Why learn C programming Language?

C programming language is a computer programming language which was developed to do system programming for the OS UNIX & is an imperative programming language.

What is programming?
Programming or Coding is the way by which we can interact with computer & can give instructions to computer.

Computer only understands Machine Language (Binary Digits)

C is widely used programming language because it is reliable, simple & easy to use language. And because of the reasons only nowadays also C language is mostly used all over the world.

C is a low-level language as in C language the programmer has most of the control of the code.

History of C language

C language was created by Dennis Ritchie at AT&T {American Telephone & Telegram (USA)} Bell Labs in 1972.

Features of C language

C is very fast & efficient programming language.

It is easy to learn language

It has many pre-defined functions & operators which make Complex program easy.

It is one of the most powerful programming language.

Importance of C language

C programming is used when we need to extract best performance of the program.

Most of the OS are developed by using C language only.

Most of the part of Linux OS is written in C language.

MySQL Database is written or Coded in C language.

Most of the Part of Ruby & Pearl Programming language is written in C language.

Python's Library i.e CPython is written in C language.

IDE & Compiler

IDE - IDE (Integrated Development Environment) is a software application which provides many comprehensive facilities to programmers for software or application development.

ide provide tools to create build & test software's. It also provides programmable editors, libraries, build automation tools etc.

It has features like Syntax highlighting, Syntax checking

etc. E.g. Net Beans, Microsoft visual Studio etc

Compiler:- It's a software that transform computer code written in one programming language into another language i.e machine language.

E.g GNU Gcc etc.

What is coding & C programming language

Coding or Programming is the way by which we can interact with computer.

CPU (processor) only understands one language i.e Machine language or binary digits {0, 1} and to write program in directly machine language or binary digits is really a very difficult task.

Code written in programming language is converted by compiler or interpreter to machine language so that CPU can understand it & can perform the desired task.

There were many languages before C also such as BCPL, B etc but C languages got lots of hype because of its performance, easy to write code, ability to create big programs, efficiency & all.

C is a machine independent language which means we can use C language code in any machine & can compile it.

C is one the fastest programming language since it is low level language & there is a saying i.e language which is most near to machine language is faster then the language which is more user-friendly it means if the language is user-friendly then we have to compromise with speed but if the language is not much user-friendly then it gives the marvelous speed because it could be easily converted to machine language without doing much efforts.

first Code :-

```
#include <stdio.h>

int main ()
{
Printf ("Hello world\n");

Return 0;
}
```

C Program mainly consists of :-

Preprocessor commands
Functions
Variables
Statements & Expression
Comments

Understand our Code :-

#include<stdio.h> :- it is a pre-processor Command. it means we want to include stdio.h header file in our program. stdio.h stands for standard input output. it is a library in C which contains many functions definations such as printf(), scanf() etc.

int main (): main is a function in c language. Execution of every c program starts from main() function. int before main means that the function will return an integer. main function is called by OS.

{} :- Curly braces encloses the body of main() function

printf("Hello world\n"); :- printf() is a function defined in stdio.h header file & this function displays whatever is written its bracket on the Screen.

return 0; As we known that main function returns an integer value (int main()), ∴ we are returning 0 here. return is a Keyword which is used to return some value from a function. Returning 0 indicates that our program has been run successfully & we terminate our main function with this return statement.

Add two no :-

```
#include <stdio.h>

int main()
{
int a, b;
```

```
printf(" Enter no. a : \n");
scanf ("%d", &a);

printf (" Enter no. b : \n");
scanf ("%d", &b);

printf (" your sum is %d", a+b);

return 0;
}
```

Steps included in execution of C program are :-

Pre-processing -> Compilation -> Assembly -> Linking.

Pre-Processing :- In this process comments are removed, macros are expanded & #include files are included.

Extension of pre-processed file is (.i)

Compiling :- In this process pre-processed file is Converted into assembly instructions.

Extension of assembly instruction file is (.s)

Assembly :- Assembly instructions are then converted into machine language instructions.

Extension of machine instruction file is (.o)

Linking :- In this process all object file i.e (.o) files are Converted to executable i.e (.exe) program.

my companion

Basic Syntax in C language

Tokens :- These are the smallest elements of a program, which are meaningful to the compiler.

Types of Tokens :- Keywords, identifiers, Constants, operator etc.

Keyward :- these are reserved (pre-defined) words in C programming language.

There are 32 general-purpose Keywards.

Identifiers :- Each program element in C is known as identifiers. They are used for naming of variables, functions, arrays etc.

These are user-defined names which consist of alphabets numbers, underscores (_);

C program is made up of keywards, identifiers, Constants, string literal & symbol.

Variables & Data types

Variables :- A variables is a name or an identifiers which is given to any storage area or memory location.

Actually, Variables doesn't hold a value but it's the name given to any memory address in RAM. It means variables is the way to access that memory address of Ram so that we can store or manipulate data in that memory address

my companion

or memory block. It's a name of memory location.

Declaring a variable :-

Variables are declared by writing:

Type of variable name;

For E.g : char abc;, int num1;, float add1;, etc.

