词法分析结果

# 输入输出方式

代码注释中已经写的很明确，这里再强调一下：

## 用法：

progname [ filein [ fileout ] ]

上述标识符含义如下：

* progname：本程序
* filein：源程序文件名
* fileout = filein\_result.txt：输出结果文件名

说明：

* 如果没有filein，则一定没有fileout。
* 若没有filein，则从控制台输入源程序文件名。此处为了避免输出乱，没有提示。
* 若没有fileout，则输出结果文件名默认为filein\_result.txt。例如源程序文件名为filein.txt，则输出结果文件名为filein.txt\_result.txt。

## 输出

* 为了方便看结果，这里既在文件中输出，又在控制台输出，正常情况下，输出结果完全一致。
* 如果源程序文件打开失败，则会在控制台输出相应的错误提示，不创建输出结果文件。

# 一些实现上的说明

我的词法分析程序用了编译开关，本次提交的版本中有几处与我以后要用的程序是不一样的，我留出了这个空间，而且词法分析部分不需要再作修改，以后用时把本次的编译开关关掉，就可以直接是方便使用的形式。已经考虑到的并且已经在编译开关中实现好了的与以后要用的版本不一致之处有：

## GrammaAnalyzer类

目前只是建立起了GrammarAnalyzer（语法分析器）调用LexicalAnalyzer（词法分析器）这样一个逻辑关系，而本次提交的GrammarAnalyzer只是简单完成输出功能。具体的grammarAnalyze（语法分析函数）以及GrammaAnalyzer类的构造函数，以后会有版本的变化。

## nextSymbol函数

这是词法分析程序的核心函数，它的高效与否对整个语法分析过程影响巨大。比如对操作符的处理，由于操作符完全是一符一类，所以正常使用时没有必要保存操作符本身。但是本次提交的版本中，为了输出好看，加上了对操作符本身的输出。在以后的版本中，操作符本身不会保存。

## nextString函数

这次作业要求是用一个正确的程序进行测试，但是我考虑到了在词法分析这一环节出错的情况，而且根据我对本次作业要求的理解，对出错情况的处理是不同的。比如字符串，在本次作业中，不管遇到什么情况，都应该把读到的东西输出出来。于是就有了下面的区别：当在字符串中间遇到非法字符时，本次作业中是这样处理的：把目前读到的合法部分当作一个字符串来读，把非法字符后面的部分（含）直到双引号之前（不含）当作一般的symbol来读。

所以比如一个字符串这样写：

"name is 一, age 15 "

会有如下输出（设符号计数从1始）：

1 DOUBLE\_QUOTE "

2 STRING name is

3 ILLEGAL 一

4 COMMA ,

5 IDENFIER age

6 NUMBER 15

7 DOUBLE\_QUOTE "

其中第2行是调用nextString函数后返回的结果。

而在以后的程序中，由于要报错，那么语法分析程序关心的只是这个错误字符是什么，它可能是希望nextString函数直接返回第3行的结果，而不需要保留这个字符串的合法部分（比如可以简单粗暴地把含有非法字符的字符串解释成空串并继续编译。

在以后的版本中，nextString对错误字符的处理会如上所述。

## 补充说明

同字符串一样，在处理字符变量即用单引号括起来的部分时，也会有非法字符的问题，而且。本次提交的版本中，nextChar函数与以后要用的版本相同，故作“补充说明”。

如果一个字符常量有如下定义：

'ei5'

则输出为（设符号计数从1始）：

1 SINGLE\_QUOTE '

2 CHARACTER e

4 IDENTIFIER i5

5 SINGLE\_QUOTE '

这部分是语法分析做的事情，错误处理只是简单的找下一个单引号并输出中间部分。

# 其他考虑

这个版本的词法分析程序，不负责限制诸如标识符长度、整数取值范围等。因为文法中没有对这个作出限制，这些只能归结为是语义处理的结果。本次提交的版本中，这个考虑没有体现在代码中，因为我觉得有多种处理方式，作为作业，限制条件应该由作业要求提出；并且，本次作业要求中没有限制这些东西，我只能尽可能给出本次作业想到的输出结果。

