Introduction of C Language

C Programming is an ANSI/ISO standard and powerful programming language for developing real time applications. C programming language was invented by Dennis Ritchie at the Bell Laboratories in 1972. It was invented for implementing UNIX operating system. C is most widely used programming language even today. All other programming languages were derived directly or indirectly from C programming concepts.

C – Language History

1. The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis Ritchie
2. C programming language features were derived from an earlier language called “B” (Basic Combined Programming Language – BCPL)
3. C language was invented for implementing UNIX operating system.
4. In 1978, Dennis Ritchie and Brian Kernighan published the first edition “The C Programming Language” and commonly known as K&R C.
5. In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or “ANSI C”, was completed late 1988.

WHICH LEVEL IS C LANGUAGE BELONGING TO?

There are 3 levels of programming languages. They are,

1. Middle Level languages
   Middle level languages don’t provide all the built-in functions found in high level languages, but provides all building blocks that we need to produce the result we want. Examples: C, C++.
2. High Level languages:
   High level languages provide almost everything that the programmer might need to do as already built into the language. Example: Java, Python
3. Low Level languages:
   Low level languages provides nothing other than access to the machines basic instruction set. Example: Assembler

C LANGUAGE IS A STRUCTURED LANGUAGE:

Structure oriented language:

- In this type of language, large programs are divided into small programs called functions
- Prime focus is on functions and procedures that operate on the data
- Data moves freely around the systems from one function to another

- Program structure follows “Top Down Approach”
- Examples: C, Pascal, ALGOL and Modula-2

Object oriented language:

- In this type of language, programs are divided into objects
- Prime focus is in the data that is being operated and not on the functions or procedures
- Data is hidden and cannot be accessed by external functions
- Program structure follows “Bottom UP Approach”
- Examples: C++, JAVA and C# (C sharp)

Non structure oriented language:

- There is no specific structure for programming this language. Examples: BASIC, COBOL, FORTRAN

STEPS TO WRITE C PROGRAMS AND GET THE OUTPUT:

Below are the steps to be followed for any C program to create and get the output. This is common to all C program and there is no exception whether its a very small C program or very large C program.

1. Create
2. Compile
3. Execute or Run
4. Get the Output

$$\begin{tikzcd}
\boxed{\text{First.C}} \arrow[r, "\text{Alt+F9}"] & \boxed{\text{First.obj}} \arrow[r, "\text{F9}"] & \boxed{\text{First.exe}} \arrow[r, "\text{Ctrl+F9}"] & \text{Output}\ \boxed{}
\end{tikzcd}$$

If everything is ok then C Complier gernerate a .obj file, which contains object code, with the same name. Then complier generated file .obj is passed to the linker who does the job of liking the necessary files to your program and gerenate the .exe version of your program.

C – Tokens and keywords

C tokens, Identifiers and Keywords are the basics in a C program. All are explained in this page with definition and simple example programs.

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 arepar of six types. They are,

1. Keywords (eg: int, while),
2. Identifiers (eg: main, total),
3. Constants (eg: 10, 20),
4. Strings (eg: “total”, “hello”),
5. Special symbols (eg: (), {}),
6. Operators (eg: +, /,-,*)

2. IDENTIFIERS IN C LANGUAGE:

- Each program elements in a C program are given a name called identifiers.
- Names given to identify Variables, functions and arrays are examples for identifiers. eg. x is a name given to integer variable in above program.

3. KEYWORDS IN C LANGUAGE:

- Keywords are pre-defined words in a C compiler.
- Each keyword is meant to perform a specific function in a C program.
- Since keywords are referred names for compiler, they can’t be used as variable name.

C language supports 32 keywords.

C – Constant

C Constants are also like normal variables. But, only difference is, their values can not be modified by the program once they are defined.

TYPES OF C CONSTANT:

1. Integer constants
2. Real or Floating point constants
3. Octal & Hexadecimal constants
4. Character constants
5. String constants

HOW TO USE CONSTANTS IN A C PROGRAM?

We can define constants in a C program in the following ways.

1. By "const" keyword
2. By "#define" preprocessor directive

C – Variable

- C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.
- The value of the C variable may get change in the program.
- C variable might be belonging to any of the data type like int, float, char etc.

