**《编译原理课程设计》报告**

Cmini编译器的设计与实现

**分组序号：1**

**设计地点：文理楼102、文理楼138**

**电子邮件：619716272@qq.com**

**分组成绩：**

**任课教师：李村合**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 专业班级 | 计算机 16- 1 班 | | | | | |
| 姓名 | 王晨宇 | 祝成飞 | 张艳芳 | 闫新照 |  |  |
| 学号 | 1502010509 | 1502010803 | 1607010105 | 1601060108 |  |  |
| 成绩比例(%) | 25 | 25 | 25 | 25 |  |  |
| 成绩 |  |  |  |  |  |  |

**2019 年 5 月20 日**

**目录**

[一、课程设计目的 1](#_Toc9467884)

[二、课程设计内容 1](#_Toc9467885)

[三、课程设计原理 1](#_Toc9467886)

[四、系统需求分析 5](#_Toc9467887)

[4.1 MiniC编译程序总体介绍 5](#_Toc9467888)

[4.2 功能需求 5](#_Toc9467889)

[4.3 用到的主要关键词 6](#_Toc9467890)

[4.4 汇编成可运行文件 6](#_Toc9467891)

[4.5 Cmini语言的文法 6](#_Toc9467892)

[五、系统设计与实现 8](#_Toc9467893)

[5.1 Cmini编译程序主要功能模块 8](#_Toc9467894)

[5.2 词法分析子程序 8](#_Toc9467895)

[5.3 语法分析 10](#_Toc9467896)

[5.4 语义分析 11](#_Toc9467897)

[六、系统测试与运行结果分析 12](#_Toc9467898)

[6.1 测试程序1（测试for循环以及加法功能） 12](#_Toc9467899)

[6.2 测试程序2（测试函数定义调用以及乘法功能） 16](#_Toc9467900)

[6.3 测试程序3（测试if-else功能和逻辑运算&&功能） 22](#_Toc9467901)

[6.4 程序测试4（测试while和去注释功能） 27](#_Toc9467902)

[6.5 程序测试5（测试char功能） 30](#_Toc9467903)

[6.6 程序测试6（测试数组功能） 32](#_Toc9467904)

[6.7 程序测试7（测试break&continue） 38](#_Toc9467905)

[6.8 程序测试8（测试Switch-case） 42](#_Toc9467906)

[6.9 程序测试9（测试调用DOS窗口进行输入的功能） 45](#_Toc9467907)

[七、特色功能展示 48](#_Toc9467908)

[7.1 界面优化 48](#_Toc9467909)

[7.2 报错功能 48](#_Toc9467910)

[八、心得体会 51](#_Toc9467911)

## 一、课程设计目的

（1）根据所掌握的编译原理课程的基本知识，编写出一个 Cmini 编译器；

（2）增强阅读和编写程序的能力；

（3）理解词法分析、语法分析和语义分析在编译程序中的作用；

（4）掌握词法分析、语法分析和语义分析程序的实现方法；

（5）实现 Cmini 编译器的编辑、编译、链接和运行。

## 二、课程设计内容

用QT编写一个可视化的 Cmini 编译器，要求界面美观，使用方便，能够实现自定义 Cmini语言的编译。

 自定义的语言词法记号：

（1）数据类型：int，char，void

（2）常量：字符常量，二/八/十进制整数。

（3）语句：赋值（=），if-else,switch-case,do-while, while，for

（4）算术运算：+，－，\*，/

（5）关系运算：==，>，<，>=，<=，!=

（6）逻辑运算：&&，||，!

（7）支持函数的定义、调用

（8）支持复合语句，即 ｛｝ 包含的语句

（9）注释： C 类型的多行注释 /\* \*/ 和 C++ 类型的单行注释 //

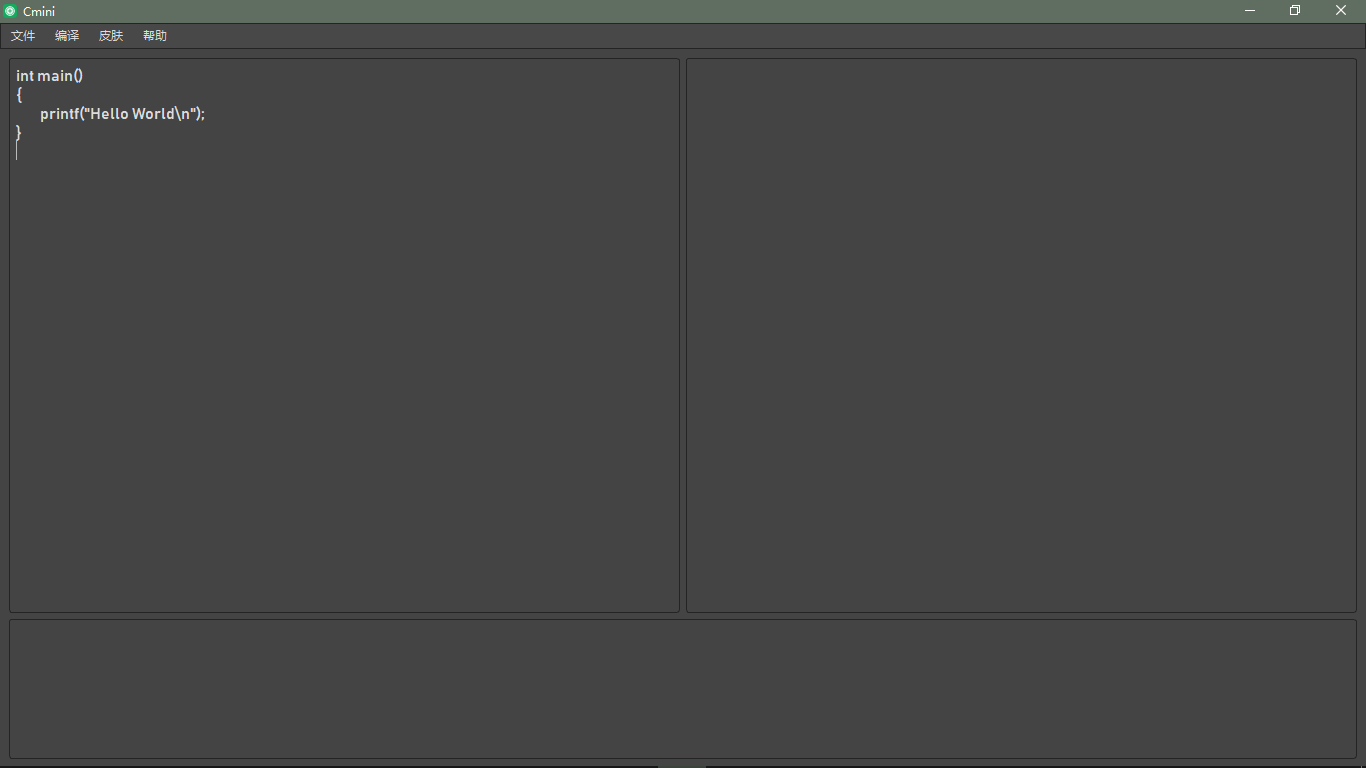
## 三、课程设计原理

MiniC 语言可以看成 C 语言的子集，它的编译程序是一个编译执行环境。

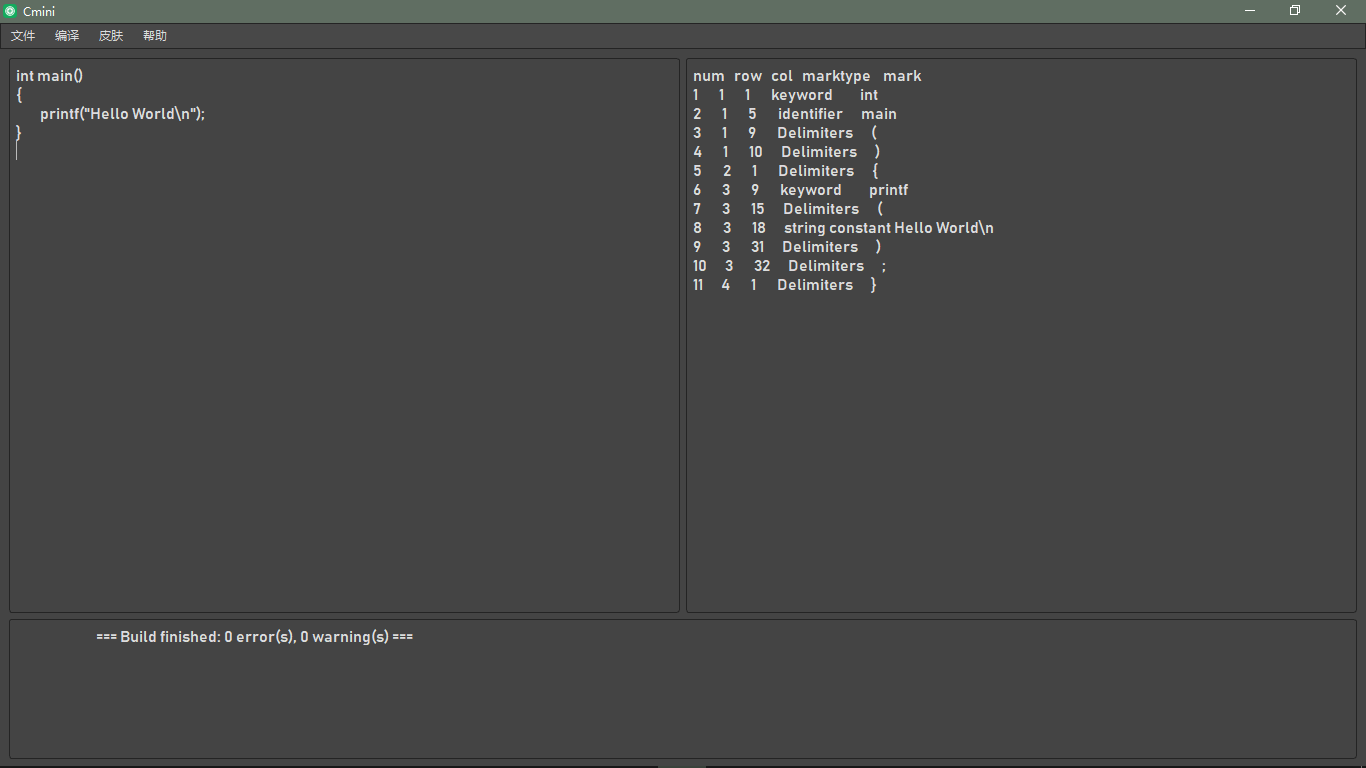
MiniC 的编译程序和目标程序的执行程序都是用 C++ 语言编写的，因此 MiniC 语言可在配备C 语言的任何机器上实现。它的编译过程采用从左到右扫描输入符号串，进行最左推导，再向右看一个符号的方式，以语法分析程序为核心，词法分析和汇编代码生成程序都是独立的过程。当语法分析需要读取输入符号串时就调用词法分析程序；而当语法分析正确后，需要生成相应的目标代码（可执行的目标文件 .exe 文件），可以通过调用汇编代码生成程序和创建EXE文件程序。

用表格管理程序建立变量、常量和过程标识符的说明与引用之间的信息联系。当源程序编译正确时，编译程序自动调用执行程序，对目标代码进行执行，并按用户程序的要求输入数据和输出运行结果。下面给出大致流程的演示：

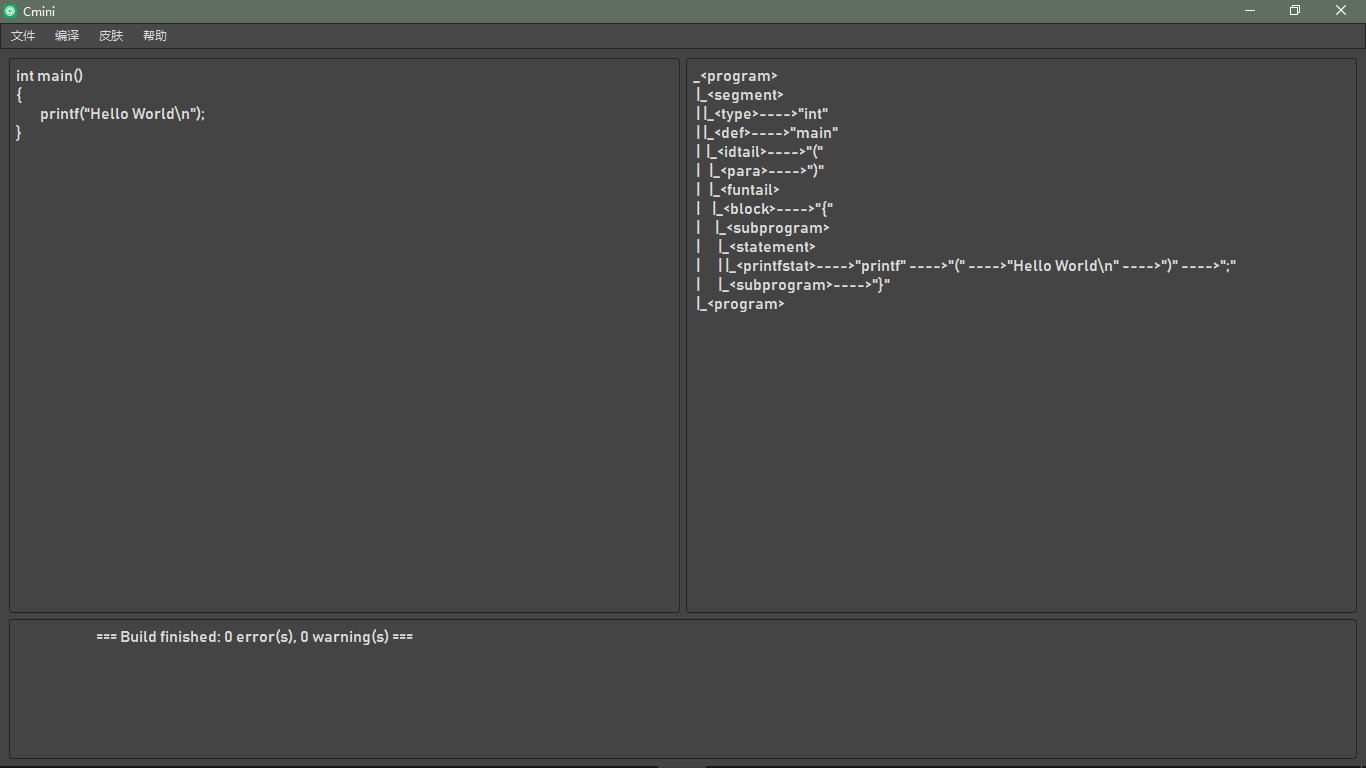
1. 运行Cmini程序，打开test.c文件。



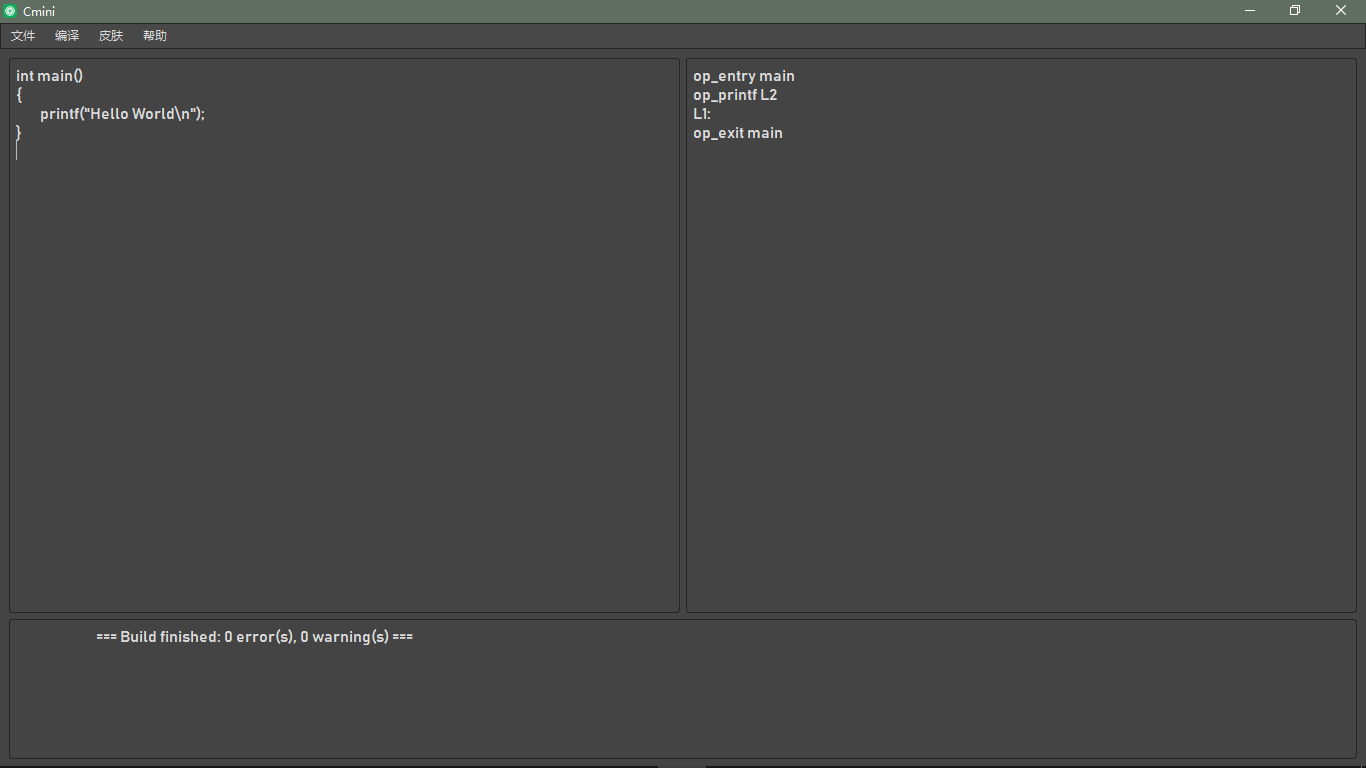
1. 选择菜单 编译-> 词法分析。



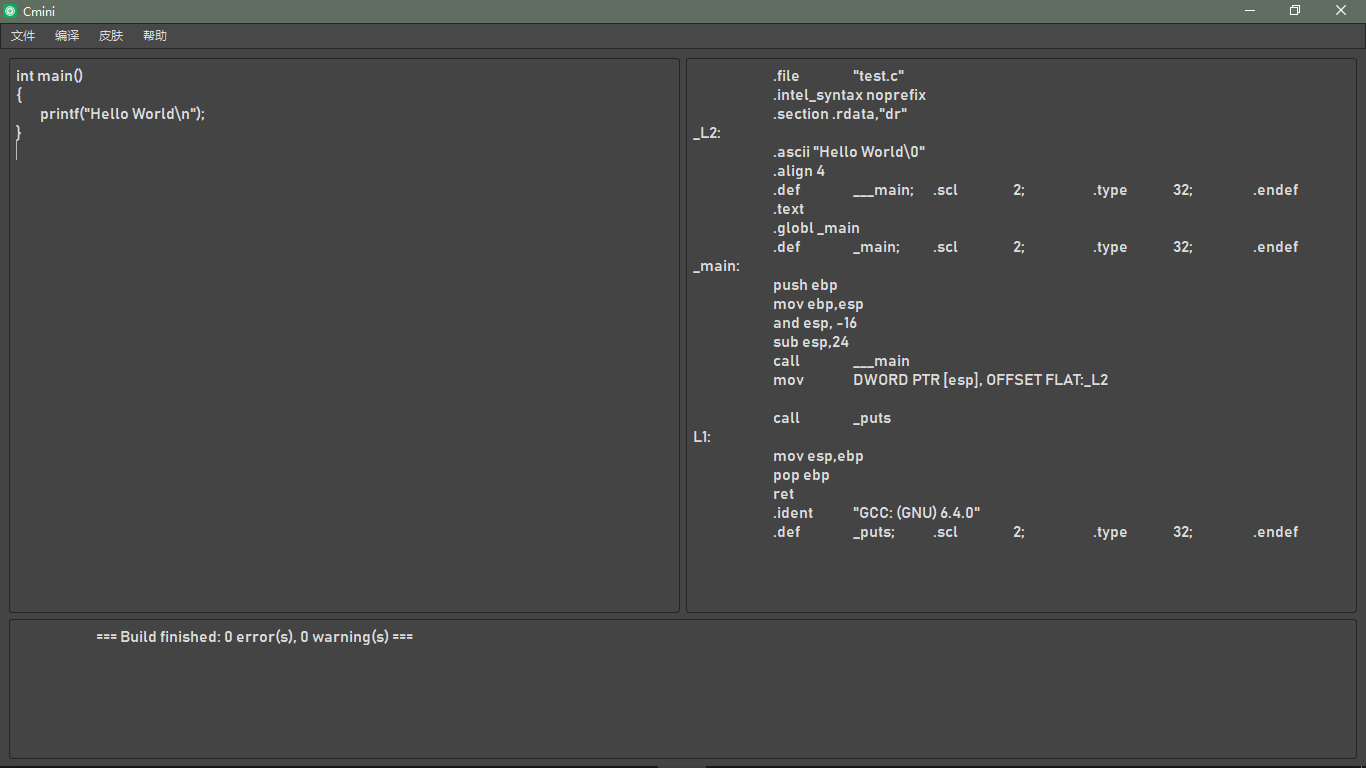
1. 选择菜单 编译-> 语法分析，生成语法树。



1. 选择菜单 编译-> 语义分析。



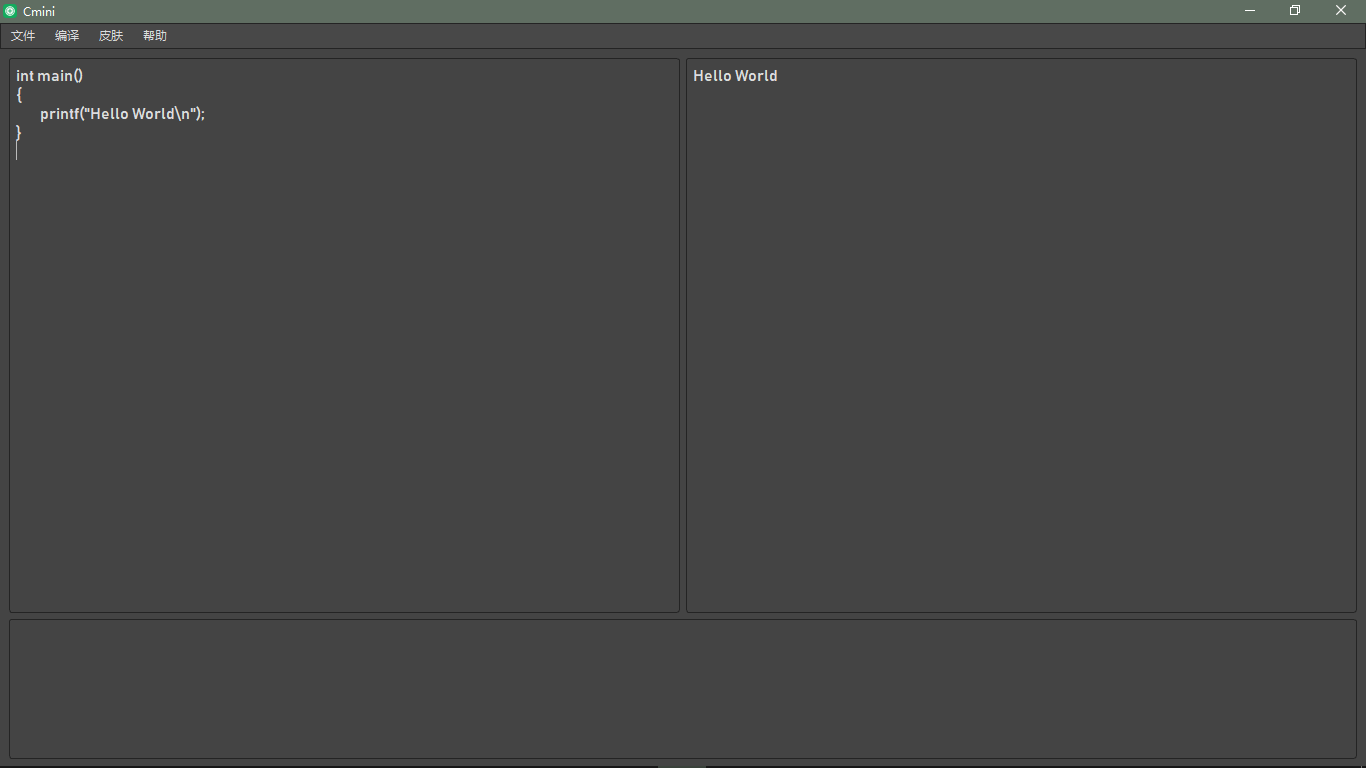
1. 选择菜单 编译-> 汇编代码



（6）选择菜单 编译-> 生成可执行文件



（7）选择菜单 编译-> 运行



## 四、系统需求分析

MiniC 语言编译过程采用从左到右扫描输入符号串，进行最左推导，再向右看一个符号的方式，对源程序分别进行 词法分析、语法分析、语义分析、目标代码生成和可执行程序生成，此外，用表格管理程序建立变量、常量和过程标识符的说明与引用之间的信息联系.用出错处理程序对词法和语法分析研究遇到的错误给出在源程序中出错的位置和错误性质，它的工作过程如下所示：

