

CACT-PR001 词法语法分析

任务说明

成员组成

实验设计

1. 设计思路
2. 实验实现
3. 其它

总结

1. 实验结果总结
2. 分成员总结

参考资料

CACT-PR001 词法语法分析

任务说明

本次实验需要完成的任务包括：

1. 安装并熟悉 **Antlr** 。
2. 编写 **.g4** 文件，使用 **Antlr** 生成词法、语法分析器（Lexer, Parser）和访问接口：（Listener/Visitor），对 **.cact** 文件进行词法分析和语法分析。
3. 修改 **Antlr** 中的文法错误处理机制：对于符合词法和语法规范的 **.cact** 文件，返回值为0，否则返回非0值。

成员组成

第7组成员：

陈飞羽 2018K8009929031

彭思邈 2018K8009908040

袁欣怡 2018K8009929021

实验设计

1. 设计思路

(1) CACT.g4

Antlr 的文法文件以 `.g4` 为文件名后缀。在这个文法文件中，我们需要根据 Antlr 的语法规则来定义算术表达式的文法。现有的 `CACT.g4` 文件中已经实现了大部分的设计，需要我们完成的是：标识符 `Ident`、整型常量 `IntConst`、布尔型常量 `BoolConst`、单精度浮点常量 `FloatConst`、双精度浮点常量 `DoubleConst`。

以整型常量 `IntConst` 为例，我们设计的文法为：

```
1  fragment
2  Digit
3      : [0-9]
4      ;
5
6  IntConst
7      : DecimalConst // 十进制
8      | OctalConst   // 八进制
9      | HexadecimalConst // 十六进制
10     ;
11
12 fragment
13 DecimalConst
14     : '0'
15     | NonzeroDigit Digit*
16     ;
17
18 fragment
19 OctalConst
20     : '0' OctalDigit+
21     ;
22
23 fragment
24 HexadecimalConst
25     : HexadecimalPrefix HexadecimalDigit+
26     ;
27
28 fragment
29 NonzeroDigit
30     : [1-9]
31     ;
32
33 fragment
34 OctalDigit
35     : [0-7]
```

```

36     ;
37
38 fragment
39 HexadecimalPrefix
40     : '0x'
41     | '0X'
42     ;
43
44 fragment
45 HexadecimalDigit
46     : [0-9a-fA-F]
47     ;

```

(2) main.cpp

main函数完成以下工作：

- 生成 `CACTLexer` 、 `CACTParser` 实例，并且将命令行指定的输入传入。
- 调用 `parser.compUnit()` 作为文法入口，生成 `ParserTree` 。
- 遍历得到的 `tree` ， `SemanticAnalysis` 的实例 `listener` 将会进行语义分析（在这一实验中暂时没有用到）。
- 打印调试内容（AST）。
- 根据是否出错确定返回值。

`VerboseErrorListener` 类继承了 `BaseErrorListener` 类，通过重写 `syntaxError` 方法实现了自定义错误处理，通过使用公共类属性 `level` 检查带编译文件状态。

具体来说，在main函数中，在 `lexer` 和 `parser` 的声明之后，构建 `ParseTree` 之前添加如下代码：

```

1 parser.removeErrorListeners();           //移除原有ErrorListener实例
2 VerboseErrorListener * ErrListenerInst = new VerboseErrorListener();
3 parser.addErrorListener(ErrListenerInst); //添加VerboseErrorListener实例

```

其中 `VerboseErrorListener` 函数在 `semanticAnalysis.h` 中实现。

(3) semanticAnalysis.h

`VerboseErrorListener` 函数的功能：继承 `BaseErrorListener` ，重写了 `syntaxError` 来自定义错误信息。这里的 `syntaxError` 函数会打印错误的行号与字符位置、调用的文法规则、以及错误的符号。

```

1 typedef int status_t;
2 enum{
3     ok,warning,critical,error
4 };
5
6 namespace antlr4{
7     class VerboseErrorListener : public BaseErrorListener{
8     public:
9         status_t level;
10        VerboseErrorListener(){
11            level = ok;
12        }

