Nombre: Dana Carolina Ramírez Velázquez

Código: 220286547

Materia: Traductores de Lenguaje 2

Actividad: Parte 1 Proyecto

Fecha: 23/11/22

Introducción:

Las indicaciones para elaborar la segunda parte del proyecto eran ¡Reutilizar la parte uno y generar el código de 3 direcciones de salida correspondiente.

Desarrollo:

En esta ultima parte del proyecto, realicé un GDA con nodos y hojas del cual obtuve las instrucciones de tres direcciones.

La realización del arbol fue igual que en la práctica 5 adecuandolo a la gramática nueva, y para la generación de código de tres direcciones el código fue el siguiente.

```
59  string AnalizadorSintactico::TresDirecciones(Nodo *root) {
60    string temp;
61    string temporal1;
62    string temporal2;
63    cout.flush();
64    if (root->left != NULL && !condition) {
65      temporal1 = TresDirecciones(root->left);
66    }
67    condition = false;
68    if (Tokens.List[root->dato].tipo == TokenType::CONDICION) {
69      label++;
70      TipoOrden(root, temporal1, temporal2);
71      Condicional nuevo;
72      nuevo.label = "LABEL" + to_string(label);
73      nuevo.inicio = root->right->left;
74      condicionales.push_back(nuevo);
75      condition = true;
76    }
77    if (Tokens.List[root->dato].tipo == TokenType::BUCLE) {
78      label++;
79      cout << "LABEL" << label << endl;
80    }
81    if (root->right != NULL) {
82      temporal2 = TresDirecciones(root->right);
83    }
84    if (root->left == NULL && root->right == NULL) {
85      if (Tokens.List[root->dato].tipo == TokenType::NUMERO) {
86        index++;
87        TipoOrden(root, temporal1, temporal2);
88        temp = "temp" + to_string(index);
89      } else if (Tokens.List[root->dato].tipo == TokenType::CONDICION_ELSE) {
90        TipoOrden(root, temporal1, temporal2);
91      } else if (temporal != "") {
92        cout << temp << ": " << temporal << endl;
93        temporal.clear();
```

```
94      } else {
95        temp = Tokens.List[root->dato].value;
96      }
97    } else if (root->left != NULL && root->right == NULL) {
98      if (Tokens.List[root->dato].tipo == TokenType::OPERADOR_ARITMETICO) {
99        temporal =
100           Tokens.List[root->dato].value + " " + "temp" + to_string(index);
101     } else {
102       if (Tokens.List[root->dato].tipo == TokenType::NUMERO) {
103         index++;
104         TipoOrden(root, temporal1, temporal2);
105         cout << "temp" << index << ": "
106              << "temp" << index << " " << temporal << endl;
107       } else if (Tokens.List[root->dato].tipo != TokenType::PARENTESIS) {
108         cout << "temp" << index << ": " << Tokens.List[root->dato].value << " "
109              << temporal << endl;
110       }
111       temporal.clear();
112     }
113     temp = "temp" + to_string(index);
114   } else if (root->left == NULL && root->right != NULL) {
115     cout << Tokens.List[root->dato].value << endl;
116     temp = "temp" + to_string(index);
117   } else if (Tokens.List[root->dato].tipo != TokenType::CONDICION &&
118              Tokens.List[root->dato].tipo != TokenType::PUNTO_Y_COMA &&
119              Tokens.List[root->dato].tipo != TokenType::CONDICION_ELSE) {
120     TipoOrden(root, temporal1, temporal2);
121     temp = "temp" + to_string(index);
122   }
123   return temp;
124 }
```

Este es el manejo de los nodos y el recorrido recursivo que hace la impresión de las instrucciones, donde clasifica que tipo de nodo es el que estamos por imprimir.

Esta funcion se va de izquierda a derecha por todos los nodos empezando en el lado de las ordenes del programa.

