PROYECTO FINAL

COMPILADOR

>Trevilla Figueroa Christopher
>Ríos Díaz De León José Pablo
>Jiménez Zempoalteca Uriel
>Sánchez Cortés Aarón
>Páez Villafuerte Eithel Agustín
>Gonzalez Gonzalez Cristian Alan

Documentación De Proyecto De Software (Compilador)

(1) Definición General Del Compilador

Idea General Del Compilador (Funcionalidad):
Lenguaje de programación interpretado, enfocado a la enseñanza de la programación, que toma como principio el patrón minimalista. Soporta las funciones básicas de un lenguaje tales como: Variables,Cadenas,If,While.

Objetivos: Realizar un compilador capaz de recibir como entrada un conjunto de caracteres que serán asignados por el usuario, para que se genere como salida un código objeto el cual será representado por gramática binaria y hexadecimal.

Dicho compilador será capaz de revisar línea por línea el código para comunicar al usuario los errores, los cuales pueden ser gramaticales.

Finalmente con ello lograr la intercomunicación del desarrollador de software hacia la computadora de una manera más simple a la que se está acostumbrado, con ello optimizando el tiempo de codificación en el desarrollo de software.

Personas O Entidades (Quienes Lo Podrán Utilizar):
Cualquier usuario con conocimientos mínimos en programación

(2) Definición De Requerimientos Del Compilador

Requisitos Generales:

>Solamente un sistema con plataforma de Windows, o similar capaz de ejecutar programas .jar, de lo contrario la aplicación no podrá ser ejecutada

Requisitos Funcionales:

>Sistema Windows o Sistema Linux
>Ejecutable del compilador

Información De Autoría:

>Trevilla Figueroa Christopher	*(Líder De Proyecto / Programador)*
>Ríos Díaz De León José Pablo	*(Programador)*
>Jiménez Zempoalteca Uriel	*(Programador)*
>Sánchez Cortés Aarón	*(Aspectos Técnicos)*
>Páez Villafuerte Eithel Agustín	*(Aspectos Técnicos)*
>Gonzalez Gonzalez Cristian Alan	*(Aspectos Técnicos)*

Alcances Del Compilador:

Conseguir la realización de un compilador que realice las distintas etapas que conlleva las cuales son:

Analizador Léxico

Leer caracteres de entrada hasta identificar los componentes léxicos.
Analizar léxicamente un conjunto de caracteres.
Elaborar como salida la secuencia de componentes léxicos para ser utilizados por el analizador sintáctico.

Analizador Sintáctico

Identificar que los tokens formen una expresión válida usando una gramática definida.
Identificar el orden en que aparecen los tokens.

Analizador Semántico

Dar sentido a la expresión evaluada, a su construcción y a su estructura en el código.

(3) Especificaciones De Procedimientos

A) Procedimientos De Desarrollo

Herramientas Utilizadas

ANTLR fue la herramienta utilizada para desarrollar nuestro compilador.

Planificación

Responsables Y Equipo De Trabajo

>Trevilla Figueroa Christopher	*(Líder De Proyecto / Programador)*
>Ríos Díaz De León José Pablo	*(Programador)*
>Jiménez Zempoalteca Uriel	*(Programador)*
>Sánchez Cortés Aarón	*(Aspectos Técnicos)*
>Páez Villafuerte Eithel Agustín	*(Aspectos Técnicos)*
>Gonzalez Gonzalez Cristian Alan	*(Aspectos Técnicos)*

Actividades Y Subtareas

Puesto que un compilador no es más que un traductor, es decir un programa que permite pasar información de un lenguaje a otro,

1) El primer paso a realizar para la elaboración del compilador es definir qué tipo de cadenas va a leer el compilador y que tipo de salida generará.
2) Establecer cuáles serán los elementos reconocibles pero que no son tokens, tales como los comentarios de línea, separadores y controladores de posición etc.
3) Crear el diccionario de palabras reservadas propias del lenguaje como lo son por mencionar solo algunas: goto, while, if , else, include, double etc.
4) Definir los patrones de sintaxis que cada sentencia debe tener así como sus variables concordes al tipo de dato int, float, double, boolean y char.
5) Establecer los tipos de errores que pueden encontrarse al validar las cadenas de caracteres.
6) Generar la salida del compilador.

