**Pascal-S编译器需求分析**

1. **引言**
2. **任务分析**

此项目是开发一个Pascal-S语言编译器，用于完成从Pascal-S语言到C语言的编译任务。

1. **项目名称**

Pascal-S编译器

1. **小组成员**

徐永杰、甘钊宇、黄勇康、田哲元、贾东港、田国梁

1. **开发环境**

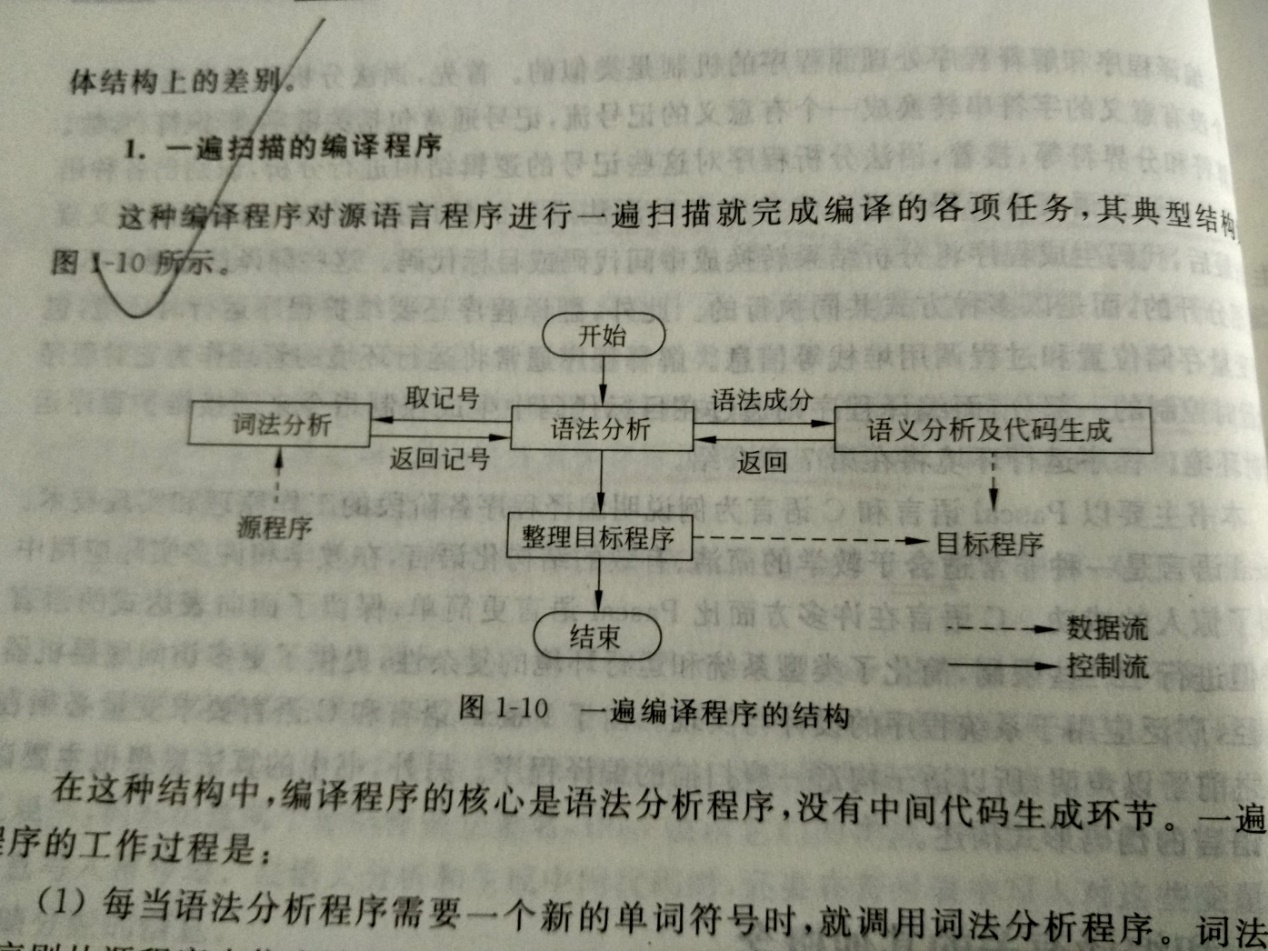
Windows操作系统，借助Flex/Bison (LEX/YACC) 进行语法分析，使用Microsoft Visual Studio 集成开发环境。

