Especificación de Gramática del Subconjunto del Lenguaje C en notación BNF

Introducción

Este documento presenta la especificación de la sintaxis para un subconjunto del lenguaje de programación C, detallando su gramática formal en notación BNF

Gramática para expresiones aritméticas y lógicas

Notación BNF

```
<Expr> ::= <Term> <Expr'>
<Expr'> ::= <PlusExpr'> | <MinusExpr'> | <LogicalExpr> | ""
<PlusExpr'> ::= "mas" <Term> <Expr'>
<MinusExpr'> ::= "menos" <Term> <Expr'>

<Term> ::= <Factor> <Term'>
<Term'> ::= <MultTerm'> | <DivTerm'> | <ModTerm'> | ""
<MultTerm'> ::= "multiplicacion" <Factor> <Term'>
<DivTerm'> ::= "division" <Factor> <Term'>
<ModTerm'> ::= "modulo" <Factor> <Term'>

<Factor> ::= "constante_entera" | "constante_flotante" | <Ope
nExpr>
<OpenExpr> ::= "(" <Expr> ")"

<LogicalExpr> ::= "operador_y" <Expr> | "operador_o" <Expr>
```

Gramática para definición de estructuras de control y bloques de código

Notación BNF

```
<BLOQUE> ::= <INSTRUCCION_B> <BLOQUE> | ""
<INSTRUCCION_B> ::= <INSTRUCCION>
                  | "condicion_if" "(" <EXPRESION> ")" <INSTR
UCCION_C> <BLOQUE_ELSE>
                  | "bucle_while" "(" <EXPRESION> ")" <INSTRU
CCION_C>
                  | "bucle_do" <INSTRUCCION_C> "bucle_while"
"(" <EXPRESION> ")" ";"
                  | "bucle_for" "(" <INSTRUCCION> <identifica
dor> <EXPRESION> ";" <identificador> <EXPRESION> ")" <INSTRUC
CION_C>
```

```
<INSTRUCCION_C> ::= "condicion_if" "(" <EXPRESION> ")" <INSTR
UCCION_C> "condicion_else" <INSTRUCCION_C>
                  | "bucle_while" "(" <EXPRESION> ")" <INSTRU
CCION_C>
                  | "bucle_do" <INSTRUCCION_C> "bucle_while"
"(" <EXPRESION> ")" ";"
                  | "bucle_for" "(" <INSTRUCCION> <identifica
dor> <EXPRESION> ";" <identificador> <EXPRESION> ")" <INSTRUC
CION_C>
                  | <INSTRUCCION>
<BLOQUE_ELSE> ::= "condicion_else" <COLA_ELSE> | ""
<COLA_ELSE> ::= "condicion_if" "(" <EXPRESION> ")" <COLA_ELSE
> | <INSTRUCCION>
<INSTRUCCION> ::= "i" | "{" <BLOQUE> "}"
<EXPRESION> ::= "o" | ""

<programa> ::= <bloque>
<bloque> ::= "{" <lista_instrucciones> "}"
<lista_instrucciones> ::= <instruccion> <lista_instrucciones>
| ε
<instruccion> ::= <if_statement> | <while_statement> | <do_wh
ile_statement> | <for_statement> | <instruccion>
<if_statement> ::= "if" "(" <exp_bool> ")" <bloque> | "if" "
(" <exp_bool> ")" <bloque> "else" <bloque>
<while_statement> ::= "while" "(" <exp_bool> ")" <bloque>
<do_while_statement> ::= "do" <bloque> "while" "(" <exp_bool>
")" ";"
<for_statement> ::= "for" "(" <instruccion> ";" <exp_bool>
";" <instruccion> ")" <bloque>
```

Gramática para declaraciones de variables, inicializaciones, y declaraciones de funciones, así como la especificación de tipos y parámetros.

Notación BNF

```
<PROGRAMA> ::= <_GLOBAL>
<_GLOBAL> ::= <GLOBAL> <_GLOBAL> | ""
<GLOBAL> ::= <TIPO> <identificador> <DECLARACION_GLBL>
           | "void" <identificador> "(" <PARAMETROS> ")" <FUN
CIONG_COLA>
<DECLARACION_GLBL> ::= <ASIGNACION> <_ASIGNACION_CONST> ";"
                     | "(" <PARAMETROS> ")" <FUNCIONG_COLA>
<FUNCIONG_COLA> ::= ";" | "{" <BLOQUE> "}"
<_ASIGNACION_CONST> ::= "," <ASIGNACION_CONST> <_ASIGNACION_C
ONST> | ""
<ASIGNACION_CONST> ::= <identificador> <ASIGNACION>
<EXPRESION_CONST> ::= <constante_entera> | <constante_charact
```

```
er> | <constante_flotante>
<ASIGNACION> ::= "=" <EXPRESION_CONST> | ""
<PARAMETROS> ::= <TIPO> <identificador> <_PARAMETROS> | ""
<_PARAMETROS> ::= "," <TIPO> <identificador> <_PARAMETROS> |
""
<TIPO> ::= "int" | "float" | "char"
<BLOQUE> ::= "b"
```

Gramática para declaraciones, instrucciones, y estructuras básicas de control y funciones.

Notación BNF

```
<S> ::= <INSTRUCCION> <_INSTRUCCION>
<_INSTRUCCION> ::= <INSTRUCCION> <_INSTRUCCION> | ""
<INSTRUCCION> ::= <identificador> <EXPRESION> ";"
                | <RETORNO> ";"
                | <TIPO> <identificador> <DECLARACION>
                | "void" <identificador> "(" <PARAMETROS> ")"
<FUNCION_COLA>
<DECLARACION> ::= <ASIGNACION> <_DECLARACION_CONT> ";"
                | "(" <PARAMETROS> ")" <FUNCION_COLA>
<FUNCION_COLA> ::= ";" | "{" <_BLOQUE> "}"
<_DECLARACION_CONT> ::= "," <DECLARACION_CONT> <_DECLARACION_
CONT> | ""
<DECLARACION_CONT> ::= <identificador> <ASIGNACION>
<ASIGNACION> ::= "=" <EXPRESION> | ""
<PARAMETROS> ::= <TIPO> <identificador> <_PARAMETROS> | ""
<_PARAMETROS> ::= "," <TIPO> <identificador> <_PARAMETROS> |
""
<TIPO> ::= "int" | "float" | "char"
<RETORNO> ::= "return" <EXPRESION>
<EXPRESION> ::= <ASIGNACION> | <LLAMAR_FUNCION>
<LLAMAR_FUNCION> ::= "(" <ARGUMENT> ")"
<ARGUMENT> ::= <EXPRESION> <_ARGUMENT> | ""
<_ARGUMENT> ::= "," <EXPRESION> <_ARGUMENT> | ""
<EXPRESION> ::= "e"

<INSTRUCCION> ::= <_INSTRUCCION> <INSTRUCCION> | ""
<_INSTRUCCION> ::= <DECLARACION> ";"
                 | <identificador> <ID> ";"
                 | <RETURN_I> ";"
<ID> ::= <D_INIT> | <FUNTION_CALL>
<TIPO> ::= "int" | "float" | "char" | "void"
<RETURN_I> ::= "return" <EXPRESION>
<FUNTION_CALL> ::= "(" <ARGUMENT> ")"
<ARGUMENT> ::= <EXPRESION> <A_ARGUMENT> | ""
<A_ARGUMENT> ::= "," <EXPRESION> <A_ARGUMENT> | ""
```

```
<DECLARACION> ::= <TIPO> <identificador> <D_INIT>
<D_INIT> ::= "=" <EXPRESION> | ""
```