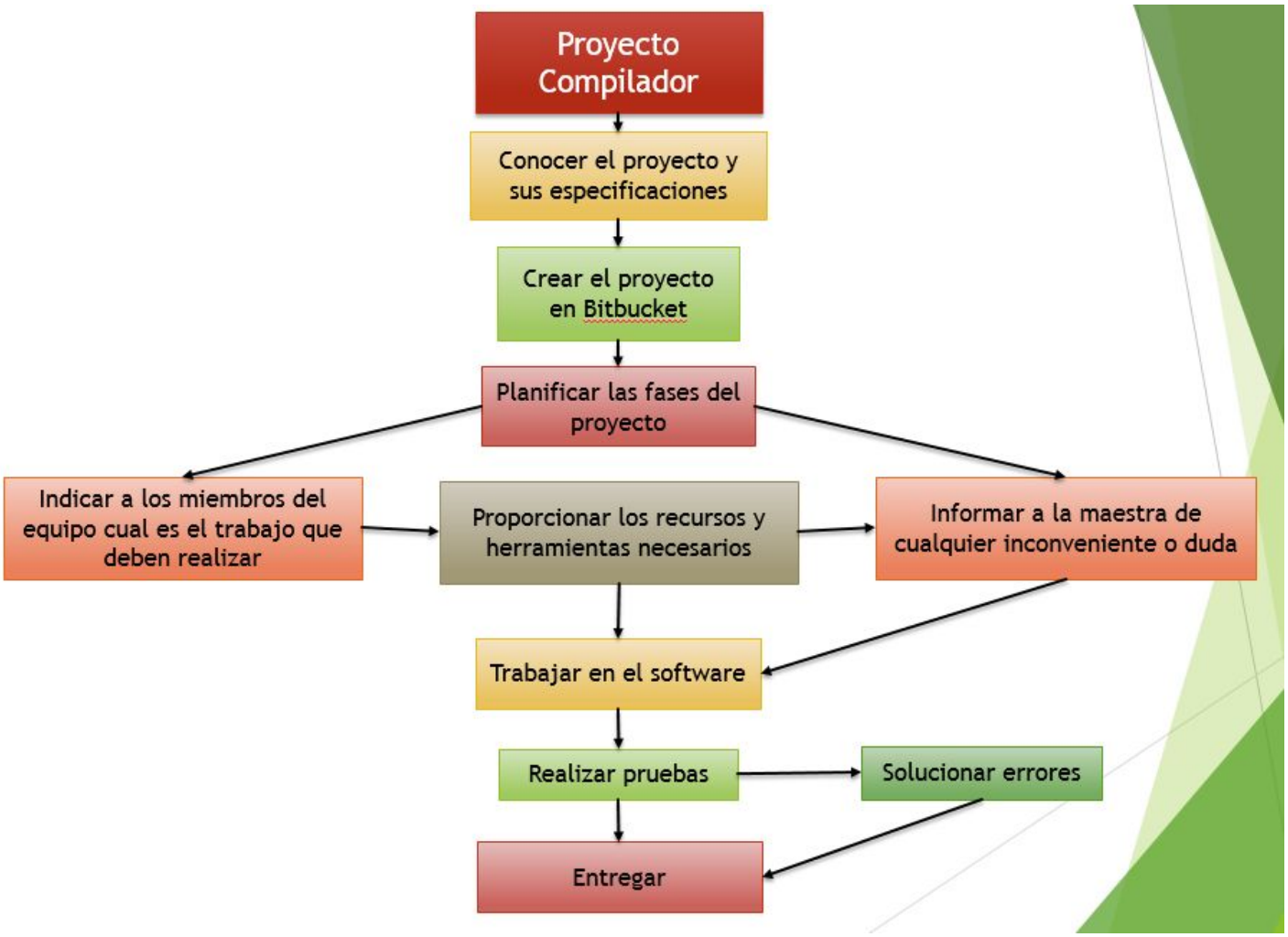
Prioridades

La prioridad es lograr concretar las actividades y especificaciones estipuladas para el proyecto, tales como definir que es un compilador, sus requisitos funcionales, los procedimientos de desarrollo, pruebas de funcionalidad, arquitectura, interfaz y finalmente la correcta ejecución y salida generada del compilador..

Fechas

Inicio Del Proyecto	22/Abril/2019
Planeación Del Proyecto	23/Abril/2019
Definir Entornos De Trabajo	24/Abril/2019
Realización Del Proyecto	26/Abril/2019
Pruebas De funcionalidad	22/Mayo/2019
Término Del Proyecto	29/Mayo/2019

Relaciones Entre Las Actividades



Recursos

>Conocimientos sobre compiladores asi como tambien sus diversas fases y etapas de desarrollo
>Ordenadores con sistema Windows instalado o en su defecto SO Linux
>Un IDE adecuado para la codificación
>Capacidad de ejecución de archivos.exe
>Capacidad mínima de programación
>Código a ejecutar

Riesgos

Los posibles riesgos o contratiempos que se pueden presentar al realizar el compilador son:

Descripción	Probabilidad	Riesgo
Al realizarlo en una plataforma digital como lo es Bitbucket, dependemos que sus servidores sigan activos y no sufran retrasos o pérdida de información.	BAJA	ALTO
Falta de conocimiento.	BAJA	ALTO
Poder computacional insuficiente debido a los recursos de procesador en las computadoras de la escuela.	MEDIO	MEDIO
Falta de entendimiento en seguir las instrucciones y requerimientos del software (Compilador)	BAJA	MEDIO
Tiempo insuficiente	ALTO	ALTO
Compromiso con el proyecto	BAJA	MEDIO

Metodología De Desarrollo De Software

La metodología de desarrollo de software que se utilizará es la llamada metodología de espiral, ya que ofrece un estructura ordenada para la elaboración del proyecto porque aprovecha la ventaja de que los proyectos de desarrollo funcionan mejor cuando son incrementales e iterativos, reflejando la relación de tareas con prototipos rápidos, mayor paralelismo y concurrencia en las actividades de diseño y construcción.

Así mismo este método se caracteriza por la planificación metódica de las tareas y entregables identificando la validez funcional de cada cambio o mejora que se realice en el software según las especificaciones de lo que se pide. **Sus fases son:**

Definir el Objetivo: Determinados conjuntamente con el equipo según los aspectos a evaluar.

Análisis y Evaluación de Riesgos: Se identifican y evalúan los riesgos potenciales, así como sus alternativas de reducción mediante prototipos o simulaciones.

Desarrollo y Prueba: El código generado es probado y migrado a un entorno de prueba varias veces hasta que el software pueda ser implementado en un entorno productivo.

Planificación Del Siguiente Ciclo: Si se producen errores, se buscan soluciones y al encontrar una alternativa viable, esta se pondrá a prueba en el siguiente ciclo para resolverlo.

B) Procedimientos De Instalación Y Prueba

Aspectos Informativos (Definición General Del Proyecto)

Idea General Del Software: Un compilador para facilitar la enseñanza de la programación
Objetivos: Proporcionar un compilador para usuarios inexpertos
Usuarios Destino: Usuarios inexpertos

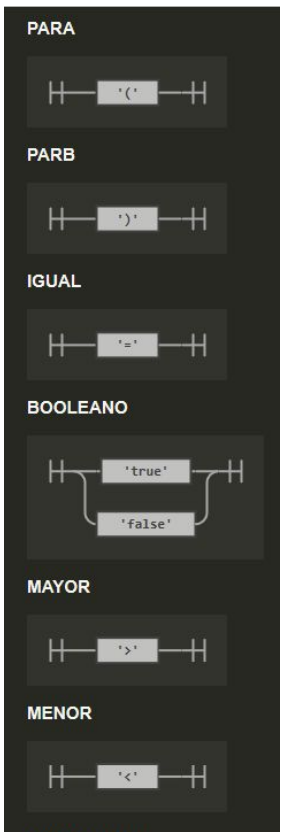
Aspectos Informativos (Detalles Técnicos)

