UNIT-I&II

C PROGRAMMING IBCA A

What is an Alogorithm? 2m

An Algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any valid input in a finite amount time.

OR

An algorithm is a precise specification of a sequence of instructions to be carried out in order to solve a given problem.

OR

An Algorithm is a step-by-step approach to solve a given problem in finite number of steps.

What are the Properties (characteristics) of an Algorithm: 3m

- It should be simple.
- It may accept zero or more inputs.
- It should be precise with no ambiguity.
- It should have finite number of steps.
- It should produce at least one output

What is the Need for an Algorithm? 5m

1. Efficient Communication: Since an algorithm is written in English like language, it seems to be the better way of communicating the logic. Ie.., a procedure to solve a given problem to the users.
2. Efficient Analysis: With the help of an algorithm problem can be analysed effectively. There can be number of algorithms for solving a given problem. Based on the analysis, a suitable and efficient algorithm can be selected.
3. Proper Documentation: The selected algorithm has to be documented, which will be useful to identify the logical errors.
4. Easy and efficient coding: the algorithm act as blue print during program development. It will be easier to write efficient program using high level language.
5. Program Maintenance: Maintaining the software or set of programs becomes easy.

How algorithm is a technology ?

- Algorithms are just like a technology. We all use latest and greatest processors but we need to run implementations of good algorithms on that computer in order to properly take benefits of our money that we spent to have the latest processor.

- Let’s make this example more concrete by pitting a faster computer(computer A) running a sorting algorithm whose running time on n values grows like n2 against a slower computer (computer B) running asorting algorithm whose running time grows like n lg n.
- They eachmust sort an array of 10 million numbers. Suppose that computer A executes 10 billion instructions per second (faster than anysingle sequential computer at the time of this writing) and computer B executes only 10 million instructions per second, so that computer A is1000 times faster than computer B in raw computing power
-

Computer A (Faster)	Computer B(Slower)
Running time grows like n^2.	Grows innlogn.
10 billion instructions per sec.	10million instruction per sec
$2n^2$ instruction.	50 nlogn instruction.

-

Time taken= $\frac{2\times(10^7)^2}{10^{10}} = 20{,}000$	$\frac{50\times10^7 \times\log 10^7}{10^7} \approx 1163$
It is more than 5.5hrs	it is under 20 mins.

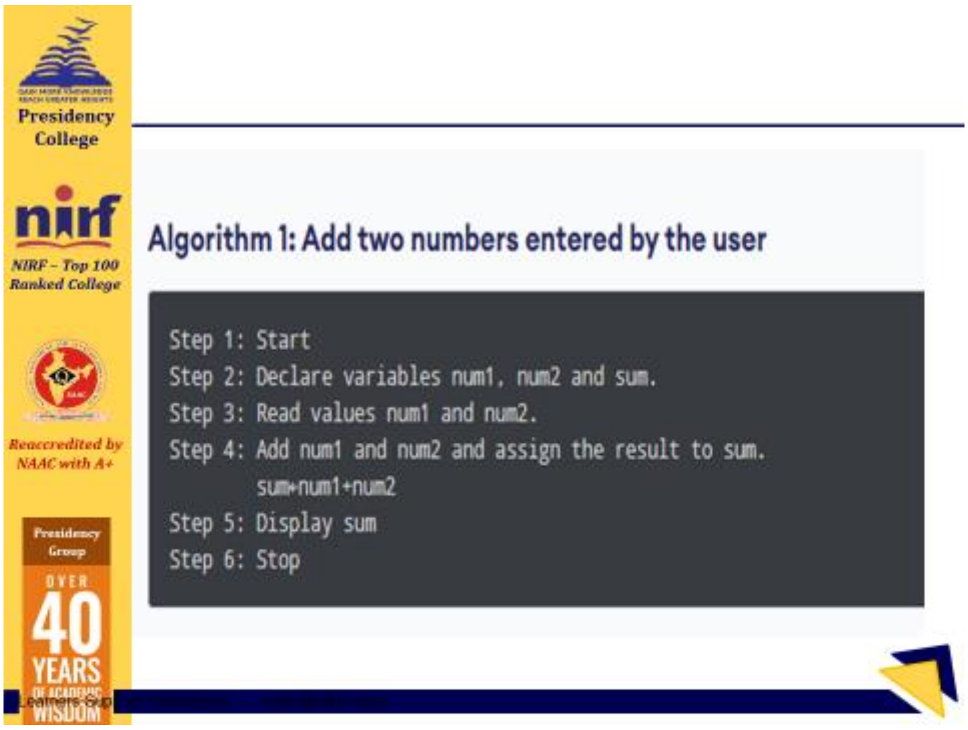
So choosing a good algorithm (algorithm with slower rate of growth) as used by computer B affects a lot.

Explain Algorithm Analysis

- Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem.
- Most algorithms are designed to work with inputs of arbitrary length. Analysis of algorithms is the determination of the amount of time and space resources required to execute it.
- Usually, the efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps, known as **time complexity**, or volume of memory, known as **space complexity**.
- **The Need for Analysis**
  Analysis of algorithm is the process of analyzing the problem-solving capability of the algorithm in terms of the time and size required

However, the main concern of analysis of algorithms is the required time or performance
Generally, we perform the following types of analysis –

- **Worst-case** − The maximum number of steps taken on any instance of size **a**.
- **Best-case** − The minimum number of steps taken on any instance of size **a**.
- **Average case** − An average number of steps taken on any instance of size **a**.
- **Amortized** − A sequence of operations applied to the input of size **a** averaged over time.



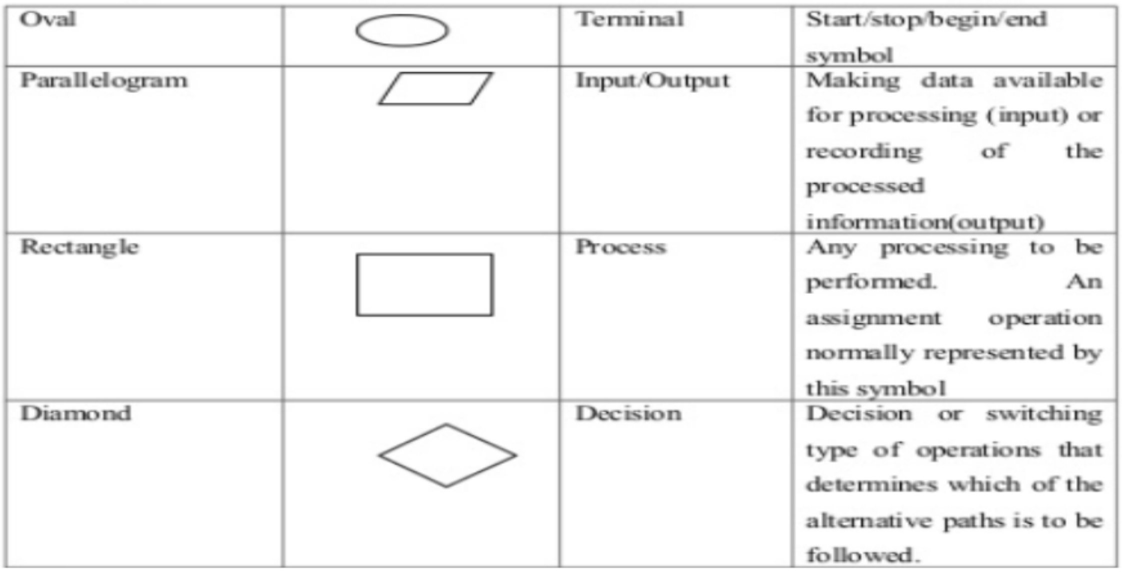
Algorithm 1: Add two numbers entered by the user

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        sum←num1+num2
Step 5: Display sum
Step 6: Stop
```

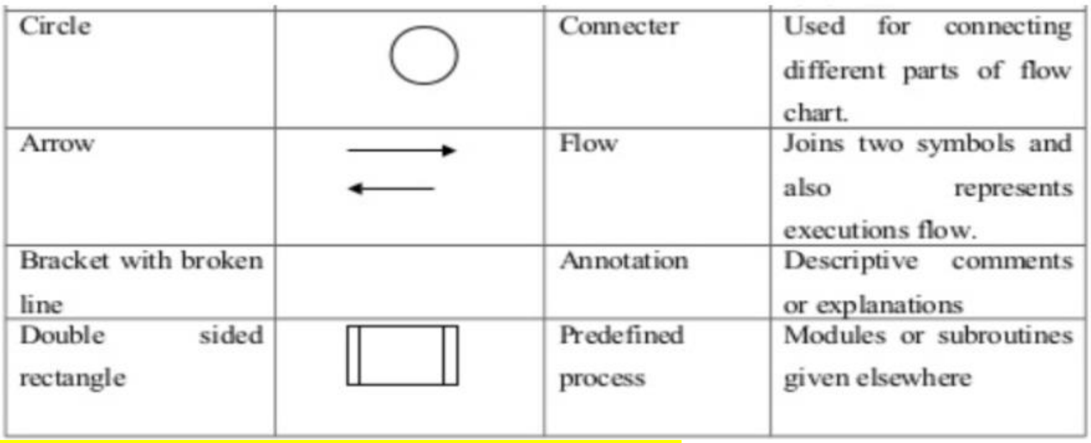
What are the disadvantages of an Algorithm. 2m
1. Developing algorithms for large and complex problems is time consuming and difficult to understand.
2. Understanding complex logic is tedious.
What is a flow chart? 2m
A flowchart is a pictorial representation of an algorithm or a stepwise process, showing the steps of boxes of various kinds and their order by connecting these with arrows.
Flowcharts are used in designing or documenting a process or program.

What are the symbols used in Flowchart? Explain. 5m

Following are the standard symbols used in drawing flowcharts. (see in next page)



Oval	⬭	Terminal	Start/stop/begin/end symbol
Parallelogram	▱	Input/Output	Making data available for processing (input) or recording of the processed information(output)
Rectangle	▭	Process	Any processing to be performed. An assignment operation normally represented by this symbol
Diamond	◇	Decision	Decision or switching type of operations that determines which of the alternative paths is to be followed.



Circle	○	Connecter	Used for connecting different parts of flow chart.
Arrow	→ ←	Flow	Joins two symbols and also represents executions flow.
Bracket with broken line		Annotation	Descriptive comments or explanations
Double sided rectangle	▯	Predefined process	Modules or subroutines given elsewhere

What are the advantages and disadvantages of flow chart? 5m

Advantages of Flow chart

1. Communication: Flow Charts are better way of communicating the logic of a system to all concerned.
2. Effective Analysis: With the help of flow chart, problem can be analyzed in more effective way.
3. Proper Documentation: Program flow charts serve as a good program documentation,
4. Efficient Coding: The flow charts act as a guide during the systems analysis and program development phase.
5. Proper Debugging: The flow chart helps in debugging(error checking) process.
6. Efficient Program Maintenance: The maintenance of operating program becomes easy.

Disadvantages

1. Alterations and Modifications are difficult.
2. As the flow chart symbols cannot be typed, reproduction of flow chart becomes a problem.
3. Time consuming and not suitable for lengthy programs.

What is structured programming? 2m

Designing and solving a programming problem using sequence, selection and iteration control instructions are called structured programming.

Explain the basic control structures of Structured programming. 5m

- Sequence
- Selection
- Iteration

selection structure or decision structure- With this structure, question is asked and, depending on the answer action is taken. Then, no matter which path it takes, it will continue with the next task.

```
ex: if condition then
    statement1;
   else
    statement2;
    statement3;   .....
```

Iteration - Also called repition. Code is executed several times based on condition given.

