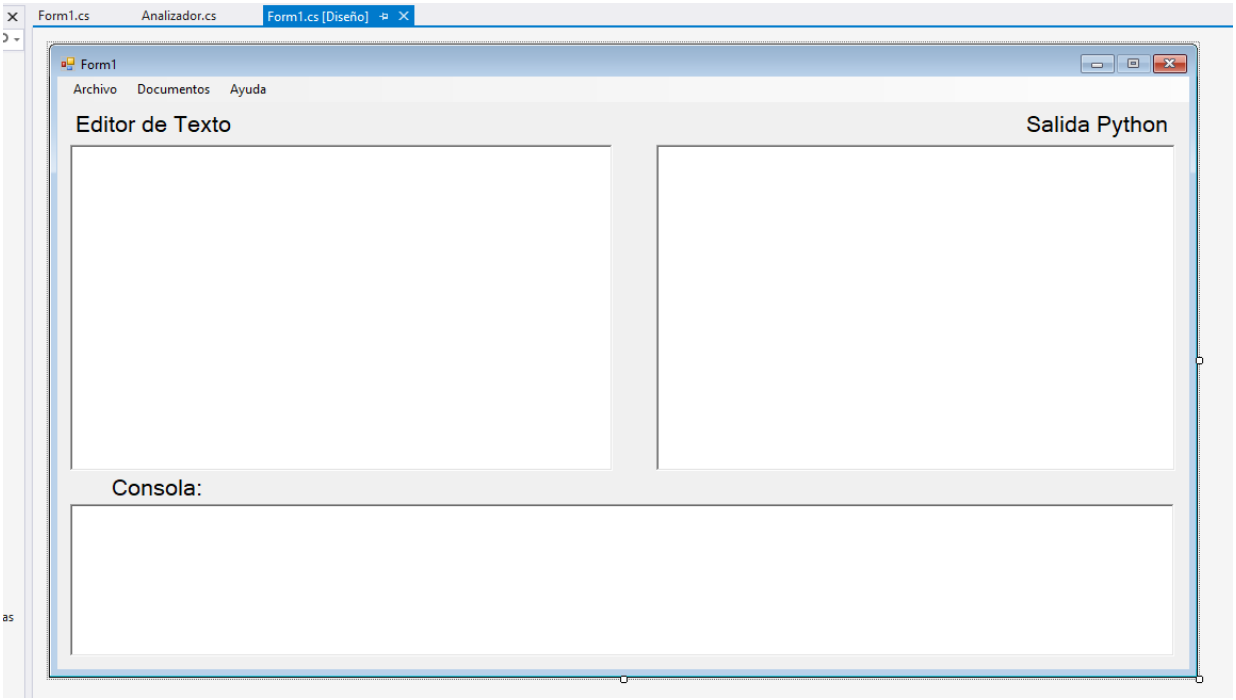
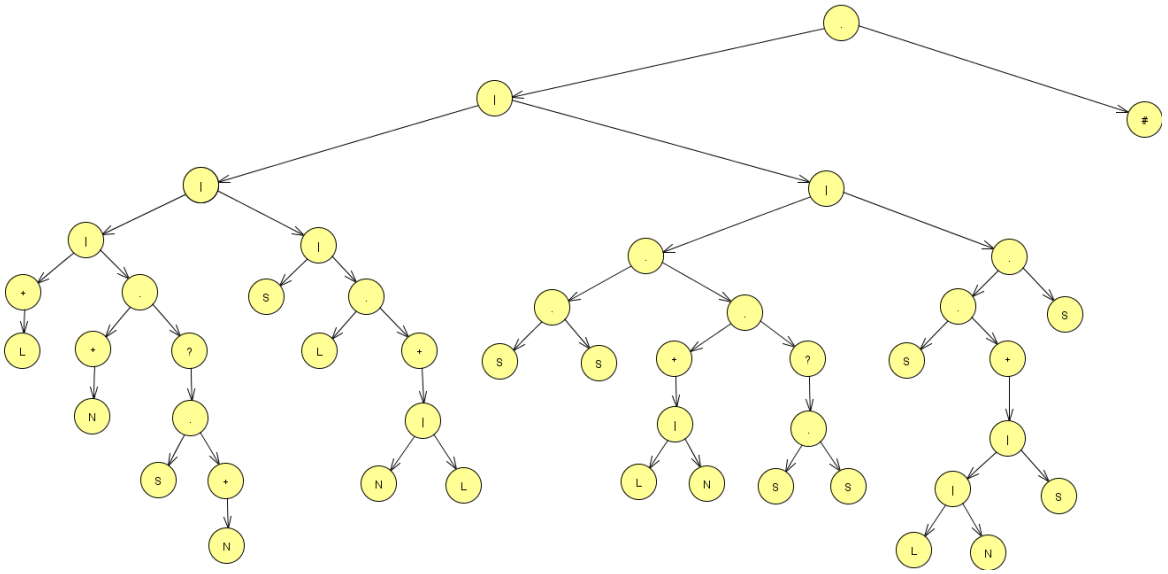
Manual Técnico (Proyecto 2)