Initializing a Variables :-

Initializing means entering data in that variables i.e. storing some value in that variables.

Variables can be initializing & declared as:

Typeofvariable name = value

E.g :- int num1 = 52;, , char demo = 'A';

float add = 56.28;, etc

Rules for defining a variable in C :-

Variable name can contain alphabets, digits & underscore (_)

E.g :- int demo_xyz;

Variable name can start with an alphabet & underscore only

E.g int_demo;, int de_mo;

It can't start with a digit.

No whitespace & reserved Keywords are allowed to use as a variables name.

Valid Variables names :-

int demo; , float demo123; , char _demo123; etc

Invalid variables names :-

9 demo; , int 123demo; , char float;

Data types :-

Data types is the type of data that variable holds.

These are used for assigning a type to a variable.

The words which are both data type & Keyword in C language are known as Primitive Data type.

Such as int, char, float, double & void

Basic Data type :- These are the fundamental data types of C language.

E.g :- int, char, float, double

Derived Data type :- These are the data types which are made with the help of Basic Data Types.
E.g :- array, pointer, structure, union.

My companion

Enumeration Data type :- enum

Void Data type :- It is the data type which denotes the empty value i.e. void data type means "no data type".

Data type size :- varies from architecture to architecture. It means 32 bit architecture system can have some other size for any specific data type than 64 bit architecture system.

Data types	Memory size	Range
char	1 byte	-128 to 127
short	2 byte	-32768 to 32,767
int	2 byte	-32768 to 32767
float	4 byte	
double	8 byte	

It is not necessary to define any variable with signed keyword because all the variables by default are declared by signed keyword only

To check size of data type in your architecture or system.

```
#include<stdio.h>

int main ()
{
char demo = 'A';

printf("Size of int is %d\n", sizeof(int));
printf("Size of char is %d\n", sizeof(demo));
printf("Size of float is %d\n", sizeof(float));

return 0;
}
```

```
#include<stdio.h>

int main ()
{
printf("%lu", sizeof(int));
return 0;
}
```

Operator :- An Operator is a symbol which is used to perform operation on operands in any programming language.

Operand :- It is the data on which the operation of operator occurs.

Types of Operators in C language :-

Arithmetic Operators
Relational Operators

Logical operators
Bitwise Operators
Assignment operators

Arithmetic Operators:- It is the type of operators which take numerical values (literal or variable) as their operands & perform some operation on them & return a single numeric value.

```
#include <stdio.h>

int main ()
{
 int a=5, b=8, c=14;

printf(" sum of a & b is %d\n", a+b);
printf (" Product of a, b & c is %d\n", a*b*c);
printf (" Difference b/w b & a is %d\n", b-a);
printf (" b/a is %d", b/a);

 return 0;
}
```

Relational operator:- It checks the relationship between two or more operands or it compares two or more operands.

if the comparison or relationship is true, then it returns 1
if the comparison or relationship is false, then it returns 0

Let us assign a = 8 & b = 4

Operator	Description	Example
==	is equal to	(a==b) is false
!=	is not equal to	(a!=b) is true
>	Greater than	(a>b) is true
<	less than	(a<b) is false
>=	Greater than equal to	(a>=b) is true
<=	less than equal to	(a<=b) is false

```
#include<stdio.h>

int main ()
{
 int a=8, b=5;

printf ("a==b is %d\n", a==b);
printf ("a!=b is %d\n", a!=b);
printf ("a>b is %d\n", a>b);
printf ("a<b is %d\n", a<b);
printf ("a>=b is %d\n", a>=b);
printf ("a<=b is %d\n", a<=b);

return 0;
}
```

Logical Operators :-

Let a = 0 & b = 10

Operator	Description	Example
&&	logical AND operator if both the operands are non-zero then the condition is true.	(a&&b) is false
||	logical OR operator if any of these two operands is non-zero, then condition becomes true	(a||b) is true
!	logical Not operator it is used to reverse the logical state of its operand It means if Condition is true then it will make is false	!(a&&b) is true.

```
#include <stdio.h>

int main ()
{
 int a=0, b=10;

printf("a&&b is %d\n", a&&b);
printf("a||b is %d\n", a||b);
printf("!(a&&b) is %d\n", !(a&&b));

return 0;
}
```

Bitwise operator :- These operators work on binary digits i.e. bits & perform bit by bit operations.

a	b	a&b	a|b	a^b
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
0	0	0		1

~ is the binary one's complement operator.

<< is the binary left shift operator.

>> is the binary right shift operator.

Assignment Operators

Operator	Description
=	Simple assignment operator: Assigns value from right side operands to left side operands.
+=	Add AND assignment operator: It adds the right operand to the left operand & assign the result to the left operand.
-=	Subtract AND assignment operator: it subtracts the right operand from the left operand & the result is assigned to the left operand.

*=	Multiply AND assignment operator :- It multiplies the right operand with the left operand & the result is assigned to the left operand.
/=	Divide AND assignment operator It divides the left operand with the right operand & the result is assigned to the left operand.

