**Gramática de MiniC#**

Características Generales:

* Un programa en MiniC# consiste de una clase con variables globales y métodos. Podrían importarse otras clases más que podrían ser instanciadas como tal. De forma interna solamente se podrán declarar otras clases que se utilizarán como estructuras de almacenamiento de datos (al estilo registro en C o Go).
* El método principal de un programa en MiniC# se llama siempre “Main()” y se ejecuta siempre que se invoque al programa. Un programa que no tiene Main, podría utilizarse como librería para ser importado en otra clase.
* MiniC# tiene:
  + Valores constantes “char”, “int”, “double”, “bool” y “string”.
  + Variables: de clase (interna o externa), parámetros o variables locales a métodos.
  + Tipos básicos: int, char(Unicode 2 bytes), double y bool
  + Tipos estructurados: listas unidimensionales-indexadas-dinámicas (que permita solo tipos simples) y clases internas con campos pero sin métodos.
  + Procedimientos predeclarados para listas: “add”, “del” y “len”.
* Metasímbolos:
  + [] significa opcional – o sea 0 o 1 vez
  + {} significa 0 o más veces
  + “” denotan los terminales (tokens)
  + | denota opciones de regla

Program = {Using} "class" ident "{" { VarDecl | ClassDecl | MethodDecl } "}"

Using = "using" ident ";"

VarDecl = Type ident { "," ident } ";"

ClassDecl = "class" ident "{" { VarDecl } "}"

MethodDecl = ( Type | "void" ) ident "(" [ FormPars ] ")" Block

FormPars = Type ident { "," Type ident }

Type = ident [“[” “]”]

Statement = Designator ( "=" Expr | "(" [ ActPars ] ")" | "++" | "--" ) ";"

| "if" "(" Condition ")" Statement [ "else" Statement ]

| “for” “(“ Expr “;” [Condition] “;” [Statement] ) Statement

| "while" "(" Condition ")" Statement

| "break" ";"

| "return" [ Expr ] ";"

| "read" "(" Designator ")" ";"

| "write" "(" Expr [ "," number ] ")" ";"

| Block

| ";"

Block = "{" { VarDecl | Statement } "}"

ActPars = Expr { "," Expr }

Condition = CondTerm { "||" CondTerm }

CondTerm = CondFact { "&&" CondFact }

CondFact = Expr Relop Expr

Cast = "(" Type ")"

Expr = [ "-" ] [Cast] Term { Addop Term }

Term = Factor { Mulop Factor }

Factor = Designator [ "(" [ ActPars ] ")" ]

| number

| charConst

| StringConst

| (true | false)

| "new" ident

| "(" Expr ")"

Designator = ident { "." ident | "[" Expr "]" }

Relop = "==" | "!=" | ">" | ">=" | "<" | "<="

Addop = "+" | "-"

Mulop = "\*" | "/" | "%"

### Estructura Léxica

Definidas por el programador siguiendo el formato de C#.

Comentarios: iniciando en /\* y terminando en \*/, además de línea //. Son permitidos comentarios anidados