|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **华南农业大学数学与信息学院**  **编译原理课程设计实验**   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **学院** | | 数学与信息学院 | **班级** | 计算机科学与技术（1班） | **学号** | 201725110105 | **姓名** | 邓泽骏 | | **实**  **验**  **题**  **目** | □设计性 ■综合性 | | | | | | | | | **自**  **我**  **评**  **价** | 本人在编译原理课程实验中完成了简单语义分析实验，本人基于ruby语言开发完成，保证了全部代码均由自己开发完成。  本代码还有其他的可改进之处，如添加 else 功能，if 多条件判断，代码量缩减等 | | | | | | | | | **教**  **师**  **评**  **语** | 能够实现实验要求的功能 □全部 □部分  算法有新意 □有 □一般  程序运行通过 □全部 □部分  算法注释说明 □完善 □仅有功能说明  接口参数说明 □有 □无  按期上交文档资料及源程序 □所有 □部分  独立完成实验 □能 □不能  体现团队合作精神。 □能够 □不能 | | | | | | | | | **成绩** |  | | | | | | | | |

|  |
| --- |
|  |

目录

[一 题目要求 2](#_Toc26716192)

[1.1 小型高级语言分析器 2](#_Toc26716193)

[二．概要设计 2](#_Toc26716194)

[2.1 结构展示 2](#_Toc26716195)

[三．详细设计 3](#_Toc26716196)

[3.1 词法分析器 3](#_Toc26716197)

[3.2语句分析器 4](#_Toc26716198)

[3.3 语法翻译器 7](#_Toc26716199)

[3.4 语法分析器 8](#_Toc26716200)

[四．测试分析 9](#_Toc26716201)

[4.1 Example1（单while） 9](#_Toc26716202)

[4.2 Example2（单if） 10](#_Toc26716203)

[4.3 Example（Combine） 10](#_Toc26716204)

[4.4 Example（Wrong） 12](#_Toc26716205)

[五.总结 13](#_Toc26716206)

[参考资料 13](#_Toc26716207)

