Unit 3 : Fundamentals of C programming

3.1 Introduction to C language

C is a procedural programming language. It was initially developed by Dennis Ritchie in 1972. It was mainly developed as a system programming language to write an operating system. The main features of the C language include low-level memory access, a simple set of keywords, and a clean style. These features make C language suitable for system programming like an operating system or compiler development.

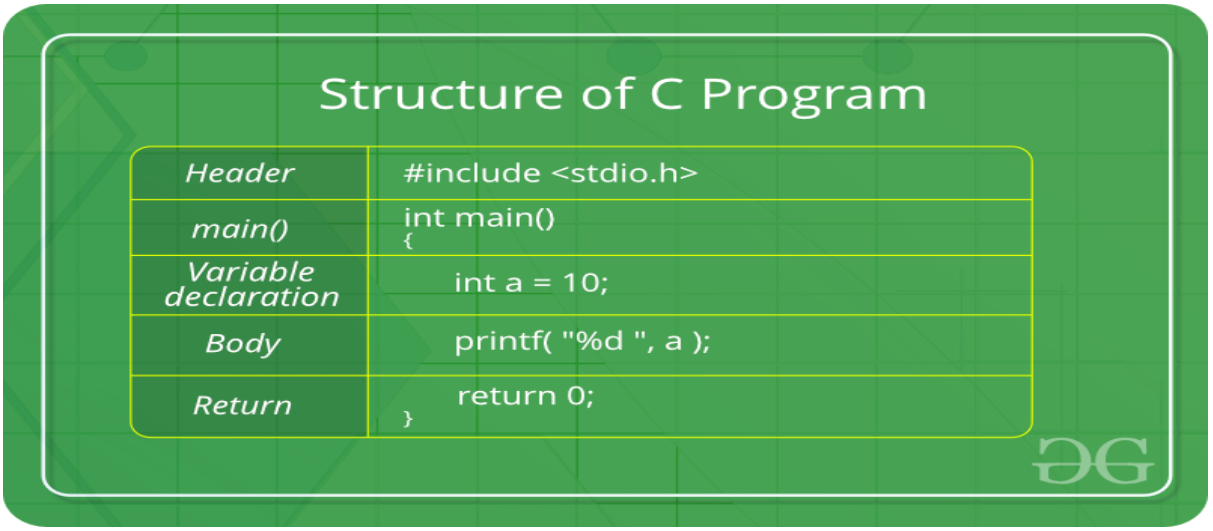
Many later languages have borrowed syntax/features directly or indirectly from the C language. Like the syntax of Java, PHP, JavaScript, and many other languages are mainly based on the C language. C++ is nearly a superset of C language (Few programs may compile in C, but not in C++).

Beginning with C programming:

Structure of a C program

After the above discussion, we can formally assess the structure of a C program. By structure, it is meant that any program can be written in this structure only. Writing a C program in any other structure will hence lead to a Compilation Error.

The structure of a C program is as follows:



The components of the above structure are:

Header Files Inclusion: The first and foremost component is the inclusion of the Header files in a C program.

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

- stddef.h – Defines several useful types and macros.
- stdint.h – Defines exact width integer types.
- stdio.h – Defines core input and output functions
- stdlib.h – Defines numeric conversion functions, pseudo-random network generator, memory allocation
- string.h – Defines string handling functions
- math.h – Defines common mathematical functions

Main Method Declaration: The next part of a C program is to declare the main() function. The syntax to declare the main function is:

Syntax to Declare the main method:

```
int main(){  }
```

Variable Declaration: The next part of any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in the C program, no variable can be used without being declared. Also in a C program, the variables are to be declared before any operation in the function.

Example:

```
int main()
{
   int a;
     float price;
…}
```

Body: The body of a function in the C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

Example:

```
int main()
{
   int a=8;

   printf(" The value of a is :%d", a);
.
.}
```

Return Statement: The last part of any C program is the return statement. The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

Example:

```
int main()
{
   int a;

   printf("%d", a);

   return 0;
}
```

Writing first program:

Following is first program in C

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

Output

```
Hello, World!
```

How "Hello, World!" Does the program work?

- The #include is a preprocessor command that tells the compiler to include the contents of the stdio.h (standard input and output) file in the program.
- The stdio.h file contains functions such as scanf() and printf() to take input and display output respectively.
- If you use the printf() function without writing #include <stdio.h>, the program will not compile.
- The execution of a C program starts from the main() function.
- printf() is a library function to send formatted output to the screen. In this program, printf() displays Hello, World! text on the screen.
- The return 0; statement is the "Exit status" of the program. In simple terms, the program ends with this statement.

