词法分析结果

题目要求:

> 测试结果文件，说明上述测试程序产生的输出结果，并说明该结果是否与预期的相符，如果不相符，存在什么问题。

我的测试程序主要是实现了计算组合数这样的一个功能。主要写了三个函数：

```
int factorial(int n) 递归求解n的阶乘
int C(int m, int n) 用公式计算组合数C(n,m)
int mymax(int x, int y) 计算x和y中的较大值
```

生成的规则如下，每一行都首先有一个数，代表是第几个生成的词法分析程序，然后后面跟着的是词法分析出的单词的类型，如Keyword(代表保留字)、Identifier(代表标识符)、然后一些符号则给出了其英文对应比如逗号是comma等等。然后针对没有出现某个单词经过词法分析但没有判断的情况，我会输出`!!!WRONG!!!`，针对其它异常情况，我会输出`!!!NotFound!!!`

具体代码见文件，然后生成后的文件如下。可以看到并没有出现!!!，代表没有出错，经逐个分析也可以看到是符合预期的正确结果。

```
1 KeyWord const
2 KeyWord int
3 Identifier N
4 assign =
5 ConstantInt 100
6 Comma ,
7 Identifier M
8 assign =
9 sub -
10 sub -
11 ConstantInt 10000
12 Semicolon ;
13 KeyWord const
14 KeyWord int
15 Identifier K
16 assign =
17 ConstantInt 0
18 Semicolon ;
19 KeyWord const
20 KeyWord char
21 Identifier ch1
```

```
22 assign =
23 ConstantChar '+'
24 Comma ,
25 Identifier ch2
26 assign =
27 ConstantChar '-'
28 Comma ,
29 Identifier ch3
30 assign =
31 ConstantChar '\*'
32 Comma ,
33 Identifier ch4
34 assign =
35 ConstantChar '/'
36 Comma ,
37 Identifier ch5
38 assign =
39 ConstantChar '_'
40 Comma ,
41 Identifier ch6
42 assign =
43 ConstantChar 'a'
44 Comma ,
45 Identifier ch7
46 assign =
47 ConstantChar '0'
48 Comma ,
49 Identifier ch8
50 assign =
51 ConstantChar '"'
52 Comma ,
53 Identifier ch9
54 assign =
55 ConstantChar '9'
56 Semicolon ;
57 KeyWord int
58 Identifier a
59 LeftBracket [
60 ConstantInt 100
61 RightBracket ]
62 Comma ,
63 Identifier i
64 Comma ,
65 Identifier j
66 Comma ,
67 Identifier n
68 Comma ,
69 Identifier m
```

```
70 Comma ,
71 Identifier k
72 Comma ,
73 Identifier x
74 Comma ,
75 Identifier y
76 Comma ,
77 Identifier z
78 Comma ,
79 Identifier mx
80 Semicolon ;
81 KeyWord char
82 Identifier ch
83 Comma ,
84 Identifier s
85 LeftBracket [
86 ConstantInt 10000
87 RightBracket ]
88 Semicolon ;
89 KeyWord int
90 Identifier mymax
91 LeftParentheis (
92 KeyWord int
93 Identifier x
94 Comma ,
95 KeyWord int
96 Identifier y
97 RightParenthesis )
98 LeftBrace {
99 KeyWord int
100 Identifier ans
101 Semicolon ;
102 Identifier ans
103 assign =
104 Identifier x
105 Semicolon ;
106 KeyWord if
107 LeftParentheis (
108 Identifier y
109 bgt >
110 Identifier x
111 RightParenthesis )
112 LeftBrace {
113 Identifier ans
114 assign =
115 Identifier y
116 Semicolon ;
117 RightBrace }
```

```
118 KeyWord else
119 Semicolon ;
120 KeyWord return
121 LeftParentheis (
122 Identifier ans
123 RightParenthesis )
124 Semicolon ;
125 RightBrace }
126 KeyWord int
127 Identifier factorial
128 LeftParentheis (
129 KeyWord int
130 Identifier n
131 RightParenthesis )
132 LeftBrace {
133 div /
134 div /
135 Identifier calc
136 Identifier n
137 !!!NotFound!!! !