```

```

13
14     void syntaxError(Recognizer *recognizer, Token * offendingSymbol, size_t line, size_t
charPositionInLine,
15         const std::string &msg, std::exception_ptr e){ //获取错误信息
16         level = error;
17         std::vector<std::string> stack = ((antlr4::Parser *)recognizer)->getRuleInvocationStack();
18         std::reverse(stack.begin(),stack.end());
19         std::cout << "rule stack: [";
20         for(int i=0;i<stack.size();i++)
21             std::cout << stack[i] << ", ";
22         std::cout << "]" << std::endl;
23         std::cout << "line: " << line << ". " << charPositionInLine << " at " << offendingSymbol-
>getText() << std::endl;
24     }
25 };
26 };

```

(4) semanticAnalysis.cpp

对 `SemanticAnalysis::exitVarDecl` 函数进行了修改：

```

1  void SemanticAnalysis::exitVarDecl(CACTParser::VarDeclContext * ctx)
2  {
3      std::cout << "variable define: " << std::endl;
4      for(const auto & var_def : ctx->varDef())
5      {
6          if(var_def->Ident()==NULL) // 需要检查是否读到Ident，否则compiler检查错误程序时可能会试图
打印null指针指向的内容
7              break;
8          std::cout << "\tname: " << var_def->Ident()->getText().c_str() \
9              << " type: " << ctx->bType()->getText().c_str() << std::endl;
10     }
11 }

```

2. 实验实现

(1) 登陆服务器账号，将源代码通过 `Gitlab` 服务器 `clone` 到本地。在 `clone` 前可以配置自己的 `git` 偏好设置，方便 `clone` 操作。

(2) 下载 `Antlr` 到本地，编辑 `.bashrc` 并进行安装，通过 `$antlr4` 指令检查安装结果。

(3) 编写 `CACT.g4` 文件，在 `grammar` 目录下运行 `Antlr`，生成 `parser`。

(4) 编写 `src` 目录下的文件，使用 `parser` 提供的接口对语法树进行操作。

(5) 编写了脚本`code_build.sh`方便测试，检查输出结果是否正确。

3. 其它

无。

总结

1. 实验结果总结

运行 `./batch_test.sh`，依次在`true_sample`上运行`compiler`，根据命名（`false/true`）和程序返回值判断一致性，一致则`pass`。运行结果：

```
1  testing 0: 00_true_main.cact...
2  variable define:
3    name: a type: int
4  debug: hello
5  -----Print AST:-----
6  (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
   (varDef a = (constInitVal (constExp (number 0)))) ;))) (blockItem (stmt return (exp (addExp (mulExp
   (unaryExp (primaryExp (number 0)))))) ;)) ;))) <EOF>)
7  -----
8  pass.
9
10
11
12
13
14
15  testing 1: 01_false_hex_num.cact...
16  rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, ]
17  line: 3:13 at x
18  variable define:
19    name: a type: int
20  debug: hello
21  -----Print AST:-----
22  (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
   (varDef a = (constInitVal (constExp (number 0)))) x ;))) (blockItem (stmt return (exp (addExp (mulExp
   (unaryExp (primaryExp (lVal a)))))) ;)) ;))) <EOF>)
23  error level: 3
24  -----
25  pass.
26
27
28
29
30
```

```

31
32 testing 2: 02_true_octo.cact...
33 variable define:
34     name: a type: int
35 debug: hello
36 -----Print AST:-----
37 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a = (constInitVal (constExp (number 0)))))) ;))) (blockItem (stmt return (exp (addExp (mulExp
    (unaryExp (primaryExp (number 0)))))) ;)) )))) <EOF>)
38 -----
39 pass.
40
41
42
43
44
45
46 testing 3: 03_false_nested_init.cact...
47 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, constInitVal, ]
48 line: 4:13 at {
49 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, ]
50 line: 4:21 at {
51 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
52 line: 4:25 at 4
53 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
54 line: 4:26 at }
55 rule stack: [compUnit, ]
56 line: 4:28 at ;
57 variable define:
58     name: a type: int
59 variable define:
60     name: a type: int
61 debug: hello
62 -----Print AST:-----
63 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 0)))) ;))
    (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef a [ 4 ] =
    (constInitVal { { (constExp (number 1)) , (constExp (number 2)) )) , varDef <missing ';'>))) (blockItem
    (stmt (block { (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number 3))))))
    <missing ';'>)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number 4))))))
    <missing ';'>)) )))) ; return 0 ; }
64 error level: 3
65 -----
66 pass.
67
68
69
70
71
72

