

测试报告

1 运行结果及分析

部分分析结果截图：

MainWindow

文件 XC++

D:/大学文件/课程文件/3_大三/编译原理/2023 编译原理/课程实验/实验1/实验1 测试数据.cpp

打开文件 保存分析结果

词法分析 更改风格

源代码:

```
1 // test1
2 /* test */
3
4 void char int float double short long signed unsigned
5 struct union enum typedef sizeof auto static register extern const volatile return
6 continue break goto if else switch case default for do while alignas alignof and
7 and_eq asm atomic_cancel atomic_commit atomic_noexcept bitand bitor bool catch
8 char8_t char16_t char32_t class compl concept consteval constexpr constinit const_cast co_await
9 co_return co_yield decltype delete dynamic_cast explicit export false friend inline
10 mutable namespace new noexcept not not_eq nullptr operator or or_eq private
11 protected public reflexpr reinterpret_cast requires static_assert static_cast synchronized template
12 this thread_local throw true try typeid typename using virtual wchar_t xor + - * / %
13 ++ -- := > <= < && || ! & | ^ ~ << >> = != == * = /= %= <<=
14 >>= &= ^= |= , -> # include iostream std ( ) [ ] { } ; : ? ::
15
16 x_ y abc123 xy _xy xy_xy23
17
18 123L 3.45 0.123 12.34 3.45e+3 345 3.45e3 3.45e-3
19
20 "This is a string"
21 'c'
22
23 void main()
24 {
25     int a=012;
26     int b=0012;
27     int c=0x0012;
28     int d=0x12;
29     int e=0x0012;
30     int f=0x12;
31     double ee=0.01;
32     double f=0.1e-1;
33     double g=0.00;
34     double j=0.0E0;
35     double k=0.0E-0;
36     double l=0.0E+0;
37
38     int h=000;
39
40     int i=0x000;
41
42
43 }
44
45
46
47 // 测试报告：对整个测试文件的数据进行测试结果的汇报，是否所有单词都能分析成功，或有哪些不能分析成功
48
```

分析结果:

Token	Type	Line
// test1	注释	行 1
/* test */	注释	行 2
void	关键字	行 4
char	关键字	行 4
int	关键字	行 4
float	关键字	行 4
double	关键字	行 4
short	关键字	行 4
long	关键字	行 4
signed	关键字	行 4
unsigned	关键字	行 4
struct	关键字	行 5
union	关键字	行 5
enum	关键字	行 5
typedef	关键字	行 5
sizeof	关键字	行 5
auto	关键字	行 5
static	关键字	行 5
register	关键字	行 5
extern	关键字	行 5
const	关键字	行 5
volatile	关键字	行 5
return	关键字	行 5
continue	关键字	行 6
break	关键字	行 6
goto	关键字	行 6
if	关键字	行 6
else	关键字	行 6
switch	关键字	行 6
case	关键字	行 6
default	关键字	行 6
for	关键字	行 6
do	关键字	行 6
while	关键字	行 6
alignas	标识符	行 6
alignof	标识符	行 6
and	标识符	行 6

