COEN 175

Phase 3 - Week 1

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

Phase 3 - Symbol Table

1. Instrument parser for scopes
2. Replace print statements with function calls to openScope() / closeScope()
3. Instrument parser for local / global declarations
4. Write type class
5. Use Type ostream operator
6. Create checker functions

- Due at 11:59PM February 7th

1. Instrument parser for scopes

- cout every time a scope is opened/closed
- Differentiate between global and function scopes

```
1   int a;
2
3   int main(int b)
4   {
5       while(1)
6       {
7           b = a;
8       }
9       return 0;
10  }
```

```
[agigliot@linux10614 phase3]$ ./scc < test_scope.c
opening scope
opening scope
opening scope
closing scope
closing scope
closing scope
[agigliot@linux10614 phase3]$
```

2. Replace couts with function calls to openScope() / closeScope()

- Create checker.h and checker.cpp
- Add checker.o to Makefile
- Create openScope() and closeScope() in checker
  - Put cout statements in these functions
  - Call them from parser.cpp

3. Instrument parser for local / global declarations

- Modify parser.cpp to pass information across functions
  - **unsigned** pointers()
  - **int** specifier()
  - void declarator(**int**)
  - …
- Print out attributes when a type is declared

```
// modified code

void declarator(int typespec) {
    unsigned indirection = pointers();
    match(ID);

    if (lookahead == '[') {
        match('[');
        match(INTEGER);
        match(']');
        cout << "(" << typespec … << ", ARRAY)" << endl;
    } else
        cout << "(" << typespec … << ", SCALAR)" << endl;
}
```

- Let's modify the code for `specifier()` to return the type specifier.

```
// modified code

int specifier() {
    int typespec = lookahead;

    if (lookahead == INT)
        match(INT);
    else if (lookahead == DOUBLE)
        match(DOUBLE);
    else
        match(CHAR);

    return typespec;
}
```

Example output at this point

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)
declare 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
```

Useful Code - identifier()

- Similar function needs to implemented for numbers

```
static string identifier()
{
    string val = lexbuf;
    match(ID);

    return val;
}
```

```
pointers();
match(ID);
```

```
unsigned indirection = pointers();
string name = identifier();
```

4. Write type class

- Write Type.h (given in class) and Type.cpp
- Remember to add Type.o to Makefile
- Type class:
  - Member variables: Specifier, indirection, kind
  - We will overload the **==** and **!=** operators to easily compare types
  - We will overload the **<<** operator to easily print out types
  - Types will be **immutable** (only providing assessors, not mutators)

5. Use Type ostream operator

- After overloading Type ostream operator, call this operator from parser when outputting information about a type

- Example:
  - cout << Type(typespec, indirection) << endl;

6. Create checker functions

- **declareVariable(), declareFunction(), defineFunction(), checkID()**
- Just put cout statements in these functions as placeholders for now

- declareVariable(), declareFunction(), defineFunction()
  - Call when encountering a declaration/definition
- checkID()
  - Call when encountering an identifier (in primaryExpression)
  - Simple cout for now:
    - E.g. cout << “check ” << name << endl;