**Effective C++**

**Chapter 1: - Accustoming yourself to C++.**

* Author told not to view C++ as a one language. C++ consists of 4 parts. Try to follow the rules and regulations of the part you are coding in.
  + **C:** - Good old C with pointers preprocessor and built-in types
  + **Object-Oriented C++: -** Basically C with classes which contains Classes, Constructor, Destructors, Virtual Functions etc. It follows all rules of object- oriented programming encapsulation, inheritance, polymorphism.
  + **Template C++: -** The generic programming part of C++.
  + **The STL: -** The standard template library of C++ which contains containers to make programmers life easier.
* Prefer const, enum, inline to #defines.
  + #defines are bad because they are processed by the preprocessor because of which compiler don’t have any entry for them in symbol table which makes it difficult to debug as a counterpart if we use const compiler will be aware of it and will be able to generate the proper error messages.
  + Enum can be used as a counterpart of #defines while we want to declare the array with some constant int value. It will stop the compiler to allocate space for it now sure how probably must look more into it.
  + Using #defines to create small functions and hope they will get inline is a bad idea because in most of cases it will create bugs hard to debug better use **inline** to archive the same.
* Use the const wherever possible I mean it. It makes the things safer and faster.
* Make sure objects are initialized before they are used otherwise the results can be undefined specially the built-in types.