**输入源代码 -> 词法分析 -> 语法分析 -> 汇编代码生成 -> exe文件生成 -> 运行**

### 4.1 MiniC编译程序总体介绍

该编译器的运行过程是编辑源文件保存后，生成汇编代码时使用汇编代码生成程序 SymTab类，调用该类的成员函数genData() 来生成代码段和genGlbVars() 来生成数据段，源代码被翻译成NASM格式的x86汇编指令程序，接着调用genAsm(),对代码段和数据段进行整合；当然我们在汇编代码生成时应该使用语义分析子程序 genir 类对词法和文法出现的各种情况进行展开分析

### 4.2 功能需求

（1）用 C++ 语言实现了类C语言，叫做 MiniC 语言；

（2）使用 LL(1) 文法，从左到右扫描输入符号串，进行最左推导，再向右看一个符号；采用递归子程序法实现语法分析，并用 C++ 语言实现了词法分析器、语法分析器、汇编代码生成器，能直接生成 intel 80x86汇编代码。

（3）在声明中实现了对静态常量、变量和函数声明的支持；赋值语句的 = ，基本的数学运算 +、-、\*、/ ；

（4）使用函数实现了对过程的调用；

（5）使用 { } 实现复合语句；

（6）使用函数 read() 来同时读入一个或多个数据，使用函数 write() 来同时输出一个或多个数据，而函数 printf() 只能用来输出字符串；

（7）在循环分支语句中实现了if语句， if……else……语句， for(…,……,……)…语句，while……语句，switch{case….}；

（8）关系表达式包括 ==(等于)、!=(不等于)、<、<=、>、>=。

（9）支持一维数组

### 4.3 用到的主要关键词

关键字等用枚举类型囊括：

enum TokenType

{

// 关键字

\_DOUBLE, \_INT, \_ELSE, \_CASE, \_ENUM, \_REGISTER, \_TYPEDEF, \_CHAR, \_CONST, \_FLOAT, \_FOR, \_SIGNED, \_VOID, \_SIZEOF, \_VOLATILE, \_IF, \_WHILE, \_READ, \_WRITE, \_PRINTF,

// 操作符

ASSIGN, PLUS, MINUS, TIMES, DIV, MOD,

BITWISE\_AND, BITWISE\_OR, BITWISE\_NOT, LOGICAL\_NOT, LT, GT,

// 界符

LPARAN, RPARAN, LBRACE, RBRACE, LSQUARE, RSQUARE, COMMA, DOT, SEMI, COLON,

//( ) { } [ ] , . ; :

// 复杂操作符

EQ/\* == \*/, NEQ/\* != \*/, PLUS\_PLUS/\* ++ \*/, MINUS\_MINUS/\* -- \*/,

PLUS\_ASSIGN/\* += \*/, MINUS\_ASSIGN/\* -= \*/, TIMES\_ASSIGN/\* \*= \*/, DIV\_ASSIGN/\* /= \*/,

NGT/\* <= \*/, NLT/\* >= \*/, LOGICAL\_AND/\* && \*/, LOGICAL\_OR/\* || \*/,

// 其他

\_EOF, \_ID, \_NUM, \_STRING, \_CHARACTER, \_LABEL, \_ERROR, \_NONE

};

### 4.4 汇编成可运行文件

采用 intel 80x86 汇编语言，然后调用 GCC汇编成可执行文件（.exe），可在相应机型的机器上运行。

### 4.5 Cmini语言的文法

1. <program> -><segment><program>|ε

2. <segment> ->\_EXTERN <type><def> | <type><def>

3. <type> ->\_INT|\_CHAR|\_VOID

4. <defdata> ->ID <varrdef>|TIMES ID <init>

5. <varrdef>->[num]|<init>

6. <init-> ASSIGN<expr>|ε

7. <deflist> ->COMMA <defdata><deflist>|SEMICON

8. <def> ->ID <idtail> |MUL ID <init><deflist>

9. <idtail> -><varrdef><deflist> |LPAREN <para> RPAREN <funtail>

10. <funtail>-> SEMICON|<block>

11. <paradata> ->TIMES ID|ID <paradatatail>

12. <paradatatail> -> [NUM]|空

13. <paralist> ->COMMA <type><paradata><paralist>|kong

14. <para> -><type><paradata><paralist>|kong

15. <block> ->{ <subprogram>}

16. <subprogram> -><localdef><subprogram>

|<statement><subprogram>

|kong

17. <localdef> -><type><defdata><deflist>

18. <assexpr> -><orexpr><asstail>

19. <asstail> -> ASSIGN<<orexpr><asstail>>|kong

20. <orexpr>-><andexpr><ortail>

21. <printf>->(STR);

22. <write>->(<arg>)

23. <read> -> read(ID);|read(ID[NUM]);|read(\*ID)

## 五、系统设计与实现

### 5.1 Cmini编译程序主要功能模块

（1）功能一：定义扫描器类Scanner，读取源文件，按序返回文件内的字符，直到文件结束，实现函数为Scanner::scan(FILE \*filein)。

（2）功能二：实现字符序列的解析，将扫描器输出的字符序列转化为词法记号的序列，实现函数是Token\* Lex::tokenize()和void Lex::analysis()。

（3）功能三：语法分析过程的实现，主程序为void Parser::analyze()。

（4）功能四：语法树的绘制，函数为void Parser::DrawTree(bool outOpt)

（5）功能五：从声明定义、表达式和语句进行语义的检查和处理，通过Var、GenIR类中的相关函数实现

（6）功能六：生成汇编代码，函数为void SymTab::genAsm()

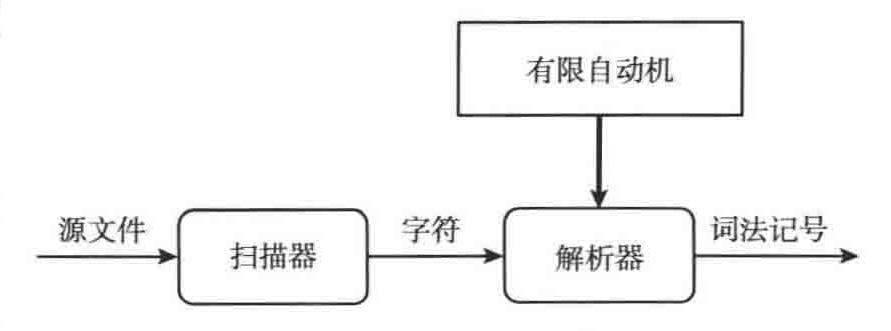
（7）功能七：词法分析，语法分析，语义分析错误处理等可以输出简单的错误警告信息

其他基本功能：  
**数据类型**：void、int、char  
**运算符**：不支持三元运算符和位运算符  
**语句**：if else、switch case、do while、while、for、break、continue  
**数组**：只支持一位数组  
**指针**：不支持指针数组，可以实现字符串初始化  
**函数调用**：支持函数调用，支持向数组传入指针和数组  
**基本输入输出**：printf、read、write  
**预处理**：不支持预处理

### 5.2 词法分析子程序

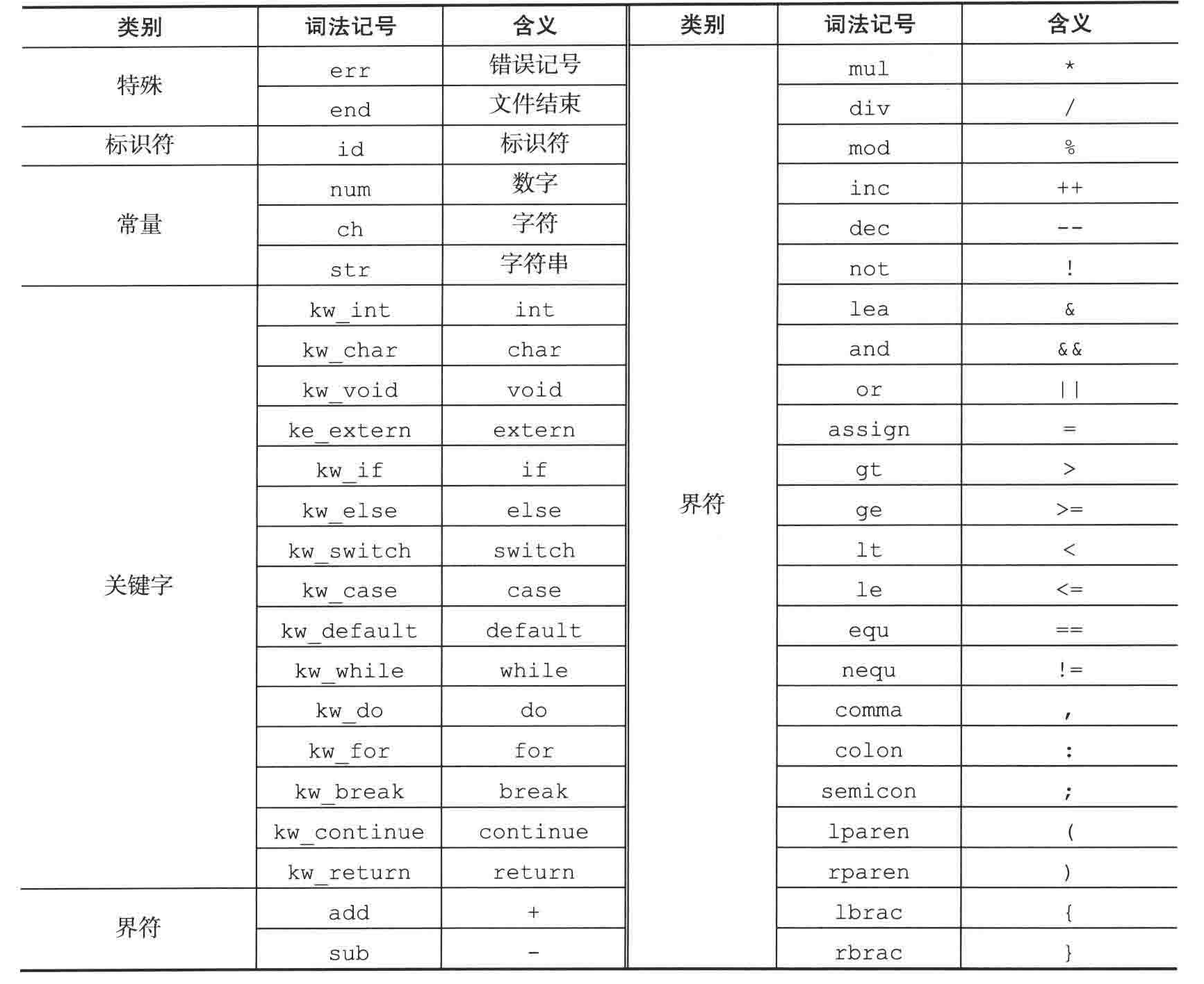
#### 5.2.1 实现方法

词法分析的功能是顺序扫描源文件内的字符，通过与词法记号的有限自动机进行匹配、产生各式各样的词法记号。实现的类主要有扫描器类Scanner、词法记号类Token、各种有限自动机类（标识符有限自动机：class Id；关键字有限自动机：class Keywords；常量有限自动机：class Str、class Char、class Num；界符有限自动机；操作符有限自动机：Operator）和解析器类Lex。



**词法分析器的分析过程：**考虑到基于表驱动的词法分析的缺点，这里采用的是基于硬编码方式的词法分析。具体的实现方法是（以标识符为例）：在Class Id中，定义变量string name用来记录接收到的字符，通过Token\* Lex::tokenize()判断字符类型，如果这个字符是字母，则继续获取字符或数字，最终可以拼成一个单词，查找static string TokenTypeArray [66]中的关键字表，对每个单词进行判断分析，最后给出其输出该单词的类型及内容。

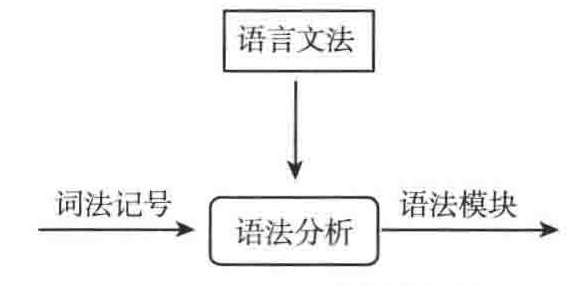
#### 5.2.2 词法记号表



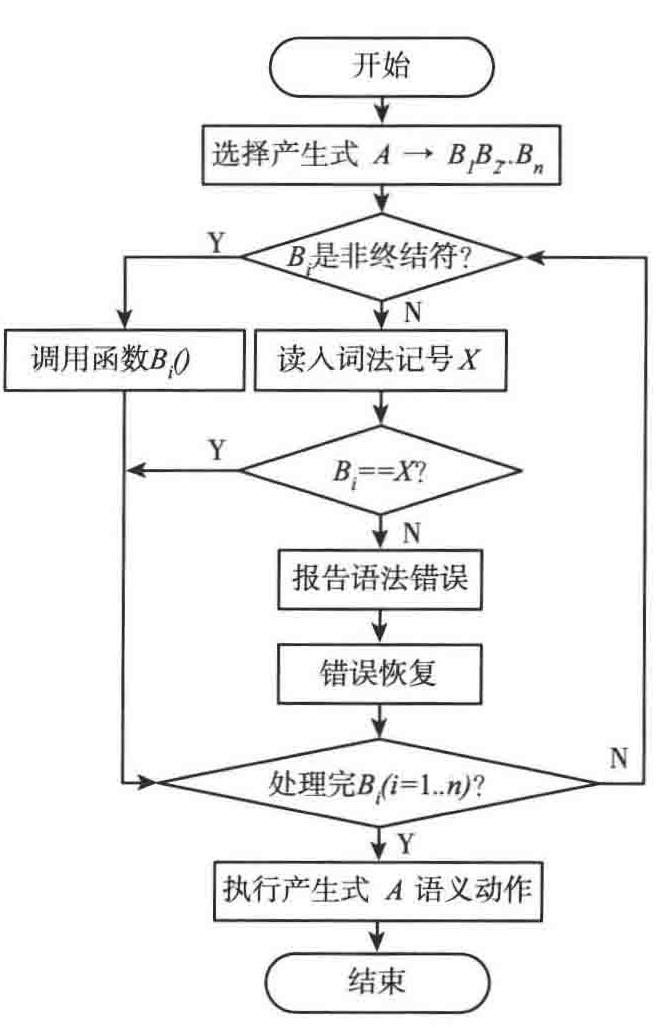
### 5.3 语法分析

#### 5.3.1 实现方法

我们使用的是LL(1)文法分析算法完成高级语言的语法分析。首先，定义好自己的文法并计算LL(1)文法的FIRST、FOLLOW和SELECT集合，构建LL(1)分析表进行语法分析。



#### 5.3.2 语法分析流程图



### 5.4 语义分析

#### 5.4.1 实现方法

语义分析与语法分析相结合，每当产生式规约成功或移进某特殊终结符时时，则根据产生式所对应的含义，进行相应的处理。最终生成相应的四元式。

#### 5.4.2语义错误与语义警告表



## 六、系统测试与运行结果分析

### 6.1 测试程序1（测试for循环以及加法功能）

**// for循环输出，打开for.txt**

int main()

{

int i;

for(i=0;i<15;i=i+1)

{

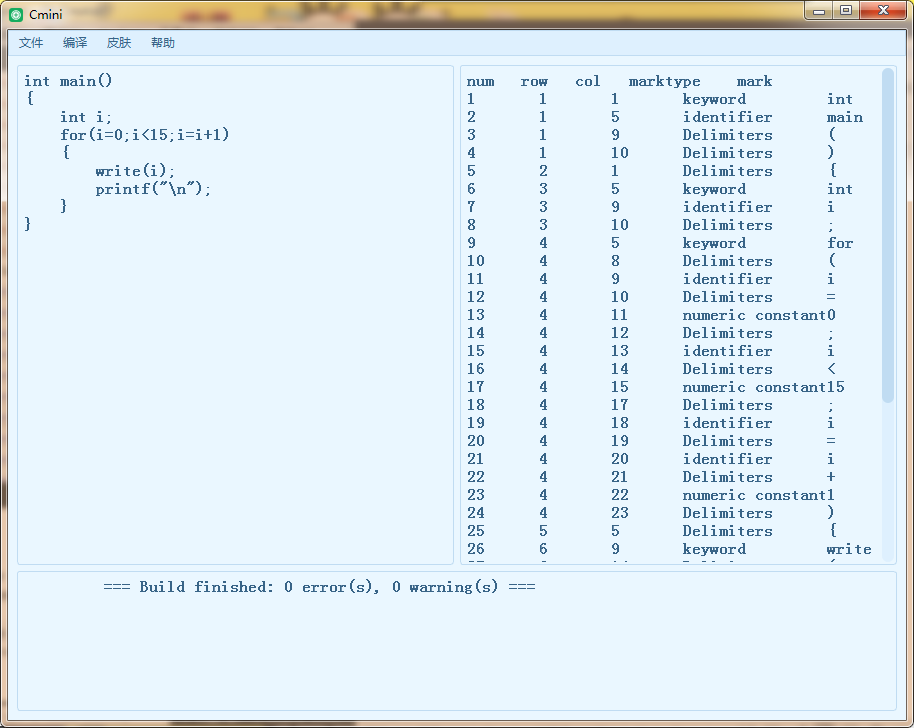
write(i);

printf("\n");

