```
EJEMPLO 1: Código complejo válido
============================================================
ANALIZADOR COMPLETO DE C++
============================================================

1. CÓDIGO A ANALIZAR:
------------------------------
 1: int main() {
 2:         int x = 5;
 3:         int y = 10;
 4:         int sum = 0;
 5:         int i = 0;
 6:         int j = 0;
 7:
 8:         sum = x + y;
 9:
10:         if (sum > 10) {
11:             cout << "Sum is greater than 10" << endl;
12:         } else {
13:             cout << "Sum is not greater than 10" << endl;
14:         }
15:
16:         while (i < 3) {
17:             cout << i << endl;
18:             i = i + 1;
19:         }
20:
21:         for (j = 0; j < 5; j = j + 1) {
22:             cout << "Loop iteration: " << j << endl;
23:         }
24:
25:         return 0;
26:     }

2. ANÁLISIS LÉXICO:
------------------------------
Token        | Valor
INT          | int
ID           | main
LPAREN       | (
RPAREN       | )
LBRACE       | {
INT          | int
ID           | x
ASSIGN       | =
NUMBER       | 5
SEMICOLON    | ;
INT          | int
ID           | y
ASSIGN       | =
NUMBER       | 10
SEMICOLON    | ;
INT          | int
ID           | sum
ASSIGN       | =
```

```
NUMBER       | 0
SEMICOLON    | ;
INT          | int
ID           | i
ASSIGN       | =
NUMBER       | 0
SEMICOLON    | ;
INT          | int
ID           | j
ASSIGN       | =
NUMBER       | 0
SEMICOLON    | ;
ID           | sum
ASSIGN       | =
ID           | x
PLUS         | +
ID           | y
SEMICOLON    | ;
IF           | if
LPAREN       | (
ID           | sum
GT           | >
NUMBER       | 10
RPAREN       | )
LBRACE       | {
COUT         | cout
SHIFT_OUT    | <<
STRING_LITERAL | Sum is greater than 10
SHIFT_OUT    | <<
ID           | endl
SEMICOLON    | ;
RBRACE       | }
ELSE         | else
LBRACE       | {
COUT         | cout
SHIFT_OUT    | <<
STRING_LITERAL | Sum is not greater than 10
SHIFT_OUT    | <<
ID           | endl
SEMICOLON    | ;
RBRACE       | }
WHILE        | while
LPAREN       | (
ID           | i
LT           | <
NUMBER       | 3
RPAREN       | )
LBRACE       | {
COUT         | cout
SHIFT_OUT    | <<
ID           | i
SHIFT_OUT    | <<
ID           | endl
SEMICOLON    | ;
ID           | i
ASSIGN       | =
```

```
ID           | i
PLUS         | +
NUMBER       | 1
SEMICOLON    | ;
RBRACE       | }
FOR          | for
LPAREN       | (
ID           | j
ASSIGN       | =
NUMBER       | 0
SEMICOLON    | ;
ID           | j
LT           | <
NUMBER       | 5
SEMICOLON    | ;
ID           | j
ASSIGN       | =
ID           | j
PLUS         | +
NUMBER       | 1
RPAREN       | )
LBRACE       | {
COUT         | cout
SHIFT_OUT    | <<
STRING_LITERAL | Loop iteration:
SHIFT_OUT    | <<
ID           | j
SHIFT_OUT    | <<
ID           | endl
SEMICOLON    | ;
RBRACE       | }
RETURN       | return
NUMBER       | 0
SEMICOLON    | ;
RBRACE       | }

3. ANÁLISIS SINTÁCTICO:
------------------------------
✓ Análisis sintáctico exitoso

Árbol de Sintaxis Abstracta (AST):
PROGRAM: None
  FUN_DEF: main
    TYPE: int
    BLOCK: None
      VAR_DECL: None
        TYPE: int
        INIT_DECL: x
          NUMBER: 5
      VAR_DECL: None
        TYPE: int
        INIT_DECL: y
          NUMBER: 10
      VAR_DECL: None
        TYPE: int
        INIT_DECL: sum
```

```
    NUMBER: 0
  VAR_DECL: None
    TYPE: int
    INIT_DECL: i
      NUMBER: 0
  VAR_DECL: None
    TYPE: int
    INIT_DECL: j
      NUMBER: 0
  ASSIGN: None
    ID: sum
    BINOP: +
      ID: x
      ID: y
  IF: None
    BINOP: >
      ID: sum
      NUMBER: 10
    BLOCK: None
      COUT: None
        STRING_LITERAL: Sum is greater than 10
        ID: endl
    BLOCK: None
      COUT: None
        STRING_LITERAL: Sum is not greater than 10
        ID: endl
  WHILE: None
    BINOP: <
      ID: i
      NUMBER: 3
    BLOCK: None
      COUT: None
        ID: i
        ID: endl
      ASSIGN: None
        ID: i
        BINOP: +
          ID: i
          NUMBER: 1
  FOR: None
    ASSIGN: None
      ID: j
      NUMBER: 0
    BINOP: <
      ID: j
      NUMBER: 5
    ASSIGN: None
      ID: j
      BINOP: +
        ID: j
        NUMBER: 1
    BLOCK: None
      COUT: None
        STRING_LITERAL: Loop iteration:
        ID: j
        ID: endl
```

```
      RETURN: None
        NUMBER: 0

4. ANÁLISIS SEMÁNTICO:
----------------------------
✗ Errores semánticos encontrados:
  - Variable 'endl' no declarada
  - Variable 'endl' no declarada
  - Variable 'endl' no declarada
  - Variable 'endl' no declarada

5. ANÁLISIS SEMÁNTICO POR LÍNEAS:
------------------------------------------------------------
Línea  | Tipo            | Descripción            | Código
------------------------------------------------------------
1      | FUNCTION_DEF    | Function Definition    | int main() {
2      | VAR_DECL_INIT   | Variable Decl+Init     |         int x = 5;
3      | VAR_DECL_INIT   | Variable Decl+Init     |         int y = 10;
4      | VAR_DECL_INIT   | Variable Decl+Init     |         int sum = 0;
5      | VAR_DECL_INIT   | Variable Decl+Init     |         int i = 0;
6      | VAR_DECL_INIT   | Variable Decl+Init     |         int j = 0;
8      | ASSIGN          | Assignment             |         sum = x + y;
10     | BRANCH          | If Statement           |         if (sum > 10) {
11     | OUTPUT          | Output Statement       |             cout << "Sum is greater than 1...
