

|  |  |
| --- | --- |
| C was developed by Dennis Ritchie between 1969 and 1973 at AT&T Bell Labs. | C++ was developed by Bjarne Stroustrup in 1979 with C++'s predecessor "C with Classes". |
| When compared to C++, C is a subset of C++ | C++ is a superset of C. |
| C supports procedural programming paradigm for code development. | C++ supports both procedural and object oriented programming paradigms. |
| C does not support object oriented programming; therefore it has no support for polymorphism, encapsulation, and inheritance. | Being an object oriented programming language C++ supports polymorphism, encapsulation, and inheritance. |
| In C (because it is a procedural programming language), data and functions are separate and free entities. | In C++ (when it is used as object oriented programming language), data and functions are encapsulated together in form of an object. |
| In C, data are free entities and can be manipulated by outside code. This is because C does not support information hiding. | In C++, Encapsulation hides the data to ensure that data structures and operators are used as intended. |
| procedural prog. function driven language. | C++, being an oop, object driven language. |
| do not support function / operator overloading. | supports both function / operator overloading. |
| do not allow functions to define in structures. | C++, functions can be used inside a structure. |
| C does not have namespace feature. | C++ uses NAMESPACE which avoid name collisions.  A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries. All identifiers at namespace scope are visible to one another without qualification. Identifiers outside the namespace can access the members by using the fully qualified name for each identifier. |
| C functions for input/output. scanf and printf. | C++ uses objects for input output. Cin, cout. |
| C does not support reference variables. | C++ supports reference variables. |
| C no support for virtual and friend functions. | C++ supports virtual and friend functions. |
| provides malloc() ,calloc() functions fo dynamic mem allocation, and free() for de-allocation. | provides new operator for memory allocation and delete operator for memory de-allocation. |
| C does not provide direct support for error handling (also called exception handling) | C++ provides support for exception handling. Exceptions are used for "hard" errors that make the code incorrect. |

# **Characteristic of A Good Programming Language**

There are various factors, why the programmers prefer one language over the another. And some of very good characteristics of a good programming language are,

**1) Clarity, Simplicity And Unity:** A Programming language provides both a conceptual framework for Algorithm planning and means of expressing them. It should provide a clear, simple and unified set of concepts that can be used as primitives in developing algorithms.

It should have

• It has minimum number of different concepts  
•- with Rules for their combina-tion being  
•-simple and regular.

This attribute is called **conceptual integrity.**

**2) Orthogonality:** It is one of the most important feature of PL orthogonality is the property that means **" Changing A does not change B".**

If I take Real world example of an orthogonal system Would be a radio, where changing the station does not change the volume and vice versa.

When the features of a language are orthogonal, language is **easier to learn** and **programs are easier to write** because only few exceptions and special cases to be remembered.

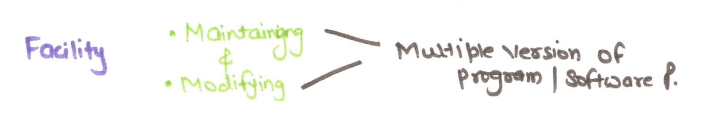
**3) Support for Abstraction:-** There is always found that a substantial gap remaining between the abstract data structure and operations that characterize the solution to a problem and their particular data structure and operations built into a language.

**4) Programming Environment:** An appropriate programming environment adds an extra utility and make language to be implemented easily like

**The availability of**- Reliable- Efficient - Well documentation

**Speeding up creation and testing by-special Editors- testing packages**

**Facility- Maintaining and Modifying-** **Multi Version of program software product.**



**5) Ease of program verification:- Reusability:**

The reusability of program written in a language is always a central concern. A program is checked by various testing technique like

**Formal verification method Desk checking Input output test checking.**

We verify the program by many more techniques. A language that makes program verification difficult maybe far more troublesome to use. **Simplicity of semantic and syntactic structure is a primary aspect that tends to simplify program verification.**

**6) portability of programs:** Programming language should be portable means it should be easy to transfer a program from which they are developed to the other computer.

A program whose definition is independent of features of a Particular machine forms can only support **Portability.** **Example:, FORTRAN, C, c++, Java.**