INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

**ISSN: 2454-132X**
**Impact factor: 4.295**
***(Volume3, Issue2)***
Available online at: www.ijariit.com

# A Survey on How Main Method Working In C Programming

**G. Anupama**
*Department of CSE , KKR&KSR Institute of Technology & Sciences*
anupama.3g@gmail.com

**M. Phani Anusha**
*Department of CSE , KKR&KSR Institute of Technology & Science*
phanianusha32@gmail.com

**Y. Vasanthi**
*Department of CSE , KKR&KSR Institute of Technology & Sciences*
vasanthi.yelavarthy@gmail.com

***Abstract: By the completion of this paper study we have the better idea about the main method. Generally many people have the doubt about main method is predefined or user defined method? If it is user defined where is the prototype of that method? Or if it is predefined in which header file its prototype is present? By the end of this paper study we have the clarity of is main method is a user defined or predefined method. And also this paper give the clarity about user defined and predefined methods in C Programming.***

***Keywords: Prototype, User Defined, Predefined, Return Type, Arguments, Invoking, Command Line Arguments.***

## 1. Introduction

In C Language, the execution of the programme starts with main method. All other methods are invoked by their method calls but in the case of main this is not happened. i.e.., every method has invoked by their corresponding method calls only but in the case of main method there is need of method call necessary why because automatically main method is invoked when programme go to execution state.

In C language there three major concerns according to methods.

- Method Declaration or Prototype of method
- Method Calling
- Method Definition

If you want to use a method in programme then that method have these three components if they are user defined.

**Ex:**

```
//Sample programme to show methods components
#include<stdio.h>
#include<conio.h>
void display(int a);//Method Declaration
void main()
{
        ----------
        display (a);//Method Invoking
        ---------
}
void display(int b)//Method Definition
{
        -----
}
```

Here we can observe that actual parameter name is different from formal parameters.

The actual parameter name and formal parameter name are may be same or not. There is no restriction to having both actual and formal have the same name. So that the actual parameter name is 'an' in method invoking and formal parameter name is 'b' in the method definition.

 

In C Language we have two types methods.

- Predefined
- User Defined

Predefined methods definition and declaration is built in the system library. The user work is to simply call that method and use it. There is no need to specify any definition to the predefined method. Some examples to predefined methods are print(),scan(),fetch(),clrscr(). All these methods are predefined once so by including header files according to methods is sufficient. See the following example program to know how they are calling.

Ex:

```
//Sample predefined method usage program
#include<stdio.h>
#include<conio.h>
void main()
{
        int var;
        clrscr();
        printf("\n Enter a value:");
        scanf("%d",&var);
        printf("\n Entered a value is:%d",var);
        getch();
}
```

Here methods printf(),scanf() and getch() are predefined methods which are defined in stdio.h header file. And clrscr() is predefined in conio.h header file. In the above program, all predefined methods are not defined and declared by the programmer in the program. Just we use the services provided by these predefined methods by invoking the method.

User defined methods are defined by the users , so the three components of the methods are defined and declared and also invoking by the user itself. The Declaration of the user-defined method is before the main method starting. And the definition of the user-defined method is the main method i.e.. generally after the main method. When the user-defined method is invoked by another method then complier checks for the declaration & definition of that invoked method is matched or not. If matched then the invoking method is called successfully, Otherwise, it will give an error message.

**Ex:**

```
//Sample user defined programme
#include<stdio.h>
#include<conio.h>
int add(int ,int );
void main()
{
        int a,b,c;
        ----------
        c=add(a,b);
       ------
        getch();
}
int add(int x,int y)
{
        return(x+y);
}
```

Here add() is the user defined method invoked by main() method. When the invoking of add() method is done then the compiler checking for the prototype of that method and also the definition of that method. If the name, number of arguments, type of the arguments and return type of the method in method calling, method prototype, method definition is matched then the method is invoked. Otherwise, method is not invoked and give compiler error.