```
ex: while condition
     {
    statement1;
    statement2;
    }

   for(i=0;i<n;i++)
   {
    statement1;
    statement2;
   }
```

INTRODUCTION TO C

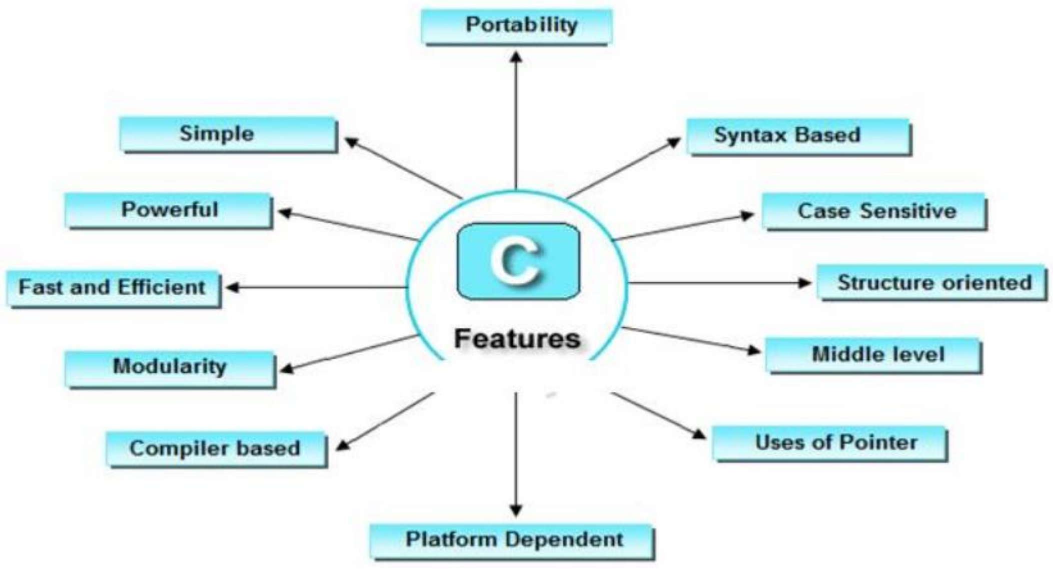
Who has invented C langauge and which year? 2m

C is a computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX Operating System. C is a simple and structure oriented programming language.

C is also called mother Language of all programming Language. It is the most widely use computer programming language, This language is used for develop system software and Operating System. All other programming languages were derived directly or indirectly from C programming concepts. Here we discuss complete C Tutorial in simple and easy way.

Explain the features of C. 5m

It is a very simple and easy language, C language is mainly used for develop desktop based application. All other programming languages were derived directly or indirectly from C programming concepts. This language have following features;

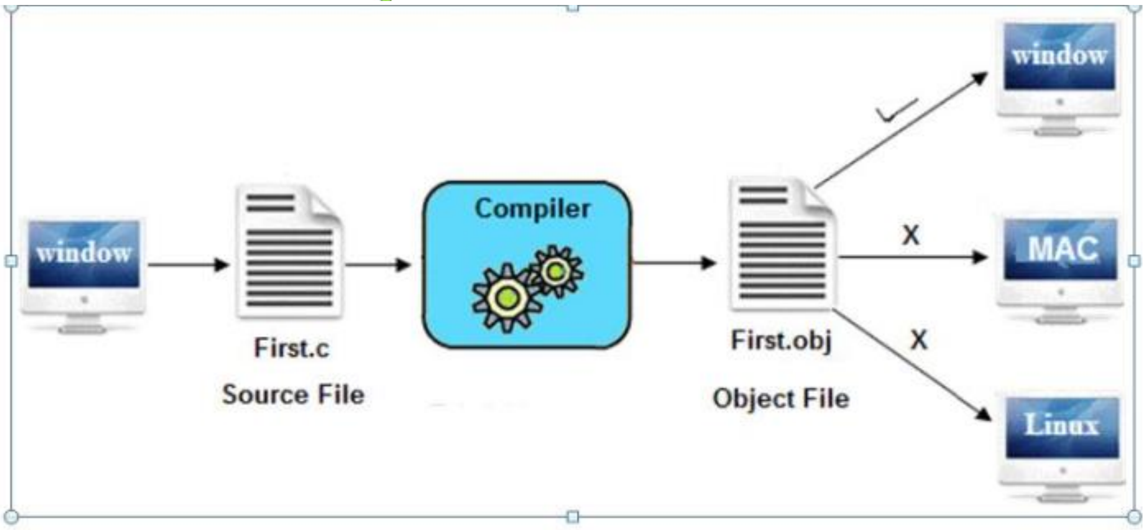


- Simple
- Portability
- Powerful
- Platform dependent
- Structure oriented
- Case sensitive
- Compiler based
- Modularity
- Middle level language
- Syntax based language
    - Use of pointers

Simple

Every c program can be written in simple English language so that it is very easy to understand and developed by programmer.
Platform dependent

C is platform dependent programming language. It can run on any Operating System.



Note: .obj file of C program is platform dependent.

Portability

When we write and compile any C program on window operating system that program easily run on other window based system.

In C Language .Cfile contain source code, we can edit also this code. .exe file contain application, only we can execute this file.

Powerful

C is a very powerful programming language, it have a wide variety of data types, functions, control statements, decision making statements, etc.
Structure oriented

Structure oriented programming language aimed on clarity of program, reduce the complexity of code. using this approach code is divided into sub-program/subroutines.

Modularity

It is concept of designing an application in subprogram that is procedure oriented approach. In c programming we can break our code in subprogram.

For example we can write a calculator programs in C language with divide our code in subprograms.

```
void sum()
{
 .....
 .....
}
void sub()
{
 .....
 .....
}
```

Case sensitive

It is a case sensitive programming language. In C programming 'break and BREAK' both are different.

If any language treats lower case latter separately and upper case latter separately than they can be called as case sensitive programming language [Example c, c++, java, .net are sensitive programming languages.] other wise it is called as case insensitive programming language [Example HTML, SQL is case insensitive programming languages].

Middle level language

C programming language can supports two level programming instructions with the combination of low level and high level language that's why it is called middle level programming language.

Compiler based

C is a compiler based programming language that means without compilation no C program can be executed. First we need compiler to compile our program and then execute.

Syntax based language

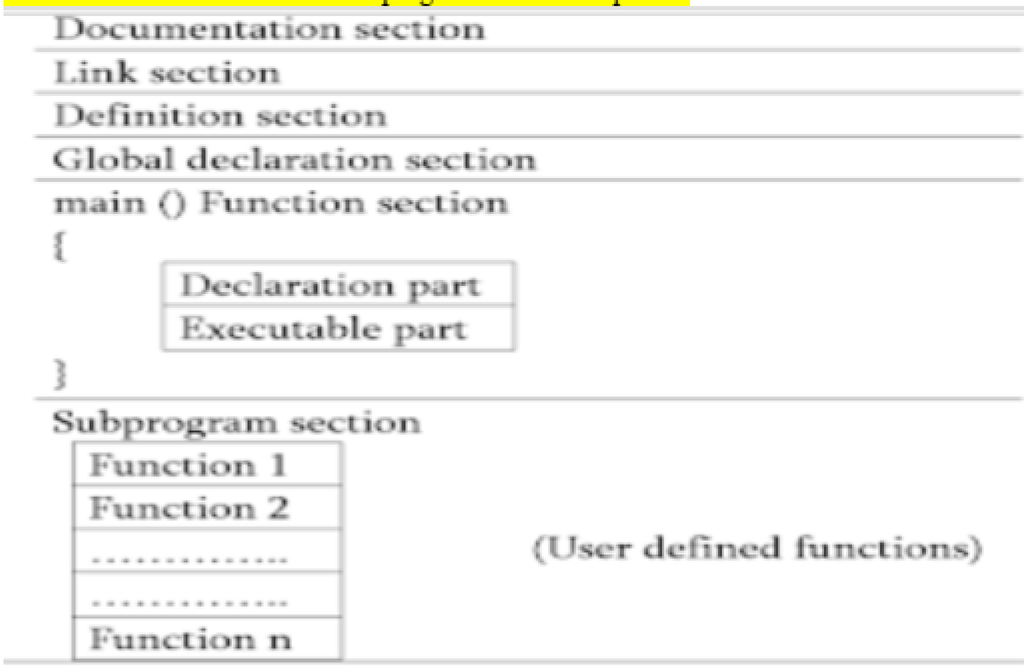
C is a strongly tight syntax based programming language. If any language follow rules and regulation very strictly known as strongly tight syntax based language. Example C, C++, Java, .net etc. If any language not follow rules and regulation very strictly known as loosely tight syntax based language.

Example HTML.

Efficient use of pointers

Pointers is a variable which hold the address of another variable, pointer directly direct access to memory address of any variable due to this performance of application is improve. In C language also concept of pointer are available.

What is the basic structure of a C program? with example 5m



1. **Documentation section**: It consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

2. **Link section**: It provides instructions to the compiler to link functions from the system library such as using the #include directive.

3. **Definition section**: The definition section defines all symbolic constants such using the #define directive.

4. **Global declaration section**: There are some variables that are used in more than one function. It is declared outside all functions.

5. **main () function section**: Every C program must have one main function section. This section contains two parts;

   a) declaration part

   b) executable part

   a) **Declaration part**: The declaration part declares all the variables used in the executable part.

   b) **Executable part**: There is at least one statement in the executable part. These two parts must appear between the opening and closing braces.

6. **Subprogram section:** If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function.

```
#include <stdio.h>                  //LINKAGE SECTION
#define  PI 3.14          //DEFINITION SECTION
Float r;            //GLOBAL DECLARATION

void main()    //MAIN SECTION
{
        float result;                 //DECLARATION SECTION
        clrscr();
        printf("Enter radius");
        scanf("%d",r);

    result = areafun(r);
    printf("area is %f",  result);
}

    float areafun(int r)                 //SUB PROGRAM SECTION
    {
            float area;
            area = PI * r * r;
            return (area);
    }

}
```

Character Set In C

Characters are used in forming either words or numbers or even expressions in C programming.

Characters in C are classified into **4 groups**:

Letters - In C programming, we can use both uppercase and lowercase letters of English language.

Uppercase Letters: A to Z

Lowercase Letters: a to z

Digits - We can use decimal digits from 0 to 9.

Special Characters - C Programming allows programmer to use following special characters:

comma , slash / period . backslash \ semicolon ; percentage % colon :

left and right brackets [ ] question mark ? left and right parenthesis ( )

White Spaces - In C Programming, white spaces contains:

Blank Spaces, Tab , Carriage Return ,New Line

Letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z
Digits	0 1 2 3 4 5 6 7 8 9
Special Characters	` ~ ! @ # % ^ & * ( ) _ - + = | { } [ ] : ; ' " / ? < > . ,
White Spaces	Blank space, tab, carriage return, new line

Identifiers

Identifiers are user-defined names of variables, functions and arrays. It comprises of combination of letters and digits. In C Programming, while declaring identifiers, certain rules have to be followed viz.

- It must begin with an alphabet or an underscore and not digits.
- It must contain only alphabets, digits or underscore.
- A keyword cannot be used as an identifier
- Must not contain white space.
- Only first 31 characters are significant.
- Let us again consider an example

```
int age1;
float height_in_feet;
```

Here, age1 is an identifier of integer data type.

Similarly height_feet is also an identifier but of floating integer data type,

Note: Every word in C program is either a keyword or an identifier.

C TOKENS:

C tokens are the basic buildings blocks in C language which are constructed together to write a C program.

Each and every smallest individual units in a C program are known as C tokens.

C tokens are of **six types**. They are,

Keywords	(eg: int, while),
Identifiers	(eg: main, total),
Constants	(eg: 10, 20),
Strings	(eg: “total”, “hello”),
Special symbols	(eg: (), {}),

Operators (eg: +, /,-,*)

Example:

```
void main()
{
  int x, y, total;
  x = 10, y = 20;
  total = x + y;
  printf ("Total = %d \n", total);
}
```

main – identifier
{,}, (,) – delimiter
int – keyword
x, y, total – identifier
main, {, }, (,), int, x, y, total – tokens

Write a note on Comments in C. 2m

Generally Comments are used to provide the description about the Logic written in program.

Comments are not display on output screen.

When we are used the comments, then that specific part will be ignored by compiler.

In 'C' language two types of comments are possible

- Single line comments
- Multiple line comments

Single line comments

Single line comments can be provided by using / /....................
Multiple line comments

Multiple line comments can be provided by using /*......................*/

Note: When we are working with the multiple line comments then nested comments are not possible.

EXAMPLE

```
// header files
  #include<stdio.h>
#include<conio.h>

void main()
{
// variable declaration
int a,b,c;
a=10;
b=20;
c=a+b;
printf("Sum= %d",c);
getch();
}
```

Nested Comments are not possible, that means comments within comments

```
void main()
{
/*
/*      comments    */
*/
}
```

Comments can be splits over more than one line.

```
void main()
{
/* main
   function
   body part
*/
}
```

Comments are not case sensitive.

