编译器实验报告

计算机 82 班 施炎江

第一次实验

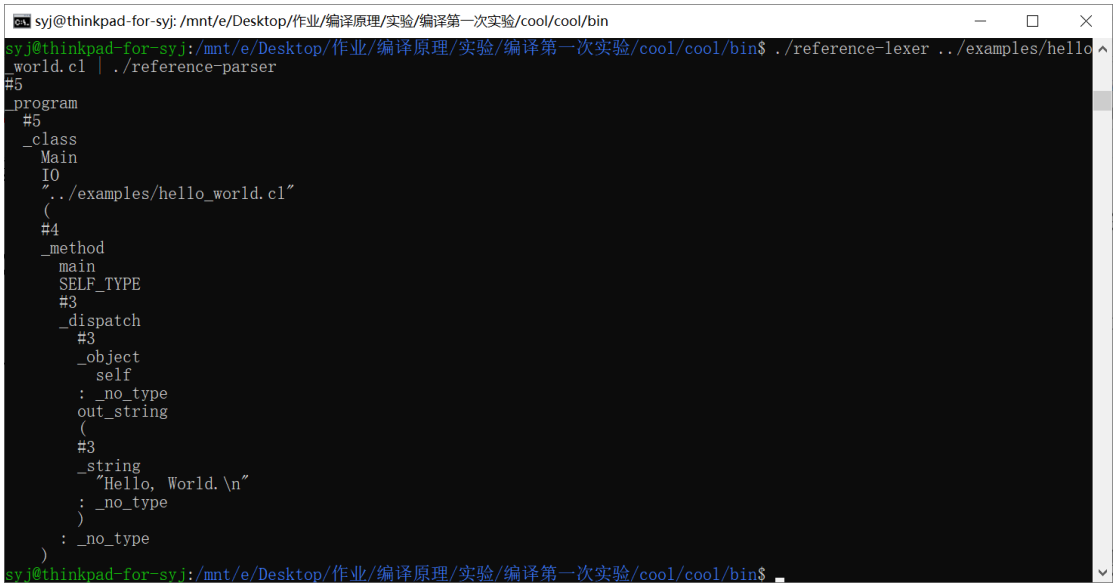
1. 编译运行 hello world

```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
li: command not found
root@thinkpad-for-syj:~#
root@thinkpad-for-syj:~# ls -al
total 12
drwx------ 1 root root 4096 Jul  4 13:21 .
drwxr-xr-x 1 root root 4096 Jun  3 15:14 ..
-rw------- 1 root root 1130 Jul  4 13:36 .bash_history
-rw-r--r-- 1 root root 3106 Dec  5  2019 .bashrc
-rw-rw-rw- 1 root root    0 Jul  4 13:21 .motd_shown
-rw-r--r-- 1 root root  161 Dec  5  2019 .profile
-rw------- 1 root root 2327 Jun  4 20:01 .viminfo
root@thinkpad-for-syj:~# cd mnt
-bash: cd: mnt: No such file or directory
root@thinkpad-for-syj:~# exit
logout
syj@thinkpad-for-syj:~$ cd mnt
-bash: cd: mnt: No such file or directory
syj@thinkpad-for-syj:~$ cd /mnt
syj@thinkpad-for-syj:/mnt$ cd e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./coolc ../examples/hello_world.cl
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./spim -trap_file ../lib/trap.handl
er -file ../examples/hello_world.s
SPIM Version 6.5 of January 4, 2003
Copyright 1990-2003 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ../lib/trap.handler
Hello, World.
COOL program successfully executed
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ su -
```

