Lab 6

COEN 175 Compilers

Overview For Lab 6

Goal

- Create a Type checker

Submission

- Submit a tarball of your cpps and make files in folder called phase3
- Due Date: Sunday February 20th

Goals for this week

1. Modify expression functions in parser.cpp
2. Modify primary expression function in parser.cpp
3. Add checker functions for expressions
4. Create abstraction functions in Type class
5. Start implementing checker functions for expressions

1. Modify Expression Functions in parser

- Have them return the Type of resulted expression
- Pass boolean lvalue in by reference
- Example for expression() provided in the assignment doc
- Put in a cout for each checked operator
- Update lvalue according to rules in assignment
  - Declare and set lvalue in statement() before it is passed into expression()
  - Only update when operator matched
  - Always update lvalue last
- Can skip postfix for this week

1. Modify Expression Functions in parser

Binary Operators Example

```
static Type expression(bool &lvalue)
{
    Type left = logicalAndExpression(lvalue);

    while (lookahead == OR) {
        match(OR);
        Type right = logicalAndExpression(lvalue);
        cout << "check ||" << endl;
        lvalue = false;
    }
    return left;
}
```

Unary Operators Example

```
static Type prefixExpression(bool &lvalue)
{
    if (lookahead == '!') {
        match('!');
        Type left = prefixExpression(lvalue);
        cout << "check !" << endl;
        lvalue = false;
        return left;

    } else if (lookahead == '-') {
```

2. Modify Primary Expression in parser

- Return Type object based on which case it hits
  - Identifier gets its type from its Symbol table entry
  - All else based simply on what type gets matched

3. Add checker functions in expressions

- Start by having checker functions with the cout from step 2.
- Replace the cout and make sure code still works
- Checker function should take in operands as parameters
- Some of them you may be able to merge into common functions
  - Take in operator as parameter

```
Type right = logicalAndExpression(lvalue);
left = checkLogicalOr(left, right);
lvalue = false;
```

```
Type left = prefixExpression(lvalue);
left = checkNot(left);
lvalue = false;
return left;
```

4. Abstraction functions in Type

- Helper functions to Type check
  - Type promote() const;
  - bool isValue() const;
  - bool isCompatibleWith(const Type &that) const;
  - bool isPointer() const;
  - bool isInteger() const;
- Refer to assignment for rules for these

5. Start checker function implementation

- Follow the rules from the assignment document
- isValue(), promote(), checkLogicalOr(), checkMultiply() provided in class today

- **Correction to checkMultiply() from lecture**
  - if (left == integer & right == integer) ⇒ if (t1 == integer && t2 == integer)

Tips

- Read carefully to translate the rules from english to C++ code
  - Individually they are not complicated logic
- Recompile your code frequently to make sure it still works
- Boolean type checker functions should account for promoted types
- **READ THE SEMANTIC RULES CAREFULLY**

Reminder for Lecture

Take the practice midterm

- You won’t be able to take the midterm if you do not