MainWindow

文件 XC++

D:/大学文件/课程文件/3_大三/编译原理/2023 编译原理/课程实验/实验1/实验1 测试数据.cpp

打开文件 保存分析结果

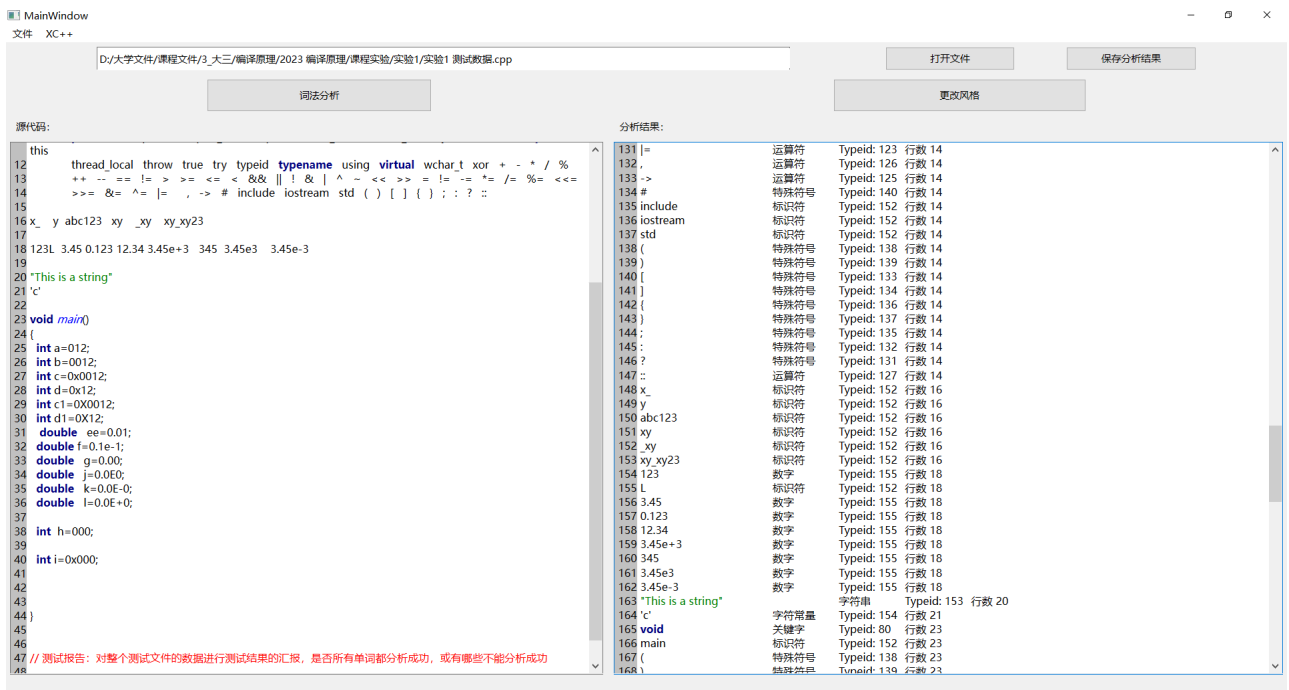
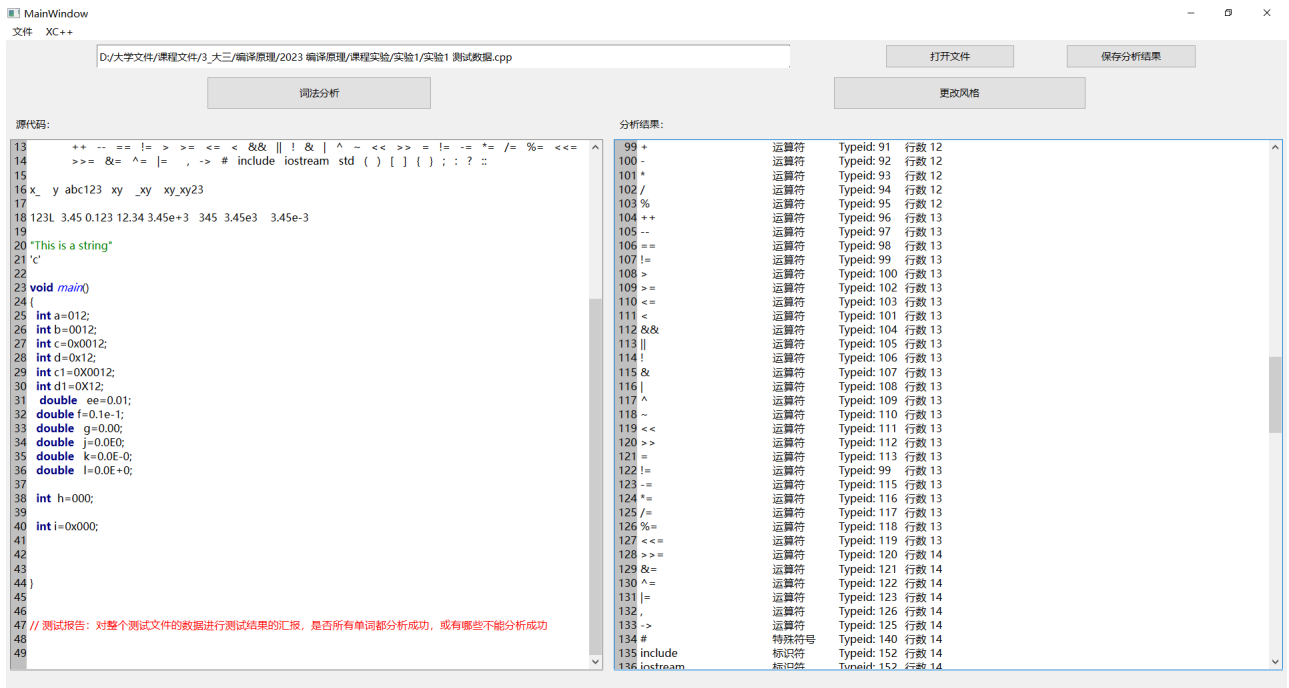
词法分析 更改风格