2. 用标准的词法分析程序，编译一个 Cool 语言程序，观察结果

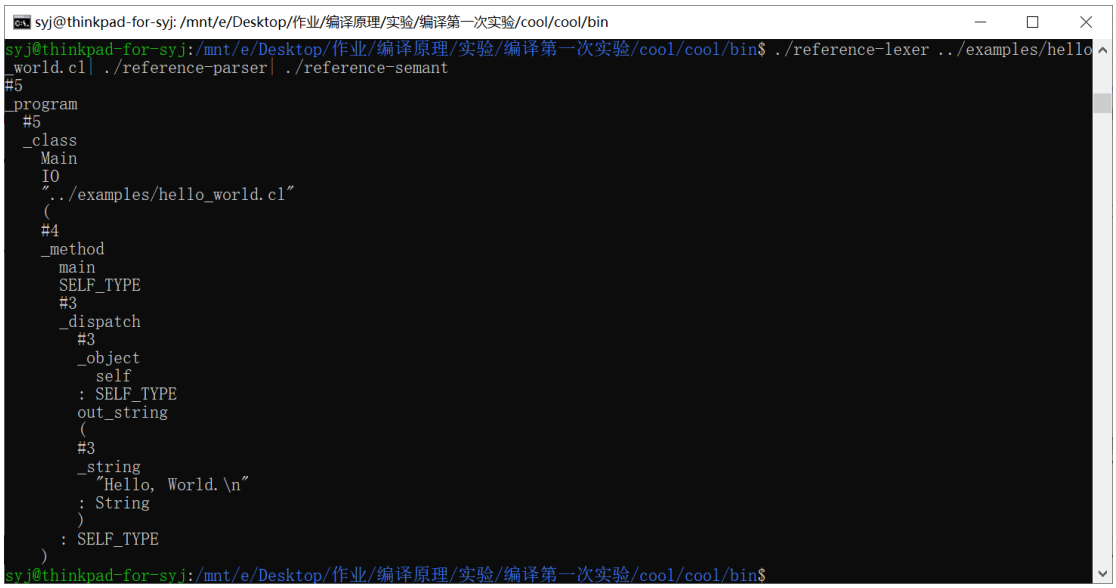
```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
E:\Desktop\作业\形式语言&编译原理\实验\编译第一次实验\cool\cool/etc/link-shared 1 Makefile
make: E:Desktop作业形式语言&编译原理实验编译第一次实验coolcool/etc/link-shared: Command not found
Makefile:26: recipe for target 'lsource' failed
make: *** [lsource] Error 127
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA1$ cd ..
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments$ cd ..
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool$ cd bin
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./reference-lexer ../examples/hello
_world.cl
#name "../examples/hello_world.cl"
#1 CLASS
#1 TYPEID Main
#1 INHERITS
#1 TYPEID IO
#1 '{'
#2 OBJECTID main
#2 '('
#2 ')'
#2 ':'
#2 TYPEID SELF_TYPE
#2 '{'
#3 OBJECTID out_string
#3 '('
#3 STR_CONST "Hello, World.\n"
#3 ')'
#4 '}'
#4 ';'
#5 '}'
#5 ';'
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$
```

3. 用标准的词法分析程序和语法分析程序，编译一个 Cool 语言程序，观察结果



```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./reference-lexer ../examples/hello
_world.cl | ./reference-parser
#5
_program
  #5
  _class
    Main
    IO
    "../examples/hello_world.cl"
    (
    #4
    _method
      main
      SELF_TYPE
      #3
      _dispatch
        #3
        _object
          self
        : _no_type
        out_string
        (
        #3
        _string
          "Hello, World.\n"
        : _no_type
        )
      : _no_type
    )
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$
```

4. 用标准的词法分析程序，语法分析程序和语义分析程序编译一个 Cool 语言程序，观察结果



```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./reference-lexer ../examples/hello
_world.cl| ./reference-parser| ./reference-semant
#5
_program
  #5
  _class
    Main
    IO
    "../examples/hello_world.cl"
    (
    #4
    _method
      main
      SELF_TYPE
      #3
      _dispatch
        #3
        _object
          self
        : SELF_TYPE
        out_string
        (
        #3
        _string
          "Hello, World.\n"
        : String
        )
      : SELF_TYPE
    )
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$
```