1. **运行环境**

Windows操作系统

1. **数据流图与功能划分**

根据编译器的一般设计架构，本编译器按功能划分为词法分析、语法分析、语义分析和代码生成四个部分。各部分之间的数据流如下：



1. **词法分析**
2. **需求分析**

词法分析主要任务是从左到右逐个字符地对源程序进行扫描，按照Pascal语言的词法规则识别出一个个单词符号，把识别出来的标识符存入符号表中，并产生用于语法分析的记号序列，在词法分析过程中还可以完成用户接口有关的一些任务，如跳过注释和空格，把来自编译程序的错误信息和源程序联系起来，如记住单词在源程序中的行/列位置，从而行号可以作为错误信息的一部分提示给用户。

**2. 单词种类、单词模式与右线性文法**

**(1) 关键字**

Pascal语言中共有三十五个关键字，关键字作为保留字,分为六类列出：

* 程序、函数和过程内的起始符号：program，function，procedure；
* 说明部分专用定义符： array，const，file，label，packed，var，record，set，type；
* 语句用符；case，of，do，for，while，repeat，until，if，then， else，to，downto，goto，with；
* 运算符：and，or，not，div，mod，in；
* 分隔符号：begin，end；
* 空指针常量：nil；

**(2) 标识符**

标识符：它是用来标识程序、函数、过程、类型、常量、变量等名字，Pascal中允许的有效长度为8个字符，可使用英文字母、数字、下划线，但必须以字母开头，后面可跟字母或数字，中间不能含空格。它有标准标识符与自定义标识符两种。

1. 标准标识符是系统预先定义好的标识符，它们有特定的含义，按规定共有四十个。

* 标准常量符：false，true，maxint；
* 标准类型名：boolean，char，integer，real，text；
* 标准文件名：input，output；
* 标准函数名：(算术函数)abs，sqr，sqrt，exp，ln，sin，cos，tan， arctan，random，frac，oddeof，eoln等；
* （转换函数）ord，pred，round，chr，succ，trunc；
* 标准过程名：read，readln，write，writeln，put，dispose，get，new， pack，page，reset，rewrite，unpack

标识符的记号id，匹配以字母开头的字母数字串。

右线性文法：

Letter->[a-zA-Z]

Digit->[0-9]

Id->letter(letter|digit)\*

1. 自定义标识符：它是由用户自己定义的标识符，它可以是变量名、常量名或过程名。但要注意，禁用关键字，不能数字开头，不含空格及非字符非数字的字符。

**(3) 算术运算符**

+、-、\*、/（除）、div（整除）、mod（求余）等6个.

**(4) 关系运算符**

relop -> =|<>|<|<=|>|>=

Addop-> +|- | or

Mulop-> \*| / |div|mod | and

Assignop-> :=

**(5) 常量与变量**

常量：指在程序运行过程中不能被修改的量。在Pascal中的常量有：整型、实型、布尔型和字符型、符号常量、字符串常量等六种。

变量:在程序运行过程中,其值可以改变的量为变量。变量有 变量名、变量类型和变量值等三个要素。

1. **注释与分隔符**

源程序中的关键字（除开头的program和末尾的end之外）前、后必须有空格符或换行符，其它词汇间的空格符是可选的。源程序中的注释：用一对花括号括起来，可以出现在任何单词之后。编译程序应该可以处理注释。

1. **错误处理**

如果词法分析程序从源程序中读入了一个不合法的字符即Pascal语言不包括此字符开头的单词符号。词法分析显示打印错误信息，并跳过这个字符，然后转开始状态继续识别和分析下一个单词符号。

1. **语法分析**
2. **需求分析**

根据Pascal-S语言的语法规则，从词法分析程序产生的记号序列中识别出各种语法成分，同时进行语法检查，为语义分析和代码生成做准备。

输入：记号序列；输出：分析树。

1. **语法结构与文法**
2. **（主）程序结构**

program <程序名>(input, output);

