# 

# 《编译原理》

# 综合实验报告

* 课 程 编 译 原 理
* 实验名称 编译原理综合实验
* 专 业 计算机
* 班 级 计算机1301
* 姓 名 匡俊
* 学 号 131310120
* 完成报告日期 2015年6月30日

**目录**

[《编译原理》 1](#_Toc30744)

[综合实验报告 1](#_Toc6484)

[一、实验名称 2](#_Toc18477)

[二、实验目的 3](#_Toc18024)

[三、实验内容及要求 3](#_Toc9237)

[四、实验环境 5](#_Toc22503)

[五、系统架构 5](#_Toc5137)

[1、词法分析器 5](#_Toc24092)

[2、语法分析器 5](#_Toc13331)

[3、类型检查程序 5](#_Toc18135)

[4、四元式产生 6](#_Toc32190)

[5、错误处理 6](#_Toc17375)

[六、软件结构 6](#_Toc18347)

[七、 实验数据、结果分析 7](#_Toc13741)

[八、总结 9](#_Toc19366)

[九、参考资料 9](#_Toc17707)

# 

# 一、实验名称

编译原理综合实验

# 二、实验目的

C语言实现一个相对完整的编译器设计，包括语法分析、词法分析、语义检查以及四元式生成

# 三、实验内容及要求

（一） 词法分析程序的设计与实现：

**要求**：设计一个词法分析程序，每调用一次就从源程序文件中顺序识别出一个单词符号。

单词种类与识别规则

标识符：首字符为字母或’$’，其后由字母、数字或’$’组成；

整 数：由一个或多个数字组成、无正负号的数字串，首位数字不能为0；

小 数：整数1 **·**整数2

**注意**：整数1不能为空，整数2可以为空，例如：23.

保留字：begin 、as、sub、end、if、then、else、call、while、do、integer、float、input、output、var

单目运算符：＋ － \* / = < >

双目运算符：<= >= <> ==

布尔运算符：and or

⑧ 界符： ( ) ， ；

此外，该词法分析程序还要能支持单行注释和多行注释（注释语法同C语言）。

（二）语法分析程序的设计与实现：

**要求**：构造下列文法的语法分析程序，应能指出源程序中首次出现的错误及出错位置。

1. <程序>—> <过程> [<过程>]
2. <过 程>—> **sub** **id** <参数列表> <语句> **end sub**
3. <参数列表>—> **(** **id** **as** <数据类型> [**，id** **as** <数据类型>] **)**

| ε

1. <语句>—> <分支语句>

| <赋值语句>

| <循环语句>

| <输入语句>

| <输出语句>

| <函数调用语句>

| <变量定义语句>

| <复合语句>

|ε

1. <复合语句>—> **begin** <语句>[ **,** <语句> ] **end**
2. <变量定义语句>—> **var** **id** [ **, id** ] **as** <数据类型> **；**
3. <数据类型>—> **integer** | **float**
4. <输入语句>—> **input** **id** [ **, id** ] **；**
5. <输出语句>—> **output** <表达式>[ **,** <表达式> ] **；**
6. <赋值语句>—> **id =** <表达式> **；**
7. <函数调用语句>—> **call** **id (** <传递参数> **) ；**
8. <传递参数>—> **id** [ **, id** ]

| ε

1. <分支语句>—> **if** <布尔表达式> **then** <语句>

**| if** <布尔表达式> **then** <语句> **else** <语句>

1. <循环语句>—> **while** <布尔表达式> **do** <语句>
2. <表达式>—> <项> [ **+**|**－** <项> ]
3. <项>—> <因子> [ **\***|**/** <因子> ]
4. <因子>—> **id** | **con** | **deci** | **(** <表达式> **)**

14. <布尔表达式>—> <关系表达式> [ **and** | **or** <布尔表达式> ]

15. <关系表达式>—> <表达式> <关系> <表达式>

16. <关系>—> **<** | **<=** | **>** | **>=** | **==** | **<>**

说明：

(1) id是标识符；con是常量，即无符号整数；deci是小数；

(2) 用左右尖括号<>括起来的是非终结符；

(3) 红色方括号[ ]相当于正则表达式中的()\*，表示其中内容重复0次或N多次；

(4) 文法中用蓝色标注的都是终结符（即单词）；

(5) 文法中and与or不存在优先级，两者优先级别一样，按左结合处理

（三） 类型检查程序的设计与实现：

**要求**：

（1）检查变量的定义和使用是否正确：变量是否重复定义、变量是否未定义就使用；

（2）过程的名称是否重复定义；

（3）过程的实参与形参的数目是否一致；

（4）过程的实参与形参的数据类型是否一致。

（四） 四元式翻译程序的设计与实现：

**要求**：将语法和语义都正确的源程序翻译成四元式。

# 四、实验环境

操作系统：Windows 7

编译工具：codeblocks 13.12 / Mingw

# 五、系统架构

## 1、词法分析器

将文件中的字符一个一个读入，当读到空格或换行说明一个单词读入结束，分析改单词的类型。将单词类型和单词保留用于语法分析。

## 2、语法分析器

按照递归下降法写语法分析：

按照语法规则，首先根据第一个单词判断匹配那种句型，然后根据该句型逐个匹配，看是否匹配，对于每个非终结符都有一个函数来描述他的产生式。按照递归下降的方法实现语法分析。