Miscellaneous Operator

Operator	Description	Example
Sizeof ( )	Returns the size of a variable	sizeof(a), where a is integer; will int's size on that architecture
&	Return the address of a variable	&a; return the actual address of the variable
*	Pointer to a variable	*a;
?:	Conditional Expression	if Condition is true ? then value X otherwise value Y

Operator Precedence in C language

Category	Operator	Associativity
Postfix	( ) [ ] -> . ++ --	Left to Right
Multiplicative	* / %	Left to Right
Additive	+ -	Left to Right
Relational	< <= > >=	Left to Right
Equality	== !=	Left to Right

The precedence of operator works up to down i.e. first program solve postfix then it will come down till Equality (as i mentioned but turns to , comma).

In programming language, the associativity of an operator is a property that determines or tells the compiler how operators of the same precedence are grouped in the absence of parentheses( ).

Format Specifier:

It is the way to tell the compiler what type of data is in a variable during taking input & displaying output to the user.

Some ex:-

%c - character
%d = Integer
%f = float

%l = Long integer
%lf = double
%Lf – long double etc

Printf ("The marks of Ram is %a.bf", variable); will print variable with 'b' decimal points in 'a' character space.

E.g : float b = 7.28

printf ("%5.2f\n", b);

It means in 5 character space with 2 decimal place accuracy print variable b.

Output : _5.28

So, format specifiers are used to tell the compiler about type of data during input & output.

```
#include<Stdio.h>

int main()
{
 float b = 5.28;
 int a = 5;
 char demo = 'A';
 float x = 1.999;

 printf("%5.2f\n", b);
 printf("a = %d\n", a);
```

-> Here %d is format specifier tells compiler 'a' variable contain integer value.

```
printf("demo = %c\n", demo);
printf ("x = %f\n", x);

return 0;
}
```

Constants

A constant is a value or variable that can't be changed in the program.

It means anything whose value is fixed i.e. which can't be changed or manipulated is known as constant.

There are two ways to define constant in C:

1. const Keyword
2. # define pre-processor

1 Const Keyword :- const keyword can be used with any variable to make the value of that variable constant so that it can't be changed later on in the program.

E.g :- const int demo = 5;

It means we have made a constant int variable, so now we can't change the value of 'demo' variable later on in your program.

2. # define preprocessor :- # define command is used to define any variable globally i.e. we can define any variable & can put some value in it so that its

value will not change in whole program.

E.g :- #define demo1 501

It means we have defined the value of demo1 as 501 & we can't change the value of demo1 variable later on in the program.

```
#include <stdio.h>
#define demo1 501

int main ()
{
 const int demo =5;

 printf ("value of demo is %d", demo);
 printf ("value of demo1 is %d", demo1);

 demo = 8;
 demo1 = 101;

 return 0;
}
```

Output

error :- assignment of read only variable 'demo'
demo = 8;
^

error - lvalue required as left operand of assignment
demo1 = 101
^

Escape Sequences:-

An Escape Sequences in C language is a sequence of characters.

It doesn't represent itself when used inside string literal or character.

It is composed of two or more character starting with backslash \.

Escape Sequence	Meaning
\a	Alarm / Beep
\b	Backspace
\n	New line
\t	Tab (Horizontal)
\v	Vertical Tab
\\	Backslash
\'	Single quote
\"	Double quote
\?	Question Mark
\0	Null

Comments in C language :-

Comments are used to make the code more understandable for programmer.

Comments are not executed by compiler.

2 Methods to write comment in C Language

1. Single Line Comment
2. Multi Line Comment

Single line Comment :- Single Line Comment are the comments which are created in Single line only i.e they occupy the space of Single line only.

we use // (double backslash to write single line Comment)

E.g // This is a Comment

Multi Line Comment :- Multi line comments are the comments which are created by using multiple lines i.e. they occupy more than one line in the program.

we use /* .... Comment ... */ for writing multi line comment.

if and Else in C language :-

C program executes sequentially i.e line after line but

we can change this pattern by entering some condition in it. so, if we put some condition then the flow of execution changes i.e conditions are checked & then executed accordingly.

Decision Control Instruction

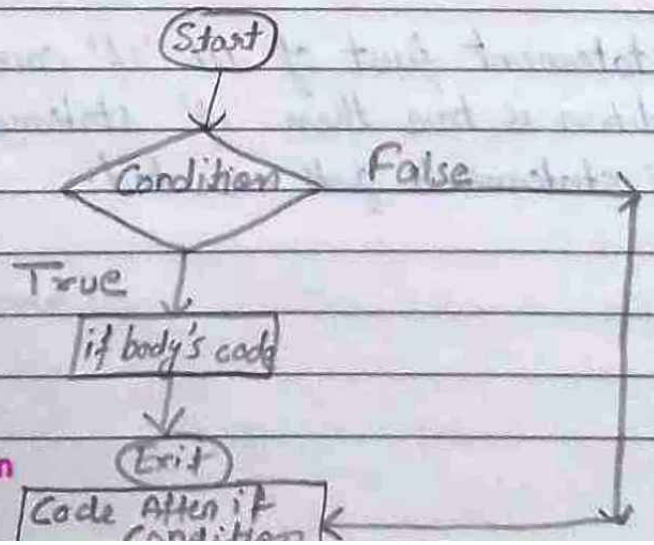
It is used to perform operations based on some condition i.e. checking condition first & then evaluating accordingly.