比如对标识符长度的处理，可以采用只考虑前n个字符有效，忽略后面多出部分且不报错的方式（但是这种方式不好，容易让用户骂娘：他认为是两个标识符的东西，被编译器认为是一个，debug时非常蛋疼）；也可以采用报错的方式，提示用户换用短一些的标识符。

又如整数取值范围，它完全是语义的问题，但是如果交给语义分析来做的话，在实现上只能把数字字符串原样返回，让语义处理程序把它翻译为计算机认识的整数，这与教材中对词法分析程序的要求不符，教材中明确指出，词法分析程序要做的事之一就是把数值常量转化为数值返回给语法分析程序。但是如果完全按照教材所说，那么就需要让词法分析多做一些词法之外的事情，个人认为本次作业实现整数取值范围的检查有点过早。

所以，**本次提交的版本中，词法分析没有对溢出做任何检查，如果源程序代码中有过大的整数，输出结果可能与源程序中的数不一致，并且程序没有对这种情况采取任何措施。特此声明。**

# 一些感想

词法分析应该是编译程序七大逻辑模块中比较简单的一个，但是本次作业我花了很多精力来写，目的有四：

* 力图本次编写的代码以后直接可以用，对这些代码的修改删除越少越好，并且以后如果想，可以很方便地通过修改几个宏就回退到目前这个版本（讨厌在windows下用IDE + git）。
* 出于以后要用的考虑，力求节省开销，包括时间开销和空间开销。
* 力求代码风格的规范、漂亮、清晰、可读、易维护、易拓展。
* 通过编写这个编译器，熟悉C++这门神奇但是我以前从来没有用过的语言。

