# 目录

[1 目录 1](#_Toc513585099)

[2 词法分析单元测试 1](#_Toc513585100)

[2.1 测试环境 1](#_Toc513585101)

[2.1.1 依赖 1](#_Toc513585102)

[2.1.2 调试输出 3](#_Toc513585103)

[2.1.3 获取可执行文件 3](#_Toc513585104)

[2.1.4 输入文件 3](#_Toc513585105)

[2.1.5 输出文件 3](#_Toc513585106)

[2.2 测试计划 4](#_Toc513585107)

[2.3 错误处理单个测试 4](#_Toc513585108)

[2.3.1 行长度超过限制 4](#_Toc513585109)

[2.3.2 标识符长度超过限制 7](#_Toc513585110)

[2.3.3 非法字符 8](#_Toc513585111)

[2.3.4 读取字符常量时遇到文件尾 9](#_Toc513585112)

[2.3.5 读取的字符常量为空 10](#_Toc513585113)

[2.3.6 读取的字符常量不止一个字符 11](#_Toc513585114)

[2.3.7 读取字符常量先遇到换行符而不是单引号 12](#_Toc513585115)

[2.3.8 读取多行注释时遇到文件尾 13](#_Toc513585116)

[2.4 错误处理组合测试 14](#_Toc513585117)

[2.4.1 错误组合测试1 14](#_Toc513585118)

[2.4.2 错误组合测试2 15](#_Toc513585119)

[2.4.3 错误组合测试3 16](#_Toc513585120)

[2.5 综合测试 17](#_Toc513585121)

# 词法分析单元测试

## 测试环境

### 调试输出

如果定义了宏LEXDEBUG，那么就可以输出词法分析程序的调试输出，即输出按顺序输出每一个记号及其属性。具体做法是，在每一个正则表达式的动作中添加如下代码（以标识符为例）

1. #ifdef LEXDEBUG
2. cout << "identifier: " << yylval->str << endl;
3. #endif

那么只要在一开始定义宏LEXDEBUG，就可以产生调试输出

另外，一旦遇到词法错误，也会输出词法错误的详细信息。

### 获取可执行文件

采用如下命令

### 输入文件

Test.pas

### 输出文件

LexOut.txt

## 测试计划

* 有针对性的测试每一种词法错误
* 组合上述词法错误，词法分析程序能否成功处理错误并恢复（有一些词法错误，例如行长度超过限制，会导致词法分析程序直接停止运行）
* 以一个较复杂的，没有错误的程序作为最终测试

## 错误处理单个测试

### 行长度超过限制

* 测试用例
* 预期结果

报行长度超过10000限制的错误，词法分析程序停止运行

* 测试结果
* 分析

报错结果和对策符合预期，词法分析程序停止运行。

### 标识符长度超过限制

* 测试用例

1. program test(input,output);
2. var asadhsadsadsadaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa:integer;
3. begin
5. end.

* 预期结果

报标识符长度超过限制100的错误；保留前100个字符作为标识符记号输出

* 测试结果
* 分析

报错结果和对策符合预期，保留了前100个字符作为当前识别的标识符，词法分析程序继续运行。

### 非法字符

* 测试用例

1. program test(input,output);
2. var a:integer;
3. begin?
4. $$
5. end.##

* 预期结果

非法字符?出现一次，$出现两次，#出现两次，所以共报五次非法字符的错误，并忽略这些非法字符

* 测试结果
* 分析

报错结果和对策符合预期，词法分析程序没有受到非法字符的影响。

### 读取字符常量时遇到文件尾

* 测试用例

1. program test(input,output);
2. **const** a='

* 预期结果

报读取常量时遇到文件尾的错误

* 测试结果
* 分析

报错结果和对策符合预期。

### 读取的字符常量为空

* 测试用例

1. program test(input,output);
2. **const** a=''
3. begin
5. end.

* 预期结果

报读取的字符常量为空的错误，并提交”\0”，作为当前识别的字符常量

* 测试结果
* 分析

报错结果和对策符合预期，提交了”\0”作为当前识别的字符常量，词法分析程序继续运行。

### 读取的字符常量不止一个字符

* 测试用例

1. program test(input,output);
2. **const** a='abc';
3. begin
4. writeln(a);
5. end.

* 预期结果

报读取的字符常量不止一个字符的错误，并提交第一个字符作为当前识别的字符常量

* 测试结果
* 分析

报错结果和对策符合预期，保留了第一个字符作为当前识别的字符常量，词法分析程序继续运行。

### 读取字符常量先遇到换行符而不是单引号

* 测试用例

1. program test(input,output);
2. **const** a='a
3. b='
4. c='cba
5. begin
7. end.

* 预期结果

报三次单引号缺失的错误，第一次提交”a”作为当前识别的字符常量，第二次提交”\0”作为当前识别的字符常量，第三次提交”c”作为当前识别的字符常量

* 测试结果
* 分析

报错结果和对策符合预期，三种情况都做出了相应的提交策略，且词法分析程序都能继续运行。

### 读取多行注释时遇到文件尾

* 测试用例

1. program test(input,output);
2. {
3. begin
5. end.

* 预期结果

报读取多行注释时遇到文件尾的错误（右花括号缺失）

* 测试结果
* 分析

报错结果和对策符合预期。

## 错误处理组合测试

### 错误组合测试1

* 测试用例

1. program test(input,output);
2. var asadhsadsadsadaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa:integer;
3. #
4. begin
6. end.

* 预期结果

报标识符长度超过限制100的错误，报非法字符的错误

* 测试结果
* 分析

报错结果和对策符合预期。

### 错误组合测试2

* 测试用例

1. program test(input,output);
2. **const** a=''
3. b='abc

* 预期结果

报读取的字符常量为空的错误，报读取字符常量时遇到文件尾的错误

* 测试结果
* 分析

报错结果和对策符合预期。

### 错误组合测试3

* 测试用例

1. program test(input,output);
2. **const** a='abc'
3. b='bac
4. begin
5. {123
6. end.

* 预期结果

报读取的字符常量不止一个字符的错误，报右单引号缺失的错误，报多行注释遇到文件尾的错误

* 测试结果
* 分析

报错结果和对策符合预期。

## 综合测试

该测试样例不包含错误，实现了快速排序的功能。

* 测试用例

1. program quicksort(input,output);
2. var
3. n,i:integer;
4. a:array[0..100000] of integer;
5. b:**char**;
7. procedure kp(l,r:integer);
8. var
9. i,j,mid:integer;
10. begin
11. b:='a';
12. **if** l>=r then exit;
13. i:=l;j:=r;mid:=a[(l+r) div 2];
14. repeat
15. begin
16. **while** a[i]<mid **do** inc(i);
17. **while** a[j]>mid **do** dec(j);
18. **if** i<=j then
19. begin
20. a[0]:=a[i];a[i]:=a[j];a[j]:=a[0];
21. inc(i);dec(j);
22. end
23. end
24. until i>j;
25. kp l,j);
26. kp(i,r)
27. end;
29. begin
30. readln(n);
31. **for** i:=1 to n **do**
32. read(a[i]);
33. kp(1,n);
34. **for** i:=1 to n **do**
35. write(a[i]);
36. end.

* 预期结果

记号流，不包含任何报错信息

* 测试结果
* 分析

记号流输出与源程序相符合，不包含任何错误信息。