哈尔滨工业大学

**<<编译原理>>**

**实验报告**

**(2016年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **吕西亚** |
| **学号：** | **1130310621** |
| **学院：** | **计算机科学与技术学院** |
| **教师：** | **骆吉洲** |

PASCAL语言的编译器实现

1. **实验目的**

实现PASCAL语言的编译程序，对给定的pascal代码程序，将其翻译成汇编代码；

实现满足该编译程序步骤中的法分析器、语法分析器、语义分析与中间代码生成器、代码优化器、目标代码器、出错处理器和符号管理器。

1. **实验环境**

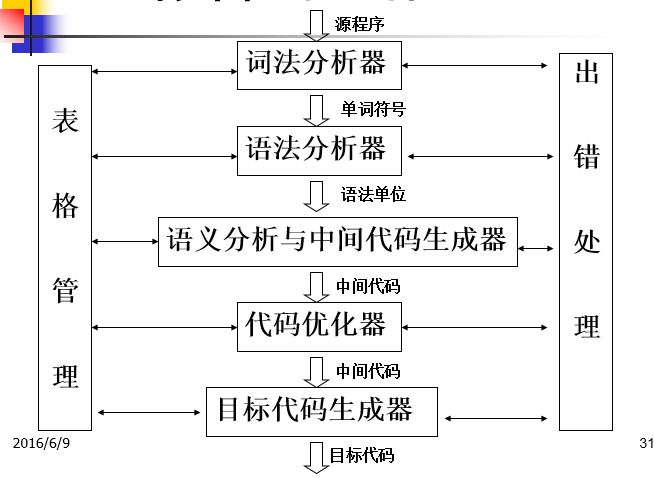
Windows7

Eclipse

Java语言

1. **实验步骤**

编译程序是一个涉及分析与综合的复杂系统。它通常由词法分析器、语法分析器、语义分析与中间代码生成器、代码优化器、目标代码器、出错处理器和符号管理器组成。其总体结构如图所示：



1. **词法分析器**
2. **基本介绍**

词法分析器，即扫描器。他的输入是组成源程序的字符串。即，词法分析器将源程序看成是一个平滑的字符流。按照组成记号的规则，通过分析，剔除多余的空白符、注解等，切分出一个个的记号，并请求符号表的管理器将这些记号中的标识符登记到符号表中。在分析过程中，如果发现错误，则进行相应的处理。。这个阶段发生的错误成为词法错误。

简单来书，词法分析器的功能是：从左到右扫描组成源程序的字符串，并将其转化成单词串，并检查组词方面的错误并进行处理。

**2.实现过程**

在本次实现过程中，通过函数**chache\_read，getch，chache\_read\_init，token\_scan**等函数实现词法分析。

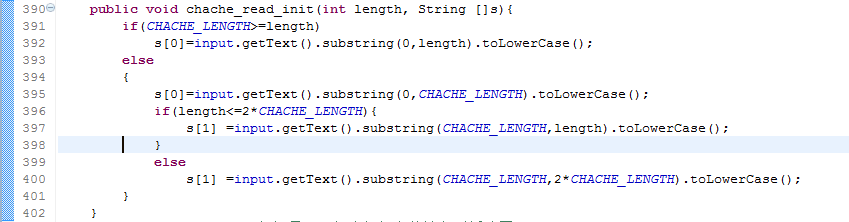
首先，通过界面输入窗口获取到目标代码的长度，然后定义一个长度为2的字符串数组即s[2]，用以作为双缓冲区，缓冲区中能存储的元素个数表示为常量CHACHE\_LENGTH；





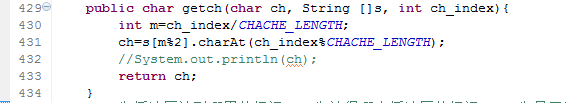
然后通过函数chache\_read\_init初始化双缓冲区，实现方式是根据已经得到的目标代码的长度length，与CHACHE\_LENGTH比较。

1. 如果length<=CACHE\_LENGTH，只用一个缓冲区即可。
2. 如果CHACHE\_LENGTH<length<=2\*CACHE\_LENGTH，两个缓冲区即可
3. 如果2\*CHACHE\_LENGTH<length，则填充满双缓冲区。

