "put the string". It print the existing string of the variable or text encoded with double quotes.

Syntax & Example

```
char name[25];
gets(variablename);
gets(name);
puts(variablename);
puts(name[]);
```

Program Example of all unformatted functions

```
#include <stdio.h> //This program is an example ofunformatted character function
#include <conio.h>
main()
{
char c, d, e;
printf("\nEnter a character getchar : ");
c = getchar();
printf("\nEntered character : getchar %c \n", c);
putchar(c);
```

```
printf("\nEnter a character 2 getch : ");
d = getch();
printf("\nEntered character getch: %c \n", d);
putch(d);

printf("\nEnter a character 2 getche : ");
e = getche();
printf("\nEntered character getche: %c \n", e);
putch(e);
return 0;
}
```

Program Output

gets & puts program example

```
#include <stdio.h> //This program is an example ofunformatted string function
#include <conio.h>
main()
{
char str1[20];
printf("\nEnter Your Name : ");
gets(str1);
printf("\n Your Name is %s \n", str1);
puts(str1);
return 0;
}
```

Program Output

```
Your Name is Ramanuj
Ramanuj
```

Field width Specifier

The printf() function allow the programmer to add formatting to the printed output. A digit preceding the decimal point in the field width specifier controls the number as it is printed.
When we use field width specifier in the format specifier inside the control string, it set the “zero” before the integer or after the decimal accordingly.

Example of field width specifier

```
#include<stdio.h>
main ()
{
   // This program is the example field width specifier

int num1 = 3476;
float num2 = 2657.345650;
```

```
printf("\n The float no without field width specifier is %f", num2);

printf("\n The float no with field width specifier is %2.3f", num2);
printf("\n The float no with field width specifier is %0.012f", num2);

printf("\n The integer no without field width specifier is %d", num1);

printf("\n The integer no with field width specifier is %8d", num1);
printf("\n The integer no with field width specifier is %08d", num1);
}
```

Program Output

The float no without field width specifier is 2657.345703
The float no with field width specifier is 2657.346
The float no with field width specifier is 2657.345703125000
The integer no without field width specifier is 3476
The integer no with field width specifier is _ _ _ _3476 (underscore means blank)
The integer no with field width specifier is 00003476

Escape sequence

An escape sequence provides special formatting control on output of the program. It allows us to use Tab, Line break (new line), Backspace, alert and so on in a program for output. It is also called Execution character. It is used inside the control string encoded with double quotes. The back slash symbol "\" is called escape in "C Program".

Escape Sequence

Escape Sequence	Purpose
\n	It insert a new line
\b	It set a back space
\f	For form feed
\'	It insert single quotes
\\	It insert back slash
\t	It insert a Tab
\r	It return carriage
\a	It Sound an alert
\"	It insert double quotes

Escape Sequence	Meaning	Elucidation
\n	New line	Used to shift the cursor control to the new line.
\t	Horizontal tab	Used to shift the cursor to a couple of spaces to the right in the same line.
\a	Audible bell	A beep is generated indicating the execution of the program to alert the user.
\r	Carriage Return	Used to position the cursor to the beginning of the current line.
\\	Backslash	Used to display the backslash character.

Example of escape sequence

syntax

printf("escape sequence name Your control message of output format specifier", variablename);

Example

printf("\n the value of the variable x is %d", x);

printf("\n the value of the variable x is \t %d", x);

Difference between printf() & scanf()

The major difference between scanf() & printf() is that in scanf() the data item argument must be preceded by the address operator "&". Example scanf("%d",&num1). It direct the system to read integer input from the user terminal. But in printf() function variable name is written without address operator and it return the output for the user.

Conditional Statements & Loops

Statements in C are executed one after another is called the normal statement. But a statement which instructs a system to perform same basic operation again and again until a condition is fulfilled is called loop. It is the automated power of C Program. These are happens using control statements.

Control statements allow us to change the sequence of instructions for execution. It is of two types…

- Conditional Branching
- Looping

Conditional Branching

When a program decide whether the next statement is to be executed or not based on the result value of the current expression, is called condition branching.

Looping

When a program performs a set of operation repeatedly until a given condition is fulfilled is called loop. In loop all the statements inside the loop are executed repeatedly. It is also called "***iteration***".

Decision Making

Decision making is a process to execute a specific task according to the conditional requirement.

Ex: - "a friend said to his friend that if it feels too hot then you put the fan switch on".

If Statement

The "if statement" is a control statement which test a particular condition. Whenever the evaluated condition comes true the an action is carried out otherwise the given set of instruction is ignored.

```
Syntax:     If (conditional expression)
                {Statements};
Example:    if (sales>=50000)
                {incentive = (sales*10)/100}
```

Here if a salesman sale a product more than or equal to 50000 then he can gain an incentive upto 10 percent, but if sales below 50000 then no incentive is given to him.

Program Example:

```
#include<stdio.h>
main()
{
// This program will calculate the incentive of a salesman (if condition)
float sales = 0, incentive = 0;
```

```
printf("Enter the sales amount");
scanf("%f", &sales);
if (sales>=50000)
        {
          incentive = (sales*10)/100;
          printf("The incentive amount is %f", incentive);
        }
        printf("\n note: Please enter the sales amount above 50000");
}
```

If_else_Statement

The else part of the "if statement" is run when the "conditional expression" of if statement comes false. We can call it the false statement.