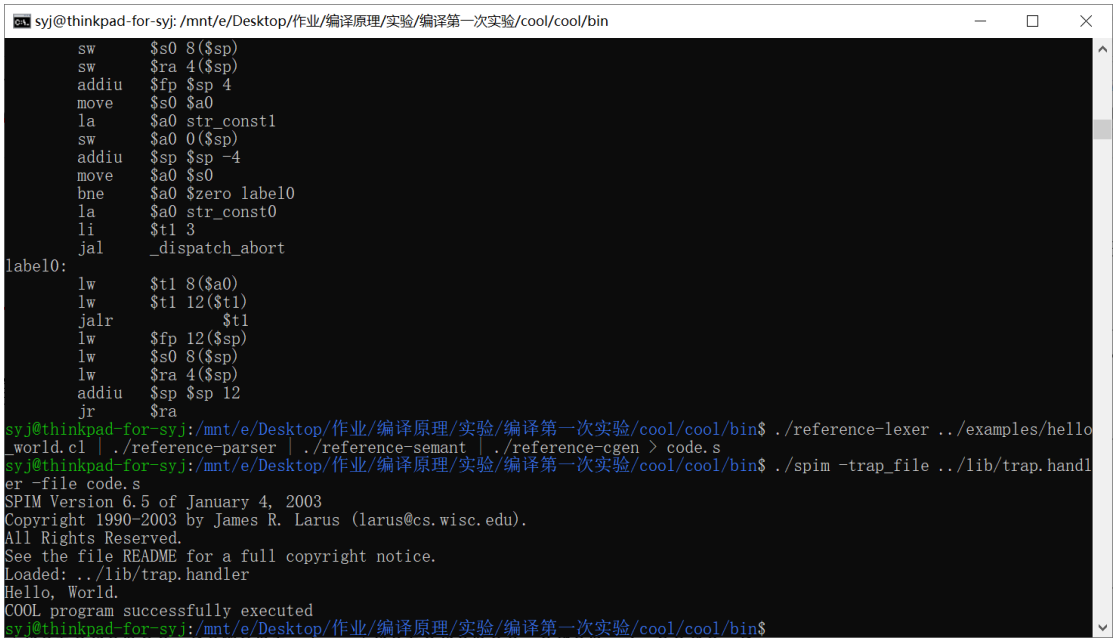
5. 用标准的词法分析程序，语法分析程序，语义分析程序和代码生成程序共同编译 Cool 语言程序，生成最终的汇编代码（由于输出较多，仅截取部分输出结果示意）

```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./reference-lexer ../examples/hello
_world.cl | ./reference-parser | ./reference-semant | ./reference-cgen
        .data
        .align  2
        .globl  class_nameTab
        .globl  Main_protObj
        .globl  Int_protObj
        .globl  String_protObj
        .globl  bool_const0
        .globl  bool_const1
        .globl  _int_tag
        .globl  _bool_tag
        .globl  _string_tag
_int_tag:
        .word   3
_bool_tag:
        .word   4
_string_tag:
        .word   5
        .globl  _MemMgr_INITIALIZER
_MemMgr_INITIALIZER:
        .word   _NoGC_Init
        .globl  _MemMgr_COLLECTOR
_MemMgr_COLLECTOR:
        .word   _NoGC_Collect
        .globl  _MemMgr_TEST
_MemMgr_TEST:
        .word   0
        .word   -1
str_const9:
        .word   5
        .word   5
        .word   String_dispTab
        .word   int_const0
        .byte   0
        .align  2
        .word   -1
str_const8:
        .word   5
        .word   6
        .word   String_dispTab
        .word   int_const1
        .ascii  "String"
        .byte   0
        .align  2
        .word   -1
str_const7:
        .word   5
        .word   6
        .word   String_dispTab
        .word   int_const2
        .ascii  "Bool"
        .byte   0
        .align  2
        .word   -1
```

6. 将得到的汇编代码输出到.s 文件中

```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin
        lw      $ra 4($sp)
        addiu   $sp $sp 12
        jr      $ra
Main.main:
        addiu   $sp $sp -12
        sw      $fp 12($sp)
        sw      $s0 8($sp)
        sw      $ra 4($sp)
        addiu   $fp $sp 4
        move    $s0 $a0
        la      $a0 str_const1
        sw      $a0 0($sp)
        addiu   $sp $sp -4
        move    $a0 $s0
        bne     $a0 $zero label0
        la      $a0 str_const0
        li      $t1 3
        jal     _dispatch_abort
label0:
        lw      $t1 8($a0)
        lw      $t1 12($t1)
        jalr            $t1
        lw      $fp 12($sp)
        lw      $s0 8($sp)
        lw      $ra 4($sp)
        addiu   $sp $sp 12
        jr      $ra
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$ ./reference-lexer ../examples/hello
_world.cl | ./reference-parser | ./reference-semant | ./reference-cgen > code.s
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/bin$
```