```
void main()
{

/*  MAIN Function BODY    */

}
```

Single line comments start with "//"

```
void main()
{

// Single line comment

}
```

What are Keywords in C? 2m

Keyword is a predefined or reserved word in C library with a fixed meaning and used to perform an internal operation. C Language supports 32 keywords.

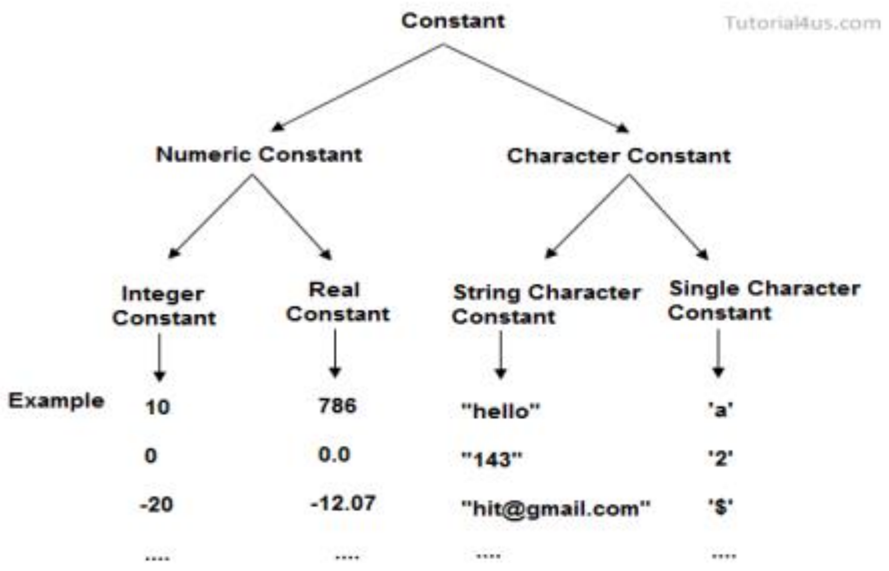
Every Keyword exists in lower case latter like auto, break, case, const, continue, int etc.

32 Keywords in C Language

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

Explain the different Types of Constant in C

It is an identifier whose value can not be changed at the execution time of program. In general constant can be used to represent as fixed values in a C program. Constants are classified into following types.



If any single character (alphabet or numeric or special symbol) is enclosed between single cotes ' ' known as single character constant.

If set of characters are enclosed between double quotes " " known as string character constant.

How to declare constant in C language. 2m

const keyword are used for declare a constant.

Syntax

```
const int height = 100;
```

Example

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

void main()
{
const int a=10;
printf("%d",a);
a=20; // gives error you can't modify const
getch();
}
```

What are SYMBOLIC CONSTANTS IN C ? 5m

- symbolic constant is name that substitute for a sequence of character that cannot be changed.
- The character may represent a numeric constant, a character constant, or a string.
- When the program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence.

Example :

```
#define FLAG 1
#define PI 3.1415
```

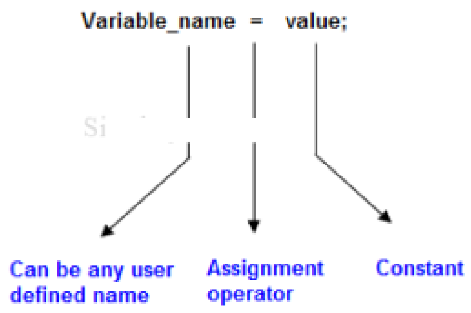
Here,FLAG and PI are symbolic constants.

```
/* program illustrating use of declaration,assignment of value to variables.also explains how to
use symbolic constants.
program to calculate area and circumference of a circle */
#include<stdio.h>
#include<conio.h>
#define PI 3.1415    /* NO SEMICOLON HERE */
void main ()
{
        float rad = 5;     /*DECLARATION AND ASSIGNMENT*/
        float area,circum;     /* DECLARATION  OF VARIABLE*/
        area=PI*rad*rad;
        circum=2*PI*rad;
        printf("AREA OF CIRCLE = %f\n",area);
        printf("CIRCUMFERENCE OF CIRCLE =%f\n",circum);
        getch();
        clrscr();
}
OUTPUT :
AREA OF CIRCLE =78.537498
CIRCUMFERENCE OF CIRCLE =31.415001
```

What is a Variable ? Variable is an identifier which holds data or another one variable. It is an identifier whose value can be changed at the execution time of program. It is used to identify input data in a program.

Syntax:

Variable_name = value;



Rules to declare a Variable

To Declare any variable in C language you need to follow rules and regulation of C Language, which is given below;

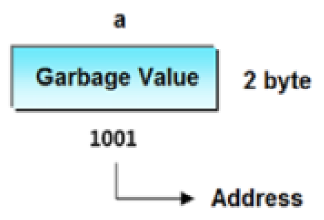
- Every variable name should start with alphabets or underscore ( ).
- No spaces are allowed in variable declaration.
- Uppercase characters are distinct from lowercase characters
- Except underscore (_) no other special symbol are allowed
- Maximum length of variable is 30 characters depend on compiler and operation system.
- Every variable name always should exist in the left hand side of assignment operator

(invalid -> 10=a; valid -> a=10;).

- keyword should not be used as variable name.
-

Variable declarations

This is the process of allocating sufficient memory space for the data in term of variable.
Syntax:
Datatype variable_name; int a;



Garbage value can be any value given by system and that is no way related to correct programs.

This is a disadvantage of C programming language and in C programming it can overcome using variable initialization.

Variable initialization

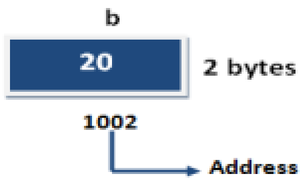
It is the process of allocating sufficient memory space with user defined values.

Syntax

Datatype nariable_name=value;

Example

int b = 30;



Variable assignment

It is a process of assigning a value to a variable.

Syntax

Variable_Name = value

Example

int a= 20;

int b;

Example

b = 25; // --> direct assigned variable

b = a; // --> assigned value in term of variable

b = a+15; // --> assigned value as term of expression

What are th difference between Local variable and Global variable? Explain with an example. 5m

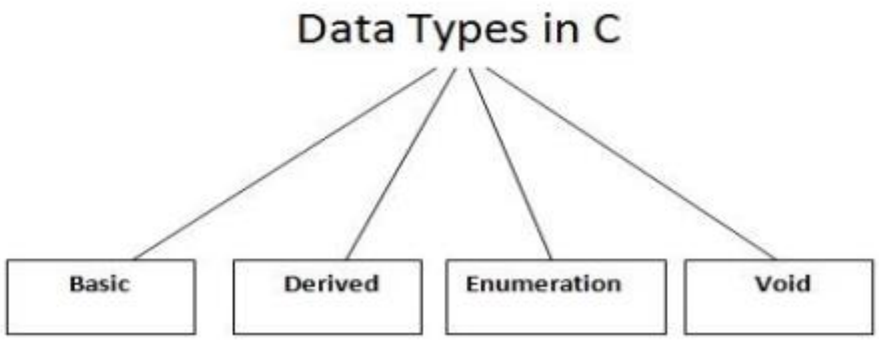
Global variables are defined outside of all the functions, generally on top of the program. The global variables will hold their value throughout the life-time of your program.

Local variable - A local variable is declared within the body of a function or a block. Local variable only use within the function or block where it is declare.

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

int a;   // global variable
void main()
{
int b;    // local variable
a=10, b=20;
printf("Value of a : %d",a);
printf("Value of b : %d",b);
getch();
}
```

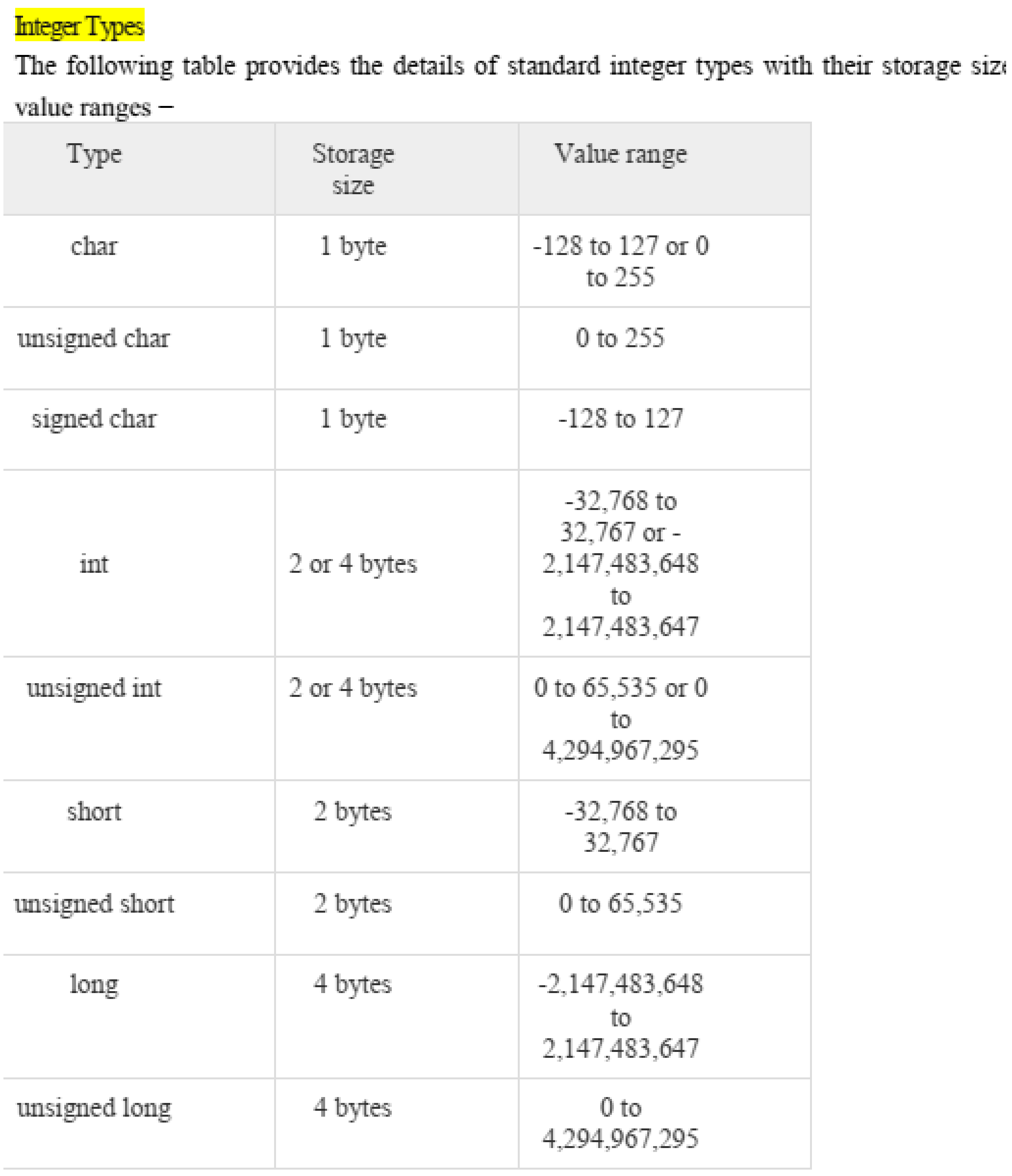
Explain various Data Types in C. 5m

A data type specifies the type of data that a variable can store such as integer, floating, character etc.



There are 4 types of data types in C language.

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void



Integer Types

The following table provides the details of standard integer types with their storage siz
value ranges –

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Primary Data Types:

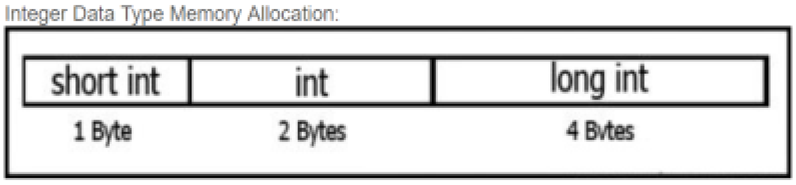
Integer Data Types:
Integers are whole numbers with a range of values, range of values are machine dependent. Generally an integer occupies 2 bytes memory space and its value range limited to -32768 to +32767 (that is, -215 to +215-1). A signed integer use one bit for storing sign and rest 15 bits for number.

To control the range of numbers and storage space, C has three classes of integer storage namely short int, int and long int. All three data types have signed and unsigned forms. A short int requires half the amount of storage than normal integer. Unlike signed integer, unsigned integers are always positive and use all the bits for the magnitude of the number. Therefore, the range of an unsigned integer will be from 0 to 65535. The long integers are used to declare a longer range of values and it occupies 4 bytes of storage space.

Syntax:
```
int <variable name>;