138 KeyWord int
139 Identifier res
140 assign =
141 ConstantInt 1
142 Semicolon ;
143 KeyWord if
144 LeftParentheis (
145 Identifier n
146 blt <
147 ConstantInt 0
148 RightParenthesis )
149 KeyWord return
150 LeftParentheis (
151 sub -
152 ConstantInt 1
153 RightParenthesis )
154 Semicolon ;
155 KeyWord switch
156 LeftParentheis (
157 Identifier n
158 RightParenthesis )
159 LeftBrace {
160 KeyWord case
161 ConstantInt 0
162 colon :
163 Identifier res
164 assign =
165 ConstantInt 1
```

```
166 Semicolon ;
167 KeyWord case
168 ConstantInt 1
169 colon :
170 Identifier res
171 assign =
172 ConstantInt 1
173 Semicolon ;
174 Identifier default
175 colon :
176 Identifier res
177 assign =
178 LeftParentheis (
179 Identifier n
180 mul *
181 Identifier factorial
182 LeftParentheis (
183 Identifier n
184 sub -
185 ConstantInt 1
186 RightParenthesis )
187 RightParenthesis )
188 Semicolon ;
189 RightBrace }
190 KeyWord return
191 LeftParentheis (
192 Identifier res
193 RightParenthesis )
194 Semicolon ;
195 RightBrace }
196 KeyWord int
197 Identifier C
198 LeftParentheis (
199 KeyWord int
200 Identifier m
201 Comma ,
202 KeyWord int
203 Identifier n
204 RightParenthesis )
205 LeftBrace {
206 div /
207 div /
208 Identifier calc
209 Identifier C
210 LeftParentheis (
211 Identifier n
212 Comma ,
213 Identifier m
```

```
214 RightParenthesis )
215 KeyWord if
216 LeftParentheis (
217 Identifier m
218 blt <=
219 Identifier n
220 RightParenthesis )
221 Semicolon ;
222 div /
223 div /
224 KeyWord for
225 Identifier test
226 KeyWord if
227 LeftParentheis (
228 Identifier m
229 bgt >
230 Identifier n
231 RightParenthesis )
232 KeyWord return
233 LeftParentheis (
234 sub -
235 ConstantInt 1
236 RightParenthesis )
237 Semicolon ;
238 KeyWord return
239 LeftParentheis (
240 Identifier factorial
241 LeftParentheis (
242 Identifier n
243 RightParenthesis )
244 div /
245 Identifier factorial
246 LeftParentheis (
247 Identifier m
248 RightParenthesis )
249 div /
250 Identifier factorial
251 LeftParentheis (
252 Identifier n
253 sub -
254 Identifier m
255 RightParenthesis )
256 RightParenthesis )
257 Semicolon ;
258 RightBrace }
259 KeyWord void
260 Identifier init
261 LeftParentheis (
```

```
262 RightParenthesis )
263 LeftBrace {
264 div /
265 div /
266 Identifier input
267 Identifier n
268 Comma ,
269 Identifier m
270 Comma ,
271 Identifier k
272 Identifier m
273 assign =
274 ConstantInt 5
275 Semicolon ;
276 Identifier scanf
277 LeftParentheis (
278 Identifier n
279 Comma ,
280 Identifier m
281 Comma ,
282 Identifier k
283 Comma ,
284 Identifier ch
285 RightParenthesis )
286 Semicolon ;
287 KeyWord return
288 Semicolon ;
289 RightBrace }
290 KeyWord char
291 Identifier work
292 LeftParentheis (
293 KeyWord int
294 Identifier m
295 Comma ,
296 KeyWord int
297 Identifier n
298 RightParenthesis )
299 LeftBrace {
300 KeyWord int
301 Identifier ans
302 assign =
303 Identifier C
304 LeftParentheis (
305 Identifier m
