

C programming

Deva

January 18, 2025

Contents

1	History	1
2	Prerequisites for writing first C program	2
2.1	Header Files:	2
2.2	Main Function:	2
2.3	Variable Declarations:	2
2.4	Statements and Expressions:	2
2.5	Comments:	2
2.6	Functions:	2
2.7	Return Statement:	2
2.8	Standard Input/Output:	2
2.9	Additional Information:	2
2.10	Preprocessor Directives:	3
2.11	Data Types:	3
2.12	Control Structures:	3
2.13	Error Handling:	3
2.14	Modularization:	3
3	First Program	3
3.1	#include <stdio.h>:	3
3.2	int main()	3
3.3	printf("Hello World!");	4
3.4	return 0;	4

1 History

History of C language is interesting to know. Here we are going to discuss a brief history of the c language.

C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

Dennis Ritchie is known as the founder of the c language.

It was developed to overcome the problems of previous languages such as B, BCPL, etc.

Initially, C language was developed to be used in UNIX operating system. It inherits many features of previous languages such as B and BCPL.

2 Prerequisites for writing first C program

2.1 Header Files:

The `#include` directives at the beginning of the program are used to include header files. Header files provide function prototypes and definitions that allow the C compiler to understand the functions used in the program.

2.2 Main Function:

Every C program starts with the main function. It is the program's entry point, and execution starts from here. The main function has a return type of `int`, indicating that it should return an integer value to the operating system upon completion.

2.3 Variable Declarations:

Before using any variables, you should declare them with their data types. This section is typically placed after the main function's curly opening brace.

2.4 Statements and Expressions:

This section contains the actual instructions and logic of the program. C programs are composed of statements that perform actions and expressions that compute values.

2.5 Comments:

Comments are used to provide human-readable explanations within the code. They are not executed and do not affect the program's functionality. In C, comments are denoted by `//` for single-line comments and `/* */` for multi-line comments.

2.6 Functions:

C programs can include user-defined functions and blocks of code that perform specific tasks. Functions help modularize the code and make it more organized and manageable.

2.7 Return Statement:

Use the return statement to terminate a function and return a value to the caller function. A return statement with a value of 0 typically indicates a successful execution in the main function, whereas a non-zero value indicates an error or unexpected termination.

2.8 Standard Input/Output:

C has library functions for reading user input (`scanf`) and printing output to the console (`printf`). These functions are found in C programs and are part of the standard I/O library (`stdio.h` header file). It is essential to include these fundamental features correctly while writing a simple C program to ensure optimal functionality and readability.

2.9 Additional Information:

There is some additional information about the C programs. Some additional information is as follows:

2.10 Preprocessor Directives:

C programs often include preprocessor directives that begin with a `#` symbol. These directives are processed by the preprocessor before actual compilation and are used to include header files, define macros, and perform conditional compilation.

2.11 Data Types:

C supports data types such as `int`, `float`, `double`, `char`, etc. It depends on the program's requirements, and appropriate data types should be chosen to store and manipulate data efficiently.

2.12 Control Structures:

C provides control structures like `if-else`, `while`, `for`, and `switch-case` that allow you to make decisions and control the flow of the program.

2.13 Error Handling:

Robust C programs should include error-handling mechanisms to handle unexpected situations gracefully. Techniques like exception handling (using `try-catch` in C++) or returning error codes are commonly employed.

2.14 Modularization:

As programs grow in complexity, it becomes essential to modularize the code by creating separate functions for different tasks. This practice improves code reusability and maintainability.

Remember, the architecture and complexity of a C program can vary significantly depending on the specific application and requirements. The outline is a general overview of a simple C program's structure.

3 First Program

Listing 1: Example C++

```
#include <stdio.h>
int main(){
    printf("Hello_C_Language");
    return 0;
}
```

Let us first study the various parts of this C program:

3.1 `#include <stdio.h>`:

In this line, the program includes the standard input/output library (`stdio.h`) due to the preprocessor directive. For input and output tasks, the `stdio.h` library contains methods like `printf` and `scanf`.

3.2 `int main()` ...

:

It is the main function which is the entry point of the C program. The program starts executing from the beginning of the main function.

3.3 printf("Hello World!");

:

Use the printf() function to print formatted output to the console. In this example, the string "Hello, C Language" is printed, followed by a newline character (n) which moves the pointer to the following line after the message is displayed.

3.4 return 0;

;

When the return statement is 0, the program has been completed. When determining the state of a program, the operating system frequently uses the value returned by the main function. A return value of 0 often indicates that the execution was successful.

After compilation and execution, this C program will quit with a status code 0 and output "Hello, C Language" to the terminal.

The "Hello, C Language" program is frequently used as an introduction to a new programming language since it introduces learners to essential concepts such as text output and the structure of a C program and provides a rapid way to validate that the working environment is correctly set up.

To write, compile, and run your first C program, follow these steps:

1. **Open a text editor** Open a text editor of your choice, such as Notepad, Sublime Text, or Visual Studio Code. It will be where you write your C code.
2. **Write the C program** Now, copy and paste the following code into the text editor:

Listing 2: Example C++

```
#include <stdio.h>
int main(){
    printf("Hello_C_Language");
    return 0;
}
```

3. **Save the file** After that, save the file with a .c extension such as first_program.c. This extension indicates that it is a C source code file.
4. **Compile the program** Now, compile the program in the command prompt.
5. **Run the program** After successful compilation, you can run the program by executing the generated executable file. Enter the following command into the terminal or command.

prompt: ./first_program The program will execute, and you will see the output on the console:

Output:

Hello, C Language