付出必有收获，虽然代码写的有点强迫症，但是看到这些**像艺术品一样美观的代码**，很有成就感。

# 测试程序词法分析输出结果（与预期完全一致，编码一次通过没有BUG）

1 CONST const

2 IDENTIFIER TEN

3 EQUAL =

4 NUMBER 10

5 COMMA ,

6 IDENTIFIER NINE

7 EQUAL =

8 PLUS +

9 NUMBER 9

10 COMMA ,

11 IDENTIFIER EIGHT1

12 EQUAL =

13 MINUS -

14 NUMBER 8

15 COMMA ,

16 IDENTIFIER A

17 EQUAL =

18 SINGLE\_QUOTE '

19 CHARATER A

20 SINGLE\_QUOTE '

21 COMMA ,

22 IDENTIFIER ZERO

23 EQUAL =

24 SINGLE\_QUOTE '

25 CHARATER 0

26 SINGLE\_QUOTE '

27 SEMICOLON ;

28 VAR var

29 IDENTIFIER int1

30 COMMA ,

31 IDENTIFIER int2

32 COLON :

33 INTEGER integer

34 SEMICOLON ;

35 IDENTIFIER ch1

36 COMMA ,

37 IDENTIFIER ch2

38 COLON :

39 CHAR char

40 SEMICOLON ;

41 IDENTIFIER intArray

42 COLON :

43 ARRAY array

44 LEFT\_SQUARE [

45 NUMBER 3

46 RIGHT\_SQUARE ]

47 OF of

48 INTEGER integer

49 SEMICOLON ;

50 IDENTIFIER chArray

51 COLON :

52 ARRAY array

53 LEFT\_SQUARE [

54 NUMBER 1

55 RIGHT\_SQUARE ]

56 OF of

57 CHAR char

58 SEMICOLON ;

59 PROCEDURE procedure

60 IDENTIFIER proc1

61 SEMICOLON ;

62 BEGIN begin

63 WRITE write

64 LEFT\_BRACKET (

65 DOUBLE\_QUOTE "

66 STRING proc1 start...

67 DOUBLE\_QUOTE "

68 RIGHT\_BRACKET )

69 SEMICOLON ;

70 IDENTIFIER int1

71 ASSIGN :=

72 MINUS -

73 LEFT\_BRACKET (

74 IDENTIFIER TEN

75 PLUS +

76 IDENTIFIER NINE

77 RIGHT\_BRACKET )

78 ASTERISK \*

79 IDENTIFIER EIGHT1

80 PLUS +

81 NUMBER 1024

82 SEMICOLON ;

83 IDENTIFIER ch1

84 ASSIGN :=

85 IDENTIFIER A

86 SEMICOLON ;

87 IDENTIFIER ch2

88 ASSIGN :=

89 IDENTIFIER ZERO

90 SEMICOLON ;

91 WRITE write

92 LEFT\_BRACKET (

93 DOUBLE\_QUOTE "

94 STRING int1:

95 DOUBLE\_QUOTE "

96 COMMA ,

97 IDENTIFIER int1

98 RIGHT\_BRACKET )

99 SEMICOLON ;

100 WRITE write

101 LEFT\_BRACKET (

102 DOUBLE\_QUOTE "

103 STRING ch1:

104 DOUBLE\_QUOTE "

105 COMMA ,

106 IDENTIFIER ch1

107 RIGHT\_BRACKET )

108 SEMICOLON ;

109 WRITE write

110 LEFT\_BRACKET (

111 DOUBLE\_QUOTE "

112 STRING ch2:

113 DOUBLE\_QUOTE "

114 COMMA ,

115 IDENTIFIER ch2

116 RIGHT\_BRACKET )

117 SEMICOLON ;

118 WRITE write

119 LEFT\_BRACKET (

120 DOUBLE\_QUOTE "

121 STRING chArray:

122 DOUBLE\_QUOTE "

123 RIGHT\_BRACKET )

124 SEMICOLON ;

125 FOR for

126 IDENTIFIER int2

127 ASSIGN :=

128 NUMBER 0

129 IDENTIFIER downto

130 NUMBER 0

131 DO do

132 BEGIN begin

133 IDENTIFIER chArray

134 LEFT\_SQUARE [

135 IDENTIFIER int2

136 RIGHT\_SQUARE ]

137 ASSIGN :=

138 IDENTIFIER ZERO

139 SEMICOLON ;

140 WRITE write

141 LEFT\_BRACKET (

142 IDENTIFIER chArray

143 LEFT\_SQUARE [

144 IDENTIFIER int2

145 RIGHT\_SQUARE ]

146 RIGHT\_BRACKET )

147 SEMICOLON ;

148 END end

149 SEMICOLON ;

150 WRITE write

151 LEFT\_BRACKET (

152 DOUBLE\_QUOTE "

153 STRING intArray:

154 DOUBLE\_QUOTE "

155 RIGHT\_BRACKET )

156 SEMICOLON ;

157 FOR for

158 IDENTIFIER int2

159 ASSIGN :=