Syntax

```
If (conditional Expression)
        {True Statement}
else
        {false Statement}
```

Example:

```
        #include<stdio.h>
main()
{
// This program will calculate the incentive of a salesman (if else statement)
float sales = 0, incentive = 0;
printf("Enter the sales amount");
scanf("%f", &sales);
if (sales>=50000)
        {
          incentive = (sales*10)/100;
          printf("The incentive amount is %f", incentive);
        }
else
        {
          incentive = 0;
        printf("The incentive amount is %f", incentive);
        }
}
```

Multiple If_Statement

In the multiple if statement those statements will run which conditions are fulfilled.

Program Example

```
#include<stdio.h>
main()
{
// This program will calculate the incentive of a salesman (if else statement)
float sales = 0, incentive = 0;
```

```
printf("Enter the sales amount");
scanf("%f", &sales);
if (sales>=50000)
        {
          incentive = (sales*10)/100;
          printf("\n The incentive amount is %f", incentive);
        }
if (sales == 0)
        {
          printf("\n You are fired");
        }
if (sales < 50000)
        {
          incentive = 0;
        printf("\n You can avail salary only but The incentive amount is %f", incentive);
        }
```

If else if Statement (Nested if)

In this statement, we can use more than one condition in a program and the program will run only that single statement of the condition which is fulfilled.

Program Example

```
#include<stdio.h>
main()
{
// This program will calculate the incentive of a salesman ( if else if statement)
float sales = 0, incentive = 0;
printf("Enter the sales amount\t");
scanf("%f", &sales);
if (sales>=50000)
        {
          incentive = (sales*10)/100;
          printf("\n The incentive amount is %0.2f \t", incentive);
        }
else if (sales == 0)

        {
          printf("\n You are fired");
        }
else if (sales < 50000)
        {
          incentive = 0;
        printf("\n You can avail salary only but The incentive amount is %d", (int)incentive);

        }
}
```

The Switch Case Statement

Instead of using if-else-if Ladder, the switch case statement is used. The only different is that in switch case statement u must have to use the integer constant as the conditional expression but in if-else-if ladder you can use any datatype variable or constant for conditional expression to return the expected result.

Syntax

```
Switch(variablename)
Case integervariablevalue:
    Statements;
    break;
```

Break statement is used to block the unwanted statement output after program execution. Because switch case statement always run all those lines from where the condition comes true. To stop the execution after the "true" statement, we must use "break" statement.

Not to do in switch case statement.

a. Never use char or string variable for "switch parameter".
b. Never use decimal constant.
c. Never use character constant.
d. Never use a string for "case level".
e. Never use variable for "case level".

Program Example

```
#include<stdio.h>
main()
{
// This program will calculate the week day name according to day no entry. (Switch case)
int dayno = 0;
printf("Enter the Day no (1 to 7)\t");
scanf("%d", &dayno);

switch(dayno)

{
case 1:
    printf("\n Sunday \t");
    break;
case 2:
    printf("\n Monday \t");
    break;
case 3:
    printf("\n Tuesday \t");
    break;
case 4:
    printf("\n Wednessday \t");
    break;
case 5:
    printf("\n Thursday \t");
    break;
case 6:
```

```
        printf("\n Friday \t");
        break;
case 7:
        printf("\n Saturday \t");
        break;
default:
        printf("\n Please enter 1 to 7 \t");

}
}
```

The Loop construct

The loop or iteration construct directs a program to perform a set of operation again and again until a specific condition is achieved. This condition cause the termination of the loop. In "C Program" there are three statements for looping they are

a. ***while*** loop
b. ***do …… while*** loop
c. ***for*** loop

While Loop

The "while loop" allows evaluates a test expression before allowing entry into the loop. It means it is an "entry control loop". while loop construct first check the condition and if the condition is come true then it execute the statements following the loop. Else it ignore the statement and break the looping.

Syntax

```
while (condition)
{
        Statement1
        Statement2
        ……..
}
```

Program Example (while loop)

```
#include<stdio.h>
main()
{
// This program will display the even no from 1 to 100 (while loop statement)
int num1 = 0;
while(num1<100)
{
        num1 = num1+2;
        printf("\n The Serial No is %d", num1);
}
}
```

Print the Fibonacci series using while loop

1, 1, 2, 3, 5, 8, 13….. upto 100

Do…. while Loop

The “do …. while loop” is an exit control loop. This loop execute the statement at least once before checking the test expression (parameter). After execution of the statement it will check whether the test expression is true or false. If test expression comes true, then it will repeat the statement else it will break the loop. We must put a semi-colon (;) mark after while statement in do…while loop construct.

Syntax

```
do
{
Statement 1
Statement 2
Statement ….
}
While (Test Expression);
```

Program Example

```
#include<stdio.h>
main()
{
// This program will display the even no from 1 to 100 (do... while loop statement)
int num1 = 0;
do
{
        num1 = num1+2;
        printf("\n The Serial No is %d", num1);
}
while(num1<100);
}
```

For…. Loop

It is an entry control loop. The execution process is same like while loop. The only different is that in for loop we write initial expression, test expression and increment or decrement statement under for loop parenthesis. So the “For loop” run the initial expression first and once, then run the statement under for loop then test the expression then increase or decrease the value of the incrementer or decrementor then it repeat the statements again if the test condition comes true.

Syntax

```
For (Initial Expression; Test Expression; Incrementor or Decrementor )
{
Statement 1
Statement 2
Statement…
}
```

Program Example

```
#include<stdio.h>
main()
{
// This program will display the even no from 1 to 100 (for..... loop statement)
```

```
int num1;
for(num1 = 0; num1<100; num1 = num1  +  2)
{
        printf("\n The Serial No is %d", num1);
}
return 0;
}
```