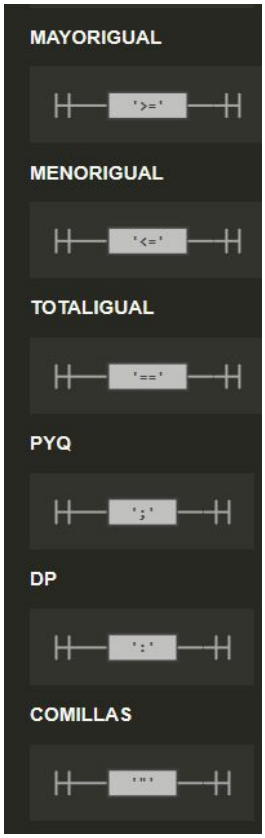
Requisitos Generales: Ordenador con Windows y JDK
Requisitos Funcionales: Ordenador con Windows
Alcance y Limitación Del Sistema: Más allá del sistema operativo no se nos podría presentar otro problema de ejecución puesto que se basa en archivos de texto

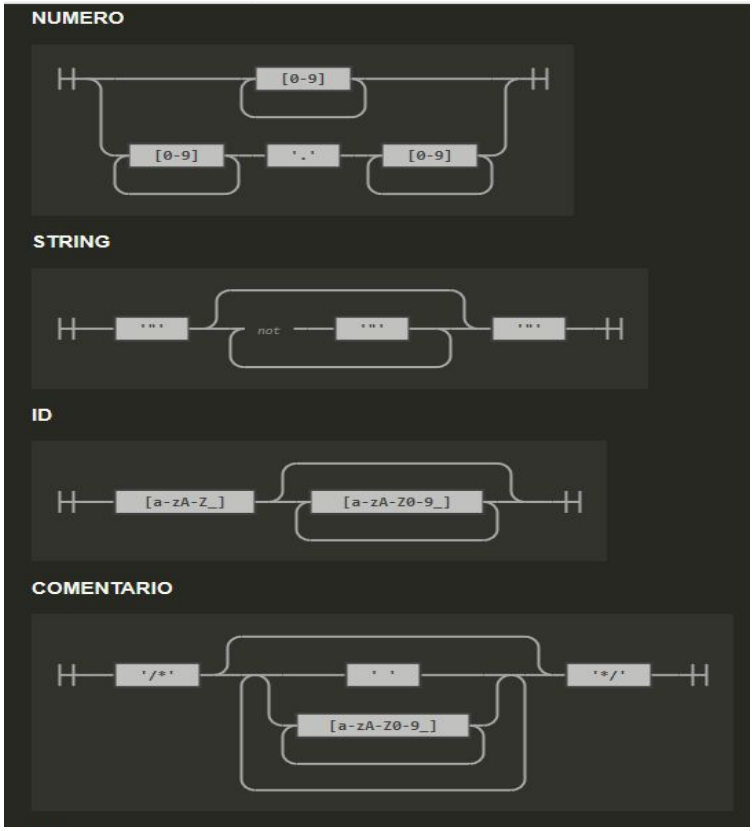
Procedimientos De Instalación …

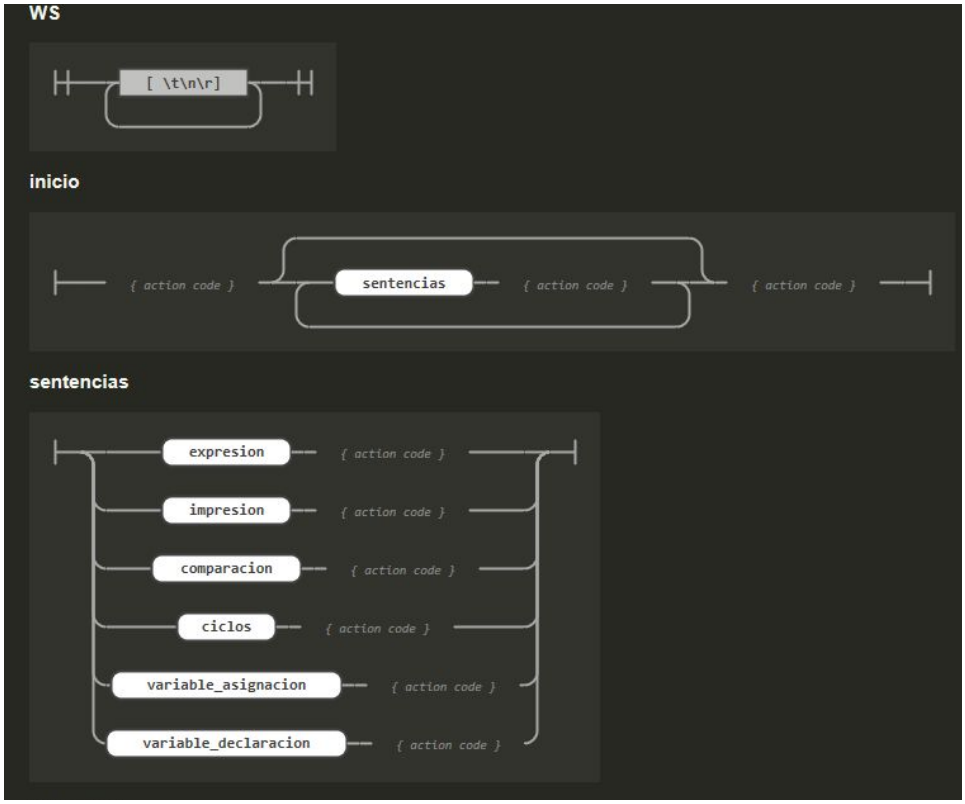
>Uso del archivo .jar proporcionado por nosotros