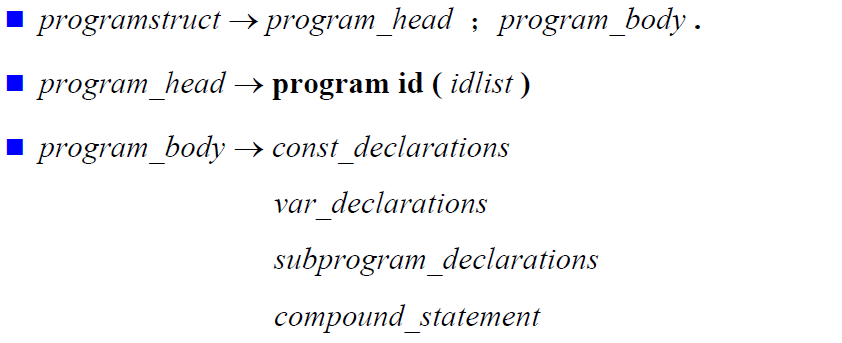
<常量声明>

<变量声明>

<函数定义/过程定义>

<主程序块>

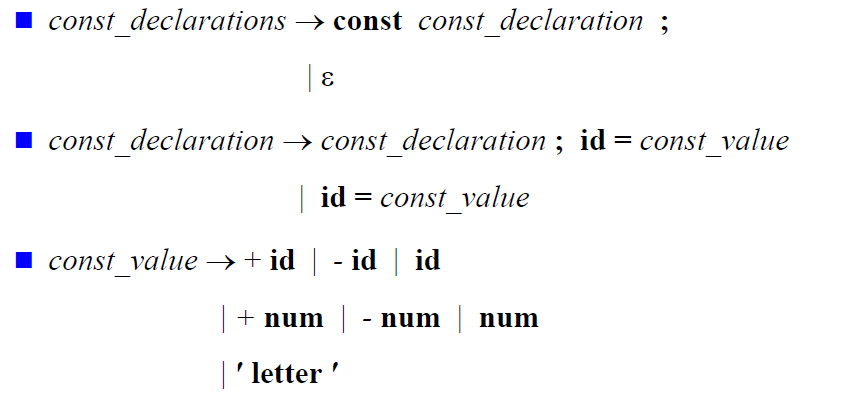
文法：



1. **常量定义**

const <常量名1>=<值1>; …; <常量名n>=<值n>;

值可以是数字也可以是字符串，不用声明数据类型（仅允许数字和字符）。

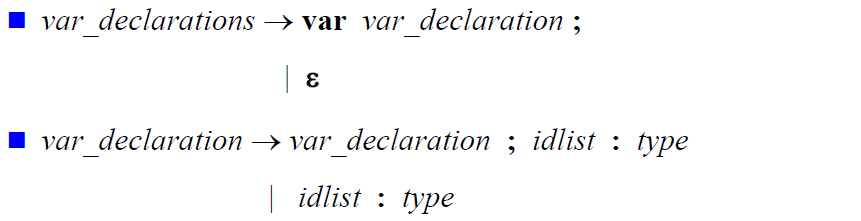


1. **变量定义**

var <变量列表1>:<类型1>; …; <变量列表n>:<类型n>;

变量列表中的变量用逗号隔开。 real / integer / char / boolean

文法：



1. **表达式/赋值语句**

<变量>:=<表达式>

表达式有算术表达式和逻辑表达式。



1. **输入/输出语句**

输入：read(<变量表>); readln(<变量表>)

输出：write(<输出表>); writeln(<输出表>)

Pascal-S暂不支持。

1. **选择结构**

* if 语句

if <条件> then

<语句1>

else

<语句2>



* case 语句

case <表达式> of

<值表1>: <语句1>

<值表2>: <语句2>

…

<值表n>: <语句n>

[else <语句>]

end

Pascal-S暂不支持。

1. **循环结构**

* for 语句

for <循环变量>:=<初值> to <终值> do <语句>

文法：



for <循环变量>:=<终止> downto <初值> do <语句>

Pascal-S暂不支持。

* while 语句

while <条件> do

<循环体>

Pascal-S暂不支持。

* repeat 语句

repeat

<循环体>

until <条件>

Pascal-S暂不支持。

1. 函数与过程结构

* 函数

function <函数名> (<形式参数表>): <返回值类型>;

