C is a structured programming language developed by Dennis Ritchie in 1973 at Bell Laboratories.

C language was developed to write the UNIX operating system, hence it is strongly associated with UNIX, which is one of the most popular network operating system in use today and heart of internet data superhighway.( OS for switches and routers )

Features of C language

1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed
8. Pointers

**Simple**

C is a simple language in the sense that it provides **structured approach** (to break the problem into parts), **rich set of library functions**, **data types** etc.

**Machine Independent or Portable**

Unlike assembly language, c programs **can be executed in many machines** with little bit or no change. But it is not platform-independent.

**Mid-level programming language**

C is **also used to do low level programming**. It is used to develop system applications such as kernel, driver etc.

**Structured programming language**

C is a structured programming language in the sense that **we can break the program into parts using functions**. So, it is easy to understand and modify.

**Rich Library**

C **provides a lot of inbuilt functions** that makes the development fast.

**Memory Management**

It supports the feature of **dynamic memory allocation**. In C language, we can free the allocated memory at any time by calling the **free()**function.

**Speed**

The compilation and execution time of C language is fast.

**Pointer**

C provides the feature of pointers. We can directly interact with the memory by using the pointers. We **can use pointers for memory, structures, functions, array** etc.

**First C program**

#include <stdio.h>

int main()

{

printf("Hello,World");

return 0;

}

Save above code in a file Hello.c

**Explanation of above program**

**#include <stdio.h>** includes the **standard input output** library functions.

The printf() function is defined in stdio.h .

**int main()** The **main() function is the entry point of every program** in c language.

//main() can be void , float etc

//add #include <stdbool.h> for boolean return type

**printf()** The printf() function is **used to print data** on the console.

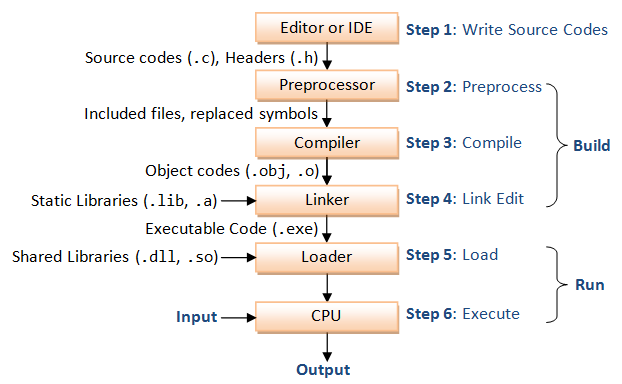
**return 0** The return 0 statement, returns execution status to the OS.

**Comments**

Use // double forward slashes for single line comment

Use /\* \*/ for multi line comments

**Execution Flow of C program**

****

**Syntax rules for C program**

* C is a case sensitive language so all C instructions must be written in lower case letter.
* All C statement must end with a semicolon.
* Whitespace is used in C to describe blanks and tabs.
* Whitespace is required between keywords and identifiers.

**Keywords**

Keywords are preserved words that have special meaning in C language. The meaning of C language keywords has already been described to the C compiler. These meaning cannot be changed. Thus, keywords cannot be used as variable names because that would try to change the existing meaning of the keyword, which is not allowed.

Ex : int ,float , break, case , for ,switch etc.

**Identifiers**

In C language identifiers are the names given to variables, constants, functions and user-defined data.

Rules for identifiers

1. An Identifier can only have alphanumeric characters(a-z , A-Z , 0-9) and underscore(\_).
2. The first character of an identifier can only contain alphabet(a-z , A-Z) or underscore (\_).
3. Identifiers are also case sensitive in C. For example **name** and **Name** are two different identifiers in C.
4. Keywords are not allowed to be used as Identifiers.
5. No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.

Operators in C

* Arithmetic operators
* Relational operators
* Logical operators
* Bitwise operators
* Assignment operators
* ternary operator
* Special operators

**Arithmetic Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Adds two operands |
| - | Subtract second operands from first |
| \* | Multiply two operand |
| / | Divide numerator by denominator |
| % | Remainder of division |
| ++ | Increment operator - increases integer value by one |
| -- | Decrement operator - decreases integer value by one |

**Relational Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | Check if two operand are equal |
| != | Check if two operand are not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |

### 

### 

### 

### 

### 

### 

### **Logical operators**

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

**Bitwise Operators**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| & | Bitwise AND |
| | | Bitwise OR |
| **^** | Bitwise exclusive OR |
| **<<** | Left shift |
| **>>** | Right shift |

**Samples :**

**Bitwise AND**

#include<stdio.h>

int main()

{

//decimal to binary conversion - keep dividing by 2, follow remainders

//bitwise and

//25 --- 1 1001

//15 --- 0 1111

//------------------

//09 0 1001

-> 0 + 2^3 + 0 + 0 + 2^0 ->

-> 0+8+0+0+1 -> 9

int a = 25,b=15;

printf("output = %d ",a&b);

return 0;

}

**Bitwise OR**

//decimal to binary conversion - keep dividing by 2, follow remainders

//25 --- 1 1001

//15 --- 0 1111

//-------------------

//31 1 1111

//31

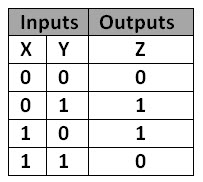
-> 2^4 + 2^3 + 2^2 + 2^1 + 2^0

-> 16 + 8 + 4 + 2 + 1

int a = 25,b=15;

printf("output = %d ",a|b);

**Bitwise XOR**

****

//decimal to binary conversion - keep dividing by 2, follow remainders

//25 --- 1 1001

//15 --- 0 1111

//-----------------

//22 1 0110 -> 2^4 + 0 + 2^2 + 2^1 + 0 -> 22

int a = 25,b=15;

printf("output = %d ",a^b);

**Left Shift Operator**

int a = 60;

printf("\nNumber is Shifted By 1 Bit : %d",a << 1);//120 - output

printf("\nNumber is Shifted By 2 Bits : %d",a << 2);//240 - output

**Right Shift Operator**

int a = 60;

printf("\nNumber is Shifted By 1 Bit : %d",a >> 1);//30

printf("\nNumber is Shifted By 2 Bits : %d",a >> 2);//15

**Assignment Operator**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Assigns values from right side operands to left side operand |
| += | Adds right operand to the left operand and assign the result to left |
| -= | subtracts right operand from the left operand and assign the result to left operand |
| \*= | multiply left operand with the right operand and assign the result to left operand.  Int i = 10;  i\*=10;  //output : i -> 100 |
| /= | divides left operand with the right operand and assign the result to left operand |
| %= | calculate modulus using two operands and assign the result to left operand |

**Ternary Operator**

**Expression 1 ? expression 2 : expression 3**

**Ex :** minVal = (a < b) ? a : b;

**Sample :**

int a = 5 , b = 10;

int minVal = (a < b) ? a : b;

printf("%d", minVal);

**Special operator**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| sizeof | Returns the size of a variable (depending on type, ex : int 2/4 bytes) |
| & | Returns the address of a variable |
| \* | Pointer to a variable |

**Ex :**

**Sizeof** operator

int a = 5;

printf("%d", **sizeof(a)**); //output : 4

**&** operator

int a = 10;

printf("%d", &a);