7. 将生成的汇编代码在 spim 上运行



第二次实验

1. Stack.cl 代码

```
(*
  The class A2I provides integer-to-string and string-to-integer
conversion routines.  To use these routines, either inherit them
in the class where needed, have a dummy variable bound to
something of type A2I, or simpl write (new A2I).method(argument).
*)


(*
  c2i   Converts a 1-character string to an integer.  Aborts
        if the string is not "0" through "9"
*)
```

```
class A2I {


    c2i(char : String) : Int {
       if char = "0" then 0 else
       if char = "1" then 1 else
       if char = "2" then 2 else
       if char = "3" then 3 else
       if char = "4" then 4 else
       if char = "5" then 5 else
       if char = "6" then 6 else
       if char = "7" then 7 else
       if char = "8" then 8 else
       if char = "9" then 9 else
       { abort(); 0; }  -
- the 0 is needed to satisfy the typchecker
       fi fi fi fi fi fi fi fi fi fi
    };


(*
   i2c is the inverse of c2i.
*)
    i2c(i : Int) : String {
```

```
        if i = 0 then "0" else
        if i = 1 then "1" else
        if i = 2 then "2" else
        if i = 3 then "3" else
        if i = 4 then "4" else
        if i = 5 then "5" else
        if i = 6 then "6" else
        if i = 7 then "7" else
        if i = 8 then "8" else
        if i = 9 then "9" else
        { abort(); ""; }  -
- the "" is needed to satisfy the typchecker
        fi fi fi fi fi fi fi fi fi fi
     };

(*
   a2i converts an ASCII string into an integer.  The empty string
is converted to 0.  Signed and unsigned strings are handled.  The
method aborts if the string does not represent an integer.  Very
long strings of digits produce strange answers because of arithmetic
overflow.
```

```
*)

     a2i(s : String) : Int {

        if s.length() = 0 then 0 else

        if s.substr(0,1) = "-" then ~a2i_aux(s.substr(1,s.length()-

1)) else

        if s.substr(0,1) = "+" then a2i_aux(s.substr(1,s.length()-

1)) else

           a2i_aux(s)

        fi fi fi

     };



(*

  a2i_aux converts the usigned portion of the string.  As a programm

ing

example, this method is written iteratively.

*)

     a2i_aux(s : String) : Int {

        (let int : Int <- 0 in

           {

               (let j : Int <- s.length() in

                  (let i : Int <- 0 in

                     while i < j loop
```

```
                        {
                            int <- int * 10 + c2i(s.substr(i,1));
                            i <- i + 1;
                        }
                    pool
                )
            );
            int;
        }
    )
};

(*
    i2a converts an integer to a string.  Positive and negative
numbers are handled correctly.
*)
    i2a(i : Int) : String {
        if i = 0 then "0" else
if 0 < i then i2a_aux(i) else
          "-".concat(i2a_aux(i * ~1))
        fi fi
    };
```

```
(*
    i2a_aux is an example using recursion.
*)
    i2a_aux(i : Int) : String {
        if i = 0 then "" else
            (let next : Int <- i / 10 in
                i2a_aux(next).concat(i2c(i - next * 10))
            )
        fi
    };

};


class List inherits IO
{
        isNil() : Bool
        {
                {
                --out_string("list\n");
                        true;
```

```
                }
        };

        head() : String
        {
                {
                        abort();
                        "";
                }
        };

        tail() : List
        {
                {
abort();
                        self;
                }
        };
        cons(i : String) : List
        {
                (new Cons).init(i, self)
        };
```

```
};


class Cons inherits List
{
        first : String;
        rest :  List;
        isNil() : Bool
        {
                {
                        --out_string("cons\n");
                        false;
                }
        };
        head() : String
        {
                first
        };
        tail() : List
        {
                rest
        };
        init(head : String, next : List) : List
```

```
        {
                {
                        first <- head;
                        rest  <- next;
                        self;
}
        };
};
class Main inherits IO
{
        stack : List;

        newline() : Object
        {
                out_string("\n")
        };
        prompt() : String
        {
                {
                        out_string(">");
                        in_string();
                }
```

```
        };
        display_stack(s : List) : Object
        {
                {
                        --out_string("hello\n");
                        if s.isNil() then out_string("")
                                        else
                                        {
                                                out_string(s.head())
;
                                                out_string("\n");
                                                display_stack(s.tail
());
                                        }
                fi;
                }
        };
        main():Object
        {
                ( let z : A2I <- new A2I  , stack : List <- new List
 in
                        while true loop
```

```
                        ( let s : String <- prompt() in
                                if s = "x" then
                                        abort()
                                else
                                        if s = "d"  then
                                                display_stack(stack)
                                        else
                                            if s = "e"  then
                                                        {
                                                                if s
tack.isNil() then out_string("")

                                                                else
                                                                if s
tack.head() = "+" then

    {

            stack <- stack.tail();

            (let a : Int <- new Int, b : Int <- new Int in
```

```
            {

                --out_string(stack.head());

                a  <- z.a2i(stack.head());

                stack <- stack.tail();

                b  <- z.a2i(stack.head());

                stack <- stack.tail();

                a <- a + b;

                --out_string(z.i2a(a));

                stack <- stack.cons(z.i2a(a));

            }

        );
```

```
    }
                                                                    else

    if stack.head() = "s" then

            {

                    stack <- stack.tail();

                    (let a : String <- new String , b : String <- ne
w String in

                            {

                                    a  <- stack.head();

                                    stack <- stack.tail();

                                    b  <- stack.head();

                                    stack <- stack.tail();
```

```
                                stack <- stack.cons(a);

                                stack <- stack.cons(b);

                            }

                    );

            }
           else
                        out_string("")

    fi

    fi
                                                                fi;
                                                        }
                                                else
                                                        stack <- sta
ck.cons(s)
                                                fi
```

```
                                        fi
                                fi
                        )
                        pool
                )
        };
```

