CSE450
Translation of Programming Languages
Lecture 14: Arrays!

MAN, YOU'RE BEING INCONSISTENT WITH YOUR ARRAY INDICES. SOME ARE FROM ONE, SOME FROM ZERO.

DIFFERENT TASKS CALL FOR DIFFERENT CONVENTIONS. TO QUOTE STANFORD ALGORITHMS EXPERT DONALD KNUTH, "WHO ARE YOU? HOW DID YOU GET IN MY HOUSE?"

WAIT, WHAT?

WELL, THAT'S WHAT HE SAID WHEN I ASKED HIM ABOUT IT.
Arrays in Tubular

Arrays in Tubular have the following form:

```plaintext
array(type) var_name;
```

This should create a new variable that is an array of the specified type. For example:

```plaintext
array(char) my_string;
```

would create an array of entries, each of type `char`. This effectively works as a `string`. 
Arrays Types

Tubular effectively has four types that you need to worry about:

val, char, array(val), array(char)

Additionally, the new type "string" should behave identically to "array(char)".

You are not required to implement multi-level arrays, such as "array(array(char))".
Using Arrays

Array use in Tubular is similar to that of C/C++. If `[ expression `]` follows an identifier, the expression inside the brackets is evaluated and is used to index into the array specified by the identifier.

The basic semantic checking that must occur is:
- The expression must evaluate to an `val`
- The identifier must be an array type
- The return type is the type inside of the array
Basic Array Examples

The basic use of arrays is straight-forward:

Accessing values:

```java
val cur_val = some_array[7];
```

Assignment:

```java
alphabet_string[4] = 'E';
```

To achieve the full functionality of strings, however, we must use a couple of helper methods.
More Tubular Additions

There are two additional commands that you need to work with arrays.

`array.size()`: returns an int indicating the current size of the array.

`array.resize(new_size)`: sets the array size to be new_size. Existing entries remain unchanged. Contents of new entries are undefined until set.

And one additional literal...

Quotes ("'): Any string contained between the quotes should be taken as a literal string (that is, of type "array(char)").
After compilation and execution, what is the result?

```c
array(char) x;
x.resize(3);
x[0] = 'a';
x[1.3] = 'b';
x[x.size() - 1] = 'c';
print(x);
x.resize(2);
print(x);
```

abc
ab
Error
Some other output
Shame on you for trick questions
# Tube Intermediate Code Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ar_get_idx</code></td>
<td>Find value at array index <code>num</code>, and put into <code>result</code>.</td>
</tr>
<tr>
<td><code>ar_set_idx</code></td>
<td>In array, set value at index <code>num1</code> to value <code>num2</code>.</td>
</tr>
<tr>
<td><code>ar_get_size</code></td>
<td>Calculate size of array and put into <code>result</code>.</td>
</tr>
<tr>
<td><code>ar_set_size</code></td>
<td>Resize array to have <code>num</code> entries.</td>
</tr>
<tr>
<td><code>ar_copy</code></td>
<td>Duplicate all values associated with array1 into array2.</td>
</tr>
</tbody>
</table>
Changes to the Symbol Table

- Previously nodes in the Abstract Syntax Tree needed links to their children, and sometimes to an entry in the symbol table (for variables).
- The indexing operator ([ ]) needs to keep track of:
  - The array its accessing
  - The index in the brackets
- The index node is used for two distinct purposes, assignment and reading.
The new errors for Project 5

ERROR(line #): cannot index into a non-array type
ERROR(line #): array indices must be of type val
ERROR(line #): array methods cannot be run on type ‘type’
   size() and resize() are the two legal array methods; the identifier before them must
   represent an array variable.
ERROR(line #): array size() method does not take any arguments.
ERROR(line #): array resize() method takes exactly one (val) argument.
ERROR(line #): array resize() method argument must be of type val.

The only expressions the type array(val) and array(char) are legal for is simple
assignment "=". All others should raise "ERROR(line #): cannot use type 'type' in
mathematical expressions".
Project steps:

We recommend that doing the project as follows:

1) Refactor your symbol table to handle array(val) and array(char)
2) Make your lexer recognize literal strings.
3) Catch literal string tokens in your “expression” rule.
4) Make sure expression does proper type-checking with arrays.
5) Setup the print command to work with arrays (allows you to test your code more easily!)
Project steps (cont’d):

6) Allow declarations of array variables (immediately translate string to array(char) )

7) Allow array variables to be used in expressions

8) Implement the size and resize methods on arrays. The size method should be an expression, but resize has no return type.

9) Allow array indexing to be used in expressions (initially on the RHS of assignments)

10) Allow array indices to be assigned to; this requires refactoring tableEntry.

[ 11) Extra Credit ]