COMS W4115
Programming Languages and Translators
Project

Array Manipulation Language (AML)
White Paper

Ayaskant Pani
UNI : ap2074
(ayas_76@yahoo.com)

Introduction

One of the fundamental data structure in computer programming languages is array. The use of linear array is very intuitive representation of a list of similar type of data. Modern high-level computer languages come with a vast set of functionality with support for arrays. Unfortunately most of these languages don't have direct support for array's operations and often these have to be implemented via auxiliary functions either developed by the user or by some form of library functions. The goal of AML is to design a simple and intuitive language which can provide the programmers with a handy language to do lots of efficient array manipulation which can either be character based operations or mathematical ones. The advantage is the low learning curve for a programmer to develop programs in the new language which can still get the array related jobs done for him. The language implementation will be made in a scalable and portable manner. The target code will be very efficient for simple array related operations. Note that AML is a generic language and hence most of it's syntax and semantics will govern array related operations. The language interpreter can be used for programming either in shell-like-interpreter mode or in a passed-in source file.

Simple

AML is a simple and yet efficient language. The syntax and semantics of the various language constructs will be made as intuitive as possible. It will be compatible with operations of various successful market products. The language comes with a simple flow control mechanisms like conditional and loop constructs. The purpose of the language is to empower developers with a new language that will demand a very low learning curve and yet deliver an efficient alternative to commercially available huge language/packages like Metlab.

Robust

The language is self-restrictive in the sense that due to support of only arrays of a floating point / character types the chances of mingling of different data types will be automatically be restricted. Built in errors checks will provide users from making any syntactic/semantic error. However logic errors will be not be resolved by the language compiler.

Portable

AML is implemented in Java hence the interpreter itself is portable. The target code generated will be in JAVA also which makes the target programs suitable to run on any platform/architecture providing support for JAVA.

High Level

The language will have high-level language syntax and semantics hiding users from the machine architecture dependent issues which is often time exposed in a hideous manner in low level languages like assembly languages. Many high-level and sought-after features of arrays like sorting, shifting, iterator operations on each elements etc will be supported.

Data types

There are two simple data types in AML: floating point numbers and characters. The only complex form of data types supported will be an array of these simple data types. All numerals in the AML program will be internally represented as floating point numbers.

Scope

All variables are globally visible in AML. Variables are instantiated when they are first time encountered in the program. The life and visibility of variables are global from the point of definition. This simple scoping rule is to make the language implementation simpler. In future a more stricter block-controlled scoping rule might be implemented.

Error Handling

All the lexical and syntactic errors will be caught by the lexer and parser constructed by ANTLR from the lexer and parser grammars. The rules will be simple and intuitive ones – like adding a float to a char or passing less/more than expected number of arguments to some of operation.

Exception

The language will not support any exception handling mechanism. As a simple language like this is intended for small programs sophisticated exception schemes are an overkill.

Control Flow Structures

The language will come with a set of control flow structures for simple control flows that can be used to translate any array related algorithm to the program in AML with ease.

Example of Syntax

Here is a sample program in AML:

```
a = [1, 2, 3, 4, 5, 6];
```

```
ch_arr = [ ‘c’, ‘b’, ‘a’, ‘l’, ‘a’ ];

a = a + 10; // add number 10 the each element.
b = 5;      // assign value 5 to b
c = a.b;    // concatenate a to b

sort(c);    // sort elements in c
print (c);    // print out c

sort(ch_arr);
print (ch_arr); // print that array ch_arr.
```

Conclusion

True to it’s name AML will a language rich with array manipulation supports yet will be a simple language to understand and program. However care will be taken to make the implementation run as efficiently as possible with portability in mind..