- Diseño Principal del Programa



- Lógica del Analizador

Diagrama de Árbol



Autómata Finito Determinista

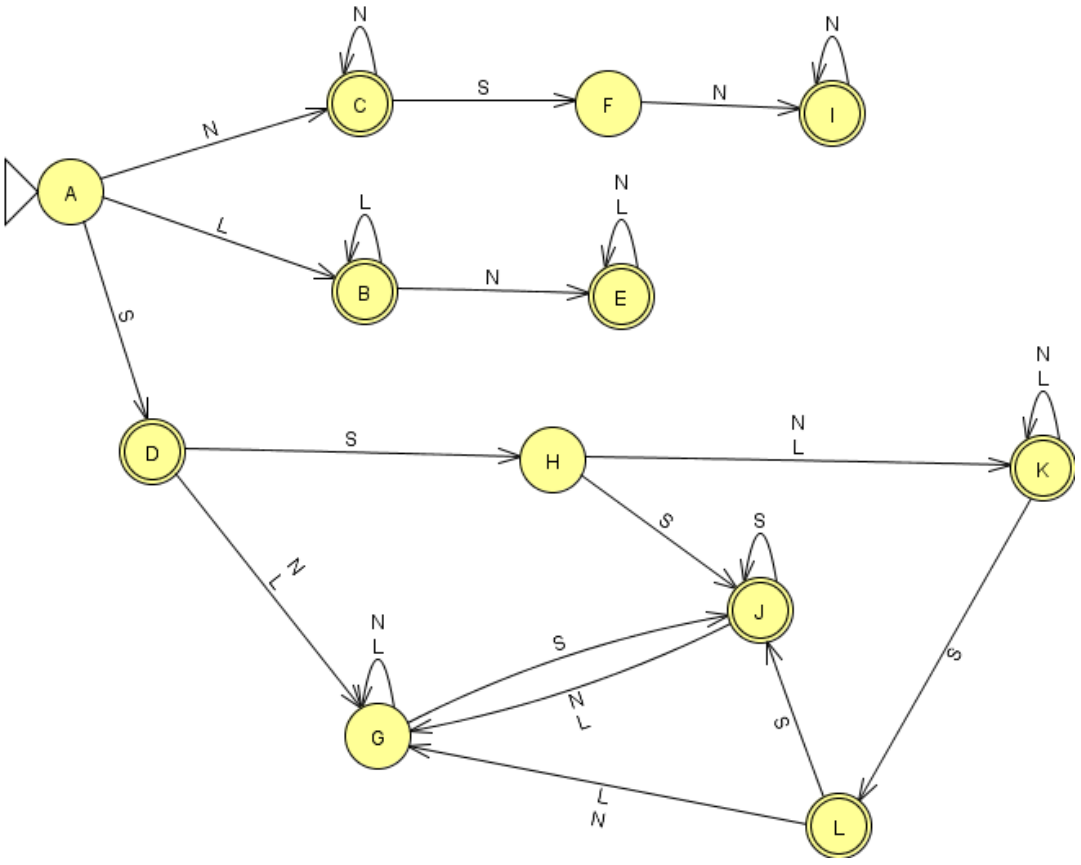


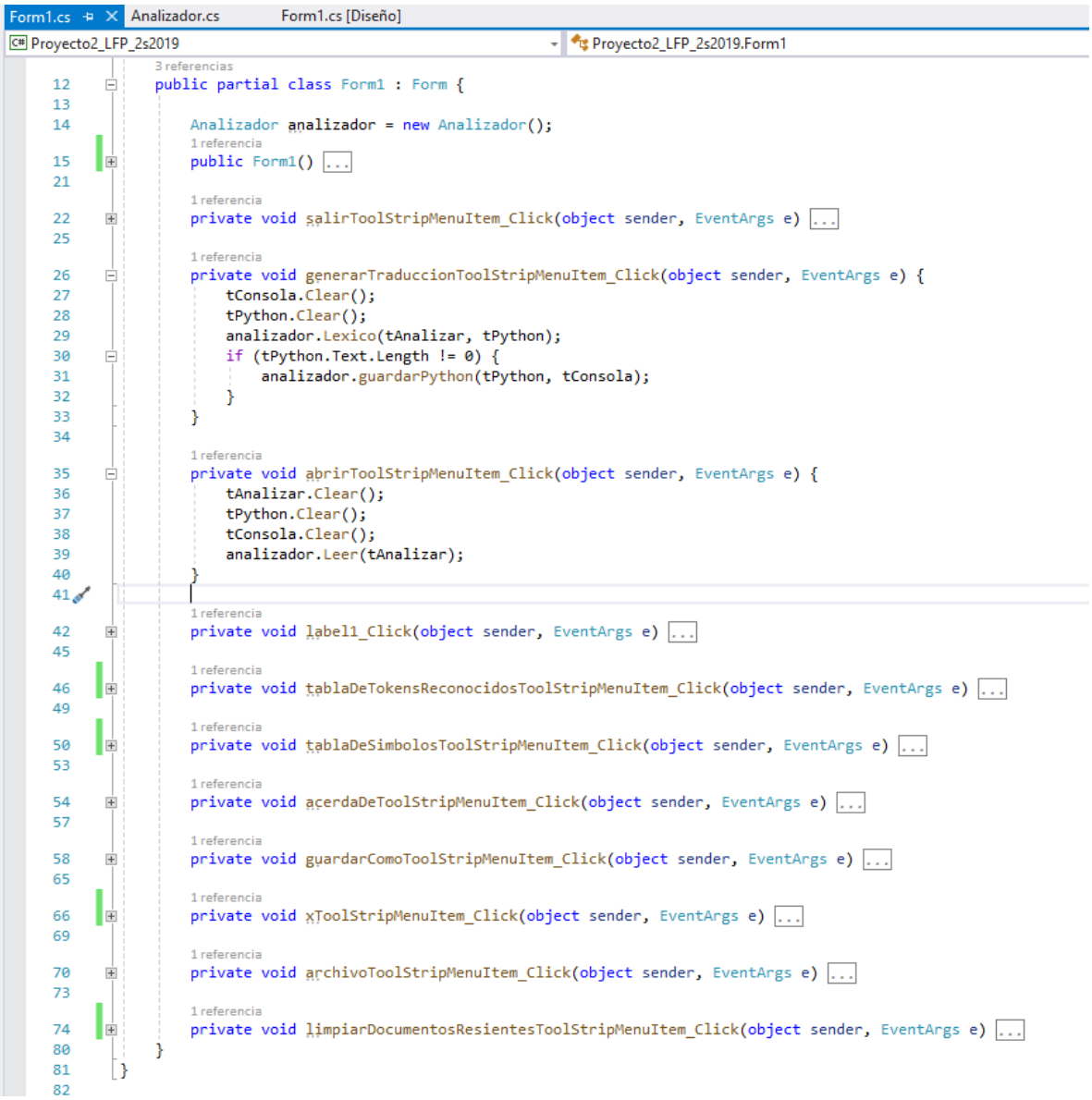
Tabla de Follows

Tabla de Follows		
no.	carácter	follow
1	L	1,20
2	N	2,3,20
3	S	4
4	N	4,20
5	S	20
6	L	7,8
7	N	7,8,20
8	L	7,8,20
9	S	10
10	S	11,12
11	L	11,12,13,20
12	N	11,12,13,20
13	S	14
14	S	20
15	S	16,17,18
16	L	16,17,18,19
17	N	16,17,18,19
18	S	16,17,18,19
19	S	20
20	#	0

