输出结果正确

\*\*\*\*\*\*\*\*\*\*\*start\*\*\*\*\*\*\*\*\*\*\*

1 Const: const

2 Int: int

3 customObj: \_x

4 assign: =

5 aNum: 0

6 comma: ,

7 customObj: m

8 assign: =

9 aNum: 2

10 semicolon: ;

11 Const: const

12 Char: char

13 customObj: y

14 assign: =

15 aChar: b

16 semicolon: ;

17 Int: int

18 customObj: z

19 comma: ,

20 customObj: h

21 semicolon: ;

22 Char: char

23 customObj: m

24 comma: ,

25 customObj: i

26 semicolon: ;

27 Int: int

28 customObj: foo

29 lParent: (

30 Int: int

31 customObj: a

32 comma: ,

33 Int: int

34 customObj: b

35 rParent: )

36 lBrace: {

37 If: if

38 lParent: (

39 customObj: a

40 Big: >

41 customObj: b

42 rParent: )

43 Printf: printf

44 lParent: (

45 aString: big

46 rParent: )

47 semicolon: ;

48 If: if

49 lParent: (

50 customObj: a

51 small: <

52 customObj: b

53 rParent: )

54 Printf: printf

55 lParent: (

56 aString: less

57 rParent: )

58 semicolon: ;

59 If: if

60 lParent: (

61 customObj: a

62 bigAndEql: >=

63 customObj: b

64 rParent: )

65 Printf: printf

66 lParent: (

67 aString: big or equal

68 rParent: )

69 semicolon: ;

70 If: if

71 lParent: (

72 customObj: a

73 smallAndEql: <=

74 customObj: b

75 rParent: )

76 Printf: printf

77 lParent: (

78 aString: less or equal

79 rParent: )

80 semicolon: ;

81 If: if

82 lParent: (

83 customObj: a

84 notEql: !=

85 customObj: b

86 rParent: )

87 Printf: printf

88 lParent: (

89 aString: not equal

90 rParent: )

91 semicolon: ;

92 If: if

93 lParent: (

94 customObj: a

95 eql: ==

96 customObj: b

97 rParent: )

98 Printf: printf

99 lParent: (

100 aString: equal

101 rParent: )

102 semicolon: ;

103 customObj: return

104 lParent: (

105 aNum: 233

106 rParent: )

107 semicolon: ;

108 rBrace: }

109 Void: void

110 customObj: foo2

111 lParent: (

112 rParent: )

113 lBrace: {

114 Int: int

115 customObj: i

116 semicolon: ;

117 For: for

118 lParent: (

119 customObj: i

120 assign: =

121 aNum: 0

122 semicolon: ;

123 customObj: i

124 small: <

125 aNum: 10

126 semicolon: ;

127 customObj: i

128 assign: =

129 customObj: i

130 plus: +

131 aNum: 1

132 rParent: )

133 lBrace: {

134 Printf: printf

135 lParent: (

136 customObj: i

137 rParent: )

138 semicolon: ;

139 rBrace: }

140 Do: do

141 lBrace: {

142 Printf: printf

143 lParent: (

144 customObj: i

145 rParent: )

146 semicolon: ;

147 customObj: i

148 assign: =

149 customObj: i

150 sub: -

151 aNum: 1

152 semicolon: ;

153 rBrace: }

154 While: while

155 lParent: (

156 customObj: i

157 Big: >

158 aNum: 0

159 rParent: )

160 semicolon: ;

161 rBrace: }

162 Int: int

163 customObj: foo3

164 lParent: (

165 Int: int

166 customObj: i

167 rParent: )

168 lBrace: {

169 If: if

170 lParent: (

171 customObj: i

172 eql: ==

173 aNum: 10

174 rParent: )

175 customObj: return

176 lParent: (

177 customObj: i

178 rParent: )

179 semicolon: ;

180 customObj: return

181 lParent: (

182 customObj: i

183 plus: +

184 customObj: foo3

185 lParent: (

186 customObj: i

187 plus: +

188 aNum: 1

189 rParent: )