int num1;
short int num2;
long int num3;
```

Example: 5, 6, 100, 2500.

Integer Data Type Memory Allocation:



Floating-Point Types

The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision –

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

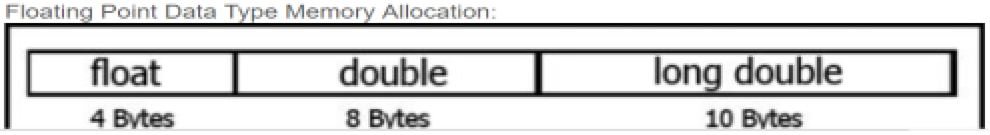
Floating Point Data Types:
The float data type is used to store fractional numbers (real numbers) with 6 digits of precision. Floating point numbers are denoted by the keyword float. When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision and takes double space (8 bytes) than float. To extend the precision further we can use long double which occupies 10 bytes of memory space.

Syntax:
float <variable name>;

float num1;
double num2;
long double num3;

Example: 9.125, 3.1254.

Floating Point Data Type Memory Allocation:



float	double	long double
4 Bytes	8 Bytes	10 Bytes

Character Data Type:
Character type variable can hold a single character and are declared by using the keyword char. As there are singed and unsigned int (either short or long), in the same way there are signed and unsigned chars; both occupy 1 byte each, but having different ranges. Unsigned characters have values between 0 and 255, signed characters have values from –128 to 127.

Syntax:
char <variable name>;

char ch = 'a';

Example: a, b, g, S, j.

Void Type:

The void type has no values therefore we cannot declare it as variable as we did in case of integer and float. The void data type is usually used with function to specify its type.

Explain various Operators in C. 10m

Operator is a special symbol that tells the compiler to perform specific mathematical or logical Operation.

- Arithmetic Operators
- Relational Operators
- Logical Operators

- Bitwise Operators
- Assignment Operators
- Ternary or Conditional Operators

Arithmetic Operators

Given table shows all the Arithmetic operator supported by C Language. Lets suppose variable Ahold 8 and B hold 3.

Operator	Example (int A=8, B=3)	Result
+	A+B	11
-	A-B	5
*	A*B	24
/	A/B	2
%	A%4	0

Relational Operators

Which can be used to check the Condition, it always return true or false. Lets suppose variable Ahold 8 and B hold 3.

Operators	Example (int A=8, B=3)	Result
<	A<B	False
<=	A<=10	True
>	A>B	True
>=	A>=B	True
==	A== B	False
!=	A!=(-4)	True

Logical Operator

Which can be used to combine more than one Condition?. Suppose you want to combined two conditions A<B and B>C, then you need to use Logical Operator like (A<B) && (B>C). Here &&is Logical Operator.

Operator	Example (int A=8, B=3, C=-10)	Result

&&	(A<B) && (B>C)	False
||	(B!=-C) || (A==B)	True
!	!(B<=-A)	True

Truth table of Logical Operator

C1	C2	C1 && C2	C1 || C2	!C1	!C2
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

Assignment operators

Which can be used to assign a value to a variable. Lets suppose variable A hold 8 and B hold 3.

Operator	Example (int A=8, B=3)	Result
+=	A+=B or A=A+B	11
-=	A-=3 or A=A-3	5
*=	A*=7 or A=A*7	56
/=	A/=B or A=A/B	2
%=	A%=5 or A=A%5	3
=a=b	Value of b will be assigned to a	

Increment and Decrement Operator in C

Increment Operators are used to increased the value of the variable by one and Decrement Operators are used to decrease the value of the variable by one in C programs.

Both increment and decrement operator are used on a single operand or variable, so it is called as a unary operator. Unary operators are having higher priority than the other operators it means unary operators are executed before other operators.

Syntax

```
++  // increment operator
--  // decrement operator
```

Note: Increment and decrement operators are can not apply on constant.

Example

```
x= 4++;  // gives error, because 4 is constant
```

Type of Increment Operator

- pre-increment
- post-increment

pre-increment (++ variable)

In pre-increment first increment the value of variable and then used inside the expression (initialize into another variable).

Syntax

```
++ variable;
```

Example pre-increment

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

void main()
{
int x,i;
i=10;
x=++i;
printf("x: %d",x);
printf("i: %d",i);
getch();
}
```

```
x: 11
i: 11
```

post-increment (variable ++)

In post-increment first value of variable is used in the expression (initialize into another variable) and then increment the value of variable.

Syntax

```
variable ++;
```

Example post-increment

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

void main()
{
int x,i;
i=10;
x=i++;
printf("x: %d",x);
printf("i: %d",i);
getch();
}
```

```
x: 10
i: 11
```

Type of Decrement Operator

- pre-decrement
- post-decrement

Pre-decrement (-- variable)

In pre-decrement first decrement the value of variable and then used inside the expression (initialize into another variable).

Syntax

```
-- variable;
```

Example pre-decrement

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

void main()
{
int x,i;
i=10;
x=--i;
printf("x: %d",x);
printf("i: %d",i);
getch();
}
```

Output

```
x: 9
i: 9
```

post-decrement (variable --)

In Post-decrement first value of variable is used in the expression (initialize into another variable) and then decrement the value of variable.

Syntax

```
variable --;
```

Example post-decrement

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

void main()
{
int x,i;
i=10;
x=i--;
printf("x: %d",x);
printf("i: %d",i);
getch();
}
```

Output

```
x: 10
i: 9
```

Example of increment and decrement operator

Example

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

void main()
{
int x,a,b,c;
a = 2;
b = 4;
c = 5;
x = a-- + b++ - ++c;
printf("x: %d",x);
getch();
}
```

WHAT IS THE OUTPUT??

```
x: 0
```

Ternary Operator in C

If any operator is used on three operands or variable is known as Ternary Operator. It can be represented with ? : . It is also called as conditional operator

Advantage of Ternary Operator

Using ?: reduce the number of line codes and improve the performance of application.

Syntax

```
expression-1 ? expression-2 : expression-3
```

In the above symbol expression-1 is condition and expression-2 and expression-3 will be either value or variable or statement or any mathematical expression. If condition will be true expression-2 will be execute otherwise expression-3 will be executed.

Syntax

```
a<b ? printf("a is less") : printf("a is greater");
```

Find largest number among 3 numbers using ternary operator

Example

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

void main()
{
int a, b, c, large;
clrscr();
printf("Enter any three number: ");
scanf("%d%d%d",&a,&b,&c);
large=a>b ? (a>c?a:c) : (b>c?b:c);
printf("Largest Number is: %d",large);
getch();
}
```

BIT WISE OPERATORS IN C:

These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise NOT), ^ (XOR), << (left shift) and >> (right shift).

TRUTH TABLE FOR BIT WISE OPERATION & BIT WISE OPERATORS:

x	y	x|y	x&y	x^y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

BELOW ARE THE BIT-WISE OPERATORS AND THEIR NAME IN C LANGUAGE.

& – Bitwise AND

| – Bitwise OR

~ – Bitwise NOT

^ – XOR

<< – Left Shift

>> – Right Shift

Consider x=40 and y=80. Binary form of these values are given below.

x = 00101000
y= 01010000

```
#include <stdio.h>

void  main()
{
   int m = 40,n = 80,AND_opr,OR_opr,XOR_opr,NOT_opr ;
   AND_opr = (m&n);
   OR_opr = (m|n);
   NOT_opr = (~m);
   XOR_opr = (m^n);
   printf("AND_opr value = %d\n",AND_opr );
   printf("OR_opr value = %d\n",OR_opr );
   printf("NOT_opr value = %d\n",NOT_opr );
   printf("XOR_opr value = %d\n",XOR_opr );
   printf("left_shift value = %d\n", m << 1);
   printf("right_shift value = %d\n", m >> 1);
}
```

Output:

```
AND_opr value = 0
OR_opr value = 120
NOT_opr value = -41
XOR_opr value = 120
left_shift value = 80
right_shift value = 20
```

Different unary operators:

- Unary plus(+)
- Unary minus (-)
- Increment (++)
- Decrement (--)
- Logical NOT (!)
- Bitwise complement (~)
- Address (&)

Different binary operators

- Arithmetic operator.
- Relational operator.
- Logical operator.
- Assignment operator.
- Bitwise operator

What is sizeof operator. explain with example. 2m

The sizeof operator is used to calculate the size of data type or variables. This operator returns the size of its variable in bytes.

For example: sizeof(a), where a is interger, will return 4

Syntax

```
sizeof(variable)
```

Example

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

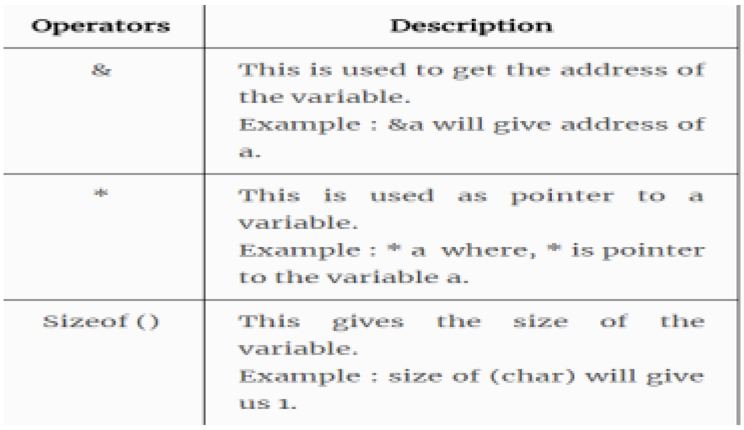
```
void main()
{
int a;
float b;
double c;
char d;
printf("Size of Integer: %d bytes\n",sizeof(a));
printf("Size of float: %d bytes\n",sizeof(b));
printf("Size of double: %d bytes\n",sizeof(c));
printf("Size of character: %d byte\n",sizeof(d));
getch();
}
```

Output

```
Size of Integer: 2

Size of float: 4

Size of double: 8

Size of character: 1
```

SPECIAL OPERATORS IN C:

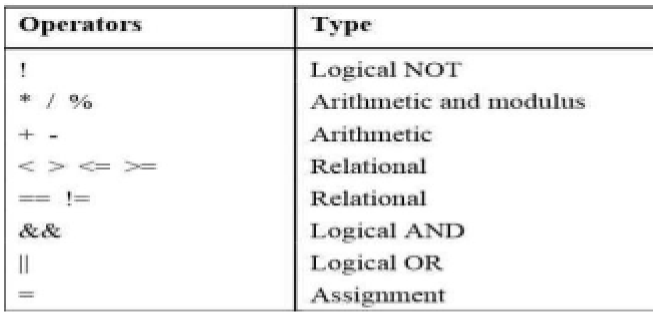
Below are some of the special operators that the C programming language offers.



Operators	Description
&	This is used to get the address of the variable. Example : &a will give address of a.
*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.

Hierarchy of operators

Operator precedence determines which **operator** is performed first in an expression with more than one**operators** with different **precedence**. ...



Operators	Type
!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
||	Logical OR
=	Assignment

For example '*' and '/' have same **precedence** and their associativity is Left to Right, so the expression “100 / 10 * 10” is treated as “(100 / 10) * 10”.

C if else Statement

The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given condition is true.

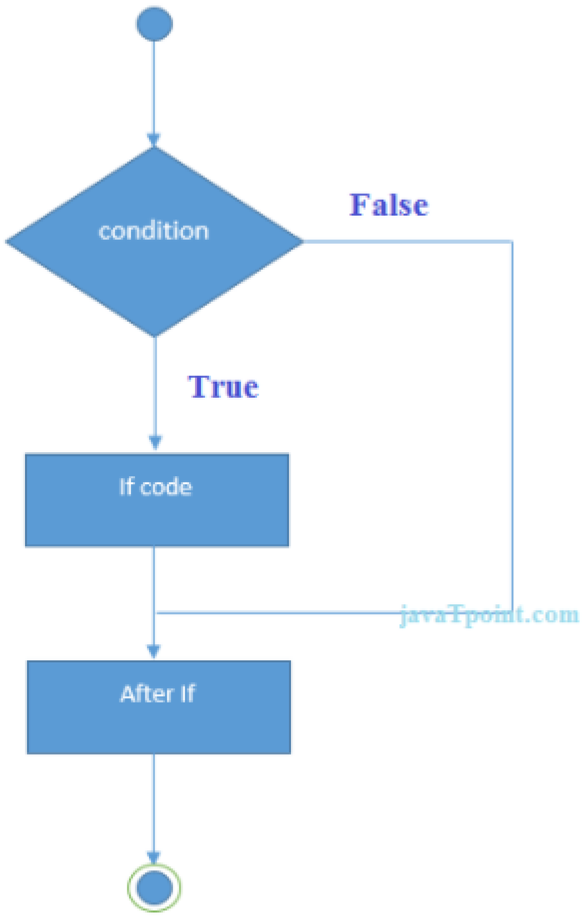
There are the following variants of if statement in C language.

- If statement
- If-else statement
- If else-if ladder
- Nested if

If Statement

The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition. It is mostly used in the scenario where we need to perform the different operations for the different conditions. The syntax of the if statement is given below.

