

Project - Part 3 - Interpreter

For this part of the project you are to write a symbol table generator and interpreter for the project statement programming language.

Specification:

1. Design and implement a “symbol” data structure. Your data structure will need to hold information about 3 types of symbols:
 - a. Operators
 - b. Identifiers
 - c. Numeric literals

A base symbol class and inherited classes for each type of symbol are recommended for those coding in C++; a struct containing a union is recommended for those coding in C.

2. Design and implement a symbol table data structure.
3. Design and implement functions to allow your parser to interact with your data structures. Minimally you will need functions to:
 - a. Create a symbol
 - b. Insert a symbol in the symbol table
 - c. Find a symbol in the symbol table
 - d. Print the symbols in the symbol table

Functions a, b, and c will be called by your parser at appropriate times; function c will be called by your main program after parsing is complete.

4. Design and implement functions to build statement queues. Since statements can be nested inside of other statements, you will need to create a stack of queues. Each queue should be able to hold an unlimited number of symbol pointers. As a statement is being parsed, entries pointing to the recognized symbols should be added to the top queue.
5. Design and implement a function to interpret a statement using the 2 stack method discussed in class. This function will be called by the parser each time it reaches the end of a statement.
6. All of the data structures and functions described here should be part of an interpreter module of your overall project.

Input: A source code file. The file name should be accepted as a command line argument.

Output: A listing of the original source code with lexical and syntactical error messages. If

there are no errors the listing should be followed by a list of the identifiers used in the source program. This listing should include each symbol's "name", type (int or double), and final value (to 5 decimal positions if it is a double).

A debugging file (filename + .dbg) containing a list of terminal and non-terminal symbols encountered while parsing the program (and other useful debugging information). You may wish to print the list of symbols to the debugging file as well as to the standard output stream.

Date Due: 10 December 2008; 11:59 pm. No late submissions will be accepted.

To turn in: Tared and zipped directory (*lastnameP3.tgz*) containing source files (headers and implementations), a makefile and a README file. Your makefile should create an executable called "proj3". Your README file should describe your project; in particular, it should describe the contents of each of your source files and any aspects of the project that you have not completed. Submit your .tgz file by copying or moving it to ~tiawatts/cs460drop

```
.....:
p3.in1
.....:
a = 4;
pi = 3.14159;
```

```
.....:
p3.in2
.....:
a = 7;
b = 10;
c = a + 2;
d = a + b;
a = a + c;
```

```
.....:
p3.in3
.....:
a = 5;
b = 7;
c = 2 + a * b;
d = a / 2 - b % 4;
e = a ^ 2;
f = 3 + b * a | 2;
```

```
.....:
p3.in4
.....:
a = 5;
b = 7;
c = 2.0 + a * b;
d = a / 2.0 - b % 4;
e = a & 2;
f = 3.0 + b * a / 2 / 2;
```

```
.....:
p3.out1
.....:
[1] a = 4;
[2] pi = 3.14159;
0 errors found.
a      4
pi     3.14159
```

```
.....:
p3.out2
.....:
[1] a = 7;
[2] b = 10;
[3] c = a + 2;
[4] d = a + b;
[5] a = a + c;
0 errors found.
a      16
b      10
c       9
d      17
```

```
.....:
p3.out3
.....:
[1] a = 5;
[2] b = 7;
[3] c = 2 + a * b;
[4] d = a / 2 - b % 4;
[5] e = a ^ 2;
[6] f = 3 + b * a | 2;
0 errors found.
a       5
b       7
c      37
d      -1
e       7
f      38
```

```
.....:
p3.out4
.....:
[1] a = 5;
[2] b = 7;
[3] c = 2.0 + a * b;
[4] d = a / 2.0 - b % 4;
[5] e = a & 2;
[6] f = 3.0 + b * a / 2 / 2;
0 errors found.
a       5
b       7
c      37.00000
d     -0.50000
e       0
f     11.00000
```

```
.....  
p3.in5  
.....  
a = 7;  
b = 5;  
c = a < b;  
d = a < 2 * b;  
e = a >= b;  
f = a >= 2 * b;
```

```
.....  
p3.out5  
.....  
[1] a = 7;  
[2] b = 5;  
[3] c = a < b;  
[4] d = a < 2 * b;  
[5] e = a >= b;  
[6] f = a >= 2 * b;  
0 errors found.  
a      7  
b      5  
c      0  
d      1  
e      1  
f      0
```

```
.....  
p3.in6  
.....  
a = 5; a++;  
b = 7; --b;  
c = 2; c += a;  
d = -3; d -= a;  
e = 7; e *= a;  
f = -6; f /= 2;  
g = 3; g %= g;  
h = 4; h >>= 2;  
i = 7; i <<= 3;  
j = 6; j ^= 4;
```

```
.....  
p3.out6  
.....  
[1] a = 5; a++;  
[2] b = 7; --b;  
[3] c = 2; c += a;  
[4] d = -3; d -= a;  
[5] e = 7; e *= a;  
[6] f = -6; f /= 2;  
[7] g = 3; g %= g;  
[8] h = 4; h >>= 2;  
[9] i = 7; i <<= 3;  
[10] j = 6; j ^= 4;  
0 errors found.  
a      6  
b      6  
c      8  
d     -9  
e     42  
f     -3  
g      0  
h      4  
i     56  
j      2
```

```
.....  
p3.in7  
.....  
a = 7;  
b = 3;  
c = a, b;  
d = a < c ? b : a + 2;  
e = a > c ? b : a + 2;
```

```
.....  
p3.out7  
.....  
[1] a = 7;  
[2] b = 3;  
[3] c = a, b;  
[4] d = a < c ? b : a + 2;  
[5] e = a > c ? b : a + 2;  
0 errors found.  
a      7  
b      3  
c      7  
d      9  
e      3
```