

INTRODUCTIONS TO C

Overview of C Program

The programming language C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories to be used by the UNIX operating system. It was named C' because many of its features were derived from an earlier language called B'. Although C was designed for implementing system software, it was later on widely used for developing portable application software. C is one of the most popular programming languages. It is being used on several different software platforms. In a nutshell, there are a few computer architectures for which a C compiler does not exist. It is a good idea to learn C because few other programming languages such as C++ and Java are also based on C which means you will be able to learn them more easily in the future.

The most popular Operating system UNIX is written in C language. This language also has the features of low level languages and hence called as “System Programming Language”.

History of C

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.

- ▶ The root of all modern languages is ALGOL,(Introduce in 1960s).
- ▶ ALGOL uses a structure programming.
- ▶ ALGOL is popular in Europe
- ▶ In 1967, Martin Richards developed a language called BCPL(Basic Combined Programming Language)
- ▶ Primarily BCPL is developed for system software
- ▶ In 1970, Ken Thompson created a new language called B
- ▶ B is created for UNIX os at Bell Laboratories.
- ▶ Both BCPL and B were “typeless” languages.
- ▶ C was evolved from ALGOL, BCPL and B.
- ▶ C was developed by Dennis Ritchie at the Bell Laboratories in 1972.
- ▶ Added new features and concepts like “data types”.
- ▶ It was developed along with the UNIX operating system.

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

- ▶ In 1983 American National Standards Institute(ANSI) appointed a technical committee to define a standard for c. The committee approved a version of C in December 1089 which is now known as ASCI C
- ▶ In 1990 international Standards Organization(ISO) has approved C and this version of C is referred to as C89.

Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
K & R C	1978	Kernighan & Dennis Ritchie
ANSI C	1989	ANSI Committee
ANSI/ISO C	1990	ISO Committee
C99	1999	Standardization Committee

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

Uses of C

- C is a very simple language that is widely used by software professionals around the globe. The uses of C language can be summarized as follows:
- C language is primarily used for system programming. The portability, efficiency, the ability to access specific hardware addresses, and low runtime demand on system resources make it a good choice for implementing operating systems and embedded system applications.
- C has been so widely accepted by professionals that compilers, libraries, and interpreters of other programming languages are often written in C.
- For portability and convenience reasons, C is sometimes used as an intermediate language for implementation of other languages. Examples of compilers which use C this way are Bit C, Gambit, the Glasgow Haskell Compiler, Squeak, and Vala.
- Basically, C was designed as a programming language and was not meant to be used as a compiler target language. Therefore, although C can be used as an intermediate language it is not an ideal option.
- C is widely used to implement end-user applications.

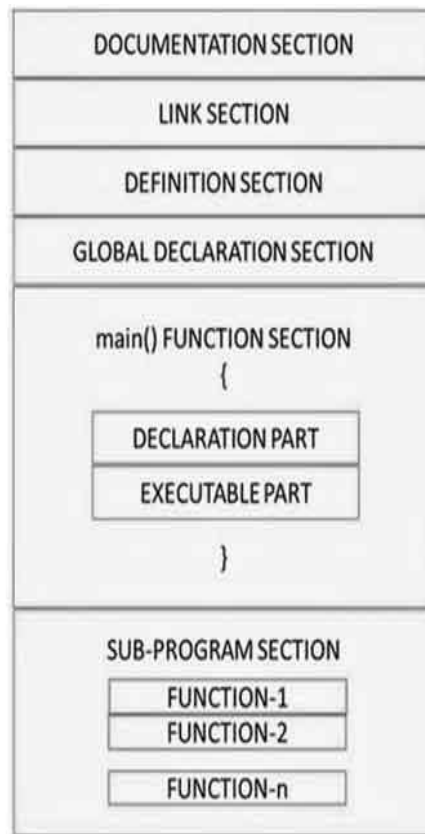
Importance of c

- C is a robust language whose rich set of built-in functions and operators can be used to write complex programs. The C compiler combines the features of assembly languages and high-level languages, which makes it best suited for writing system software as well as business packages. Some basic characteristics of C language that define the language and have led to its popularity as a programming language are listed below.
- C is a high-level programming language, which enables the programmer to concentrate on the problem at hand and not worry about the machine code on which the program would be run.
- Small size C has only 32 keywords. This makes it relatively easy to learn as compared to other languages.
- C makes extensive use of function calls.
- C is well suited for structured programming. In this programming approach, C enables users to think of a problem in terms of functions/modules where the collection of all the modules makes up a complete program. This feature facilitates ease in program debugging, testing, and maintenance.
- Unlike PASCAL it supports loose typing (as a character can be treated as an integer and vice versa).
- Structured language as the code can be organized as a collection of one or more functions.

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

Basic Structure of a C programming



1. Documentation Section:

The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

The comment lines are simply ignored by the compiler, that means they are not executed. In C, there are two types of comments.

