

CS 415 Project - Spring 2000

Department of Computer Science
Rutgers University

Due 7pm, Monday 4/3

1 Building a Type Checker

The purpose of this project is to give you experience with symbol table manipulation and semantic error detection. You need to build a symbol tables containing entries for declared procedures, functions, parameters, and variables for the Pascal subset in phase 2. Names are case sensitive. Your implementation should have the following components:

attr.h & attr.c:	declarations of types, attribute operations
symtab.h & symtab.c:	symbol table operations
scan.l:	lex input
parse.y:	yacc input

You should enforce the following restrictions on the types of objects.

Declarations

```
"\n***Error: duplicate declaration of %s\n"
```

Array declaration

```
"\n***Error: lower bound exceeds upper bound\n"
```

Procedure stmt

```
"\n***Error: id %s is not a procedure\n"
```

If stmt

```
"\n***Error: exp in if stmt must be boolean\n"
```

While stmt

```
"\n***Error: exp in while stmt must be boolean\n"
```

Writeln stmt (accepts only single integers or string constants)

```
"\n***Error: illegal type for write\n"
```

Assignment stmt

```
"\n***Error: function name on left side of :=\n"  
    NOTE: only an error outside of function returning a value  
"\n***Error: assignment types do not match\n"  
"\n***Error: assignment to nonscalar\n"
```

Function invocation in exp

```
"\n***Error: id %s is not a function\n"
```

Array subscript operation id[exp] in exp

```
"\n***Error: subscript exp not type integer\n"  
"\n***Error: id %s is not an array\n"
```

Identifier

```
"\n***Error: undeclared identifier %s\n"
```

Correspondence between formal and actual parameters

```
"\n***Error: type of actual parameter does not match %s\n"  
    NOTE: The number of elements of array parameters is ignored.  
"\n***Error: numbers of formal and actual parameters do not match\n"
```

Arithmetic, logical, and relational operators in exp

```
"\n***Error: left operand of %s must be %s\n"  
"\n***Error: right operand of %s must be %s\n"  
"\n***Error: operands of %s have different types\n"  
"\n***Error: operand of %s must be %s\n"
```

You must use the scanner and parser that you built for the first two phases. The files `attr.h`, `attr.c`, `syntab.h`, and `syntab.c` contain some suggestions on how to implement this project. Feel free to use or ignore any of the code provided. The data structure definitions are incomplete – you may need to add additional fields to each struct. Some hints:

- Names are case sensitive. Remember that Pascal provides nested scoping. The approach you should take for this project is to provide a symbol table for each scope, linking together the symbol tables from different scopes. Looking for an identifier thus may require searching in several symbol tables. Note: There are no forward declarations of procedures or functions in our Pascal subset. Therefore only previously declared variables are visible.
- Efficiency is not the main concern of this project. You may use simple unordered lists to implement your symbol table. If you design your symbol table interface well, it should be simple to replace the underlying table implementation with a hashing scheme later.
- Remember that in Pascal, functions return values and procedures don't. The return value is indicated by assigning a value to the name of the function. See the demo files for examples.

- Logical operators (and, or, not) require boolean arguments. The remaining operators (e.g., +) require integer arguments, with the exception of the equality/inequality operators (==, !=), which are applicable to any of the base types as long as the two arguments match.

2 What To Do

To get started, you should download `project3.tar.gz` from the class web. As usual, you create the executable by typing `make`. There are two demo files that you can use during the validation process of your type checker. For this project you may need to change the files `attr.c`, `attr.h`, `syntab.c`, and `syntab.h` (as well as the scanner and parser you implemented previously).