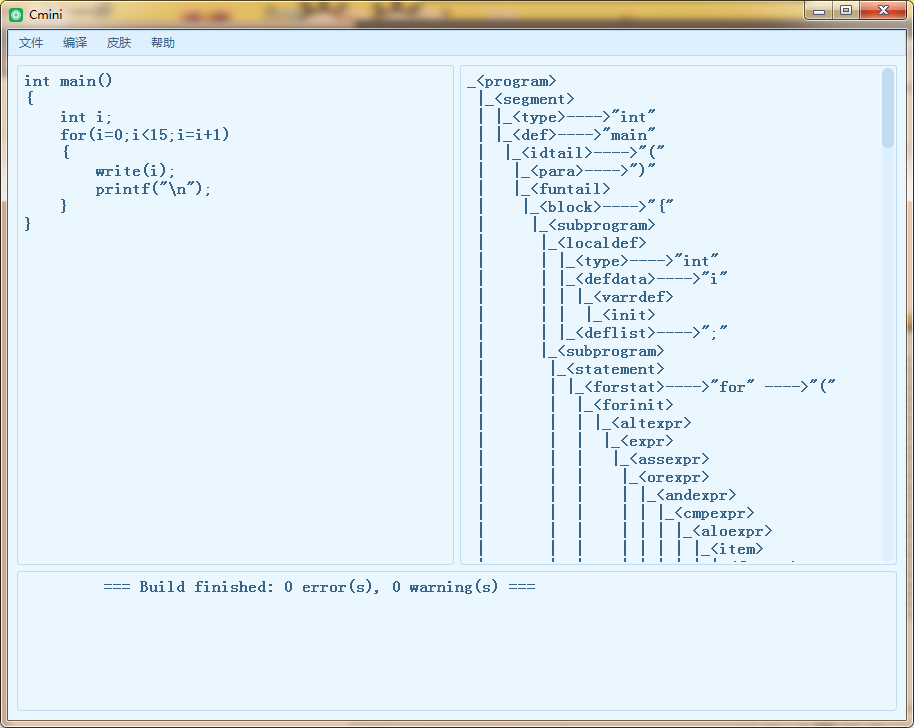
}

}

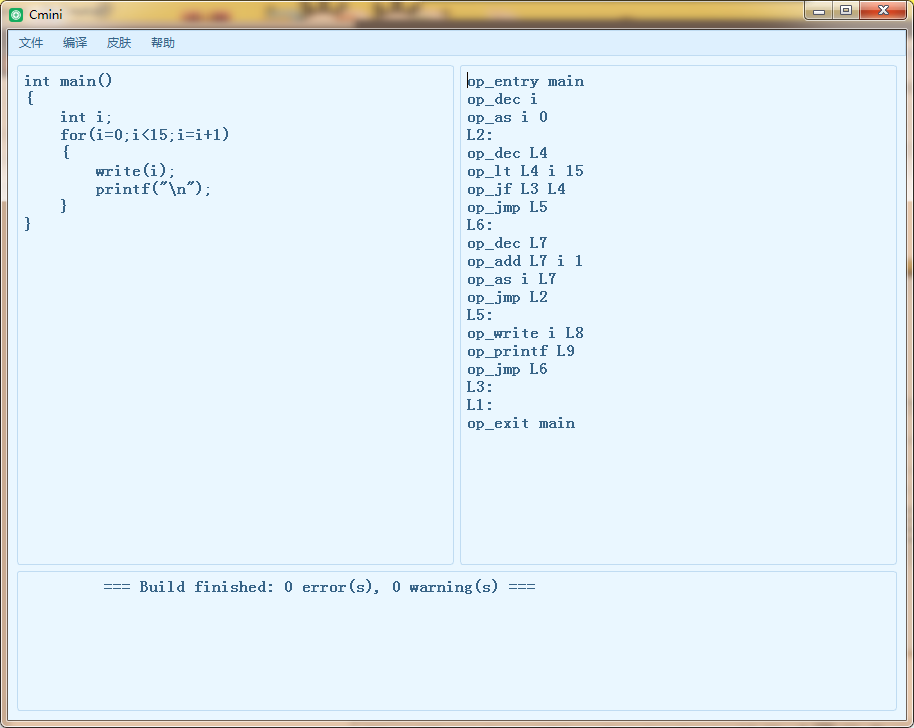
**词法分析：**



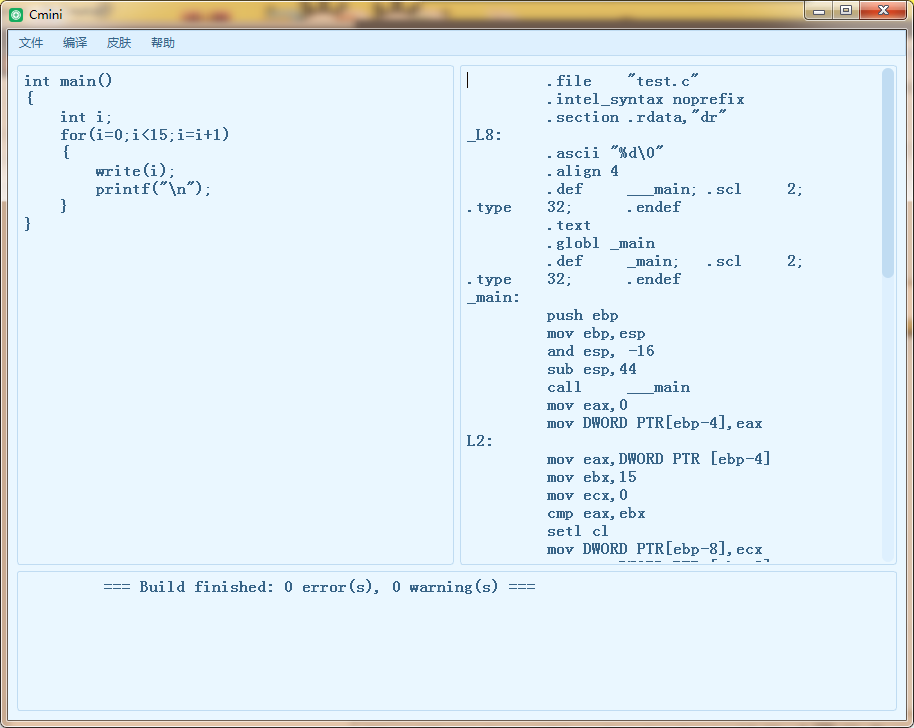
**语法分析：**



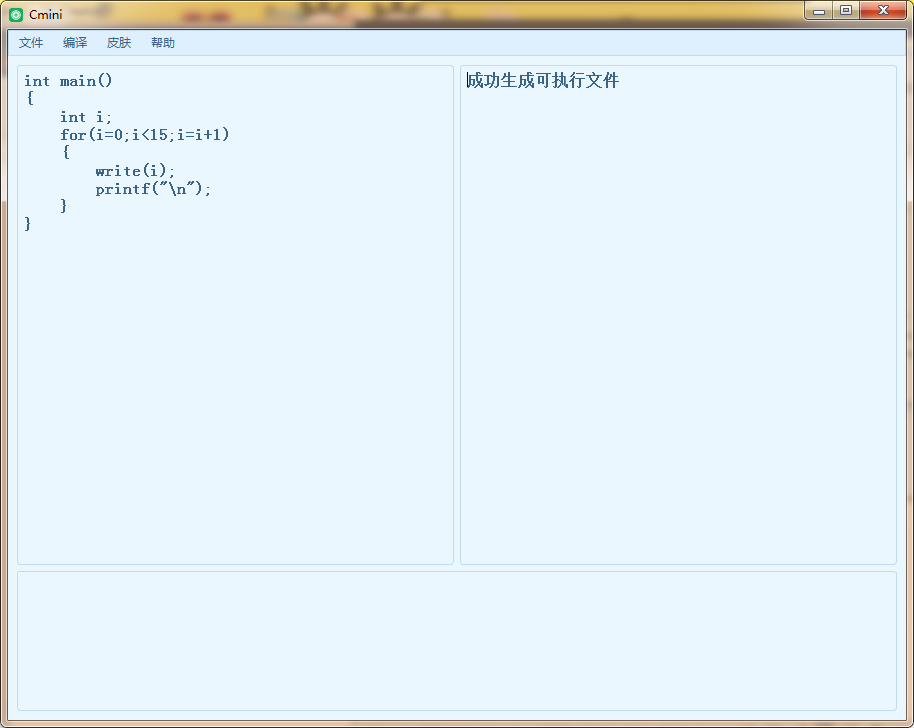
**语义分析：**



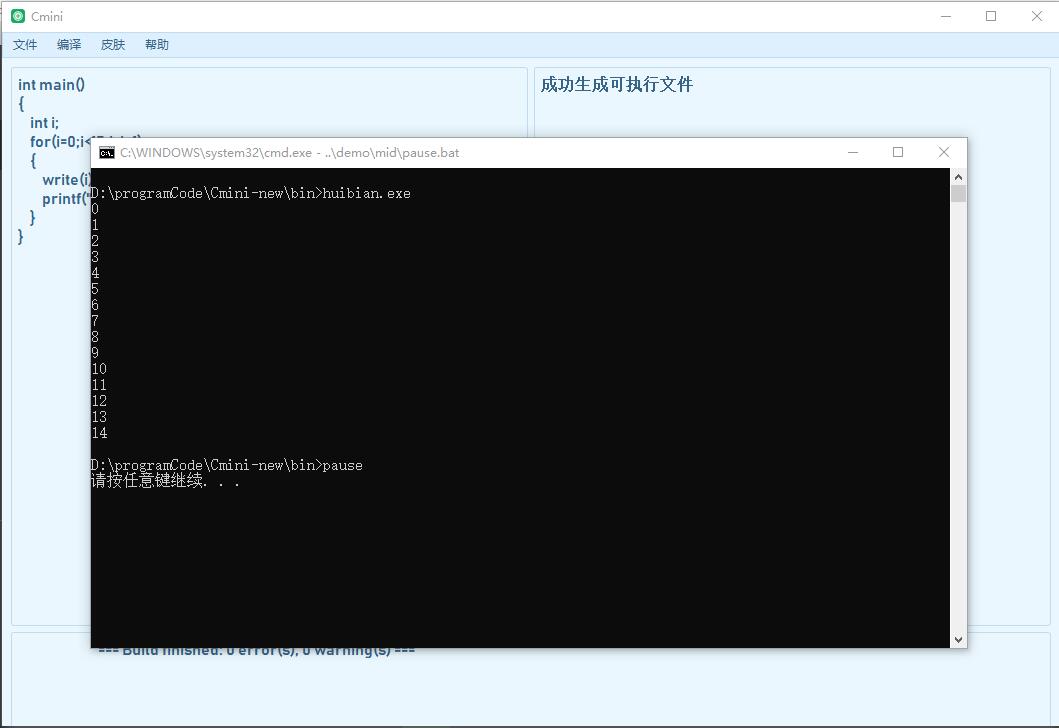
**汇编代码：**



**生成可执行文件：**



**运行：**



### 6.2 测试程序2（测试函数定义调用以及乘法功能）

**// 打开Function.txt**

int fun(int a,int b)

{

int c;

c=a\*b;

return c;

}

int main()

{

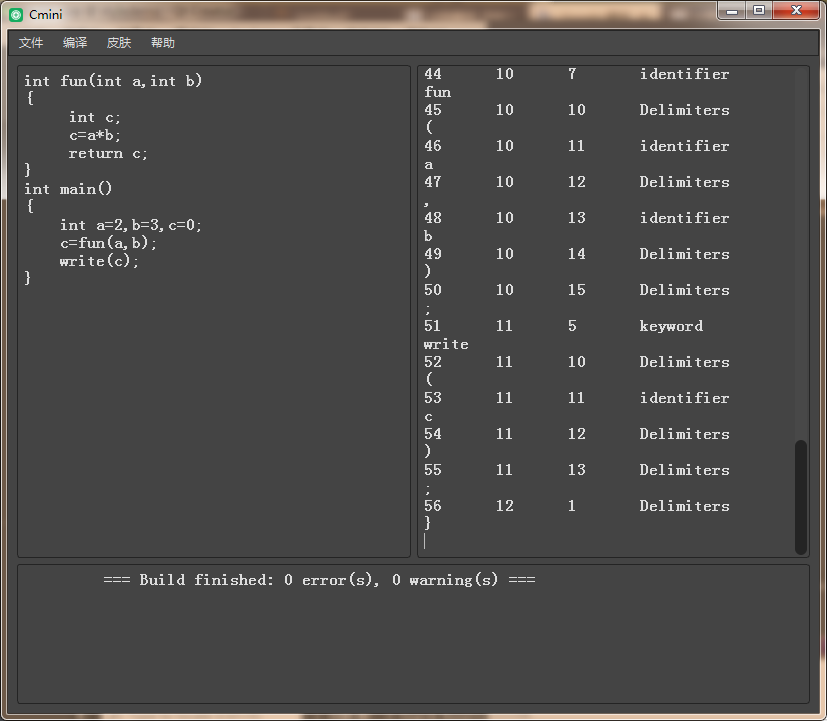
int a=2,b=3,c=0;

c=fun(a,b);

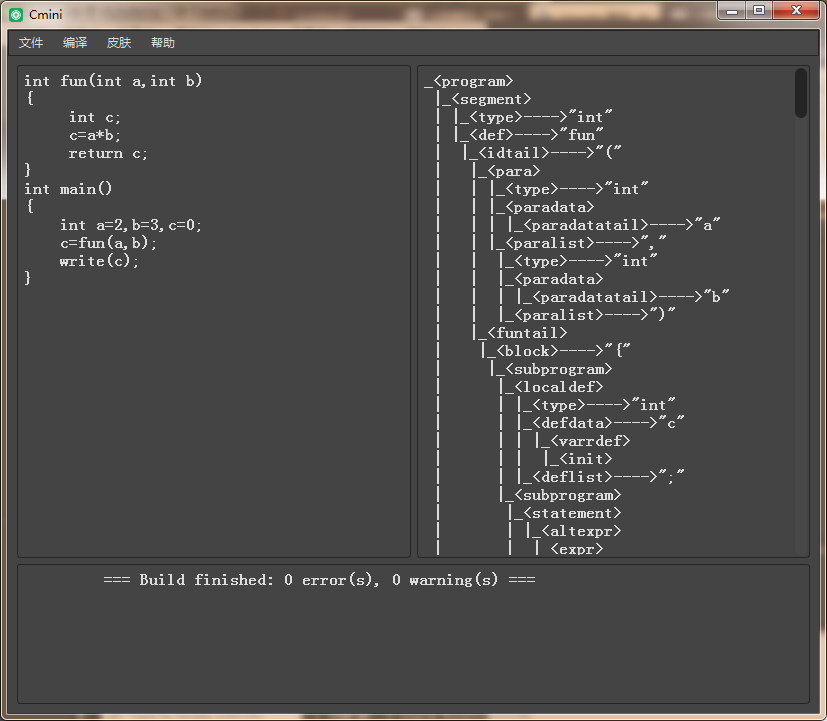
write(c);

}

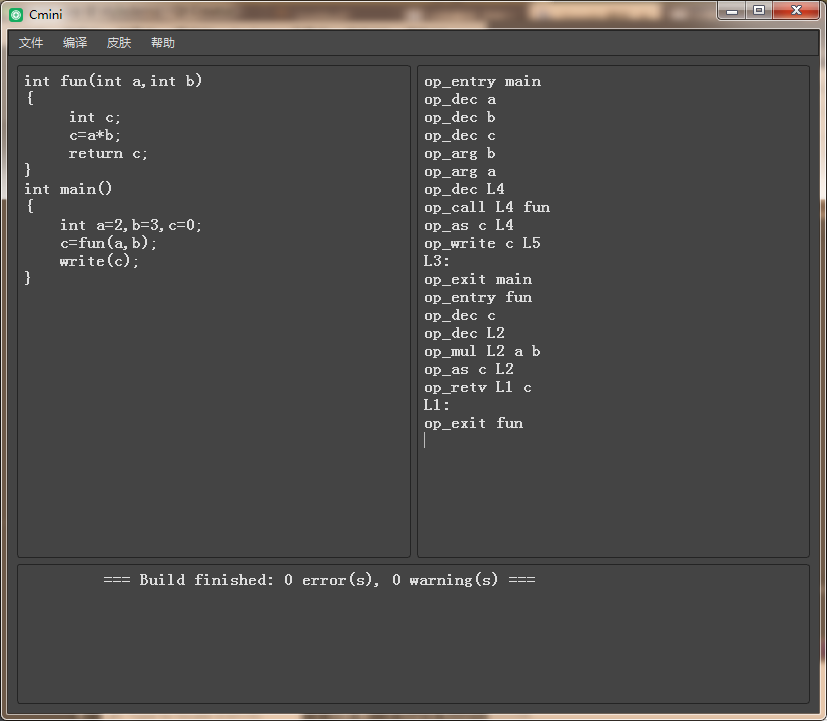
**词法分析：**



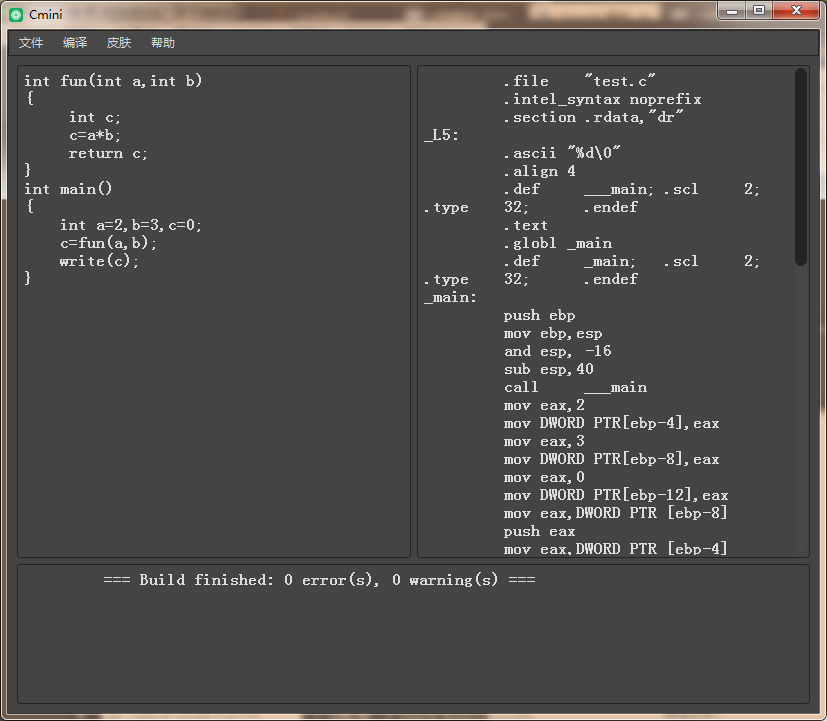
**语法分析：**



**语义分析：**



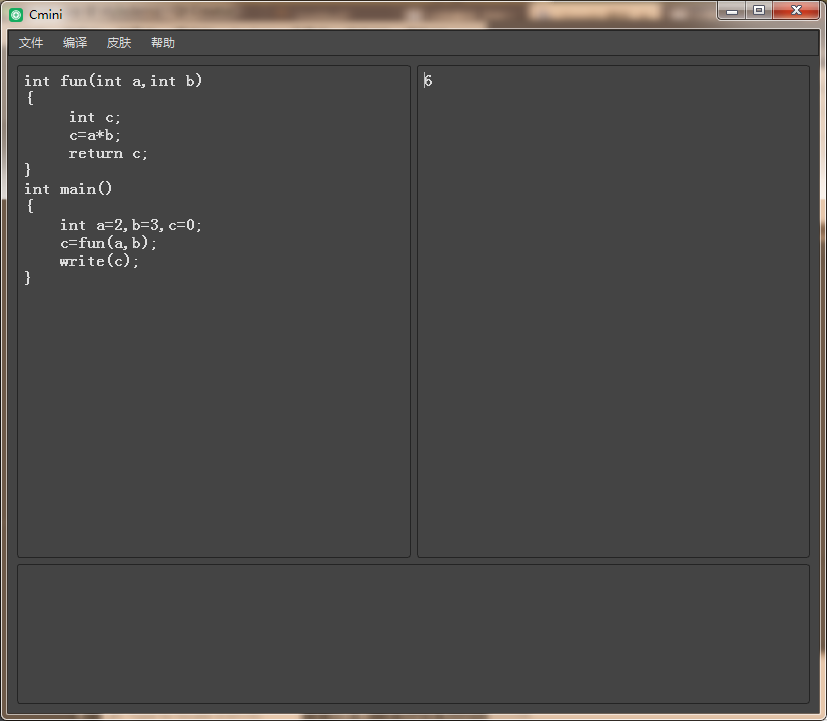
**汇编代码：**



**生成可执行文件：**



**运行：**



### 6.3 测试程序3（测试if-else功能和逻辑运算&&功能）

**// 打开if-else.txt**

int main()

{

int a=0,b=1;

if(a==0&&a<1)

{

write(a);

printf("\n");

}

else

{

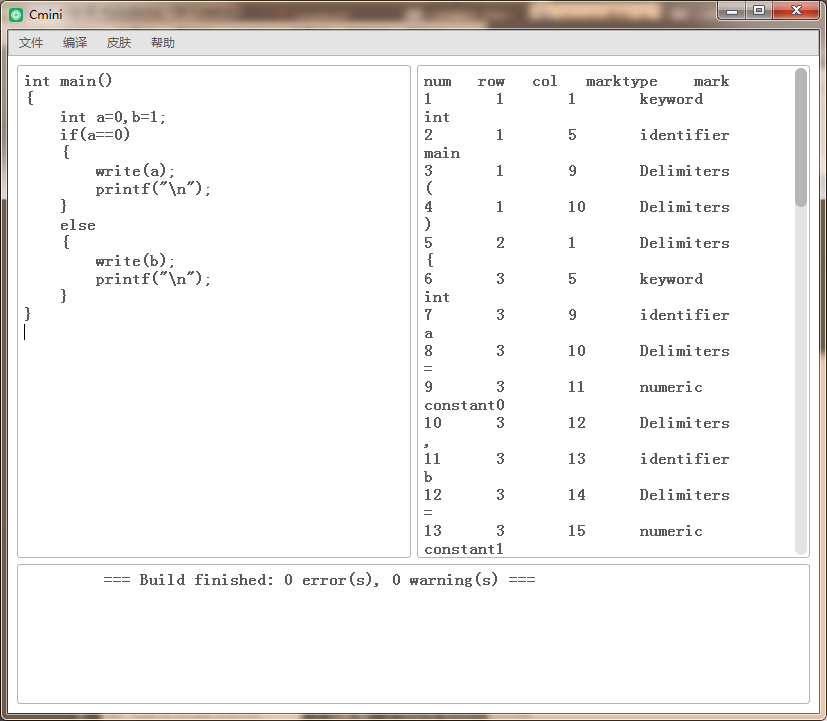
write(b);

printf("\n");

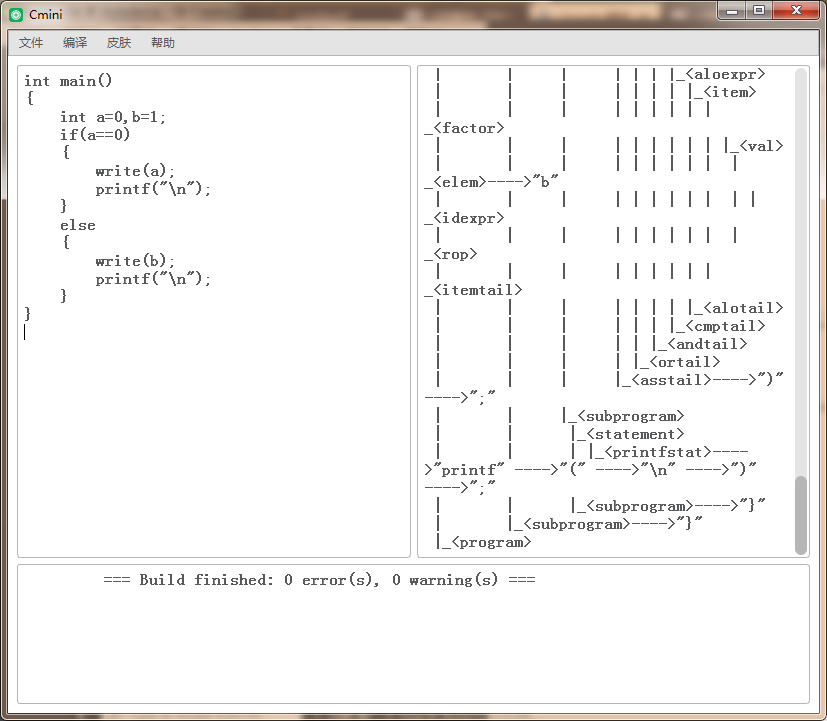
}

}

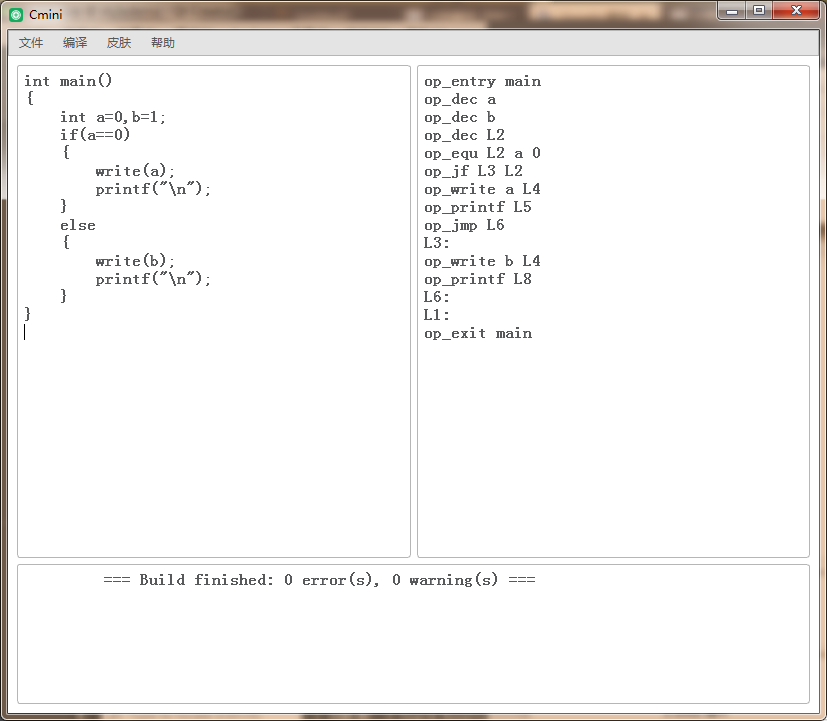
**词法分析：**



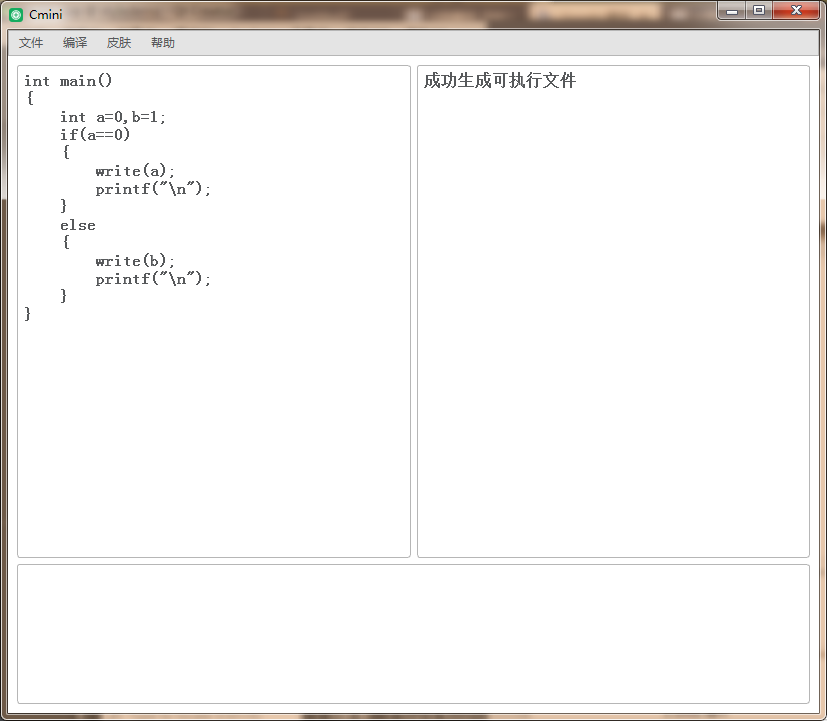
**语法分析：**



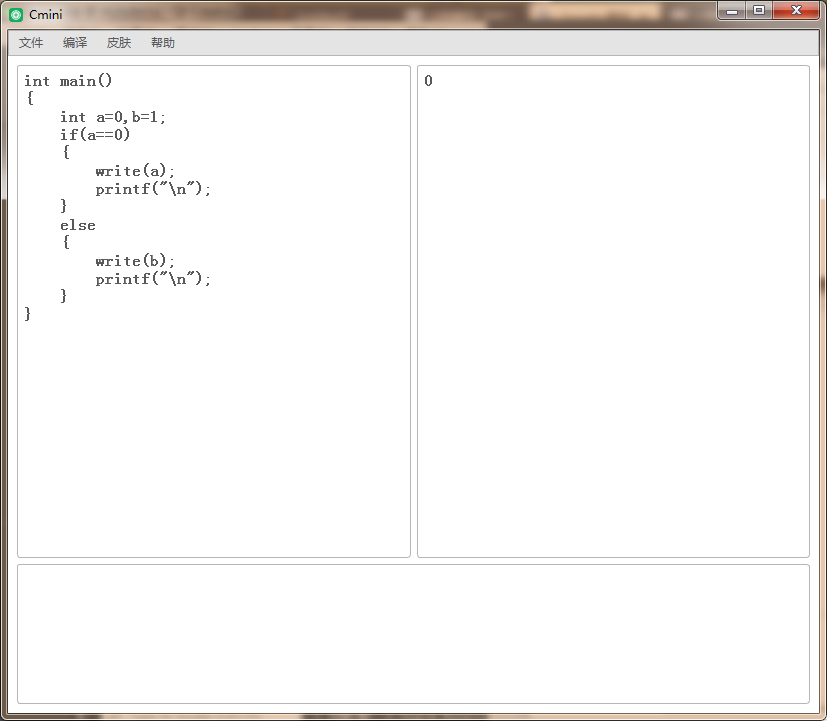
**语义分析：**



**生成可执行文件：**



**运行：**



### 6.4 程序测试4（测试while和去注释功能）

//打开while.txt,

int main()