```

```

73 testing 4: 04_false_multi_dim_array.cact...
74 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, ]
75 line: 6:9 at [
76 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
77 line: 6:11 at ]
78 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
79 line: 6:17 at ,
80 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
81 line: 6:20 at ,
82 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
83 line: 6:23 at ,
84 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, block, blockItem, stmt, ]
85 line: 6:26 at }
86 rule stack: [compUnit, transUnit, funcDef, block, ]
87 line: 9:0 at <EOF>
88 variable define:
89     name: a type: int
90 variable define:
91     name: b type: int
92 debug: hello
93 -----Print AST:-----
94 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a = (constInitVal (constExp (number 0)))) ;)) (blockItem (decl (varDecl (bType int) (varDef b [
    2 ] [])) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number 2)))))) ] =))
    (blockItem (stmt (block { (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number
    1)))))) ,)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number 2)))))) ,))
    (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (number 3)))))) ,)) (blockItem (stmt
    (exp (addExp (mulExp (unaryExp (primaryExp (number 4)))))) } ;)) (blockItem (stmt return (exp
    (addExp (mulExp (unaryExp (primaryExp (lVal a)))))) ;)) ))) <missing '}'>)) <EOF>)
95 error level: 3
96 -----
97 pass.
98
99
100
101
102
103
104 testing 5: 05_false_number.cact...
105 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, constDecl, constDef, constInitVal, ]
106 line: 4:16 at a
107 const variable define:
108     line 3:1 name: a type: int
109 const variable define:
110     line 4:1 name: b type: int
111 debug: hello
112 -----Print AST:-----

```

```

113 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (constDecl const
    (bType int) (constDef a = (constInitVal (constExp (number 3)))) ;))) (blockItem (decl (constDecl const
    (bType int) (constDef b = constInitVal) a ;))) (blockItem (stmt return (exp (addExp (mulExp (unaryExp
    (primaryExp (lVal a)))))) ;)) ;))) <EOF>)
114 error level: 3
115 -----
116 pass.
117
118
119
120
121
122
123 testing 6: 06_false_oct_num.cact...
124 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, constInitVal, ]
125 line: 3:9 at 0129
126 variable define:
127     name: k type: int
128 debug: hello
129 -----Print AST:-----
130 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef k = (constInitVal 0129)) ;))) (blockItem (stmt return (exp (addExp (mulExp (unaryExp
    (primaryExp (number 0)))))) ;)) ;))) <EOF>)
131 error level: 3
132 -----
133 pass.
134
135
136
137
138
139
140 testing 7: 07_false_hex_num.cact...
141 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, ]
142 line: 3:12 at G
143 variable define:
144     name: n type: int
145 debug: hello
146 -----Print AST:-----
147 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef n = (constInitVal (constExp (number 0x3)))) G ;))) (blockItem (stmt return (exp (addExp
    (mulExp (unaryExp (primaryExp (number 0)))))) ;)) ;))) <EOF>)
148 error level: 3
149 -----
150 pass.
151
152
153
154

```



```

155
156
157 testing 8: 08_false_int_num_decl.cact...
158 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, constInitVal, ]
159 line: 3:10 at ha
160 variable define:
161     name: i type: int
162 debug: hello
163 -----Print AST:-----
164 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef i = constInitVal) ha ;))) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp
    (number 0)))))) ;)) ;))) <EOF>)
165 error level: 3
166 -----
167 pass.
168
169
170
171
172
173
174 testing 9: 09_false_val_name.cact...
175 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, ]
176 line: 3:5 at 7
177 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
178 line: 4:9 at j
179 variable define:
180     name: j type: int
181 debug: hello
182 -----Print AST:-----
183 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef 7 j = (constInitVal (constExp (number 5)))) ;))) (blockItem (stmt return (exp (addExp (mulExp
    (unaryExp (primaryExp (number 7)))))) j ;)) ;))) <EOF>)
184 error level: 3
185 -----
186 pass.
187
188
189
190
191
192
193 testing 10: 10_false_array_visit.cact...
194 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
195 line: 4:4 at ,
196 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
197 line: 4:4 at ,
198 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
199 line: 4:7 at ]

