### 词法分析结果

结果与预期相符，无问题。

具体结果如下：

1 CONSTSYM const

2 INTSYM int

3 ID constInt1

4 ASSIGN =

5 PLUS +

6 NUM 7

7 COMMA ,

8 ID constInt2

9 ASSIGN =

10 MINUS -

11 NUM 17

12 COMMA ,

13 ID constInt3

14 ASSIGN =

15 NUM 10

16 SEMICOLON ;

17 CONSTSYM const

18 CHARSYM char

19 ID constChar1

20 ASSIGN =

21 QUOTE '

22 CHAR a

23 QUOTE '

24 COMMA ,

25 ID constChar2

26 ASSIGN =

27 QUOTE '

28 CHAR -

29 QUOTE '

30 COMMA ,

31 ID constChar3

32 ASSIGN =

33 QUOTE '

34 CHAR /

35 QUOTE '

36 SEMICOLON ;

37 INTSYM int

38 ID variableInt1

39 COMMA ,

40 ID variableInt2

41 COMMA ,

42 ID variableIntArray

43 LBRACK [

44 NUM 10

45 RBRACK ]

46 SEMICOLON ;

47 CHARSYM char

48 ID variableChar1

49 COMMA ,

50 ID variableCharArray

51 LBRACK [

52 NUM 1

53 RBRACK ]

54 SEMICOLON ;

55 INTSYM int

56 ID funcReturnInt

57 LPARENT (

58 INTSYM int

59 ID a

60 COMMA ,

61 INTSYM int

62 ID b

63 RPARENT )

64 LBRACE {

65 INTSYM int

66 ID c

67 SEMICOLON ;

68 ID c

69 ASSIGN =

70 MINUS -

71 ID a

72 PLUS +

73 ID b

74 PLUS +

75 LPARENT (

76 NUM 1

77 PLUS +

78 NUM 7

79 RPARENT )

80 MULTI \*

81 NUM 2

82 SEMICOLON ;

83 RETURNSYM return

84 LPARENT (

85 ID c

86 RPARENT )

87 SEMICOLON ;

88 RBRACE }

89 CHARSYM char

90 ID funcReturnChar

91 LPARENT (

92 CHARSYM char

93 ID c

94 RPARENT )

95 LBRACE {

96 CHARSYM char

97 ID d

98 SEMICOLON ;

99 ID d

100 ASSIGN =

101 ID c

102 SEMICOLON ;

103 RETURNSYM return

104 LPARENT (

105 ID d

106 RPARENT )

107 SEMICOLON ;

108 RBRACE }

109 VOIDSYM void

110 ID funcRelation

111 LPARENT (

112 INTSYM int

113 ID n

114 RPARENT )

115 LBRACE {

116 IFSYM if

117 LPARENT (

118 ID n

119 LESS <

120 NUM 7

121 RPARENT )

122 LBRACE {

123 PRINTFSYM printf

124 LPARENT (

125 DOUQUOTE "

126 STRING <

127 DOUQUOTE "

128 RPARENT )

129 SEMICOLON ;

130 RETURNSYM return

131 SEMICOLON ;

132 RBRACE }

133 IFSYM if

134 LPARENT (

135 ID n

136 LESSEQU <=

137 NUM 7

138 RPARENT )

139 LBRACE {

140 PRINTFSYM printf

141 LPARENT (

142 DOUQUOTE "

143 STRING <=

144 DOUQUOTE "

145 RPARENT )

146 SEMICOLON ;

147 RETURNSYM return

148 SEMICOLON ;

149 RBRACE }

150 IFSYM if

151 LPARENT (

152 ID n

153 EQUAL ==

154 NUM 17

155 RPARENT )

156 LBRACE {

157 PRINTFSYM printf

158 LPARENT (

159 DOUQUOTE "

160 STRING ==

161 DOUQUOTE "

162 RPARENT )

163 SEMICOLON ;

164 RETURNSYM return

165 SEMICOLON ;

166 RBRACE }

167 IFSYM if

168 LPARENT (

169 ID n

170 GREATEQU >=

171 NUM 77

172 RPARENT )

173 LBRACE {

174 PRINTFSYM printf

175 LPARENT (

176 DOUQUOTE "

177 STRING >=

178 DOUQUOTE "

179 RPARENT )

180 SEMICOLON ;

181 RBRACE }

182 ELSESYM else

183 LBRACE {

184 PRINTFSYM printf

185 LPARENT (

186 DOUQUOTE "

187 STRING >

188 DOUQUOTE "

189 RPARENT )

190 SEMICOLON ;

191 RBRACE }

192 RBRACE }

193 VOIDSYM void

194 ID funcPrint

195 LPARENT (

196 RPARENT )

197 LBRACE {

198 PRINTFSYM printf

199 LPARENT (

200 DOUQUOTE "

201 STRING !#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}

202 DOUQUOTE "

203 RPARENT )

204 SEMICOLON ;

205 PRINTFSYM printf

206 LPARENT (

207 ID constInt1

208 PLUS +

209 ID constInt2

210 RPARENT )

211 SEMICOLON ;

212 PRINTFSYM printf

213 LPARENT (

214 DOUQUOTE "

215 STRING %c

216 DOUQUOTE "

217 COMMA ,

218 ID constChar1

219 RPARENT )

220 SEMICOLON ;

221 RBRACE }

222 VOIDSYM void

223 ID funcDoWhileAndFor

224 LPARENT (

225 RPARENT )

226 LBRACE {

227 INTSYM int

228 ID i

229 ASSIGN =

230 NUM 0

231 SEMICOLON ;

232 DOSYM do

233 LBRACE {

234 ID variableIntArray

235 LBRACK [

236 ID i

237 RBRACK ]

238 ASSIGN =

239 ID i

240 SEMICOLON ;

241 ID i

242 ASSIGN =

243 ID i

244 PLUS +