RULES FOR NAMING C VARIABLE:

1. Variable name must begin with letter or underscore.
2. Variables are case sensitive
3. They can be constructed with digits, letters.
4. No special symbols are allowed other than underscore.
5. sum, height, _value are some examples for variable name

Type	**Syntax**
Variable declaration	data_type variable_name; Example: int x, y, z; char flat, ch;

Variable initialization	data_type variable_name = value;
	
	Example: int x = 50, y = 30; char flag = 'x', ch='l';

THERE ARE TWO TYPES OF VARIABLES IN C PROGRAM THEY ARE,

1. Local variable
2. Global variable

A local variable is a variable that is declared inside a function. A global variable is a variable that is declared outside **all** functions. A local variable can only be used in the function where it is declared. A global variable can be used in all functions.

C – DATA TYPES:

There are four data types in C language. They are,

Types	Data Types
Basic data types	Int 2 byte, char 1byte, float 4 byte , double 8 byte
Enumeration data type	Enum
Derived data type	pointer, array, structure, union
Void data type	Void

1.3.2. MODIFIERS IN C LANGUAGE:

- The amount of memory space to be allocated for a variable is derived by modifiers.
- Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable.
- For example, storage space for int data type is 4 byte for 32 bit processor. We can increase the range by using long int which is 8 byte. We can decrease the range by using short int which is 2 byte.
- There are 5 modifiers available in C language. They are,
  1. short
  2. long
  3. signed
  4. unsigned
  5. long long

Operators :-

The symbols which are used to perform logical and mathematical operations in a C program are called C operators.

TYPES OF C OPERATORS:

C language offers many types of operators. They are,

1. Arithmetic operators
2. Assignment operators
3. Relational operators
4. Logical operators
5. Conditional operators (ternary operators)
6. Increment/decrement operators
7. Special operators

CONTINUE ON TYPES OF C OPERATORS:

Click on each operator name below for detailed description and example programs.

Types of Operators	**Description**
Arithmetic_operators	These are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus
Assignment_operators	These are used to assign the values for the variables in C programs.
Relational operators	These operators are used to compare the value of two variables.
Logical operators	These operators are used to perform logical operations on the given two variables.
Conditional (ternary)	Conditional operators return one value

operators	if condition is true and returns another value is condition is false.
Increment/decrement operators	These operators are used to either increase or decrease the value of the variable by one.
Special operators	&, *, sizeof( ) and ternary operators.

Decision Control statement

- In decision control statements (if-else and nested if), group of statements are executed when condition is true. If condition is false, then else part statements are executed.
- There are 3 types of decision making control statements in C language. They are,

1. if statements
2. if else statements
3. nested if statements

EXAMPLE PROGRAM FOR NESTED IF STATEMENT IN C:

- In “nested if” control statement, if condition 1 is false, then condition 2 is checked and statements are executed if it is true.
- If condition 2 also gets failure, then else part is executed.

```
#include <stdio.h>
int main()
{
  int m=40,n=20;
  if (m>n) {
 printf("m is greater than n");
  }
  else if(m<n) {
  printf("m is less than n");
  }
  else {
  printf("m is equal to n");
  }
}
```

OUTPUT:

```
m is greater than n
```

Loop control statements

Loop control statements in C are used to perform looping operations until the given condition is true. Control comes out of the loop statements once condition becomes false.

TYPES OF LOOP CONTROL STATEMENTS IN C:
There are 3 types of loop control statements in C language. They are,

1. for
2. while
3. do-while

EXAMPLE PROGRAM (FOR LOOP) IN C:
In for loop control statement, loop is executed until condition becomes false.

```
#include <stdio.h>

int main()
{
  int i;

  for(i=0;i<10;i++)
  {
     printf("%d ",i);
  }

}
```

OUTPUT:

```
0 1 2 3 4 5 6 7 8 9
```

EXAMPLE PROGRAM (WHILE LOOP) IN C:

In while loop control statement, loop is executed until condition becomes false.

```
#include <stdio.h>

int main()
{
  int i=3;

  while(i<10)
  {
    printf("%d\n",i);
    i++;
  }

}
```

OUTPUT:

```
3 4 5 6 7 8 9
```

EXAMPLE PROGRAM (DO WHILE LOOP) IN C:

In do..while loop control statement, while loop is executed irrespective of the condition for first time. Then 2nd time onwards, loop is executed until condition becomes false.

```
#include <stdio.h>

```

```c
int main()
{
  int i=1;

  do
  {
     printf("Value of i is %d\n",i);
     i++;
  }while(i<=4 && i>=2);

}
```

OUTPUT:

```
Value of i is 1
Value of i is 2
Value of i is 3
Value of i is 4
```

Array

C Array is a collection of variables belongings to the same data type. You can store group of data of same data type in an array.