```
if(expression){
//code to be executed
}
```


condition
False
True
If code
javaTpoint.com
After If

```
#include<stdio.h>
int main(){
int number=0;
printf("Enter a number:");
scanf("%d",&number);
if(number%2==0){
printf("%d is even number",number);
}
return 0;
}
```

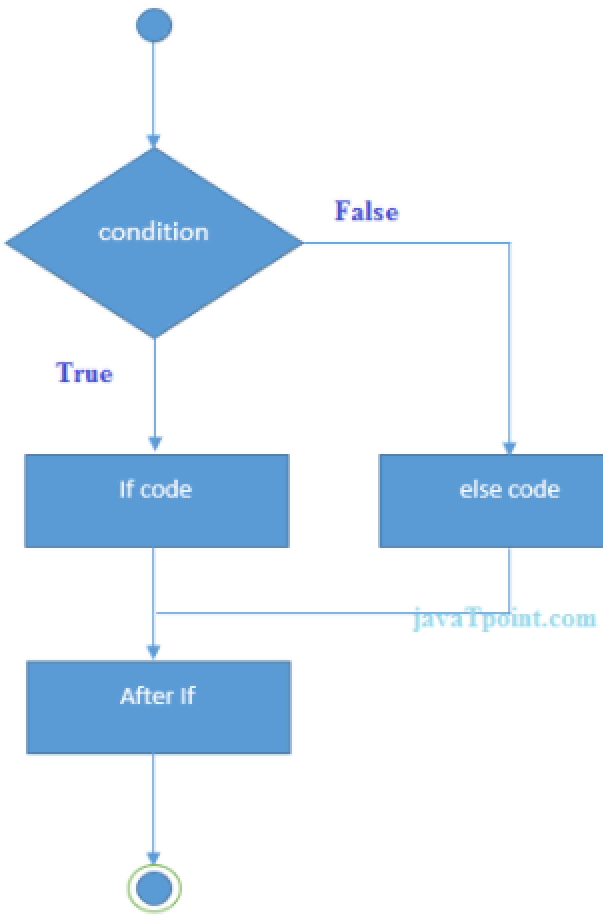
Output

```
Enter a number:4
4 is even number
enter a number:5
```

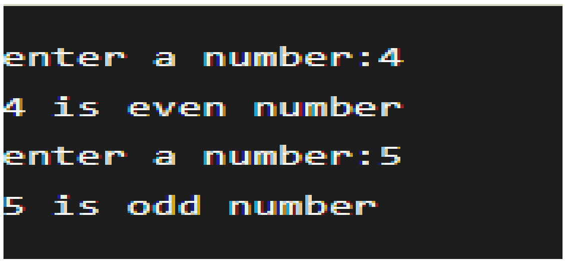
If-else Statement

- The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition.
- Here, we must notice that if and else block cannot be executed simiulteneously. Using if-else statement is always preferable since it always invokes an otherwise case with every if condition. The syntax of the if-else statement is given below.

```
if(expression){
//code to be executed if condition is true
}else{
//code to be executed if condition is false
}
```



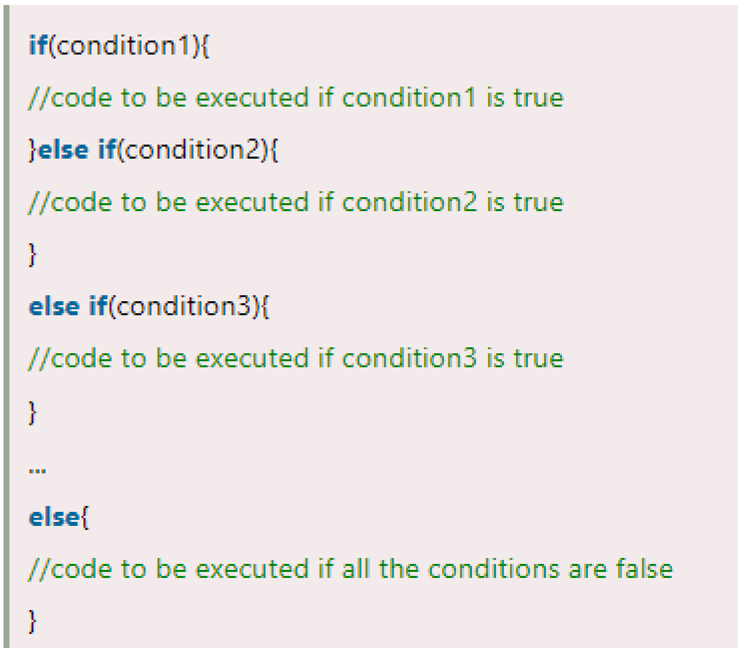
```
#include<stdio.h>
int main(){
int number=0;
printf("enter a number:");
scanf("%d",&number);
if(number%2==0){
printf("%d is even number",number);
}
else{
printf("%d is odd number",number);
}
return 0;
}
```



```
enter a number:4
4 is even number
enter a number:5
5 is odd number
```

If else-if ladder Statement

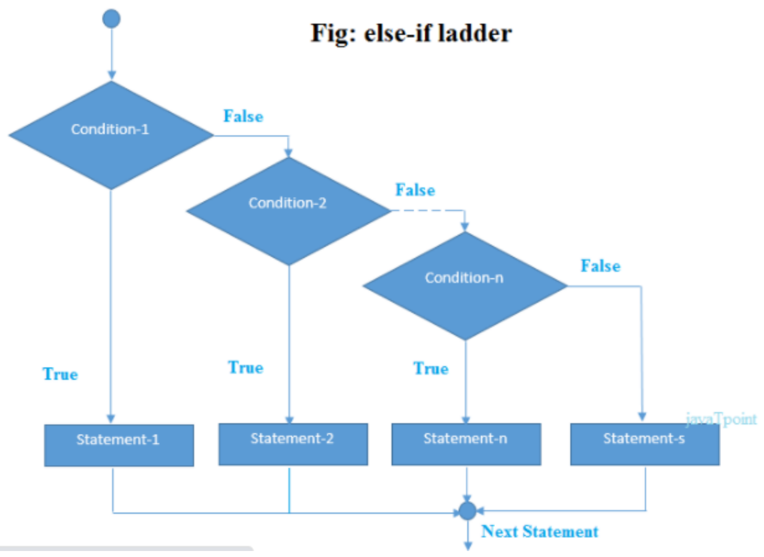
- The if-else-if ladder statement is an extension to the if-else statement. It is used in the scenario where there are multiple cases to be performed for different conditions.
- In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed.
- There are multiple else-if blocks possible. It is similar to the switch case statement where the default is executed instead of else block if none of the cases is matched.



```
if(condition1){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```

-

Fig: else-if ladder



```
#include<stdio.h>
int main(){
int number=0;
printf("enter a number:");
scanf("%d",&number);
if(number==10){
printf("number is equals to 10");
}
else if(number==50){
printf("number is equal to 50");
}
else if(number==100){
printf("number is equal to 100");
}
else{
printf("number is not equal to 10, 50 or 100");
}
return 0;
}
```

C Switch Statement

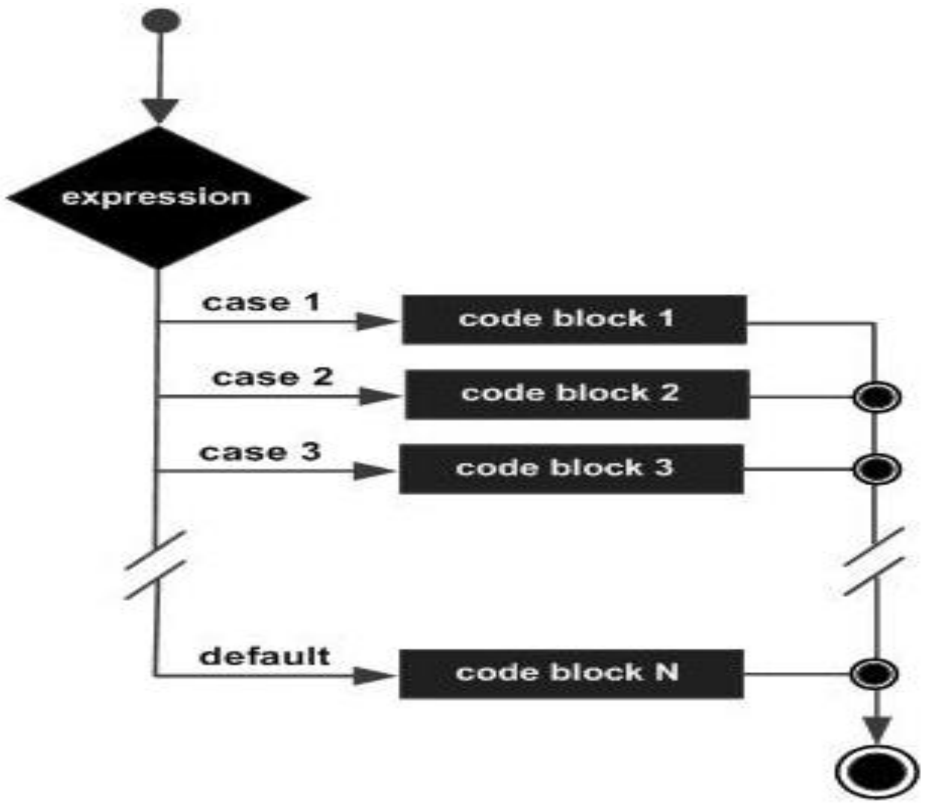
The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, We can define various statements in the multiple cases for the different values of a single variable.

The syntax of switch statement in c language is given below:

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
 code to be executed if all cases are not matched;
}
```

The syntax of switch statement in c language is given below:

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
 code to be executed if all cases are not matched;
}
```



```c
#include <stdio.h>

int main () {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade );

   return 0;
}
```

C Loops

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language.

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times.

For example, if we need to print the first 10 natural numbers then, instead of using the printf statement 10 times, we can print inside a loop which runs up to 10 iterations.

Types of C Loops

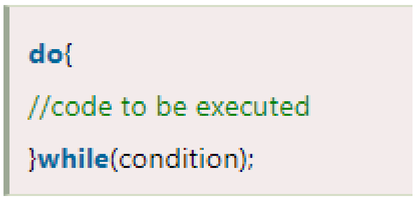
There are three types of loops in C language that is given below:

1. do while
2. while
3. for

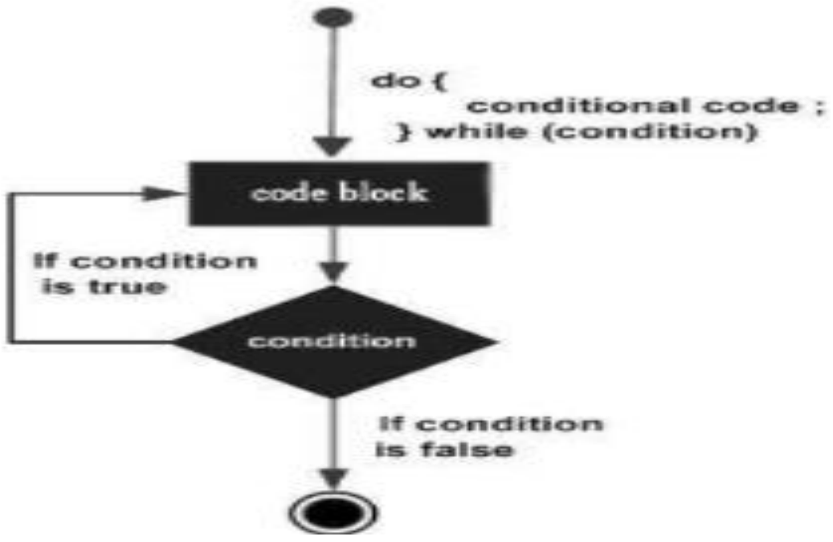
do-while loop in C

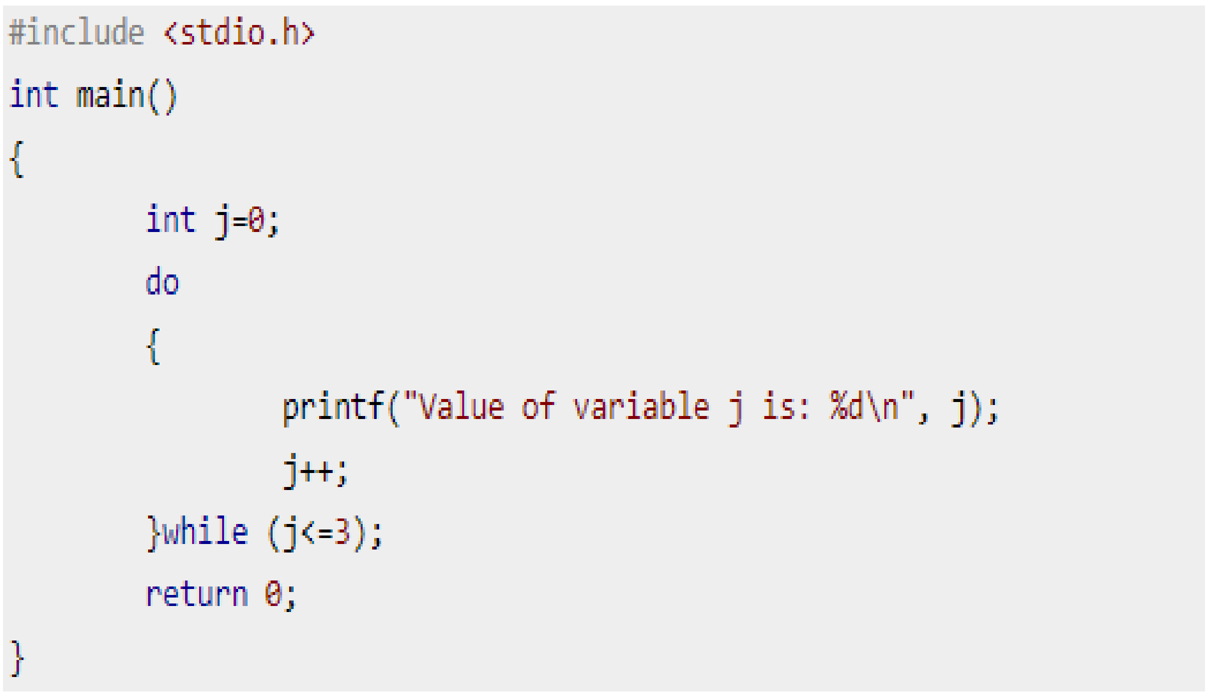
The do-while loop continues until a given condition satisfies. It is also called post tested loop. It is used when it is necessary to execute the loop at least once (mostly menu driven programs).

