西安交通大学
XI'AN JIAOTONG UNIVERSITY

2022-2023 学年第二学期

《编译器设计专题实验》

实验报告 4

学　　院：电信学部

班　　级：[redacted]

学　　号：[redacted]

姓　　名：[redacted]

二〇二三年 五月

目录

《实验 4-词法分析(二),实现 COOL 词法分析》

一、实验内容（必做）

1. 完成 PA2 的 COOL 词法分析器；
2. 完成 cool 所有嵌套词素的匹配检查，例如 if 和 fi 的匹配，case 和 esac 的匹配，{和}的匹配等等；
3. 回答问题：如何使 f6.lex+make 文件生成的程序完整处理 test3 文件？

二、实验分析

(1) 注释

对于单行注释，处理较为简单，使用--.*$直接进行匹配即可；

对于多行注释，则需要使用多重入口来处理：若匹配到"(*"则进入多行注释入口，若匹配到"*)"则提示"Unmatched *)"错误。进入多行注释后，还需要处理嵌套的"(*"与"*)"、换行符\n，以及当注释中出现文件结束符<<EOF>>时，需要提示"EOF in comment"错误。

(2) 换行符与空白字符

对于换行符\n，递增 curr_lineno 即可；

对于空白字符[\t\r\v\f\b]+，不做任何处理。

(3) 运算符与定界符

对于多字符的运算符（"=>"、"<-"以及"<="），返回 cool-parse.h 文件中的宏定义即可；

对于单字符的运算符与定界符，返回对应的 ASCII 码即可。

(4) 关键字

需要注意的是，cool 语言对关键字的大小写不敏感，匹配后返回 cool-parse.h 文件中的宏定义即可。

(5) 标识符

标识符由大小写字母、数字以及下划线组成，大写字母开头的标识符识别为 TYPEID，小写字母开头的标识符识别为 OBJECTID。此外，存在两个特殊的标识符："SELF_TYPE"为 TYPEID，"self"为 OBJECTID。

(6) 布尔常量与整型常量

布尔常量 true 和 false 的第一个字母为小写，后面的字母对大小写不敏感。

对于整型常量,需要注意的是 cool 语言中只存在非负整数常量。

(7) 字符串常量

字符串常量的处理较为麻烦，且需要使用多重入口。当在字符串中换行时，应提示"Unterminated string constant"错误；当在字符串中出现空字符时，应提示"String contains escaped null character"错误；当字符串中出现文件结束符时，应提示"EOF in string constant"错误；当字符串过长时，应继续匹配字符串而不读取，且提示"String constant too long"错误。此外，还应处理好字符串中的转义符，如：\n、\"、\\、\t 等，以及字符串中的续行：即在字符串中输入\并回车。

(8) 其它字符

当上述规则均无法匹配某个字符时，将该字符视为错误字符。

(9) 嵌套词素的匹配检查

对于每一对嵌套词素，如 if 与 fi，定义变量 if_fi 并初始化为 0，当匹配 if 时，递增变量，当匹配 fi 时，递减变量。若 if_fi 小于 0，说

明 fi 前面缺少 if，提示"FI : Lack of IF"错误，并将 if_fi 重置为 0。当词法分析器分析完整个文本后，若 if_fi 大于 0，说明 if 后面缺少 fi，提示"Missing if_fi FI"错误。

注意，在词法分析的过程中，先出现 if 再出现 fi 是正确的，但此时 if_fi 大于 0，因此分析错误的工作应在词法分析完成后进行，故应在 lextest.cc 文件中补充相关代码。

三、实验结果

（1）刚开始遇到的问题及解决方法

```
GuoSongjian(Fri May 5 18:52:41):~/compiler_exp/cool/cool/assignments/PA2$ python3 comp.py
yourtest: #1 STR_CONST ""123\\"
standard: #1 STR_CONST "\"123\\"
```

comp.py 文件是对比自己实现的词法分析器与标准词法分析器的脚本。对于测试用例"\"123\\"，自己实现的词法分析器在处理转义字符\"时输出错误。

负责输出词法分析结果的是 utilities.cc 文件，其中对于转义字符\"的处理方法为 case '\"' : str << "\""; break;

将其改为 case '\"' : str << ((i == 0 || i == len-1)? "\"": "\\\""); break;后，得到了与标准词法分析器相同的输出结果。