第三次实验：

1. 统计字符个数，单词个数，行数

```
%{
    int num_char=0;
    int num_line=0;
    int num_word=0;
%}

%%
[ \t]
\n {num_line++;}
[^ \t\n]+ {num_char+=yyleng; num_word++;}
%%

int yywrap(){}
```

```
int main(){

    yylex();

    printf("num_char=%d\nnum_word=%d\nnum_line=%d\n",num_char,num_wo

rd,num_line);

    return 0;

}
```

被统计文件

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ cat word.txt
dasicz
2434
asfdfd
vcx
0189
```

运行结果

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ flex count.cl
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ gcc lex.yy.c -o mycount
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ ./mycount < word.txt
num_char=23
num_word=5
num_line=5
```

2. 统计 if 语句

```
%{

int num_if=0;

%}



%%

(if) {num_if++;}

%%

int yywrap(){}



int main (){

yylex();
```

```
printf("num_ if=%d\n",num_if);

return 0;

}
```

被统计文件

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ cat if.txt
if weather permit, we will go to the park tomorrow.
listen to the tune see if you can remember the words.
if he wins and it's a big if he'll be the first Englishman to win for twenty years.
```

运行结果

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ flex count_if.cl
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ gcc lex.yy.c -o mycount_if
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ ./mycount_if < if.txt
 weather permit, we will go to the park tomorrow.
listen to the tune see  you can remember the words.
 he wins and it's a big  he'll be the first Englishman to win for twenty years.
num_ if=4
```

3. 多重入口

```
%{

#include<stdio. h>

%}

%start AA BB CC

%%

^a {ECHO; BEGIN AA; }

^b {ECHO; BEGIN BB; }

^C { ECHO; BEGIN CC; }

\n|(\t)+|"'

"+ {ECHO;BEGIN 0;}

<AA>magic {printf("first\n"); }

<BB>magic {pr intf(" second\n");}

<CC>magic {printf("third\n");}

magic {printf(" zero\n");}
```

```
%%

int yywrap(){}

int main ( ){

pr intf("please tpe any text(ctrl+d to quit)\n");

yyLex();

return ;
```

运行结果

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ flex multiply_entries.cl
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ gcc lex.yy.c -o mymulent
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ ./mymulent
please type any text(ctrl+d to quit)
magic
zero

amagic
afirst

bmagic
bsecond

cmagic
cthird
```

第四次实验

1. cool.flex 文件

```
/*

 *  这个文件用来生成一个 COOL 语言的词法分析程序.

 */



/*

 *  lex 文件的第一个部分，也就是包含在“%{”和“ %}”之间的部分，是用来像未来

的词法分析程序输出代
```

```
 *  码的，也就是说这里的需要 include 头文件，extern 外部变量，因为这部分是
要直接照搬到以后的.c 文件中去的
 */
%{
#include <cool-parse.h>  //记号的定义放在 cool-parse.h 文件中
#include <stringtab.h>
#include <utilities.h>