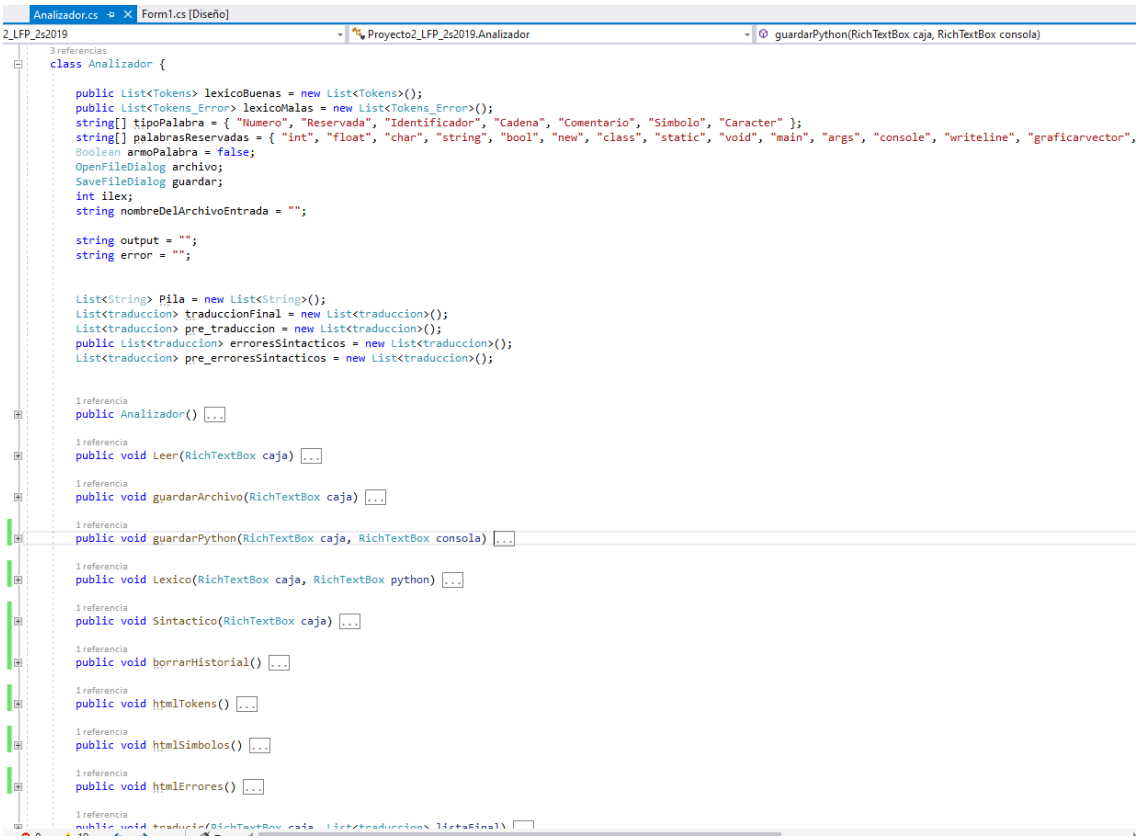
Tabla de Transiciones

Tabla de Transiciones				
Estado	L	S	N	Fin de Cadena
A={1,2,5,6,9,15}	B	C	D	****
B={1,20,7,8}	B	E	****	SI
C={2,3,20}	****	C	F	SI
D={20,10,16,17,18}	G	G	H	SI
E={7,8,20}	E	E	****	SI
F={4}	****	I	****	****
G={16,17,18,19}	G	G	J	****
H={11,12,16,17,18,19}	K	K	J	****
I={4,20}	****	I	****	SI
J={16,17,18,19,20}	G	G	J	SI
K={11,12,13,20,16,17,18,19}	K	K	L	SI
L={14,16,17,18,19,20}	G	G	J	SI