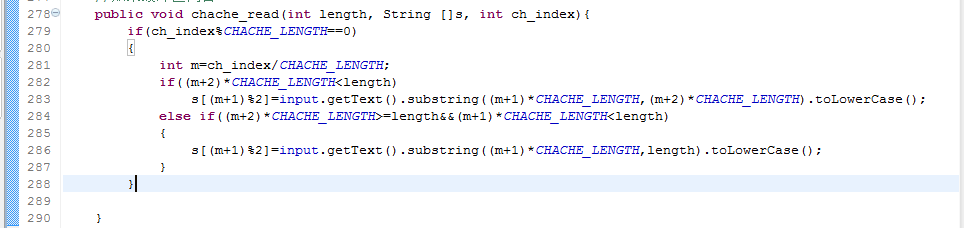


初始化完成后，从缓冲区中依次读出字符，这时通过调用getch函数实现。

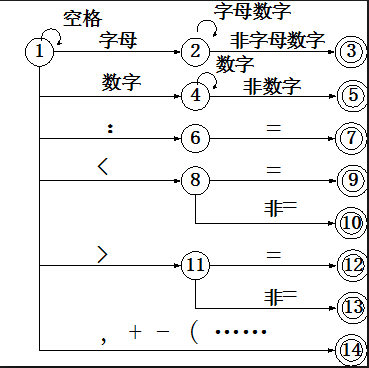
参数依次是读出的字符ch，双缓冲区，要读的字符在总的字符串中的位置，其实也是已经读了的字符的个数。



通过一个循环来循环读写字符，如果读写的个数大于或者等于程序字符数时，结束循环，表示字符读写完毕。并且每读写一个字符都要进行判断是否要更新缓冲区。更新缓冲区的操作如下：



读取字符的识别单词过程可以用状态图表示为如图所示：



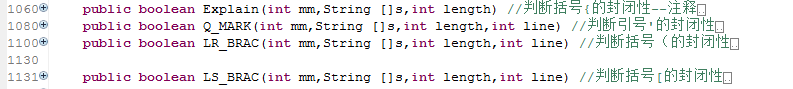
此时注意有一个回退的过程。

**3. 错误处理**

几种基本的错误处理过程：

A．非法字符，比如2sad。在该程序识别时，认为2是一个整数，sad为一个单词。

B．注解或者括号及字符常数不封闭。在该程序中，通过若干函数依次判断是否封闭，即每读到一个’(’，向后面查找对应的’)’，如果找到了则是封闭的。当有多个’（）’嵌套时，定义一个变量a=0,每读一个’(’,a=a+1,当每读‘)’,a=a-1。因此，每出现一个’(’，便定义一个a，如果能找到对应的a=0的那个位置，则认为封闭，否则不封闭。



1. **输出结果**

输出结果表示为二元组数组（种别，属性值）；

种别有关键字，标识符，常数，运算符，分解符等多种表示；

关键字：（关键字的大写形式，）

标识符：（ID，标识符名）

常数：（INT，值）或者（REAL，值）

运算符：（运算符对应符号，）

在该程序中，通过建立一个动态数组存储词法分析器结果。



1. **语法分析器**

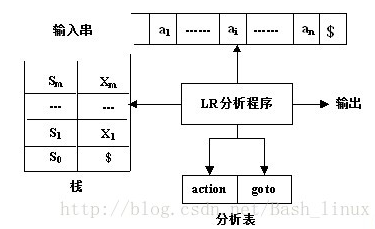
**1.工作原理**

语法分析是编译程序的核心，他的输入是词法分析程序的输出，即单词序列。他的输出是这些单词组成的程序的结构，也就是不同层次的语法成分。

在此次实验过程中采用自底向上的LR（1）分析算法实现语法分析器。如图为LR语法分析器的总体结构：

以表格为主（table-based）由下而上的分析器可用图一描述其结构，它包含：

* 一个输入[缓冲器](https://zh.wikipedia.org/wiki/%E7%B7%A9%E8%A1%9D%E5%8D%80)，输入的源代码存储于此，分析将由第一个符号开始依序向后扫描。
* 一座[堆栈](https://zh.wikipedia.org/wiki/%E5%A0%86%E7%96%8A)，存储过去的状态与化简中的符号。
* 一张状态转移表（goto table），决定状态的移转规则。
* 一张动作表（action table），决定目前的状态碰到输入符号时应采取的文法规则，输入符号指的是终端符号（Terminals）与非终端符号（Non-terminals）。



LR分析过程：

1. 将结尾字符$与起始状态0依序压入空堆栈，之后的状态与符号会被压入堆栈的顶端。
2. 根据目前的状态以及输入的终端符号，到动作表中找到对应动作：
   * 移位（shift）s*n*:
     + 将目前的终端符号由输入缓冲器中移出并压入堆栈
     + 再将状态*n*压入堆栈并成为最新的状态
   * 化简（reduce）r*m*:
     + 考虑第m条文法规则，假设该文法的*右边（right-hand side）*有X个符号，则将2X个元素从堆栈中弹出
     + 此时过去的某个状态会回到堆栈顶端
     + 在*状态转移表*中查找此状态遇到文法*左边（left-hand side）*的符号时的状态转移
     + 将文法左手边的符号压入堆栈
     + 将查找到的新状态压入堆栈
   * 接受，输入字符串解析完成。
   * 无对应动作，此情形即为文法错误。
3. 重复步骤二直到输入的字符串被接受或侦测到文法错误。

**2.实现过程**

在该程序中，语法分析器重要通过以下几个主要模块组成：

**CLOSURE求闭包函数：**

J:=I;

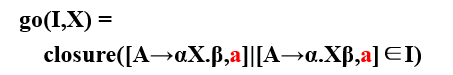
repeat

J=J∪{[B→.η,b]|[A→α.Bβ,a]∈J,

b∈FIRST(βa)}

until J 不再扩大

**GO转移函数：**



**CLO\_FULL求项目集规范族函数**

C={I0}∪{I|∃J∈C，X∈V∪T，I=go(J,X)}称为G’的LR(1)项目集规范族（算法：P185)

begin

C:= {closure({ S'→.S,#})};

repeat

for ∀I∈C，∀ X ∈ V∪T

if go(I,X)≠Φ & go(I,X)∉C then

C=C∪go(I,X)

until C不变化

end.

**LRtable求分析表：**

1．令I0= CLOSURE({S'→.S})，构造C={ I0, I1, …, In}，即G'的LR(1)项目集规范族。

2．从Ii构造状态i，0为初始状态。

for k=0 to n do

begin

⑴ if [A→α.aβ, b]∈Ik & a∈T & GO(Ik, a)=Ij then action[k,a]:=Sj;

⑵ if GO(Ik, B)=Ij & B∈V then goto[k,B]:=j;

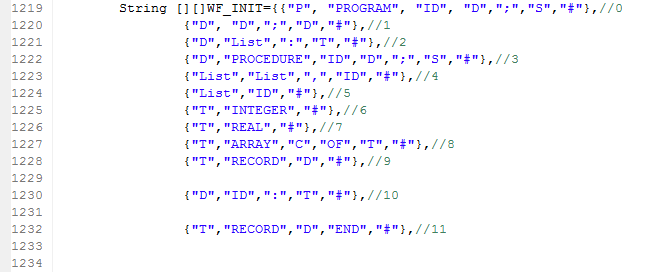
⑶ if [A→α., a]∈Ik & A→α为G'的第j个产生式then action[k,a]:=rj;

* + - 1. if [S'→S., #]∈Ik then action[k,#]:=acc;

end

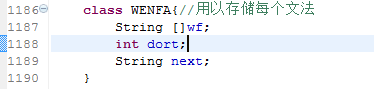
上述⑴到⑷步未填入信息的表项均置为error。

除此之外，还有一些辅助上述核心函数的其他函数，如程序中的判断两个闭包相等的函数等。

存储文法的方式是以多个字符串数组进行存储：

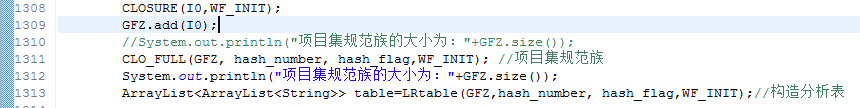
闭包的存储方式为：

ArrayList<WENFA>，其中结构体WENFA为：



Wf为文法表达式，dort表示小数点所在的位置，next为展望符。

然后依次调用上述函数，最终实现分析表：



最终的分析表存储在table中，然后调用LR（1）分析算法：

LR1\_main(table,w,hash\_flag, WF\_INIT,fu\_table,err);