160 NUMBER 0

161 IDENTIFIER to

162 NUMBER 2

163 DO do

164 BEGIN begin

165 IDENTIFIER intArray

166 LEFT\_SQUARE [

167 IDENTIFIER int2

168 RIGHT\_SQUARE ]

169 ASSIGN :=

170 IDENTIFIER int2

171 SEMICOLON ;

172 WRITE write

173 LEFT\_BRACKET (

174 IDENTIFIER intArray

175 LEFT\_SQUARE [

176 IDENTIFIER int2

177 RIGHT\_SQUARE ]

178 RIGHT\_BRACKET )

179 END end

180 SEMICOLON ;

181 BEGIN begin

182 WRITE write

183 LEFT\_BRACKET (

184 DOUBLE\_QUOTE "

185 STRING proc1 end.

186 DOUBLE\_QUOTE "

187 RIGHT\_BRACKET )

188 SEMICOLON ;

189 END end

190 END end

191 SEMICOLON ;

192 FUNCTION function

193 IDENTIFIER func1

194 LEFT\_BRACKET (

195 IDENTIFIER num

196 COLON :

197 INTEGER integer

198 RIGHT\_BRACKET )

199 COLON :

200 INTEGER integer

201 SEMICOLON ;

202 VAR var

203 IDENTIFIER steps

204 COLON :

205 INTEGER integer

206 SEMICOLON ;

207 PROCEDURE procedure

208 IDENTIFIER mov

209 LEFT\_BRACKET (

210 IDENTIFIER tstart

211 COMMA ,

212 IDENTIFIER tend

213 COMMA ,

214 IDENTIFIER tmid

215 COMMA ,

216 IDENTIFIER num

217 COLON :

218 INTEGER integer

219 SEMICOLON ;

220 VAR var

221 IDENTIFIER count

222 COLON :

223 INTEGER integer

224 RIGHT\_BRACKET )

225 SEMICOLON ;

226 BEGIN begin

227 IF if

228 IDENTIFIER num

229 GREATER >

230 NUMBER 0

231 BEGIN begin

232 IDENTIFIER mov

233 LEFT\_BRACKET (

234 IDENTIFIER tstart

235 COMMA ,

236 IDENTIFIER tmid

237 COMMA ,

238 IDENTIFIER tend

239 COMMA ,

240 IDENTIFIER num

241 MINUS -

242 NUMBER 1

243 COMMA ,

244 IDENTIFIER count

245 RIGHT\_BRACKET )

246 SEMICOLON ;

247 IDENTIFIER count

248 ASSIGN :=

249 IDENTIFIER count

250 PLUS +

251 NUMBER 1

252 SEMICOLON ;

253 WRITE write

254 LEFT\_BRACKET (

255 DOUBLE\_QUOTE "

256 STRING step

257 DOUBLE\_QUOTE "

258 COMMA ,

259 IDENTIFIER count

260 RIGHT\_BRACKET )

261 SEMICOLON ;

262 WRITE write

263 LEFT\_BRACKET (

264 DOUBLE\_QUOTE "

265 STRING from:

266 DOUBLE\_QUOTE "

267 COMMA ,

268 IDENTIFIER tstart

269 RIGHT\_BRACKET )

270 SEMICOLON ;

271 WRITE write

272 LEFT\_BRACKET (

273 DOUBLE\_QUOTE "

274 STRING to:

275 DOUBLE\_QUOTE "

276 COMMA ,

277 IDENTIFIER tend

278 RIGHT\_BRACKET )

279 SEMICOLON ;

280 IDENTIFIER mov

281 LEFT\_BRACKET (

282 IDENTIFIER tmid

283 COMMA ,

284 IDENTIFIER tend

285 COMMA ,

286 IDENTIFIER tstart

287 COMMA ,

288 IDENTIFIER num

289 MINUS -

290 NUMBER 1

291 COMMA ,

292 IDENTIFIER count

293 RIGHT\_BRACKET )

294 SEMICOLON ;

295 END end

296 END end

297 SEMICOLON ;

298 BEGIN begin

299 WRITE write

300 LEFT\_BRACKET (

301 DOUBLE\_QUOTE "

302 STRING func1 start...

303 DOUBLE\_QUOTE "

304 RIGHT\_BRACKET )

305 SEMICOLON ;

306 WRITE write

307 LEFT\_BRACKET (

308 DOUBLE\_QUOTE "

309 STRING argument is num: integer

310 DOUBLE\_QUOTE "

311 RIGHT\_BRACKET )

312 SEMICOLON ;

313 WRITE write