源代码:

```
12 this
13 thread_local throw true try typeid typename using virtual wchar_t xor + - * / %
14 ++ -- := > <= < && || ! & | ^ ~ << >> = != == * = /= %= <<=
15 >>= &= ^= |= , -> # include iostream std ( ) [ ] { } ; : ? ::
16 x_ y abc123 xy _xy xy_xy23
17
18 123L 3.45 0.123 12.34 3.45e+3 345 3.45e3 3.45e-3
19
20 "This is a string"
21 'c'
22
23 void main()
24 {
25     int a=012;
26     int b=0012;
27     int c=0x0012;
28     int d=0x12;
29     int e=0x0012;
30     int f=0x12;
31     double ee=0.01;
32     double f=0.1e-1;
33     double g=0.00;
34     double j=0.0E0;
35     double k=0.0E-0;
36     double l=0.0E+0;
37
38     int h=000;
39
40     int i=0x000;
41
42
43 }
44
45
46
47 // 测试报告：对整个测试文件的数据进行测试结果的汇报，是否所有单词都能分析成功，或有哪些不能分析成功
48
```

分析结果:

Token	Type	Line
and	标识符	行 6
and_eq	标识符	行 7
asm	关键字	行 7
atomic_cancel	标识符	行 7
atomic_commit	标识符	行 7
atomic_noexcept	标识符	行 7
bitand	标识符	行 7
bitor	标识符	行 7
bool	关键字	行 7
catch	关键字	行 7
char8_t	标识符	行 7
char16_t	标识符	行 8
char32_t	标识符	行 8
class	关键字	行 8
compl	标识符	行 8
concept	标识符	行 8
constexpr	标识符	行 8
consteval	标识符	行 8
constexpr	标识符	行 8
constinit	标识符	行 8
const_cast	标识符	行 8
co_await	标识符	行 8
co_return	标识符	行 9
co_yield	标识符	行 9
decltype	标识符	行 9
delete	关键字	行 9
dynamic_cast	标识符	行 9
explicit	关键字	行 9
export	关键字	行 9
false	关键字	行 9
friend	关键字	行 9
inline	关键字	行 9
mutable	关键字	行 10
namespace	关键字	行 10
new	关键字	行 10
noexcept	标识符	行 10
not	标识符	行 10
not_eq	标识符	行 10
nullptr	标识符	行 10



由分析结果截图可以看出，本程序把部分关键字识别成了标识符，如`alignas`，`and_eq`，`atomic_noexcept`，`concept`等，但c++中常见的关键字基本能识别出来。本程序能成功识别全部运算符与特殊符号（见附录99行），但是把`include`，`iostream`，`std`识别成了标识符（见附录135~137行）。

程序能成功识别测试文件中的全部标识符（见附录148~153行），以及大部分的数字包括十六进制数、八进制数、科学计数法以及带符号数，除了数字 `123L`（见附录154行），程序将该数字识别成了数字`123`与标识符`L`。对于字符串与字符常量，程序也能正常识别（见附录163~164行）。

2 附录

完整分析结果如下:

1	// test1	注释	Typeid: 151	行数 1
2	/* test */	注释	Typeid: 151	行数 2
3	void	关键字	Typeid: 80	行数 4
4	char	关键字	Typeid: 27	行数 4
5	int	关键字	Typeid: 49	行数 4
6	float	关键字	Typeid: 43	行数 4
7	double	关键字	Typeid: 35	行数 4
8	short	关键字	Typeid: 61	行数 4
9	long	关键字	Typeid: 50	行数 4
10	signed	关键字	Typeid: 62	行数 4
11	unsigned	关键字	Typeid: 77	行数 4
12	struct	关键字	Typeid: 66	行数 5
13	union	关键字	Typeid: 76	行数 5
14	enum	关键字	Typeid: 38	行数 5
15	typedef	关键字	Typeid: 73	行数 5
16	sizeof	关键字	Typeid: 63	行数 5
17	auto	关键字	Typeid: 22	行数 5
18	static	关键字	Typeid: 64	行数 5
19	register	关键字	Typeid: 58	行数 5
20	extern	关键字	Typeid: 41	行数 5
21	const	关键字	Typeid: 29	行数 5
22	volatile	关键字	Typeid: 81	行数 5
23	return	关键字	Typeid: 60	行数 5
24	continue	关键字	Typeid: 31	行数 6
25	break	关键字	Typeid: 24	行数 6
26	goto	关键字	Typeid: 46	行数 6
27	if	关键字	Typeid: 47	行数 6
28	else	关键字	Typeid: 37	行数 6
29	switch	关键字	Typeid: 67	行数 6
30	case	关键字	Typeid: 25	行数 6
31	default	关键字	Typeid: 32	行数 6
32	for	关键字	Typeid: 44	行数 6
33	do	关键字	Typeid: 34	行数 6
34	while	关键字	Typeid: 83	行数 6
35	alignas	标识符	Typeid: 152	行数 6
36	alignof	标识符	Typeid: 152	行数 6
37	and	标识符	Typeid: 152	行数 6
38	and_eq	标识符	Typeid: 152	行数 7
39	asm	关键字	Typeid: 21	行数 7
40	atomic_cancel	标识符	Typeid: 152	行数 7
41	atomic_commit	标识符	Typeid: 152	行数 7
42	atomic_noexcept	标识符	Typeid: 152	行数 7
43	bitand	标识符	Typeid: 152	行数 7
44	bitor	标识符	Typeid: 152	行数 7
45	bool	关键字	Typeid: 23	行数 7
46	catch	关键字	Typeid: 26	行数 7
47	char8_t	标识符	Typeid: 152	行数 7
48	char16_t	标识符	Typeid: 152	行数 8
49	char32_t	标识符	Typeid: 152	行数 8
50	class	关键字	Typeid: 28	行数 8

51	compl	标识符	Typeid: 152	行数 8
52	concept	标识符	Typeid: 152	行数 8
53	consteval	标识符	Typeid: 152	行数 8
54	constexpr	标识符	Typeid: 152	行数 8
55	constinit	标识符	Typeid: 152	行数 8
56	const_cast	关键字	Typeid: 30	行数 8
57	co_await	标识符	Typeid: 152	行数 8
58	co_return	标识符	Typeid: 152	行数 9
59	co_yield	标识符	Typeid: 152	行数 9
60	decltype	标识符	Typeid: 152	行数 9
61	delete	关键字	Typeid: 33	行数 9
62	dynamic_cast	关键字	Typeid: 36	行数 9
63	explicit	关键字	Typeid: 39	行数 9
64	export	关键字	Typeid: 40	行数 9
65	false	关键字	Typeid: 42	行数 9
66	friend	关键字	Typeid: 45	行数 9
67	inline	关键字	Typeid: 48	行数 9
68	mutable	关键字	Typeid: 51	行数 10
69	namespace	关键字	Typeid: 52	行数 10
70	new	关键字	Typeid: 53	行数 10
71	noexcept	标识符	Typeid: 152	行数 10
72	not	标识符	Typeid: 152	行数 10
73	not_eq	标识符	Typeid: 152	行数 10
74	nullptr	标识符	Typeid: 152	行数 10
75	operator	关键字	Typeid: 54	行数 10
76	or	标识符	Typeid: 152	行数 10
77	or_eq	标识符	Typeid: 152	行数 10
78	private	关键字	Typeid: 55	行数 10
79	protected	关键字	Typeid: 56	行数 10
80	public	关键字	Typeid: 57	行数 11
81	reflexpr	标识符	Typeid: 152	行数 11
82	reinterpret_cast	关键字	Typeid: 59	行数 11
83	requires	标识符	Typeid: 152	行数 11
84	static_assert	标识符	Typeid: 152	行数 11
85	static_cast	关键字	Typeid: 65	行数 11
86	synchronized	标识符	Typeid: 152	行数 11
87	template	关键字	Typeid: 68	行数 11
88	this	关键字	Typeid: 69	行数 11
89	thread_local	标识符	Typeid: 152	行数 12
90	throw	关键字	Typeid: 70	行数 12
91	true	关键字	Typeid: 71	行数 12
92	try	关键字	Typeid: 72	行数 12
93	typeid	关键字	Typeid: 74	行数 12
94	typename	关键字	Typeid: 75	行数 12
95	using	关键字	Typeid: 78	行数 12
96	virtual	关键字	Typeid: 79	行数 12
97	wchar_t	关键字	Typeid: 82	行数 12
98	xor	标识符	Typeid: 152	行数 12
99	+	运算符	Typeid: 91	行数 12
100	-	运算符	Typeid: 92	行数 12
101	*	运算符	Typeid: 93	行数 12
102	/	运算符	Typeid: 94	行数 12
103	%	运算符	Typeid: 95	行数 12
104	++	运算符	Typeid: 96	行数 13
105	--	运算符	Typeid: 97	行数 13

