C LANGUAGE AND ITS DIFFERENT TYPES OF FUNCTIONS

Manish
Dronacharya College Of Engineering,
Maharishi Dayanand University, Gurgaon, Haryana, India

***Abstract*- C – Language History: The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis Ritchie C programming language features were derived from an earlier language called "B" (Basic Combined Programming Language – BCPL) C language was invented for implementing UNIX operating system In 1978, Dennis Ritchie and Brian Kernighan published the first edition "The C Programming Language" and commonly known as K&R C 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.**

I. C PROGRAMMING LANGUAGE STANDARDS

- C89/C90 standard – First standardized specification for C language was developed by the American National Standards Institute in 1989. C89 and C90 standards refer to the same programming language.
- C99 standard – Next revision was published in 1999 that introduced new features like advanced data types and other changes.

II. FEATURES OF C PROGRAMMING LANGUAGE

- Reliability
- Portability
- Flexibility
- Interactivity
- Modularity
- Efficiency and Effectiveness

III. USES OF C PROGRAMMING LANGUAGE

The C programming language is used for developing system applications that forms a major portion of operating systems such as Windows, UNIX and Linux. Below are some examples of C being used.

- Database systems
- Graphics packages
- Word processors
- Spreadsheets
- Operating system development
- Compilers and Assemblers
- Network drivers
- Interpreters
-

C – Data Type

- C data types are defined as the data storage format that a variable can store a data to perform a specific operation.
- Data types are used to define a variable before to use in a program.
- Size of variable, constant and array are determined by data types.

IV. C – DATA TYPES

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

1. Basic data types in C:

1.1. Integer data type:

- Integer data type allows a variable to store numeric values.
- "int" keyword is used to refer integer data type.
- The storage size of int data type is 2 or 4 or 8 byte.
- It varies depend upon the processor in the CPU that we use. If we are using 16 bit processor, 2 byte (16 bit) of memory will be allocated for int data type.
- Like wise, 4 byte (32 bit) of memory for 32 bit processor and 8 byte (64 bit) of memory for 64 bit processor is allocated for int datatype.
- int (2 byte) can store values from -32,768 to +32,767
- int (4 byte) can store values from -2,147,483,648 to +2,147,483,647.

- If you want to use the integer value that crosses the above limit, you can go for "long int" and "long long int" for which the limits are very high.

Note:

- We can't store decimal values using int data type.
- If we use int data type to store decimal values, decimal values will be truncated and we will get only whole number.
- In this case, float data type can be used to store decimal values in a variable.

1.2. Character data type:

- Character data type allows a variable to store only one character.
- Storage size of character data type is 1. We can store only one character using character data type.
- "char" keyword is used to refer character data type.
- For example, 'A' can be stored using char datatype. You can't store more than one character using char data type.
- Please refer C – Strings topic to know how to store more than one characters in a variable.

1.3. Floating point data type:

Floating point data type consists of 2 types. They are,

1. float
2. double

1. float:

- Float data type allows a variable to store decimal values.
- Storage size of float data type is 4. This also varies depend upon the processor in the CPU as "int" data type.
- We can use up-to 6 digits after decimal using float data type.
- For example, 10.456789 can be stored in a variable using float data type.

2. double:

- Double data type is also same as float data type which allows up-to 10 digits after decimal.
- The range for double datatype is from 1E–37 to 1E+37.

1.3.2. Modifiers in C:

- 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

2. Enumeration data type in C:

- Enumeration data type consists of named integer constants as a list.
- It start with 0 (zero) by default and value is incremented by 1 for the sequential identifiers in the list.
- Enum syntax in C:

enum identifier [optional{ enumerator-list }];

- Enum example in C:

enum month { Jan, Feb, Mar }; or

/* Jan, Feb and Mar variables will be assigned to 0, 1 and 2 respectively by default */
enum month { Jan = 1, Feb, Mar };
/* Feb and Mar variables will be assigned to 2 and 3 respectively by default */
enum month { Jan = 20, Feb, Mar };
/* Jan is assigned to 20. Feb and Mar variables will be assigned to 21 and 22 respectively by default */

- The above enum functionality can also be implemented by "#define" preprocessor directive as given below. Above enum example is same as given below.

