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