The syntax of do-while loop in c language is given below:



```
do{
//code to be executed
}while(condition);
```





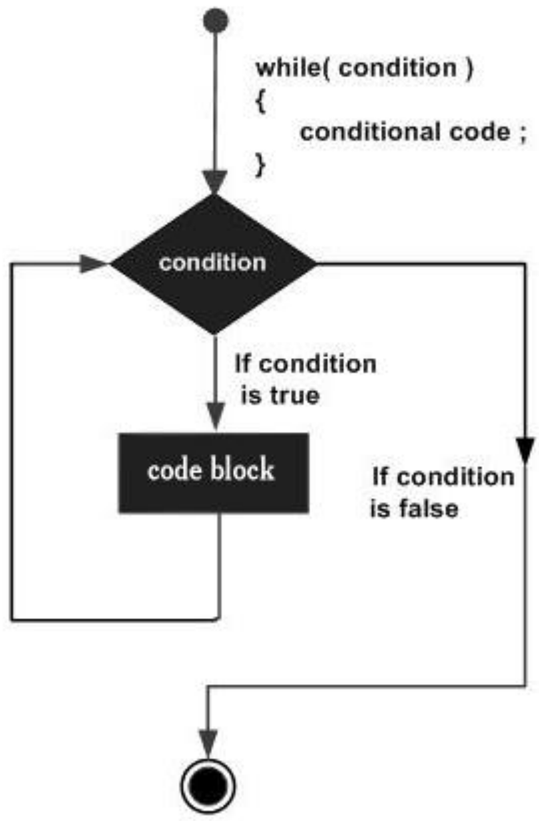
```
#include <stdio.h>
int main()
{
        int j=0;
        do
        {
                printf("Value of variable j is: %d\n", j);
                j++;
        }while (j<=3);
        return 0;
}
```

while loop in C

The while loop in c is to be used in the scenario where we don't know the number of iterations in advance. The block of statements is executed in the while loop until the condition specified in the while loop is satisfied. It is also called a pre-tested loop.

The syntax of while loop in c language is given below:

```
while(condition){
//code to be executed
}
```



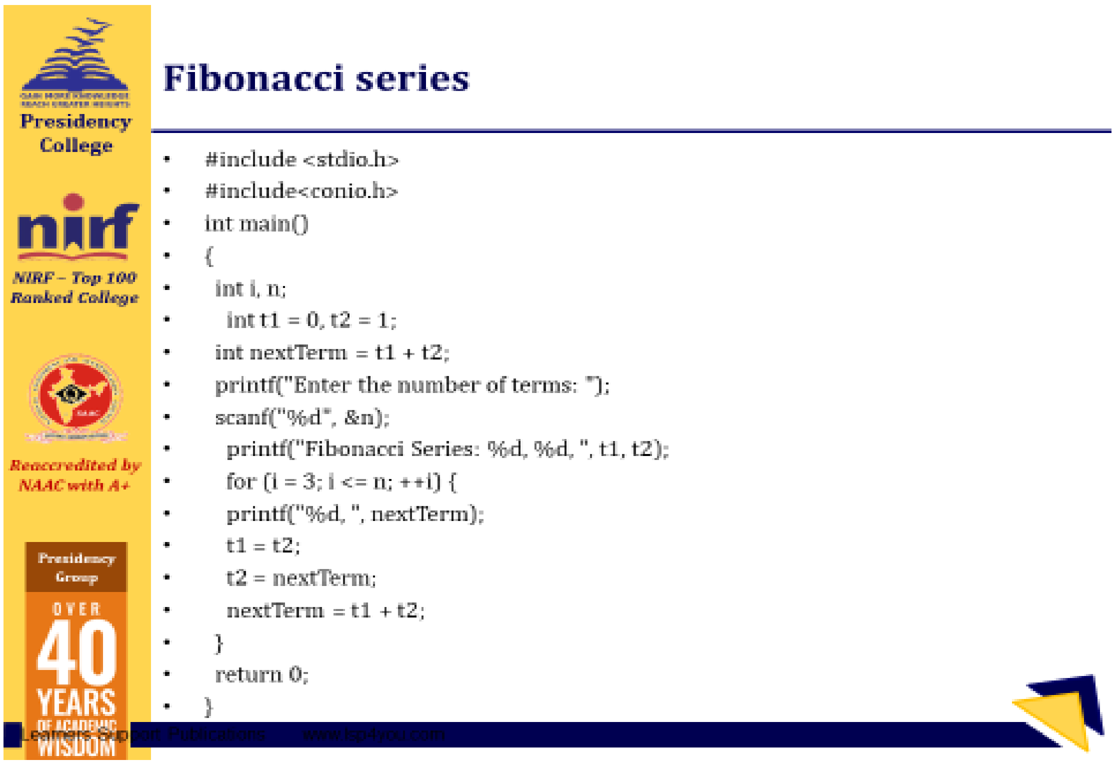
```
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

for loop in C

The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance.

The syntax of for loop in c language is given below:

```
for(initialization;condition;incr/decr){
//code to be executed
}
```



Fibonacci series

```
•  #include <stdio.h>
•  #include<conio.h>
•  int main()
•  {
•   int i, n;
•     int t1 = 0, t2 = 1;
•   int nextTerm = t1 + t2;
•   printf("Enter the number of terms: ");
•   scanf("%d", &n);
•     printf("Fibonacci Series: %d, %d, ", t1, t2);
•     for (i = 3; i <= n; ++i) {
•     printf("%d, ", nextTerm);
•     t1 = t2;
•     t2 = nextTerm;
•     nextTerm = t1 + t2;
•   }
•   return 0;
•  }
```

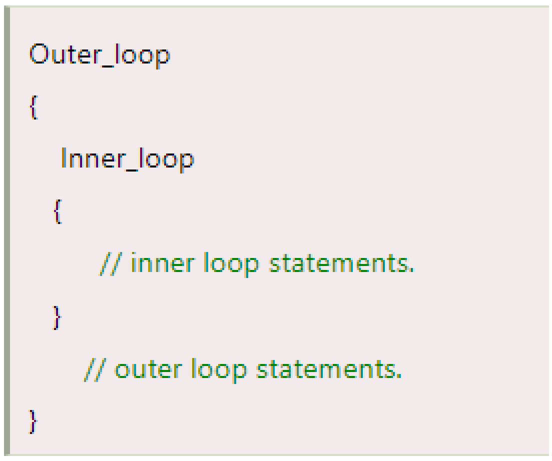
Presidency College
NIRF – Top 100 Ranked College
Reaccredited by NAAC with A+
Presidency Group
OVER 40 YEARS OF ACADEMIC WISDOM
Learners Support Publications   www.lsp4you.com

Nested Loops in C

C supports nesting of loops in C. **Nesting of loops** is the feature in C that allows the looping of statements inside another loop. Let's observe an example of nesting loops in C.

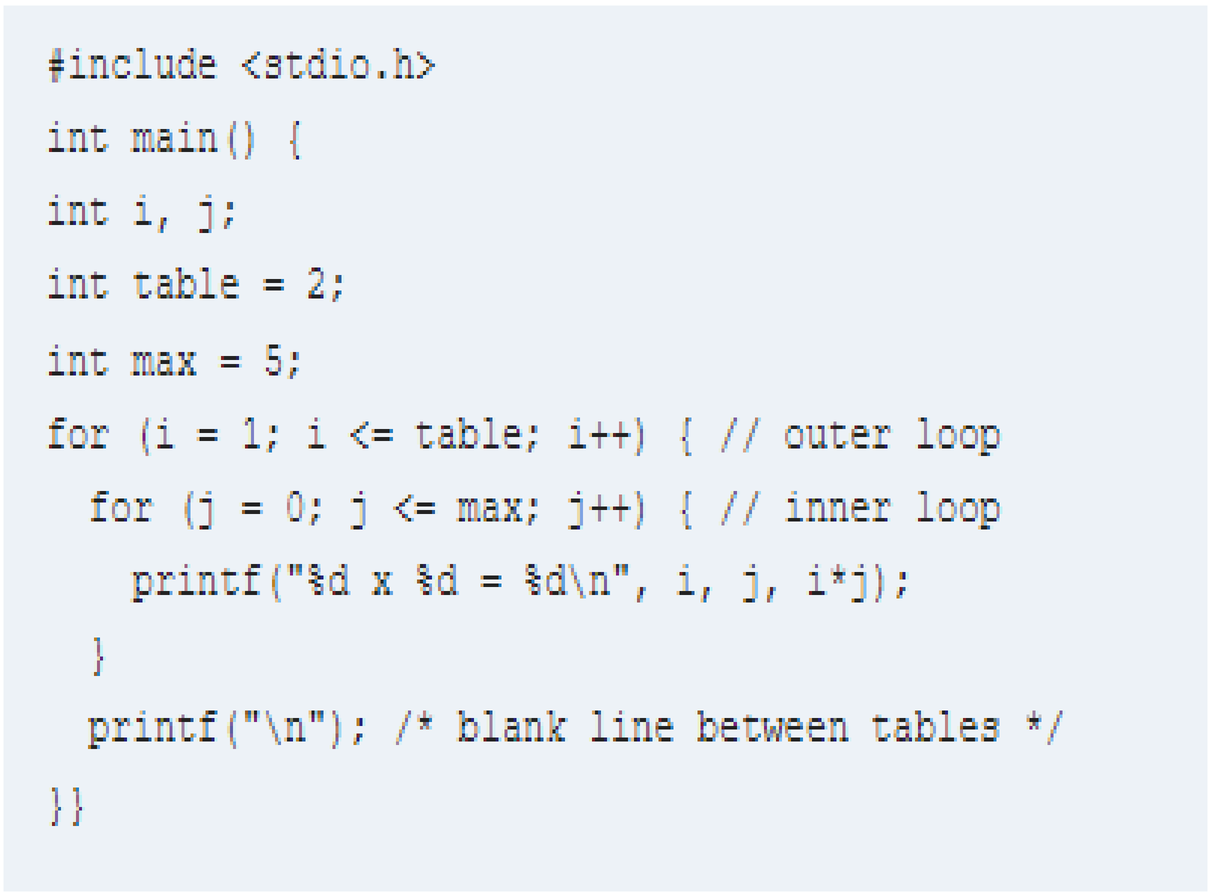
Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops. The nesting level can be defined at n times. You can define any type of loop inside another loop; for example, you can define '**while**' loop inside a '**for**' loop.

Syntax of Nested loop



```
Outer_loop
{
   Inner_loop
  {
       // inner loop statements.
  }
     // outer loop statements.
}
```

Outer_loop and **Inner_loop** are the valid loops that can be a 'for' loop, 'while' loop or 'do-while' loop.