If & Else conditions are the parts of Decision Control Instruction.

Type of if statement

if statement
if-else Statement
if-else if statement or if else-if ladder
Nested if/else

if Statement:- In if statement condition is checked, & if it is true then 'if' statement is executed otherwise it is skipped & rest of the code is executed.



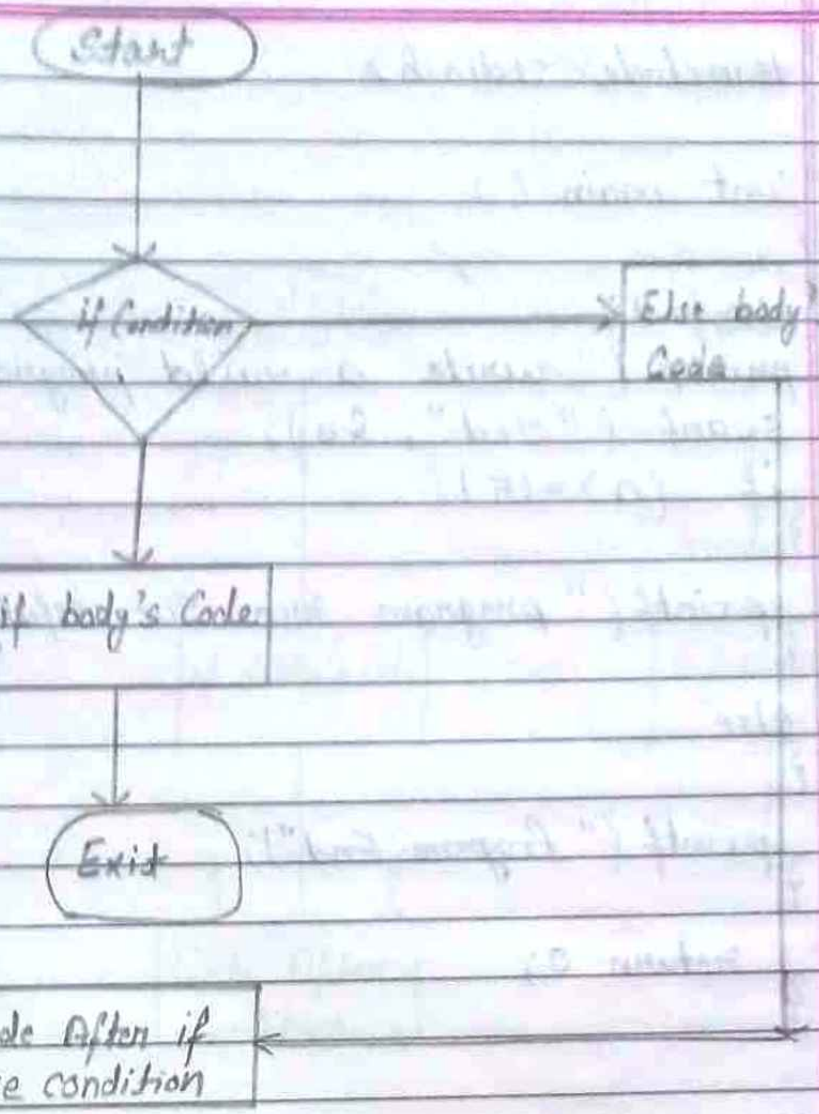
Syntax

```
if ( Condition )
{
 statement 1;
 statement 2;
 ....
}
```

```
#include <Stdio.h>

int main ()
{
  int a = 10
 if ( a >= 10 )
{
 printf (" Program Executed Successfully");
}
 printf ("program End :)");

 return 0;
}
```

if-else statement :-

In if-else statement first of all 'if' condition is checked & if the condition is true then 'if' statement gets executed otherwise 'else' statements gets executed.