```

```

200 variable define:
201     name: a type: float
202 debug: hello
203 -----Print AST:-----
204 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType float)
    (varDef a [ 10 ] ) ;)) (blockItem stmt) (blockItem (stmt a [])) (blockItem (stmt (exp (addExp (mulExp
    (unaryExp (primaryExp (number 2)))))) ,)) (blockItem (stmt (exp (addExp (mulExp (unaryExp
    (primaryExp (number 4)))))) ] =)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp
    (number 4.0)))))) ;)) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (number
    0)))))) ;)) ;))) <EOF>)
205 error level: 3
206 -----
207 pass.
208
209
210
211
212
213
214 testing 11: 11_false_if_else.cact...
215 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, stmt, block, blockItem, stmt, ]
216 line: 9:1 at }
217 variable define:
218     name: a type: int
219 variable define:
220     name: i type: int
221 debug: hello
222 -----Print AST:-----
223 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a [ 10 ] ) ;)) (blockItem (decl (varDecl (bType int) (varDef i) ;)) (blockItem (stmt if ( (cond
    (lOrExp (lAndExp (eqExp (eqExp (relExp (addExp (mulExp (unaryExp (primaryExp (lVal a [ (exp
    (addExp (mulExp (unaryExp (primaryExp (number 1)))))) ])))))) == (relExp (addExp (mulExp
    (unaryExp (primaryExp (number 0))))))))) ) (stmt (block { (blockItem (stmt (lVal i) = (exp (addExp
    (mulExp (unaryExp (primaryExp (number 1)))))) <missing '>')) else (stmt (block { (blockItem (stmt
    (lVal i) = (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))))) ;)) ;))) (blockItem (stmt return
    (exp (addExp (mulExp (unaryExp (primaryExp (lVal i)))))) ;)) ;))) <EOF>)
224 error level: 3
225 -----
226 pass.
227
228
229
230
231
232
233 testing 12: 12_true_comment.cact...
234 variable define:
235     name: a type: int
236 variable define:

```

```

237     name: i type: int
238 debug: hello
239 -----Print AST:-----
240 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a [ 10 ] ) ;)) (blockItem (decl (varDecl (bType int) (varDef i = (constInitVal (constExp (number
    3)))) ;)) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal i)))))) ;)) ;))
    <EOF>))
241 -----
242 pass.
243
244
245
246
247
248
249 testing 13: 13_false_nested_comment.cact...
250 rule stack: [compUnit, transUnit, funcDef, block, ]
251 line: 9:1 at *
252 variable define:
253     name: a type: int
254 variable define:
255     name: i type: int
256 debug: hello
257 -----Print AST:-----
258 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a [ 10 ] ) ;)) (blockItem (decl (varDecl (bType int) (varDef i = (constInitVal (constExp (number
    3)))) ;)) * / (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal i)))))) ;)) ;))
    <EOF>))
259 error level: 3
260 -----
261 pass.
262
263
264
265
266
267
268 testing 14: 14_true_sample.cact...
269 variable define:
270     name: a type: int
271 variable define:
272     name: i type: int
273 debug: hello
274 -----Print AST:-----
275 (compUnit (transUnit (decl (varDecl (bType int) (varDef a [ 4 ] = (constInitVal { }))) ;)) (funcDef
    (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef i))) (blockItem (stmt
    (IVal i) = (exp (addExp (addExp (mulExp (unaryExp (primaryExp (IVal i)))))) + (mulExp (unaryExp
    (primaryExp (number 1)))))) ;)) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp
    (IVal i)))))) ;)) ;)) <EOF>))

```

```

276 -----
277 pass.
278
279
280
281
282
283
284 testing 15: 15_true_syntax_false_semantic.cact...
285 variable define:
286     name: a type: int
287 variable define:
288     name: b type: int
289 debug: hello
290 -----Print AST:-----
291 (compUnit (transUnit (decl (varDecl (bType int) (varDef a) ;)) (funcDef (funcType int) func1 (
(funcFParams (funcFParam (bType int) a)) ) (block { (blockItem (stmt (IVal a) = (exp (addExp
(addExp (mulExp (unaryExp (primaryExp (IVal a)))))) - (mulExp (unaryExp (primaryExp (number
1)))))) ;)) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal a)))))) ;)) })))
(funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef b) ;)))
(blockItem (stmt (IVal b) = (exp (addExp (mulExp (unaryExp func2 ( (funcRParams (exp (addExp
(mulExp (unaryExp (primaryExp (IVal a)))))) ) ) ) ) ;)) (blockItem (stmt return (exp (addExp (mulExp
(unaryExp (primaryExp (IVal b)))))) ;)) }))) <EOF>)
292 -----
293 pass.
294
295
296
297
298
299
300 testing 16: 16_false_if_else.cact...
301 rule stack: [compUnit, transUnit, funcDef, block, ]
302 line: 10:1 at else
303 variable define:
304     name: a type: int
305 variable define:
306     name: i type: int
307 debug: hello
308 -----Print AST:-----
309 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
(varDef a [ 10 ] ) ;)) (blockItem (decl (varDecl (bType int) (varDef i) ;)) (blockItem (stmt if ( (cond
(LOrExp (LOrExp (eqExp (eqExp (relExp (addExp (mulExp (unaryExp (primaryExp (IVal a [ (exp
(addExp (mulExp (unaryExp (primaryExp (number 1)))))) ])))))) == (relExp (addExp (mulExp
(unaryExp (primaryExp (number 0)))))) ) (stmt (IVal i) = (exp (addExp (mulExp (unaryExp
(primaryExp (number 3)))))) ;)) (blockItem (stmt (IVal a [ (exp (addExp (mulExp (unaryExp
(primaryExp (IVal i)))) ) ] = (exp (addExp (mulExp (unaryExp (primaryExp (number 8)))))) ;)) else
(blockItem (stmt (IVal i) = (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))))) ;)) (blockItem
(stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal i)))))) ;)) }))) <EOF>)