{

int a=1,b=2;//定义变量

while(a==1)

{

b=3;

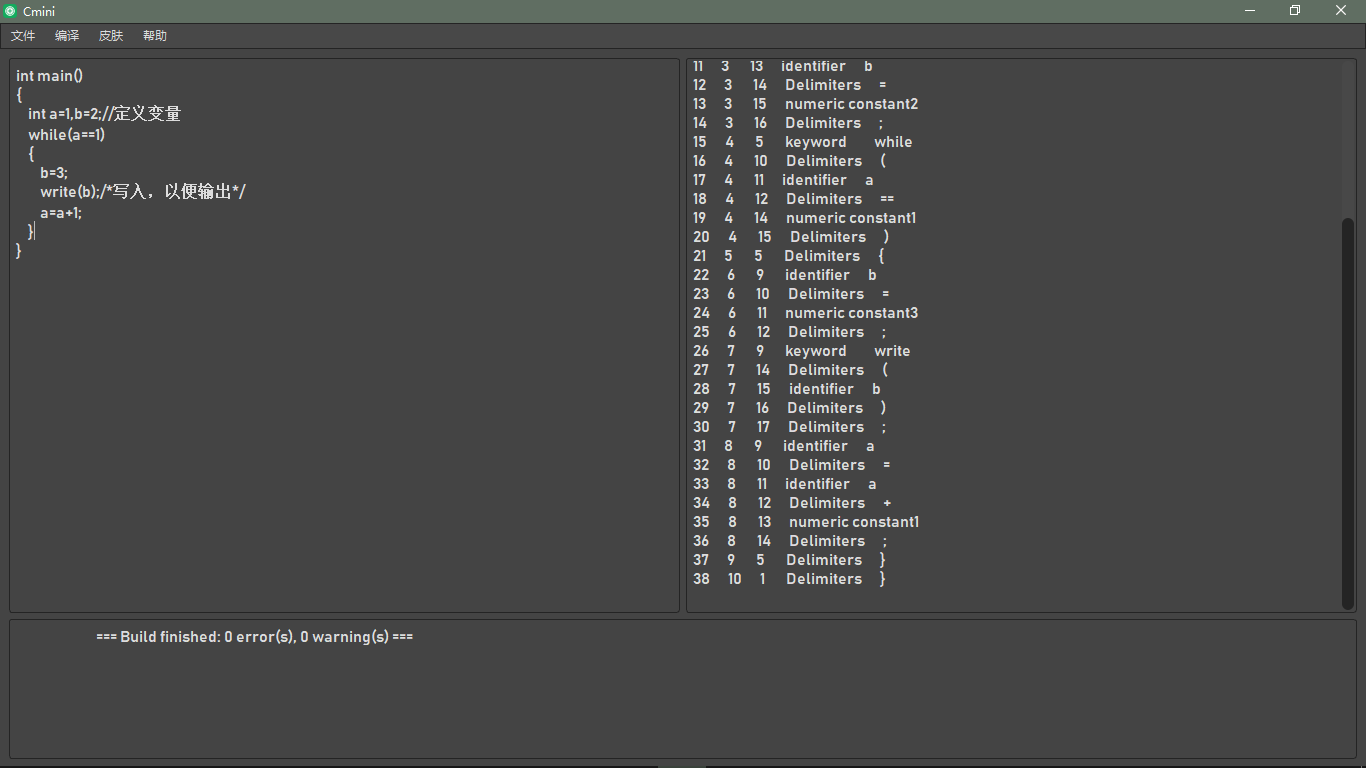
write(b);/\*写入，以便输出\*/

a=a+1;

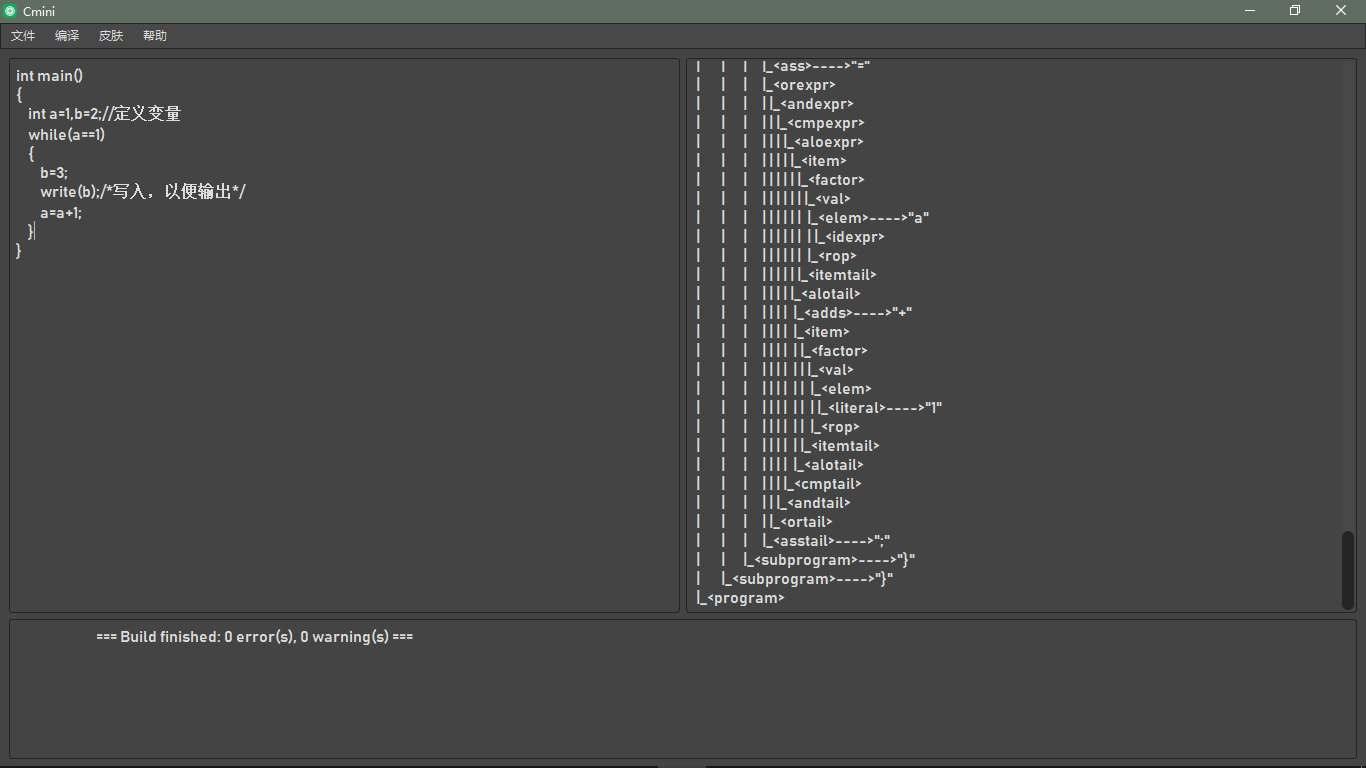
}

}

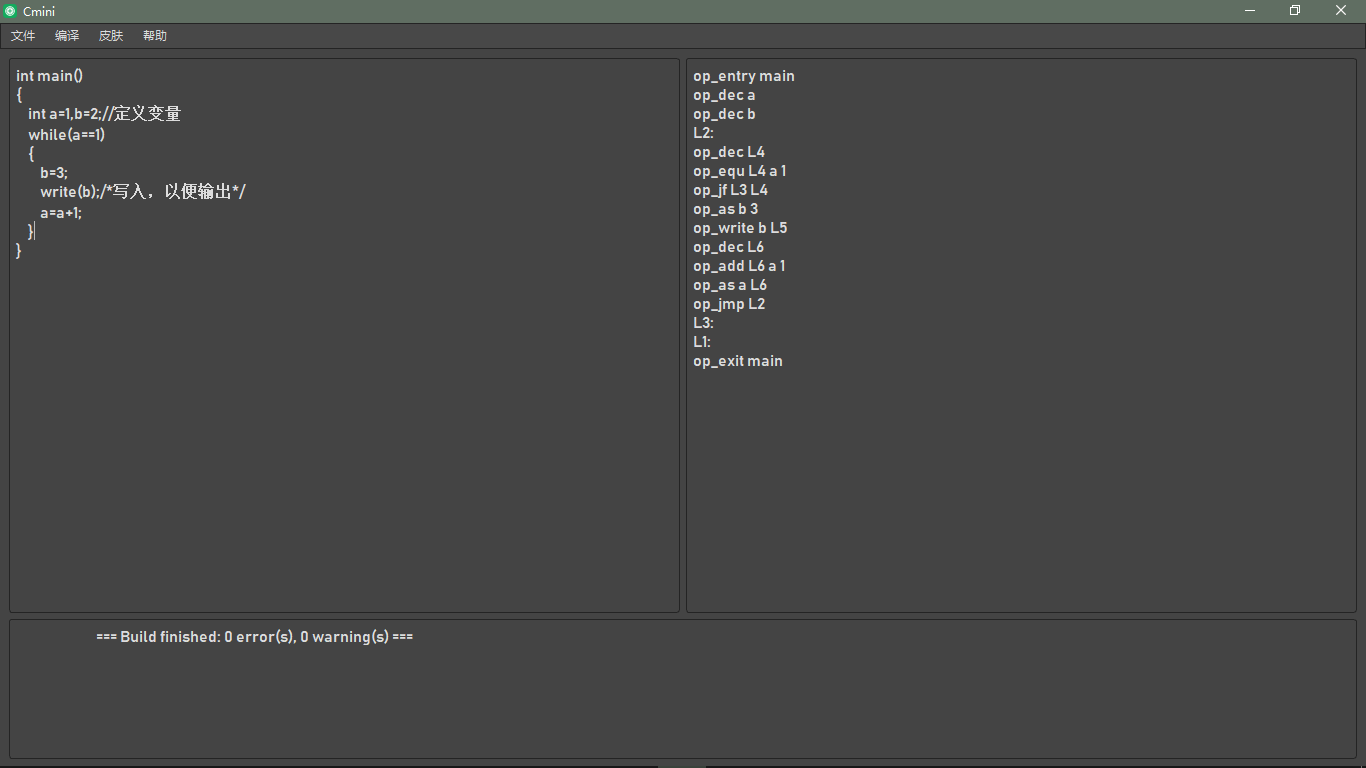
**词法分析：**



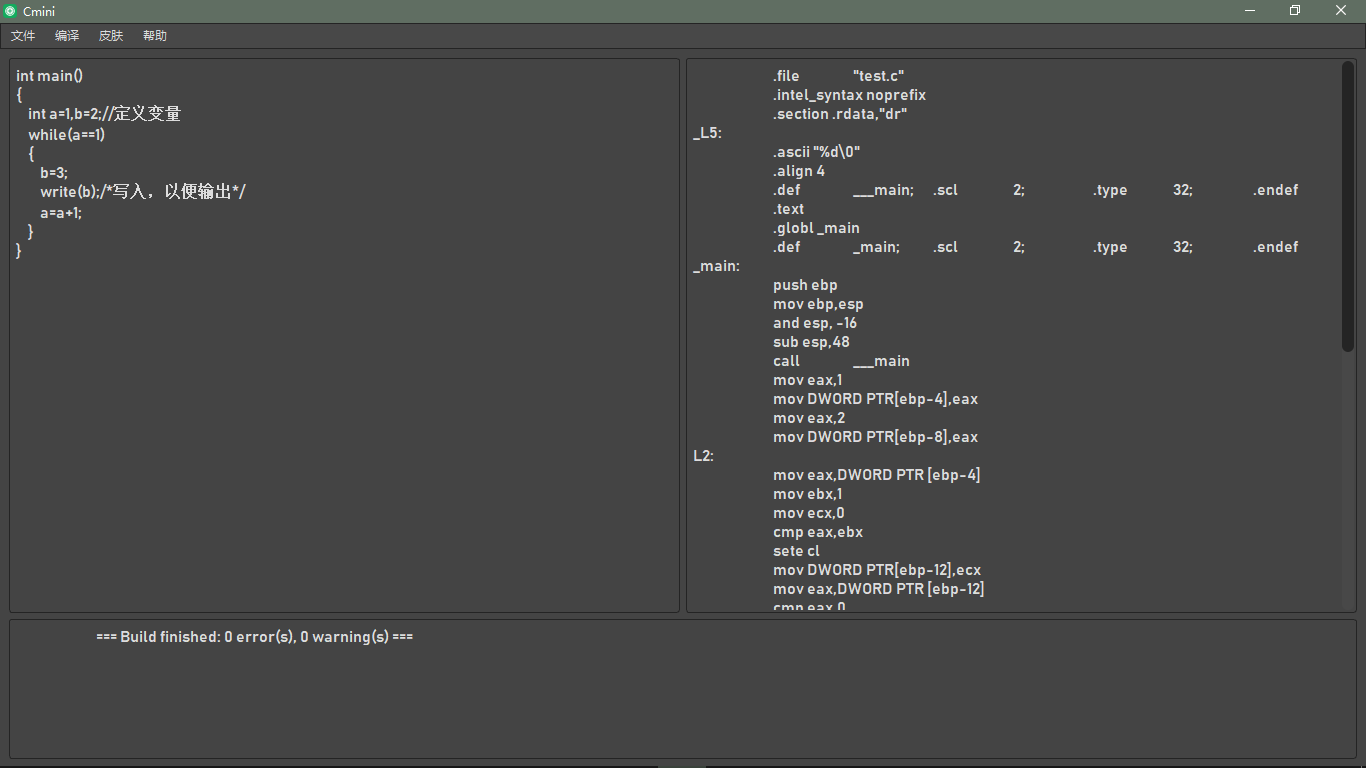
**语法分析：**



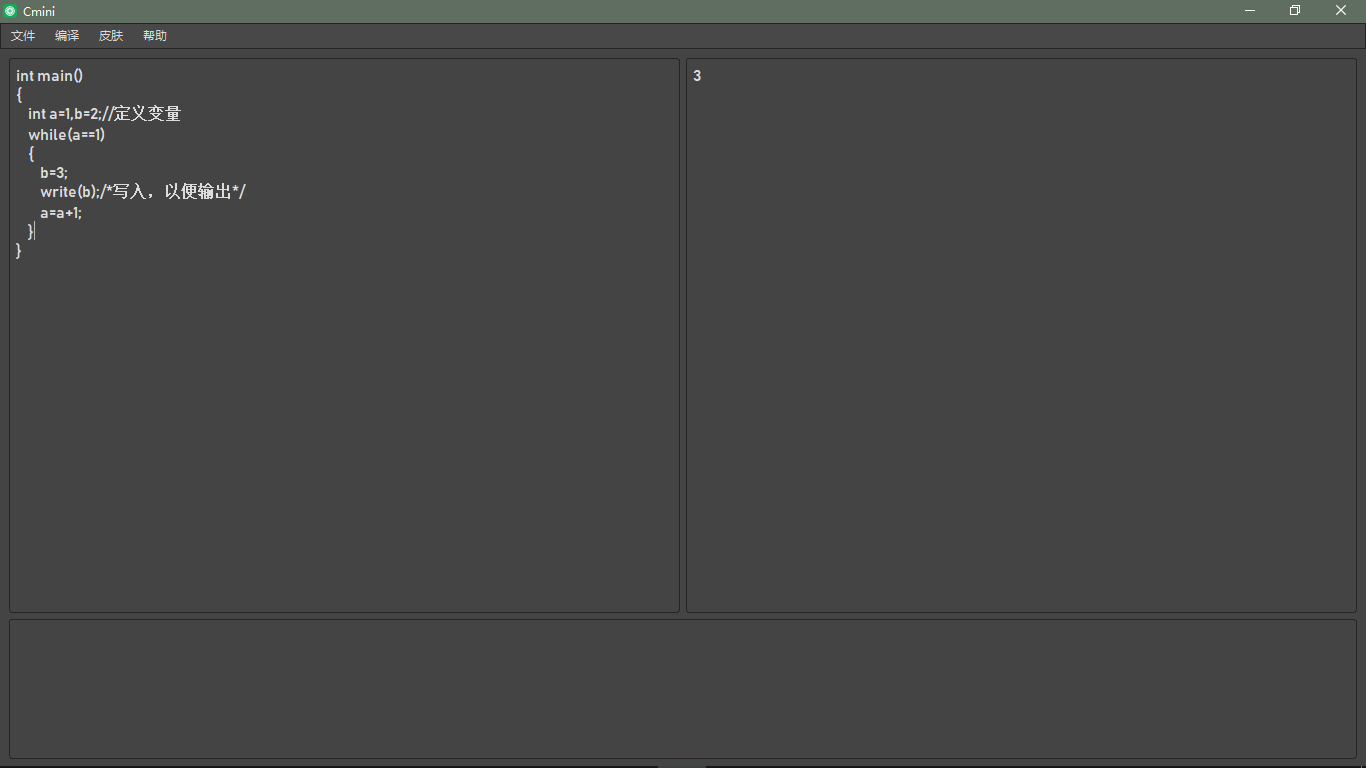
**语义分析：**



**汇编代码：**



**运行：**



### 6.5 程序测试5（测试char功能）

//**打开char.txt文**

int main()

{

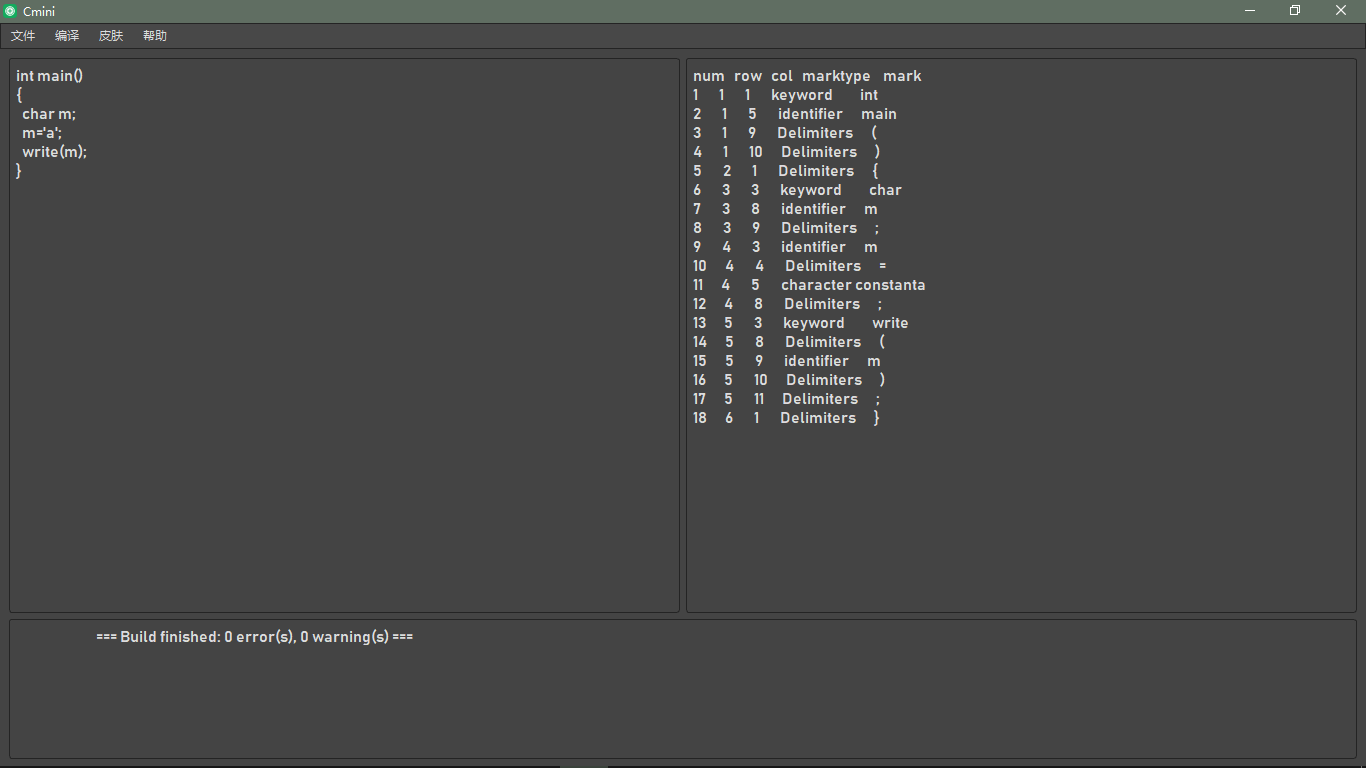
char m;

m='a';