```
#include <stdio.h>
int main() {
int i, j;
int table = 2;
int max = 5;
for (i = 1; i <= table; i++) { // outer loop
  for (j = 0; j <= max; j++) { // inner loop
    printf("%d x %d = %d\n", i, j, i*j);
  }
  printf("\n"); /* blank line between tables */
}}
```

```
1 x 0 = 0
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5

2 x 0 = 0
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
```

C break statement

The break is a keyword in C which is used to bring the program control out of the loop. The break statement is used inside loops or switch statement.

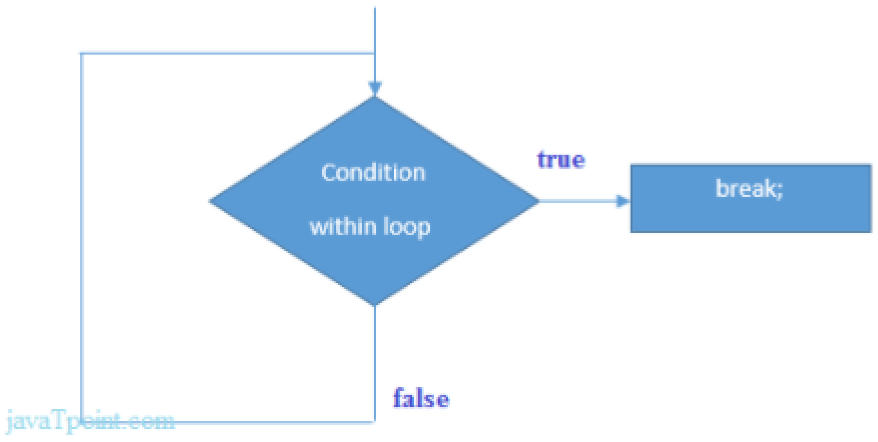
The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in C can be used in the following two scenarios:

1. With switch case
2. With loop

Syntax:

```
//loop or switch case
break;
```

Flowchart of break in c



```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    int i;
    for(i = 0; i<10; i++)
    {
        printf("%d ",i);
        if(i == 5)
        break;
    }
    printf("came outside of loop i = %d",i);

}
```

Output

```
0 1 2 3 4 5 came outside of loop i = 5
```

C continue statement

The **continue statement** in C language is used to bring the program control to the beginning of the loop. The continue statement skips some lines of code inside the loop and continues with the next iteration. It is mainly used for a condition so that we can skip some code for a particular condition.

```
//loop statements
continue;
//some lines of the code which is to be skipped
```

```
#include<stdio.h>
void main ()
{
    int i = 0;
    while(i!=10)
    {
        printf("%d", i);
        continue;
        i++;
    }
}
```

```
infinite loop
```

C goto statement

- The goto statement is known as jump statement in C. As the name suggests, goto is used to transfer the program control to a predefined label. The goto statment can be used to repeat some part of the code for a particular condition.
- It can also be used to break the multiple loops which can't be done by using a single break statement. However, using goto is avoided these days since it makes the program less readable and complecated.

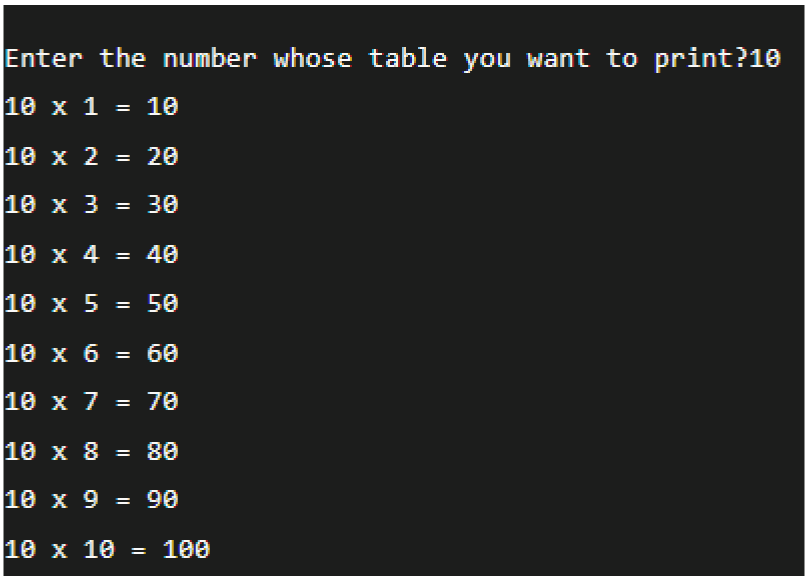
Syntax:

```
label:
//some part of the code;
goto label;
```

goto example

Let's see a simple example to use goto statement in C language.

```
#include <stdio.h>
int main()
{
  int num,i=1;
  printf("Enter the number whose table you want to print?");
  scanf("%d",&num);
  table:
  printf("%d x %d = %d\n",num,i,num*i);
  i++;
  if(i<=10)
  goto table;
}
```



```
Enter the number whose table you want to print?10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

C Functions

In c, we can divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the C program

. In other words, we can say that the collection of functions creates a program. The function is also known as *procedure*or *subroutine*in other programming languages.

Advantage of functions in C

There are the following advantages of C functions.

- By using functions, we can avoid rewriting same logic/code again and again in a program.
- We can call C functions any number of times in a program and from any place in a program.
- We can track a large C program easily when it is divided into multiple functions.
- Reusability is the main achievement of C functions.
- However, Function calling is always a overhead in a C program.

Function Aspects

There are three aspects of a C function.

- **Function declaration** A function must be declared globally in a c program to tell the compiler about the function name, function parameters, and return type.

- **Function call** Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of functions as it is declared in the function declaration.

- **Function definition** It contains the actual statements which are to be executed. It is the most important aspect to which the control comes when the function is called. Here, we must notice that only one value can be returned from the function.

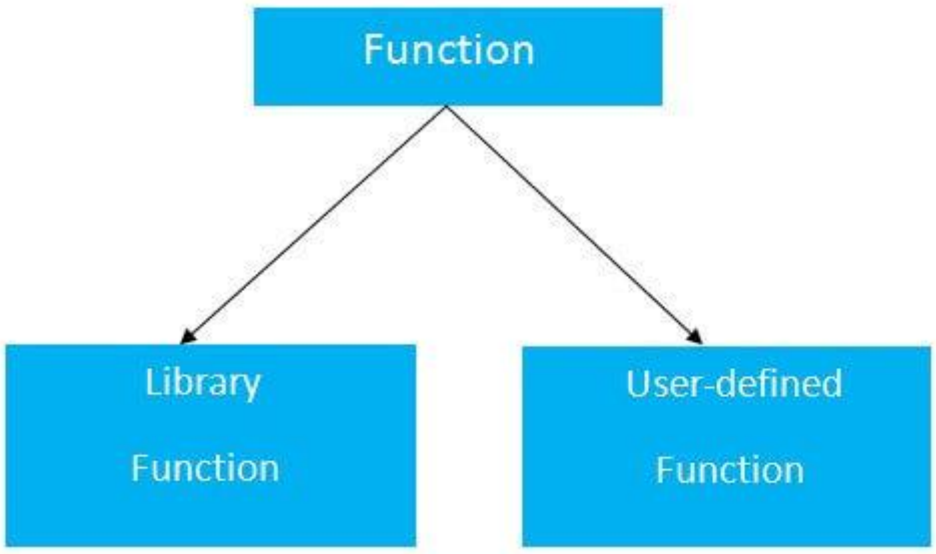
SN	C function aspects	Syntax
1	Function declaration	return_type function_name (argument list);
2	Function call	function_name (argument_list)
3	Function definition	return_type function_name (argument list) {function body;}

```
return_type function_name(data_type parameter...){
//code to be executed
```

Types of Functions

There are two types of functions in C programming:

1. **Library Functions**: are the functions which are declared in the C header files such as scanf(), printf(), gets(), puts(), ceil(), floor() etc.
2. **User-defined functions**: are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.



Return Value

A C function may or may not return a value from the function. If you don't have to return any value from the function, use void for the return type.

Let's see a simple example of C function that doesn't return any value from the function.

Example without return value:

```
void hello(){
printf("hello c");
}
```

Example with return value:

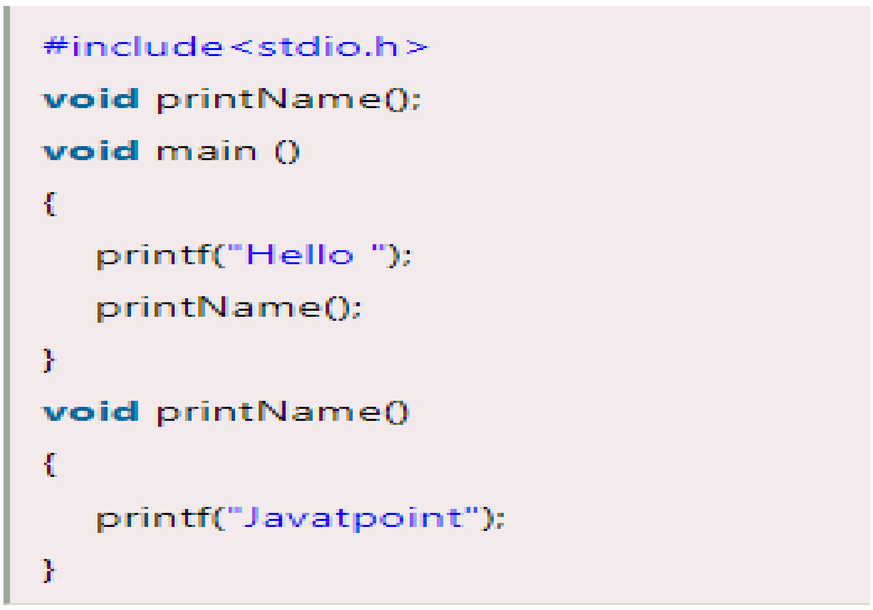
```
int get(){
return 10;
}
```

Different aspects of function calling

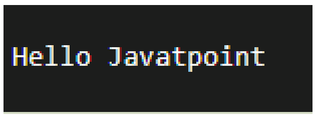
A function may or may not accept any argument. It may or may not return any value. Based on these facts, There are four different aspects of function calls.

- function without arguments and without return value
- function without arguments and with return value
- function with arguments and without return value
- function with arguments and with return value

Function without argument and return value

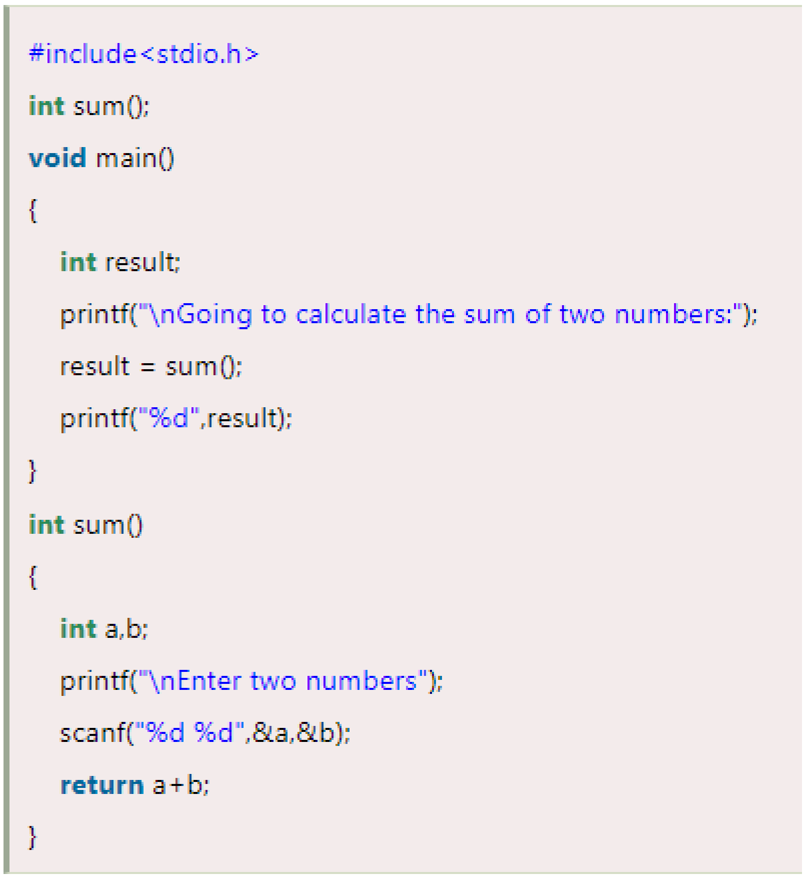


```c
#include<stdio.h>
void printName();
void main ()
{
    printf("Hello ");
    printName();
}
void printName()
{
    printf("Javatpoint");
}
```