```

```

310 error level: 3
311 -----
312 pass.
313
314
315
316
317
318
319 testing 17: 17_false_multi_dim_fparam.cact...
320 rule stack: [compUnit, transUnit, funcDef, ]
321 line: 3:22 at [
322 variable define:
323     name: a type: int
324 debug: hello
325 -----Print AST:-----
326 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 2)))) ;))
(funcDef (funcType int) foo ( (funcFParams (funcFParam (bType int) a) , (funcFParam (bType int) b [
])) [ 3 ] ) { return c ; }) (funcDef (funcType int) main ( ) (block { (blockItem (stmt return (exp (addExp
(mulExp (unaryExp (primaryExp (IVal a)))))) ;)) ;))) <EOF>)
327 error level: 3
328 -----
329 pass.
330
331
332
333
334
335
336 testing 18: 18_true_whole_sample.cact...
337 const variable define:
338     line 1:0 name: a type: int
339 variable define:
340     name: b type: int
341 variable define:
342     name: b type: int
343 variable define:
344     name: c type: int
345 variable define:
346     name: c type: int
347 variable define:
348     name: d type: int
349 variable define:
350     name: e type: double
351 variable define:
352     name: f type: double
353 variable define:
354     name: g type: double
355 debug: hello

```

```

356 -----Print AST:-----
357 (compUnit (transUnit (decl (constDecl const (bType int) (constDef a = (constInitVal (constExp
(number 0)))) ;)) (decl (varDecl (bType int) (varDef b = (constInitVal (constExp (number 1)))) ;))
(funcDef (funcType int) func1 ( (funcFParams (funcFParam (bType int) a)) ) (block { (blockItem (decl
(varDecl (bType int) (varDef b = (constInitVal (constExp (number 3)))) ;)) (blockItem (decl (varDecl
(bType int) (varDef c [ 4 ] = (constInitVal { (constExp (number 0x1)) , (constExp (number 012)) ,
(constExp (number 0xF3)) , (constExp (number 4)) ;)) ;)) (blockItem (stmt return (exp (addExp
(addExp (addExp (mulExp (unaryExp (primaryExp (IVal a)))) + (mulExp (unaryExp (primaryExp (IVal
b)))) + (mulExp (unaryExp (primaryExp (IVal c [ (exp (addExp (mulExp (unaryExp (primaryExp
(number 2)))))) ;)) ;)) ;)) ;)) (funcDef (funcType bool) func2 ( (funcFParams (funcFParam (bType
double) a)) ) (block { (blockItem (stmt if ( (cond (lOrExp (lAndExp (eqExp (eqExp (relExp (addExp
(mulExp (unaryExp (primaryExp (IVal a)))))) == (relExp (addExp (mulExp (unaryExp (primaryExp
(number 0.0)))))) ;)) ;)) (stmt return (exp true) ;) else (stmt return (exp false) ;)) ;)) (funcDef (funcType
int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef c = (constInitVal (constExp (number
8)))) ;)) (blockItem (decl (varDecl (bType int) (varDef d = (constInitVal (constExp (number 9)))) ;))
(blockItem (stmt (IVal d) = (exp (addExp (mulExp (unaryExp func1 ( (funcRParams (exp (addExp
(mulExp (unaryExp (primaryExp (IVal c)))))) ;)) ;)) (blockItem (decl (varDecl (bType double) (varDef
e [ 8 ] = (constInitVal { ;)) ;)) (blockItem (decl (varDecl (bType double) (varDef f [ 8 ] = (constInitVal {
(constExp (number 1.0)) , (constExp (number 2.0)) , (constExp (number 4.5e-2)) , (constExp
(number 3E2)) , (constExp (number .5e3)) ;)) ;)) (blockItem (decl (varDecl (bType double) (varDef g
[ 8 ] ;)) (blockItem (stmt (IVal g) = (exp (addExp (addExp (mulExp (unaryExp (primaryExp (IVal e))))
+ (mulExp (unaryExp (primaryExp (IVal f)))) ;)) (blockItem (stmt if ( (cond (lOrExp (lAndExp (eqExp
(relExp (addExp (mulExp (unaryExp func2 ( (funcRParams (exp (addExp (mulExp (unaryExp
(primaryExp (IVal g [ (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))))) ;)) ;)) ;)) ;)) ;))
(stmt (exp (addExp (mulExp (unaryExp print_double ( (funcRParams (exp (addExp (mulExp
(unaryExp (primaryExp (IVal g [ (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))))) ;)) ;)) ;)) ;))
)))) ;) else (stmt (exp (addExp (mulExp (unaryExp print_bool ( (funcRParams (exp false)) ;)) ;)) ;))
(blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))) ;)) ;)) <EOF>)
358 -----
359 pass.
360
361
362
363
364
365
366 testing 19: 19_false_val_init.cact...
367 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, constInitVal, ]
368 line: 6:9 at b
369 variable define:
370     name: a type: int
371 const variable define:
372     line 2:0 name: b type: int
373 variable define:
374     name: c type: int
375 debug: hello
376 -----Print AST:-----