A Brief History of C Programming

The Beginning

The C programming language came out of Bell Labs in the early 1970s. According to the Bell Labs paper The Development of the C Language by Dennis Ritchie, “The C programming language was devised in the early 1970s as a system implementation language for the nascent Unix operating system. Derived from the typeless language BCPL, it evolved a type structure; created on a tiny machine as a tool to improve a meager programming environment.” Originally, Ken Thompson, a Bell Labs employee, desired to make a programming language for the new Unix platform. Thompson modified the BCPL system language and created B. However, not many utilities were ever written in B due to its slow nature and inability to take advantage of PDP-11 features in the operating system. This led to Ritchie improving on B, and thus creating C.

What is the C character set?

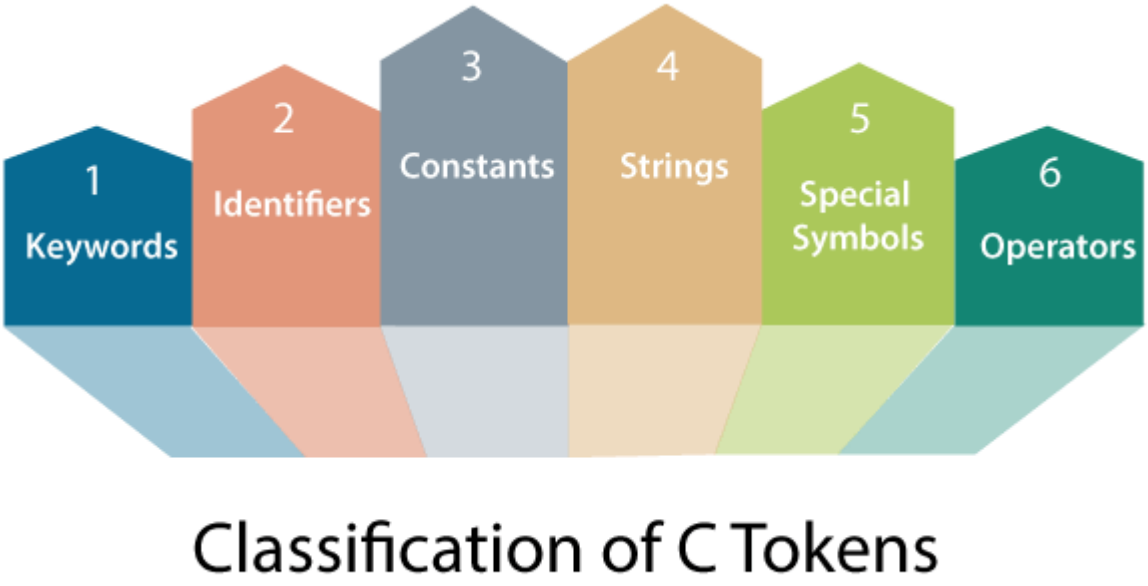
In the C programming language, the character set refers to a set of all the valid characters that we can use in the source program for forming words, expressions, and numbers. The source character set contains all the characters that we want to use for the source program text.

Tokens in C

Tokens in C is the most important element to be used in creating a program in C. We can define the token as the smallest individual element in C. For `example, we cannot create a sentence without using words; similarly, we cannot create a program in C without using tokens in C. Therefore, we can say that tokens in C is the building block or the basic component for creating a program in C language.

Classification of tokens in C

Tokens in C language can be divided into the following categories:



Classification of C Tokens

- Keywords in C
- Identifiers in C

Let's understand each token one by one.

Keywords in C

Keywords in C can be defined as the pre-defined or the reserved words having its own importance, and each keyword has its own functionality. Since keywords are the pre-defined words used by the compiler, they cannot be used as the variable names. If the keywords are used as the variable names, it means that we are assigning a different meaning to the keyword, which is not allowed. C language supports 32 keywords given below:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union

const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifiers in C

Identifiers in C are used for naming variables, functions, arrays, structures, etc. Identifiers in C are the user-defined words. It can be composed of uppercase letters, lowercase letters, underscore, or digits, but the starting letter should be either an underscore or an alphabet. Identifiers cannot be used as keywords. Rules for constructing identifiers in C are given below:

- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Delimiter

A delimiter is one or more characters that separate text strings. Common delimiters are commas (,), semicolon (;), quotes (", '), braces ({}), pipes (|), or slashes (/ \ ). When a program stores sequential or tabular data, it delimits each item of data with a predefined character.

C - Variables

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. Based on the basic types explained in the previous chapter, there will be the following basic variable types −

Sr.No.	Type & Description
1	char Typically a single octet(one byte). It is an integer type.

2	int The most natural size of integer for the machine.
3	float A single-precision floating point value.
4	double A double-precision floating point value.
5	void Represents the absence of type.