/* 词法分析程序需要的宏定义 */
#define yylval cool_yylval
#define yylex  cool_yylex

/* 字符串常量的最大长度 */
#define MAX_STR_CONST 1025
#define YY_NO_UNPUT   /* 让 g++的编译结果变得友好 */

extern FILE *fin; /* 从这个文件指针读取记号 */

/* 定义 YY_INPUT 以后我们就可以从 fin 中读取记号了:
 */
#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]; /* 记录字符串的字符数组*/
char *string_buf_ptr;

extern int curr_lineno;
extern int verbose_flag;

extern YYSTYPE cool_yylval;

/*
 *  在这里添加你自己的头文件和变量
 */

int uniqueIndex =1;

%}

/*
 * 第二部分用来定义正则表达式需要的“元素”
```

```
 */

/* 下面是我们给出的基本实例 */
TYPEID          [A-Z]+[_A-Za-z0-9]*
OBJECTID        [a-z]+[_a-zA-Z0-9]*

STR_CONST       \".*\"
INT_CONST       [0-9]+

WHITE           [ \t]+
LINE            \n

%s              MutiCom

%%/*第二部分结束*/

"--"[^\n]*        {/* 忽略掉一行的注释 */}
"(*"              {BEGIN(MutiCom);}

<MutiCom>[^\n*]*   {}
<MutiCom>"*"+[^*)\n]* {}
<MutiCom>\n       {++curr_lineno;}
```

```
<MutiCom>"*"+")"  {BEGIN(INITIAL);}



"class"           {return CLASS;}

"inherits"        {return INHERITS;}


"if"              {return IF;}

"else"            {return ELSE;}

"then"            {return THEN;}

"fi"              {return FI;}


"let"             {return LET;}

"in"              {return IN;}


"while"           {return WHILE;}

"loop"            {return LOOP;}

"pool"            {return POOL;}


"case"            {return CASE;}

"esac"            {return ESAC;}

"of"              {return OF;}
```

```
"new"             {return NEW;}


"isvoid"          {return ISVOID;}


"true"            {
                     cool_yylval.boolean = 1;
                     return BOOL_CONST;
                  }
"false"           {
                     cool_yylval.boolean=0;
                     return BOOL_CONST;
                  }
"not"             {return NOT;}


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


"self"            {
```

```
                    cool_yylval.symbol=new IdEntry(yytext,strlen(yy
text),uniqueIndex++);
                    return OBJECTID;
                }


"le"            {return LE;}


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


"{"             {return '{';}
"}"             {return '}';}
"("             {return '(';}
")"             {return ')';}
":"             {return ':';}
";"             {return ';';}
"+"             {return '+';}
"-"             {return '-';}
"*"             {return '*';}
"/"             {return '/';}
"="             {return '=';}
"<"             {return '<';}
```

```
"."                 {return '.';}

"~"                 {return '~';}

","                 {return ',';}

"@"                 {return '@';}



{TYPEID}            {

                      cool_yylval.symbol = new IdEntry(yytext,strlen(y

ytext),uniqueIndex++);

                      return TYPEID;

                    }


{OBJECTID}          {

                      cool_yylval.symbol = new IdEntry(yytext,strlen(y

ytext),uniqueIndex++);

                      return OBJECTID;

                    }



{STR_CONST}         {

                      cool_yylval.symbol = new StringEntry(yytext,strl

en(yytext),uniqueIndex++);

                      return STR_CONST;
```

```
                    }


{INT_CONST}         {
                      cool_yylval.symbol = new IntEntry(yytext,strlen(
yytext),uniqueIndex++);
                      return INT_CONST;
                    }


{WHITE}             {}


{LINE}              {curr_lineno++;}


.                  {
                      cool_yylval.error_msg = new char[strlen(yytext+1
)];
                      strcpy(cool_yylval.error_msg,yytext);
                      return ERROR;
                    }


%%


int yywrap()
```

```
{

    return 1;

}
```

Hello_world.cl 文件

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/2020编译第三次实验$ cat ../编译第一次实验/cool/co
ol/examples/hello_world.cl
class Main inherits IO {
   main(): SELF_TYPE {
        out_string("Hello, World.\n")
   };
};
```