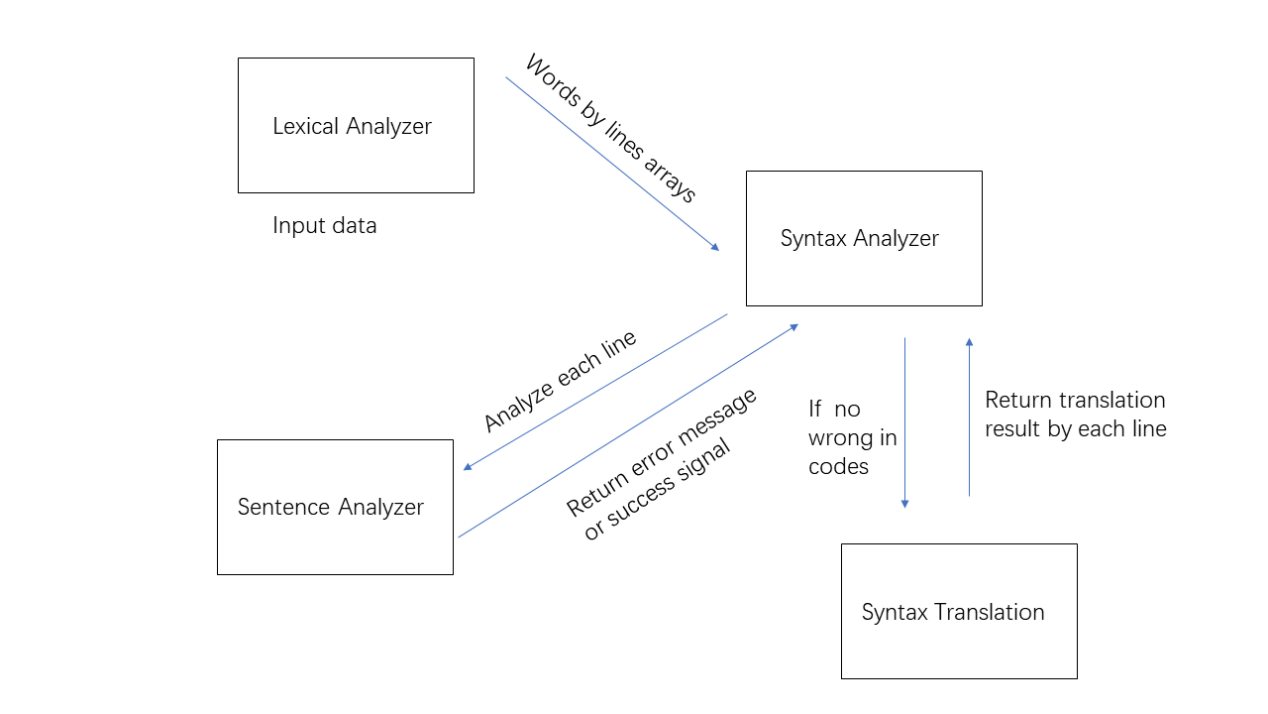
一 题目要求

1.1 小型高级语言分析器

编程语言逐渐发展的今天，种类和特性越来月多样化。结构功能越来越复杂，而它们被编译器翻译成机器能理解的机器代码或者中间代码，在此，设计并实现一个一遍扫描的编译前端，将简化高级语言的部分语法成分（含赋值语句、分支语句、循环语句等）翻译成四元式（或三地址代码），还要求有合理的语法出错报错和错误恢复功能。

二．概要设计

2.1 结构展示



其中 Lexical Analyzer 为词法分析器，Sentence Analyzer 为句子分析器,Syntax Translation 为语句翻译器，Syntax Analyzer 为主要模块，负责协调各个模块之间的协作。

三．详细设计

3.1 词法分析器

功能：负责将每行输入转换为词数组



3.1.1 判断变量名和数字的判断函数

3.2语句分析器

功能：根据词法分析器结果分析每句是否存在错误

**def** assignment\_analyzer(array)

*#@pos = array.find\_index(":=")*

*#analyzer the right part*

count\_left = 0

flag\_number = false

result = []

*#puts array.inspect*

**if** (@pos == nil)

result = error\_load(result, "miss ':='")

**return** result

**end**

**for** i **in** (@pos + 1)...array.length

**if** (array[i] == ";")

**break**

**end**

*#puts array[i].inspect*

**if** (array[i] == "(")

**if** (flag\_number)

*#error*

result = error\_load(result, ["**\'#{**array[i]**}\'**", "wrong symbol"])

**return** result

**end**

count\_left += 1

flag\_number = false

**next**

**end**

**if** (array[i] == ")")

**if** (!flag\_number)

*#error*

result = error\_load(result, ["**\'#{**array[i]**}\'**", "wrong symbol"])

**return** result

**end**

count\_left -= 1

**if** (count\_left < 0)

result = error\_load(result, "miss '('")

**return** result

**end**

flag\_number = false

**next**

**end**

**if** (LA::judge\_digit(array[i]))

flag\_number = true

**else**

**if** (LA::judge\_letter(array[i][0]))

flag\_number = true

**else**

**if** ("+-\*/%".include?(array[i]))

**if** (!flag\_number)

*#error*

result = error\_load(result, ["**\'#{**array[i]**}\'**", "wrong symbol"])

**return** result

**end**

flag\_number = false

**else**

*#error*

result = error\_load(result, ["**\'#{**array[i]**}\'**", "wrong symbol"])

**return** result

**end**

**end**

**end**

**end**

**if** count\_left > 0

result = error\_load(result, "miss ')'")