（2）词法分析器结果

```
GuoSongjian(Fri May 5 19:12:36):~/compiler_exp/cool/cool/assignments/PA2$ make
../..//etc/link-object 2 parser
linking parser
parser is not compiled for x86_64
../..//etc/link-object 2 semant
linking semant
semant is not compiled for x86_64
../..//etc/link-object 2 cgen
linking cgen
cgen is not compiled for x86_64
./lexer test.cl
#name "test.cl"
#3 CLASS
#3 TYPEID CellularAutomaton
#3 INHERITS
#3 TYPEID IO
#4 OBJECTID population_map
#4 ':'
#4 TYPEID SELF_TYPE
```

```
#4 ';'
#6 OBJECTID init
#6 '('
#6 OBJECTID map
#6 ':'
#6 TYPEID String
#6 ')'
#7 TYPEID SELF_TYPE
#8 '{'
#9 WHILE
#9 LOOP
#9 POOL
#10 '}'
```

```
#11 ';'
#12 OBJECTID cell_at_next_evolution
#13 '('
#13 LET
#13 OBJECTID a
#13 ':'
#13 TYPEID Int
#13 ASSIGN
#13 OBJECTID num
#13 ERROR "["
#13 ERROR "]"
#13 IN
#13 ')'
#14 ERROR "String constant too long"
#16 ERROR "Unterminated string constant"
#16 STR_CONST "this is a string.\n"
#18 STR_CONST "12\n3"
#19 STR_CONST ""
#20 STR_CONST "\"123\\"
#21 STR_CONST "\b"
#22 STR_CONST "x"
#23 ERROR "FI : Lack of IF"
#23 ERROR "FI : Lack of IF"
#23 ERROR "FI : Lack of IF"
#23 IF
#24 CASE
#25 '('
#25 '('
#25 '('
#25 '('
#25 '('
#25 '('
#25 '('
#25 ')'
#25 ')'
#26 ERROR "'}' : Lack of '{'"
#27 ERROR "Unmatched *)"
#29 ERROR "EOF in comment"

Missing 5 )
Missing 1 FI
Missing 1 ESAC
GuoSongjian(Fri May 5 19:12:46):~/compiler_exp/cool/cool/assignments/PA2$
```

从上图中可以发现，词法分析器正确识别出了关键字、标识符、字符串常量，并正确处理了注释、嵌套词素匹配以及各种错误。

(3) 回答问题

由于 f6.lex 中未定义空白字符与标识符的匹配规则，因此词法分

析器无法识别“　p”，于是将其原样输出，对于“ t ”来说也是这样。当匹配到“=”时，执行 return 语句将退出词法分析，因此程序结束运行，不再匹配后续文本。