1. **Single Line Comments:** Single line comment begins with // symbol.
2. **Multiple Lines Comments:** Multiple lines comment begins with /* symbol and ends with */.

In a C program, the comment lines are optional.

Example: /* program to find the area of a circle */

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

2. Link section: The link section provides instructions to the compiler to link functions from the system library such as using the `#include` statement to include header file into our program.

Example: 1. `#include<stdio.h>` which contains `printf()`, `scanf()`, library function.

Example: 2. `#include<conio.h>` which contains `clrscr()`;

Example: 3. `#include<math.h>` which contains `sqrt()`, `sin()`, `cos()` etc...

3. Definition section: The definition section defines all symbolic constants such using the `#define` statement. Example: `#define PI 3.14`

4. Global declaration section: There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the **user-defined functions.**

5. main() section: Every C program must have one main function section. This section contains two parts; declaration part and executable part

1. Declaration part: The declaration part declares all the variables used in the executable part.

2. Executable part: There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.

6. Subprogram section: If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function. User defined functions are generally placed immediately after the main () function, although they may appear in any order.

All section, except the main () function section may be absent when they are not required.

Example:

PROGRAM FOR AREA OF A CIRCLE:

`/* Program name: Area of a circle.`

`Input: radius. Output: Area */`

`#include<stdio.h>`

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

```
#include<conio.h>

#define PI 3.14

void main()

{

Float r, area, PI=3.14;
clrscr();
printf("enter a radius\n");
scanf("%f",&r);
area=PI*r*r;
printf("Area=%f\n",area);
getch ();

}
```

Some basic syntax rule for C program

- C is a case sensitive language so all C instructions must be written in lower case letter.
- All C statement must end with a semicolon.
- Whitespace is used in C to describe blanks and tabs.
- Whitespace is required between keywords and identifiers.

Character Set: The character set is the fundamental raw-material for any language. Like natural languages, computer languages will also have well defined character-set, which is useful to build the programs.

- ▶ The C language consists of two character sets namely
- ▶ Source character set
- ▶ Execution character set.

Source character set: This type of character set includes three types of characters namely alphabets, Decimals and special symbols.

- i. Alphabets: A to Z, a to z and Underscore(_)
- ii. Decimal digits: 0 to 9
- iii. Special symbols: + - * / ^ % = & ! () { } [] " ' etc.

Execution character set: This set of characters are also called as non-graphic characters because these are invisible and cannot be printed or displayed directly.

Ex: \n, \t, \h, \v etc...

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

C TOKENS

Tokens are the basic building blocks in C language. You may think of a token as the smallest individual unit in a C program. This means that a program is constructed using a combination of these tokens.

The following are the types of tokens:

- Keywords
- Identifiers
- Constant
- Strings
- Operators

Keywords

Keywords are predefined, reserved words in C and each of which is associated with specific features. These words help us to use the functionality of C language. They have special meaning to the compilers. There are total 32 keywords in C.

++Auto	double	Int	struct
Break	else	Long	switch
Case	enum	Register	typedef
Const	extern	Return	union
Char	float	Short	unsigned
Continue	for	Signed	Volatile
Default	goto	Sizeof	Void
Do	if	Static	While

Identifiers

Identifiers, as the name suggests, help us to identify data and other objects in the program. Identifiers are basically the names given to program elements such as variables, arrays, and functions. Identifiers may consist of sequence of letters, numerals, or underscores.

Rules for Forming identifier Names Some rules have to be followed while forming identifier names. They are as follows:

- Identifiers cannot include any special characters or punctuation marks (like #, \$, ^, ?, .., etc.) except the underscore_.
- There cannot be two successive underscores.

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

- Keywords cannot be used as identifiers.
- The case of alphabetic characters that form the identifier name is significant.

For example, 'FIRST' is different from 'first' and 'First'.

Examples of valid identifiers include: roll_number, marks, name, emp_number, basic_pay, HRA, DA, dept_code, DeptCode, RollNo, EMP_NO

Examples of invalid identifiers include: 23 student, %marks, @name, #emp_number, basic.pay, -HRA, (DA), &dept_code, aut

CONSTANTS

Constants are identifiers whose values do not change. While values of variables can be changed at any time, values of constants can never be changed. Constants are used to define fixed values like mathematical constant π or the charge on an electron so that their value does not get changed in the program even by mistake. A constant is an explicit data value specified by the programmer. The value of the constant is known to the compiler at the compile time. C allows the programmer to specify constants of integer type, floating point type, character type, and string type.

Following are the various types of constants:

- Integer constants
- Character constants
- String constants
- Real Constants

Literals: The values assigned to each constant variables are referred to as the *literals*.