**end**

*#puts result.inspect*

*#result << "success"*

**return** result.flatten

**end**

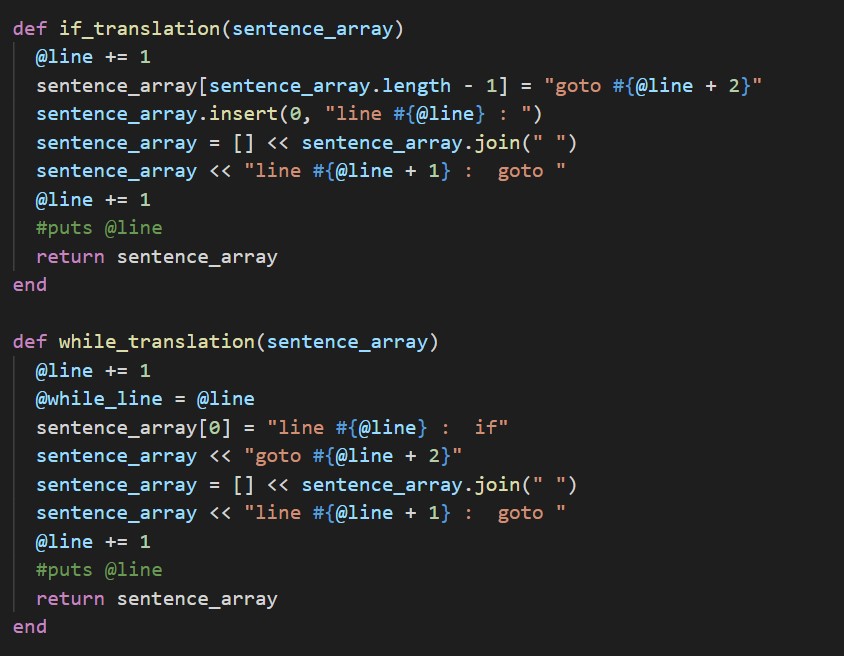
3.2.1 赋值语句判断



3.2.2 条件语句判断

3.3 语法翻译器

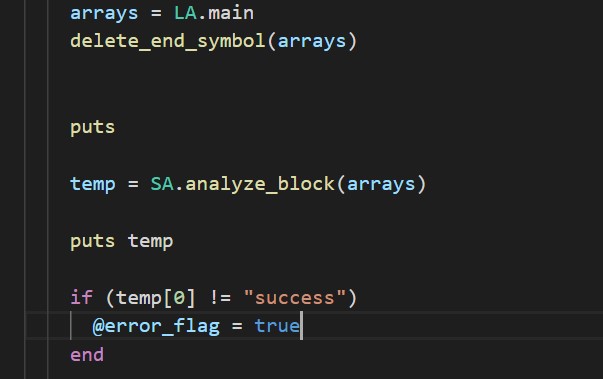
功能：负责翻译每行语句



3.3.1 if和while语句翻译

3.4 语法分析器

功能：负责协调统筹各个模块



3.4.1 分析模块协调

四．测试分析

4.1 Example1（单while）

**begin**

i:=1+20;

j:=1+30\*8;

**while** (j>3)

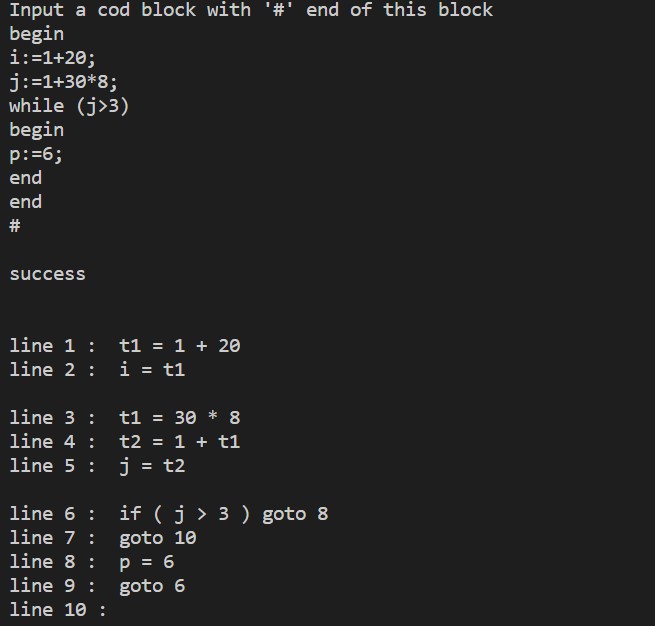
**begin**

p:=6;

**end**

**end**

*#*



4.2 Example2（单if）

**begin**

i:=1+20;

j:=1+30\*8;

**if** (j>3) **then**

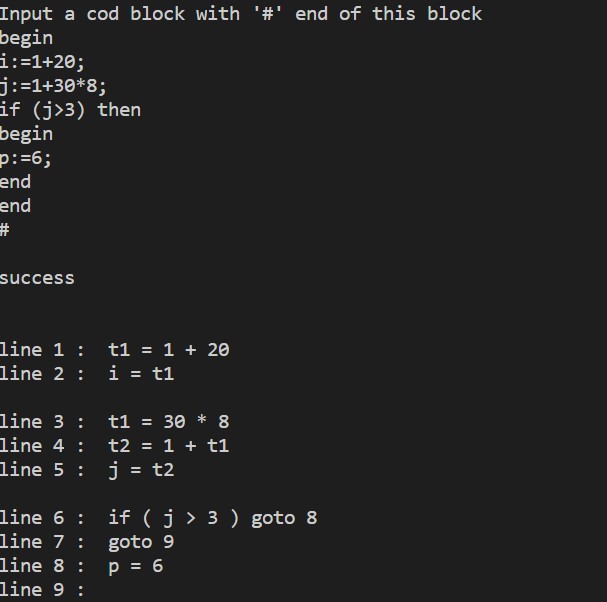
**begin**

p:=6;

**end**

**end**

*#*



4.3 Example（Combine）

**begin**

i:=1+20;

j:=1+30\*8;

**if** (i<=2) **then**

**begin**

i:= 2/9\*3+(1+3)

**while** (i<9)

**begin**

**if** (j>3) **then**

**begin**

p:=6;

