COEN 175

Phase 4 - Week 1

TAs

- Chris Desiniotis: cdesiniotis@scu.edu
 - o Office Hours: Friday 12 2 pm
- Antonio Gigliotti: <u>agigliotti@scu.edu</u>
 - 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

Important Reminders

- Read the assignment carefully!
 - Too many of your questions can be answered by simply reading the assignment
- Follow our slides each week
 - Too many of you are not finishing week 1 objectives prior to week 2
- Stay the entire lab section and attend office hours
 - Do not depend on getting your questions answered via email on Saturday/Sunday

Phase 4 - Type Checking

Goal for this week

- Finish checker for all of expression
- Next week we will do function calls, assignment, statements, and return
- Modify parser
- 2. Modify primaryExpression
- 3. Write additional Type.cpp functions
- 4. Writer checker functions for operators

Due February 21th 11:59PM

1. Modify Parser

- Return a Type from expression functions
- Each expression function takes an Ivalue

Binary operator example

Unary operator example

```
static (ype prefixExpression(bool &lvalue)
{
    (Iype expr;

    if (lookahead == '!') {
        match('!');
        expr = prefixExpression(lvalue);
        cout << "not" << endl;
} ... {
    } else
        expr = postfixExpression(lvalue);
    eturn expr;
}</pre>
```

1. Modify Parser

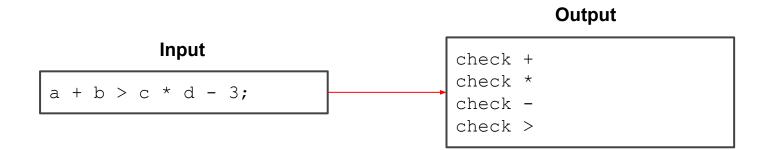
- Update Ivalues in expression functions
 - Initially declare Ivalue in statement(), before you call expression()
 - Only update Ivalue when an operator is matched
 - Always update Ivalue last (see example below)
- Add print statements for type checking operations
 - These will be replaced by function calls to checker.cpp later

```
static Type expression(bool &lvalue)
{
    Type left = logicalAndExpression(lvalue);
    while (lookahead == OR) {
        match(OR);
        Type right = logicalAndExpression(lvalue);
        cout << "check ||" << endl;
        Ivalue = false;
    }
    return left;
}</pre>
```

2. Modify primaryExpression

- Return a Type object for each scenario in primaryExpression()
- Example: if you match an integer, return an integer Type object
- If an identifier, return the type from its Symbol
 - checkIdentifier() returns a Symbol * and can retrieve type from there

At this point



3. Write additional Type.cpp functions

- Helper functions for type checking
 - Type promote() const;
 - bool isCompatibleWith(const Type &that) const;
 - bool isInteger() const;
 - bool isPointer() const;
 - bool isValue() const;
- Read assignment carefully for these

4. Write checker functions for operators

- Follow lab guide closely for what to check
- Suggested implementation order (by difficulty)
 - Multiplicative, Equality, Relational
 - Logical
 - Postfix
 - Additive, Prefix
- Call checker functions from parser

```
static Type expression(bool &lvalue)
    Type left = logicalAndExpression(lvalue);
    while (lookahead == OR) {
        match(OR);
        Type right = logicalAndExpression(lvalue);
        left = checkLogicalOr(left, right);
        lvalue = false;
    return left;
```