**What Makes a Good Programming Language?**

**M**any new hundreds of languages are being designed and developed. Now a days, which are used for a period of Time and then allowed to fall into disuse.

There are many reasons behind success or failure of a language, sometimes the reason may be External to the Language itself. **For Example:** use of *COBOL or Ada* in US was forced in certain areas of Programming development industries by Government mandate. So extensive Usage of A particular language by a large amount of people leads to Greater Success and most of the developments are done in the same Language preferably. Likewise there is reason for success of *FORTRAN*,cause most of the computer manufacturers strongly supported this language. Many more languages are there which are successful due to many External reasons. Though many of them are failures in history, and to indicate their failure level most of us don’t know the name of languages which have failed to make them self on the long list!!!

**Attributes of a Good Language**

Due to major importance of some of these external influences, it is the programmer who ultimately decides which language will survive and which will die!!!

There are many reasons for a Language to Survive in this tough competitive world. Some of the important reasons are enlisted here:

**1. Clarity Simplicity and Unity**

A programming Language provides a medium to conceptual thinking of new algorithms and also a medium to Execute your thought Process into real Coding Statements. For algorithms to be implemented on a Language it’s a basic need is that the language is quite *clear, simple and Unified* in structure. Such that the Primitives of language can be utilized to develop algorithms. It is desirable to have a minimum number of different concepts, so that combining multiple concepts won’t be that complex in nature. It should be simple and regular as possible. This attribute of a language is known as conceptual Integrity.

The main concern of a language now a day is its readability. The syntax of language affects the ease with which programs are written, tested and later used for knowledge or research purpose. A complex syntax language may be easy to write program in, but it proves to be difficult to read and debug the code for later sessions. **For example** APL programs are so complicated that even the own developers find it difficult to understand after 1-2 months. The language should be simple enough to understand or point out errors.

**2. Orthogonality**

The term orthogonality refers to the attribute of being able to combine various features of a language in all possible combinations, with every combination being meaningful. **For example,** Suppose a language provides with a expression lets say an arithmetical calculation operator. Taking Another Expression facilitated by the language like conditional Statement, which has 2 outputs either 0 or 1 (in some cases TRUE or FALSE ). Now the language should support combination of these two expressions. So that new statement can be formed, and this orthogonality helps to develop many new algorithms.

**3. Naturalness for the application**

A language needs a Syntax that, when applied properly, allows the program Structure to reflect the logical structure what a programmer wanted it to. *Arithmetic Algorithms ,concurrent algorithms , logic Algorithms* and other type of statement have differing natural structures , that can be represented by the Program in that language. The language should provide appropriate *data structures, operations, control structures and a natural syntax* for the problem to be solved.

**For Example:**

Consider a real life condition of plates being placed above each plate, this structure is known as Stack. This Stack can be implemented into programming world also. This is used as a data Structure in most of the Languages.

**4. Support for Abstraction**

Many times languages fail to implement many real life problems into Programs. There is always a gap between abstract data structures and operations. Even most natural Programming language fails to bridge the gap. **For Example:** Consider a situation where a scheduling is to be done for college *student for attending a lecture in a class section, teacher*. Suppose the requirement is to assign a student a section lecture and teacher to attend, which are common task for natural application, but are not Provided by C.

The need of point is to design an appropriate abstraction for the problems solution and then implementing these abstraction using most primitive features of a language. Ideally, the language should provide the data structures, data types and operations to maintain such abstractions .C++ is one of the most used language, that provide such facilities.

**5. Ease of Program Verification**

The reliability of a programming Language written in a language is always a central Concern. There are many techniques which can be used to keep track of correct functionality of a language. Sometimes testing the Program with random values of the inputs and obtaining corresponding outputs. Program verification should be provided by languages to check and minimize the errors.

**6. Programming Environment**

The environment also play a vital role is success of a Language. The environment which is technically weak may get a bad response of Programmer, rather than a language that has less facility than the former but its environment is technically good. Some of the good featured of an environment are, Special editors and testing packages tailored to the language may greatly speed up the creation and testing of Programs.

**7. Portability of Programs**

The important criterion for many programming projects is the Transportability of the resulting program from one computer to other systems. A language which is widely available and does not support different features on different computer System, which may have different hardware, is considered a good language. **For Example** *C , C++* and most of the language now days are Portable in nature.

**8. Cost of Use**

The most tricky point that always matter a lot in any system that uses resources. It’s a major element to decide the Evaluation of any programming language, but cost means many different things :

**(a). Cost of Program Execution**

Program Execution costs total amount which has been used to implement the program. The research work on design, optimizing compilers, data allocation registers etc. These are the basic thing which comes under the cost of Program Execution.

**(b). Cost of Program Translation**

The next concern is program compilation. The program is compiled many times than it is being executed. In such case, it is important to have a speed and efficient compiler to handle this Job.

**(c). Cost of Program Creation, Testing and Use**

Another aspect of Cost management. This includes the cost which a programmer charges for his work of creating project with the specified features, the cost involving the Testing issues.

**(d). Cost Of Program Maintenance**

After a program is being installed in a System, then after certain intervals it needs maintenance to run smoothly. The maintenance includes the rectification of Error propagated in real time, the updating of program as need of time.

These are many of the Aspects and Features of a Good Programming Language .