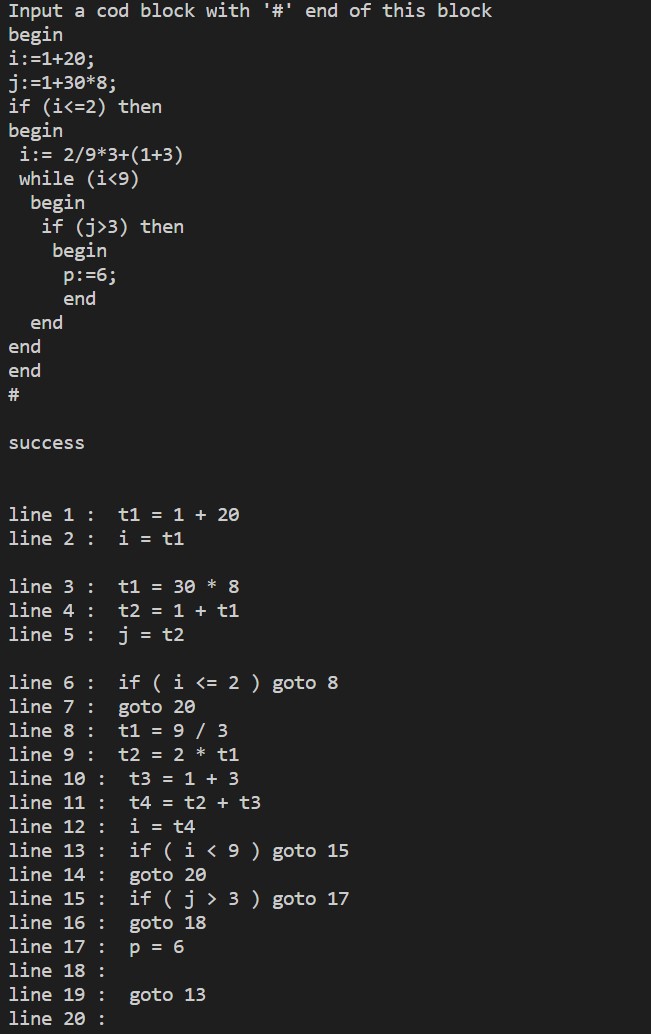
**end**

**end**

**end**

**end**

*#*



4.4 Example（Wrong）

**begin**

i:=1+20;

j:=1+30)\*8;

**if** (i<=(2) **then**

**begin**

i:= 2/9\*3+(1+3)

**while** (i<9)

**begin**

**if** (j(>3) **then**

**begin**

p:1=6;

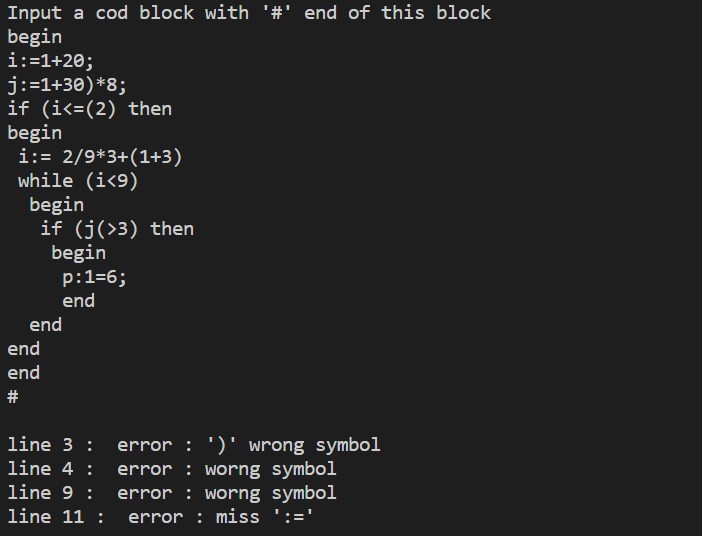
**end**

**end**

**end**

**end**

*#*



五.总结

在此次实验过程中，我充分地了解到了基于“递归下降法”的编译器的一点点实现过程，更加了解了ruby语言的语言特性

参考资料

<https://stackoverflow.com/questions/1235863/how-to-test-if-a-string-is-basically-an-integer-in-quotes-using-ruby> 判断字符串是否是一个数字

<https://www.thoughtco.com/making-deep-copies-in-ruby-2907749> ruby 深拷贝

<https://www.runoob.com/ruby/ruby-module.html> ruby module 菜鸟教程

<https://ruby-doc.org/core-2.5.0/Module.html> ruby Module official

<https://ruby-doc.org/core-2.4.1/Array.html> ruby Array official

<https://ruby-doc.org/core-2.6/String.html> ruby String official

<https://www.rubyguides.com/2015/10/ruby-case/> ruby Case Statement tutorial

<https://www.cnblogs.com/lulipro/p/7450886.html> 逆波兰变换