También aquí mismo se genera una lista de LABEL que sirven para el entendimiento de los bucles y los condicionales.

```
26  void AnalizadorSintactico::TipoOrden(Nodo *root, string t_left,
27                                       string t_right) {
28    if (Tokens.List[root->dato].tipo == TokenType::CONDICION ||
29        Tokens.List[root->dato].tipo == TokenType::BUCLE) {
30      cout << "IF " << t_left
31           << " GOTO "
32              "LABEL"
33           << label << endl;
34    } else if (Tokens.List[root->dato].tipo == TokenType::CONDICION_ELSE) {
35      cout << "GOTO SIG" << sig << endl << endl;
36    } else if (Tokens.List[root->dato].tipo == TokenType::BUCLE_END) {
37      label++;
38      cout << "LABEL" << label << endl << endl;
39    } else if (Tokens.List[root->dato].tipo == TokenType::PARENTESIS_END) {
40      sig++;
41      cout << "SIG" << sig << endl;
42    } else if (Tokens.List[root->dato].tipo == TokenType::NUMERO) {
43      cout << "temp" << index << ": " << Tokens.List[root->dato].value << endl;
44    } else if (Tokens.List[root->dato].tipo == TokenType::IDENTIFICADOR) {
45      cout << Tokens.List[root->dato].value << endl;
46    } else if (Tokens.List[root->dato].tipo == TokenType::OPERADOR_BOOL) {
47      index++;
48      cout << "temp" << index << ": " << t_left << " "
49           << Tokens.List[root->dato].value << " " << t_right << endl;
50    } else if (Tokens.List[root->dato].tipo == TokenType::PARENTESIS) {
51      cout << t_left << ": " << t_left << " "
52           << Tokens.List[root->right->dato].value << " " << t_right << endl;
53    } else {
54      cout << t_left << ": " << t_left << " " << Tokens.List[root->dato].value
55           << " " << t_right << endl;
56    }
57  }
```

Esta función imprime las intrucciones dependiendo el tipo de instrucción que se identificó, así como ls indicadores de GOTO, los LABEL y SIG para el entendimiento del código de tres direcciones cuando tenemos algun bucle o condicional.

Resultados:

Intendando con la siguiente entrada:

AnalizadorLexico.cpp | AnalizadorLexico.h | Ana

prueba.txt

You, 2 hours ago | 1 author (You)

```
1  begin
2  entero var1,var2;
3  real var3;
4  var1 := 3;
5  var2 := 4 +(3-2);
6  if(var1>var2)
7  var3 := var1 + (21 - 30) + 9;
8  var2 := 5;        You, 5 days ago • Funciona a
9  else
10 var2 := 1;
11 end;
12 var1 := 23;
13 while(var1<var2)
14 var3 := var1;
15 var2 := 9;
16 var1 := 4+2+23;
17 var2 := var3;
18 endwhile;
19 var1 := 777 - 999;
20 end       You, 5 days ago • Funciona a medias
```

El resultado es el siguiente:

```
temp1: 3
var1: var1 := temp1
temp2: 2
temp3: 3
temp3: temp3 - temp2
temp4: 4
temp4: temp4 + temp3
var2: var2 := temp4
temp5: var1 > var2
IF temp5 GOTO LABEL1
temp6: 1
var2: var2 := temp6
SIG1
temp7: 23
var1: var1 := temp7
temp8: var1 < var2
LABEL2
var3: var3 := var1
temp9: 9
var2: var2 := temp9
temp10: 23
temp11: 2
temp11: temp11 + temp10
temp12: 4
temp12: temp12 + temp11
var1: var1 := temp12
var2: var2 := var3
IF temp8 GOTO LABEL2
temp13: 999
temp14: 777
temp14: temp14 - temp13
var1: var1 := temp14

LABEL1
temp15: 30
temp16: 21
temp16: temp16 - temp15
temp17: 9
temp16: temp16 + temp17
temp17: var1 + temp17
var3: var3 := temp17
temp18: 5
var2: var2 := temp18
GOTO SIG1

SINTAXIS CORRECTA!!!
[1] + Done                       "/us
```

