4100

Final Project Report

Jake Tully

Introduction:

I used The C language and knowledge of numerical methods to create this language. I created this language because I have a background in engineering physics and it is an application for engineers and physicists. Languages Like MATLAB and Root6 Data Analysis Framework are examples of languages I have been introduced to in the past and what got me interested in computer science. Also, Solvers like COMSOL are important tools when working on engineering projects. These languages and solvers where created to aid engineers and are much more complex than my language. I chose C because it is fast and a procedural language that is well suited for numerical methods (an old professor did all of his modeling and simulations in C). My language briefly outlines what can be done with a computer in the application of engineering problems by demonstrating a solver for ordinary differential equations, ordinary differential equation boundary value problem and partial differential equations. Although most Solvers like COMSOL use Finite element methods that I have not learned. This langue is built with C and therefor a compiled language. Most of the following will also default to what is done in C. This Language is Only a demonstration of what could be created and not an example of a full language.

Chapter 2: Programing language Syntax

Programing language syntax refers to the rules that create a correctly structured sequence of symbols that can then be used to form a correctly structured program. Meaning is then added through the semantics and grammar of a programing language. When making a Language like this the syntax of the language is that of C but like MATLAB type declarations could taken care of by the langue such that the programmer would not have to worry about types. This would remove the need for Contextual Keywords. C and MATLAB both allow for nested comments. C recognizes 32 possible keywords although my implementation only uses double, int, and for. It also uses operators. The dangling else is not a problem in C and there are no else clauses in my implementation. Because it is implemented in C it follows the basic C syntax Rules such as Case sensitivity, statements must end with a semicolon, and whitespace is required between keywords and identifiers.

Chapter 3: Names, Scope, and Binding

A name is an identifier that is used to refer to variables, constants, functions, types, Operators and so forth. The binding refers to the association of a name with an object. The scope refers to the lifetime and binding of a name to an object. My implementation has an array passed to the function all my functions are void inside the nmthds.h header but takes advantage of side effects to alter the array passes to it as y to result in the array containing the desired solution. This is a bad way of doing it. In MATLAB ODE45 is a solver for ordinary differential equations that takes in arguments and makes assignment to a variable. This makes all variables used in calculation hidden from the scope of the main program and the coder in MATLAB only has access to the results.

Chapter 4: Semantic Analysis

Semantic analysis is the job of making sure that the declarations and statements of a program are correct. Such that there meaning is clear and consistent with the way structures and types and supposed to be used. Symantec analysis is handled by the C language which is done with the gcc compiler.

Chapter 5: Target Machine Architecture

For a C program the code must be transferred from C code to assembly code then to a relocatable binary code finally to machine code. The gap if the target is machine code is.

Global variable → global static memory

Unbounded number of interchangeable local variable → fixed number of registers of various incompatible kinds, plus number of stack locations

Built in parameter passing and result returning → calling conventions defining where arguments and results are stored and which registers may be overwritten by the Calle

Statements → machine instructions

Conditional branches based on integers representing Boolean values → conditional branches based on condition codes

Chapter 6: Control Flow

Control flow is the order in which statements are executed and evaluated. It determines whether a language is considered imperative or declarative. C is an imperative language, i.e. it uses statements to change the programs state. More specifically C is procedural. Computer processors provide hardware support for procedural programming though a stack register and instructions for calling procedures and returning from them. In my language there are only for loops to affect the flow of the program. To do this a variable local to the loop is created and incremented allowing for the repetition of a block of code until a condition is reached.

Chapter 7: Type Systems

C is a weakly typed language and statically typed. This means that each variables type is determined at compile time. The only types used in my language would be int, double, and float.

Chapter 8: Composite Types

Composite types refer to the structure of a row or record it is a list of field names and their data types. C does not implement garbage collection and it is up to the programmer to insure there are no memory leaks. C has composite types but I did not use any structures.

Chapter 9: Subroutines and Control Abstraction

Subroutines are the main method to build control abstraction, which is the process a programmer uses to define new control constructs which specify constraints on statement ordering separately form an implementation of that ordering. In my language I use call by name in C for most parameters except the array I want modified which is called by reference.

Chapter 10:

My language does not support Object Orientation because there is no need for it only basic types are used and returned.