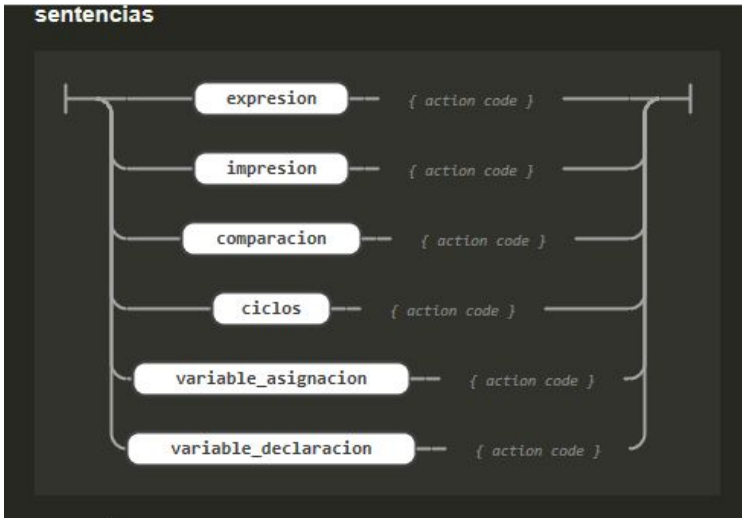
(4) Arquitectura

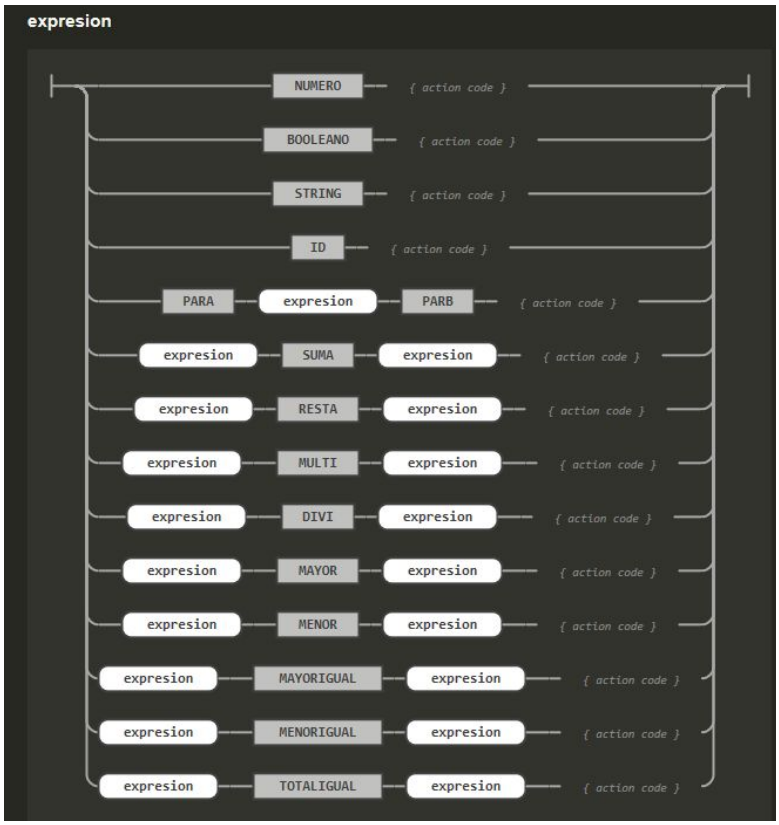


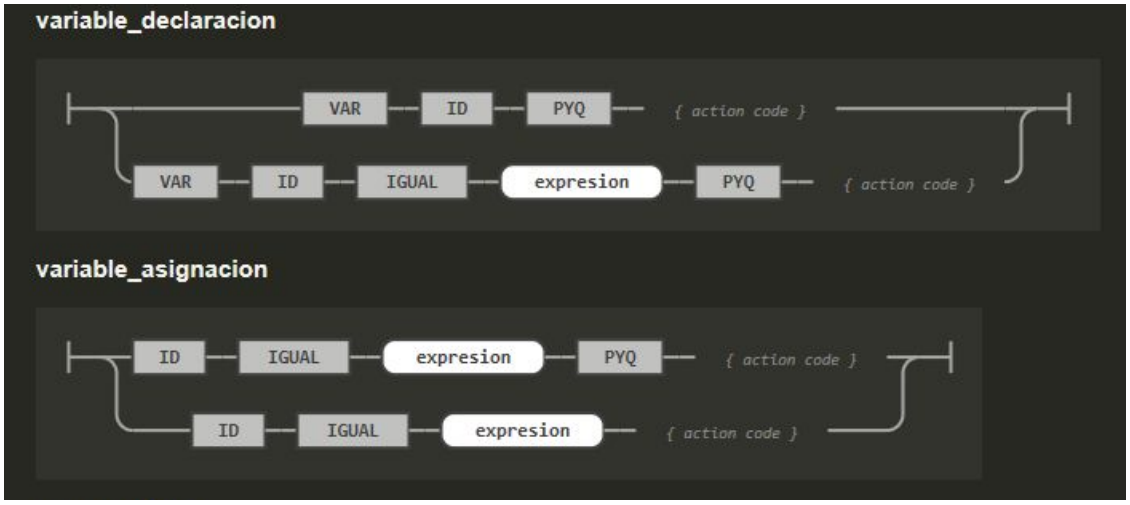




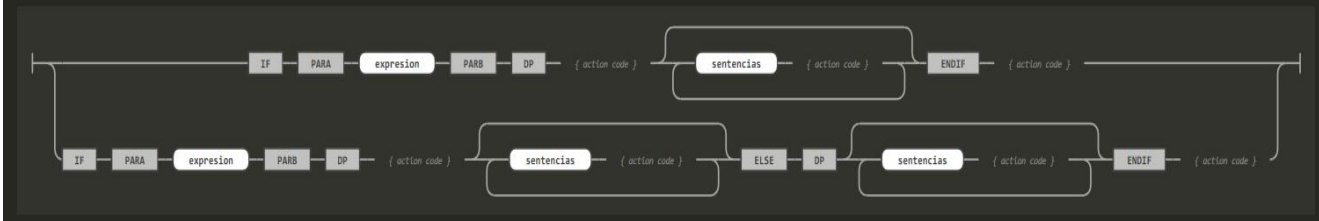



sentencias
expresion
{ action code }
impresion
{ action code }
comparacion
{ action code }
ciclos
{ action code }
variable_asignacion
{ action code }
variable_declaracion
{ action code }