245 NUM 1

246 SEMICOLON ;

247 RBRACE }

248 WHILESYM while

249 LPARENT (

250 ID i

251 LESS <

252 ID constInt3

253 RPARENT )

254 SEMICOLON ;

255 FORSYM for

256 LPARENT (

257 ID i

258 ASSIGN =

259 NUM 0

260 SEMICOLON ;

261 ID i

262 LESS <

263 NUM 10

264 SEMICOLON ;

265 ID i

266 ASSIGN =

267 ID i

268 PLUS +

269 NUM 1

270 RPARENT )

271 LBRACE {

272 PRINTFSYM printf

273 LPARENT (

274 ID variableIntArray

275 LBRACK [

276 ID i

277 RBRACK ]

278 RPARENT )

279 SEMICOLON ;

280 RBRACE }

281 RBRACE }

282 INTSYM int

283 ID funcRecursion

284 LPARENT (

285 INTSYM int

286 ID n

287 RPARENT )

288 LBRACE {

289 IFSYM if

290 LPARENT (

291 ID n

292 LESSEQU <=

293 NUM 0

294 RPARENT )

295 LBRACE {

296 PRINTFSYM printf

297 LPARENT (

298 DOUQUOTE "

299 STRING n cannot be smaller or equal than 0!

300 DOUQUOTE "

301 RPARENT )

302 SEMICOLON ;

303 RETURNSYM return

304 LPARENT (

305 MINUS -

306 NUM 1

307 RPARENT )

308 SEMICOLON ;

309 RBRACE }

310 IFSYM if

311 LPARENT (

312 ID n

313 EQUAL ==

314 NUM 1

315 RPARENT )

316 RETURNSYM return

317 LPARENT (

318 NUM 1

319 RPARENT )

320 SEMICOLON ;

321 IFSYM if

322 LPARENT (

323 ID n

324 EQUAL ==

325 NUM 2

326 RPARENT )

327 RETURNSYM return

328 LPARENT (

329 NUM 1

330 RPARENT )

331 SEMICOLON ;

332 RETURNSYM return

333 LPARENT (

334 ID funcRecursion

335 LPARENT (

336 ID n

337 MINUS -

338 NUM 1

339 RPARENT )

340 PLUS +

341 ID funcRecursion

342 LPARENT (

343 ID n

344 MINUS -

345 NUM 2

346 RPARENT )

347 RPARENT )

348 SEMICOLON ;

349 RBRACE }

350 VOIDSYM void

351 MAINSYM main

352 LPARENT (

353 RPARENT )

354 LBRACE {

355 SCANFSYM scanf

356 LPARENT (

357 ID variableInt1

358 COMMA ,

359 ID variableInt2

360 COMMA ,

361 ID variableChar1

362 RPARENT )

363 SEMICOLON ;

364 ID variableCharArray

365 LBRACK [

366 NUM 0

367 RBRACK ]

368 ASSIGN =

369 QUOTE '

370 CHAR z

371 QUOTE '

372 SEMICOLON ;

373 INTSYM int

374 ID tempInt

375 SEMICOLON ;

376 CHARSYM char

377 ID tempChar

378 SEMICOLON ;

379 IFSYM if

380 LPARENT (

381 ID variableInt1

382 EQUAL ==

383 ID variableInt2

384 RPARENT )

385 LBRACE {

386 ID tempInt

387 ASSIGN =

388 ID funcReturnInt

389 LPARENT (

390 ID variableInt1

391 COMMA ,

392 ID variableInt1

393 RPARENT )

394 SEMICOLON ;

395 PRINTFSYM printf

396 LPARENT (

397 ID tempInt

398 RPARENT )

399 SEMICOLON ;

400 RBRACE }

401 IFSYM if

402 LPARENT (

403 ID variableInt1

404 GREAT >

405 ID variableInt2

406 RPARENT )

407 LBRACE {

408 ID tempChar

409 ASSIGN =

410 ID funcReturnChar

411 LPARENT (

412 ID variableChar1

413 RPARENT )

414 SEMICOLON ;

415 PRINTFSYM printf

416 LPARENT (

417 ID tempChar

418 RPARENT )

419 SEMICOLON ;

420 RBRACE }

421 IFSYM if

422 LPARENT (

423 ID variableChar1

424 EQUAL ==

425 ID constChar1

426 RPARENT )

427 LBRACE {

428 ID funcRelation

429 LPARENT (

430 ID variableInt1

431 RPARENT )

432 SEMICOLON ;

433 RBRACE }

434 IFSYM if

435 LPARENT (

436 ID variableChar1

437 EQUAL ==

438 ID constChar2

439 RPARENT )

440 LBRACE {

441 ID funcPrint

442 LPARENT (

443 RPARENT )

444 SEMICOLON ;

445 RBRACE }

446 IFSYM if

447 LPARENT (

448 ID variableChar1

449 EQUAL ==

450 ID constChar3

451 RPARENT )

452 LBRACE {

453 ID funcDoWhileAndFor

454 LPARENT (

455 RPARENT )

456 SEMICOLON ;

457 RBRACE }

458 IFSYM if

459 LPARENT (

460 ID variableChar1

461 EQUAL ==

462 ID variableCharArray

463 LBRACK [

464 NUM 0

465 RBRACK ]

466 RPARENT )

467 LBRACE {

468 PRINTFSYM printf

469 LPARENT (

470 ID funcRecursion

471 LPARENT (

472 ID variableInt1

473 RPARENT )

474 RPARENT )

475 SEMICOLON ;

476 RBRACE }

477 RBRACE }