<常量声明>;

<变量声明>;

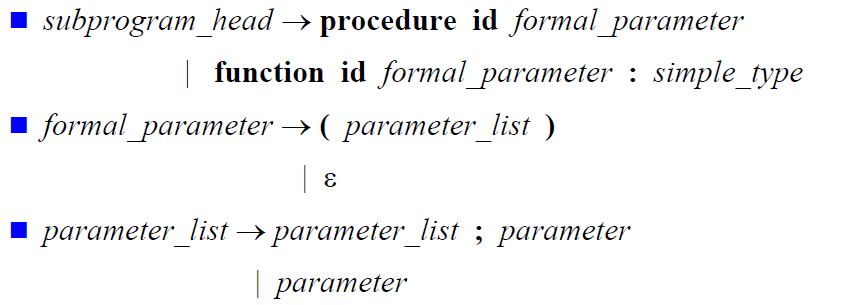
begin

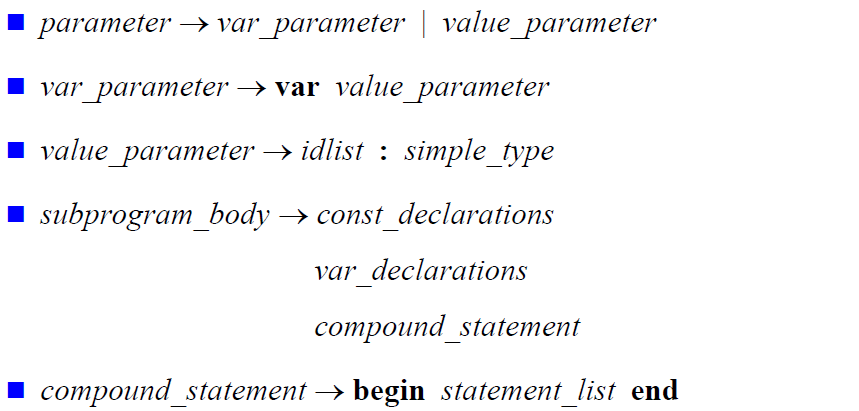
<函数体>

end

形式参数表与声明变量的文法一致

文法：





* 过程

procedure <过程名> (<形式参数表>);

<常量声明>;

<变量声明>;

begin

<过程体>

end

文法同上。

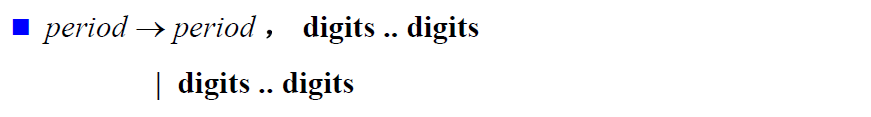
1. **自定义数据类型**

type <类型名>=<类型>

* 子界类型

<上界> .. <下界>

文法：



* 枚举类型

(<枚举列表>)

Pascal-S暂不支持。

* 集合类型

set of <类型名>

Pascal-S暂不支持。

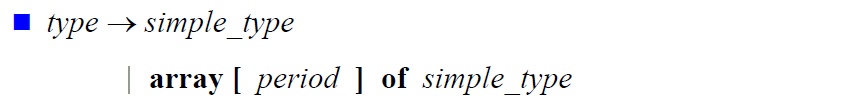
* 指针类型

^<类型名>

Pascal-S暂不支持。

* 数组类型

array [<子界1>,…,<子界n>] of <类型名>



* 记录类型

Record

<变量名1>:<类型1>;

…

<变量名n>:<类型n>;

End;

Pascal-S暂不支持。

* 文件类型

file of <类型名>

Pascal-S暂不支持。

1. **语法错误类型**

* **缺少元素**：缺少“(”或“)”，算术表达式缺少运算对象，过程/函数定义没有过程体，缺少分号 等

例如： 2+3)\*5、6/

program HelloWorld(input, output);

{没有过程体}

* **多余的元素**：多余的“(”或“)”，if...else不匹配 等

if a<b then

c:=a

else

c:=b

else {多余的元素}

c:=0

* **错误**

for i:=1 tooooooooooooooooo 9 do

readln(a[i])

