代码生成作业要求及文法语义说明：

请在词法分析和语法分析的基础上，为编译器实现语义分析、代码生成功能。输入输出及处理要求如下：

（1）需根据文法规则及语义约定，采用自顶向下的语法制导翻译技术，进行语义分析并生成目标代码；

（2）如下两项要求任选其一实现：

1）完成编译器和解释执行程序，将源文件（统一命名为 testfile.txt）编译生成 PCODE 目标代码并解释执行，得到解释执行的结果（以标准输出的形式给出），具体要求包括：

a）需自行设计 PCODE，可参考 PASCAL-S 编译器的处理

b）将生成的 PCODE 直接进行解释执行，在提交的作业中不需要输出 PCODE

2）完成编译器，将源文件（统一命名为 testfile.txt）编译生成 MIPS 汇编（统一命名为 mips.txt），具体要求包括：

a）需自行设计四元式中间代码，再从中间代码生成 MIPS 汇编，中间代码的设计格式要求见文件“中间代码格式要求 2019.docx”

b）若选择此项任务，后续的作业需参加竞速排序，请提前预留代码优化有关的接口，并设计方便切换开启/关闭优化的模式

c）自行调试时，可使用 Mars 仿真器（使用方法见“Mars 仿真器使用说明.docx”），提交到平台的编译器只需要能按统一的要求生成 MIPS 汇编代码文件即可

＜加法运算符＞ ::= +｜-

＜乘法运算符＞ ::= *｜/

＜关系运算符＞ ::= <｜<=｜>｜>=｜!=｜==

＜字母＞ ::= _｜a｜...｜z｜A｜...｜Z

＜数字＞ ::= 0｜＜非零数字＞

＜非零数字＞ ::= 1｜...｜9

＜字符＞ ::= '＜加法运算符＞'｜'＜乘法运算符＞'｜'＜字母＞'｜'＜数字＞'

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

＜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＞

＜常量说明＞::= const＜常量定义＞;{ const＜常量定义＞;}

＜常量定义＞ ::= int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞}

| char＜标识符＞＝＜字符＞{,＜标识符＞＝＜字符＞}

＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝| 0

＜整数＞ ::= ［＋｜－］＜无符号整数＞

＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝ //标识符和保留字都区分大小写

＜声明头部＞ ::= int＜标识符＞ |char＜标识符＞

＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}

＜变量定义＞ ::= ＜类型标识符＞(＜标识符＞|＜标识符＞'['＜无符号整数＞']'){,(＜标识符＞|＜标识符＞'['＜无符号整数＞']')}

//＜无符号整数＞表示数组元素的个数，其值需大于 0

//变量没有初始化的情况下没有初值

＜类型标识符＞ ::= int | char

＜有返回值函数定义＞ ::= ＜声明头部＞'('＜参数表＞')' '{'＜复合语句＞'}'

＜无返回值函数定义＞ ::= void＜标识符＞'('＜参数表＞')''{'＜复合语句＞'}'

＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞

＜参数表＞ ::= ＜类型标识符＞＜标识符＞{,＜类型标识符＞＜标识符＞}| ＜空＞

＜主函数＞ ::= void main‘(‘)’ ‘{’＜复合语句＞‘}’

＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞} //[+|-]只作用于第一个<项>

＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}

＜因子＞　::= ＜标识符＞｜＜标识符＞'['＜表达式＞']'|'('＜表达式＞')'｜＜整数＞|＜字符＞｜＜有返回值函数调用语句＞　//char 类型的变量或常量，用字符的 ASCII 码对应的整数参加运算

//＜标识符＞'['＜表达式＞']'中的＜表达式＞只能是整型，下标从 0 开始

//单个＜标识符＞不包括数组名，即数组不能整体参加运算，数组元素可以参加运算

＜语句＞　::= ＜条件语句＞｜＜循环语句＞| '{'＜语句列＞'}'| ＜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＞;|＜返回语句＞;

＜赋值语句＞　::= ＜标识符＞＝＜表达式＞|＜标识符＞'['＜表达式＞']'=＜表达式＞

//＜标识符＞＝＜表达式＞中的<标识符>不能为常量名和数组名

＜条件语句＞ ::= if '('＜条件＞')'＜语句＞［else＜语句＞］

＜条件＞　::= ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞ //表达式需均为整数类型才能进行比较，第二个侯选式中表达式为 0 条件为假，否则为真

＜循环语句＞　::= while '('＜条件＞')'＜语句＞| do＜语句＞while '('＜条件＞')' |for'('＜标识符＞＝＜表达式＞;＜条件＞;＜标识符＞＝＜标识符＞(+|-)＜步长＞')'＜语句＞　　//for 语句先进行条件判断，符合条件再进入循环体

＜步长＞::= ＜无符号整数＞

＜有返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

＜无返回值函数调用语句＞ ::= ＜标识符＞'('＜值参数表＞')'

＜值参数表＞　::= ＜表达式＞{,＜表达式＞}｜＜空＞

//实参的表达式不能是数组名，可以是数组元素

//实参的计算顺序,要求生成的目标码运行结果与 Clang8.0.0 编译器运行的结果一致

＜语句列＞ ::= ｛＜语句＞｝

＜读语句＞　::= scanf '('＜标识符＞{,＜标识符＞}')'

//从标准输入获取<标识符>的值，该标识符不能是常量名和数组名

//生成 PCODE 代码的情况：需要处理为一个 scanf 语句中，若有多个<标识符>，无论标识符的类型是 char 还是 int，每输入一项均需回车

//生成 MIPS 汇编的情况：按照 syscall 指令的用法使用即可

＜写语句＞　::= printf '(' ＜字符串＞,＜表达式＞ ')'| printf '('＜字符串＞ ')'| printf '('＜表达式＞')'

//printf '(' ＜字符串＞,＜表达式＞ ')'

//printf '(' ＜字符串＞,＜表达式＞ ')'输出时，先输出字符串的内容，再输出表达式的值，两者之间无空格

//表达式为字符型时，输出字符；为整型时输出整数

//＜字符串＞原样输出（不存在转义）

//每个 printf 语句的内容输出到一行，按结尾有换行符\n 处理

＜返回语句＞　::= return['('＜表达式＞')']

//无返回值的函数中可以没有 return 语句，也可以有形如 return;的语句

//有返回值的函数只要出现一条带返回值的 return 语句即可，不用检查每个分支是否有带返回值的 return 语句

另：关于类型和类型转换的约定：

1. 表达式类型为 char 型有以下三种情况：

1）表达式由<标识符>或＜标识符＞'['＜表达式＞']'构成，且<标识符>的类型为 char，即 char 类型的常量和变量、char 类型的数组元素。

2）表达式仅由一个<字符>构成，即字符字面量。

3）表达式仅由一个有返回值的函数调用构成，且该被调用的函数返回值为 char 型

除此之外的所有情况，<表达式>的类型都是 int

2. 只在表达式计算中有类型转换，字符型一旦参与运算则转换成整型，包括小括号括起来的字符型，也算参与了运算，例如('c')的结果是整型。

3. 其他情况，例如赋值、函数传参、if/while 条件语句中关系比较要求类型完全匹配，并且＜条件＞中的关系比较只能是整型之间比，不能是字符型，if '('＜条件＞')'和 while '('＜条件＞')'里边，如果<条件>是单个表达式，则必须是整型。