12     | UNKNOWN         | Unknown                |         } else {
13     | OUTPUT          | Output Statement       |             cout << "Sum is not greater th...
14     | UNKNOWN         | Unknown                |         }
16     | WHILE_LOOP      | While Loop             |         while (i < 3) {
17     | CALCULATIONS    | Math Calculation       |             cout << i << endl;
18     | CALCULATIONS    | Math Calculation       |             i = i + 1;
19     | UNKNOWN         | Unknown                |         }
21     | CALCULATIONS    | Math Calculation       |         for (j = 0; j < 5; j = j + 1) {
22     | OUTPUT          | Output Statement       |             cout << "Loop iteration: " << ...
23     | UNKNOWN         | Unknown                |         }
25     | RETURN          | Return Statement       |         return 0;
26     | UNKNOWN         | Unknown                |     }
------------------------------------------------------------

ESTADÍSTICAS:
- Total de líneas analizadas: 21
- Líneas clasificadas: 16
- Líneas sin clasificar: 5
- Porcentaje de clasificación: 76.2%

============================================================
\n\nEJEMPLO 2: Código con errores
============================================================
ANALIZADOR COMPLETO DE C++
============================================================

1. CÓDIGO A ANALIZAR:
----------------------------
 1: int main() {
 2:         int x = 5;
 3:         y = x + 3;          // variable y no declarada
 4:         void z;             // objeto de tipo void no válido
```

```
 5:
 6:         if (x > 2) {
 7:             cout << "x is greater than 2" << endl;
 8:         }
 9:
10:         return "hello";     // devuelve string en función int
11:     }

2. ANÁLISIS LÉXICO:
----------------------------
Token         | Valor
INT           | int
ID            | main
LPAREN        | (
RPAREN        | )
LBRACE        | {
INT           | int
ID            | x
ASSIGN        | =
NUMBER        | 5
SEMICOLON     | ;
ID            | y
ASSIGN        | =
ID            | x
PLUS          | +
NUMBER        | 3
SEMICOLON     | ;
VOID          | void
ID            | z
SEMICOLON     | ;
IF            | if
LPAREN        | (
ID            | x
GT            | >
NUMBER        | 2
RPAREN        | )
LBRACE        | {
COUT          | cout
SHIFT_OUT     | <<
STRING_LITERAL | x is greater than 2
SHIFT_OUT     | <<
ID            | endl
SEMICOLON     | ;
RBRACE        | }
RETURN        | return
STRING_LITERAL | hello
SEMICOLON     | ;
RBRACE        | }

3. ANÁLISIS SINTÁCTICO:
----------------------------
Error sintáctico en token 'void' línea 5
✓ Análisis sintáctico exitoso

Árbol de Sintaxis Abstracta (AST):
PROGRAM: None
```

```
  FUN_DEF: main
    TYPE: int
    BLOCK: None
      VAR_DECL: None
        TYPE: int
        INIT_DECL: x
          NUMBER: 5
      ASSIGN: None
        ID: y
        BINOP: +
          ID: x
          NUMBER: 3
      ID: z
      IF: None
        BINOP: >
          ID: x
          NUMBER: 2
        BLOCK: None
          COUT: None
            STRING_LITERAL: x is greater than 2
            ID: endl
      RETURN: None
        STRING_LITERAL: hello

4. ANÁLISIS SEMÁNTICO:
------------------------------
✗ Errores semánticos encontrados:
  - Variable 'y' no declarada
  - Variable 'z' no declarada
  - Variable 'endl' no declarada
  - Tipo de retorno incompatible: int vs string

5. ANÁLISIS SEMÁNTICO POR LÍNEAS:
------------------------------------------------------------
Línea  | Tipo            | Descripción            | Código
------------------------------------------------------------
1      | FUNCTION_DEF    | Function Definition    | int main() {
2      | VAR_DECL_INIT   | Variable Decl+Init     |         int x = 5;
3      | ASSIGN          | Assignment             |         y = x + 3;          // variable y ...
4      | UNKNOWN         | Unknown                |         void z;             // objeto de t...
6      | BRANCH          | If Statement           |         if (x > 2) {
7      | OUTPUT          | Output Statement       |             cout << "x is greater than 2" ...
8      | UNKNOWN         | Unknown                |         }
10     | RETURN          | Return Statement       |         return "hello";     // devuelve st...
11     | UNKNOWN         | Unknown                |     }
------------------------------------------------------------

ESTADÍSTICAS:
- Total de líneas analizadas: 9
- Líneas clasificadas: 6
- Líneas sin clasificar: 3
- Porcentaje de clasificación: 66.7%

============================================================
```