

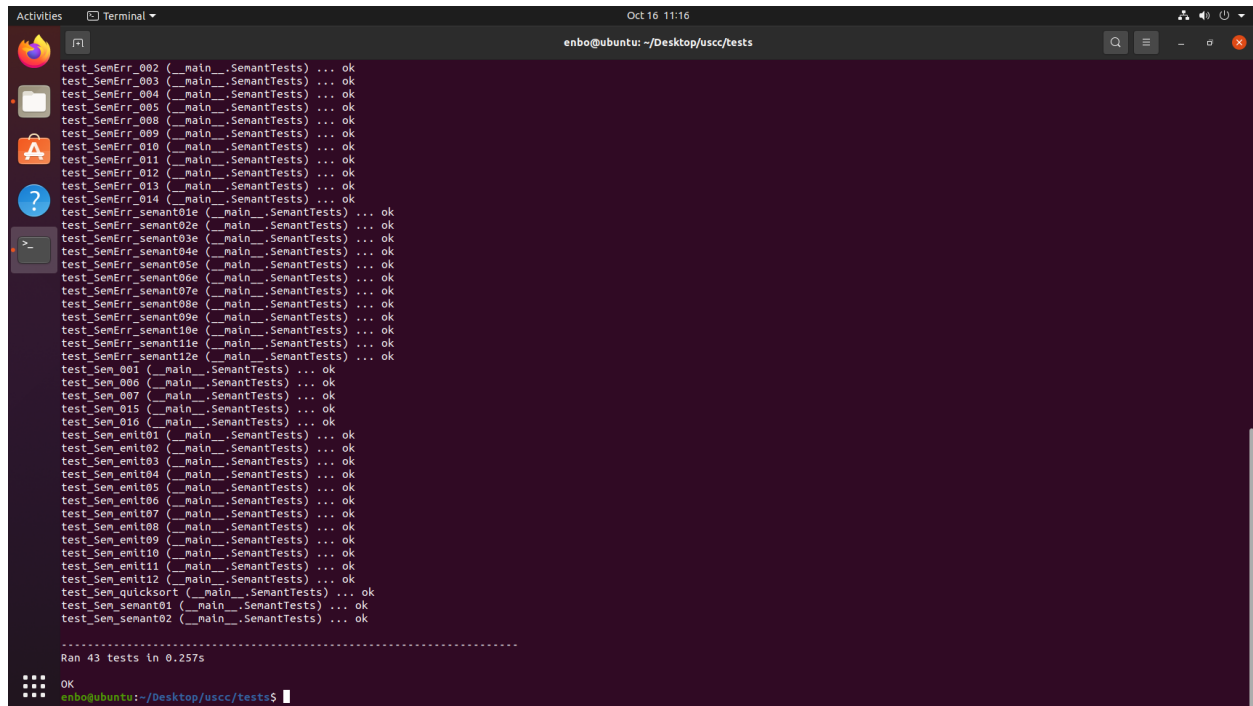
CSE 504 Compiler Design PA-02 Report

Name: Enbo Yu

ID: 113094714

Date: 16 Oct

I. Test Result:



```
test_SemErr_002 (__main___.SemantTests) ... ok
test_SemErr_003 (__main___.SemantTests) ... ok
test_SemErr_004 (__main___.SemantTests) ... ok
test_SemErr_005 (__main___.SemantTests) ... ok
test_SemErr_008 (__main___.SemantTests) ... ok
test_SemErr_009 (__main___.SemantTests) ... ok
test_SemErr_010 (__main___.SemantTests) ... ok
test_SemErr_011 (__main___.SemantTests) ... ok
test_SemErr_012 (__main___.SemantTests) ... ok
test_SemErr_013 (__main___.SemantTests) ... ok
test_SemErr_014 (__main___.SemantTests) ... ok
test_SemErr_senant01e (__main___.SemantTests) ... ok
test_SemErr_senant02e (__main___.SemantTests) ... ok
test_SemErr_senant03e (__main___.SemantTests) ... ok
test_SemErr_senant04e (__main___.SemantTests) ... ok
test_SemErr_senant05e (__main___.SemantTests) ... ok
test_SemErr_senant06e (__main___.SemantTests) ... ok
test_SemErr_senant07e (__main___.SemantTests) ... ok
test_SemErr_senant08e (__main___.SemantTests) ... ok
test_SemErr_senant09e (__main___.SemantTests) ... ok
test_SemErr_senant10e (__main___.SemantTests) ... ok
test_SemErr_senant11e (__main___.SemantTests) ... ok
test_SemErr_senant12e (__main___.SemantTests) ... ok
test_Sem_001 (__main___.SemantTests) ... ok
test_Sem_006 (__main___.SemantTests) ... ok
test_Sem_007 (__main___.SemantTests) ... ok
test_Sem_015 (__main___.SemantTests) ... ok
test_Sem_016 (__main___.SemantTests) ... ok
test_Sem_entit01 (__main___.SemantTests) ... ok
test_Sem_entit02 (__main___.SemantTests) ... ok
test_Sem_entit03 (__main___.SemantTests) ... ok
test_Sem_entit04 (__main___.SemantTests) ... ok
test_Sem_entit05 (__main___.SemantTests) ... ok
test_Sem_entit06 (__main___.SemantTests) ... ok
test_Sem_entit07 (__main___.SemantTests) ... ok
test_Sem_entit08 (__main___.SemantTests) ... ok
test_Sem_entit09 (__main___.SemantTests) ... ok