306 Comma ,
307 Identifier n
308 RightParenthesis )
309 Semicolon ;
```

```
310 KeyWord switch
311 LeftParentheis (
312 Identifier ans
313 sub -
314 ConstantInt 2
315 mul *
316 LeftParentheis (
317 Identifier ans
318 div /
319 ConstantInt 2
320 RightParenthesis )
321 RightParenthesis )
322 LeftBrace {
323 div /
324 div /
325 Identifier ans
326 !!!WRONG!!! %
327 ConstantInt 2
328 KeyWord case
329 ConstantInt 1
330 colon :
331 KeyWord return
332 ConstantChar 'O'
333 Semicolon ;
334 div /
335 div /
336 Identifier odd
337 Identifier default
338 colon :
339 KeyWord return
340 ConstantChar 'E'
341 Semicolon ;
342 div /
343 div /
344 Identifier even
345 RightBrace }
346 KeyWord return
347 LeftParentheis (
348 ConstantChar '*'
349 RightParenthesis )
350 Semicolon ;
351 RightBrace }
352 KeyWord void
353 KeyWord main
354 LeftParentheis (
355 RightParenthesis )
356 LeftBrace {
357 Identifier x
```

```
358 assign =
359 Identifier mymax
360 LeftParentheis (
361 ConstantInt 1
362 Comma ,
363 sub -
364 sub -
365 ConstantInt 2
366 RightParenthesis )
367 Semicolon ;
368 div /
369 div /
370 sub -
371 sub -
372 ConstantInt 2
373 assign =
374 ConstantInt 2
375 Identifier x
376 assign =
377 Identifier mymax
378 LeftParentheis (
379 Identifier x
380 add +
381 add +
382 ConstantInt 8
383 Comma ,
384 ConstantInt 9
385 RightParenthesis )
386 Semicolon ;
387 div /
388 div /
389 Identifier x
390 add +
391 LeftParentheis (
392 add +
393 ConstantInt 8
394 RightParenthesis )
395 Comma ,
396 Identifier x
397 assign =
398 ConstantInt 10
399 Identifier y
400 assign =
401 ConstantInt 4
402 Semicolon ;
403 Identifier z
404 assign =
405 ConstantInt 7
```

```
406 Semicolon ;
407 KeyWord if
408 LeftParentheis (
409 Identifier x
410 bgt >
411 Identifier y
412 RightParenthesis )
413 LeftBrace {
414 KeyWord if
415 LeftParentheis (
416 Identifier x
417 bgt >
418 Identifier z
419 RightParenthesis )
420 Identifier mx
421 assign =
422 Identifier x
423 Semicolon ;
424 KeyWord else
425 Identifier mx
426 assign =
427 Identifier z
428 Semicolon ;
429 RightBrace }
430 KeyWord else
431 KeyWord if
432 LeftParentheis (
433 Identifier y
434 bgt >
435 Identifier z
436 RightParenthesis )
437 Identifier mx
438 assign =
439 Identifier y
440 Semicolon ;
441 KeyWord else
442 Identifier mx
443 assign =
444 Identifier z
445 Semicolon ;
446 Identifier printf
447 LeftParentheis (
448 ConstantString "mx = "
449 Comma ,
450 Identifier mx
451 RightParenthesis )
452 Semicolon ;
453 Identifier init
```

```
454 LeftParentheis (
455 RightParenthesis )
456 Semicolon ;
457 div /
458 div /
459 Identifier input
460 Identifier n
461 Comma ,
462 Identifier m
463 Comma ,
464 Identifier k
465 Identifier and
466 Identifier ch
467 KeyWord for
468 LeftParentheis (
469 Identifier i
470 assign =
471 Identifier k
472 add +
473 ConstantInt 1
474 Semicolon ;
475 Identifier i
476 bge >=
477 ConstantInt 0
478 Semicolon ;
479 Identifier i
480 assign =
481 Identifier i