1. **分析方法**

借助YACC工具自动分析。

1. **改写文法**

目前首要目标是实现上述所有基本文法，然后视工程进度进行文法扩展，实现其它Pascal-S未支持的语法。

**五、 语义分析**

**1. 需求分析**

在语法分析的基础上，对抽象语法树进行分析，并生成报错信息。

1. **类型**

显式声明，通过类型表达式进行判断。

1. **类型检查**

静态类型检查。

**（1）需要检查程序内的类型信息是否一致；**

**（2）类型表达式分为不同类型：**

a). 基本类型；

b). 类型名；

c). 类型构造器作用于类型表达式的结果；

d). 类型表达式可以包含变量，变量的值是类型表达式；

**（3）类型等价的判定：**

a). 结构等价的判定；

b). 名字等价的判定；

**4. 符号表**

**(1) 符号表内容**

* + - 名字
    - 类型
    - 存储地址
    - 维数及参数个数
    - 声明行
    - 引用行
    - 链域

**(2) 符号表组织**

采用栈式散列符号表；通过块索引表进行定位以及重定位操作。

**5．作用域的判定**

通过定位以及重定位操作进行作用域的判定。

**六、 代码生成**

**1. Pascal-s — C语言的映射关系**

programstruct → program\_head ；program\_body .

{

program\_head(转换为相应C头文件);

program\_body

}

program\_head → program id ( idlist )

{与C语言不同，转换为C语言格式}

program\_body →

const\_declarations{注意相应的变量作用域以及实现}

var\_declarations{注意相应的变量作用域以及实现}

subprogram\_declarations

compound\_statement

idlist → idlist , id | id

const\_declarations → const const\_declaration ; {相应作用域转换}| ∅

const\_declaration → const\_declaration ; id = const\_value | id = const\_value 

const\_value → + id | - id | id | + num | - num | num | ′ letter ′ {转换为c语言相应格式}

var\_declarations → var var\_declaration ; | ∅

{var\_declarations → var\_declaration ; | ∅}

var\_declaration → var\_declaration ; idlist : type | idlist : type

{ idlist : type变换为 type idlist }

type → simple\_type | array [ period ] of simple\_type

{变换为相应的C语言规范}

simple\_type → integer | real | boolean | char

period → period ， digits .. digits | digits .. digits

subprogram\_declarations →subprogram\_declarations subprogram; | ∅

{subprogram\_declarations → subprogram; | ∅}

subprogram → subprogram\_head ; subprogram\_body

subprogram → subprogram\_head

{

subprogram\_body

}

subprogram\_head → procedure id formal\_parameter

| function id formal\_parameter: simple\_type

subprogram\_head →simple\_type function id formal\_parameter

formal\_parameter → ( parameter\_list ) | ∅

parameter\_list → parameter\_list ; parameter | parameter

parameter → var\_parameter | value\_parameter

var\_parameter→ var value\_parameter

{

var.type value\_parameter

}

value\_parameter→idlist : simple\_type

subprogram\_body→const\_declarations

var\_declarations

compound\_statement

compound\_statement→begin statement\_list end

{

statement\_list（并根据C语言相应规范进行对应）

}

statement\_list→statement\_list ; statement | statement

statement→variable assignop expression

| procedure\_call

| compound\_statement

| if expression then statement else\_part

{

If(expression){statement}

else\_part

}

| for id assignop expression to expression do statement

{

For(id=;expression to expression)

{ statement}

}

|∅ 

variable→id id\_varpart

id\_varpart →[ expression\_list ] | ∅

procedure\_call→id | id ( expression\_list )

else\_part→else statement | ∅

expression\_list→expression\_list , expression | expression

expression→simple\_expression relop simple\_expression

| simple\_expression

simple\_expression→simple\_expression addop term | term

term→term mulop factor | factor

factor→num

| variable

| id ( expression\_list )

| ( expression )

| not factor

| uminus factor

**2. Pascal-S—C语言的语法映射关系**

有 ~~删除线~~ 的项目是当前基本Pascal-S语言不支持的语法，需要扩展文法。

* **主程序结构**

Pascal-S代码：

program <程序名>(input, output);

<常量声明>

<变量声明>

<函数定义/过程定义>

<主程序块>

C代码：

<全局常量声明>

<全局变量声明>

<函数定义>

int main(int argc, char\* argv[])