```



```

377 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 2)))) ;))
    (decl (constDecl const (bType int) (constDef b = (constInitVal (constExp (number 3)))) ;)) (funcDef
    (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef c = constInitVal) b ;)))
    (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal c)))))) ;)) ;)) <EOF>)
378 error level: 3
379 -----
380 pass.
381
382
383
384
385
386
387 testing 20: 20_false_val_init_op.cact...
388 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, ]
389 line: 3:14 at *
390 variable define:
391     name: a type: int
392 debug: hello
393 -----Print AST:-----
394 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
    (varDef a = (constInitVal (constExp (number 2)))) *))) (blockItem (stmt (exp (addExp (mulExp
    (unaryExp (primaryExp (number 3)))))) ;)) (blockItem (stmt return (exp (addExp (mulExp (unaryExp
    (primaryExp (IVal a)))))) ;)) ;)) <EOF>)
395 error level: 3
396 -----
397 pass.
398
399
400
401
402
403
404 testing 21: 21_false_token.cact...
405 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
406 line: 13:8 at a
407 variable define:
408     name: a type: int
409 variable define:
410     name: b type: int
411 variable define:
412     name: c type: int
413 debug: hello
414 -----Print AST:-----

```

```

415 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 2)))) ;))
    (funcDef (funcType int) foo ( (funcFParams (funcFParam (bType int) a) , (funcFParam (bType int) b [
    ])) ) (block { (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (IVal c)))))) ;)) ;))
    (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int) (varDef b [ 4 ])) ;))
    (blockItem (decl (varDecl (bType int) (varDef c) ;)) (blockItem (stmt (IVal c) = (exp (addExp (mulExp
    (unaryExp foo ( (funcRParams (exp (addExp (mulExp (unaryExp (primaryExp (IVal a)))))) , (exp
    (addExp (mulExp (unaryExp (primaryExp (IVal b)))))) ;)) (blockItem stmt) (blockItem (stmt
    return)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (IVal a)))))) ;)) ;)) <EOF>))
416 error level: 3
417 -----
418 pass.
419
420
421
422
423
424
425 testing 22: 22_false_global_array_size.cact...
426 rule stack: [compUnit, transUnit, decl, varDecl, varDef, ]
427 line: 2:6 at a
428 rule stack: [compUnit, transUnit, decl, varDecl, varDef, ]
429 line: 2:15 at 2
430 rule stack: [compUnit, transUnit, decl, varDecl, varDef, ]
431 line: 2:18 at 3
432 rule stack: [compUnit, transUnit, decl, varDecl, varDef, ]
433 line: 2:21 at 4
434 rule stack: [compUnit, transUnit, decl, varDecl, varDef, ]
435 line: 2:24 at 5
436 const variable define:
437     line 1:0 name: a type: int
438 variable define:
439     name: b type: int
440 debug: hello
441 -----Print AST:-----
442 (compUnit (transUnit (decl (constDecl const (bType int) (constDef a = (constInitVal (constExp
    (number 5)))) ;)) (decl (varDecl (bType int) (varDef b [ a ] = { 1 } , (varDef 2) , (varDef 3) , (varDef 4) ,
    (varDef 5 ) ;)) (funcDef (funcType int) main ( ) (block { (blockItem (stmt return (exp (addExp (mulExp
    (unaryExp (primaryExp (IVal a)))))) ;)) ;)) <EOF>))
443 error level: 3
444 -----
445 pass.
446
447
448
449
450
451
452 testing 23: 23_false_val_init_func.cact...
453 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, constInitVal, ]