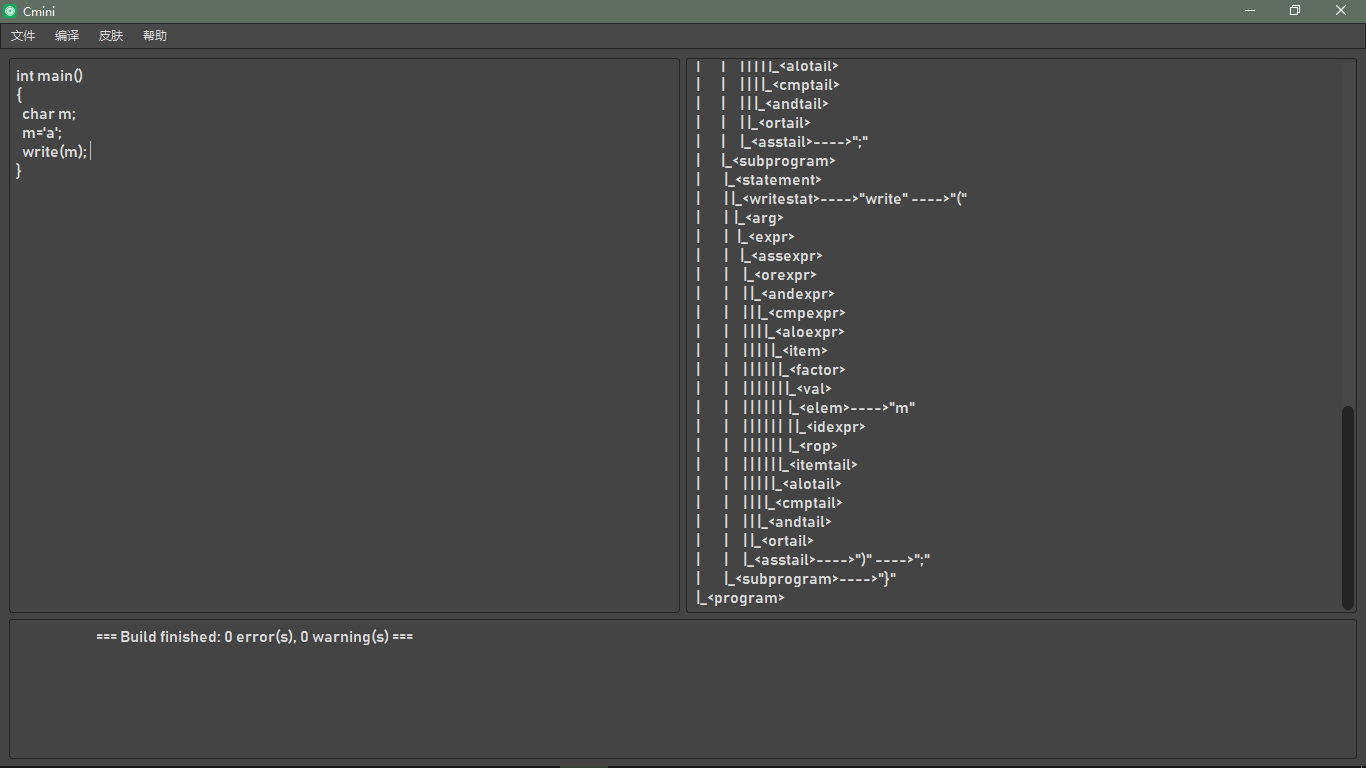
write(m);

}

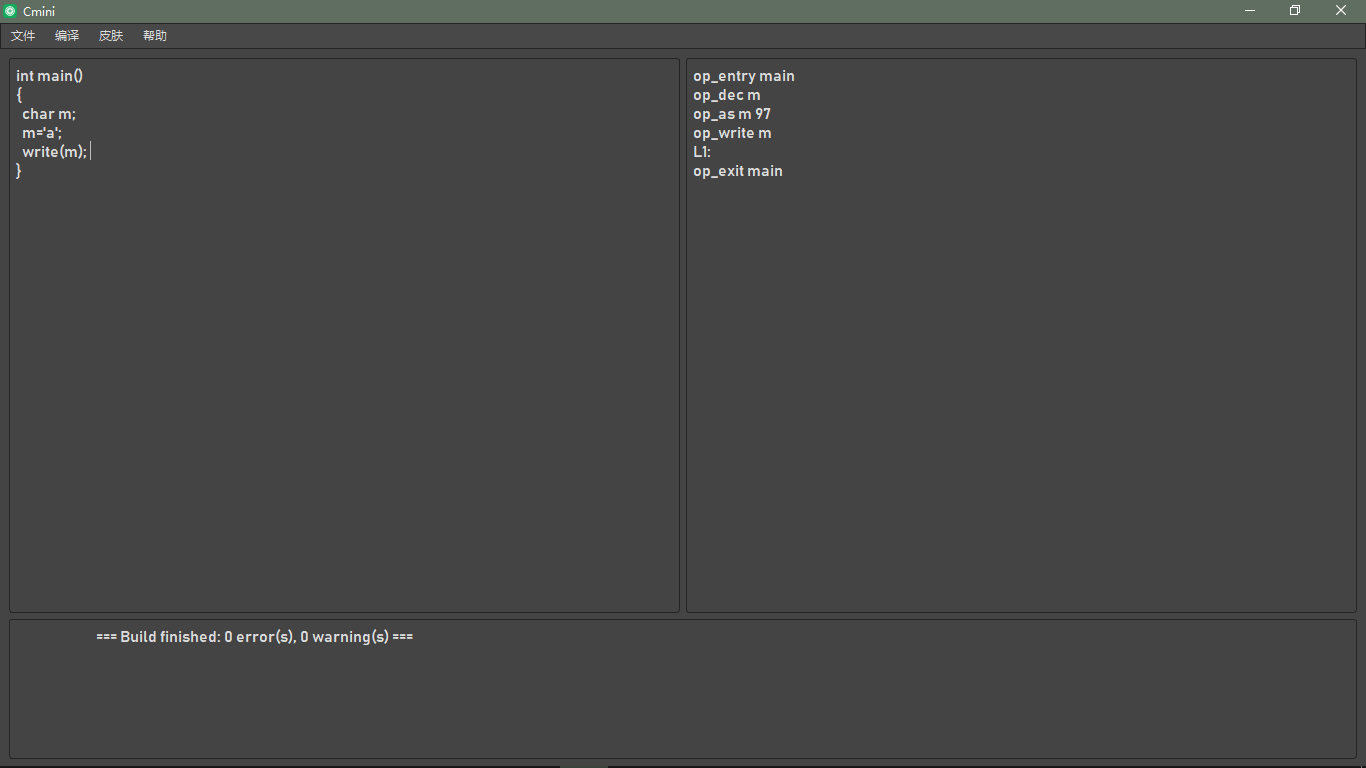
**词法分析：**



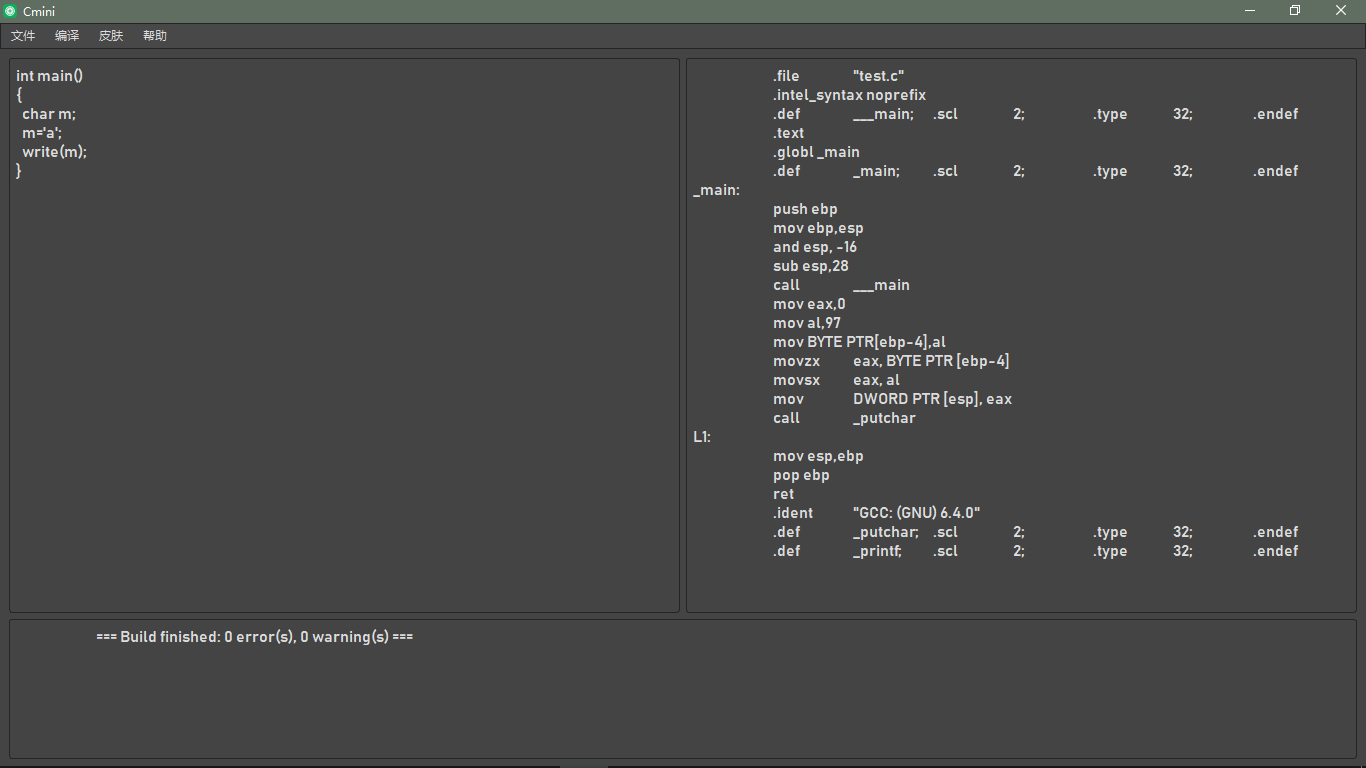
**语法分析：**



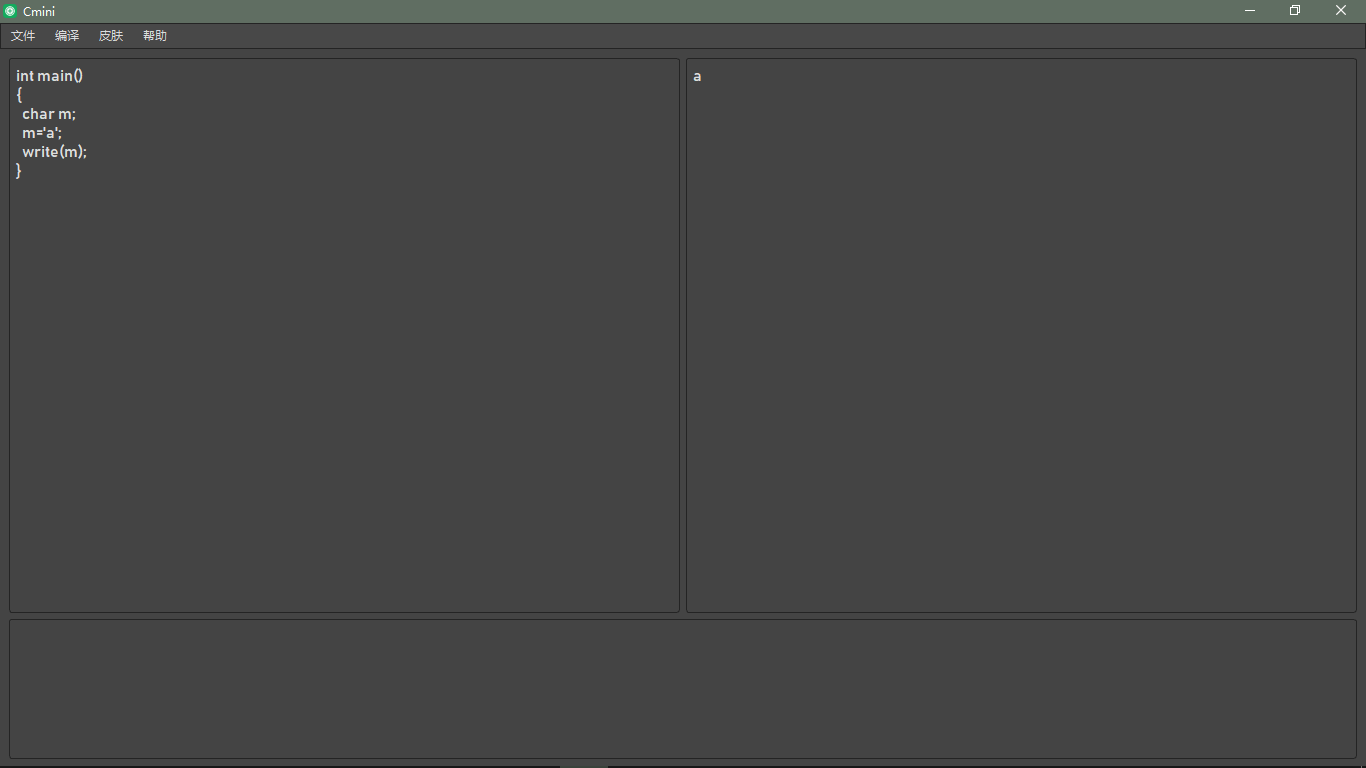
**语义分析：**



**汇编代码：**



**运行：**



### 6.6 程序测试6（测试数组功能）

int main()

{

int a[10];

for(int i=0; i<10; i++)

{

a[i]=i;

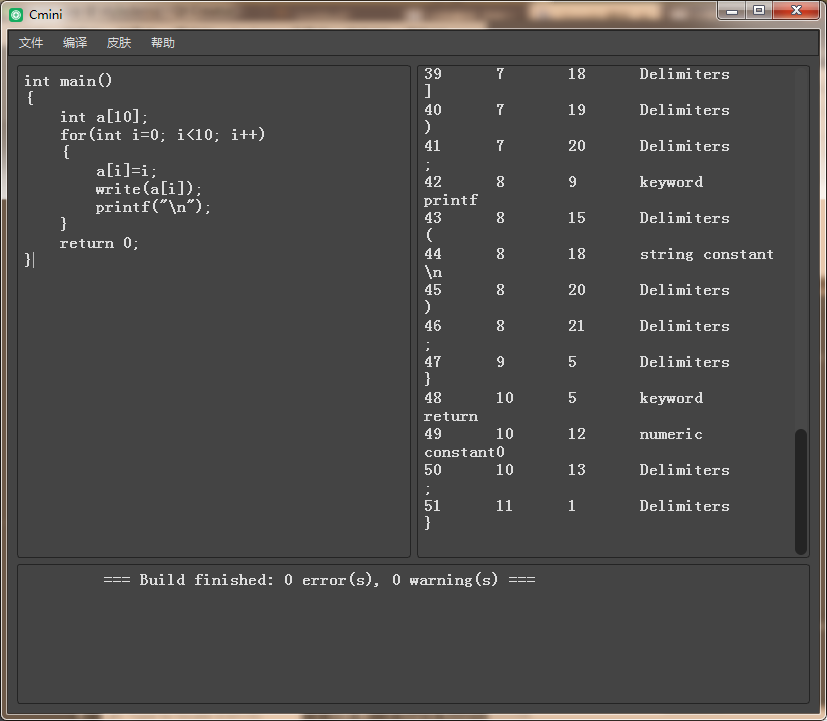
Write(a[i]);

}

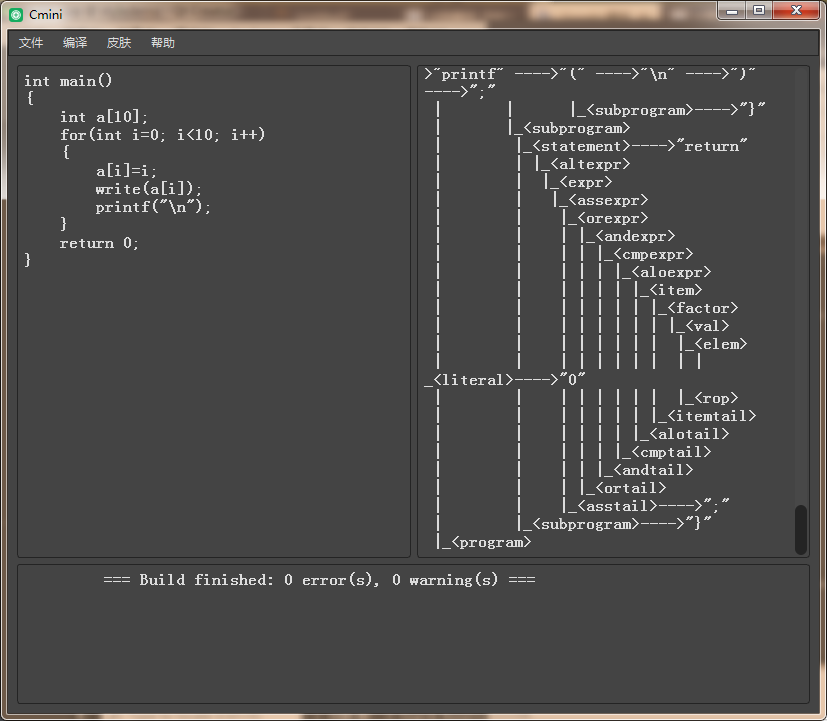
return 0;

}

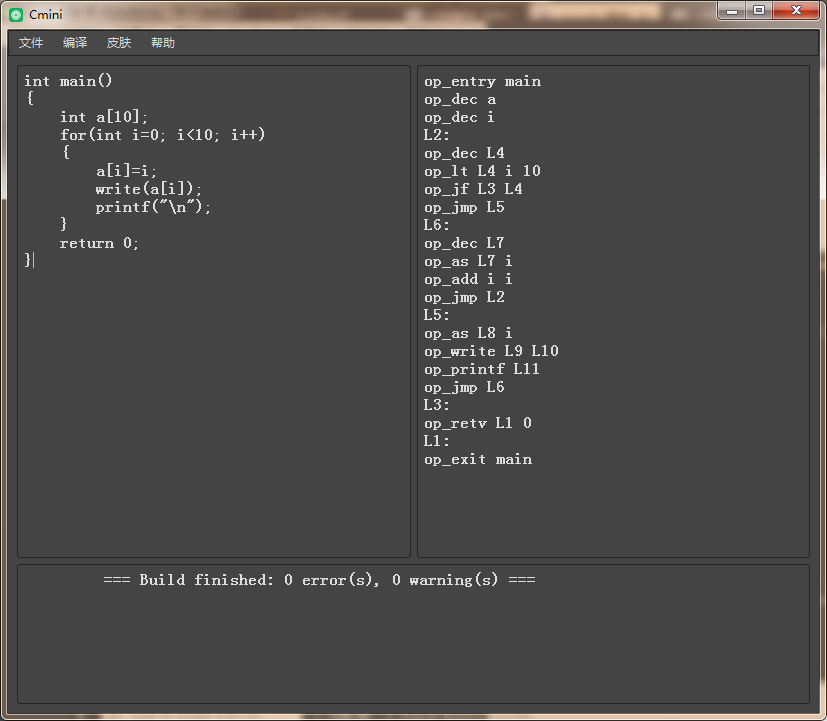
**词法分析**：



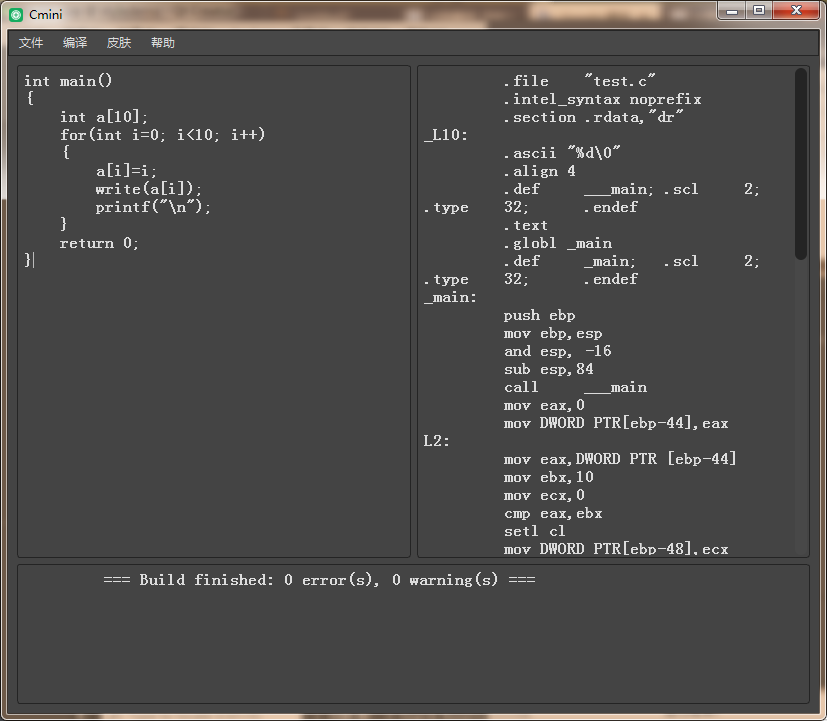
**语法分析：**



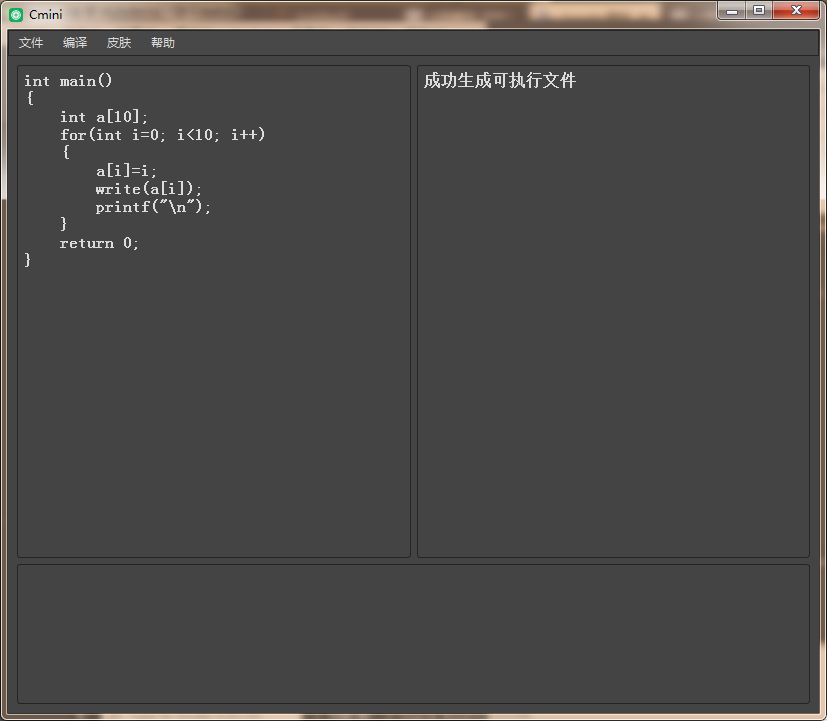
**语义分析：**



**汇编代码：**



**生成可执行文件：**



**运行结果：**



### 6.7 程序测试7（测试break&continue）

//打开break&continue.txt

int main()

