Brian Cullinan

Project 2: Expression grammar and scanner

9/10/08

My approach to designing this language was simple. I wanted a language that is powerful, yet very very short. I don’t want to type a lot. Function names will inherently be very short, and all the syntax is short. The scanner class reads in and separated everything into tokens. Initially I thought about writing a spin-off of LOLCODE. But I decided it was already done too much.

<?php

// language spec

// The objective of my language is to be as shorthanded as possible.

Program ::= <? Command ?>

Command ::= V-Name = Expression

| Identifier ( Expression )

Expression ::= Integer-Literal

| V-Name

| Operator Expression

| ( Expression )

V-Name ::= Identifier

Declaration ::= v Identifier Type

| c Identifier Type

Type ::= Identifier

Operator ::= + | - | \* | / | ^

Identifier ::= Letter | Identifier Letter | Identifier Digit

Integer-Literal ::= Digit | Integer-Literal Digit

Comment ::= # Letter eol

?>

<?php

require\_once 'reader.php';

require\_once 'token.php';

class Scanner

{

var $file;

var $tokens = array();

function Scanner($file)

{

$this->file = $file;

$reader = new SourceReader($this->file);

$buffer = '';

while(true)

{

$int = $reader->read();

if($int == -1)

break;

if((chr($int) == ' ' || $int == 10 || $int == 13) && $buffer != '')

{

switch($buffer)

{

case 'v':

case 'c':

$this->tokens[] = new Token($this->file, $reader->line, $reader->col, T\_DECLARATION, $buffer);

break;

case '+':

case '-':

case '\*':

case '/':

case '^':

$this->tokens[] = new Token($this->file, $reader->line, $reader->col, T\_OPERATOR, $buffer);

break;

default:

$this->tokens[] = new Token($this->file, $reader->line, $reader->col, T\_IDENTIFIER, $buffer);

}

$buffer = '';

}

else

{

if($int == 10 || $int == 13)

continue;

$buffer .= chr($int);

}

}

}

}

?>

<?php

// the different types

define('T\_COMMAND', 0);

define('T\_EXPRESSION', 1);

define('T\_V-NAME', 2);

define('T\_DECLARATION', 3);

define('T\_TYPE', 4);

define('T\_OPERATOR', 5);

define('T\_IDENTIFIER', 6);

define('T\_LITERAL', 7);

define('T\_COMMENT', 8);

class Token

{

var $file;

var $line;

var $col;

var $type;

var $content;

function Token($file, $line, $col, $type, $content)

{

$this->file = $file;

$this->line = $line;

$this->col = $col;

$this->type = $type;

$this->content = $content;

}

}

?>