A constant is defined in two ways:

1. Using **#define** preprocessor directive
2. Using a **const** keyword

1. Using #define preprocessor directive:

By using the **#define** pre-processor directive which doesn't use memory for storage and without putting a semicolon character at the end of that statement

Syntax: #define identifierName value

MODULE-02

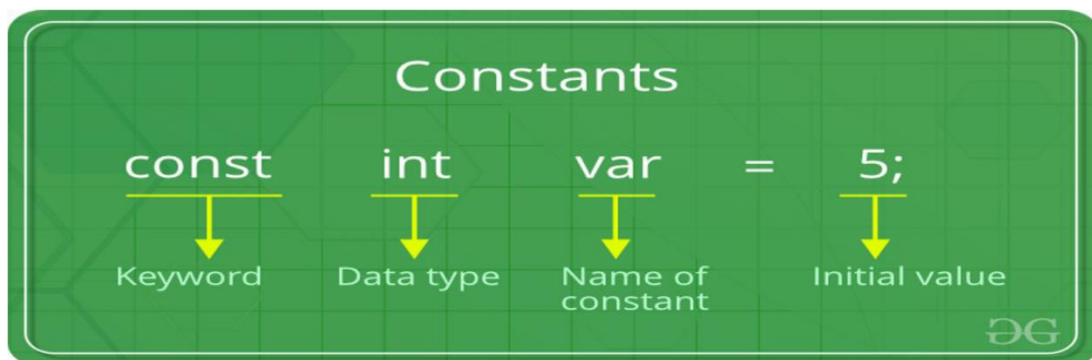
PROBLEM SOLVING USING C PROGRAMMING (BCA103)

- **identifierName:** It is the name given to constant.
- **value:** This refers to any value assigned to identifierName.

Example:

```
#define a 10  
#define floatVal 4.5  
#define charVal 'G'
```

2. using a const keyword: By using the **const** keyword in a variable declaration which will reserve a storage memory
Syntax: `const data_type variable_name = value;`



Declaring Constants

To declare a constant, precede the normal variable declaration with **const** keyword and assign it a value.

For example, `const float pi = 3.14;`

However, another way to designate a constant is to use the pre-processor command **define**. Like other preprocessor commands, **define** is preceded with a **#** symbol. Although **#define** statements can be placed anywhere in a C program, it is always recommended that these statements be placed at the beginning of the program to make them easy to find and modify at a later stage. Look at the example given below which defines the value of **pi** using **define**.
`#define pi 3.14159`

Variables:

A **variable** is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.

It is a way to represent memory location through symbol so that it can be easily identified.

Syntax: `data_type variable_Name;`

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

Example: `int a;`

`float b;` Here, a, b, c are variables. The int, float, char are the data types.

`char c;`

We can also provide values while declaring the variables as given below:

1. `int a=10,b=20;` //declaring 2 variable of integer type
2. `float f=20.8;`
3. `char c='A';`

Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.

Valid variable names:

1. `int a;`
2. `int _ab;`
3. `int a30;`

Invalid variable names:

1. `int 2;`
2. `int a b;`
3. `int long;`

Operators

An operator is a symbol that specifies the mathematical, logical, or relational operation to be performed. C language supports different types of operators, which can be used with variables and constants to form expressions. These operators can be categorized into the following major groups:

- Arithmetic operators
- Relational operators
- Logical operators
- Unary operators
- Conditional operators
- Bitwise operators
- Assignment operators
- Comma operator
- sizeof operator

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

Strings

A string is a sequence of characters and can contain letters, numbers, symbols and even spaces. It must be enclosed in quotation marks for it to be recognized as a string.

For example, the word “liquid” and the phrase “What is liquid? It's a template language.” are both strings.

Data Types:

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.

Data types are classified into three types:

1. Primitive data types
2. Derived data types
3. User-defined data types

There are five primary fundamental data types

1. int for integer data
2. char for character data
3. float for floating point numbers
4. double for double precision floating point numbers
5. void

Array, functions, pointers, structures are derived data types.

Basic Data Type:

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 int

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

(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 “longint” and “long longint” for which the limits are very high.

Example: `inta,b,c;`

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.
- char (1 byte) can store values from -127 to +128
- For example: `char var='A';`

‘A’ can be stored using char datatype. You can’t store more than one character using char data type.

3. Floating Point Data type:

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

1. float
2. double

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.
- Example: `float num;`
num is a variable of type float it can store 6 precision value after the decimal.

4. 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 $1E+37$.
sizeof() function is used to find the memory space allocated for each C data types.

MODULE-02

PROBLEM SOLVING USING C PROGRAMMING (BCA103)

5. void:

➤ Void is an empty data type that has no value.

➤ This can be used in functions and pointers.

For example, if a function is not returning anything, its return type should be void .

Example:

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
int a; char b; float c; double d;
```

```
printf("Storage size for int data type:%d \n",sizeof(a));
```

```
printf("Storage size for char data type:%d \n",sizeof(b));
```

```
printf("Storage size for float data type:%d \n",sizeof(c));
```

```
printf("Storage size for double data type:%d\n",sizeof(d));
```

```
}
```