**2.Is the main() method is user defined or predefined?**

Many of the people have a confusion about main() method is either predefined or user defined method. The main() method is a user-defined method which has the predefined prototype. The prototype of the main() method is predefined by the compiler. So that we can say that in C Programming main() is user defined as well as predefined method why because it's prototype is predefined.

The execution of the program is started with the main() method. So that main() method is said to be an entry point of the program. There is no need to call the main() method, why because the operating system automatically invokes the main() method when the program is executed.

Some of the main() prototypes's given below:

- main() method without return type and without arguments

  void main(void)

 

```
{
        /*Declaration part*/
        /*Execution part*/
}
```

- main() method with return type and without arguments

```
int main(void)
{
        /*Declaration part*/
        /*Execution part*/
}
```

- main() method with return type and with command line arguments

```
int main(int argc,char *argv[])
{
        /*Declaration part*/
        /*Execution part*/
}
        (or)
int main(int argc,char **argv)
{
        /*Declaration part*/
        /*Execution part*/
}
```

**3.Return type and arguments of main() method:**

Default return type of any method is int. In general, main method return type is void in C language. But the return type of the main() must be int. It should return a value 0 on successful completion of the program. Otherwise, it returns a value 1 on error or failure.

Based on compiler C program main() method is executed i.e., in Microsoft visual studio you can declare with void or int data type. But in other compilers like DEV_C++, you need to declare with int ; Otherwise, it will give you an error message. In GCC compiler , the main() method return type is void then it will give a warning. But the program is executed. Why because every compiler has the start-up code to execute the main while programming execution. In some compilers, the start-up code will assume that the main() method is defined in a standard manner and it doesn't care about the return value of the main() method. So the main() method with a void return type or int return type is considered in some compilers.

The statement exit( ) also returns values to the operating systemas the return( ) in main( ). The exit takes only two values 0 and 1. (Many people use exit(2), exit(3)…. All these are wrong!) So exit should be used as:

a) For normal termination exit ( 0 ); or exit( EXIT_SUCCESS);

b) For abnormal termination exit( 1 ); or exit( EXIT_FAILURE);

**4.Arguments to main() method:**

main( ) should be declared without any arguments or with two arguments:

int main( void ) or int main( int argc, char *argv[] )

Here "large" represents the number of arguments passed to the program. And "large" is one-dimensional array of strings. And argv[0] always represent the name of the program. So the argc is least one. The first argument of main() method is int, second one a character array .

**Ex:** Let the program is shown below

```
#include<stdio.h>
#include<conio.h>
int main(int argc , char *argv[])
{
        printf("\n Name of the program:%s",argv[0]);
        if(argc==2)
                printf("\n The argument passed is %s",argv[1]);
        else if(argc>2)
                printf("\n Too many arguments");
        else
                printf("\n One argument");
}
```

Then the output Analysis is shown below

```
$ ./a.out "test1 test2"
Name of the program :./a.out
The argument passed is test1 test2
```

These arguments are called command line arguments, which are used from outside of the program to control the program.

 

## CONCLUSION

Finally, we can say that the main() method is user defined method with the predefined prototype. And its return type is int. It may or may not take any arguments. Once the user has that clarity, then they may use the argument main () method effectively for controlling their program. And also the user may know the better utilization of command line arguments in their program.

## REFERENCES

[1]Programming in C by Bharat Kinariwala, Tep Dobry

[2]Programming in C: UNIX System Calls and Subroutines using C, by A. D. Marshall - Cardiff University, 1999

[3]Programming in C UNIX System Calls and Subroutines using C. A. D. Marshall 1994-2005 *Substantially Updated March 1999*

[4]http://www.cs.utah.edu/~germain/PPS/Topics/C_Language/c_functions.html

[5]http://www.dummies.com/programming/c/the-importance-of-the-main-function-in-c-programming/

 