EXAMPLE FOR C ARRAYS:

- int a[10]; // integer array
- char b[10]; // character array i.e. string

EXAMPLE PROGRAM FOR ONE DIMENSIONAL ARRAY IN C:

```
#include<stdio.h>

int main()
{
  int i;
  int arr[5] = {10,20,30,40,50};

    // declaring and Initializing array in C
    //To initialize all array elements to 0, use int arr[5]={0};
    /* Above array can be initialized as below also
    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    arr[3] = 40;
    arr[4] = 50; */

  for (i=0;i<5;i++)
  {
    // Accessing each variable
    printf("value of arr[%d] is %d \n", i, arr[i]);
  }

}
```

Function

C functions are basic building blocks in a program. All C programs are written using functions to improve re-usability, understandability and to keep track on them.

WHAT IS C FUNCTION?

A large C program is divided into basic building blocks called C function. C function contains set of instructions enclosed by “{ }” which performs specific operation in a C program. Actually, Collection of these functions creates a C program.

C FUNCTION DECLARATION, FUNCTION CALL AND FUNCTION DEFINITION:

There are 3 aspects in each C function. They are,

- Function declaration or prototype – This informs compiler about the function name, function parameters and return value’s data type.
- Function call – This calls the actual function
- Function definition – This contains all the statements to be executed.

C functions aspects	**syntax**
function definition	Return_type function_name (arguments list) { Body of function; }
function call	function_name (arguments list);
function declaration	return_type function_name (argument list);

SIMPLE EXAMPLE PROGRAM FOR C FUNCTION:

```
#include<stdio.h>
// function prototype, also called function declaration
float square ( float x );
// main function, program starts from here

int main( )
{
   float m, n ;
   printf ( "\nEnter some number for finding square \n");
```

```
    scanf ( "%f", &m ) ;
    // function call
    n = square ( m ) ;
    printf ( "\nSquare of the given number %f is %f",m,n );
}

float square ( float x )   // function definition
{
    float p ;
    p = x * x ;
    return ( p ) ;
}
```

OUTPUT:

```
Enter some number for finding square
2
Square of the given number 2.000000 is 4.000000
```

Pointer

- Pointers in C language is a variable that stores/points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.
- Pointer Syntax : data_type *var_name; Example : int *p; char *p;
- Where, * is used to denote that “p” is pointer variable and not a normal variable.

KEY POINTS TO REMEMBER ABOUT POINTERS IN C:

- Normal variable stores the value whereas pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address.
- Always C pointer is initialized to null, i.e. int *p = null.
- The value of null pointer is 0.
- & symbol is used to get the address of the variable.
- * symbol is used to get the value of the variable that the pointer is pointing to.
- If a pointer in C is assigned to NULL, it means it is pointing to nothing.
- Two pointers can be subtracted to know how many elements are available between these two pointers.
- But, Pointer addition, multiplication, division are not allowed.
- The size of any pointer is 2 byte (for 16 bit compiler).

```
#include <stdio.h>
int main()
{
   int *ptr, q;
   q = 50;
   /* address of q is assigned to ptr */
   ptr = &q;
   /* display q's value using ptr variable */
   printf("%d", *ptr);
   return 0;
}
```

POINTERS IN C – PROGRAM OUTPUT:

```
50
```

Structure

C Structure is a collection of different data types which are grouped together and each element in a C structure is called member.

- If you want to access structure members in C, structure variable should be declared.
- Many structure variables can be declared for same structure and memory will be allocated for each separately.
- It is a best practice to initialize a structure to null while declaring, if we don't assign any values to structure members.