其伪代码可以表示为：

输入：文法*G*的*LR*分析表和输入串*w*;

输出：如果*w*∈*L*(*G*)，则输出*w*的自底向上分析，否则报错;

步骤：

1．将#和初始状态*S*0压入栈，将*w*#放入输入缓冲区；

2．令输入指针ip指向*w*#的第一个符号；

3．令*S*是栈顶状态，*a*是ip所指向的符号;

4．repeat

5．if *action*[*S*,*a*]=*Si* then /\* *Si*表示移进*a*并转入状态*i*\*/

6． begin

7． 把符号*a*和状态*i*先后压入栈；

8． 令ip指向下一输入符号

9． End

10．elseif *action*[*S*,*a*]=*rk*then /\* *ri*表示按第*k*个产生式*A*→*β*归约 \*/

11． begin

12． 从栈顶弹出2\*|*β*|个符号；

13． 令*S*'是现在的栈顶状态；

14． 把*A*和*goto*[*S*',*A*]先后压入栈中；

15． 输出产生式 *A*→*β*

16． end

17．elseif *action*[*S*,*a*]= acc then

18． return

19．else

1. error();

**3. 输出结果**

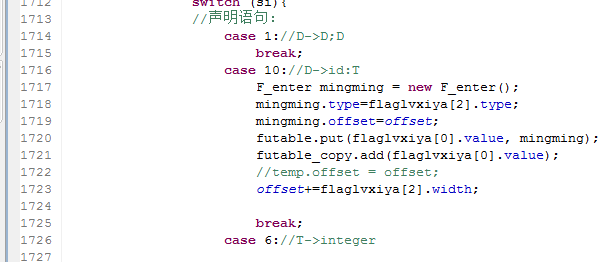
输出结果为每次规约所用到的产生式，在语义分析和中间代码生成阶段将用对应的函数进行替换输出产生式的操作。

1. **语义分析与中间代码生成**

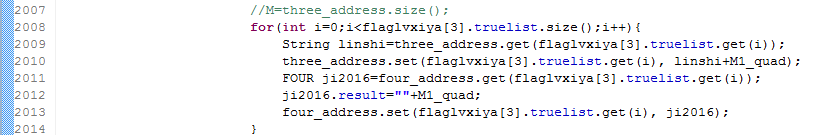
语法分析完成由语法分析器识别出来的语法成分的语义的分析。所以，语法分析器向上与语法分析器密切相关，通常在语法分析器分析出语法成分的同时进行该语法成分的语义分析。

中间代码生成以中间代码的形式实现对语义分析结果的表示。一般来说，对一种类型的计算机来说，无论机器代码还是汇编语言，都是紧密依赖于机器的。如果直接用他们来表示语义分析的结果，要考虑许多机器有关的细节，这不仅会影响翻译的效率，使翻译完全收到机器的约束，而且在一种类型的机器上实现的编译很难用到其他类型的机器上，严重影响编译程序的移植。这样不利于代码的优化，所以人们通常会先用中间代码的形式表示分析出来的语义。在此实验中使用三地址码的四元式表示中间代码。

在该代码通过判断所使用的规约文法进行对应的语义分析，如图：



需要注意的是在判断语句和循环语句中有回填操作，如下图：



当进行对应的语义操作之后将对应的规约规则表示成三地址码的四元式表示，并利用该四元式进行目标代码生成。

1. **代码优化**

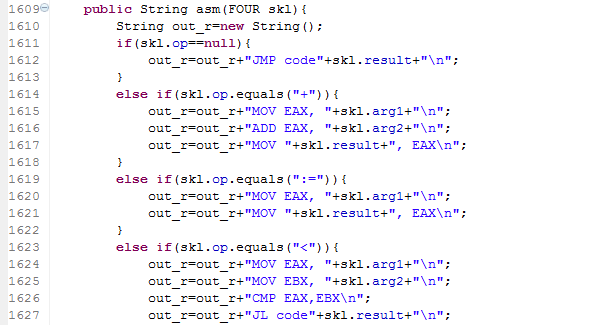
代码优化是对中间代码进行优化处理，使程序运行能够尽量节省存储空间，更有效地利用机器资源，使程序运行速度更快，效率更高。当然这种优化变换必须是等价的。