{

int i;

for(i=0;i<10;i=i+1)

{

if(i==6)

{

continue;

}

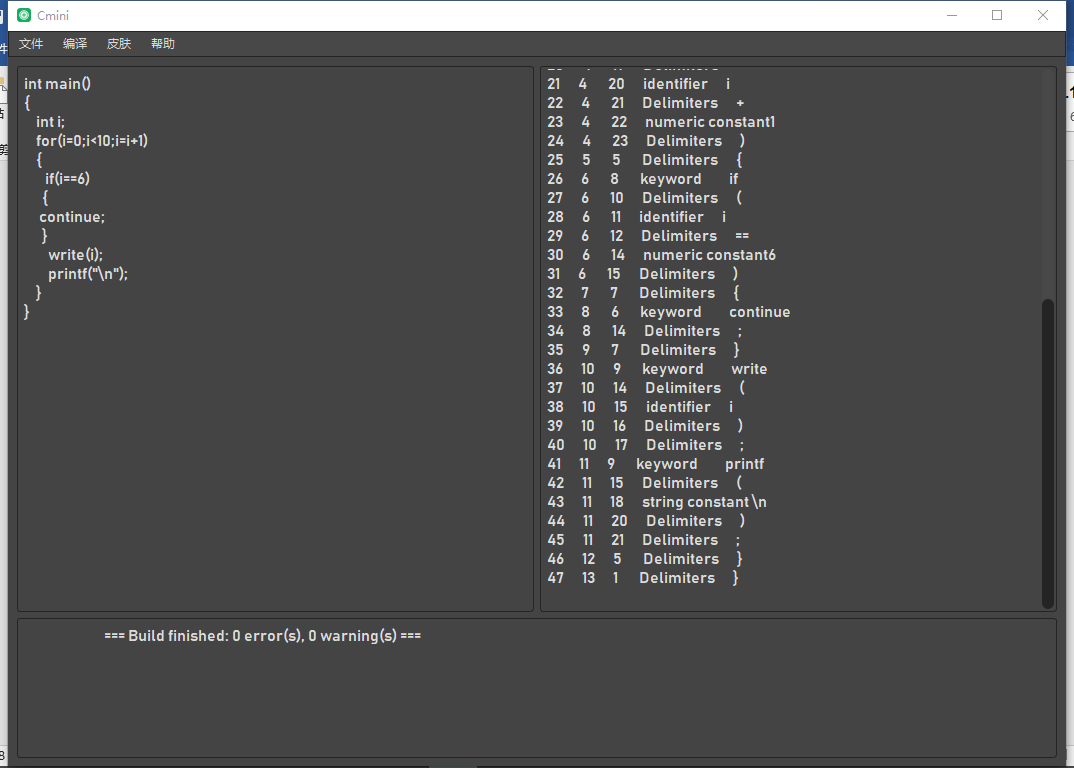
write(i);

printf("\n");

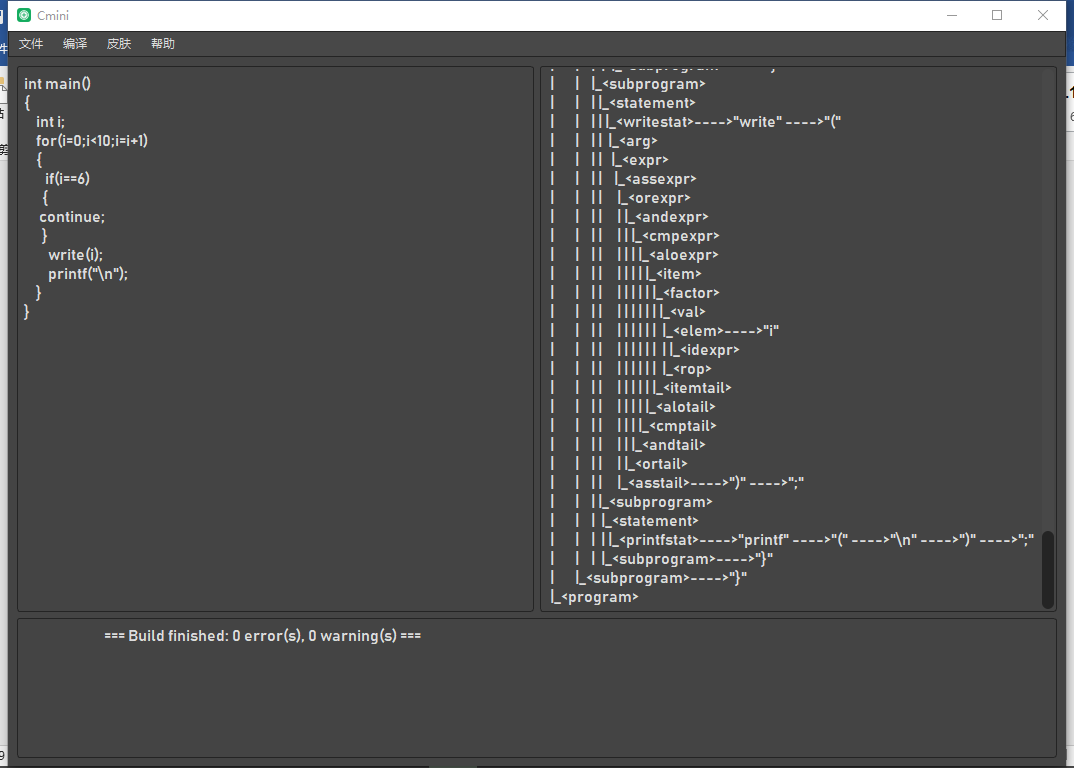
}

}

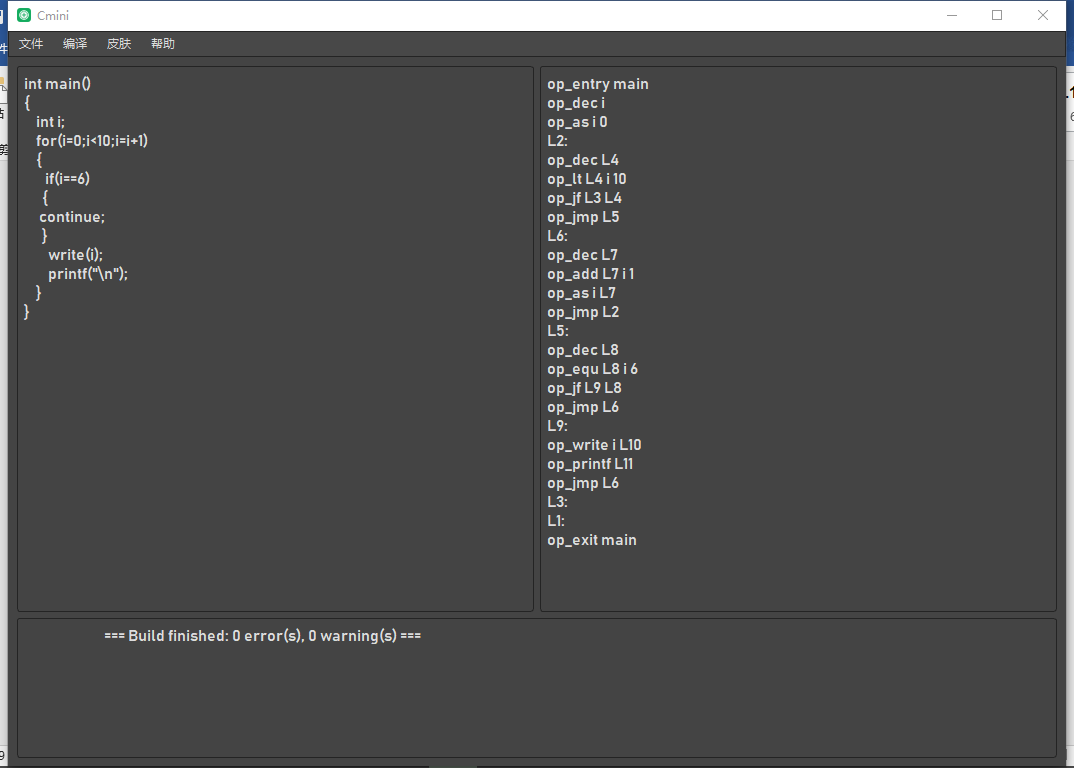
**词法分析：**



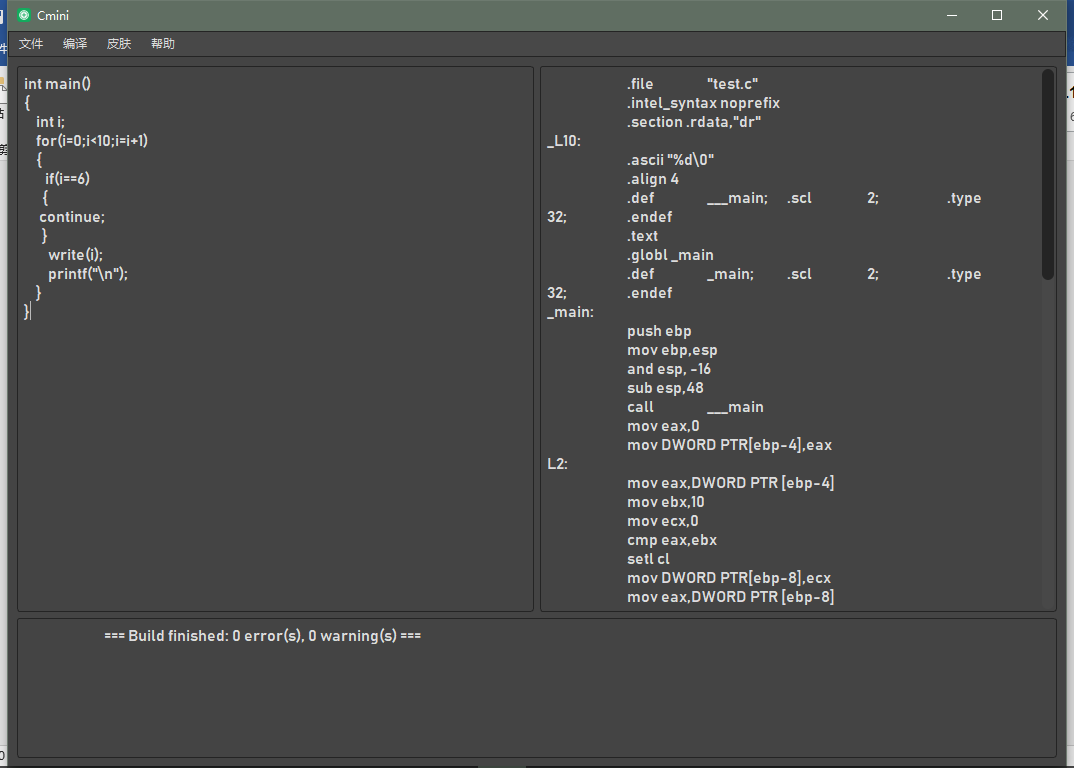
**语法分析：**



**语义分析：**



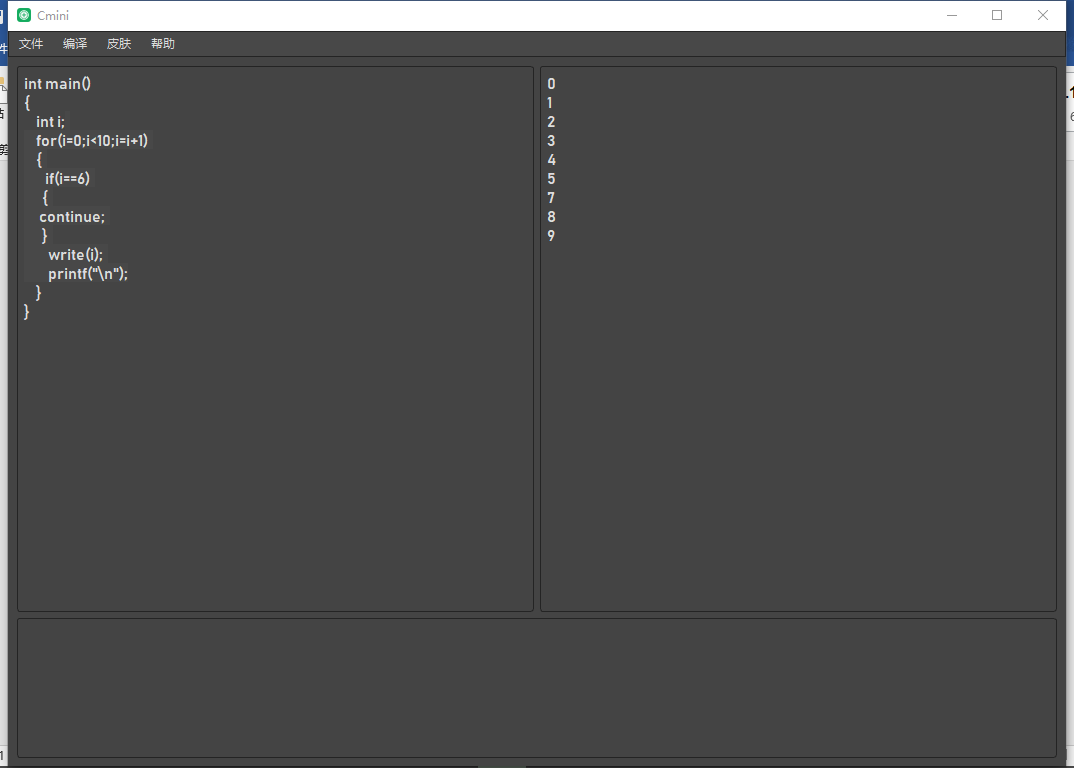
**汇编代码：**



**生成可执行文件：**



**运行：**



### 6.8 程序测试8（测试Switch-case）

//打开Switch-case.txt

int main()

{

int m=5;

switch(m)

{

case 1:printf("I am 1.");

case 2:printf("I am 2.");

case 3:printf("I am 3.");

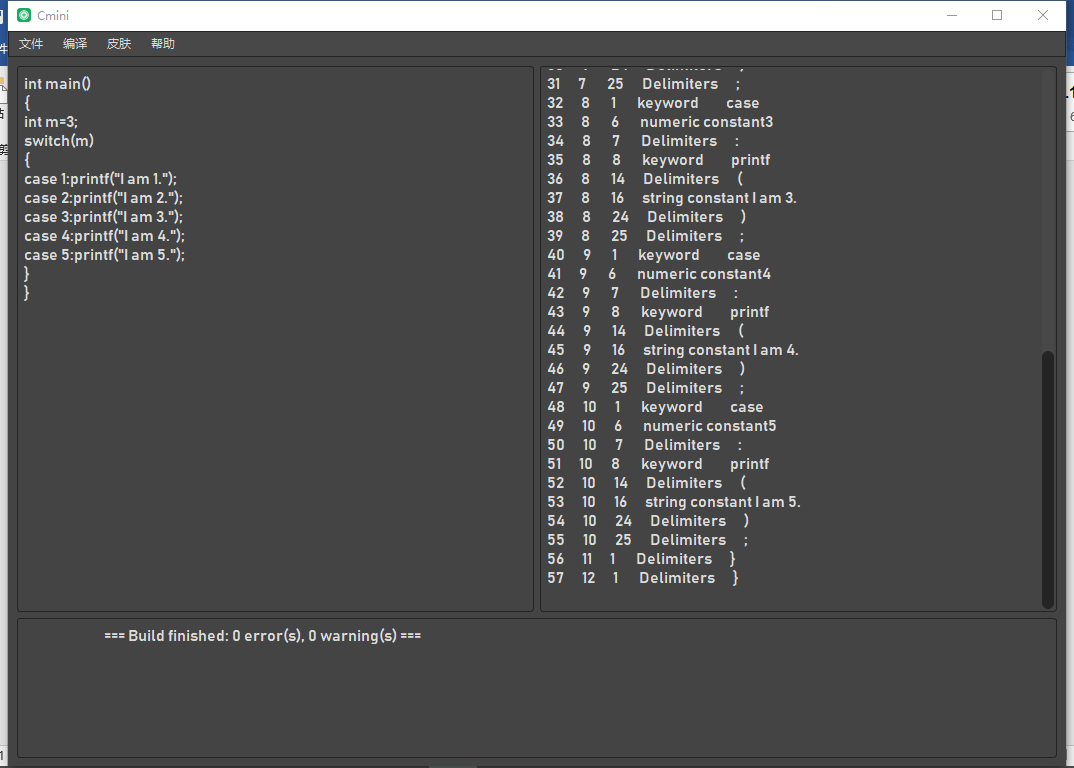
case 4:printf("I am 4.");

case 5:printf("I am 5.");

