Nombre del equipo

Los Vengadores.

Nombre de los integrantes y rol respectivo

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

Nombre del lenguaje de programación

JAAH

Gramática:

```
super:
	inicio
	;

inicio:
	concat_links principal
	| principal
	;

principal:
	declarations method principal
	| declarations method
	;

declarations:
	assign PYC declarations
	| ID PYC declarations
	|
	;

assign:
	ID IGUAL tipos
	| ID IGUAL operaciones
	| arreglo IGUAL tipos
	| arreglo IGUAL operaciones
	ID IGUAL method_call;
	;
```

```
concat_links:
	LINK JAAH concat_links
	| LINK JAAH
	;

operaciones:
	operando OPERACIONES operando {if($2[0] == '/' && $2[0] == '/'){printError("Error
logico, division por cero",'b');};}
	| operaciones OPERACIONES operando {if($2[0] == '/' && $2[0] ==
'/'){printError("Error logico, division por cero",'b');};}
	;

tipos:
	operando
	| BOOL
	| HILERA
	;

operando:
	ID
	| NUM
	| arreglo
	| ID PUNTO ID
	| POINTER method_call
	| ID NUM //Error cuando se pone identificador - num, no recoger ud y num negativo
	;


instrucciones:
	assign PYC instrucciones
	| SALIDA DOSP tipos PYC instrucciones
	| rule_for instrucciones
	| rule_ELSE instrucciones
	| rule_IF instrucciones
	| rule_While instrucciones
	| ENTRADA DOSP ID PYC instrucciones
	| ret PYC instrucciones
	| PYC instrucciones
	| method_call PYC instrucciones
	|
	;

method:
	INI method_call DOSP
	instrucciones FIN ID
	;

method_call:
	ID method_arguments
	;
```

```
method_arguments:
        PARI concat_IDs PARD
        | PARI concat_operations PARD
        | PARI PARD
        ;

concat_IDs:
        tipos COM concat_IDs
        | tipos
        ;

concat_operations:
        operaciones COM concat_operations
        | operaciones
        ;

rule_for:
        FOR DOSP assign COM comparaciones COM operaciones PARI instrucciones PARD
        ;

comparaciones:
        comparacionesbooleanas |
        comparaciones AND_OR comparacionesbooleanas
        ;

comparacionesbooleanas:
        tipos COMPARACION tipos
        | tipos COMPARACION operaciones
        | operaciones COMPARACION tipos
        | operaciones COMPARACION operaciones
        ;

ret:
        RET tipos | RET operaciones | RET comparaciones
        ;

rule_IF:
        IF DOSP comparaciones PARI instrucciones PARD
        ;

rule_While:
        WHILE DOSP comparaciones PARI instrucciones PARD
        ;

rule_ELSE:
        rule_IF ELSE PARI instrucciones PARD
        ;

arreglo:
        ID CORI tamano_arreglo CORD
        ;
```

```
tamano_arreglo:
        operando
        | operaciones
        ;
```

Instrucciones para ejecución del programa:

Windows:

-Escribir en la terminal los siguientes comandos de compilación:

bison -d newWhole.bison -ogramaticas.tab.c

flex newOld.flex

gcc -oparser.exe lex.yy.c gramaticas.tab.c

-Luego de compilado, puede ejecutarlo escribiendo el consola:

parser.exe < nombreDelARchivoAAnalizar.extensiónDelMismo.

Linux:

-Escribir make en consola para compilar.

-Luego de compilado, puede ejecutarlo escribiendo el consola:

./parser.exe < nombreDelARchivoAAnalizar.extensiónDelMismo.

Cambios:

-Se agrega el ";" para delimitar una línea ya que era muy complicado usar el cambio de línea, de ésta forma podemos hacer nuestras gramáticas más entendibles.

-Se agrega el "->" para indicar que se está haciendo el llamado a una función, ya que sin él nos estaba dando problemas nuestro lenguaje por conflictos con los usos que le damos al identificador.

-Se cambia la inicialización de los métodos, ahora para hacerlo se debe hacer de la siguiente forma:

ini identificador (parámetros) :

Este cambio lo hicimos para poder delimitar dónde inician las acciones del método.

Errores manejados:

Imprime en pantalla un mensaje que indica el token inesperado y los posibles tokens que iban en ese campo. También imprime un error lógico (o semántico) en el caso de división por cero, pero éste no bota el programa.

Finalmente, en el caso que se ponga ID NUM, si el número es positivo se indica que es un error sintáctico y se detiene el análisis, en caso que el número sea negativo se toma como una operación y continúa el análisis de forma normal.

Para todos los errores anteriores, se imprime en pantalla la línea donde se encuentra.