test_Sem_entit10 (__main___.SemantTests) ... ok
test_Sem_entit11 (__main___.SemantTests) ... ok
test_Sem_entit12 (__main___.SemantTests) ... ok
test_Sem_quickSort (__main___.SemantTests) ... ok
test_Sem_senant01 (__main___.SemantTests) ... ok
test_Sem_senant02 (__main___.SemantTests) ... ok
-----
Ran 43 tests in 0.257s
OK
enbo@ubuntu: ~/Desktop/uscc/tests$
```

Ran 43 tests in 0.257s

OK

II. Class Details

ASTExpr.cpp

-finalizeOp:

There are four sub-class which are helped by the function of finalizeOP. These functions just need to set the type of the node to integer, and return true if both lhs and rhs are integers

```

27 bool ASTLogicalAnd::finalizeOp() noexcept
28 {
29     if (mLHS->getType() == Type::Int && mRHS->getType() == Type::Int)
30     {
31         this->mType = Type::Int;
32         return true;
33     }
34     return false;
35 }
36
37 bool ASTLogicalOr::finalizeOp() noexcept
38 {
39     if (mLHS->getType() == Type::Int && mRHS->getType() == Type::Int)
40     {
41         this->mType = Type::Int;
42         return true;
43     }
44     return false;
45 }
46
47 bool ASTBinaryCmpOp::finalizeOp() noexcept
48 {
49     if ((mLHS->getType() == Type::Int || mLHS->getType() == Type::Char) &&
50         (mRHS->getType() == Type::Int || mRHS->getType() == Type::Char))
51     {
52         this->mType = Type::Int;
53         return true;
54     }
55     return false;
56 }
57
58 bool ASTBinaryMathOp::finalizeOp() noexcept
59 {
60     if ((mLHS->getType() == Type::Int || mLHS->getType() == Type::Char) &&
61         (mRHS->getType() == Type::Int || mRHS->getType() == Type::Char))
62     {
63         this->mType = Type::Int;
64         return true;
65     }
66     return false;
67 }
68

```

Symbols.cpp

-ScopeTable:

By the function of ScopeTable, we can create the elements in the constructor of ScopeTable. After printing the symbol table, the current table elements will be added in its parents' children lists.

-search:

By the function of Search, we can search for the analogous or equivalent identifier and go through the following linked scope tables.

```

179 // Exits the current scope and moves the current scope back to
180 // the previous scope table.
181 void SymbolTable::exitScope()
182 {
183     mCurrScope = mCurrScope->getParent();
184 }
185
186 SymbolTable::ScopeTable::ScopeTable(ScopeTable* parent) noexcept
187 : mParent(parent)
188 {
189     if (parent)
190         parent->mChildren.push_back(this);
191 }
192
193 SymbolTable::ScopeTable::~ScopeTable() noexcept
194 {
195     // mParent->mChildren.remove(this);
196
197     for (auto& it : mSymbols)
198         delete it.second;
199
200     for (auto& it : mChildren)
201         delete it;
202
203     mSymbols.clear();
204     mChildren.clear();
205 }
206
207 // Adds the requested identifier to the table
208 void SymbolTable::ScopeTable::addIdentifier(Identifier* ident)
209 {
210     mSymbols.insert(std::make_pair(ident->getName(), ident));
211 }
212

```

```

213 // Searches this scope for an identifier with
214 // the requested name. Returns nullptr if not found.
215 Identifier* SymbolTable::ScopeTable::searchInScope(const char* name) noexcept
216 {
217     auto result = mSymbols.find(name);
218     if (result != mSymbols.end())
219         return result->second;
220     return nullptr;
221 }
222
223 // Searches this scope first, and if not found searches
224 // through parent scopes. Returns nullptr if not found.
225 Identifier* SymbolTable::ScopeTable::search(const char* name) noexcept
226 {
227     Identifier *ident = nullptr;
228     ScopeTable *scope = this;
229     while (scope != nullptr) {
230         ident = scope->searchInScope(name);
231         if (ident)
232             break;
233         scope = scope->mParent;
234     }
235     return ident;
236 }
237

```

Parse.cpp

-getVariable:

By the function of getVariable, we can check the following input identifier at first. When we get a null pointer, this function will return the error that Use of undeclared identifier... Also, the dummy variable will be returned as a substitute. Which means that this function should first try to get the identifier from the symbol table, and return the identifier if it's found.

-charToInt() and intToChar:

We can use these two functions to transform the type of input, like char to int or int to char. For example, if the expression passed to charToInt is already an integer, we can just return the expression directly. Otherwise, if it receives an expression of type char

```
351 Identifier* Parser::getVariable(const char* name) noexcept
352 {
353     Identifier *ident = mSymbols.getIdentifier(name);
354     if (ident == nullptr)
355     {
356         std::string msg = "Use of undeclared identifier ";
357         msg += name;
358         msg += "\'";
359         reportSemantError(msg);
360         ident = mSymbols.getIdentifier("@@variable");
361     }
362     return ident;
363 }
364
365 const char* Parser::getTypeText(Type type) const noexcept
366 {
367     switch (type)
368     {
369     case Type::Char:
370         return "char";
371     case Type::Int:
372         return "int";
373     case Type::Void:
374         return "void";
375     case Type::CharArray:
376         return "char[]";
377     case Type::IntArray:
378         return "int[]";
379     case Type::Function:
380         return "function";
381     }
382 }
```

```

387 std::shared_ptr<ASTExpr> Parser::charToInt(std::shared_ptr<ASTExpr> expr) noexcept
388 {
389     std::shared_ptr<ASTExpr> retVal = expr;
390
391     if (expr->getType() == Type::Char)
392     {
393         ASTConstantExpr *constExpr = dynamic_cast<ASTConstantExpr*>(expr.get());
394         if (constExpr)
395         {
396             constExpr->changeToInt();
397         }
398         else
399         {
400             ASTToIntExpr* toCharExpr = dynamic_cast<ASTToIntExpr*>(expr.get());
401             if (toCharExpr)
402                 return toCharExpr->getChild();
403             return make_shared<ASTToIntExpr>(expr);
404         }
405     }
406     return retVal;
407 }
408

```

```

410 std::shared_ptr<ASTExpr> Parser::intToChar(std::shared_ptr<ASTExpr> expr) noexcept
411 {
412     std::shared_ptr<ASTExpr> retVal = expr;
413
414     if (expr->getType() == Type::Int)
415     {
416         ASTConstantExpr* constExpr = dynamic_cast<ASTConstantExpr*>(expr.get());
417         if (constExpr)
418         {
419             constExpr->changeToChar();
420         }
421         else
422         {
423             ASTToCharExpr* toIntExpr = dynamic_cast<ASTToCharExpr*>(expr.get());
424             if (toIntExpr)
425                 return toIntExpr->getChild();
426             return make_shared<ASTToCharExpr>(expr);
427         }
428     }
429
430     return retVal;
431 }
432

```