- Código Principal de formulario



- Método Analizador



Carga de Archivo

```
public void Leer(RichTextBox caja) {
    archivo = new OpenFileDialog();
    archivo.DefaultExt = "txt";
    archivo.Filter = "Archivos de texto (*.txt)|*.txt";

    if (archivo.ShowDialog() == DialogResult.OK) {
        nombreDelArchivoEntrada = archivo.SafeFileName;
        StreamReader leer;
        leer = new StreamReader(archivo.FileName);
        while (leer.Peek() > -1) {
            //peek revisa el siguiente caracter y tiene que ser mayor a -1 que representa que no hay nada
            String linea = leer.ReadLine().Trim();
            if (!String.IsNullOrEmpty(linea)) {
                caja.AppendText(linea + "\n");

            }
        }

    } else {
        MessageBox.Show("Error Archivo no Seleccionado");

    }

}
```

Analizador Léxico

```
public void Lexico(RichTextBox caja, RichTextBox python) {
    python.Clear();
    if (caja.Text.Length == 0) {
        MessageBox.Show("Caja Vacia");
    } else {

        lexicoBuenas.Clear();
        lexicoMalas.Clear();

        double opcion = 0;
        String palabraArmada = "";
        int filaE = 0;
        int columnaE = 0;
        filaE += 1;
        char[] letra = caja.Text.ToArray();
        for (int i = 0; i < letra.Length; i++)...

        Console.WriteLine("--------------");
        if (lexicoBuenas.Count != 0)...

        if (lexicoMalas.Count != 0)...

        void revisarSimbolo(char signo)...

        Sintactico(python);
    }


}
```

Guardar Archivo

```
public void guardarArchivo(RichTextBox caja) {

    guardar = new SaveFileDialog();
    guardar.Filter = "Archivos de texto (*.cs)|*.cs";
    guardar.DefaultExt = "cs";
    guardar.ShowDialog();
    if (guardar.FileName != "") {
        File.WriteAllText(guardar.FileName, caja.Text);
    } else {
        MessageBox.Show("No hay Nombre");
    }

}
```

Generar HTML

```
public void htmlTokens() {

    string webB = Path.Combine(Application.StartupPath, "Proyecto2_Tokens.html");
    string inicio = "<html>" +
        "<head>" +
        "</head>" +
        "<body style='background-color:#34495E'>";
    string fin =
        "</table>" +
        "<body>" +
        "</html>";

    string fecha = DateTime.Now.ToString("dddd, dd MMMM yyyy HH:mm:tt");
    string medioB = "<h1 align=center><font color ='white'> Lista de Tokens de: " + fecha + "</h1>" + "<br>" +
        "<h1 align=center><font color ='white'> Archivo de Entrada: " + nombreDelArchivoEntrada + "</h1>" +
        "<h1 align=center><font color ='white'> Archivo de Salida: python.py </h1>" +
        "<table border=11 , align='center', bordercolor='orange'>" +
        "<tr align=center> <th><font color ='white'> Numero </th> <th><font color ='white'> Palabra </th> <th><font color ='white'> Tipo </th></tr>";
    if (lexicoBuenas.Count != 0) {
        for (int i = 0; i < lexicoBuenas.Count; i++) {
            if (lexicoBuenas[i].tipoPalabra.Equals("Reservada_" + lexicoBuenas[i].palabra.ToLower()) | lexicoBuenas[i].tipoPalabra.Equals("Numero_entero") |
                medioB += "\n<tr align=center> <td><font color ='white'> " + (i + 1) + " </td> <td><font color ='white'> " + lexicoBuenas[i].palabra + " </td

            }
        }
    }


    string contenidoB = inicio + medioB + fin;

    File.WriteAllText(webB, contenidoB);

    Process start = new Process();
    start.StartInfo.FileName = webB;
    start.Start();
```