106	==	运算符	Typeid: 98	行数 13
107	!=	运算符	Typeid: 99	行数 13
108	>	运算符	Typeid: 100	行数 13
109	>=	运算符	Typeid: 102	行数 13
110	<=	运算符	Typeid: 103	行数 13
111	<	运算符	Typeid: 101	行数 13
112	&&	运算符	Typeid: 104	行数 13
113		运算符	Typeid: 105	行数 13
114	!	运算符	Typeid: 106	行数 13
115	&	运算符	Typeid: 107	行数 13
116		运算符	Typeid: 108	行数 13
117	^	运算符	Typeid: 109	行数 13
118	~	运算符	Typeid: 110	行数 13
119	<<	运算符	Typeid: 111	行数 13
120	>>	运算符	Typeid: 112	行数 13
121	=	运算符	Typeid: 113	行数 13
122	!=	运算符	Typeid: 99	行数 13
123	-=	运算符	Typeid: 115	行数 13
124	*=	运算符	Typeid: 116	行数 13
125	/=	运算符	Typeid: 117	行数 13
126	%=	运算符	Typeid: 118	行数 13
127	<<=	运算符	Typeid: 119	行数 13
128	>>=	运算符	Typeid: 120	行数 14
129	&=	运算符	Typeid: 121	行数 14
130	^=	运算符	Typeid: 122	行数 14
131	=	运算符	Typeid: 123	行数 14
132	,	运算符	Typeid: 126	行数 14
133	->	运算符	Typeid: 125	行数 14
134	#	特殊符号	Typeid: 140	行数 14
135	include	标识符	Typeid: 152	行数 14
136	iostream	标识符	Typeid: 152	行数 14
137	std	标识符	Typeid: 152	行数 14
138	(特殊符号	Typeid: 138	行数 14
139)	特殊符号	Typeid: 139	行数 14
140	[特殊符号	Typeid: 133	行数 14
141]	特殊符号	Typeid: 134	行数 14
142	{	特殊符号	Typeid: 136	行数 14
143	}	特殊符号	Typeid: 137	行数 14
144	;	特殊符号	Typeid: 135	行数 14
145	:	特殊符号	Typeid: 132	行数 14
146	?	特殊符号	Typeid: 131	行数 14
147	::	运算符	Typeid: 127	行数 14
148	x_	标识符	Typeid: 152	行数 16
149	y	标识符	Typeid: 152	行数 16
150	abc123	标识符	Typeid: 152	行数 16
151	xy	标识符	Typeid: 152	行数 16
152	_xy	标识符	Typeid: 152	行数 16
153	xy_xy23	标识符	Typeid: 152	行数 16
154	123	数字	Typeid: 155	行数 18
155	L	标识符	Typeid: 152	行数 18
156	3.45	数字	Typeid: 155	行数 18
157	0.123	数字	Typeid: 155	行数 18
158	12.34	数字	Typeid: 155	行数 18
159	3.45e+3	数字	Typeid: 155	行数 18
160	345	数字	Typeid: 155	行数 18

```

161 3.45e3      数字 Typeid: 155 行数 18
162 3.45e-3    数字 Typeid: 155 行数 18
163 "This is a string" 字符串 Typeid: 153 行数 20
164 'c'        字符常量 Typeid: 154 行数 21
165 void        关键字 Typeid: 80 行数 23
166 main        标识符 Typeid: 152 行数 23
167 (           特殊符号 Typeid: 138 行数 23
168 )           特殊符号 Typeid: 139 行数 23
169 {           特殊符号 Typeid: 136 行数 24
170 int         关键字 Typeid: 49 行数 25
171 a           标识符 Typeid: 152 行数 25
172 =           运算符 Typeid: 113 行数 25
173 012         数字 Typeid: 155 行数 25
174 ;           特殊符号 Typeid: 135 行数 25
175 int         关键字 Typeid: 49 行数 26
176 b           标识符 Typeid: 152 行数 26
177 =           运算符 Typeid: 113 行数 26
178 0012        数字 Typeid: 155 行数 26
179 ;           特殊符号 Typeid: 135 行数 26
180 int         关键字 Typeid: 49 行数 27
181 c           标识符 Typeid: 152 行数 27
182 =           运算符 Typeid: 113 行数 27
183 0x0012      数字 Typeid: 155 行数 27
184 ;           特殊符号 Typeid: 135 行数 27
185 int         关键字 Typeid: 49 行数 28
186 d           标识符 Typeid: 152 行数 28
187 =           运算符 Typeid: 113 行数 28
188 0x12        数字 Typeid: 155 行数 28
189 ;           特殊符号 Typeid: 135 行数 28
190 int         关键字 Typeid: 49 行数 29
191 c1          标识符 Typeid: 152 行数 29
192 =           运算符 Typeid: 113 行数 29
193 0X0012      数字 Typeid: 155 行数 29
194 ;           特殊符号 Typeid: 135 行数 29
195 int         关键字 Typeid: 49 行数 30
196 d1          标识符 Typeid: 152 行数 30
197 =           运算符 Typeid: 113 行数 30
198 0X12        数字 Typeid: 155 行数 30
199 ;           特殊符号 Typeid: 135 行数 30
200 double      关键字 Typeid: 35 行数 31
201 ee          标识符 Typeid: 152 行数 31
202 =           运算符 Typeid: 113 行数 31
203 0.01        数字 Typeid: 155 行数 31
204 ;           特殊符号 Typeid: 135 行数 31
205 double      关键字 Typeid: 35 行数 32
206 f           标识符 Typeid: 152 行数 32
207 =           运算符 Typeid: 113 行数 32
208 0.1e-1      数字 Typeid: 155 行数 32
209 ;           特殊符号 Typeid: 135 行数 32
210 double      关键字 Typeid: 35 行数 33
211 g           标识符 Typeid: 152 行数 33
212 =           运算符 Typeid: 113 行数 33
213 0.00        数字 Typeid: 155 行数 33
214 ;           特殊符号 Typeid: 135 行数 33
215 double      关键字 Typeid: 35 行数 34

```

```

216 j      标识符 Typeid: 152 行数 34
217 =      运算符 Typeid: 113 行数 34
218 0.0E0   数字   Typeid: 155 行数 34
219 ;      特殊符号 Typeid: 135 行数 34
220 double  关键字 Typeid: 35  行数 35
221 k      标识符 Typeid: 152 行数 35
222 =      运算符 Typeid: 113 行数 35
223 0.0E-0   数字   Typeid: 155 行数 35
224 ;      特殊符号 Typeid: 135 行数 35
225 double  关键字 Typeid: 35  行数 36
226 l      标识符 Typeid: 152 行数 36
227 =      运算符 Typeid: 113 行数 36
228 0.0E+0   数字   Typeid: 155 行数 36
229 ;      特殊符号 Typeid: 135 行数 36
230 int     关键字 Typeid: 49  行数 38
231 h      标识符 Typeid: 152 行数 38
232 =      运算符 Typeid: 113 行数 38
233 000     数字   Typeid: 155 行数 38
234 ;      特殊符号 Typeid: 135 行数 38
235 int     关键字 Typeid: 49  行数 40
236 i      标识符 Typeid: 152 行数 40
237 =      运算符 Typeid: 113 行数 40
238 0x000   数字   Typeid: 155 行数 40
239 ;      特殊符号 Typeid: 135 行数 40
240 }      特殊符号 Typeid: 137 行数 44
241 // 测试报告: 对整个测试文件的数据进行测试结果的汇报, 是否所有单词都分析成功, 或有哪些不能分
析成功      注释   Typeid: 151 行数 47
242

```