190 rParent: )

191 semicolon: ;

192 rBrace: }

193 Void: void

194 customObj: foo4

195 lParent: (

196 Int: int

197 customObj: a

198 comma: ,

199 Int: int

200 customObj: b

201 rParent: )

202 lBrace: {

203 Printf: printf

204 lParent: (

205 customObj: a

206 sub: -

207 customObj: b

208 rParent: )

209 semicolon: ;

210 Printf: printf

211 lParent: (

212 customObj: a

213 plus: +

214 customObj: b

215 rParent: )

216 semicolon: ;

217 Printf: printf

218 lParent: (

219 customObj: a

220 mul: \*

221 customObj: b

222 rParent: )

223 semicolon: ;

224 Printf: printf

225 lParent: (

226 customObj: a

227 div: /

228 customObj: b

229 rParent: )

230 semicolon: ;

231 rBrace: }

232 Char: char

233 customObj: foo5

234 lParent: (

235 rParent: )

236 lBrace: {

237 Char: char

238 customObj: a

239 semicolon: ;

240 Char: char

241 customObj: b

242 semicolon: ;

243 customObj: a

244 assign: =

245 aChar: a

246 semicolon: ;

247 customObj: b

248 assign: =

249 aChar: b

250 semicolon: ;

251 customObj: return

252 lParent: (

253 customObj: a

254 plus: +

255 customObj: b

256 rParent: )

257 semicolon: ;

258 rBrace: }

259 Int: int

260 customObj: foo6

261 lParent: (

262 rParent: )

263 lBrace: {

264 Int: int

265 customObj: a

266 lBracket: [

267 aNum: 2

268 rBracket: ]

269 semicolon: ;

270 customObj: a

271 lBracket: [

272 aNum: 0

273 rBracket: ]

274 assign: =

275 aNum: 1

276 semicolon: ;

277 customObj: a

278 lBracket: [

279 aNum: 1

280 rBracket: ]

281 assign: =

282 aNum: 2

283 semicolon: ;

284 customObj: a

285 lBracket: [

286 aNum: 2

287 rBracket: ]

288 assign: =

289 customObj: a

290 lBracket: [

291 aNum: 0

292 rBracket: ]

293 mul: \*

294 lParent: (

295 customObj: a

296 lBracket: [

297 aNum: 1

298 rBracket: ]

299 plus: +

300 customObj: foo5

301 lParent: (

302 rParent: )

303 rParent: )

304 div: /

305 customObj: foo

306 lParent: (

307 aNum: 1

308 comma: ,

309 aNum: 2

310 rParent: )

311 semicolon: ;

312 customObj: return

313 lParent: (

314 customObj: a

315 lBracket: [

316 aNum: 2

317 rBracket: ]

318 rParent: )

319 semicolon: ;

320 rBrace: }

321 Void: void

322 customObj: main

323 lParent: (

324 rParent: )

325 lBrace: {

326 Int: int

327 customObj: a

328 semicolon: ;

329 Char: char

330 customObj: b

331 semicolon: ;

332 Scanf: scanf

333 lParent: (

334 customObj: b

335 rParent: )

336 semicolon: ;

337 customObj: a

338 assign: =

339 customObj: foo

340 lParent: (

341 aNum: 2

342 comma: ,

343 aNum: 3

344 rParent: )

345 semicolon: ;

346 customObj: foo2

347 lParent: (

348 rParent: )

349 semicolon: ;

350 customObj: foo3

351 lParent: (

352 aNum: 1

353 rParent: )

354 semicolon: ;

355 customObj: foo4

356 lParent: (

357 aNum: 1

358 comma: ,

359 aNum: 2

360 rParent: )

361 semicolon: ;

362 customObj: foo6

363 lParent: (

364 rParent: )

365 semicolon: ;

366 customObj: return

367 semicolon: ;

368 rBrace: }

\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*