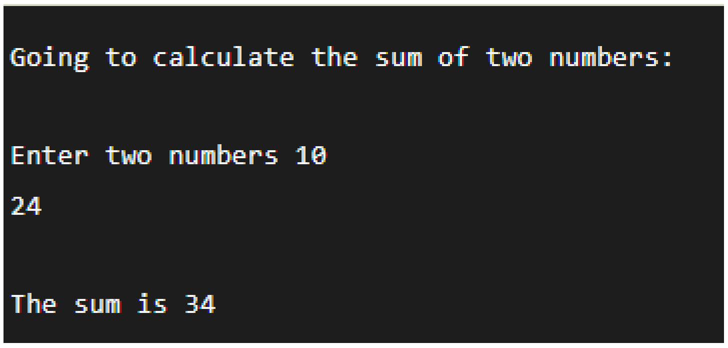


```
Hello Javatpoint
```

Function without argument and with return value



```
#include<stdio.h>
int sum();
void main()
{
    int result;
    printf("\nGoing to calculate the sum of two numbers:");
    result = sum();
    printf("%d",result);
}
int sum()
{
    int a,b;
    printf("\nEnter two numbers");
    scanf("%d %d",&a,&b);
    return a+b;
}
```



```
Going to calculate the sum of two numbers:

Enter two numbers 10
24

The sum is 34
```

Function with argument and without return value

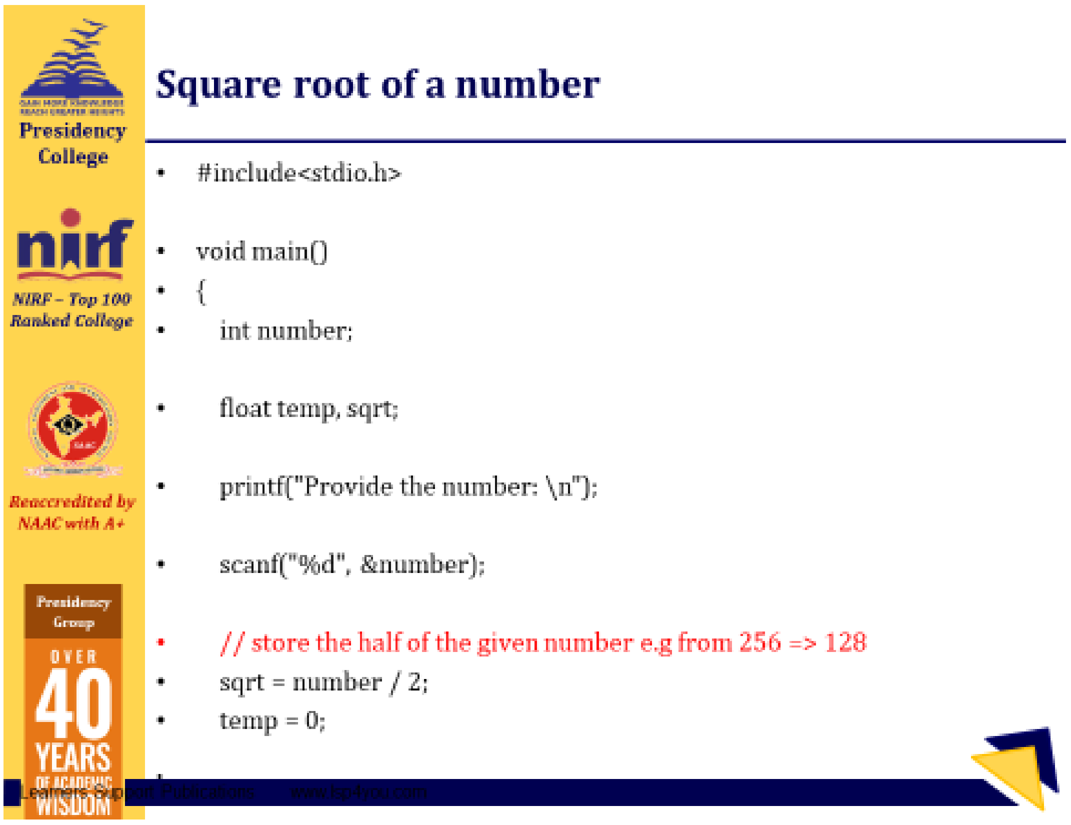
```c
#include<stdio.h>
void sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two numbers:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    sum(a,b);
}
void sum(int a, int b)
{
    printf("\nThe sum is %d",a+b);
}
```

```
Going to calculate the sum of two numbers:

Enter two numbers 10
24

The sum is 34
```

Function with argument and with return value

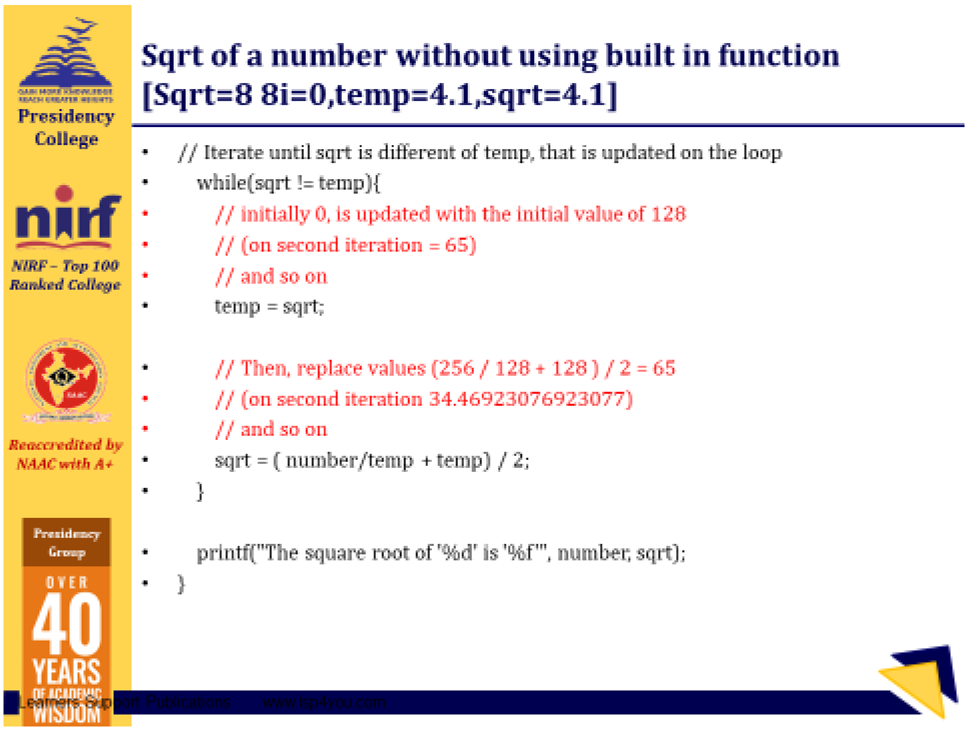
```c
#include<stdio.h>
int sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two numbers:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    result = sum(a,b);
    printf("\nThe sum is : %d",result);
}
int sum(int a, int b)
{
    return a+b;
}
```

```
Going to calculate the sum of two numbers:
Enter two numbers:10
20
The sum is : 30
```



Square root of a number

```
#include<stdio.h>

void main()
{
   int number;

   float temp, sqrt;

   printf("Provide the number: \n");

   scanf("%d", &number);

   // store the half of the given number e.g from 256 => 128
   sqrt = number / 2;
   temp = 0;
```



Sqrt of a number without using built in function [Sqrt=8 8i=0,temp=4.1,sqrt=4.1]

```
  // Iterate until sqrt is different of temp, that is updated on the loop
     while(sqrt != temp){
        // initially 0, is updated with the initial value of 128
        // (on second iteration = 65)
        // and so on
        temp = sqrt;

        // Then, replace values (256 / 128 + 128 ) / 2 = 65
        // (on second iteration 34.46923076923077)
        // and so on
        sqrt = ( number/temp + temp) / 2;
     }

     printf("The square root of '%d' is '%f'", number, sqrt);
  }
```

12=4X3 [Prime factorization]

- #include<stdio.h>
- Int main()
- {

int n;

Printf (“enter the number”);

Scanf(“%d”,&n);

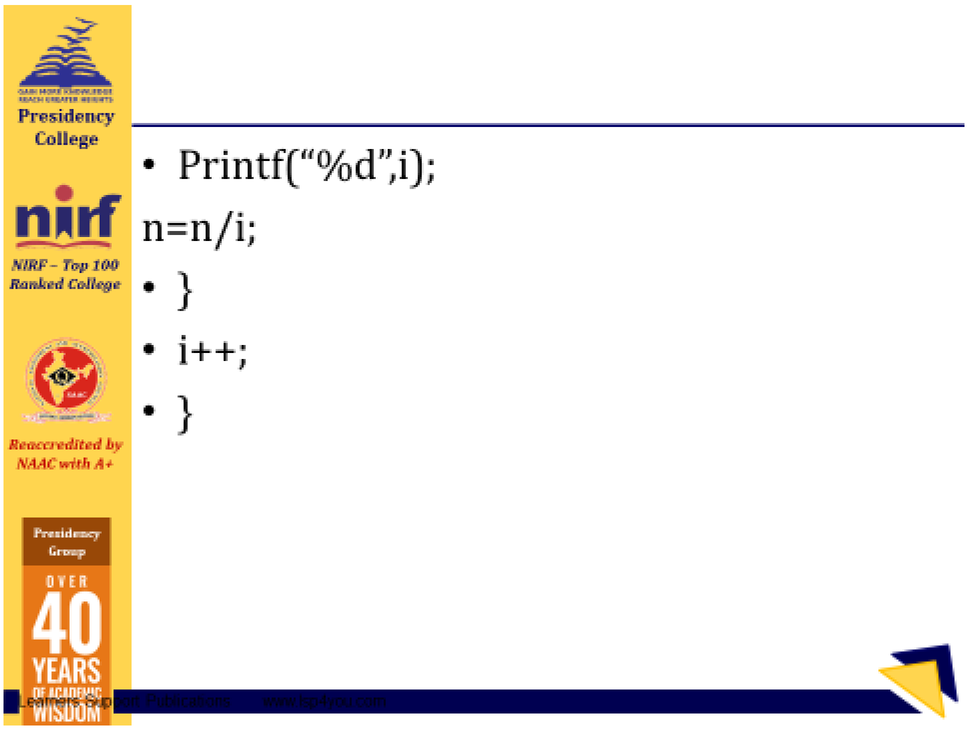
primefactor(n);

Return 0;

}

12=2X2X3

- Void primefactor(int n)
- {
- Int i=2;
- While(n!=1)
- {
- If(n%i==0)
- {
- While(n%i==0)
- {


Presidency College
NIRF – Top 100 Ranked College
Reaccredited by NAAC with A+
Presidency Group
OVER 40 YEARS OF ACADEMIC WISDOM

- Printf("%d",i);

n=n/i;

- }
- i++;
- }

Learners Support Publications www.lsp4you.com

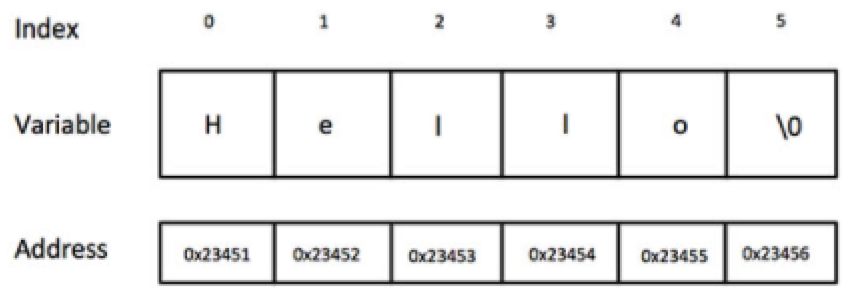
C Programming Strings

- Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.
- Thus a null-terminated string contains the characters that comprise the string followed by a **null**.
- The following declaration and initialization create a string consisting of the word "Hello".
- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."
-
```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```
-
```
char greeting[] = "Hello";
```

If you follow the rule of array initialization then you can write the above statement as follows −

```
char greeting[] = "Hello";
```

Following is the memory presentation of the above defined string in C/C++ −



Sr.No.	Function & Purpose
	strcpy(s1, s2); Copies string s2 into string s1.
	strcat(s1, s2); Concatenates string s2 onto the end of string s1.
	strlen(s1); Returns the length of string s1.
	strcmp(s1, s2); Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
	strchr(s1, ch); Returns a pointer to the first occurrence of character ch in string s1.
	strstr(s1, s2); Returns a pointer to the first occurrence of string s2 in string s1.

```
#include <stdio.h>
int main() {
char name[10];
int age;
printf("Enter your first name and age: \n");
scanf("%s %d", name, &age);
printf("You entered: %s %d",name,age);
}
```

Output:

```
Enter your first name and age:
John_Smith 48
```