International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

C Programming Language Better Understanding

Mr. Arun kumar A

Technical Trainer, DCPD, Chandigarh Group of Colleges, Jhanjeri

Abstract
C Programming language one of the oldest and most famous programming language in computer science field invented at AT&T bells laboratory by Dennis Ritchie way back in the year 1972. It was first developed for developing an operating system unix which is famous in networking applications. Now also C programming language hold good in system side development in linux also. It has wide application and used by many learners as starting for learning programming language. It is simple and easy to learn programming language for beginners. C has pointer concept which many learners find difficult but once the master it they get to know it easy. C is compiler based programming language before starting to learn check for good compiler both windows and linux it supports.

Keywords: C programming language, printf, understanding, simple to learn

1. Introduction

C programming language is the easiest and good programming language for starters although many programming languages have been evolved like C++ and Java which is Object oriented programming and compiler based and we have Python which is interpreted programming language. C++ is used in game development and Java with its security and Robust feature used in standalone application, Enterprise application and mobile development. But if you master C Programming language then others follow its construct like if else, switch case, continue, break etc.,

C programming language has many interesting things to start with learning programming for example a printf() function returns a value which is number of successful characters printed on the console screen. C basically uses header files which is where it holds the function prototype and use any built in functions in c we have to first include the header file in the header section. Built in libraries use angle bracket and it is fetched from location which compiler sets during installation and user defined header files are to be given within double quotes.

There are many header files in C programming language stdio.h, conio.h, time.h, limits.h, float.h, string.h and many more to tell. We can do graphics programming also in C and as told C is famous in system side programming it is procedural oriented or structured programming language and C++ is nothing but C with Classes where C++ supports object oriented programming language.

Structure of C Programming language

Header file	->	#include <stdio.h>
Global declaration	->	int a;
Function declaration(or)		
Function prototype	->	int sum(int,int);
Macros	->	#define PI 3.14

```
Main                    ->          int main(){
Local variable          ->           int b;
Calling functions       ->          printf("%d",a);
Return statement        ->          return 0;
End of main             ->           }
Function definition     ->      void sum(int a,int b){
Return statement        ->            return a+b;
End of Function         ->      }
```

This is the basic structure of C programming language
It contains header section where we include header files and we have Global declaration section it is optional function declaration where we declare the functions whose definition will be given at the bottom in subroutine section and we have macros which is also optional and main function which is the entry point of execution of our application here int main so the last statement must be return 0. 0 tells successful execution of our application.

Printf() and scanf() function

As discussed printf() function returns number of successful characters printed on the console screen. This printf("Welcome") returns an integer value 7 as spelled Welcome spaces also considered as one character.

If we give printf("%d",(printf("%d",4567)-printf("%d",34)); this statement has three printf function nested one inside another. Can we subtract two printf() function in C programming language, yes its possible since printf() function returns an integer value which is number of successful characters printed on the console screen printf("%d",4567) returns 4 since 4 characters are printed and printf("%d",34) returns 2 so when we subtract we get 4-2 = 2 and it is nested inside another printf function which prints this return value and output is 2.

Similarly can we subtract two scanf() function, yes scanf() function returns number of inputs it receive For example if we give int a = scanf("%d%d",&b,&c) it returns 2 since it has successfully read two inputs and we use ampersand & symbol in scanf to place the value in the address it points printf("%d",scanf("%d%d",&a,&b)); will receive two inputs and returns 2 and it will be printed.

Keep experimenting the things so that we will get better understanding.

sizeof()

sizeof() does not valuate any expression it just returns the sizeof datatype it is passed.

sizeof(printf("Hello World")) it just returns 2 or 4 that is size of integer and size of integer varies from compiler to compiler it is 2 bytes for 16 bit compiler and 4 bytes for 32 bit compiler.

If we give sizeof(4*3%4) it will not be error and sizeof() just returns 2 or 4 since it is integer and sizeof integer is 2 bytes or 4 bytes.

if()

In C programming language anything zero is false and anything other than zero is true for example 0.2 is true and -5 is also true and 0 is false.

In if() conditional statement we test a condition, if it is true statements within the if block gets executed and if false statements within the else block gets executed.

if(1) the condition is true so statements within if block gets executed.
if(printf("Hello")) we just discussed print() returns an integer value which is number of successful characters printed on the console screen it returns 5 so condition is true.

```
if(printf("%s","\0")){
printf("Hai");
}else{
printf("Bye");
}
```

Here in this case we try to print null character so printf returns 0 condition is false else gets executed and we get "Bye" as output

```
float a = 0.2f;
if(a==0.2){
printf("Hai");
}else{
printf("Bye");
}
```

This prints "Bye" though rvalue and lvalue are same but why rvalue is double for float and double default data type is double a is float but 0.2 is double we compare float with double so the result is false.
But how can we say result is false convert 0.2 into binary you will get the result, for float first 7 fractional digits are considered and for double 15 fractional digits are considered.

0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6
0.6 * 2 = 1.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 *2 = 1.6

If u check first 7 digits and 15 digits after converting to binary will not be same.
Just by experimenting these things we get deeper understanding and clear about language.

Pointer concept

When talking about pointer we have function pointer, void or generic pointer, null pointer, dangling pointer and so on.
There is a Dynamic memory allocation for which we use stdlib.h header file in this we have functions like malloc() and calloc() and realloc() which returns a void pointer which can be casted to any of the data types.
What is void pointer

```
void *a;
int b = 20;
float c = 30.0f;
a = &b;
printf("a = %d\n",*(int *)a);
a = &c;
```

printf("a = %f\n",*(float *)a);
As you can see first void pointer initialized to integer and value printed next it is assigned to float and value is got back using proper casting.
int *a = (int *)malloc(2*sizeof(int));
printf("%d",sizeof(a));
Here you see malloc returns void pointer and it can be cast to any data type and sizeof a is 8 bytes.

Structure and Unions

Structures and unions are both same in C programming language in structure size of data type is found by size of each of its members where as in union which member has maximum size only for it size is allotted and rest of the members use same memory. We can use typedef keyword to allocate different names to structure and union instead of calling as struct mystruct.

References

1. The C Programming Language - Dennis Ritchie, Brian Kernighan
2. C Programming: A Modern Approach: K. N King
3. {Anderson, 1980} B. Anderson, Type syntax in the language C: an object lesson in syntactic innovation, SIGPLAN Notices, Vol. 15, No. 3, Mar. 1980, pp. 21--27.
4. {Kernighan, 1981} B. W. Kernighan, Why Pascal is not my favorite programming language, Computer Science Technical Report No. 100, AT&T Bell Laboratories, 1981.