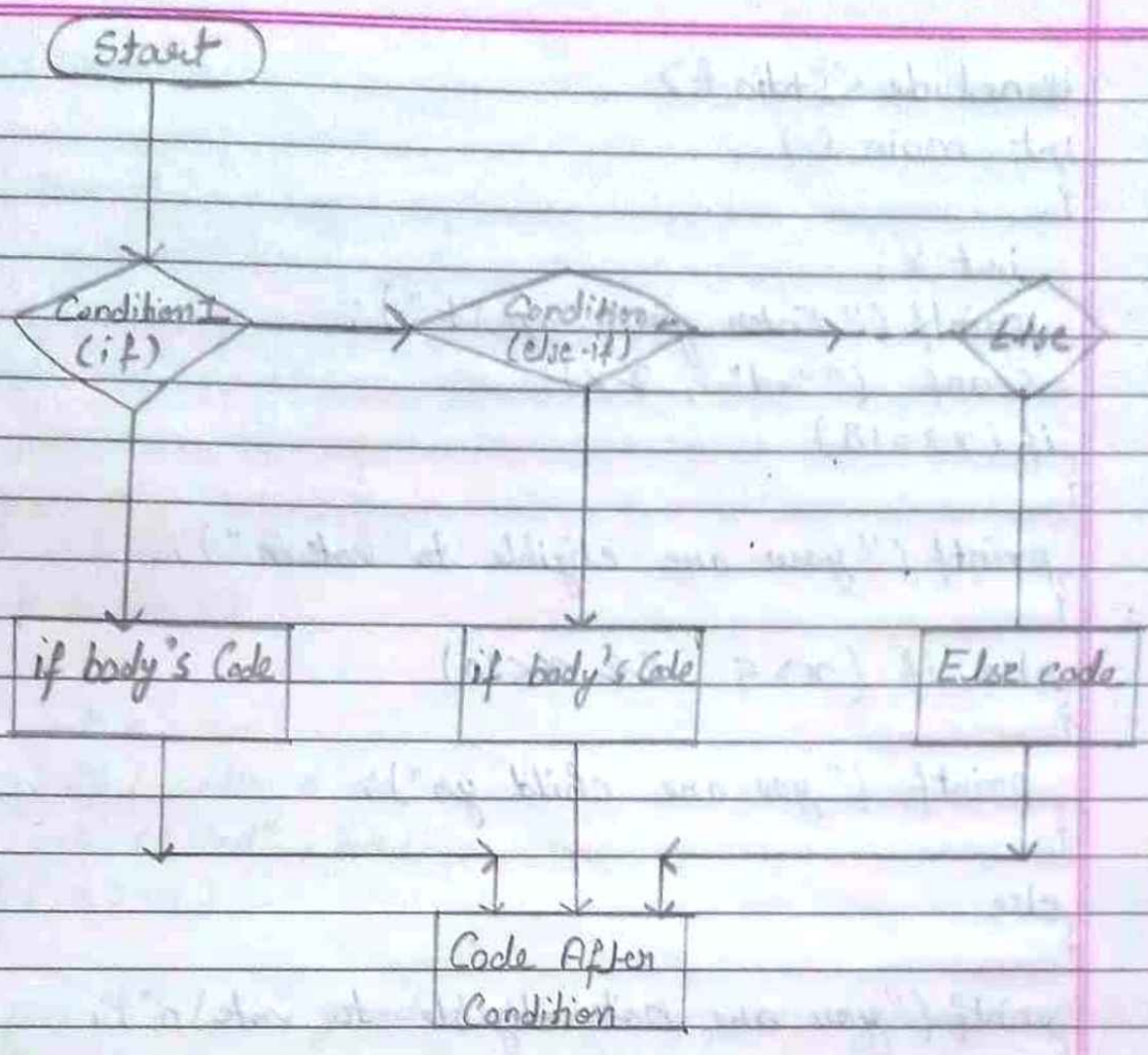


Syntax

```
if (Condition)
{
Statement 1;
Statement 2,
...
}
else
{
Statement 1;
...
}
```

```
#include <stdio.h>

int main ()
{

 int a;
printf (" write a vaild program ? \n");
scanf ("%d", &a);
if (a>=15)
{

printf(" program run Successfully");
}
else
{

printf (" Program End");
}

 return 0;
}
```

if - else if statement: In if-else if statements, if 'if' condition is true then 'if' statement gets executed & if it is false then 'else if' condition is checked & if it is true then 'else if' gets executed otherwise condition moves to next statement & if all condition are false then 'else' statement gets executed.



Syntax

```
if (condition)
{ statement;}
else if (Condition)
{
 Statement; }
else if (condition)
{
 statement; }
else
{
statement;
}
```

```
#include <stdio.h>
int main ()
{
  int x;
  printf (" Enter your age\t");
  scanf ("%d", &x);
if (x>=18)
{
printf ("you are eligible to vote/n");
else if (x>5 && x<10)
{
printf ("you are child yr");
else
{
printf ("you are not eligible to vote\n");
return 0;
}
```

Nested if-else :-

It means we can use if or else inside if or else i.e using if else statements inside if else statements. These are used when we want to check conditions inside any particular condition.

Syntax :-

```
if (condition)
{
Statement;
if (condition)
{
Statement; }
}


# include <stdio.h>
int main ()
{
  int x;
printf ("Enter a no. :");
scanf ("%d", &x);
if (x>15)
{
 printf (" No. greater entered by you is greater than 15 \n");
if (x<50)
{
 printf (" And it lie b/w 15 & 50 ");
}
}
else
{
printf (" Enter valid No. ");
}

return 0;
}
```

Loops in C

The versatility of the computer lies in its ability to perform the set of instruction repeatedly. This involves repeating some code in the program, either a specified number of times or until a particular condition is satisfied. Loop control instructions are used to perform this repetitive operation.

Three types of loop

For - loop
While loop
do- while loop

There are two kinds of loop

① Entry Controlled loops :- In entry controlled loops, the test condition is evaluated befor entering the loop body. The for loop & while Loop are an ex. of entry Controlled loop.

② Exit controlled Loop :- In exit Controlled loops, the test condition is tested at the end of of the loop. The loop body will execute at least once, whether the test Condition is true / false. The do-while loop is an example of an exit controlled loop.