在该程序中并未进行中间代码优化。

1. **目标代码生成**

目标代码生成是编译程序的最后一个阶段，它完成从中间代码到目标机器上的机器指令代码或者目标机器上的汇编代码的转换。

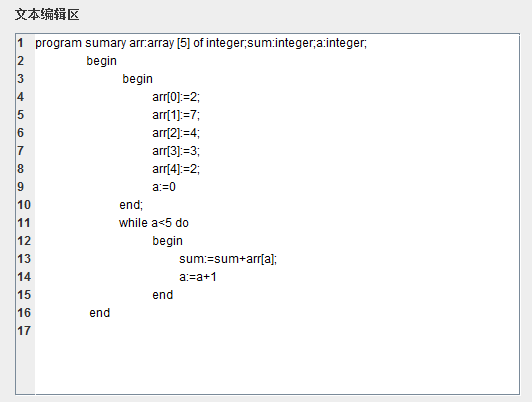
在该程序中，目标代码通过函数asm实现，如下图：



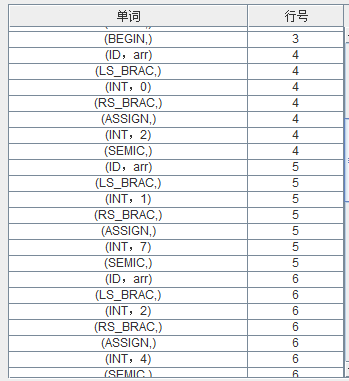
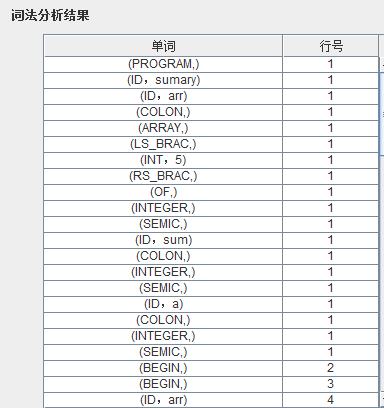
该函数输入为四元式，并对四元式中的操作符进行判断进而产生相应的目标代码。

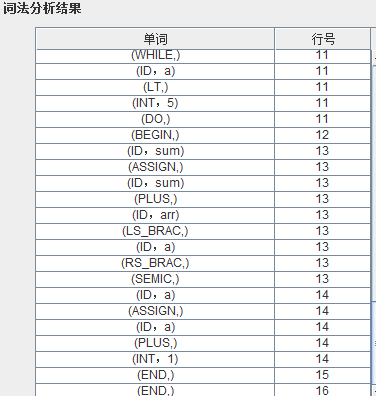
1. **实验结果**

**输入：**

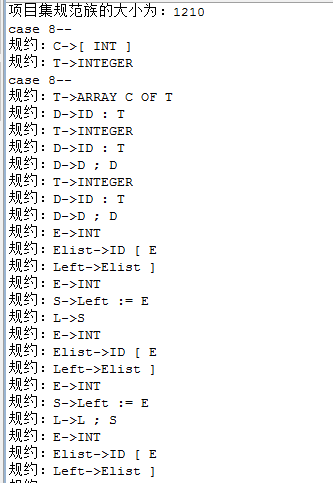
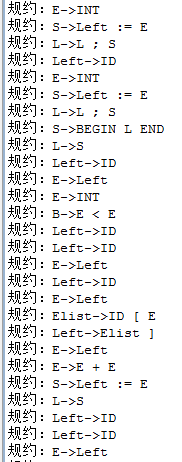
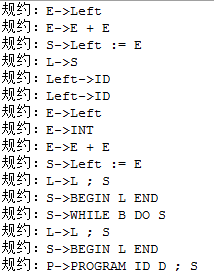