Traducción

```
public void traducir(RichTextBox caja, List<traduccion> listaFinal) {

    int i = 0;
    Boolean print = false;
    Boolean mif = false;
    Boolean ultimaLlaveIf = false;
    Boolean mifelse = false;
    Boolean instruccionesIf = false;
    Boolean mwhile = false;

    while (i < listaFinal.Count) {
        if (listaFinal[i].Referencia.Equals("Comentario")) {
            if (listaFinal[i].palabra.StartsWith("//")) {
                if (instruccionesIf == true) {
                    caja.AppendText("\t # " + listaFinal[i].palabra.Substring(2) + "\n");
                } else {
                    caja.AppendText("# " + listaFinal[i].palabra.Substring(2) + "\n");
                }
                i++;
            } else if (listaFinal[i].palabra.StartsWith("/*")) {
                caja.AppendText("..." + listaFinal[i].palabra.Substring(2, listaFinal[i].palabra.Length - 4) + "...\n");
                i++;
            }
        } else if (listaFinal[i].Referencia.Equals("true_false")) {
            if (listaFinal[i].palabra.Equals("true", StringComparison.OrdinalIgnoreCase)) {
                caja.AppendText("True");
                i++;
            } else if (listaFinal[i].palabra.Equals("false", StringComparison.OrdinalIgnoreCase)) {
                caja.AppendText("False");
                i++;
            }
        } else if (listaFinal[i].Referencia.Equals("Reservada_console")) {
            i += 2;
            if (instruccionesIf == true) {
                caja.AppendText("\t print");
            } else {
                caja.AppendText("print");
            }
            print = true;
            i++;
        } else if (print == true && listaFinal[i].palabra.Equals("+")) {
            caja.AppendText(" , ");
            i++;
        } else if (print == true && listaFinal[i].Referencia.Equals("punto_coma")) {
            caja.AppendText("\n");
            print = false;
            i++;
        } else if (print == false && (listaFinal[i].Referencia.Equals("punto_coma") | listaFinal[i].Referencia.Equals("coma"))) {
            caja.AppendText("\n");
            i++;

        } else if (listaFinal[i].Referencia.Equals("Reservada_if")) {
```

Gramática Análisis Sintáctico

⇨ PROGRAMA -> Class id { <MAIN> }

<MAIN> -> Static Void Main (<ARGUMENTOS>) { <CUERPO> }

<ARGUMENTOS> -> String [ ] args
| λ

<CODIGO> -> <DECLARACION> <CODIGO>
| <ASIGNACION> <CODIGO>
| <PRINT> <CODIGO>
| <IF> <CODIGO>

<DECLARACION> -> Tipo <ASIGNACION>

<ASIGNACION> -> Id <VALOR> ;
| [ ] id <VALOR> ;

<VALOR> -> = <CONJUNTO>
| <SIGNO> <INCREMENTO>
| <MULTIPLE>

<CONJUNTO> -> < JERARQUIA >
| id [< JERARQUIA >]
| { valor <MULTIPLE> }

<OPERACIÓN> -> <SIGNO> <JERARQUIA>

| <MULTIPLE>

< JERARQUIA > -> valor <OPERACIÓN>

| id <OPERACIÓN>

<INCREMENTO> -> <SIGNO> <OPERACIÓN>

<MULTIPLE> -> , <ARREGLO>

| λ

<ARREGLO> -> id <VALOR>

| valor <VALOR>

<PRINT> -> console . writeline (<CPRINT>) ;

<CPRINT> -> cadena <MPRINT>

| id <MPRINT>

| <MPRINT>

<MPRINT> -> + <CPRINT>

| λ

<IF> -> if (<CONDICIONAL>) { <INSTRUCCIÓN> } <ELSE>

<WHILE> -> while (<CONDICIONAL>) { <INSTRUCCIÓN> }

<CONDICIONAL> -> id <PSIMBOLO>

| valor < PSIMBOLO >

< PSIMBOLO > -> <>=! <SSIMBOLO>

| <ENLACE>

<SSIMBOLO> -> = <VALORC>

| <VALORC>

<VALORC> -> valor <ENLACE>

| id <ENLACE>

<ENLACE> -> & <SENLACE>

| | < CONDICIONAL >

| <SENLACE>

<SENLACE> -> & < CONDICIONAL >

| λ

<INSTRUCCIÓN> -> Tipo <ASIGNACION> <INSTRUCCIÓN>

| Id <VALOR> ; <INSTRUCCIÓN>

| if (<CONDICIONAL>) { <INSTRUCCIÓN> } <ELSE>

| console . writeline (<CPRINT>) ; <INSTRUCCIÓN>

| λ

<ELSE> -> else { <INSTRUCCIÓN> }

| <INSTRUCCIÓN>