#define Jan 20;
#define Feb 21;
#define Mar 22;

3. Derived data type in C:

- Array, pointer, structure and union are called derived data type in C language.
- To know more about derived data types, please visit "C – Array" , "C – Pointer" , "C – Structure" and "C – Union" topics in this tutorial.

4. Void data type in C:

- Void is an empty data type that has no value.
- This can be used in functions and pointers.
- Please visit "C – Function" topic to know how to use void data type in function with simple call by value and call by reference example programs.

V. C – 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. You can learn below concepts of C functions in this section in detail.

1. What is C function?
2. Uses of C functions
3. C function declaration, function call and definition with example program
4. How to call C functions in a program?
    1. Call by value
    2. Call by reference
5. C function arguments and return values
    1. C function with arguments and with return value
    2. C function with arguments and without return value
    3. C function without arguments and without return value
    4. C function without arguments and with return value
6. Types of C functions
    1. Library functions in C
    2. User defined functions in C

1.Creating/Adding user defined function in C library

7. Command line arguments in C
8. Variable length arguments in C

1. 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.

2. Uses of C functions:

- C functions are used to avoid rewriting same logic/code again and again in a program.
- There is no limit in calling C functions to make use of same functionality wherever required.
- We can call functions any number of times in a program and from any place in a program.
- A large C program can easily be tracked when it is divided into functions.
- The core concept of C functions are, re-usability, dividing a big task into small pieces to achieve the functionality and to improve understandability of very large C programs.

3. 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.

*** example program for C function:**

- As you know, functions should be declared and defined before calling in a C program.
- In the below program, function "square" is called from main function.
- The value of "m" is passed as argument to the function "square". This value is multiplied by itself in this function and multiplied value "p" is returned to main function from function "square".

```
#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
```

4. How to call C functions in a program?

There are two ways that a C function can be called from a program. They are,

1. Call by value
2. Call by reference

1. Call by value:

- In call by value method, the value of the variable is passed to the function as parameter.
- The value of the actual parameter can not be modified by formal parameter.
- Different Memory is allocated for both actual and formal parameters. Because, value of actual parameter is copied to formal parameter.

Note:

- Actual parameter – This is the argument which is used in function call.
- Formal parameter – This is the argument which is used in function definition

Example program for C function (using call by value):

- In this program, the values of the variables "m" and "n" are passed to the function "swap".
- These values are copied to formal parameters "a" and "b" in swap function and used.

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int a, int b);

int main()
{
   int m = 22, n = 44;
   // calling swap function by value
   printf(" values before swap  m = %d \nand n = %d", m, n);
   swap(m, n);
}

void swap(int a, int b)
{
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
   printf(" \nvalues after swap m = %d\n and n = %d", a, b);
}
```

Output:

```
values before swap m = 22
and n = 44
values after swap m = 44
and n = 22
```

2. Call by reference:

- In call by reference method, the address of the variable is passed to the function as parameter.
- The value of the actual parameter can be modified by formal parameter.
- Same memory is used for both actual and formal parameters since only address is used by both parameters.

Example program for C function (using call by reference):

- In this program, the address of the variables "m" and "n" are passed to the function "swap".
- These values are not copied to formal parameters "a" and "b" in swap function.
- Because, they are just holding the address of those variables.
- This address is used to access and change the values of the variables.

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int *a, int *b);

int main()
{

   int m = 22, n = 44;
   //  calling swap function by reference
   printf("values before swap m = %d \n and n = %d",m,n);
   swap(&m, &n);
}
```

```
void swap(int *a, int *b)
{
   int tmp;
   tmp = *a;
   *a = *b;
   *b = tmp;
   printf("\n values after swap a = %d \nand b = %d", *a, *b);
}
```

Output:

```
values before swap m = 22
and n = 44
values after swap a = 44
and b = 22
```

REFRENCES

[1] ANSI 89 – American National Standards Institute, American National Standard for Information Systems Programming Language C, 1989.

[2] Kernighan 78 – B. W. Kernighan and D. M. Ritchie, The C Programming Language, Prentice-Hall: Englewood Cliffs, NJ, 1978. Second edition, 1988.

[3] Thinking 90 – C* Programming Guide, Thinking Machines Corp. Cambridge Mass., 1990.