For Loop :- A for loop is a repetition control statement that follows us to efficiently write a loop that will execute a specific no. of times. working / loop

(i) The initialization statement is executed only once; in this statement, we initialize a variable to some value.

(ii) In the second step, the test expression is evaluated. suppose the test expression is evaluated to true. In that case, the for loop keeps running & the test expression is re-evaluated, but if the test expression is evaluated to false, then the for loop terminates.

The loop keeps executing until the test expression is false. when the test expression is false, then the loop terminates.

While loop :-

While loop is called a pre-tested loop. The while loop allows code to be executed multiple times, depending upon a boolean condition that is given as a test expression. while studying for loop, we have seen that the number of iterations is known, whereas while loop are used in situations where we do not know the exact number of iterations of the loop. The while loop execution is terminated on the basis of the boolean (true / false) test Condition.

do - while Loop :- In do-while loops, the execution is terminated on the basis of the test Condition. The main difference b/w the do-while loop & while loop is that, in the do-while loop, the condition is tested at the end of the loop body, whereas the other two loops are entry controlled loops.

mycompanion

Note :- In do - while loop, the loop body will execute at least once irrespective of the test Condition.

What about Infinite loop

An infinite loop also known as an endless loop occurs when a Condition always evaluates to true. Usually, this is considered an error.

Sometimes, while executing a loop, it becomes necessary to jump out of the loop. For this, we will use the break statement or Continue statement.

break statement

When a break statement is encountered inside a loop whether it is a for loop or a while loop, the loop is terminated & the program continues with the statement immediately following the loop.

Continue statement :- Using a Continue statement in the loop will cause the control to go directly to the test - condition & then it will continue the loop process.

Switch case control

The control statement that allows us to make a decision effectively from the number of choices is called a Switch, or a switch case - default since these three Keywords go together

my companion

to make up the control statement the expression in switch returns an integral value, which is then compared with the different cases. switch executes the block of code, which matches the case value. If the value does not match with any of the cases, then the default block is executed. Syntax of Switch:

```
switch (integer expression)
{
 Case value 1:
 do this;
 Case value 2:
 do this;
 Case value 3:
 do this;
 default:
 do this;
}
```

Explanation of general form of Switch.

The expression following the switch can be an integer expression or a character expression. The case value 1, 2 are case labels that are used to identify each case individually. Remember that case labels should be different. if it is same, it may create a problem while executing a program. At the end of the case labels, we always use a colon (:). Each case is associated with a block. A block contains multiple statements that is grouped for a particular case.
Whenever the switch case is executed. the value of test-expression is compared with all cases present in switch statements. when the case found, the block of statements

my companion

associated with that particular case will execute. The break keyword indicates the end of a particular case. If we do not put the break in each case, then even though the specific case is executed, C's switch will continue to execute all the cases until the end is reached. The default case is optional. Whenever the expression's value is not matched with any of the cases inside the switch, then the default will be execute.

```
#include <stdio.h>

int main (){
int i = 9;

switch (i) {
Case 5:
 printf ("Value is 7");
 break;

 case 9:
 printf("value is 9");
 break;

 default:
 printf("value is not present");
 break;
}
```

Note:- It is not necessary to use break keyword after every case. When we do not want to terminate our case at that time, we will not use the break statement.

Nested Switch in C

we can also use nested switch statements i.e., switch inside another switch Also, the case constants of the inner & outer switch may have common values without any conflick.

Syntax:-

```
switch (expression 1)
{
case 1:
printf(" Switch Statement 1");

switch (expression 2)
{
Case 1:
printf(" Switch Statement 2");
break;

Case 2:
do this;
}
break;

Case 2:
Do this;
}
```

why do we need to use Switch case?
There is one problem with the if statement: the program's complexity increases whenever the number of if statements

increases. if we use multiple if-else statements in the program, the code might become difficult to read & comprehend. Sometimes it also confuses the developer who himself wrote the program.

Rules for Switch Statement

① The test expression of switch must be an int or char.

② The value of the case should be an integer or character.

③ Cases should be inside the switch statement.

④. Using the break keyword in the switch statement is not necessary.

⑤ The case labels values inside the switch should be unique

Difference b/w switch & if:-

Switch statements cannot evaluate float conditions, & the test expression can be an integer or a character, whereas if statements can evaluate float condition.

The switch statement cannot evaluate relational operators i.e they are not allowed in switch statements, whereas if statements can evaluate relational operators. Switch

cases in the switch can never have variable expression; for ex., we cannot write case a +3:

Do - While Loop :- A do while loop executes the statement inside the body of the do-while loop before checking the condition. so if a condition is false in the first place, then they do while would run once. A do-while loop is similar to a while loop, ~~execute~~ except the fact that it is guaranteed to execute at least one time.

Unlike for & while loops, which test the loop condition first, then execute the code written inside the body of the loop, the do-while loop checks its condition at the end of the loop. Syntax of do-while loop

```
do {
  (Statements);
} while (test Condition);
```

if the test condition returns true, the flow of control jumps back up to do, & the set of statements in the loop executes again. this process repeats until the given test condition becomes false.

How does the do-while loop work ?