2. 使用 make 指令生成的 lexer 和标准词法分析器分别对 hello_world.cl 进行词法分析

```
syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA2
Could not open input file ../examples/hello_world.cl
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA2$ ./reference-lexer ../..
/examples/hello_world.cl
#name "../../examples/hello_world.cl"
#1 CLASS
#1 TYPEID Main
#1 INHERITS
#1 TYPEID IO
#1 '{'
#2 OBJECTID main
#2 '('
#2 ')'
#2 ':'
#2 TYPEID SELF_TYPE
#2 '{'
#3 OBJECTID out_string
#3 '('
#3 STR_CONST "Hello, World.\n"
#3 ')'
#4 '}'
#4 ';'
#5 '}'
#5 ';'
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA2$ ./lexer ../../examples/
hello_world.cl
#name "../../examples/hello_world.cl"
#1 CLASS
#1 TYPEID Main
#1 INHERITS
#1 TYPEID IO
#1 '{'
#2 OBJECTID main
#2 '('
#2 ')'
#2 ':'
#2 TYPEID SELF_TYPE
#2 '{'
#3 OBJECTID out_string
#3 '('
#3 STR_CONST "Hello, World.\\n"
#3 ')'
#4 '}'
#4 ';'
#5 '}'
#5 ';'
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA2$
```

经对比发现，二者输出结果相同，都对目标代码进行了词法分析

第五次实验

1. 简易计算器

token.l 文件

```
%{
#include "y.tab.h"
%}

%%
[0-9]+ {yylval=atoi(yytext); return T_NUM;}
[-/+*()\n] {return yytext[0];}
. {return 0;}
%%

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

parser.y 文件

```
%{
#include<stdio.h>
extern int yylex();
extern int yyparse();
void yyerror(const char* msg){}
%}

%token T_NUM
%left '+' '-'
```

```
%left '*' '/'



%%

S: S E '\n' {printf("ans=%d\n",$2);}

 |          {  }

 ;



E: E '+' E {$$=$1+$3;}

 | E '-' E {$$=$1-$3;}

 | E '*' E {$$=$1*$3;}

 | E '/' E {$$=$1/$3;}

 | T_NUM {$$=$1;}

 | '(' E ')' {$$=$2;}

 ;

%%



int main (){

return yyparse();

}
```

运行结果

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ bison -vtdy parser.y | flex token
.l
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ gcc -o calculate lex.yy.c y.tab.c
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ ./calculate
4*(1+4)
ans=20
3*2
ans=6
(2+3)*(4-2)
ans=10
```

2. 布尔表达式

token.l 文件

```
%{
#include"y.tab.h"
%}

%%
true {yylval=1;return T_BOOL;}
false {yylval=0;return T_BOOL;}
&& {return T_AND;}
\|\| {return T_OR;}
! {return T_NOT;}
\( {return T_LS;}
\) {return T_RS;}
\n {return T_NEWLINE;}
. {return 0;}
%%

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

parser.y 文件

```
%{
#include<stdio.h>
```

```
extern int yylex();

extern int yyparse( );

void yyerror(const char* msg){};

%}


%token T_BOOL T_AND T_OR T_NOT T_LS T_RS T_NEWLINE


%%

S : E T_NEWLINE {printf("ans=%s\n",$1?"true":"false");}

  |             {  }

  ;


E : T T_OR T {$$=$1||$3;}

  | T {$$=$1;}

  ;


T : F T_AND F {$$=$1&&$3;}

  | F {$$=$1;}

  ;

F : T_LS E T_RS {$$=$2;}

  | T_NOT F {$$=!$2;}

  | T_BOOL {$$=$1;}
```

```
  ;

%%



int main () {

return yyparse();

}
```

运行结果

```
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ bison -vtdy parser2.y | flex toke
n2.l
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ gcc -o bool lex.yy.c y.tab.c
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第五次实验$ ./bool
true&&(false||true)
ans=true
```

第六次实验

1. 先用 lexer 对 good.cl 进行词法分析，然后分别用 make 生成的语法分析器和标准语法分析器对结果进行语法分析

```
选择syj@thinkpad-for-syj: /mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA3
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA3$ ../../bin/reference-lex
er ./good.cl | ../../bin/reference-parser
#5
_program
  #2
  _class
    A
    Object
    "./good.cl"
    (
    )
  #5
  _class
    BB__
    A
    "./good.cl"
    (
    )
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA3$ ../../bin/reference-lex
er ./good.cl | ./parser
#5
_program
  #2
  _class
    A
    Object
    "./good.cl"
    (
    )
  #5
  _class
    BB__
    A
    "./good.cl"
syj@thinkpad-for-syj:/mnt/e/Desktop/作业/编译原理/实验/编译第一次实验/cool/cool/assignments/PA3$
```

对比结果发现，二者输出相同。