要想完整处理 test3 文件，还需要定义空白字符、标识符、常量等的匹配规则，并将 return 语句改为其它处理语句。

四、源代码

（1）词法分析器- cool.flex

```
%{
#include <cool-parse.h>
#include <stringtab.h>
#include <utilities.h>

#define yylval cool_yylval
#define yylex  cool_yylex

#define MAX_STR_CONST 1025
#define YY_NO_UNPUT

extern FILE *fin;

#undef YY_INPUT
#define YY_INPUT(buf,result,max_size) \
        if ( (result = fread( (char*)buf, sizeof(char), max_size,
fin)) < 0) \
                YY_FATAL_ERROR( "read() in flex scanner failed");

char string_buf[MAX_STR_CONST];
char *string_buf_ptr;

extern int curr_lineno;
extern int verbose_flag;

extern YYSTYPE cool_yylval;

#include <string.h>

int uniqueIndex = 1;
```

```
int comment_num = 0;
int string_len = 0;
bool null_flag = false;
bool long_flag = false;

extern int if_fi;
extern int loop_pool;
extern int case_esac;
extern int pare1;
extern int pare2;
%}

%x MutiCom STRING

%%

--.*$             { /* do nothing */ }

"(*"              { comment_num++; BEGIN MutiCom; }
"*)"              { strcpy(cool_yylval.error_msg, "Unmatched *)");
return ERROR; }
<MutiCom>"(*"      { if (comment_num >= 0x7fffffff)
{strcpy(cool_yylval.error_msg, "Too many comments");return ERROR;}
comment_num++; }
<MutiCom>"*)"      { comment_num--; if (comment_num == 0) BEGIN
0; }
<MutiCom>\n        { curr_lineno++; }
<MutiCom><<EOF>>    { strcpy(cool_yylval.error_msg, "EOF in
comment"); BEGIN 0; return ERROR; }
<MutiCom>.         { /* do nothing */ }

\n                { curr_lineno++; }

[ \t\r\v\f\b]+     { /* do nothing */ }

"=>"              { return DARROW; }
"<-"              { return ASSIGN; }
"<="              { return LE; }

"("               { pare1++; return '('; }
")"               { pare1--; if (pare1 < 0)
{pare1++;strcpy(cool_yylval.error_msg, "')' : Lack of '('");return
ERROR;} return ')'; }
"{"               { pare2++; return '{'; }
```

```
"}"              { pare2--; if (pare2 < 0)
{pare2++;strcpy(cool_yylval.error_msg, "'}' : Lack of '{'");return
ERROR;} return '}'; }
":"              { return ':'; }
";"              { return ';'; }
"="              { return '='; }
"<"              { return '<'; }
"."              { return '.'; }
","              { return ','; }
"~"              { return '~'; }
"+"              { return '+'; }
"-"              { return '-'; }
"*"              { return '*'; }
"/"              { return '/'; }
"@"              { return '@'; }

(?i:class)        { return CLASS; }
(?i:else)         { return ELSE; }
(?i:fi)           { if_fi--; if (if_fi < 0)
{if_fi++;strcpy(cool_yylval.error_msg, "FI : Lack of IF");return
ERROR;} return FI; }
(?i:if)           { if_fi++; return IF; }
(?i:in)           { return IN; }
(?i:inherits)      { return INHERITS; }
(?i:let)          { return LET; }
(?i:loop)         { loop_pool++; return LOOP; }
(?i:pool)         { loop_pool--; if (loop_pool < 0)
{loop_pool++;strcpy(cool_yylval.error_msg, "POOL : Lack of
LOOP");return ERROR;} return POOL; }
(?i:then)         { return THEN; }
(?i:while)        { return WHILE; }
(?i:case)         { case_esac++; return CASE; }
(?i:esac)         { case_esac--; if (case_esac < 0)
{case_esac++;strcpy(cool_yylval.error_msg, "ESAC : Lack of
CASE");return ERROR;} return ESAC; }
(?i:of)           { return OF; }
(?i:new)          { return NEW; }
(?i:isvoid)       { return ISVOID; }
(?i:not)          { return NOT; }

t(?i:rue)         { cool_yylval.boolean = 1; return BOOL_CONST; }
f(?i:alse)        { cool_yylval.boolean = 0; return BOOL_CONST; }
```

```
"SELF_TYPE"        { cool_yylval.symbol = new
IdEntry(yytext,strlen(yytext),uniqueIndex++); return TYPEID; }

"self"            { cool_yylval.symbol = new
IdEntry(yytext,strlen(yytext),uniqueIndex++); return OBJECTID; }

[A-Z][A-Za-z0-9_]*  { cool_yylval.symbol = new
IdEntry(yytext,strlen(yytext),uniqueIndex++); return TYPEID; }

[a-z][A-Za-z0-9_]*  { cool_yylval.symbol = new
IdEntry(yytext,strlen(yytext),uniqueIndex++); return OBJECTID; }

[0-9]+            { cool_yylval.symbol = new
IntEntry(yytext,strlen(yytext),uniqueIndex++); return INT_CONST; }

\"                { string_buf_ptr = string_buf;
                    string_len = 0;
                    null_flag = false;
                    long_flag = false;
                    string_buf[string_len++] = '\"';
                    BEGIN STRING; }
<STRING>\n         { curr_lineno++;
                    strcpy(cool_yylval.error_msg, "Unterminated
string constant");
                    BEGIN 0;
                    return ERROR; }
<STRING>\0         { null_flag = true; }
<STRING><<EOF>>     { strcpy(cool_yylval.error_msg, "EOF in string
constant");
                    BEGIN 0;
                    return ERROR; }
<STRING>\\\n       { curr_lineno++;
                    if (string_len >= MAX_STR_CONST - 1)
                        long_flag = true;
                    else
                        string_buf[string_len++] = '\n'; }
<STRING>\\.        { if (string_len >= MAX_STR_CONST - 1)
                        long_flag = true;
                    else
                        switch(yytext[1]) {
                            case '\"': string_buf[string_len++] =
'\"'; break;
                            case '\\': string_buf[string_len++] =
'\\'; break;
```

```
                        case 'b': string_buf[string_len++] = '\b';
break;
                        case 'f': string_buf[string_len++] = '\f';
break;
                        case 'n': string_buf[string_len++] = '\n';
break;
                        case 't': string_buf[string_len++] = '\t';
break;
                        default: string_buf[string_len++] =
yytext[1];
                    } }
<STRING>\"          { BEGIN 0;
                    if (null_flag) {
                        strcpy(cool_yylval.error_msg, "String
contains escaped null character");
                        return ERROR;
                    }
                    if (long_flag){
                        strcpy(cool_yylval.error_msg, "String
constant too long");
                        return ERROR;
                    }
                    string_buf[string_len++] = '\"';
                    cool_yylval.symbol = new
StringEntry(string_buf,string_len,uniqueIndex++);
                    return STR_CONST; }
<STRING>.           { if (string_len >= MAX_STR_CONST - 1)
                        long_flag = true;
                    else
                        string_buf[string_len++] = yytext[0]; }

.                  { cool_yylval.error_msg = yytext; return ERROR; }

%%

int yywrap() { return 1; }
```