482 sub -
483 ConstantInt 1
484 RightParenthesis )
485 LeftBrace {
486 Identifier a
487 LeftBracket [
488 Identifier i
489 RightBracket ]
490 assign =
491 Identifier k
492 add +
493 Identifier i
494 Semicolon ;
495 Identifier s
496 LeftBracket [
497 Identifier i
498 RightBracket ]
499 assign =
500 Identifier ch
501 add +
```

```
502 Identifier i
503 Semicolon ;
504 RightBrace }
505 KeyWord for
506 LeftParentheis (
507 Identifier i
508 assign =
509 ConstantInt 0
510 Semicolon ;
511 Identifier i
512 bgt >
513 Identifier k
514 Semicolon ;
515 Identifier i
516 assign =
517 Identifier i
518 sub -
519 ConstantInt 1
520 RightParenthesis )
521 LeftBrace {
522 div /
523 div /
524 KeyWord for
525 Identifier test
526 Identifier k
527 assign =
528 Identifier k
529 add +
530 ConstantInt 1
531 Semicolon ;
532 RightBrace }
533 Identifier printf
534 LeftParentheis (
535 ConstantString "k = "
536 Comma ,
537 Identifier k
538 RightParenthesis )
539 Semicolon ;
540 Identifier printf
541 LeftParentheis (
542 ConstantString "ch = "
543 Comma ,
544 Identifier s
545 LeftBracket [
546 Identifier k
547 RightBracket ]
548 RightParenthesis )
549 Semicolon ;
```

```
550 div /
551 div /
552 Identifier print
553 Identifier the
554 KeyWord char
555 Identifier that
556 Identifier is
557 Identifier k
558 Identifier behind
559 Identifier ch
560 KeyWord for
561 LeftParentheis (
562 Identifier i
563 assign =
564 ConstantInt 0
565 Semicolon ;
566 Identifier i
567 blt <=
568 Identifier m
569 Semicolon ;
570 Identifier i
571 assign =
572 Identifier i
573 add +
574 ConstantInt 1
575 RightParenthesis )
576 LeftBrace {
577 Identifier j
578 assign =
579 Identifier m
580 sub -
581 Identifier i
582 Semicolon ;
583 KeyWord if
584 LeftParentheis (
585 Identifier i
586 assign =
587 ConstantInt 0
588 RightParenthesis )
589 LeftBrace {
590 Identifier printf
591 LeftParentheis (
592 Identifier work
593 LeftParentheis (
594 Identifier j
595 Comma ,
596 Identifier n
597 RightParenthesis )
```

```
598 RightParenthesis )
599 Semicolon ;
600 div /
601 div /
602 Identifier print
603 Identifier C
604 LeftParentheis (
605 Identifier n
606 Comma ,
607 Identifier j
608 RightParenthesis )
609 RightBrace }
610 Identifier printf
611 LeftParentheis (
612 ConstantString "N    "
613 Comma ,
614 Identifier n
615 RightParenthesis )
616 Semicolon ;
617 Identifier printf
618 LeftParentheis (
619 ConstantString "M    "
620 Comma ,
621 Identifier j
622 RightParenthesis )
623 Semicolon ;
624 Identifier printf
625 LeftParentheis (
626 ConstantString "C    "
627 Comma ,
628 Identifier C
629 LeftParentheis (
630 Identifier j
631 Comma ,
632 Identifier n
633 RightParenthesis )
634 RightParenthesis )
635 Semicolon ;
636 Identifier printf
637 LeftParentheis (
638 ConstantString ""
639 RightParenthesis )
640 Semicolon ;
641 RightBrace }
642 KeyWord return
643 LeftParentheis (
644 ConstantInt 0
645 RightParenthesis )
```

```
646 Semicolon ;
647 RightBrace }
```