```



```

454 line: 11:9 at foo
455 variable define:
456   name: a type: int
457 variable define:
458   name: b type: int
459 variable define:
460   name: c type: int
461 debug: hello
462 -----Print AST:-----
463 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 0)))) ;))
(funcDef (funcType int) foo ( (funcFParams (funcFParam (bType int) a)) ) (block { (blockItem (stmt
return (exp (addExp (mulExp (mulExp (unaryExp (primaryExp (IVal a)))) * (unaryExp (primaryExp
(IVal a)))))) ;)) ;)) (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
(varDef b = (constInitVal (constExp (number 2)))) ;)) (blockItem (decl (varDecl (bType int) (varDef c =
constInitVal) <missing ';'>)) (blockItem (stmt (exp (addExp (mulExp (unaryExp foo ( (funcRParams
(exp (addExp (mulExp (unaryExp (primaryExp (IVal a)))))) ;)) ;)) (blockItem (stmt return (exp
(addExp (mulExp (unaryExp (primaryExp (IVal c)))))) ;)) ;)) <EOF>)
464 error level: 3
465 -----
466 pass.
467
468
469
470
471
472
473 testing 24: 24_false_array_size_func.cact...
474 rule stack: [compUnit, transUnit, funcDef, block, blockItem, decl, varDecl, varDef, ]
475 line: 11:7 at foo
476 variable define:
477   name: a type: int
478 variable define:
479   name: b type: int
480 variable define:
481 debug: hello
482 -----Print AST:-----
483 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 0)))) ;))
(funcDef (funcType int) foo ( (funcFParams (funcFParam (bType int) a)) ) (block { (blockItem (stmt
return (exp (addExp (addExp (mulExp (unaryExp (primaryExp (IVal a)))) + (mulExp (unaryExp
(primaryExp (IVal a)))))) ;)) ;)) (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl
(bType int) (varDef b = (constInitVal (constExp (number 2)))) ;)) (blockItem (decl (varDecl (bType int)
varDef <missing ';'>)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (IVal c [ (exp
(addExp (mulExp (unaryExp foo ( (funcRParams (exp (addExp (mulExp (unaryExp (primaryExp (IVal
b)))))) ;)) ;)) ;)) ;)) ;)) ;)) ;)) ;)) ;)) ;)) ;)) <EOF>)
484 error level: 3
485 -----
486 pass.
487

```

```

488
489
490
491
492
493 testing 25: 25_false_continuous_equation.cact...
494 rule stack: [compUnit, transUnit, funcDef, block, blockItem, stmt, ]
495 line: 5:7 at =
496 variable define:
497     name: a type: int
498 variable define:
499     name: b type: int
500     name: c type: int
501 debug: hello
502 -----Print AST:-----
503 (compUnit (transUnit (funcDef (funcType int) main ( ) (block { (blockItem (decl (varDecl (bType int)
(varDef a = (constInitVal (constExp (number 3)))) ;)) (blockItem (decl (varDecl (bType int) (varDef b)
, (varDef c) ;)) (blockItem (stmt (lVal c) = (exp (addExp (mulExp (unaryExp (primaryExp (lVal b))))))
=)) (blockItem (stmt (exp (addExp (mulExp (unaryExp (primaryExp (lVal a)))))) ;)) (blockItem (stmt
return (exp (addExp (mulExp (unaryExp (primaryExp (lVal c)))))) ;)) ;)) <EOF>)
504 error level: 3
505 -----
506 pass.
507
508
509
510
511
512
513 testing 26: 26_false_global_exp.cact...
514 rule stack: [compUnit, ]
515 line: 4:0 at a
516 variable define:
517     name: a type: int
518 debug: hello
519 -----Print AST:-----
520 (compUnit (transUnit (decl (varDecl (bType int) (varDef a = (constInitVal (constExp (number 0)))) ;))
a = 7 / 2 ; int main ( ) { return 0 ; })
521 error level: 3
522 -----
523 pass.
524
525
526
527
528
529
530