1. **词法分析器**

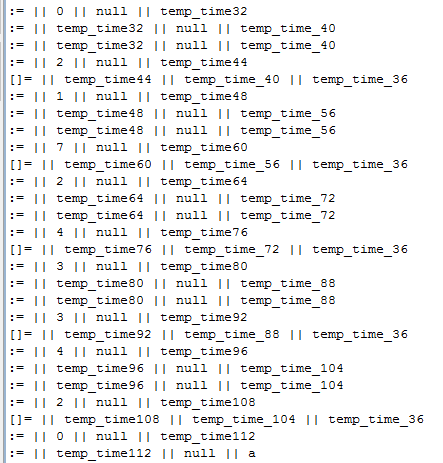




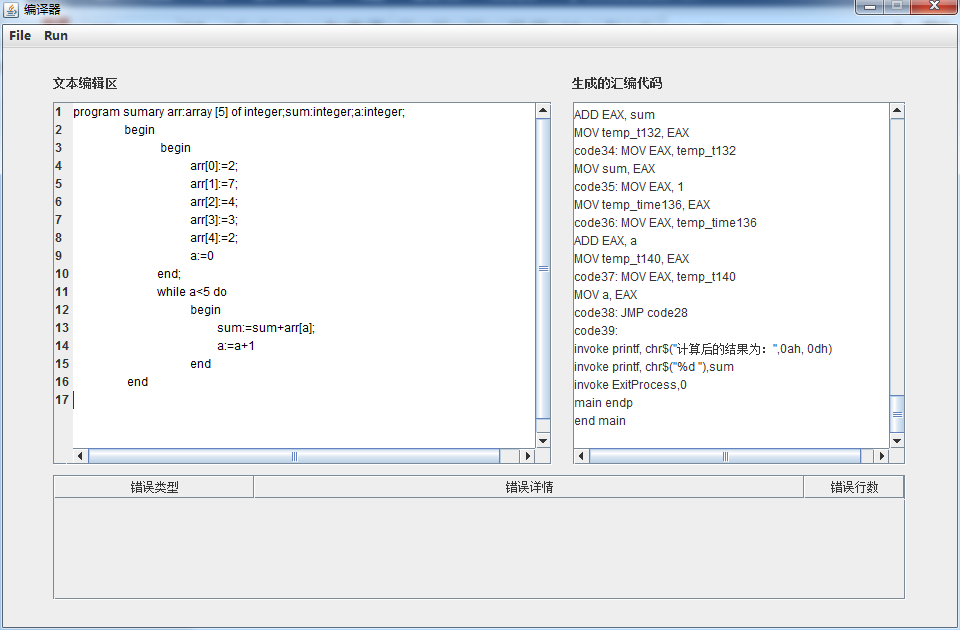
1. **语法分析器**

1. **中间代码**



1. **汇编代码**

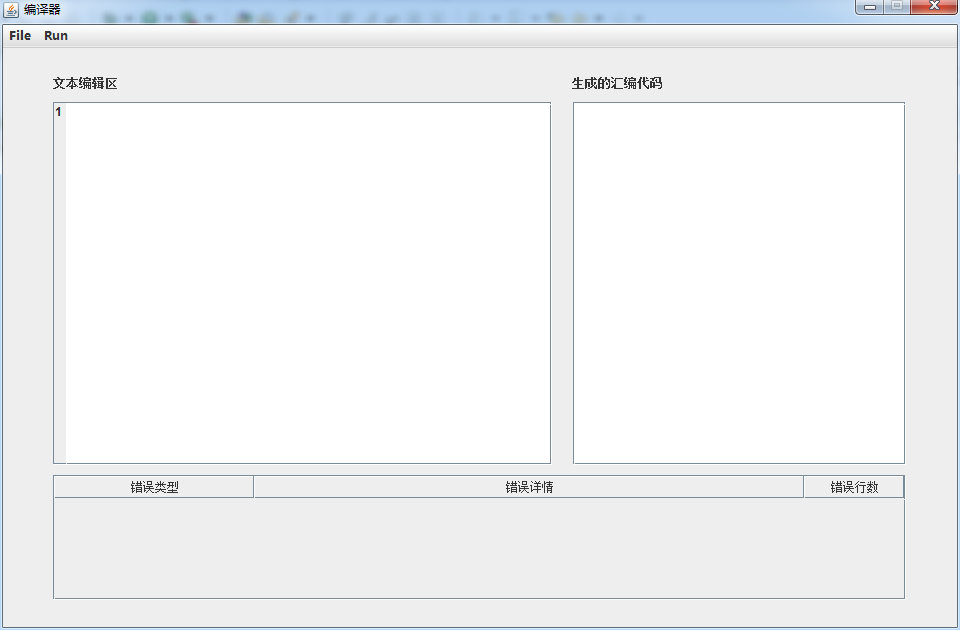


1. **实验特点及操作说明**

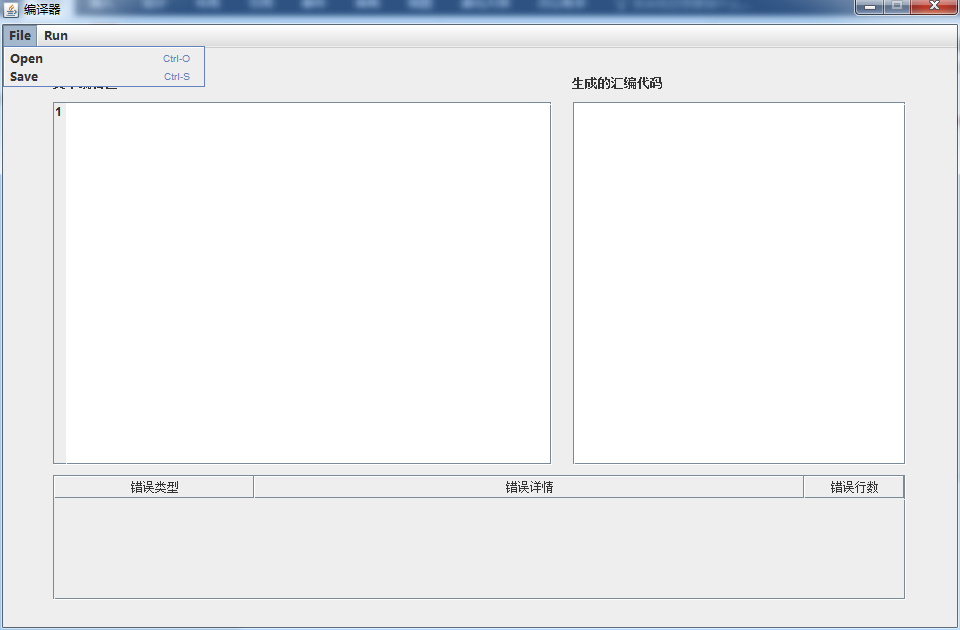
本程序的特点是使用java编写，且使用swing编写了界面。

使用说明：

1. 运行之后界面如图示：

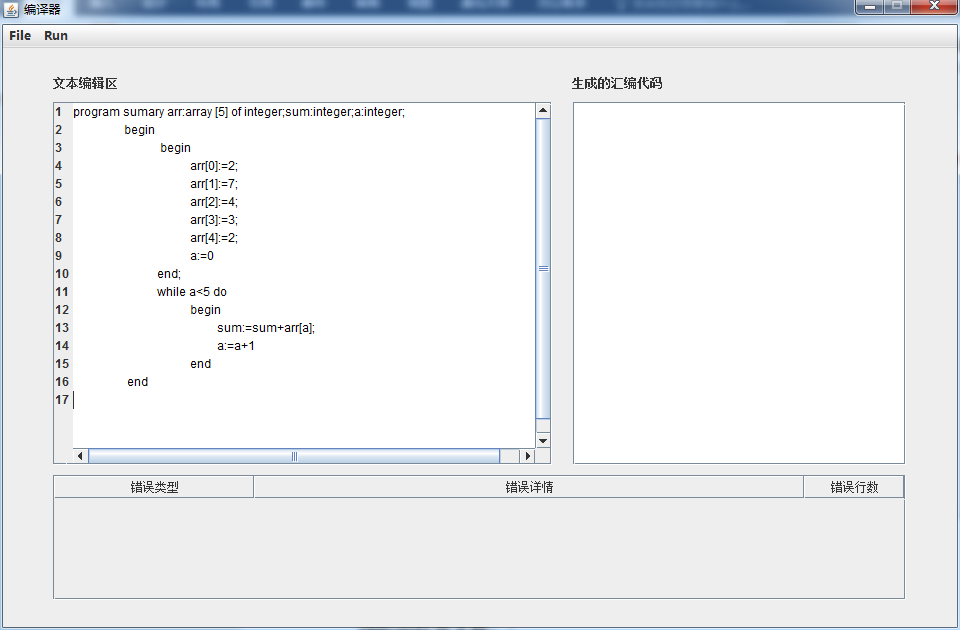