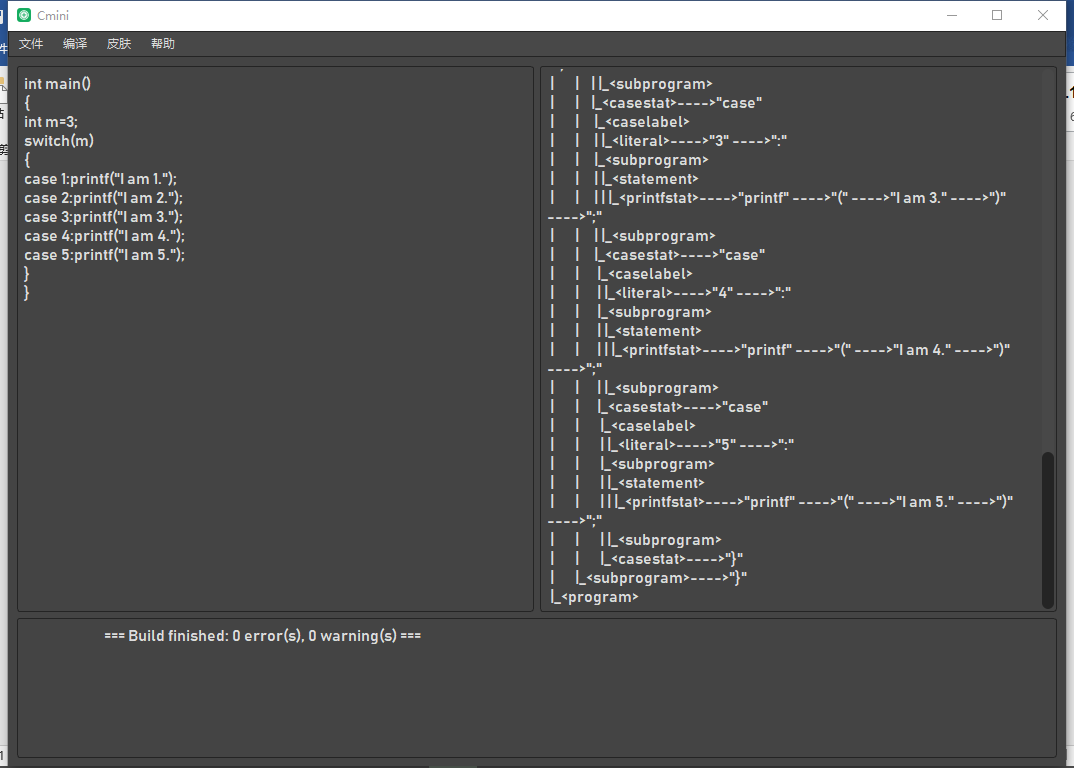
}

}

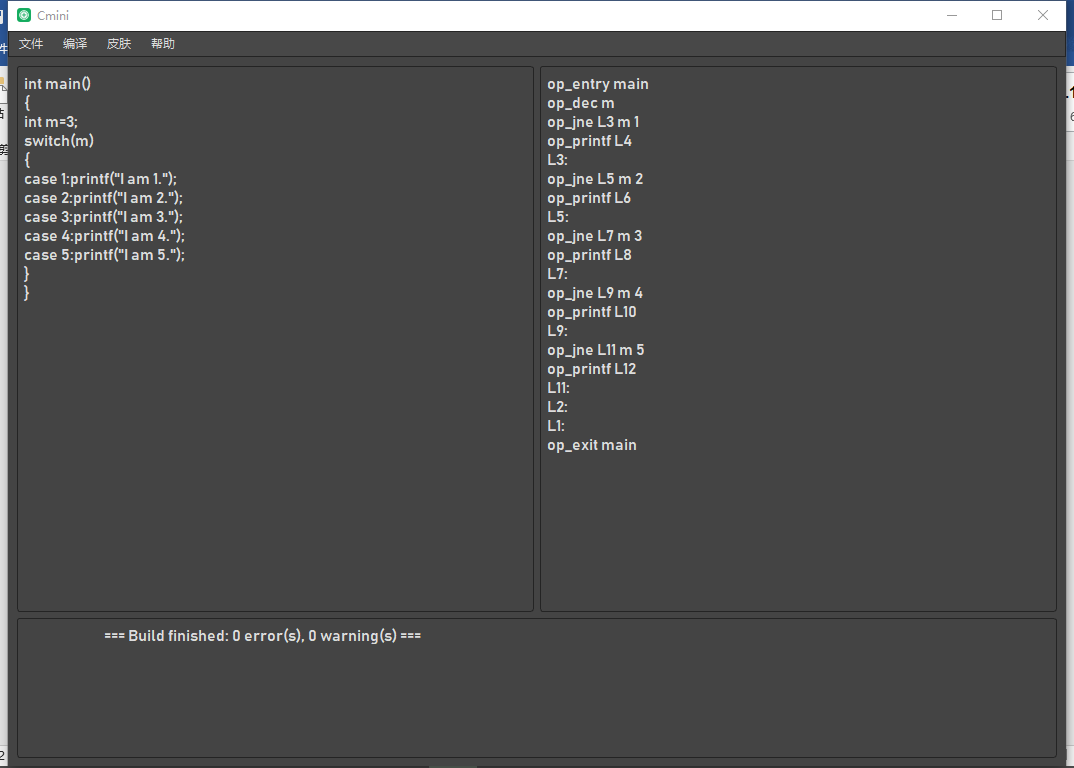
**词法分析：**



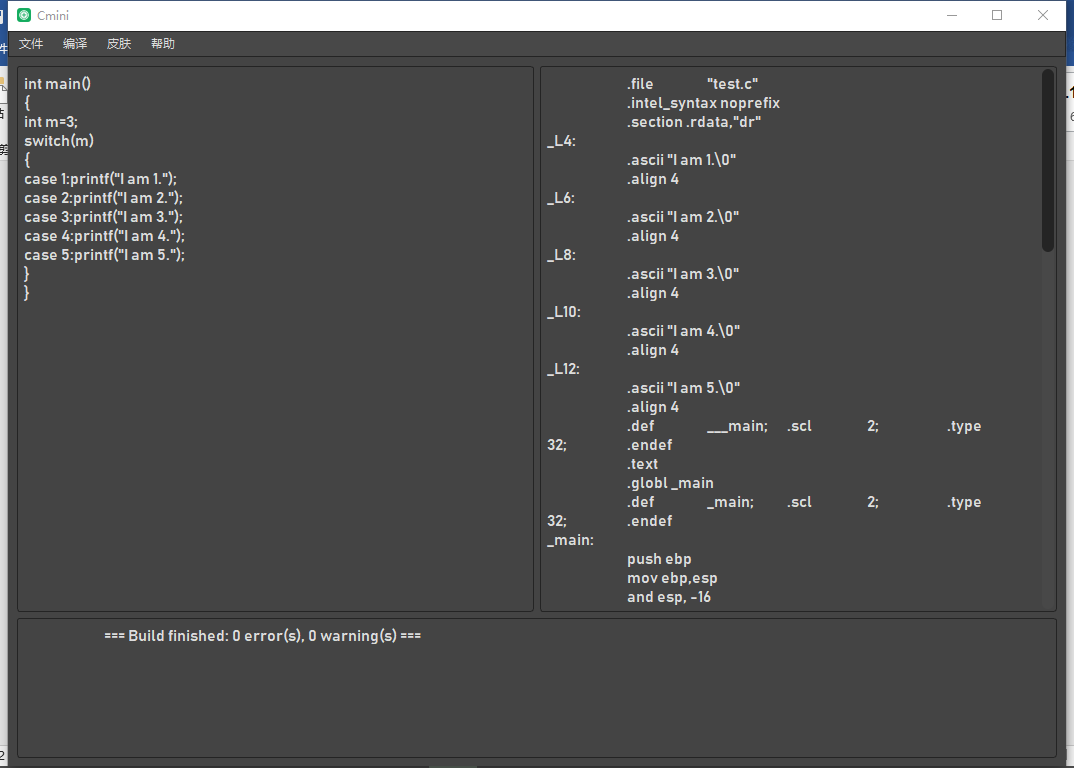
**语法分析：**



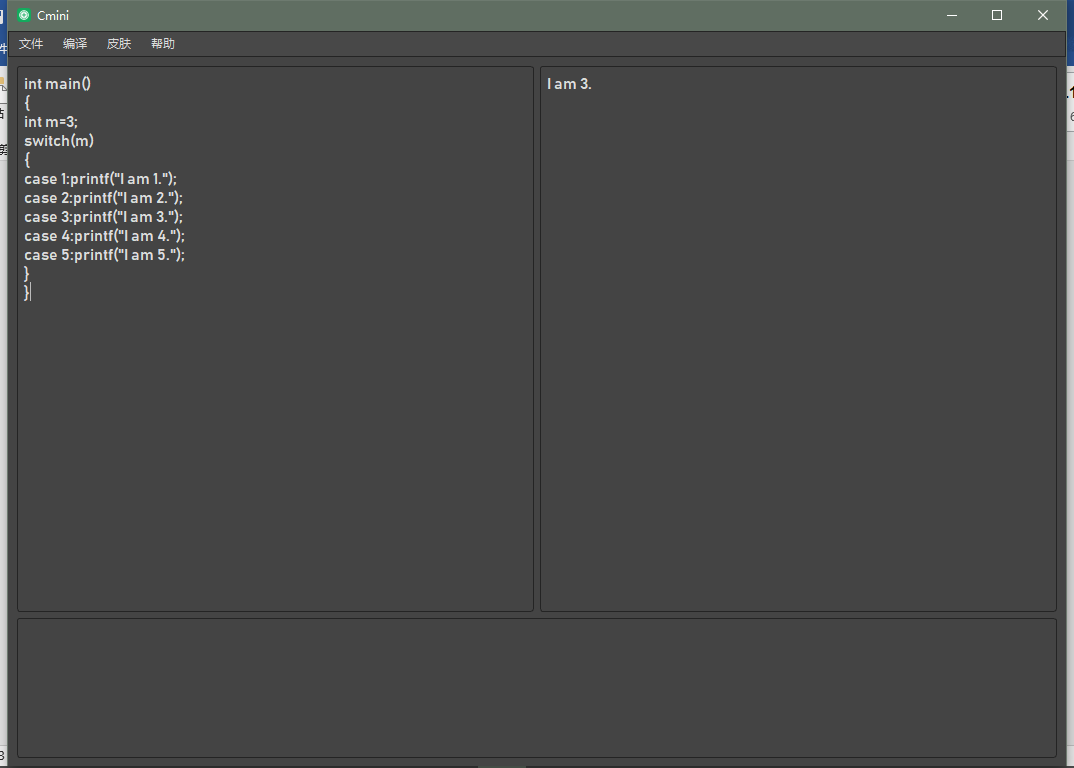
**语义分析：**



**汇编代码生成：**



**运行：**



### 6.9 程序测试9（测试调用DOS窗口进行输入的功能）

//打开read.txt

int main()

{

int i=0;

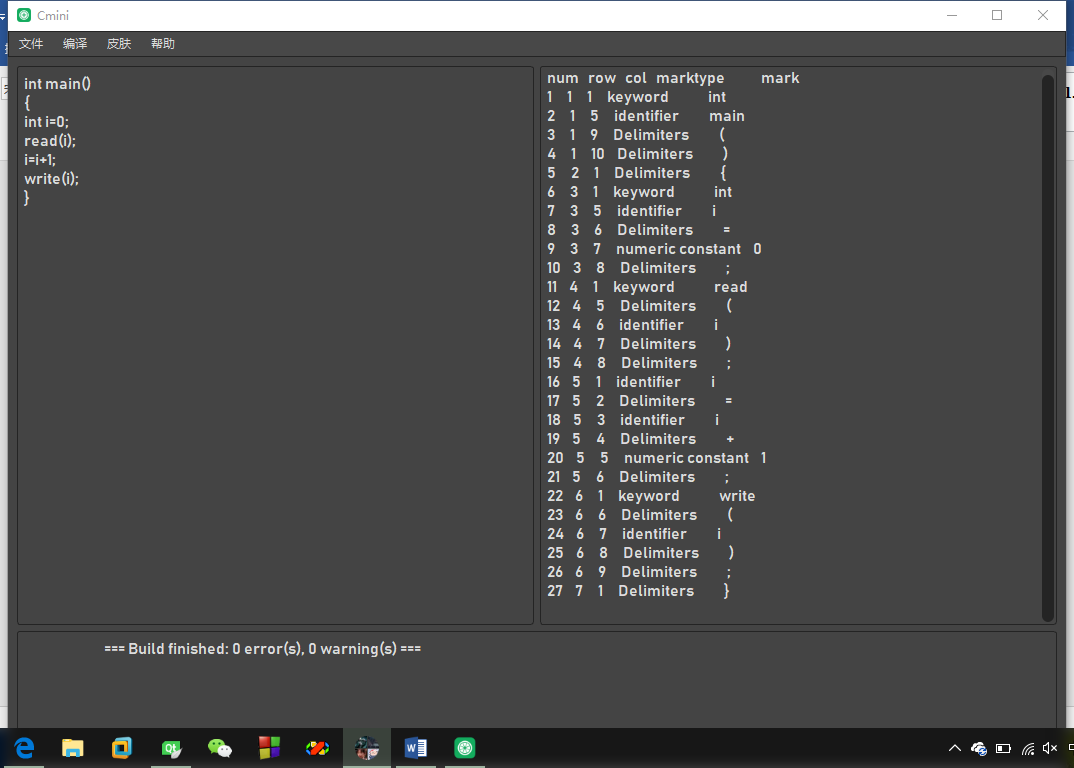
read(i);

i=i+1;

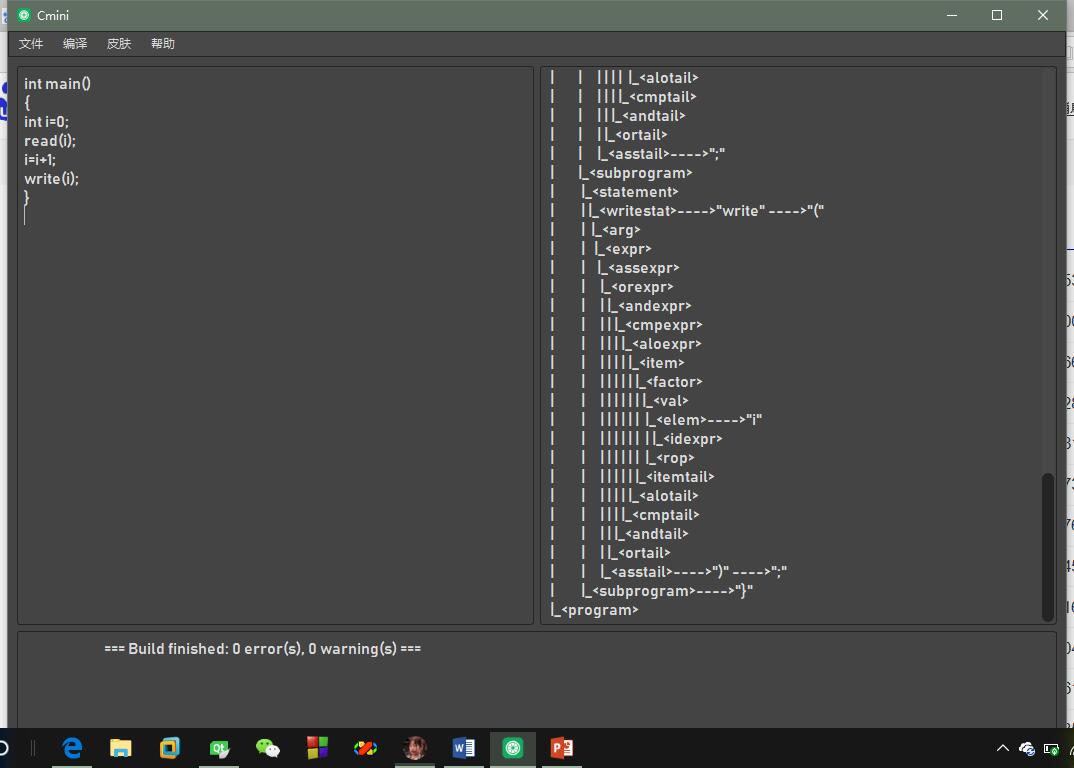
write(i);

}

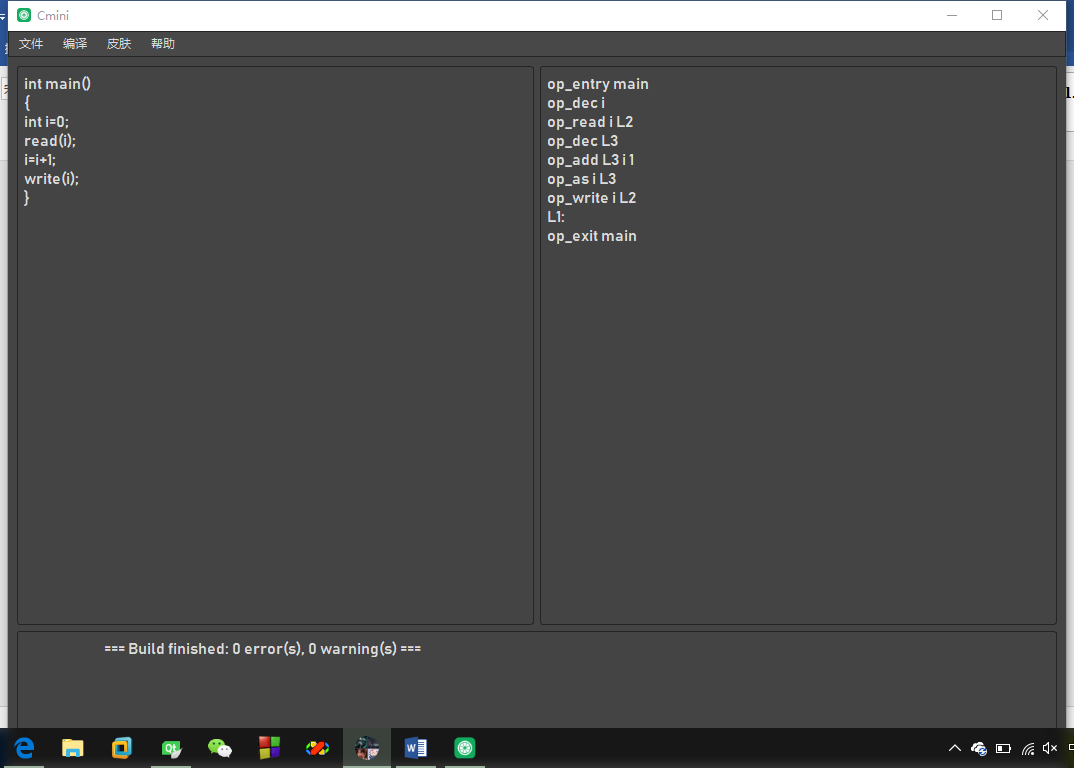
**词法分析：**



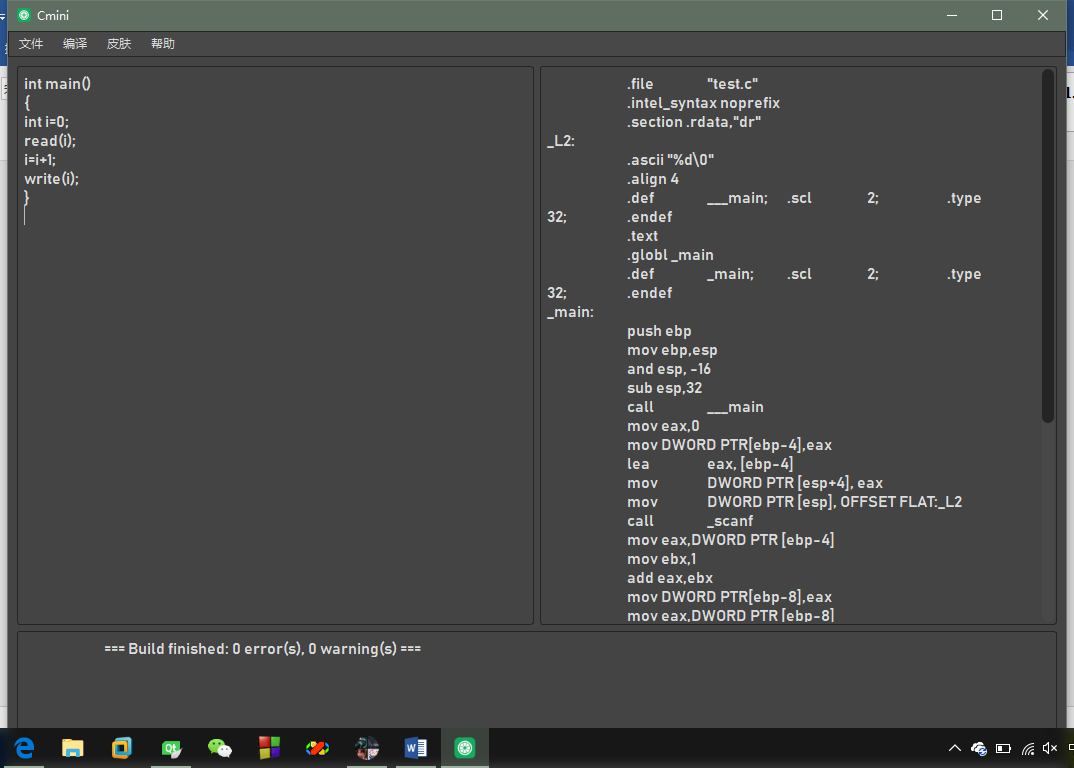
**语法分析：**



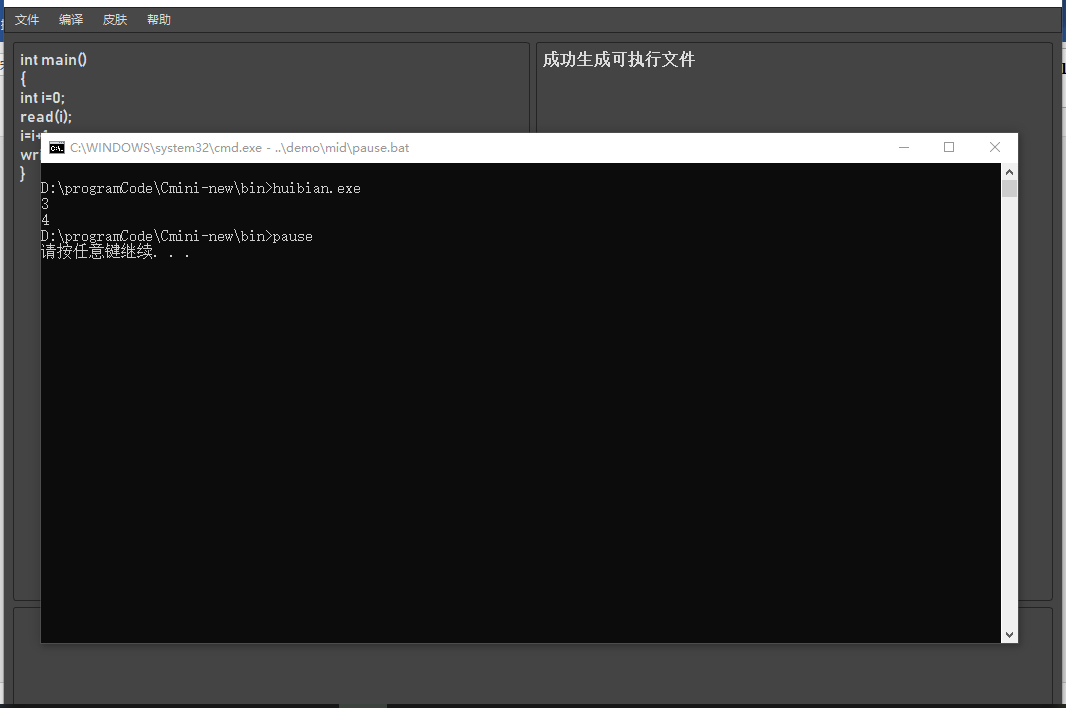
**语义分析：**



**汇编代码:**



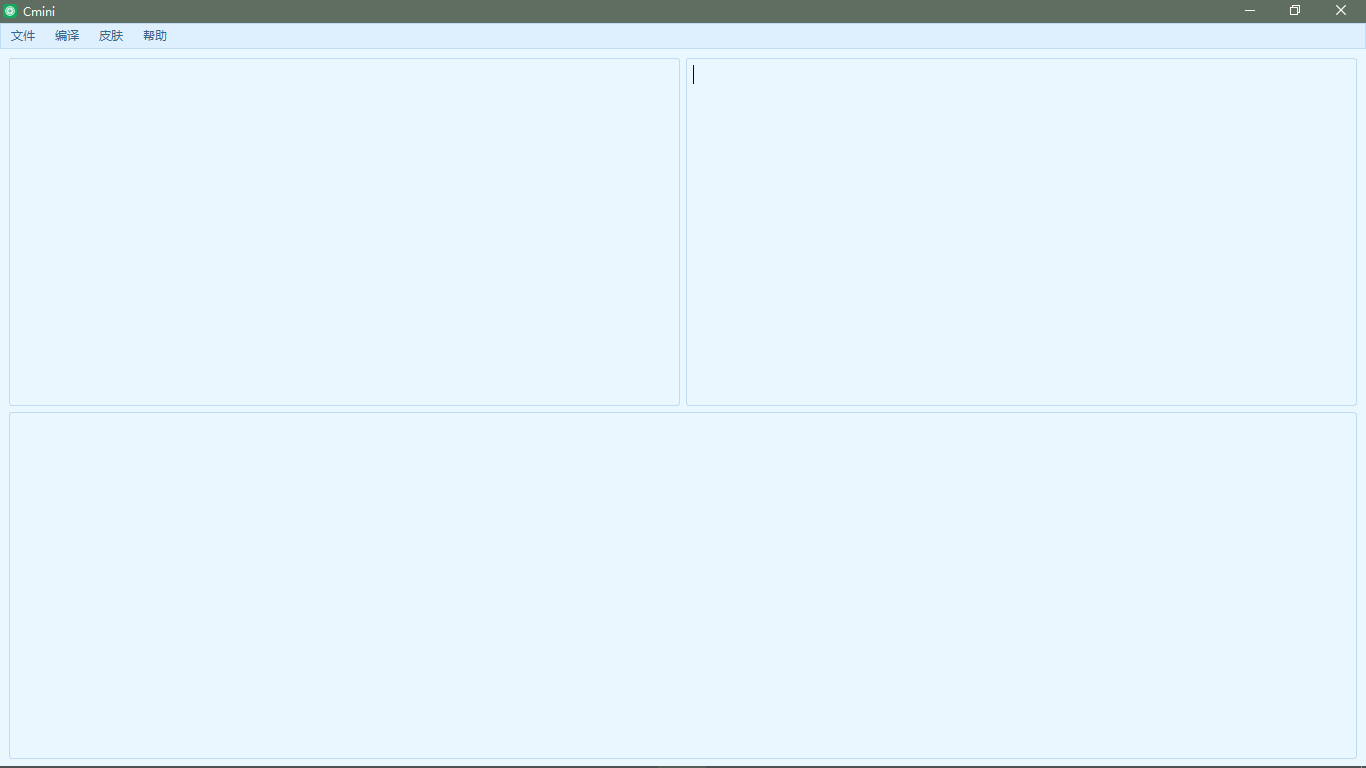
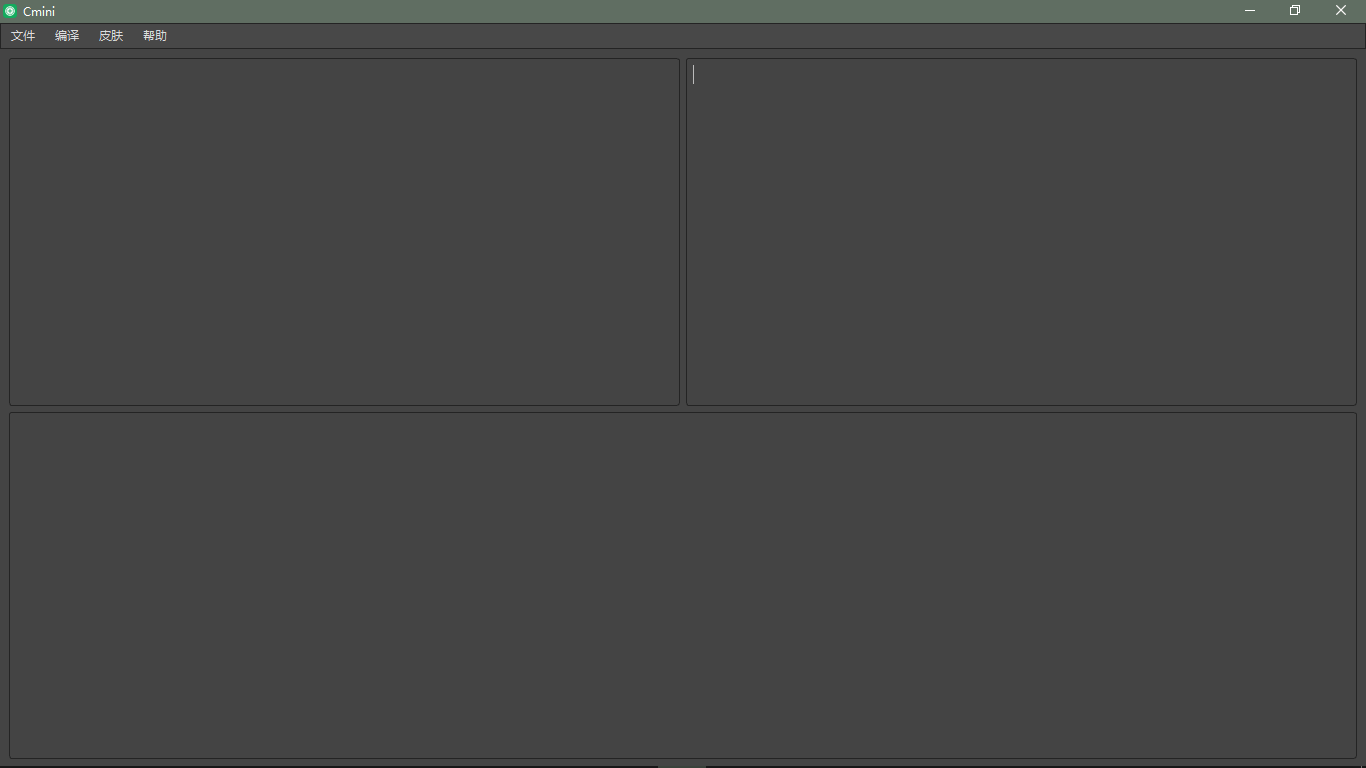
**运行：**