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

struct student
{
            int id;
            char name[20];
            float percentage;
} record;

int main()
{

            record.id=1;
            strcpy(record.name, "Raju");
            record.percentage = 86.5;

            printf(" Id is: %d \n", record.id);
            printf(" Name is: %s \n", record.name);
            printf(" Percentage is: %f \n", record.percentage);
            return 0;
}
```

OUTPUT:

```
Id is: 1
Name is: Raju
Percentage is: 86.500000
```

Some Important Question of ‘C’ Programming for IA Exam

(1) Which of the following statements is TRUE?
(A) C is an object oriented programming language
(B) **struct** is a legal name for a variable in a C program
(C) The variable declaration **“car ford;”**is illegal in any C program
(D) A function may return one value at a time
(E) A variable name may begin with the symbol #

(2) Functions are useful in a program because
(A) they increase the execution speed of a program
(B) they make the program organization clearer
(C) a program cannot compile if it does not contain a least one function
(D) the compiler automatically switches to the more powerful function mode when compiling a program with functions
(E) the more functions in a program the less bugs

(3-4) Consider the following code fragment

```
int pH;
scanf(“%i”,&pH); /* line A */
if (pH<7)
  printf(“Acidic\n”);
  if (pH<2)
    printf(“Very Acidic\n”);
else
  printf(“Alkaline\n”);
  if (pH>12)
    printf(“Very Alkaline\n”);
else if (pH%7==0)
   printf(“Neutral\n”);
```

(3) Assume that the user inputs the integer 14 on line A. What is the output displayed once the code is executed?

(A) Neutral
(B) Acidic
    Neutral
(C) Alkaline
    Very Alkaline
(D) Very Alkaline
(E) Alkaline
    Very Alkaline
    Neutral

(4) The above code is executed once again. Assume that the user inputs the integer 0 on line A. What is the output once the code is executed?

(A) Neutral
(B) Acidic
Very Acidic
Neutral

(C) Acidic
Very Acidic

(D) Acidic
Neutral

(E) Very Acidic
Neutral

(5) Consider the following code fragment (read carefully):

```
scanf(“%i”,&i);  /* line A */
if ( (i==1)  || (i=2) )
   i=i+1;
printf(“i=%i”,i); /* line B */
```

Assume that the user enters **1** through the **scanf** on line A. What is printed by the **printf** statement on line B?

(A) **i=1**
(B) **i=2**
(C) **i=3**
(D) **i=i**
(E) This program cannot execute. A compilation error is generated.

(6) Among the following, which one is NOT a valid C identifier?

(A) concrete
(B) brick
(C) stone
(D) struct
(E) truss

(7). C is called C because
(A) it is not a very good language and as such deserves a C grade
(B) it started as the B language (B for Bell laboratories) and later evolved into the C language (big hint: This could be the right answer)
(C) the programmer who wrote it was fond of whistling the C note
(D) all the other letters of the alphabet were already used for names of computer languages
(E) in its early developments, the programs written in C often crashed. The language was nicknamed C as a short for Crash.

8. We can insert pre written code in a C program by using

A. #read
B. #get
C. #include
D. #put

9. The first expression in a for loop is

A. Step value of loop
B. Value of the counter variable
C. Any of above
D. None of above

10. Break statement is used for

A. Quit a program
B. Quit the current iteration
C. Both of above
D. None of above

11. Continue statement used for

a) To continue to the next line of code
b) To stop the current iteration and begin the next iteration from the beginning
c) To handle run time error
d) None of above

12. What will be output of

```
#include
void main()
{
char test =`S`;
printf("\n%c",test);
}
```

A. S
B. Error
C. Garbage value
D. None of above

13. Due to variable scope in c

E. Variables created in a function cannot be used another function
F. Variables created in a function can be used in another function
G. Variables created in a function can only be used in the main function
H. None of above

14. What will be the output of following program

```
#include
main()
{
int x,y = 10;
x = y * NULL;
printf(\"%d\",x);
}
```

I. error
J. 0
K. 10
L. Garbage value

15. Which of the following below is/are valid C keywords

A. integer
B. int
C. null
D. none of above

16. total number of keywords in C are

A. 30
B. 32
C. 48
D. 132

17. What is use of \r in c

A. used to insert a vertical tab
B. used to insert a tab
C. places cursor at the end of line
D. places cursor at the start of line

18. Difference between structure and union is

   A. We can define functions within structures but not within a union
   B. We can define functions within union but not within a structure
   C. The way memory is allocated
   D. There is no difference

19. To access the members of structure which symbol is used

A. *
B. -
C. ,
D. .

20. A member is a

A. Variable in a structure
B. Datatype of structure
C. Structure pointer
D. None of above

Answer :-

1) D
2) B
3) C
4) B
5) B
6) D
7) B
8) C
9) B
10) B
11) B
12) A
13) A
14) B
15) B
16) B
17) D
18) C
19) D
20) A