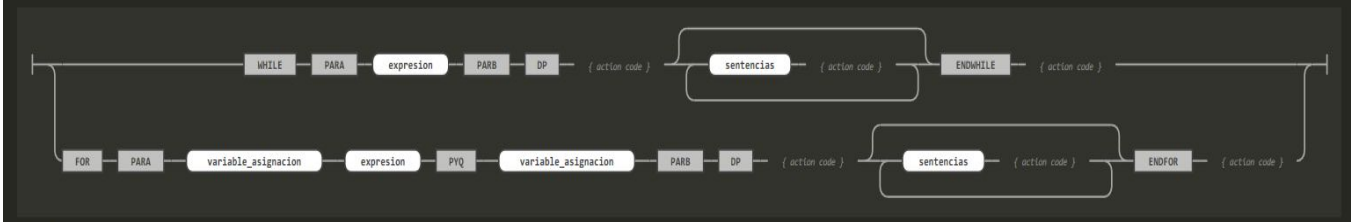

expresion
NUMERO
{ action code }
BOOLEANO
{ action code }
STRING
{ action code }
ID
{ action code }
PARA
expresion
PARB
{ action code }
expresion
SUMA
expresion
{ action code }
expresion
RESTA
expresion
{ action code }
expresion
MULTI
expresion
{ action code }
expresion
DIVI
expresion
{ action code }
expresion
MAYOR
expresion
{ action code }
expresion
MENOR
expresion
{ action code }
expresion
MAYORIGUAL
expresion
{ action code }
expresion
MENORIGUAL
expresion
{ action code }
expresion
TOTALIGUAL
expresion
{ action code }


variable_declaracion
VAR
ID
PYQ
{ action code }
VAR
ID
IGUAL
expresion
PYQ
{ action code }
variable_asignacion
ID
IGUAL
expresion
PYQ
{ action code }
ID
IGUAL
expresion
{ action code }

comparacion



ciclos



impresion

(5) Código E Interfaces De Ejecución

Main.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / src / **Main.java** Edit ...

```
import org.antlr.v4.runtime.*;

import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {

        if(args.length < 1){
            System.out.println("Fallo al abrir el documento");
            return;
        }

        astralLexer lexico = new astralLexer(CharStreams.fromFileName(args[0]));
        CommonTokenStream tokens = new CommonTokenStream(lexico);
        astralParser sintactico = new astralParser(tokens);

        sintactico.inicio();

        System.out.println("\nInterpretacion finalizada.");
    }
}

```

astral.g4

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / src / **astral.g4** Edit ...

```
grammar astral;

@parser::header{

import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import org.astral.interprete.ast.*;

}

inicio
    :
    {
        List<NodoAST> cuerpo = new ArrayList<NodoAST>();
        Map<String, Object> tablaSimbolos = new HashMap<String, Object>();

    }
      (sentencias {cuerpo.add($sentencias.nodo);})*
      {
        for(NodoAST lineas : cuerpo)
        {
            lineas.ejecutar(tablaSimbolos);
        }
      }
    ;

sentencias returns[NodoAST nodo]
    : expresion {$nodo = $expresion.nodo;}
    | impresion {$nodo = $impresion.nodo;}
```

astral.g4

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / src / **astral.g4** Edit ...

```
      | comparacion {$nodo = $comparacion.nodo;}
      | ciclos {$nodo = $ciclos.nodo;}
      | variable_asignacion {$nodo = $variable_asignacion.nodo;}
      | variable_declaracion {$nodo = $variable_declaracion.nodo;}
      ;

expresion returns[NodoAST nodo]
      : NUMERO {$nodo = new Constante( Float.parseFloat($NUMERO.text) );}
      | BOOLEANO {$nodo = new Constante(Boolean.parseBoolean($BOOLEANO.text));}
      | STRING {$nodo = new Constante($STRING.text);}
      | ID {$nodo = new VariableReferencia($ID.text);}
      | PARA expresion PARB {$nodo = $expresion.nodo;}

      | a=expresion SUMA b=expresion {$nodo = new Suma($a.nodo, $b.nodo);}
      | a=expresion RESTA b=expresion {$nodo = new Resta($a.nodo, $b.nodo);}
      | a=expresion (MULTI b=expresion) {$nodo = new Multiplicacion($a.nodo, $b.nodo);}
      | a=expresion (DIVI b=expresion) {$nodo = new Division($a.nodo, $b.nodo);}

      | a=expresion (MAYOR b=expresion) {$nodo = new Mayor($a.nodo, $b.nodo);}
      | a=expresion (MENOR b=expresion) {$nodo = new Menor($a.nodo, $b.nodo);}
      | a=expresion (MAYORIGUAL b=expresion) {$nodo = new MayorIgual($a.nodo, $b.nodo);}
      | a=expresion (MENORIGUAL b=expresion) {$nodo = new MenorIgual($a.nodo, $b.nodo);}
      | a=expresion (TOTALIGUAL b=expresion) {$nodo = new TotalIgual($a.nodo, $b.nodo);}
      ;
variable_declaracion returns[NodoAST nodo]
      : VAR ID PYQ {$nodo = new VariableDeclaracion($ID.text);}
      | VAR ID IGUAL expresion PYQ {$nodo = new VariableAsignacion($ID.text, $expresion.nodo);}
;
variable_asignacion returns[NodoAST nodo]
      : ID IGUAL expresion PYQ {$nodo = new VariableAsignacion($ID.text, $expresion.nodo);}
      | ID IGUAL expresion {$nodo = new VariableAsignacion($ID.text, $expresion.nodo);}
```


astral.g4

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / src / **astral.g4** Edit ...


```
;

comparacion returns[NodoAST nodo]
      : IF PARA expresion PARB DP {List<NodoAST> verdadero = new ArrayList<NodoAST>();}
       (sentencias {verdadero.add($sentencias.nodo);})*
       ENDIF {$nodo = new If($expresion.nodo, verdadero);}

      | IF PARA expresion PARB DP
      {List<NodoAST> verdadero = new ArrayList<NodoAST>(); List<NodoAST> falso = new ArrayList<NodoAST>();}
       (v=sentencias {verdadero.add($v.nodo);})*
       ELSE DP
       (f=sentencias {falso.add($f.nodo);})*
       ENDIF {$nodo = new IfElse($expresion.nodo, verdadero, falso);}

      ;

ciclos returns[NodoAST nodo]
      : WHILE PARA expresion PARB DP
          {List<NodoAST> sublineas = new ArrayList<NodoAST>();}
      (sentencias {sublineas.add($sentencias.nodo);})*
      ENDWHILE {$nodo = new While($expresion.nodo, sublineas);}

      | FOR PARA (a=variable_asignacion) expresion PYQ (b=variable_asignacion) PARB DP
          {List<NodoAST> sublineasFor = new ArrayList<NodoAST>();}
       (sentencias {sublineasFor.add($sentencias.nodo);})*
       ENDFOR {$nodo = new For($a.nodo, $expresion.nodo, $b.nodo, sublineasFor);}
;

impresion returns[NodoAST nodo]: PRINT PARA expresion PARB PYQ {$nodo = new Print($expresion.nodo);};
```