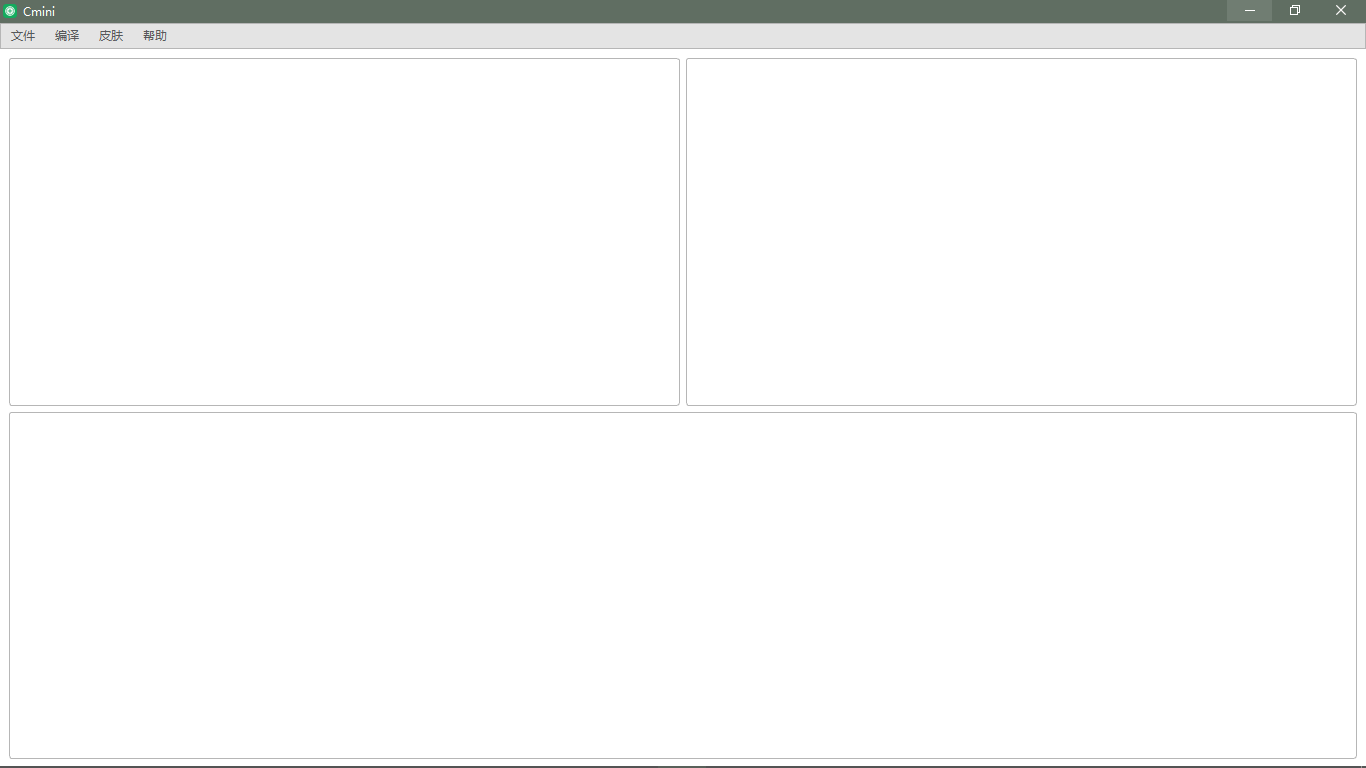
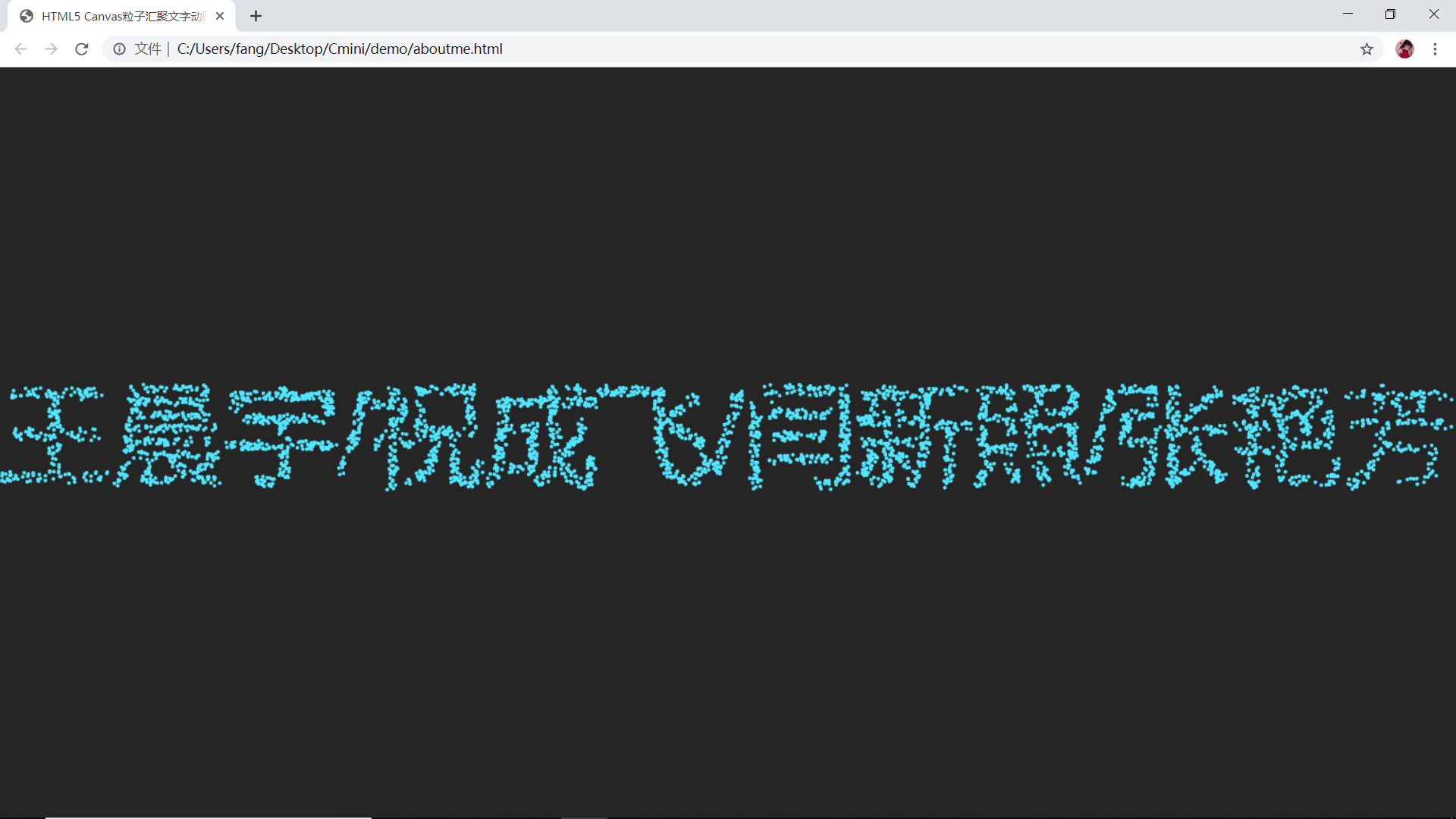


## 七、特色功能展示

### 7.1 界面优化

//各种可选颜色以及用html进行自我介绍



### 7.2 报错功能

#### 7.2.1（词法错误测试）

//打开LexicalError.txt，点击词法分析

//字符缺失右向导，词法错误

int main()

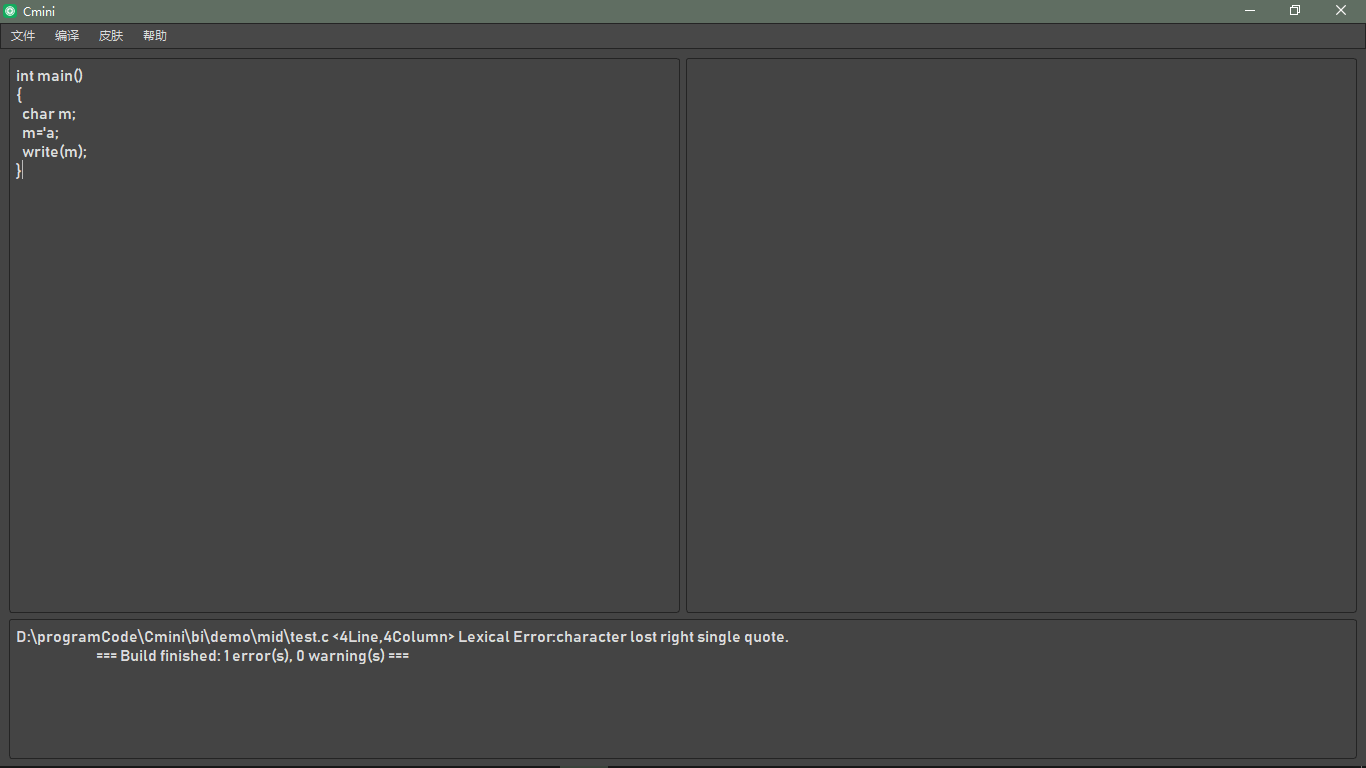
{

char m;

m='a;//此处缺少右向导

write(m);

}



**结果**：错误正常显示，测试正确

#### 7.2.2（语法错误测试）

//打开SyntacticError.txt，点击语法分析

//倒数第四行缺少逗号，即语法错误

int main()

{

int a=0,b=1;

if(a==0)

{

write(a);

printf("\n");

}

else

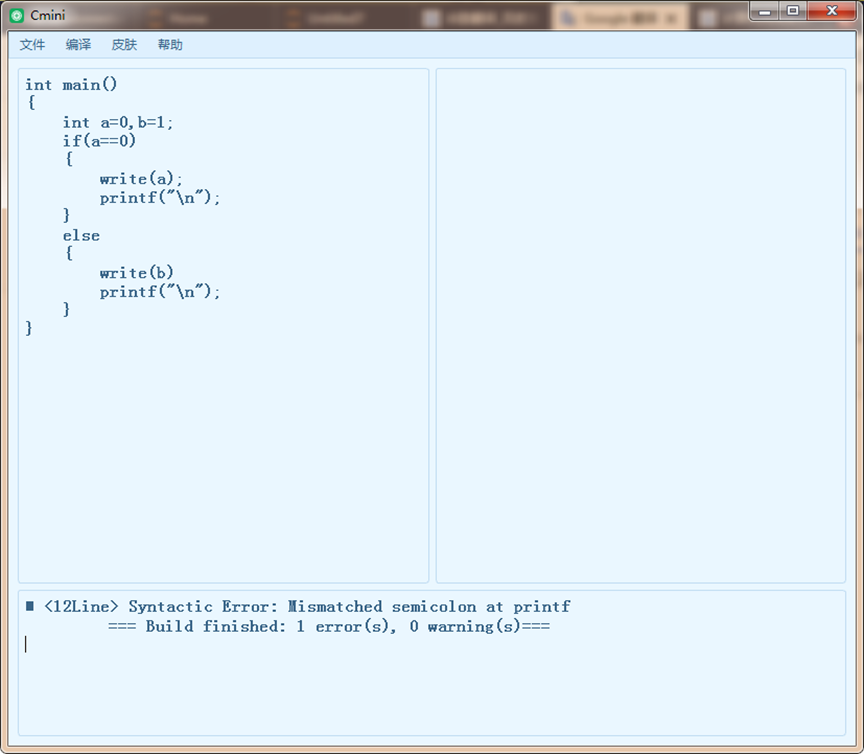
{

write(b)//此处缺少冒号

printf("\n");

}

}



**结果**：错误正常显示，测试正确

#### 7.2.3（语义错误测试）

//打开SemanticError.txt

//变量重定义，出现语义错误

int main()

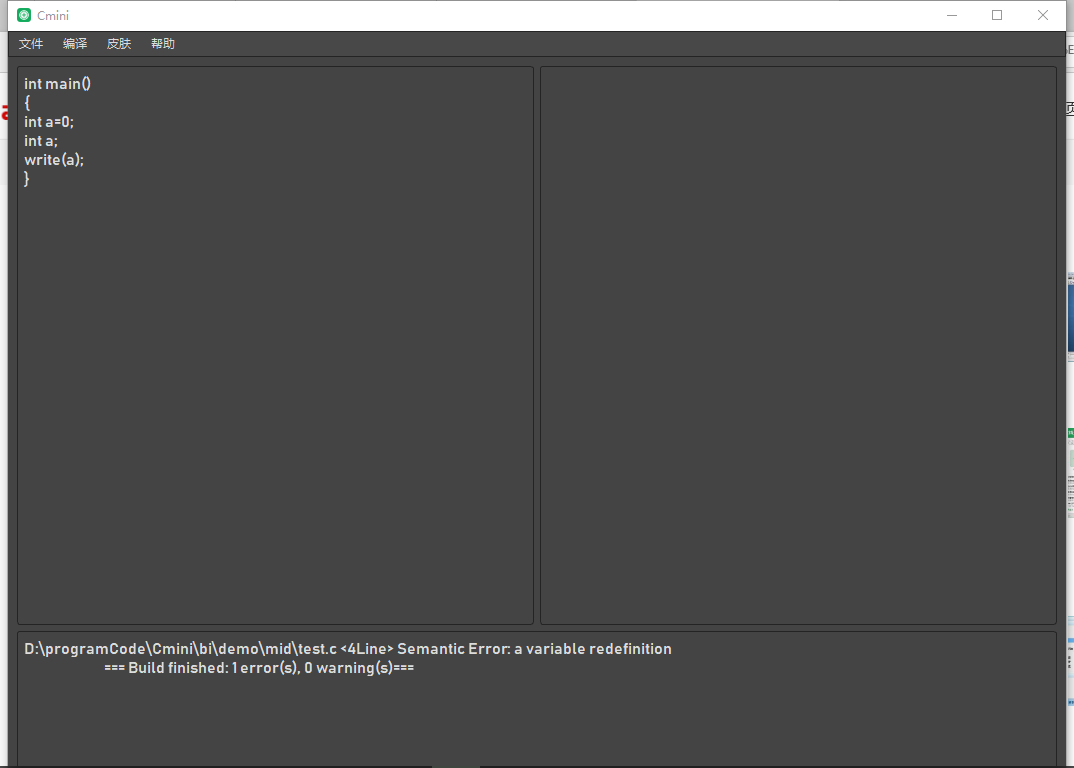
{

int a=0;

int a;//变量重定义

write(a);

}



**结果**：错误正常显示，测试正确

## 八、心得体会

**祝成飞：**这学期的编译原理课设已经接近尾声，此次课程设计，在课堂学习的基础上，通过实现一个小编译器，来加深对编译原理的理解和实践。看似一个简单的编译器，其实并不简单，正所谓麻雀虽小，五脏俱全，要实现从程序的编译、汇编、链接到生成可执行文件的整个过程，不管是哪一个一环节都很重要，从一开始的词法分析、语法分析、语义分析、中间代码到最后的目标代码生成，都需要对每个过程有着深刻的理解和掌握。并且，在实际编写过程中我们也发现，完成一个相对较复杂的工程，不仅要完全掌握课堂上学习的内容，更需要在此基础上拓展思维，学习并借鉴其他优秀的资料或程序。虽然整个实现的过程磕磕绊绊，但经过这大半个学期的时间和组内队员之间的集思广益，最终将这个编译器实现，获益匪浅！在这个过程里面，随着编译器的实现，也逐步加深了对编译原理相关知识的理解，真正的实现了将所学的知识运用到实际的项目里面，对整个编译的流程和架构的认识也逐渐清晰明了。最后，这次编译原理课设真实地锻炼了自己的项目实践和合作能力、学以致用的能力以及分析和解决问题的能力，也加深了对编译原理这门课程的理解，可以说，这是一次宝贵又难忘的经历！

**张艳芳**： 经过大半个学期的课程设计，掌握了什么是编译程序，了解了编译的过程，更加全面系统的理解了编译原理程序构造的一般原理和基本实现方法，把学过的编译原理通过设计的程序表示出来，加深了对理论知识的理解。以前对计算机操作系统认识很模糊，现在通过自己做实验，从实践上认识了操作系统的如何处理命令。这个课程设计是用C++，使用Qt开发，也因此学习到了QSS等图形化界面语言。词法分析、语法分析、语义分析、生成汇编链接执行，每一步都很重要。我们小组过程中遇到了很多问题，通过慢慢debug，不断尝试新的方法，最终还是交上了一份还算令人满意的答卷，虽然其中也有很多想要完善但是没来得及的地方，比如Qsciscinilla插件的使用，比如一步执行run。每一个在南堂南教熬的夜，也没有辜负。这个课程设计真的让人成长很大，学到很多，将知识运用到实践中会让我们学的更深入也会给人很大成就感，感谢课程设计过程中，小组成员的互帮互助，其他同学的帮助，老师的指导。

**闫新照**：实践出真知，没开始做的时候大家都曾将听说写一个编译器就会变得多么多么厉害，大家也都是踌躇满志的，好像随手一做就可做出一个编译程序，但是事实却不是那么简单的。我们的编译原理是上一个学期学的，知识都忘得差不多了，这就意味着我们要边复习边做。除此以外，大家手头的资料有图书馆借的书，别人汇编程序的参考。开始做的时候是从词法分析开始做起，感觉还行，再往下做，就仿佛发现了新大陆，一个一个的问题开始冒出来，下面基本部分设计的不完善导致后面的处理出问题，以及边做边对可以处理的数据类型和文法类型的增加，到最后基本实现一个比较完善的编译器。在分析交流中大家分析出了自己对问题的不同看法和解决方法，让大家知道了，原来同一个问题原来还有不同的解决方法和更方便的处理方案。但是共通的是当大家看到生成语法树或者最后执行处最终的结果的时候，那时候的心情是难以用语言描述的。完成了基本的功能之后，我们琢磨着再增加别人没有的功能，这些东西就要我们从零开始做，没有别人的参考没有学长学姐们的帮助，全靠我们自己一点一点摸索，有些东西做出来了，有些东西没做出来。但是最终有成果的时候还是很令人高兴的。

**王晨宇**：半学期多的编译原理课设到这里也算结束了，此课程是基于上学期所学的编译原理理论课学习的基础上开设的，回首这13周也是感触颇多。首先，我们可以通过此课程可以更好地了解上学期所重点学习的词法分析，语法分析，语义分析，比如上学期所学的自动机，只是知道如何按照规律将其确定化，最小化但不是很理解如何应用到编译器中，在查阅相关书籍以及看网上博客的过程中我在写注释有限自动机以及字符串有限自动机的学习中了解到了词法分析原来这么巧妙，对语法分析中的语法树也是有了更深的理解。另外，锻炼了自己解决问题的能力，其实想自己从头到尾写一个编译器是不太现实的，就想老师第一节课说到的，要查阅相关资料，学习人家的思路，这也是一种历练，当时有几个问题我印象很深，就是队友发现我们的语义分析只能生成一次，第二次就会报错，并退出程序，当时令全组人都费解，并有些不知所措。后来我一步步通过std：：cout<<”123”<<std::endl;去排查，发现是在调用一次语义分析的相关函数后，没有将其中的一个哈希表清零，导致第二次点击按钮触发事件的时候，会出现main函数重定义的现象，之后在相关按钮的事件函数的末尾将其清零，问题解决。再有就是在做报错的时候，我们是用stringstream变量记录错误的，每次都将上次的清零 然而上次的记录还是在界面上，后来找资料可知，stringstream不该用.clear()清零而应该用str(“”)。虽然老师的要求是做到汇编就好，但是我们觉得还是近乎将其做到完美，在生成汇编后，调用cmd里面的gcc，可以将汇编变成exe文件，再次调用就可执行exe文件，在这个过程中我了解到了qt中qprocess的用法，即学即用，自学能力才是计算机人才的核心竞争力嘛。虽然我们做的编译器不能用codeblocks vc++等去相比，但是这一路走来心里还是很惬意的，算是为今后的程序员生涯做了一次演练，很高兴能拥有一次这样的机会，令自己终身难忘。

**教 师 评 语 专 栏**

|  |  |  |  |
| --- | --- | --- | --- |
| **任**  **课**  **教**  **师**  **评**  **语** |  | | |
| **成**  **绩** |  | **任课教师：**  **年 月 日** |