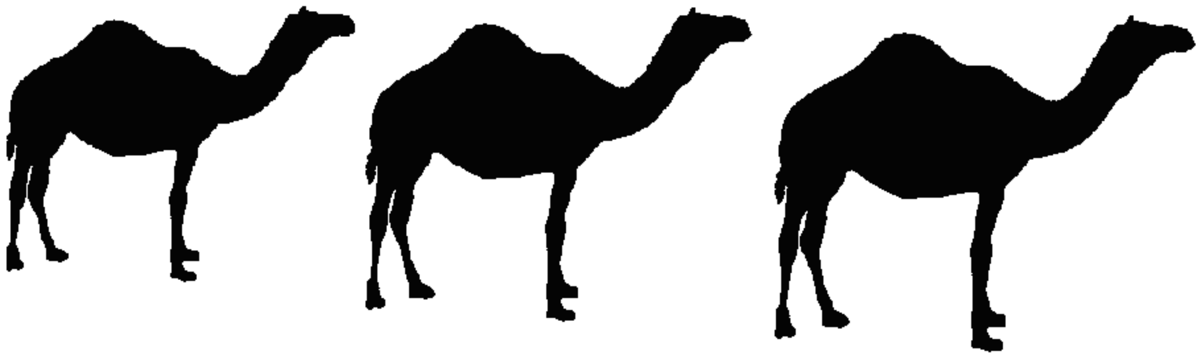
[K]AML



Array Manipulation Language

***K**aori Fukuoka* *kf2284@columbia.edu*
***A**nkush Goel* *ag2812@columbia.edu*
***M**aninder Singh* *ms3770@columbia.edu*
*Mayur **L**odha* *mdl2130@columbia.edu*

COMS 4115: Programming Languages and Translators, Fall 2008: Prof. Stephen Edwards.

Motivation

In Computer Science, most of the programming languages have concept of Arrays as one of the fundamental data structures which consists of ordered, integer indexed collection of objects. Array manipulations form an important as well as error prone part of many algorithms. For functionalities which have support for arrays, it becomes important and necessary to have a language which is handy and allows one to write more effective code. So far, no language has direct support for array manipulations as they have for integer or other primitive datatypes. We need a language using which array manipulating operations like adding all elements, returning a subarray satisfying certain conditions, array concatenation, copying arrays, set operations on arrays can be performed effectively.

Description

[K]AML is an expressive and concise Array Manipulation language which features rich set of operations on Arrays. Unlike other structured programming languages, **[K]AML** treats the array as a primitive datatype and array manipulations is done using high level operators. Thus, it would not require a beginner to think of arrays as a collection of data but as a single data-type in itself. Using **[K]AML**, it becomes possible to express a computable function using just an expression in a single line of code. This reduces the potential number of loops and allows for concise and compact programs.

Programming Features

1. Declaration

```
x = 0, y = 3;              // Declaration & initialization of numbers

[ ] a = {1,2,3,4,5};       // Declaration & initialization of an array
```

2. Manipulating Operations

```
[ ] r = a;                 // Copying all elements of array 'a' to array 'r'

r = a[x..y];               // Copying elements from index x=0 to y=3 of array ‘a’ to array 'r'

a[4] <- {1,2,3};           // Insert elements 1,2,3 at index 4 of array ‘a’

a[2] -> 3;                 // Remove next 3 element of array ‘a’ starting from index 2

[ ]c = a + r;              // Concatenate array ‘a’ & ‘r’ into array ‘c’.

c = + b[1,2,3,6];          // Store sum of elements at index 1,2,3,6 of array 'b' into number ‘c’

[ ] c = %+(a,b);           // Performs union of arrays ‘a’ & ‘b’ and stores it in array ‘c’

[ ]c = -a;                 // Copies all elements of array ‘a’ in array ‘c’ but in reverse order
```

```
c = ?(a,<10);                  // Copies elements into array ‘c’ from array ‘a’ if element is < 10

#a                             // Represents the total number of elements in array ‘a’

show("Enter 2 No:");           // Print statement

get(a,b);                      // Stores the 2 values in ‘a’ & ‘b’
```

Sample Program

Consider a 2 dimensional array ‘a’ which stores the marks secured by various students in different subjects. The rows represent the students and the columns represent the subjects.
The following program shows various operations that can be performed on this array.

```
for (j=0..#a[ ])
{
        show(“Maximum marks of any student in this subject is:” >>a[ ][j]);
        show(“Minimum marks of any student in this subject is:” <<a[ ][j]);
        show(“Average marks of the class in this subject is” < >a[ ][j]);

        [ ]c1=?(a[ ][j],<40);
        [ ]c2=?(a[ ][j],>=60 && <80 );
        [ ]c3=?(a[ ][j],>80 && <90);
        [ ]c4=?(a[ ][j],=100 );

        show(“Number of students getting F in this course are” #c1);
        show(“Number of students getting C in this course are” #c2);
        show(“Number of students getting B+ in this course are” #c3);
        show(“Number of students getting A+ in this course are” #c4);
}
```