astral.g4

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / src / **astral.g4** Edit ...

```
93   PRINT: 'print';
94   VAR: 'var';
95   IF: 'if';
96   ELSE: 'else';
97   ENDIF:'endif';
98   WHILE: 'while';
99   ENDWHILE: 'endwhile';
100  FOR: 'for';
101  ENDFOR:'endfor';
102
103  SUMA: '+';
104  RESTA: '-';
105  MULTI: '*';
106  DIVI: '/';
107  PARA: '(';
108  PARB: ')';
109  IGUAL: '=';
110
111  BOOLEANO: 'true' | 'false';
112
113  MAYOR: '>';
114  MENOR: '<';
115  MAYORIGUAL: '>=';
116  MENORIGUAL: '<=';
117  TOTALIGUAL: '==';
118
119  PYQ: ';';
120  DP: ':';
121  COMILLAS:'"';
122
123  NUMERO: [0-9]+ | [0-9]+'.'[0-9]+;
124  STRING: '"' ~('"')* '"';
125  ID: [a-zA-Z_][a-zA-Z0-9_]*;
126
127  COMENTARIO:'/*'(' ' | [a-zA-Z0-9_]+)*'*/' -> skip;
128  WS : [ \t\n\r]+ -> skip;
```

Constante.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Constante.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Constante implements NodoAST {
    private Object numero;

    public Constante(Object numero) {
        this.numero = numero;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return numero;
    }
}

```

Division.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Division.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Division implements NodoAST {

    private NodoAST numero_1;
    private NodoAST numero_2;

    public Division(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) / (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

For.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **For.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.List;
import java.util.Map;

public class For implements NodoAST {
    private NodoAST variable;
    private NodoAST comparacion;
    private NodoAST aumento;
    private List<NodoAST> subCuerpo;

    public For(NodoAST variable, NodoAST comparacion, NodoAST aumento, List<NodoAST> subCuerpo) {
        this.variable = variable;
        this.comparacion = comparacion;
        this.aumento = aumento;
        this.subCuerpo = subCuerpo;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        for(variable.ejecutar(tablaSimbolos); (boolean)comparacion.ejecutar(tablaSimbolos); aumento.ejecutar(tablaSimbolos))
        {
            for(NodoAST lineas : subCuerpo)
            {
                lineas.ejecutar(tablaSimbolos);
            }
        }
        return null;
    }
}

```

If.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / If.java Edit ...

```java
package org.astral.interprete.ast;

import java.util.List;
import java.util.Map;

public class If implements NodoAST {
    private NodoAST condicion;
    private List<NodoAST> verdadero;

    public If(NodoAST condicion, List<NodoAST> verdadero) {
        this.condicion = condicion;
        this.verdadero = verdadero;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        if((boolean)condicion.ejecutar(tablaSimbolos))
        {
            for(NodoAST lineas : verdadero)
            {
                lineas.ejecutar(tablaSimbolos);
            }
        }
        return null;
    }
}

```

IfElse.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / IfElse.java Edit ...

```java
package org.astral.interprete.ast;

import java.util.List;
import java.util.Map;

public class IfElse implements NodoAST {
    private NodoAST condicion;
    private List<NodoAST> verdadero;
    private List<NodoAST> falso;

    public IfElse(NodoAST condicion, List<NodoAST> verdadero, List<NodoAST> falso) {
        this.condicion = condicion;
        this.verdadero = verdadero;
        this.falso = falso;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        if((boolean)condicion.ejecutar(tablaSimbolos))
        {
            for(NodoAST lineas : verdadero)
            {
                lineas.ejecutar(tablaSimbolos);
            }
        }
        else
        {
            for(NodoAST lineas : falso)
            {
                lineas.ejecutar(tablaSimbolos);
            }
        }
        return null;
    }
}

```

Mayor.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Mayor.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Mayor implements NodoAST {
    private NodoAST numero_1;
    private NodoAST numero_2;

    public Mayor(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) > (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

MayorIgual.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **MayorIgual.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class MayorIgual implements NodoAST {
    private NodoAST numero_1;
    private NodoAST numero_2;

    public MayorIgual(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) >= (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

Menor.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Menor.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Menor implements NodoAST {
    private NodoAST numero_1;
    private NodoAST numero_2;

    public Menor(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) < (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

MenorIgual.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **MenorIgual.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class MenorIgual implements NodoAST {
    private NodoAST numero_1;
    private NodoAST numero_2;

    public MenorIgual(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) <= (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

Multiplicacion.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Multiplicacion.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Multiplicacion implements NodoAST {

    private NodoAST numero_1;
    private NodoAST numero_2;

    public Multiplicacion(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) * (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

NodoAST.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **NodoAST.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public interface NodoAST {
    public Object ejecutar(Map<String, Object> tablaSimbolos);
}

```

Print.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Print.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Print implements NodoAST {
    private NodoAST datos;

    public Print(NodoAST datos) {
        this.datos = datos;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        System.out.println(datos.ejecutar(tablaSimbolos));
        return null;
    }
}

```

Resta.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Resta.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Resta implements NodoAST {

    private NodoAST numero_1;
    private NodoAST numero_2;

    public Resta(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) - (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

Suma.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **Suma.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class Suma implements NodoAST {

    private NodoAST numero_1;
    private NodoAST numero_2;

    public Suma(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) + (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

TotalIgual.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **TotalIgual.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class TotalIgual implements NodoAST {
    private NodoAST numero_1;
    private NodoAST numero_2;

    public TotalIgual(NodoAST numero_1, NodoAST numero_2) {
        this.numero_1 = numero_1;
        this.numero_2 = numero_2;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return (float)numero_1.ejecutar(tablaSimbolos) == (float)numero_2.ejecutar(tablaSimbolos);
    }
}

```

VariableAsignacion.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **VariableAsignacion.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class VariableAsignacion implements NodoAST {
    private String nombre;
    private NodoAST expresion;

    public VariableAsignacion(String nombre, NodoAST expresion) {
        this.nombre = nombre;
        this.expresion = expresion;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        tablaSimbolos.put(this.nombre,expresion.ejecutar(tablaSimbolos));
        return null;
    }
}

```

VariableDeclaracion.java

Pull requests Check out ...