C programming language also allows to define various other types of variables, which we will cover in subsequent chapters like Enumeration, Pointer, Array, Structure, Union, etc. For this chapter, let us study only basic variable types.

Variable Definition in C

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows −

```
type variable_list;
```

Here, type must be a valid C data type including char, w_char, int, float, double, bool, or any user-defined object; and variable_list may consist of one or more identifier names separated by commas. Some valid declarations are shown here −

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

The line int i, j, k; declares and defines the variables i, j, and k; which instruct the compiler to create variables named i, j and k of type int.

Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows −

```
type variable_name = value;
```

For example:

```
int age;
float weight;
char gender;
```

In these examples, age, weight and gender are variables which are declared as integer data type, floating data type and character data type respectively.

Constants in C

A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming" etc.
There are different types of constants in C programming.
List of Constants in C

Constant	Example
Decimal Constant	10, 20, 450 etc.
Real or Floating-point Constant	10.3, 20.2, 450.6 etc.
Octal Constant	021, 033, 046 etc.
Hexadecimal Constant	0x2a, 0x7b, 0xaa etc.
Character Constant	'a', 'b', 'x' etc.
String Constant	"c", "c program", "c in javatpoint" etc.

2 ways to define constant in C
There are two ways to define constant in C programming.

1. const keyword
2. #define preprocessor

1) C const keyword
The const keyword is used to define constant in C programming.
History of Java

```
1. const float PI=3.14;
```

Now, the value of PI variable can't be changed.

```
1. #include<stdio.h>
2. int main(){
3.     const float PI=3.14;
4.     printf("The value of PI is: %f",PI);
5.     return 0;
6. }
```

Output:
The value of PI is: 3.140000

If you try to change the the value of PI, it will render compile time error.

```
1. #include<stdio.h>
2. int main(){
3. const float PI=3.14;
4. PI=4.5;
5. printf("The value of PI is: %f",PI);
6.     return 0;
7. }
```

2) C #define preprocessor

The #define preprocessor is also used to define constant. We will learn about #define preprocessor directive later.

Data Types in C

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Let us briefly describe them one by one:

Following are the examples of some very common data types used in C:

- char(1 byte): The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- int(2/4 bytes): As the name suggests, an int variable is used to store an integer.
- float(4 bytes): It is used to store decimal numbers (numbers with floating point value) with single precision.
- double(8 bytes): It is used to store decimal numbers (numbers with floating point value) with double precision.

Different data types also have different ranges upto which they can store numbers. These ranges may vary from compiler to compiler. Below is a list of ranges along with the memory requirement and format specifiers on a 32 bit gcc compiler.

C Expressions

An expression is a formula in which operands are linked to each other by the use of operators to compute a value. An operand can be a function reference, a variable, an array element or a constant. Let's see an example:

1. a-b;

In the above expression, minus character (-) is an operator, and a, and b are the two operands.

Comments in C

A well-documented program is a good practice as a programmer. It makes a program more readable and error finding become easier. One important part of good documentation is Comments.

- In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program
- Comments are statements that are not executed by the compiler and interpreter.

In C there are two types of comments :

1. Single line comment
2. Multi-line comment

Single line Comment

Represented as // double forward slash

It is used to denote a single line comment. It applies comment to a single line only. It is referred to as C++-style comments as it is originally part of C++ programming.

for example:

// single line comment

Example: This example goes the same for C and C++ as the style of commenting remains same for both the language.


```c
// C program to illustrate
// use of multi-line comment
#include <stdio.h>
int main(void)
{

    // Single line Welcome user comment
    printf("Welcome to GeeksforGeeks");
    return 0;
}
```

Multi-line comment

Represented as /* any_text */ start with forward slash and asterisk (/*) and end with asterisk and forward slash (*/).

It is used to denote multi-line comment. It can apply comment to more than a single line. It is referred to as C-Style comment as it was introduced in C programming.

/*Comment starts
continues
continues
.
.
.
Comment ends*/

Example: This example goes the same for C as the style of commenting remains same for both the language.

```c
/* C program to illustrate
use of
multi-line comment */
#include <stdio.h>
int main(void)
{

    /* Multi-line Welcome user comment
    written to demonstrate comments
    in C/C++ */
    printf("Welcome to GeeksforGeeks");
    return 0;
}
```

Symbolic Constants in C – Symbolic Constant is a name that substitutes for a sequence of characters or a numeric constant, a character constant or a string constant.
When program is compiled each occurrence of a symbolic constant is replaced by its corresponding character sequence.
For example,
#define printf print
#define MAX 50
#define TRUE 1
#define FALSE 0
#define SIZE 15