# Compiler

Aim: Implement a fast general purpose language with constructs akin to C++ yet eases users into programming with C# and java like constructs.

Designed with Console handling in mind multiplatform using assemblers from different platforms(unix, linux, mac osx, windows). Includes networking, database handling, Windows/Mac OSX/Linux/Unix GUI features such as windowing and what not. Includes Linked Lists, Arrays, Binary Trees and other templates and library features. Basic graphics and what not. Preprocessor.

## Implementation

Implementation needs to include the following:

* Variables and typing (objects and possibly pointers as well) as well as static, constant, global
* Functions with return types, void, multiple arguments, inlining
* Garbage collection with option to handle it manually through preprocessor or what not.
* Importing like java and using like C++ which enables things like System.out to be used as out(“blah”).
* Control statements such as if, ternary operator
* Looping statements such as while, do while, for, for each, and looping through an array and even lists such as for(Strings: string; blah; blah).
* Try, catch and throwing exceptions.
* asserting
* Basic algebra such as addition, subtraction, division, multiplication, brackets, and POWERS! Library to handle other features such as matrix, vectors, sets and what not
* Object Oriented so, Classes, public, private, property to allow for quick set and get, and inlining. Inheritance and interfaces
* Packaging into a library
* Including multiple language and even assembler into the code through object linking
* Lambda functions and blocks of code{ }

## Order of Implementation

1. Printing
2. Importing and Using
3. Variables and typing
4. Control Statements
5. Loops
6. Functions
7. Garbage Collection/Pointers
8. Try, catch, throw
9. Asserting
10. Basic Algebra
11. Object Oriented
12. Packaging
13. File I/O
14. Including Multiple Language and assembly into code or through linking
15. Lambda functions

## Future Goals

* Database handling
* Networking
* Generic Collections (Lists, Queues, Stacks, Vectors)
* Graphics (2D, 3D)

## Implementation Stage

1) Printing – Need Parsing of files using Expression Trees.

Parse continually, separated by commas.

Parse by reading one character at a time. If it is number keep reading until a delimiter is presented(space, comma, bracket, square bracket, braces) or if a letter is presented it is a string. If delimiter is encountered convert to number. And put into a purse node. Put delimiter into. Put these data into parse tree.