(2) 对 utilities.cc 的修改

```
void print_escaped_string(std::ostream& str, const char *s)
{
  int i = 0, len = strlen(s);
  while (*s) {
    switch (*s) {
    case '\\' : str << "\\\\"; break;
```

```
    case '\"' : str << ((i == 0 || i == len-1)? "\"": "\\\"");
break;
    //case '\"' : str << "\""; break;
    case '\n' : str << "\\n"; break;
    case '\t' : str << "\\t"; break;
    case '\b' : str << "\\b"; break;
    case '\f' : str << "\\f"; break;

    default:
      if (isprint(*s))
         str << *s;
      else
         //
         // Unprintable characters are printed using octal
equivalents.
         // To get the sign of the octal number correct, the
character
         // must be cast to an unsigned char before coverting it to
an
         // integer.
         //
         str << '\\' << oct << setfill('0') << setw(3)
            << (int) ((unsigned char) (*s))
            << dec << setfill(' ');
      break;
    }
    s++;
    i++;
  }
}
```

(3) 对 lextest.cc 的修改

```
int if_fi = 0;
int loop_pool = 0;
int case_esac = 0;
int pare1 = 0;
int pare2 = 0;

int main(int argc, char** argv) {
        int token;

        handle_flags(argc,argv);

        while (optind < argc) {
```

```
            fin = fopen(argv[optind], "r");
            if (fin == NULL) {
                cerr << "Could not open input file " << argv[optind]
<< endl;
                exit(1);
            }

           // sm: the 'coolc' compiler's file-handling loop resets
           // this counter, so let's make the stand-alone lexer
           // do the same thing
           curr_lineno = 1;

            //
            // Scan and print all tokens.
            //
            cout << "#name \"" << argv[optind] << "\"" << endl;
            while ((token = cool_yylex()) != 0) {
                dump_cool_token(cout, curr_lineno, token,
cool_yylval);
            }
            fclose(fin);
            optind++;
            std::cout << std::endl;
            if (pare1 > 0) std::cout << "Missing " << pare1 << " )"
<< std::endl;
            if (pare2 > 0) std::cout << "Missing " << pare2 << " }"
<< std::endl;
            if (if_fi > 0) std::cout << "Missing " << if_fi << " FI"
<< std::endl;
            if (loop_pool > 0) std::cout << "Missing " << loop_pool
<< " POOL" << std::endl;
            if (case_esac > 0) std::cout << "Missing " << case_esac
<< " ESAC" << std::endl;
        }
        exit(0);
}
```

（4）与标准词法分析器进行对比的 Python 脚本-comp.py

```
import os
command = "./lexer test.cl > test.output"
os.system(command)
command = "./reference-lexer test.cl > standard.output"
os.system(command)
flag = True
```

```
with open('test.output', 'r') as f1, open('standard.output', 'r')
as f2:
    for line1, line2 in zip(f1, f2):
        if line1 != line2:
            flag = False
            print(f"yourtest: {line1.strip()}")
            print(f"standard: {line2.strip()}")
if flag:
    print("Pass test.")
```

（5）测试 COOL 程序-test.cl

```
(* balabala
*)
class CellularAutomaton inherits IO
population_map : SELF_TYPE;
-- fff
init(map : String)
SELF_TYPE
{
while loop (* nothing *) pool
}
;
cell_at_next_evolution
( let a : Int <- num[] in)
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX"
"test
"this is a string.\n"
"12\
3"
""
"\"123\\"
"\b"
"\x"
fi fi fi if
case
(((((())
}
*)
(* bababababa
```