1. 点击‘File‘，出现如下图：

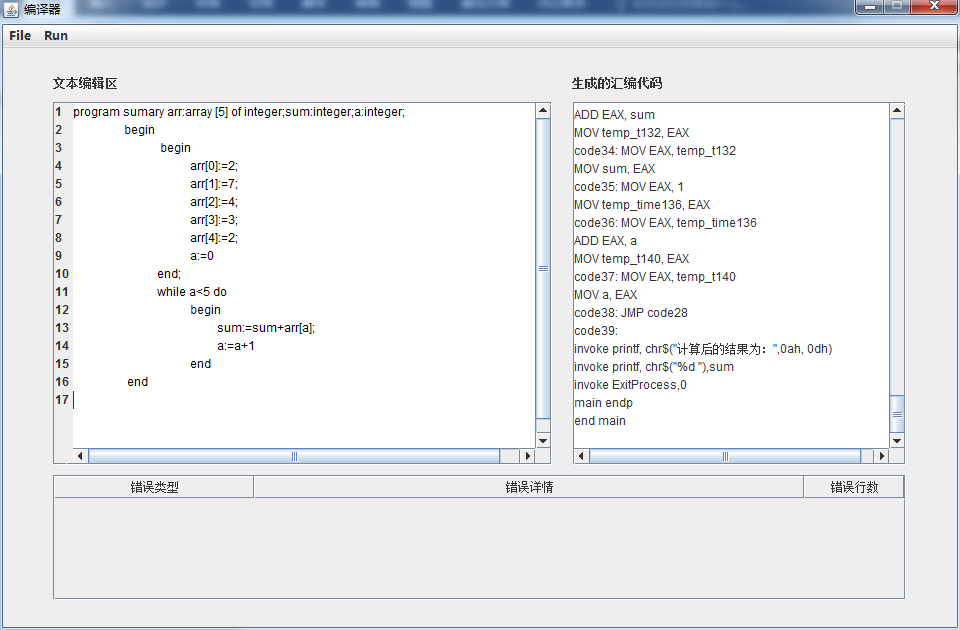


点击”open”为打开文件，点击”save”是保存“文件编辑区“的文件。

可以直接在”文件编辑区“写代码，或者使用”open”(快捷键为ctrl+o)打开文件。



3.编写完成后，点击”open”或者快捷键—ctrl+r运行程序，即可出现结果



1. **实验感悟**

经过一个学期的编译原理课程，我收获了很多，首先掌握了编译器的流程及其每一个流程中的具体实现及操作，掌握了很多新的概念，并且通过这次实验又加深了对编译原理的理解。当然这个实验中还有很多缺陷，不算是个完整的实现代码，但是基础的工作均能够实现。

另外这次实验，也让我明白了在编写大点的项目过程中报告文档的重要性以及代码书写规范的重要性，因为在隔一个时间再去看这段代码我发现自己已经又很多地方很难理解了。此外，在实现过程中也让我进一步对java有了一定的了解。

当然最重要的还是编译原理本身的一些了解，至少懂了当我去编写一个编译器的时候从何处入手，整个过程如何，词法分析器如何设计，语法分析器是采用自底向上还是自上而下的方式，中间代码如何表示，如何进行优化等以及我需要哪些资源才能入手去编写程序等。

在这个过程中当然也遇到了很多难题，如写语法分析器的时候如何存储文法，如何存储和识别和比较闭包等，还有语义分析的时候各个属性采用的参数很多，如何去设计这些参数比较合理等，但是世上无难事，最终这些问题也成功的解决了。当完成的那一刻我很兴奋！！！

感谢老师和辛苦的付出和谆谆教诲！