{

<主程序块>

return 0;

}

* **常量定义**

Pascal-S代码：

const <常量名1>=<值1>; …; <常量名n>=<值n>;

C代码：

const <常量1类型> <常量名1>=<值1>;

…

const <常量n类型> <常量名n>=<值n>;

* **变量定义**

Pascal-S代码：

var <变量列表1>:<类型1>; …; <变量列表n>:<类型n>;

C代码：

<类型1> <变量列表1>;

…

<类型n> <变量列表n>;

* **表达式/赋值语句**

Pascal-S代码：<变量名>:=<表达式>

C代码：<变量名>=<表达式>

* **if 语句**

Pascal-S代码：

if <条件> then

<语句1>

else

<语句2>

C代码：

if (<条件>)

<语句1>

else

<语句2>

* **~~case 语句（分支判断）~~**

Pascal-S代码：

case <表达式> of

<值表1>: <语句1>

<值表2>: <语句2>

…

<值表n>: <语句n>

[else <语句>]

end

C代码：

switch (<表达式>)

{

/\* 值表1 \*/

case <值1-1>:

case <值1-2>:

…

case <值1-n>:

<语句1>

break;

…

/\* 值表n \*/

case <值n-1>:

…

case <值n-n>:

<语句n>

break;

[default: <语句>]

}

* **for语句**

Pascal-S语言：

for <循环变量>:=<初值> to <终值> do <语句>

~~for <循环变量>:=<初值> downto <终值> do <语句>~~

C语言：

for (<循环变量>:=<初值>; <循环变量><=<终值>; <循环变量>++)

<语句>

for (<循环变量>:=<初值>; <循环变量>>=<终值>; <循环变量>--)

<语句>

* **~~while语句~~**

Pascal-S语言：

while <条件> do

<循环体>

C语言：

while (<条件>)

<循环体>

* **~~repeat语句（do-while循环）~~**

Pascal-S语言：

repeat

<循环体>

until <条件>

C语言：

do

<循环体>

while (<条件>)

* **函数**

Pascal-S语言：

function <函数名> (<形式参数表>): <返回值类型>;

<常量声明>;

<变量声明>;

begin

<函数体>

end

C语言：

<返回值类型> <函数名> (<形式参数表>)

{

<常量声明>;

<变量声明>;

<函数体>

}

* **过程**

Pascal-S语言：

procedure <过程名> (<形式参数表>);

<常量声明>;

<变量声明>;

begin

<过程体>

end

C语言：

void <过程名> (<形式参数表>)

{

<常量声明>

<变量声明>

<过程体>

}

**七、测试用例**

以下提供三个正确的测试用例，用于测试各模块是否能正确工作。测试过程中，可以在代码中加入错误因子，以检查各模块的错误处理机制是否能够正常工作。

1. **你好世界！**

program HelloWorld;

begin

writeln('Hello world!')

end.

1. **辗转相除法求最大公约数**

program example(input,output);

var x,y:integer;

function gcd(a,b:integer):integer;

begin

if b=0 then gcd:=a

else gcd:=gcd(b, a mod b)

end;

begin

read(x, y);

write(gcd(x, y))

end.

1. **快速排序（需要更多Pascal语法支持）**

program sort (input, output);

var a: array[0..10] of integer;

x: integer;

procedure readarray;

var i: integer;

begin

for i:=1 to 9 do read(a[i])

end;

procedure exchange (i,j: integer);

begin

x:=a[i]; a[i]:=a[j]; a[j]:=x

end;

procedure quicksort (m,n:integer);

var k: integer;

function partition (y,z: integer): integer;

var i,j: integer;

var pivot: integer;

begin

pivot:=a[y];

i:=y;

j:=z;

while i<j do

begin

while (i<j) and (a[i] <= pivot) do i:=i+1;

while (i<j) and (a[j] >= pivot) do j:=j-1;

if (i<j) then exchange(i, j);

end;

exchange(i, y);

exit(i);

end;

begin

if (m >= n) then exit();

k:=partition(m, n);

quicksort(m, k-1);

quicksort(k+1, n)

end;

begin

writeln('Please input a sequence:');

readarray;

quicksort(1, 9);

for x:=1 to 9 do write(a[x])

end.