First, the body of the do-while loop is executed once. Only then, the test Condition is evaluated.

if the test Condition returns true, the set of instruction inside the body of the loop is executed again, & the test Condition is evaluated.

this looping process goes on until the test Condition becomes

false.

if the test Condition returns false, then the loop terminates

following program will add num until the user enters Zero.

```
#include<Stdio.h>
int main ()
{
 int n, sum=0;
 do
 {
 printf ("Enter a number");
 scanf ("%d", &n);

    sum  += n;
 }
   while (n != 0);
   printf (" Sum is = %d", sum);

 return 0;
}
```

Diff. b/w While Loop & do-while Loop

while loop is executed when given test condition return true, whereas, do-while loop is executed for the first time irrespective the test condition is true or false, because the test is checked after executing while loop for

my companion

the first time.

```
main ()
{
 while (2 < 1)
 printf ("Hello World \n");
}
```

Since the condition fails the first time itself, the printf () statement will not get executed. Let's write the same code using do-while.

```
main ()
{
do
 {
  printf ("Hello World \n");
 }
  while (2 < 1);
}
```

the printf () statement would be executed once, since first the body of the loop is executed, & then the test Condition is evaluated.

```
#include <stdio.h>

int main ()
{
  int num, index = 0;
  printf ("Enter a number");
  scanf ("%d", & num);
```

```
do
{
  printf ("%d \n", index +1);

  index = index +1;
}
  while (index <num);

  return 0;
}
```

While Loop in C

while loop is also called as a pre-tested loop. A while loop allows code to be executed multiple times, depending upon a given Boolean (true or false) condition. The while loop is mostly used in the case where the number of iterations is not known. if the number of iteration is known, then we use for loop.

```
while (Condition test)
{
  // set of Statement
}
```

```
#include <stdio.h>
int main ()
{
  int i = 0;
  while (i <= 5)
{
```

```
printf ("%d\n", i);
i++;
}
return 0;
}
```

Explanation of program

We have intialized a variable i with value 0; This code will print from 0 to 4; hence the variable is initialized with value 0.

In a while loop, we have provided a Condition (i<=5) which means the loop will execute the body until the value of i becomes 5. After that, the loop will be terminated.

in the body of a loop, we have print function to print our number & an increment operation (i ++) to increment the value per execution of a loop. This process will Continue until the value becomes 5 & then it will print the number & then terminate the loop.

Properties of while loop :-

A Conditional expression written in the brackets of while is used to check the Condition. The set of statements defined inside the while loop will execute until the given condition returns false.

The condition will return 0 if it is true. The

Condition will be false if it returns any non-Zero number.

In the while Loop, we cannot execute the loop until we do not Specify the Condition expression

It is possible to execute a while loop without any Statement. This will give no error.

We can have multiple condition expressions in a while Loop.

Braces are optional if the body loop contains only one Statement.

```
#include <Stdio.h>

int main ()
{
  int i = 0;
  while <i < 50)
  {
    printf ("%d\n", i);
     i = i+1;
  }

  return 0;
}
```

For Loop

The "For" Loop is used to repeat a Specific code until a Specific condition is satisfied. The for-loop statement is very specialized. We use for a loop when we know the number of iterations we want, whereas when we do not know about the number of iterations, we use while loop. Syntax of loop.

```
for ( initialize counter; test counter; increment/decre
        counter; )
{
// set of instruction
}
```

initialize counter : It will initialize the loop counter value, i.e, i=0

test counter :- It verifies whether the condition is true.

Increment / decrement :- Incrementing / decrementing the counter.

set of Statements :- Execute the set of statement

```
#include <stdio.h>
int main()
{
  int num = 10;
  int i;
for ( i=0; i< num; i++)
```

```
{
}printf ("%d", i);

}return 0;
```

Explanation :- First, the initialization expression will initialize loop variables. The expression i=0 executes once when the loop starts. Then the condition i<num is checked if it is true, then the statements inside the body of the loop are execute. After executing the statements inside the body, the control of the program is transferred to increment the variable by 1 (i++). The expression i++ modifies the loop variables. Then the condition i<num is evaluated again. if the condition is still true, the body of the loop will execute once more. the for loop terminates when i< num before false.

we can have for loop inside another for a loop. This is known as nested for loop. Similarly, while loop & do while loop can also be nested.

```
for (initialization ; test Counter ; increment) {
for (initialization ; test Condition ; increment) {

    // Set of Statements
}
    // Set of Statements
}
```

Note :- There is no rule that a loop must be nested inside its own type. For loop can be nested the while loop & vice versa.

~~Why~~ Why we prefer For Loops ?

It is clear to a developer exactly how many times the loop will execute. So, if the developer has to dry run the code, it will become easier. Also, the Syntax of the for loop is almost the same as other programming languages like C++ & java.

```
#include <Stdio.h>

int main ()
{
 printf ("Hello World\n");
 int i, j = 0;

for (i=0 ;  ; )
{
 printf ("%d %d\n", i, j);

  i++; j++;
}
 return 0;
}
```

Break & Continue

Break statement :-

Break statement is used to break the loop or switch case statements execution & brings the control to the next block of Code after loop or switch case.