## 3、类型检查程序

（1）过程是否重复定义，调用过程时参数传递是否正确的检查：

用一个结构数组把所有定义过的过程名称及参数存好，在分析定义过程的语句时，首先看这个过程名称是否使用过，如果使用过则报错。在调用函数时，检查结构体内储存的参数情况是否一致。

（2） 变量名重复定义，变量未定义就使用，变量使用是否正确的检查

使用STL库中的容器set，由于set自动排序，查找效率为O（nlogn）， 因此使用set来储存变量，set中存的结构体包括变量名和变量类型， 按照变量名的字典序进行排序，每次分析含有变量（标识符）的语句， 判断是否使用过或者是否使用类型正确，每次检测定义变量的语句时检 查是否定义过

## 4、四元式产生

本实验的难点就在四元式产生，虽然做的过程中出现很多问题，但都修 复了。接下来阐述我四元式如何产生的：

使用的主要是双栈算法，即把中缀表达式转成后缀表达式的算法。主要是把操作数和操作符分别存在两个栈内，再以一定的方法弹出和压进，具体方法的不加以叙述，网上有十分详细的描述。

对于if和while表达式的回填，我的做法是把所有未填的表达式压入栈中，在遇到恰当时机，把表达式弹出，从下往上填写result。

## 5、错误处理

错误恢复策略：

每出现一个错误直接报错然后退出程序。

# 六、软件结构

**1、程序功能**

**基本实现上述所有功能**

**3、程序流程**

**1）读取源文件**

**2）语法分析（词法分析为语法分析服务，类型检查、四元式 产生在语法分析时实现）**

**3）输出四元式**

**4、代码结构**

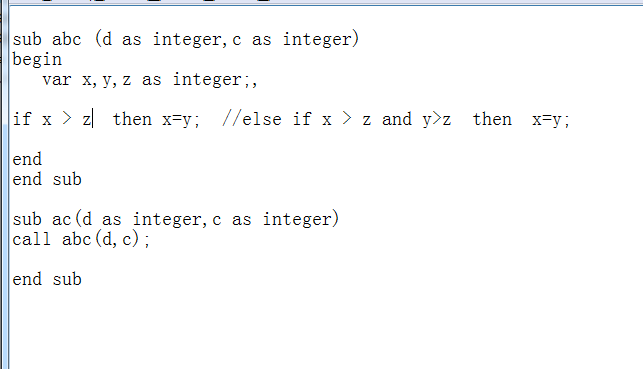
**1）词法分析器、语法分析器（包含了类型检查以及四元式产生**

**2）头文件“data.h”中包括很多定义**

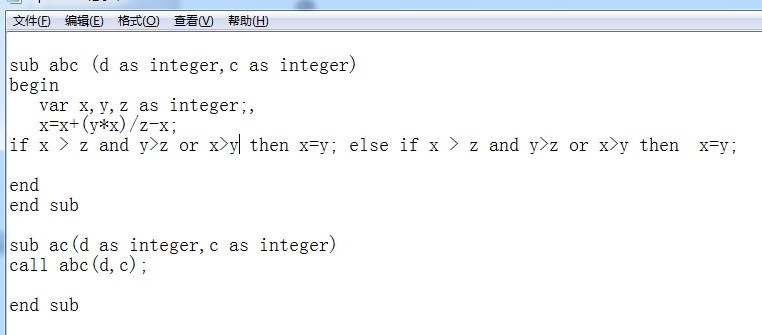
# 实验数据、结果分析

**1、测试数据(附件)：**

**①**

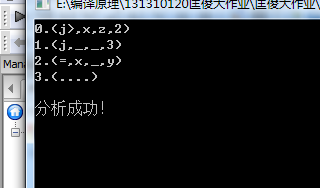


**②**

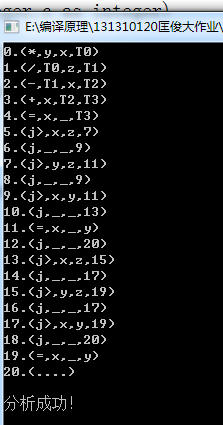


**2、实验结果**

**①**



②



# 八、总结

实验要求基本符合，这次实验使我更加清楚了编译器编译的流程，本次实验无论是代码量还是复杂度都是我大学以来最难的一次实验，最后虽然可能有点小问题（比如在检查时发现的问题，后来得到改正），但是基本功能都的以实现。

# 九、参考资料

**《编译原理》第二版**