314 LEFT\_BRACKET (

315 IDENTIFIER num

316 RIGHT\_BRACKET )

317 SEMICOLON ;

318 WRITE write

319 LEFT\_BRACKET (

320 DOUBLE\_QUOTE "

321 STRING Hanoi Tower with levels:

322 DOUBLE\_QUOTE "

323 COMMA ,

324 IDENTIFIER num

325 RIGHT\_BRACKET )

326 SEMICOLON ;

327 IDENTIFIER steps

328 ASSIGN :=

329 NUMBER 0

330 SEMICOLON ;

331 IDENTIFIER steps

332 ASSIGN :=

333 IDENTIFIER mov

334 LEFT\_BRACKET (

335 NUMBER 1

336 COMMA ,

337 NUMBER 3

338 COMMA ,

339 NUMBER 2

340 COMMA ,

341 IDENTIFIER num

342 COMMA ,

343 IDENTIFIER steps

344 RIGHT\_BRACKET )

345 SEMICOLON ;

346 WRITE write

347 LEFT\_BRACKET (

348 DOUBLE\_QUOTE "

349 STRING The total number of steps is:

350 DOUBLE\_QUOTE "

351 COMMA ,

352 IDENTIFIER steps

353 RIGHT\_BRACKET )

354 SEMICOLON ;

355 WRITE write

356 LEFT\_BRACKET (

357 DOUBLE\_QUOTE "

358 STRING func1 end with:

359 DOUBLE\_QUOTE "

360 COMMA ,

361 IDENTIFIER steps

362 RIGHT\_BRACKET )

363 SEMICOLON ;

364 IDENTIFIER func1

365 ASSIGN :=

366 IDENTIFIER steps

367 SEMICOLON ;

368 END end

369 SEMICOLON ;

370 FUNCTION function

371 IDENTIFIER func2

372 COLON :

373 CHAR char

374 SEMICOLON ;

375 BEGIN begin

376 WRITE write

377 LEFT\_BRACKET (

378 DOUBLE\_QUOTE "

379 STRING func2 start...

380 DOUBLE\_QUOTE "

381 RIGHT\_BRACKET )

382 SEMICOLON ;

383 WRITE write

384 LEFT\_BRACKET (

385 DOUBLE\_QUOTE "

386 STRING Please input ch1, ch2, int1...

387 DOUBLE\_QUOTE "

388 RIGHT\_BRACKET )

389 SEMICOLON ;

390 READ read

391 LEFT\_BRACKET (

392 IDENTIFIER ch1

393 COMMA ,

394 IDENTIFIER ch2

395 RIGHT\_BRACKET )

396 SEMICOLON ;

397 READ read

398 LEFT\_BRACKET (

399 IDENTIFIER int1

400 RIGHT\_BRACKET )

401 SEMICOLON ;

402 IF if

403 IDENTIFIER ch1

404 LESS\_EQUAL <=

405 IDENTIFIER ch2

406 THEN then

407 IF if

408 IDENTIFIER ch1

409 LESS <

410 IDENTIFIER ch2

411 THEN then

412 IF if

413 IDENTIFIER int1

414 EQUAL =

415 NUMBER 0

416 THEN then

417 WRITE write

418 LEFT\_BRACKET (

419 DOUBLE\_QUOTE "

420 STRING <=

421 DOUBLE\_QUOTE "

422 RIGHT\_BRACKET )

423 ELSE else

424 WRITE write

425 LEFT\_BRACKET (

426 DOUBLE\_QUOTE "

427 STRING <

428 DOUBLE\_QUOTE "

429 RIGHT\_BRACKET )

430 ELSE else

431 IF if

432 IDENTIFIER int1

433 GREATER\_EQUAL >=

434 NUMBER 0

435 THEN then

436 IF if

437 IDENTIFIER int1

438 GREATER >

439 NUMBER 0

440 THEN then

441 WRITE write

442 LEFT\_BRACKET (

443 DOUBLE\_QUOTE "

444 STRING <>

445 DOUBLE\_QUOTE "

446 RIGHT\_BRACKET )

447 ELSE else

448 WRITE write

449 LEFT\_BRACKET (

450 DOUBLE\_QUOTE "

451 STRING =

452 DOUBLE\_QUOTE "

453 RIGHT\_BRACKET )

454 ELSE else

455 WRITE write

456 LEFT\_BRACKET (

457 DOUBLE\_QUOTE "

458 STRING <>

459 DOUBLE\_QUOTE "

460 RIGHT\_BRACKET )

461 ELSE else

462 IF if

463 IDENTIFIER int1

464 INEQUAL <>

465 NUMBER 0

466 THEN then

467 WRITE write