Break statements are used to bring the program control out of the loop.

The break statement is used inside loops or switch statement in C language.

Continue Statement :-

The continue statement is used inside loops in C language. When a Continue statement is encountered inside the loop, controls jumps to the begining of the loop for next iterations, skipping the execution of statement inside the body of loop after continue statement

It is used to bring the control to the next iteration of the loop.

The Continue statement skips some code inside the loop & continue with the next iteration.

It is mainly used for a condition so that we can skip some lines of code for a particular condition

It forces next iteration in loop i.e. as break terminates the loop but Continue forces the next iteration of the loop.

```
#include <stdio.h>

int main ()
{
 printf (" Hello World \n");

 int i, age;

 for (i=0, i< 10 ; i++)
 {
  printf ("%d \n Enter your age \n", i);
   scanf ("%d", &age);

  if (age > 10)
  {
    break;
  }
  if (age > 15 )
  {
    continue;
  }
  printf (" we have not come across continue statement);
 }
  return 0;
}
```

Goto Statement

A goto statement in C programming languages provides an unconditional jump from the goto to a labeled statement in the same function.

NOTE :- Use of goto statements in highly discouraged or avoided in any programming language because it makes difficult to trace the control flow of a program to fellow programmers, making the program hard to understand & hard to modify or manipulate. Any program which uses goto can be modified to avoid goto statements.

These are also called 'jump statement'.

It is used to transfer the control to a predefined label.

It's use is avoided since it causes confusion for the fellow programmers in understanding code.

goto statement is preferable when we need to break multiple loops using a single statement at the same time.

Syntax.

```
goto label;
  ...
  ...
  ...
label : statement;
```

```
#include <stdio.h>
int main ()
{
  label:
  printf (" we are inside label");
   goto end;
   printf ("Hello World\n");
   goto label;
    end:
   printf (" we are at end");

int num;
for (int i = 0; i < 8; i++)
{
  printf ("%d\n", i);
for (int j = 0; j < 8; j++)
{
 printf ("Enter the number. enter 0 to exit\n");
 scanf ("%d", &num);
 if (num == 0){
   goto end;
 }
 }
 }
 end:

 return 0;
}
```

Type Casting

Type casting is a way to convert one data type into another one. It is also known as data conversion or type conversion.

C Languages Provides two method of type casting

① Implicit type casting
② Explicit type casting

Implicit type casting :- Implicit type casting means conversion of data types from one to another without losing its original meaning or sense.

In this type casting program automatically converts the variable from one data type to other. It follows strict rules in converting data type of variables as it always converts lower data types to higher ones.

```
#include <stdio.h>

int main ()
{
  short a = 10; //initializing variable of short data type
  int b; // declaring int variable 'b'.
  b = a; // Implicit type casting
   printf ("%d\n", a);
   printf ("%d\n", b);
```

Explicit type casting :- Explicit type casting means conversion

of data type from one to another forcefully by programmer. It is user defined conversion.

Syntax:

(type_name) value/expression

```
#include <stdio.h>
int main ()
{
  int a;
  float b;
  char c;

printf ("Enter the value of a\n");
scanf ("%d", &a);
printf (" A is %d\n", a);

printf ("Enter the value of b\n");
scanf ("%f", &b);
printf ("B is %d\n", (int) b);

printf (" Type a character : \n");
scanf ("%c", &c);

printf (" character is %d", (int) c);

return 0;

}
```

```
#include <stdio.h>
int main ()
{
  int a = 3;
  float n = (float) 54/5
  printf("The value of a is %f\n", b);

  return 0;
}
```

Functions in C

Functions are used to divide a large C program into smaller pieces.

Function can be called multiple or ~~ser~~ several time to provide reusability & modularity to the C program

Functions are called as procedure or subroutines.

It is a piece of code to accomplish certain operation.

Function is nothing but a group of code put together & given a name & it can be called anytime without writing the whole code again & again in a program.

Syntax of Function :

```
return_type function_name (data type parameter 1, data_type
                                parameter 2)
{
      // Code to be executed
}
```

my companion

Advantages of Functions

we can avoid rewriting same logic or code through functions.

we can divide the work among programmers using functions.

we can easily debug or can find bugs in any program using functions.

Function Aspects :

1. Declaration
2. Defination
3. Call

A function is declared to tell the compiler about its existence.

A function is defined to get some task done. (It means when we define function we write whole code of that function. In this actual implementation of function is done.)

A function is called in order to be used.

Types of Functions :

Library Functions :- These are pre-defined functions in C language. These are the functions which are included in C header files.

E.g. printf( ), Scanf( ) etc.

User defined Functions - ~~which~~ Functions created by programmer to reduce complexity of a program i.e. these are the functions which are created by user or programmer.

E.g. Any function created by programmer.

C Functions

Library Functions

User Defined Functions

NOTE :- Every C Program must contain one function i.e. main( ) function as execution of every C program starts from main( ) function.

4 ways define function

Without Arguments & ~~wh~~ without return value.
without arguments & with return value.
with arguments & without return value.
with arguments & with return value