Nombre del equipo

Los Vengadores.

Nombre de los integrantes y rol respectivo

- Adrián Álvarez Rodríguez: Clarificador
- Ana Laura Monge Soto: Desarrolladora
- Hilary Ovares Varga: Generadora
- Jose Andrés Mejías Rojas: Implementador.

Nombre del lenguaje de programación

JAAH

Gramática:

```
method:
      cambios_linea ini def_method cambios_linea
      instrucciones cambios_linea
      FIN id
      |;
def_method:
      id dosp concat_ids
      | id dosp
      ;
concat_ids:
      id com concat_ids
      | id
      ;
```

```
rule_for:
      for  dosp  assign  com  comparaciones  com  operaciones  PARI
cambios_linea
instrucciones PARD
      ;
assign:
      id igual id
      | id igual num
      | id igual BOOL
      | id igual operaciones
      ;
comparaciones:
      comparacionesbooleanas |
      comparaciones AND_OR comparacionesbooleanas
      ;
comparacionesbooleanas:
            tipos comparacion tipos
      ;
tipos:
      id
      | num
      ;
operaciones:
      operando operar operando
      | operaciones operar operando
      ;
llamado_metodo:
      ID ':' listarID |
      ID ':' |
      operaciones_en_metodo
      ;
```

```
instrucciones:
      assign cambios_linea instrucciones
      | rule_for cambios_linea instrucciones
      | SALIDA dosp ID cambios_linea instrucciones
      | SALIDA dosp num cambios_linea instrucciones
      | SALIDA dosp HILERA cambios_linea instrucciones
      | cambios_linea
      ;
cambios_linea:
      LF cambios_linea
      | LF
      ;
id:
      ID {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
      ;
ini:
      INI {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
      ;
num:
      NUM {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
      ;
com:
      COM {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
;
dosp:
      DOSP {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
```

```
    ;
igual:
    IGUAL {$$ = strdup(yytext);} | ERROR {printf("Error
Léxico\n"); exit(-1);}
    ;
comparacion:
    COMPARACION {$$ = strdup(yytext);} | ERROR {printf("Error
Léxico\n"); exit(-1);}
    ;
for:
    FOR {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
    ;
operar:
    OPERACIONES {$$ = strdup(yytext);} | ERROR {printf("Error
Léxico\n"); exit(-1);}
    ;
fin:
    FIN INI {$$ = strdup(yytext);} | ERROR {printf("Error
Léxico\n"); exit(-1);}
    ;
bool:
    BOOL {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
    ;
hilera:
    HILERA {$$ = strdup(yytext);} | ERROR {printf("Error
Léxico\n"); exit(-1);}
    ;
```

```
and_or:
     AND_OR   {$$   =   strdup(yytext);}   |   ERROR   {printf("Error
Léxico\n"); exit(-1);}
;
pari:
     PARI {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
     ;

pard:
     PARD {$$ = strdup(yytext);} | ERROR {printf("Error Léxico\n");
exit(-1);}
     ;
salida:
     SALIDA   {$$   =   strdup(yytext);}   |   ERROR   {printf("Error
Léxico\n"); exit(-1);}
     ;
```

Instrucciones para ejecución del programa:

-Requiere tener Bison y flex instalados y la carpeta bin de éstos en el path de las variables de entorno del computador.

-Dado que el programa tiene un make, para compilarlo solamente es necesario abrir en terminal la ruta donde están los archivos Expresiones.flex y Gramaticas.Bison y escribir make.

-Para ejecutar el programa:

Windows: Abrir en consola la ruta del ejecutable y escribir:

parser.exe < nombreDelArchivoARevisar.extensión

Linux: Abrir en consola la ruta del ejecutable y escribir:

./parser.exe < nombreDelArchivoARevisar.extensión