prueba.txt

You, 57 seconds ago | 1 author (You)

```
1  begin
2  entero var1,var2;
3  real var3;
4  var1 := 3;
5  var2 := 4 +(3-2);
6  while(var1<var2)
7  var3 := var1;
8  var1 := 4+2+23;
9  endwhile;
10 var1 := 777 - 999;
11 if(var1>var2)
12 var3 := var1 + (21 - 30) + 9;
13 var2 := 5;
14 else
15 var2 := 1;
16 var1 := 123;
17 end;
18 end      You, 5 days ago • Funciona a
```

```
temp1: 3
var1: var1 := temp1
temp2: 2
temp3: 3
temp3: temp3 - temp2
temp4: 4
temp4: temp4 + temp3
var2: var2 := temp4
temp5: var1 < var2
LABEL1
var3: var3 := var1
temp6: 23
temp7: 2
temp7: temp7 + temp6
temp8: 4
temp8: temp8 + temp7
var1: var1 := temp8
IF temp5 GOTO LABEL1
temp9: 999
temp10: 777
temp10: temp10 - temp9
var1: var1 := temp10
temp11: var1 > var2
IF temp11 GOTO LABEL2
temp12: 1
var2: var2 := temp12
temp13: 123
var1: var1 := temp13
SIG1

LABEL2
temp14: 30
temp15: 21
temp15: temp15 - temp14
temp16: 9
temp15: temp15 + temp16
temp16: var1 + temp16
var3: var3 := temp16
temp17: 5
var2: var2 := temp17
GOTO SIG1

SINTAXIS CORRECTA!!!
[1] + Done                    "/
```

Entrada Erronea:

Mensaje resultado:

prueba.txt

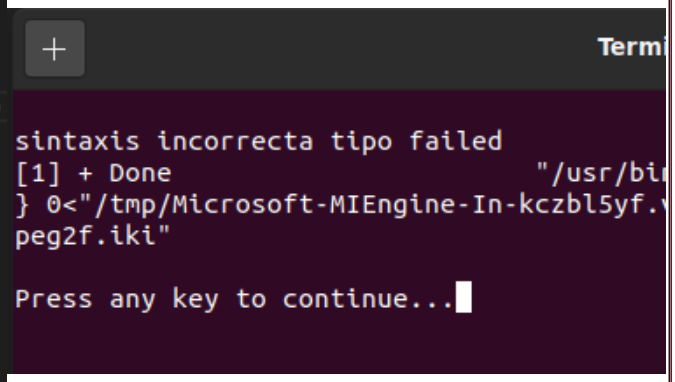
You, 3 minutes ago | 1 author (You)

```
1  begin
2  entero var1,var2;
3  real var3;
4  var1 := cuatro;
5  var2 := 4 +(3-2);
6  while(var1<var2)
7  var3 := var1;
8  var1 := 4+2+23;
9  endwhile;
10 var1 := 777 - 999;
11 if(var1>var2)
12 var3 := var1 + (21 - 30) + 9;
13 var2 := 5;
14 else
15 var2 := 1;
16 var1 := 123;
17 end;
18 end      You, 5 days ago • Funciona
```

prueba.txt

You, 3 minutes ago | 1 author (You)

```
1  begin
2  entero var1,var2;
3  decimal var3;
4  var1 := 3;
5  var2 := 4 +(3-2);      You, 4 hours ago
6  while(var1<var2)
7  var3 := var1;
8  var1 := 4+2+23;
9  endwhile;
10 var1 := 777 - 999;
11 if(var1>var2)
12 var3 := var1 + (21 - 30) + 9;
13 var2 := 5;
14 else
15 var2 := 1;
16 var1 := 123;
17 end;
18 end
```



Conclusiones:

Esta fue la parte más dificl del proyecto, debido a que tenemos que tener muy presente la secuencia de las instrucciones. Yo elaboré un GDA del cual saqué las instrucciones de 3 direcciones, fue sencillo la impresión porque todos los nodos tenían solo 2 hijos que es equivalente a una instrucción de 3 direcciones.

Código identado: