

# Plan de pruebas del Proyecto 1a

# Análizador Sintáctico y Léxico

## *Modificación del Compilador de Mini Triangle de Watt y Brown*

**Versión 1.0**

**Preparado por:**

- **Joel Barrantes Garro**
  **2013120962**
- **Edisson López Díaz**
  **2013103311**
- **Katerine Molina Sanchez**
  **2013109225**

**Instituto Tecnológico de Costa Rica**

**23 de octubre de 2017**

Índice

**9. Análisis de la cobertura del plan de pruebas** **2**

**10. Plan de pruebas** **2**

10.1. Pruebas para el analizador léxico 2

Sentencia BEGIN 2

Sentencia ε 2

Sentencia SKIP 3

Sentencia WHILE 3

10.2. Pruebas para el analizador sintáctico 4

Sentencia IF THEN ELSE (estructura eliminada) 4

Sentencia IF THEN ELSE END (estructura nueva) 4

Sentencia FOR VAR FROM TO WHILE 10

Sentencia FUNC ( ) : (estructura nueva) 12

Sentencia LET IN (estructura eliminada) 13

Sentencia LET IN END 13

Sentencia LOCAL IN END (estructura nueva) 15

Sentencia PAR END (estructura nueva) 17

Sentencia PROC ( ) END (estructura nueva) 18

Sentencia RECURSIVE y PROC FUNC (estructura nueva) 20

Sentencia REPEAT WHILE DO END (estructura nueva) 22

Sentencia REPEAT UNTIL DO END (estructura nueva) 23

Sentencia REPEAT DO WHILE END (estructura nueva) 25

Sentencia REPEAT DO UNTIL END (estructura nueva) 26

Sentencia INITIALIZED VAR (estructura nueva) 28

## 9. Análisis de la cobertura del plan de pruebas

El plan de pruebas se hará en dos fases (léxica y sintáctica). Las pruebas de estructuras eliminadas, como el **begin**, solo tienen uno o dos casos de prueba de error, y ningún caso de prueba (no se puede hacer un caso de prueba positiva). Las pruebas de estructuras modificadas, como el **local in end**, o estructuras nuevas, como **recursive end**, tiene dos o más casos de error y dos o más casos de éxito.

## 10. Plan de pruebas

### 10.1. Pruebas para el analizador léxico

a. Sentencia **BEGIN**

| Nombre | BeginErr1.tri |
|---|---|
| Objetivo | Reconocer el error léxico de la estructura **BEGIN**. Debe detectar que **begin** ya no forma parte del lenguaje. |
| Diseño | let<br>  var a : Integer<br>in begin<br>  getint (var a);<br>  putint (a)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | Compilation was unsuccessful. |

b. Sentencia **ε**

| Nombre | EpsilonErr1.tri |
|---|---|
| Objetivo | Reconocer el error léxico de la estructura **vacía**. Debe detectar que un archivo vacío ya no puede ser reconocido. |
| Diseño | ! este archivo está vacío |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "" cannot start a command 1..1<br>Compilation was unsuccessful. |

c. Sentencia **SKIP**

| Nombre | SkipErr1.tri |
|---|---|
| Objetivo | Reconocer el error léxico de la estructura **SKIP**. Debe detectar que **skip** también debe llevar **;** al final de la sentencia. |
| Diseño | skip skip |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "skip" not expected after end of program 1..1<br>Compilation was unsuccessful. |

| Nombre | SkipOK1.tri |
|---|---|
| Objetivo | Reconocer la nueva estructura **SKIP**. Debe detectar que **skip** forma parte del lenguaje. |
| Diseño | skip |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | SkipOK2.tri |
|---|---|
| Objetivo | Reconocer la nueva estructura **SKIP**. Debe detectar que **skip** forma parte del lenguaje. |
| Diseño | skip; skip; skip; skip;<br>skip; skip; skip; skip |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

d. Sentencia **WHILE**

| Nombre | WhileErr1.tri |
|---|---|
| Objetivo | Reconocer el error léxico de la estructura **WHILE**. Debe detectar que **while** ya no forma parte del lenguaje. |
| Diseño | while true do<br>  skip |

| | |
|---|---|
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "while" cannot start a command 1..1<br>Compilation was unsuccessful. |

## 10.2. Pruebas para el analizador sintáctico

### e. Sentencia **IF THEN ELSE** (estructura eliminada)

| Nombre | IfThenElseErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **IF**. Debe detectar que falta el **end** de la estructura **if**. |
| Diseño | let<br>  var a := 1<br>in<br>  if a > 0 then<br>  put ('M')<br>  else<br>  put ('m')<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 13..13<br>Compilation was unsuccessful. |

### f. Sentencia **IF THEN ELSE END** (estructura nueva)

| Nombre | IfThenElseEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **IF**. Debe detectar que falta el **else** de la estructura **if**. |
| Diseño | let<br>  var a := 1<br>in<br>  if a > 0 then<br>  put ('M')<br>  end<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "else" expected here 10..10<br>Compilation was unsuccessful. |

| Nombre | IfThenElseEndErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **IF**. Debe detectar que falta el **then** de la estructura **if** (se escribió un **do** en lugar del **then**). |
| Diseño | let<br>  var a := 1<br>in<br>  if a > 0 do<br>  put ('M')<br>  else<br>  a := a<br>  end<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "then" expected here 8..8<br>Compilation was unsuccessful. |

| Nombre | IfThenElseEndErr4.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **IF**. Debe detectar que falta el **then** de la estructura **if**. |
| Diseño | let<br>  var a := 1<br>in<br>  if (a > 0)<br>  put ('M')<br>  else<br>  a := a<br>  end<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "then" expected here 9..9<br>Compilation was unsuccessful. |

| Nombre | IfThenElseEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **IF** con el objetivo de imprimir una ‘M’ |
| Diseño | let<br>  var a := 1 |

| | |
|---|---|
| | in<br>  if a > 0 then<br>  put ('M')<br>  else<br>  put ('m')<br>  end<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | IfThenElseEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **IF** con el objetivo de imprimir un 1 |
| Diseño | if true then<br>  print (1)<br>else<br>  print (0)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

| Nombre | IfThenElseEndOK3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **IF** con el objetivo de imprimir un 0 |
| Diseño | if false then<br>  print (1)<br>else<br>  print (0)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

### g. Sentencia **FOR VAR FROM TO**

| Nombre | ForVarFromToErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR**. Debe detectar que falta |

| | |
|---|---|
| | el **var** de la estructura **for**. |
| Diseño | for<br>  i := 1 to 10 do<br>  putint (1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "var" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR**. Debe detectar que falta **:=** de la estructura **for**. |
| Diseño | for<br>  var i  1 to 10 do<br>  putint (1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: ":=" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR**. Debe detectar que falta **to** de la estructura **for**. |
| Diseño | for<br>  var i := 1  10 do<br>  putint (1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "to" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR** con el objetivo de imprimir una secuencia desde 1 hasta 10. |
| Diseño | for |

| | |
|---|---|
| | var i := 1 to 10 do<br>putint (1)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR** con el objetivo de imprimir diez veces los números uno y dos. |
| Diseño | for<br>var i := 1 to 10 do<br>putint (1);<br>putint (2)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToOK3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR** con el objetivo de imprimir diez veces el número uno. |
| Diseño | let<br>var k := 1<br>in<br>for var i := k to 10 do<br>k := k + 1 ;<br>print (1)<br>end<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

### h. Sentencia **FOR VAR FROM TO UNTIL**

| Nombre | ForVarFromToUntilErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR UNTIL**. Debe detectar que falta la condición de la estructura **for**. |
| Diseño | for<br>var i := 1 to 10 until do |

| | |
|---|---|
| | putint (1);<br>putint (2)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "do" cannot start an expression 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToUntilErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR UNTIL**. Debe detectar que sobra un **do** de la estructura **for**. |
| Diseño | for<br>var i := 1 to 10 until true do do<br>putint (1);<br>putint (2)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "do" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToUntilOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR UNTIL** con el objetivo de no imprimir nada. |
| Diseño | for<br>var i := 1 to 10 until false do<br>putint (1);<br>putint (2)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToUntilOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR UNTIL** con el objetivo de imprimir veinte veces el número uno. |
| Diseño | for<br>var i := 1 to 10 until i > 20 do<br>putint (1) |

| | |
|---|---|
| | end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToUntilOK3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR UNTIL** con el objetivo de imprimir ocho veces el número uno. |
| Diseño | for<br>  var i := 1 to 10 until i > 8 do<br>    putint (1)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

### i. Sentencia **FOR VAR FROM TO WHILE**

| Nombre | ForVarFromToWhileErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR WHILE**. Debe detectar que falta la condición de la estructura **for**. |
| Diseño | for<br>  var i := 1 to 10 while do<br>    putint (1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "do" cannot start an expression 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToWhileErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FOR WHILE**. Debe detectar que sobra un **do** de la estructura **for**. |
| Diseño | for<br>  var i := 1 to 10 while true do do<br>    putint (1)<br>end |
| Resultados Esperados | Error |

| | |
|---|---|
| Resultados Observados | ERROR: "do" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | ForVarFromToWhileOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR WHILE** con el objetivo de imprimir diez veces el número uno. |
| Diseño | for<br>var i := 1 to 10 while true do<br>putint (1)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToWhileOK3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR WHILE** con el objetivo de imprimir veinte veces el número uno. |
| Diseño | for<br>var i := 1 to 10 while i < 20 do<br>putint (1)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ForVarFromToWhileOK4.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FOR UNTIL** con el objetivo de imprimir del número uno al cinco. |
| Diseño | for<br>var i := 1 to 10 while i < 6 do<br>putint (i)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo. |
| Resultados Observados | Compilation was successful. |

j. Sentencia **FUNC ( ) :** (estructura nueva)

| Nombre | Func():Err1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FUNC () :**. Debe detectar que falta el símbolo **:** para luego indicar el tipo de dato que retorna la estructura **proc () end**. |
| Diseño | let<br>  func soyUnaFuncion()  integer ~<br>  2+3<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: ":" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | Func():Err2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **FUNC (): END**. Debe detectar que falta el símbolo ~ para indicar el inicio del **expresión** de la estructura **proc () end**. |
| Diseño | let<br>  func soyUnaFuncion() : integer<br>  2+3<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "~" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | Func()OK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **PROC ( ) END** con el objetivo de devolver un integer. |
| Diseño | let<br>  func soyUnaFuncion() : integer ~<br>  2+3<br>in |

| | |
|---|---|
| | skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | Func()OK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **FUNC ( )** con el objetivo de recibir un integer y devolver un integer. |
| Diseño | let<br>func soyUnaFuncion(parameter1:integer) : integer ~<br>2+3<br>in<br>skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

k. Sentencia **LET IN** (estructura eliminada)

| Nombre | LetInErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **LET**. Debe detectar que falta el **end** de la estructura **let** |
| Diseño | let<br>var a := 1<br>in<br>putint (a) |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 10..10<br>Compilation was unsuccessful. |

l. Sentencia **LET IN END**

| Nombre | LetErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **LET**. Debe detectar que falta alguna declaración dentro de la estructura **let in end** |
| Diseño | let |

| | |
|---|---|
| | in<br>  putint(n)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "in" cannot start a declaration 6..6<br>Compilation was unsuccessful. |

| Nombre | LetErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **LET**. Debe detectar que falta algún comando de la estructura **let in end** |
| Diseño | let<br>  var n := 1<br>in<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" cannot start a command 10..10<br>Compilation was unsuccessful. |

| Nombre | LetOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **LET** con el objetivo de imprimir un 1 con la variable n. |
| Diseño | let<br>  var n := 1<br>in<br>  putint(n)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

| Nombre | LetOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **LET** con el objetivo de imprimir un 1 con la variable a. |
| Diseño | let<br>  var a := 1<br>in |

| | |
|---|---|
| | skip ;<br>print (a)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

| Nombre | LetOK3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **LET** con el objetivo de imprimir un 2 con la variable a. |
| Diseño | let<br>var a := 1<br>in<br>a := a + 1 ;<br>print (a)<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

| Nombre | LetOK4.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **LET** con el objetivo de imprimir un 1 con la variable a. |
| Diseño | let<br>var a := 0<br>in<br>let<br>const b ~ a ;<br>var a := b + 1<br>in<br>print (a)<br>end<br>end |
| Resultados Esperados | Compilación exitosa del objetivo |
| Resultados Observados | Compilation was successful. |

m. Sentencia **LOCAL IN END** (estructura nueva)

| Nombre | LocalErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **LOCAL**. Debe detectar que falta el **in** o **;** de la estructura **local**. |
| Diseño | let<br>  local<br>  const a ~ 1<br>  const b ~ a + 1<br>  end<br>in<br>  putint(b)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "in" expected here 7..7<br>Compilation was unsuccessful. |

| Nombre | LocalErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **LOCAL**. Debe detectar que falta el **end** de la estructura **local**. |
| Diseño | let<br>  local<br>  const a ~ 1<br>  in<br>  const b ~ a + 1<br>in<br>  putint(b)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 10..10<br>Compilation was unsuccessful. |

| Nombre | LocalOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **LOCAL** con la objetivo de imprimir un 2 con la variable b. |
| Diseño | let<br>  local<br>  const a ~ 1 |

| | |
|---|---|
| | in<br>  const b ~ a + 1<br> end<br>in<br> putint(b)<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

n. Sentencia **PAR END** (estructura nueva)

| Nombre | ParEndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **PAR END**. Debe detectar que falta el **and** de la estructura **par end**. |
| Diseño | let<br> par var asd := 78 end<br>in<br> skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "and" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | ParEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **PAR END**. Debe detectar que faltan las declaraciones de la estructura **par end**. |
| Diseño | let<br> par end<br>in<br> skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" cannot start a declaration 2..2<br>Compilation was unsuccessful. |

| Nombre | ParEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **PAR END** con la objetivo de inicializar dos variables. |
| Diseño | let<br>  par var variable1 := 78 and var variable2 := 78 end<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | ParEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **PAR END** con la objetivo de inicializar cuatro variables. |
| Diseño | let<br>  par var variable1 := 1 and var variable2 := 2<br>  and var variable3 := 3 and var variable4 := 4 end<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

o. Sentencia **PROC ( ) END** (estructura nueva)

| Nombre | Proc()EndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **PROC () END**. Debe detectar que falta el símbolo **:** para indicar el tipo de dato de los parámetros de la estructura **proc () end**. |
| Diseño | let<br>  proc soyUnProcedimiento(parametro1, parametro2) ~<br>  print(parametro1)<br>  end<br>in<br>  skip<br>end |

| | |
|---|---|
| Resultados Esperados | Error |
| Resultados Observados | ERROR: ":" expected here 2..2<br>Compilation was unsuccessful. |

| Nombre | Proc()EndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **PROC () END**. Debe detectar que falta el símbolo ~ para indicar el inicio del **command** de la estructura **proc () end**. |
| Diseño | let<br>  proc soyUnProcedimiento(parametro1:integer, parametro2:integer)<br>  print(parametro1)<br>  end<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "~" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | Proc()EndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **PROC ( ) END** con la objetivo de imprimir el parámetro1 de tipo integer. |
| Diseño | let<br>  proc soyUnProcedimiento(parametro1:integer, parametro2:integer) ~<br>  print(parametro1)<br>  end<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | Proc()EndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **PROC ( ) END** con la objetivo de imprimir el parametro1 y parametro2 ambos de tipo integer. |

| | |
|---|---|
| Diseño | let<br>  proc soyUnProcedimiento(parametro1:integer, parametro2:integer) ~<br>  print(parametro1);<br>  skip;<br>  print(parametro2)<br>  end<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

p. Sentencia **RECURSIVE** y **PROC FUNC** (estructura nueva)

| Nombre | RecursiveErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **RECURSIVE**. Debe detectar que falta el **end** de la estructura **recursive**. |
| Diseño | let<br>  recursive<br>  proc proc1() ~<br>  skip end and func func1(): Integer ~<br>  2 + 1 and proc proc2() ~<br>  skip<br>  end<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 9..9<br>Compilation was unsuccessful. |

| Nombre | RecursiveErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **RECURSIVE**. Debe detectar que falta el **and** de la estructura **recursive**, para agregar otra proc o func. |
| Diseño | let<br>  recursive<br>  proc proc1() ~<br>  skip<br>  end |

| | |
|---|---|
| | end<br>in<br>skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "and" expected here 6..6<br>Compilation was unsuccessful. |

| Nombre | RecursiveOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **RECURSIVE** con el objetivo de validar cada proc y func dentro de la estructura recursive. |
| Diseño | let<br>recursive<br>proc proc1() ~<br>skip end and func func1(): Integer ~<br>2 + 1 and proc proc2() ~<br>skip<br>end<br>end<br>in<br>skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | RecursiveOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **RECURSIVE** con el objetivo de validar cada proc y func dentro de la estructura recursive. |
| Diseño | let<br>recursive<br>proc proc1() ~<br>skip end and func func1(): Integer ~<br>2 + 1 and proc proc2() ~<br>skip end and func func2(): Integer ~<br>4 + 3 and proc proc3() ~<br>skip<br>end<br>end<br>in<br>skip |

| | |
|---|---|
| | end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

q. Sentencia **REPEAT WHILE DO END** (estructura nueva)

| Nombre | RepeatWhileDoEndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT WHILE**. Debe detectar que falta la expresión de la estructura **repeat while**. |
| Diseño | repeat while do<br>  putint(1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "do" cannot start an expression 1..1<br>Compilation was unsuccessful. |

| Nombre | RepeatWhileDoEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT WHILE**. Debe detectar que falta el **end** de la estructura **repeat while**. |
| Diseño | repeat while true do<br>  putint(67) |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | RepeatWhileDoEndErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT WHILE**. Debe detectar que falta el comando de la estructura **repeat while**. |
| Diseño | repeat while true do<br>end |
| Resultados Esperados | Error |

| | |
|---|---|
| Resultados Observados | ERROR: "end" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | RepeatWhileDoEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT WHILE** con el objetivo de imprimir un 0. |
| Diseño | repeat while true do<br>  putint(0)<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | RepeatWhileDoEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT WHILE** con el objetivo de imprimir un 1. |
| Diseño | repeat while true do<br>  skip;<br>  skip;<br>  putint(1)<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

r. Sentencia **REPEAT UNTIL DO END** (estructura nueva)

| Nombre | RepeatUntilDoEndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT UNTIL**. Debe detectar que falta la expresión de la estructura **repeat until**. |
| Diseño | repeat until do<br>  putint(1)<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "do" cannot start an expression 1..1<br>Compilation was unsuccessful. |

| Nombre | RepeatUntilDoEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT UNTIL**. Debe detectar que falta el **end** de la estructura **repeat until**. |
| Diseño | repeat until true do<br>  putint(67) |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | RepeatUntilDoEndErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT UNTIL**. Debe detectar que falta el comando de la estructura **repeat until**. |
| Diseño | repeat until true do<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | RepeatUntilDoEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT UNTIL** con el objetivo de imprimir un 0. |
| Diseño | repeat until false do<br>  print(0)<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | RepeatUntilDoEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT UNTIL** con el objetivo de imprimir un 1. |

| | |
|---|---|
| Diseño | repeat until false do<br>  skip;<br>  skip;<br>  print(1)<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

s. Sentencia **REPEAT DO WHILE END** (estructura nueva)

| Nombre | RepeatDoWhileEndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT DO WHILE**. Debe detectar que falta la expresión de la estructura **repeat do while**. |
| Diseño | repeat do<br>  putint(1)<br>while<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" cannot start an expression 4..4<br>Compilation was unsuccessful. |

| Nombre | RepeatDoWhileEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT DO WHILE**. Debe detectar que falta el **end** de la estructura **repeat do while**. |
| Diseño | repeat do<br>  putint(67)<br>while true |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | RepeatDoWhileEndErr3.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT DO WHILE**. Debe detectar que falta el comando de la estructura **repeat do while**. |

| | |
|---|---|
| Diseño | repeat do<br>while true<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "while" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | RepeatDoWhileEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT DO WHILE** con el objetivo de imprimir un 0. |
| Diseño | repeat do<br>  putint(0)<br>while true<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | RepeatDoWhileEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT DO WHILE** con el objetivo de imprimir un 1. |
| Diseño | repeat do<br>  skip;<br>  skip;<br>  putint(1)<br>while true<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

t. Sentencia **REPEAT DO UNTIL END** (estructura nueva)

| Nombre | RepeatDoUntilEndErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT DO UNTIL**. Debe detectar que falta la expresión de la estructura **repeat do until**. |

| | |
|---|---|
| Diseño | repeat do<br>  putint(1)<br>until<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" cannot start an expression 4..4<br>Compilation was unsuccessful. |

| Nombre | RepeatDoUntilEndErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **REPEAT DO UNTIL**. Debe detectar que falta el **end** de la estructura **repeat do until**. |
| Diseño | repeat do<br>  putint(67)<br>until true |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "end" expected here 4..4<br>Compilation was unsuccessful. |

| Nombre | RepeatDoUntilEndErr3.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT DO UNTIL**. Debe detectar que falta el comando de la estructura **repeat do until**. |
| Diseño | repeat do<br>until true<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "until" cannot start a command 2..2<br>Compilation was unsuccessful. |

| Nombre | RepeatDoUntilEndOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT DO UNTIL** con el objetivo de imprimir un 0. |
| Diseño | repeat do<br>  print(0)<br>until false |

| | |
|---|---|
| | end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | RepeatDoUntilEndOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **REPEAT DO UNTIL** con el objetivo de imprimir un 1. |
| Diseño | repeat do<br>  skip;<br>  skip;<br>  print(1)<br>until false<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

u. Sentencia **INITIALIZED VAR** (estructura nueva)

| Nombre | InitVarErr1.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **INITIALIZED VAR**. Debe detectar que falta la expresión de la estructura **var** (no está asignando nada). |
| Diseño | let<br>  var soyElAnoActual :=<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: "in" cannot start an expression 3..3<br>Compilation was unsuccessful. |

| Nombre | InitVarErr2.tri |
|---|---|
| Objetivo | Reconocer el(los) error(es) sintácticos de la estructura **INITIALIZED VAR**. Debe detectar que falta la expresión y la asignación de la estructura **var** (solo crea la variable). |

| | |
|---|---|
| Diseño | let<br>  var soyElAnoActual<br>in<br>  skip<br>end |
| Resultados Esperados | Error |
| Resultados Observados | ERROR: ":" expected here 3..3<br>Compilation was unsuccessful. |

| Nombre | InitVarOK1.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **INITIALIZED VAR** con el objetivo de crear una variable y asignarle un valor. |
| Diseño | let<br>  var soyElAnoActual  := 2017<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |

| Nombre | InitVarOK2.tri |
|---|---|
| Objetivo | Reconocer la estructura léxica y sintáctica de la estructura **INITIALIZED VAR** con el objetivo de crear tres variables y asignarles un valor. |
| Diseño | let<br>  var soyElAnoAnterior  := 2016;<br>  var soyElAnoActual  := 2017;<br>  var soyElAnoSiguiente := 2018<br>in<br>  skip<br>end |
| Resultados Esperados | Compilación exitosa del código y cumplimiento del objetivo. |
| Resultados Observados | Compilation was successful. |