<字母> ::= \_｜a｜…｜z｜A｜…｜Z

<数字> ::= ０｜<非零数字>

<非零数字> ::= 1 | … | 9

词法分析部分：

<+ -> ::= +｜-

<\* /> ::= \*｜/

<!=> ::= <｜<=｜>｜>=｜!=｜==

Character ::= '<+ ->'｜'<\* />'｜'<字母>'｜'<数字>'

string ::= "{十进制编码为32,33,35-126的ASCII字符}"

Integer ::= <非零数字> {<数字>}

Id ::= <字母> {<字母>｜<数字>}

Zero ::= 0

语法分析部分：

<常量说明> ::= const<常量定义>;{ const<常量定义>;}

<常量定义> ::= int Id= ([<+ ->] Integer｜0) { , Id= ([<+ ->] Integer｜0) }

| char Id=Character{,Id=Character}

<变量说明> ::= <变量定义>;{<变量定义>;}

<变量定义> ::= (int | char) (Id | Id‘[’Integer‘]’) { , (Id | Id‘[’Integer‘]’ )}

<因子> ::= Id｜Id‘[’<表达式>‘]’ | ‘(’<表达式>‘)’｜[<+ ->] Integer｜0 | Character

| <有返回值函数调用语句>

<项> ::= <因子> {<\* /> <因子>}

<表达式> ::= [<+ ->] <项> {<+ -> <项>}

<条件> ::= <表达式> <!=> <表达式>｜<表达式> //只有表达式为0才为假

<语句> ::= <条件语句>｜<循环语句>| ‘{’ {<语句>} ‘}’｜<有返回值函数调用语句>; | <无返回值函数调用语句>;｜<赋值语句>;｜<读语句>;｜<写语句>;

｜<空>;|<情况语句>｜<返回语句>;

<赋值语句> ::= Id = <表达式> | Id ‘[’<表达式>‘]’ = <表达式>

<条件语句>::= if ‘(’<条件>‘)’<语句>else<语句>

<循环语句> ::= while ‘(’<条件>‘)’<语句>

<情况表> ::= case ( [<+ ->] Integer｜0 | Character) : <语句>

{ case ( [<+ ->] Integer｜0 | Character) : <语句>}

<缺省> ::= default : <语句>

<情况语句> ::= switch ‘(’<表达式>‘)’ ‘{’<情况表>[<缺省>] ‘}’

<读语句> ::= scanf ‘(’Id {, Id}‘)’

<写语句> ::= printf ‘(’ string,<表达式> ‘)’ | printf ‘(’string ‘)’| printf ‘(’<表达式>‘)’

<返回语句> ::= return [‘(’<表达式>‘)’]

<参数> ::= (int | char) Id { , (int | char) Id}

<值参数表> ::= <表达式>{ , <表达式> }

<有返回值函数定义> ::= (int | char) Id ‘(’<参数>‘)’ ‘{’<复合语句>‘}’

| (int | char) Id ‘{’<复合语句>‘}’

//第一种选择为有参数的情况，第二种选择为无参数的情况

<无返回值函数定义> ::= void Id ‘(’<参数>‘)’‘{’<复合语句>‘}’

| void Id ‘{’<复合语句>‘}’

//第一种选择为有参数的情况，第二种选择为无参数的情况

<有返回值函数调用语句> ::= Id‘(’<值参数表>‘)’ | Id

//第一种选择为有参数的情况，第二种选择为无参数的情况

<无返回值函数调用语句> ::= Id‘(’<值参数表>‘)’ | Id

//第一种选择为有参数的情况，第二种选择为无参数的情况

<复合语句> ::= [<常量说明>][<变量说明>]{<语句>}

<主函数> ::= void main ‘(’ ‘)’ ‘{’<复合语句>‘}’

<程序> ::= [<常量说明>][<变量说明>]{<有返回值函数定义> | <无返回值函数定义>}

<主函数>

注意：

<有返回值函数定义> ::= (int | char) Id ‘(’<参数>‘)’ ‘{’<复合语句>‘}’

| (int | char) Id ‘{’<复合语句>‘}’

//第一种选择为有参数的情况，第二种选择为无参数的情况

<变量说明> ::= <变量定义>;{<变量定义>;}

<变量定义> ::= (int | char) (Id | Id‘[’Integer‘]’) { , (Id | Id‘[’Integer‘]’ )}

改写成:

先读进(int | char) Id ，然后：

为 ( 是 <有返回值函数定义>

为 , [ ; 是 <变量说明>

注意：

<主函数> ::= void main ‘(’ ‘)’ ‘{’<复合语句>‘}’

<无返回值函数定义> ::= void Id ‘(’<参数>‘)’‘{’<复合语句>‘}’

| void Id ‘{’<复合语句>‘}’

//第一种选择为有参数的情况，第二种选择为无参数的情况

改写成：

先读进void , 然后：

为main 是 <主函数>

为 Id 是 <无返回值函数定义>