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **VariableDeclaracion.java** Edit ...

```java
package org.astral.interprete.ast;

import java.util.Map;

public class VariableDeclaracion implements NodoAST {
    private String nombre;

    public VariableDeclaracion(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        tablaSimbolos.put(this.nombre, new Object() );
        return null;
    }
}

```

VariableReferencia.java

Pull requests Check out •••

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **VariableReferencia.java** Edit •••

```java
package org.astral.interprete.ast;

import java.util.Map;

public class VariableReferencia implements NodoAST {
    private String nombre;

    public VariableReferencia(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        return tablaSimbolos.get(nombre);
    }
}

```

While.java

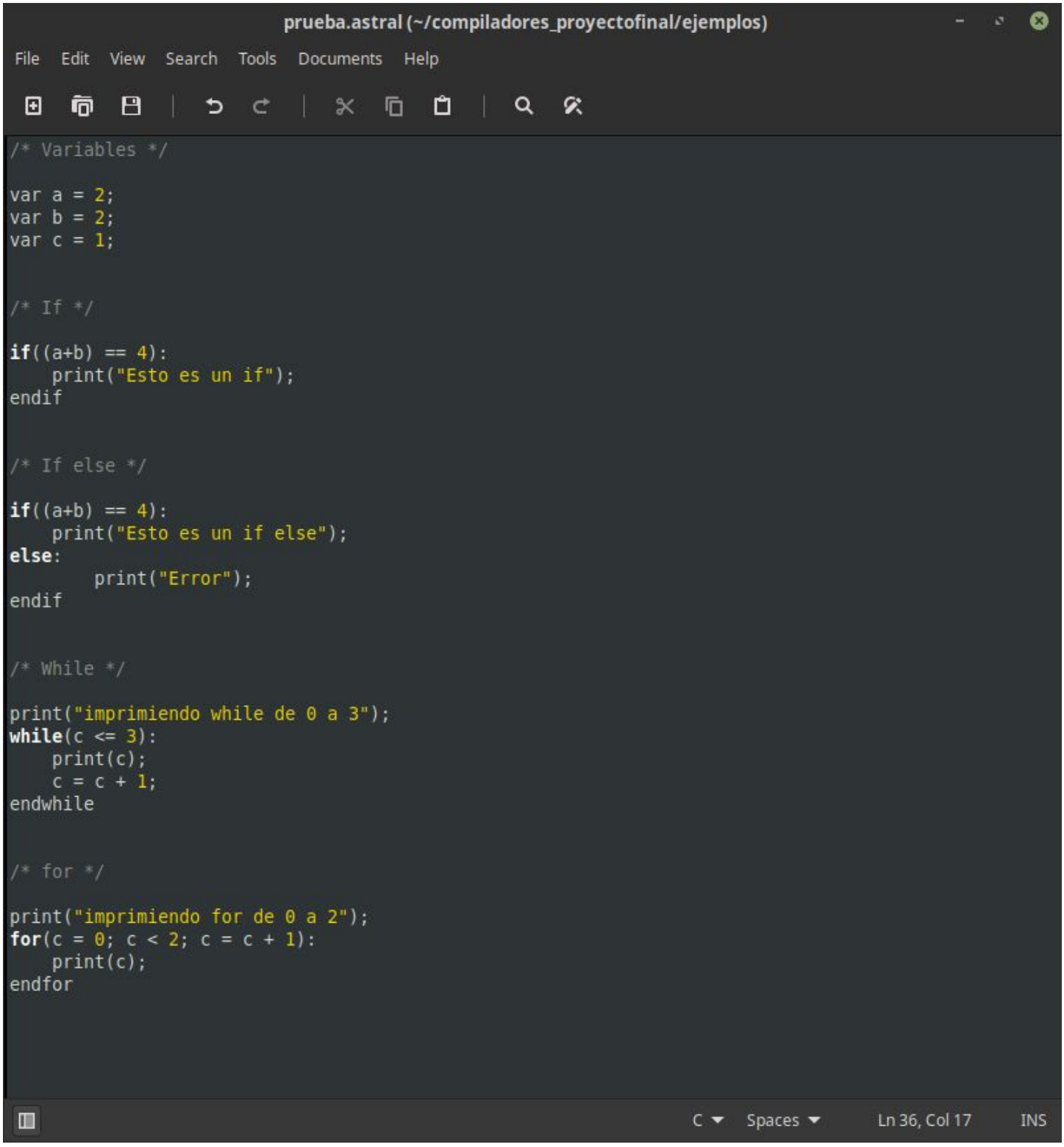
Pull requests Check out •••

Source master 9e2ac5e Full commit

compiladores_proyectofinal / AST / **While.java** Edit •••

```java
package org.astral.interprete.ast;

import java.util.List;
import java.util.Map;

public class While implements NodoAST {
    private NodoAST condicion;
    private List<NodoAST> subCuerpo;

    public While(NodoAST condicion, List<NodoAST> subCuerpo) {
        this.condicion = condicion;
        this.subCuerpo = subCuerpo;
    }

    @Override
    public Object ejecutar(Map<String, Object> tablaSimbolos) {
        while((boolean)condicion.ejecutar(tablaSimbolos))
        {
            for(NodoAST lineas : subCuerpo)
            {
                lineas.ejecutar(tablaSimbolos);
            }
        }
        return null;
    }
}