```

通过了所有测试，表明我们的实现无误。

同时在实验中，我们还注意到：

1. `antlr4` 的文法规则支持并不完备。例如类似 $A \rightarrow A?B$ 的文法无法被处理（本质上是一个直接左递归，但没有被识别）。修改为 $A \rightarrow ABIB$ 即可。
2. 在设计双精度浮点数时，一开始设计的文法考虑并不周全：

```
1 DoubleConst
2   : DecimalFloatConst // support decimal only
3   ;
4
5 DecimalFloatConst
6   : FracConst ExpPart? // ! 没有涵盖所有情况
7   ;
8
9 FracConst
10  : DigitSeq? '.' DigitSeq
11  | '.' DigitSeq
12  ;
13
14 ExpPart
15  : 'e' ('+' | '-')? DigitSeq
16  | 'E' ('+' | '-')? DigitSeq
17  ;
18
19 DigitSeq
20  : Digit+
21  ;
```

这种文法无法识别 `3E2` 这样的浮点数。于是修改了规则：

```
1 DecimalFloatConst
2   : FracConst ExpPart?
3   | DigitSeq ExpPart
4   ;
```

这样就可以涵盖所有的情况。

2. 分成员总结

彭思叡：

本次实验，我根据讲义的文法实现了 `CACT.g4` 文件，并且补充了讲义中待补充的几条词法规则；查阅资料，模仿 *The Definitive ANTLR 4 Reference* 中 9.2 章的自定义错误程序实现了 `VerboseErrorListener` 类；实现了用于快速测试的脚本 `batch_test.sh`；尽管有了一些初步的使用经验，但我对 `ANTLR4` 接口的核心概念（诸如上下文 `antlr4::ParserRuleContext` 和 `listener` 模式）还是一知半解，之后的实验内容还需要进一步学习。

陈飞羽：

本次实验中，我主要帮助彭思睿同学检查、调试语法文档及代码。阅读实验讲义及参考书籍《The Definitive ANTLR 4 Reference》对于理解ANTLR4的代码有很大帮助。

- **CACT.g4** 文档的检查、调试：发现 **ANTLR4** 不能识别 **A→A?B** 类型的直接左递归的问题并调整 **.g4** 文档；发现 **SemanticAnalysis::exitVarDecl** 函数可能会访问空指针触发 **segment fault** 异常并修正。
- **ANTLR4** 代码阅读，修正初版代码的遗漏。
- 代码整合与版本管理

袁欣怡：

本次实验主要负责实验报告的撰写。通过此次实验，使得我对理论课词法分析、语法分析部分的知识有了更为深刻的认识，书本内容结合自己的思考和实践，让复杂的理论有了实际落地的感觉。另外，实验过程加深了我对 **Antlr** 的理解与掌握，并让我学习怎么借助 **Antlrworks** 进行可视化操作。实验过后，我能够更为自如地使用 **git** 进行版本管理与协作开发。

参考资料

1. Parser Rules: <https://github.com/antlr/antlr4/blob/master/doc/parser-rules.md>
2. Lexer Rules: <https://github.com/antlr/antlr4/blob/master/doc/lexer-rules.md>
3. ANTLR4文法文档的基本书写语法: <http://yijun1171.github.io/2015/03/30/antlr4%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0-%E8%AF%AD%E6%B3%95%E5%AD%97%E5%85%B8-Grammar-Lexicon/>
4. The Definitive ANTLR 4 Reference, 2nd Edition by Terence Parr, Chapter 9.2
5. grammar-v4: C: <https://github.com/antlr/grammars-v4/tree/master/c>
6. CACT specification
7. how-to-get-context-line-number-in-antlr-4-parser-rule: <https://stackoverflow.com/questions/48739334/how-to-get-context-line-number-in-antlr-4-parser-rule>)