COEN 175

Phase 3 - Week 2

TAs

- Chris Desiniotis: cdesiniotis@scu.edu
  - Office Hours: Friday 12 - 2 pm
- Antonio Gigliotti: agigliotti@scu.edu
  - Office Hours: Thursday 11 - 1 pm

Extra Help/Tutoring

- Tau Beta Pi Tutoring
- Link to Tutoring schedule
  - https://sites.google.com/scu.edu/scutaubetapi/tutoring?authuser=1&pli=1

Function Clarification

- Functions should always be defined in the global scope
- Where you output “define function” does not determine which scope you are defining the function in
- 2 approaches to this issue

Input File

```
int x, *p;

int f(int a, int b)
{
    char c[10];
}
```

Output File

```
open file scope
declare x as (INT, 0, SCALAR)
declare p as (INT, 1, SCALAR)
define f as (INT, 0, FUNCTION)
open function scope
declare a as (INT, 0, SCALAR)
declare b as (INT, 0, SCALAR)
declare c as (CHAR, 0, ARRAY)
close function scope
close file scope
```

Two Approaches

```
// Cout before function scope is opened
static void globalOrFunction()
{
	//…
	Parameters *params = new Parameters;
	_type = Type(typespec, indirection, params);
	defineFunction(name, _type);
	openScope();
	parameters(params);
	match(‘)’);
	//…
	closeScope();
	//…
}
```

```
// Cout after function scope is opened
static void globalorFunction()
{
	//…
	openScope();
	Parameters *params = parameters();
	_type = Type(typespec, indirection, params);
	defineFunction(name, _type);
	match(‘)’);
	//…
	closeScope();
	//…
}
```

Rest of Phase 3

1. Write Symbol and Scope classes
2. Write checker functions

- Due February 7th 11:59PM

But first … Did you break your parser?

- Run the phase 2 examples and test cases and make sure your parser can still parse!
- You should not need to modify parser.cpp for the rest of phase 3

1. Write Symbol and Scope Classes

- Symbol.h and Scope.h given in class
- Write Symbol.cpp
  - A symbol is simply a name and a Type
- Write Scope.cpp
  - If using the Linked List method:
    - Each scope object should have a pointer to its enclosing scope and a vector of Symbol pointers
  - Necessary member functions:
    - insert - add symbol to this scope (add symbol pointer to vector)
    - find - find and return symbol within this scope
    - lookup - find and return nearest symbol within this scope and enclosing scopes
    - accessor functions

2. Write checker.cpp

- Use global Scope pointers to reference:
  - the global scope
  - the current scope
- Create global strings for the 5 types of errors
  - Use report function with those error strings (see next slide)
- Functions:
  - openScope - open new scope and set as the current scope
  - closeScope - close current scope and set current scope to the enclosing scope
  - defineFunction - ensure it hasn’t been defined before and matches previous declaration if any
  - declareFunction - add symbol to global scope and if previously declared, make sure type matches
  - declareVariable - check for possible errors and add symbol to current scope (pass in name and Type)
  - checkIdentifier - check if a name has been declared

How to use report()

- Define a static string for error reporting (in checker):
  - `static string redefined = "redefinition of '%s'";`
- When encountering an error, call report:
  - `report(redefined, name)`
  - Note: ‘name’ is replacing %s in the string

How to consume stderr

- Remember: stderr is being checked for phase 3 (NOT stdout)
- For testing, redirect stderr to a file:
  - `./scc < hello.c` **`2>hello.myerr`**
- Optionally, throw away stdout
  - `./scc < hello.c 2>hello.myerr` **`>/dev/null`**
- Compare your error file with the solutions
  - `diff hello.myerr hello.err`