```

Prueba


prueba.astral (~/compiladores_proyectofinal/ejemplos)
File Edit View Search Tools Documents Help
/* Variables */
var a = 2;
var b = 2;
var c = 1;
/* If */
if((a+b) == 4):
print("Esto es un if");
endif
/* If else */
if((a+b) == 4):
print("Esto es un if else");
else:
print("Error");
endif
/* While */
print("imprimiendo while de 0 a 3");
while(c <= 3):
print(c);
c = c + 1;
endwhile
/* for */
print("imprimiendo for de 0 a 2");
for(c = 0; c < 2; c = c + 1):
print(c);
endfor
C
Spaces
Ln 36, Col 17
INS

Ejecución

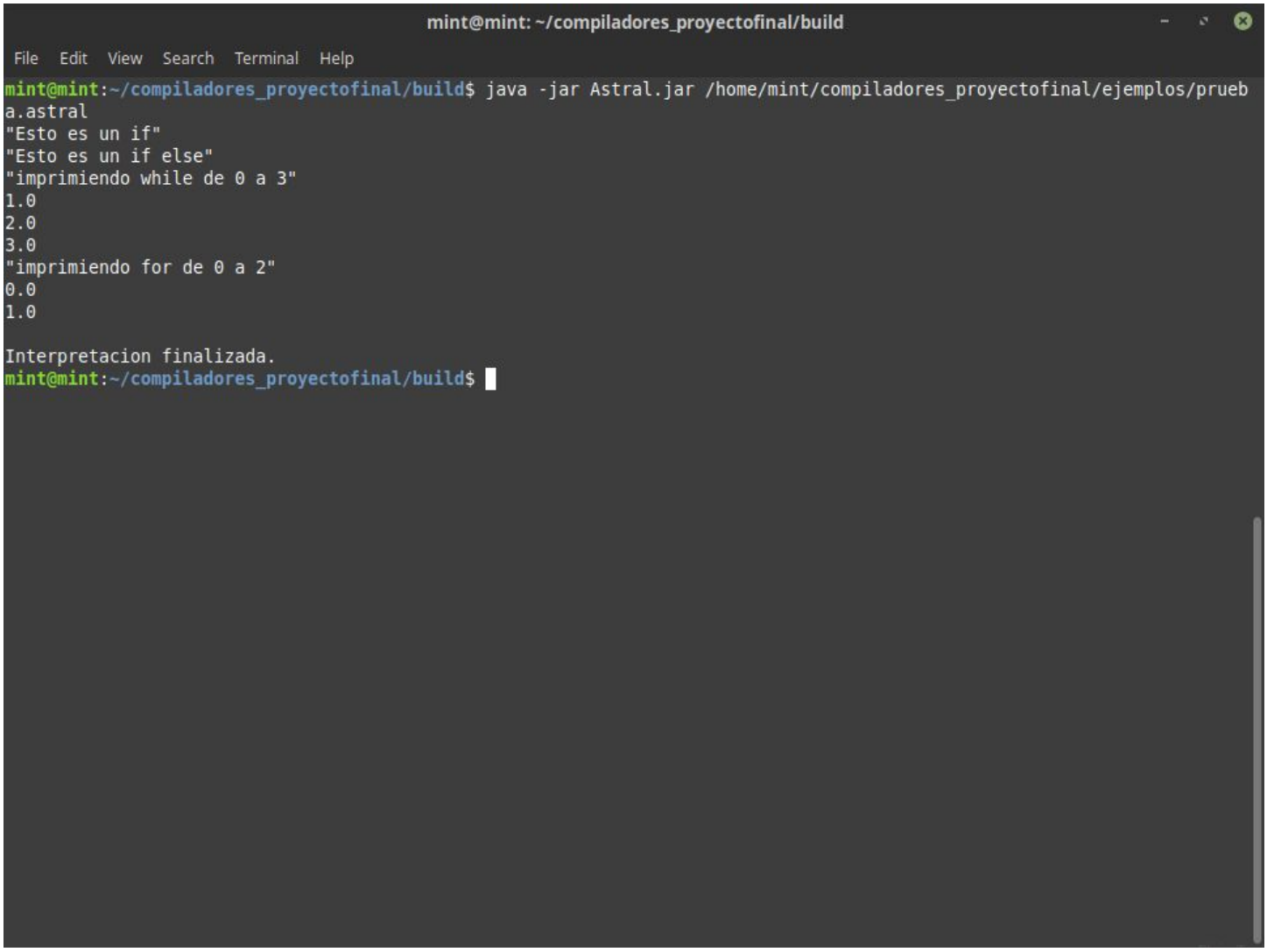

mint@mint: ~/compiladores_proyectofinal/build
File Edit View Search Terminal Help
mint@mint:~/compiladores_proyectofinal/build$ java -jar Astral.jar /home/mint/compiladores_proyectofinal/ejemplos/prueba.astral
"Esto es un if"
"Esto es un if else"
"imprimiendo while de 0 a 3"
1.0
2.0
3.0
"imprimiendo for de 0 a 2"
0.0
1.0
Interpretacion finalizada.
mint@mint:~/compiladores_proyectofinal/build$

Tabla de símbolos

Token	Lexema	Descripción
Palabra reservada	int	Declaración de un número entero
Palabra reservada	if	Comparador
Palabra reservada	else	sí no
Palabra reservada	endif	Fin del comparador
Palabra reservada	printf	Impresión en pantalla
Palabra reservada	while	Ciclo
Palabra reservada	for	Ciclo
Número	[0-9]+	Número entero
ID	[a-zA-z][a-zA-Z0-9]*	Variable o identificador
Operador	(	Paréntesis Abierto
Operador	)	Paréntesis Cerrado
Operador	[	Corchete Abierto
Operador	]	Corchete Cerrado
Operador	;	Punto y coma
Operador	:	Dos puntos
Operador de asignación	=	igual
Operador aritmético	+	Más
Operador aritmético	-	Menos
Operador aritmético	/	Entre
Operador aritmético	*	Por
Operador Relacional	<	Menor que

Operador Relacional	>	Mayor que
Operador Relacional	<=	Menor o igual que
Operador Relacional	>=	Mayor o igual que
Operador Relacional	==	Totalmente igual