**Little-Duck: entrega #2**

**Roberto Calleja A01024834**

**Scanner y Parser**

Para el Scanner y el Parser se selecciono la herramienta Antlr, en conjunto con Python. La gramática se transcribió de la entrega pasada a un archivo llamado “little\_duck.g4”. Al ejecutar este código, junto con la herramienta Antlr se generaron 7 archivos que en conjunto actúan como Scanner y Parser y estos archivos son llamados por el programa “Driver.py” para compilar los diferentes códigos.

La entrega cuenta con todos los códigos previamente mencionados, junto con una carpeta de test la cual cuenta con 7 archivos de tipo “.txt” los cuales contienen las diferentes pruebas que se realizaron. La información de como ejecutar los archivos se encuentra en el archivo README.txt.

**Tabla de consideraciones semánticas (Cubo semántico)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operando 1 | Operando 2 | Operadpr | | | | | | | |
| + | - | \* | / | > | < | != | = |
| int | int | int | int | int | float | bool | bool | bool | int |
| int | float | float | float | float | float | bool | bool | bool | error |
| int | string | error | error | error | float | error | error | error | error |
| float | int | float | float | float | float | bool | bool | bool | error |
| float | float | float | float | float | float | bool | bool | bool | float |
| float | string | error | error | error | float | error | error | error | error |
| string | int | error | error | error | float | error | error | error | error |
| string | float | error | error | error | float | error | error | error | error |
| string | string | error | error | error | float | error | error | error | String |

Para el lenguaje se implementó este cubo semántico, el cual cuenta con las relaciones entre los tipos de variable, int, float y string. Con esta tabla se implementó una función “check\_cubo\_semantico” la cual recibe los dos operandos y el operador, y da como resultado el tipo de valor que se espera de esa operación o en caso de operadores no compatibles se levanta una excepción.

**Estructuras de datos (Directorio de Funciones y a las Tablas de Variables)**

|  |
| --- |
| **Variable** |
| + name: str  +type: str  +value: str |
|  |

|  |
| --- |
| **VarTable** |
| +variables: {Variable} |
|  |

|  |
| --- |
| **Function** |
| + name: str + var\_table: VarTable |
|  |

|  |
| --- |
| FunctionDirectory |
| + functions: {Function}  + global\_var\_table: VarTable |
| -add\_function(name):void  - add\_variable(variable, function\_name ):void  - test\_print():void |

1. Crear todas las variables y arreglos.

2. Agregar variables a current\_var\_arr

3. Leer Type y guardarlo en current\_typr

3.1Agregar variables a función\_directory

3.2 Limpiar arreglo current\_var\_arr

4. Crea Función

4.1Actualiza current\_function

5. Actualiza current\_function a global

6. Elimina todas las tablas e imprime la tabla

**Puntos neurálgicos**

A black and white diagram with text

Description automatically generated A computer code with text and symbols

Description automatically generated with medium confidence A graph paper with text and symbols

Description automatically generated A computer screen shot of a computer code

Description automatically generated

Estos puntos neurálgicos fueron implementados en el archivo little\_duckListener.py, en las funciones de relacionadas a la gramática correspondiente, y cada punto esta marcado con las iniciales DFTV2-PN (Directorio de funciones y tablas de variables entrega 2 Puntos Neuralgicos) acompañado del numero de punto correspondiente

**Expresiones Regulares**

INT: [0-9]+

FLOAT: [0-9]+ ‘.’ [0-9]+

STRING: “[^”]\*”

ID: [a-zA-Z0-9]\*

**Lista de tokens**

|  |  |  |
| --- | --- | --- |
| Main | Program | While |
| End | Id | Do |
| Var | Cte.String | If |
| Print | Cte.Int | Else |
| void | Cte.float | int |
| float | [ | ] |
| ( | ) | { |
| } | ; | : |
| , | + | - |
| \* | / | = |
| < | > | != |

**Gramática libre de contexto**

A diagram of a system

Description automatically generated

<programa> -> program id ; <vars> <funcs> main <bodyend

A diagram of a diagram

Description automatically generated

<vars> -> var < var0>

<var0> -> <var1>: <type>; <var2>

<var1> -> id

<var1> -> id, <var1>

<var2> -> E

<var2> <var1>

A diagram of a type

Description automatically generated

<type> - float

<type> -> int

A black rectangle with black text

Description automatically generated

<body> -> { <body0>

<body0> -> }

<body0> -> <statement> < body1> }

<body1> -> < statement> <body1>

<body1> -> E

A diagram of a statement

Description automatically generated

<statement> -> <assign>

<statement> -> <condition>

<statement> -> < cycle>

<statement> -> <F\_Call>

<statement> -> <print>

A diagram of a expression

Description automatically generated

<print> -> print ( print0);

<print0> -> <expresion> <print1>

<print0> -> cte.string <print1>

<print1> -> E

<print1> -> , <print0>

A diagram of a computer flowchart

Description automatically generated

<cycle> -> while <body> do ( <expression>);

A diagram of a diagram

Description automatically generated

<condition> -> if (<expresión>) <body> <condition0>

<condition0> -> E

< condition0> -> else <body>

A diagram of a problem

Description automatically generated

<expresión> -> <exp> < expresión0>

< expresión0> -> E

< expresión0> -> < expresión1> <exp>

< expresión1> -> >

< expresión1>-> <

< expresión1> -> ¡=

A diagram of a machine

Description automatically generated

< exp> -> <termino> <exp0>

<exp0> -> E

<exp0> -> <exp1> <exp>

<exp1> -> -

<exp1> -> +

A diagram of a flowchart

Description automatically generated

<cte> -> cte\_float

<cte> -> cte\_int

A diagram of a factor

Description automatically generated

<termino> -> < factor> < termino0>

< termino0> -> E

< termino0> -> < termino1> < termino>

< termino1>-> \*

< termino1>-> /

A diagram of a computer

Description automatically generated

<factor> -> ( < expresión> )

<factor> -> <factor0> <factor1>

<factor0> -> +

<factor0>-> -

<factor0>-> E

<factor1>-> id

<factor1>-> <cte> A diagram of a type

Description automatically generated<Funcs> -> void id (<Funcs0>) [<Func3 > < body> ];

<Funcs0> -> E

<Funcs0>-> <Funcs1>

<Funcs1>-> id : <type> < Funcs2>

<Funcs2>-> E

<Funcs2>-> , <Funcs1>

<Funcs3> -> <vars>

<Funcs3> -> E

A diagram of a person's expression

Description automatically generated

<f\_call> -> id ( < f\_call0>) ;

< f\_call0> -> E

< f\_call0> -> < f\_call1>

<f\_call1> -> <expression> < f\_call 2>

< f\_call2> -> E

<f\_call2> -> , < f\_call2>

A black and white diagram with a white rectangle and a black rectangle with a black text

Description automatically generated with medium confidence

<assign> -> id = <Expresion> ;