468 LEFT\_BRACKET (

469 DOUBLE\_QUOTE "

470 STRING >

471 DOUBLE\_QUOTE "

472 RIGHT\_BRACKET )

473 ELSE else

474 WRITE write

475 LEFT\_BRACKET (

476 DOUBLE\_QUOTE "

477 STRING >=

478 DOUBLE\_QUOTE "

479 RIGHT\_BRACKET )

480 SEMICOLON ;

481 WRITE write

482 LEFT\_BRACKET (

483 DOUBLE\_QUOTE "

484 STRING func2 end with:

485 DOUBLE\_QUOTE "

486 COMMA ,

487 IDENTIFIER ZERO

488 RIGHT\_BRACKET )

489 SEMICOLON ;

490 IDENTIFIER func2

491 ASSIGN :=

492 IDENTIFIER ZERO

493 SEMICOLON ;

494 END end

495 SEMICOLON ;

496 PROCEDURE procedure

497 IDENTIFIER proc2

498 LEFT\_BRACKET (

499 IDENTIFIER int1

500 COMMA ,

501 IDENTIFIER int2

502 COLON :

503 INTEGER integer

504 SEMICOLON ;

505 VAR var

506 IDENTIFIER int3

507 COLON :

508 INTEGER integer

509 RIGHT\_BRACKET )

510 SEMICOLON ;

511 BEGIN begin

512 WRITE write

513 LEFT\_BRACKET (

514 DOUBLE\_QUOTE "

515 STRING proc2 start...

516 DOUBLE\_QUOTE "

517 RIGHT\_BRACKET )

518 SEMICOLON ;

519 WRITE write

520 LEFT\_BRACKET (

521 DOUBLE\_QUOTE "

522 STRING arguments are int1, int2: integer; var int3: integer

523 DOUBLE\_QUOTE "

524 RIGHT\_BRACKET )

525 SEMICOLON ;

526 WRITE write

527 LEFT\_BRACKET (

528 IDENTIFIER int1

529 RIGHT\_BRACKET )

530 SEMICOLON ;

531 WRITE write

532 LEFT\_BRACKET (

533 IDENTIFIER int2

534 RIGHT\_BRACKET )

535 SEMICOLON ;

536 WRITE write

537 LEFT\_BRACKET (

538 IDENTIFIER int3

539 RIGHT\_BRACKET )

540 SEMICOLON ;

541 IDENTIFIER int3

542 ASSIGN :=

543 NUMBER 0

544 SEMICOLON ;

545 DO do

546 BEGIN begin

547 IDENTIFIER int3

548 ASSIGN :=

549 IDENTIFIER int3

550 PLUS +

551 IDENTIFIER int2

552 SEMICOLON ;

553 IDENTIFIER int1

554 ASSIGN :=

555 IDENTIFIER int1

556 PLUS +

557 NUMBER 1

558 SEMICOLON ;

559 END end

560 WHILE while

561 IDENTIFIER int1

562 LESS <

563 IDENTIFIER int2

564 SEMICOLON ;

565 WRITE write

566 LEFT\_BRACKET (

567 DOUBLE\_QUOTE "

568 STRING int3 now is :

569 DOUBLE\_QUOTE "

570 COMMA ,

571 IDENTIFIER int3

572 RIGHT\_BRACKET )

573 SEMICOLON ;

574 WRITE write

575 LEFT\_BRACKET (

576 DOUBLE\_QUOTE "

577 STRING proc2 end.

578 DOUBLE\_QUOTE "

579 RIGHT\_BRACKET )

580 SEMICOLON ;

581 END end

582 SEMICOLON ;

583 BEGIN begin

584 IDENTIFIER proc1

585 SEMICOLON ;

586 IDENTIFIER proc2

587 LEFT\_BRACKET (

588 IDENTIFIER int1

589 COMMA ,

590 IDENTIFIER int2

591 COMMA ,

592 IDENTIFIER intArray

593 LEFT\_SQUARE [

594 NUMBER 3

595 ASTERISK \*

596 NUMBER 3

597 MINUS -

598 NUMBER 2

599 ASTERISK \*

600 NUMBER 5

601 PLUS +

602 NUMBER 1

603 RIGHT\_SQUARE ]

604 RIGHT\_BRACKET )

605 SEMICOLON ;

606 IDENTIFIER int1

607 ASSIGN :=

608 IDENTIFIER func1

609 LEFT\_BRACKET (

610 NUMBER 4

611 RIGHT\_BRACKET )

612 SEMICOLON ;

613 IDENTIFIER ch1

614 ASSIGN :=

615 IDENTIFIER func2

616 SEMICOLON ;

617 END end

618 PERIOD .