

# CPLP - EMT 1 - EMTs viejas

A.- Realice en EBNF la gramática de una dirección de email. Considere todas las posibilidades de formato x ej  
unnombre@uno.dos.tres.cuatro, o  
otronombre@gmail.com

B.- De un ejemplo en pascal like de un identificador cuyo alcance sea mayor que su tiempo de vida.

C.- Describa los atributos r-valor y l-valor de las variables

A.-  $G = (N, T, S, P)$

$N = \{ \langle \text{email} \rangle, \langle \text{cadena} \rangle, \langle \text{letra} \rangle, \langle \text{digito} \rangle, \langle \text{nombre} \rangle, \langle \text{dominio} \rangle \}$

$T = \{ a, \dots, z, A, \dots, Z, @, ., 0, \dots, 9 \}$

$S = \{ \langle \text{email} \rangle \}$

$P = \{$

$\langle \text{email} \rangle ::= \langle \text{nombre} \rangle @ \langle \text{dominio} \rangle$

$\langle \text{nombre} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle | \langle \text{digito} \rangle)^* \}$

$\langle \text{dominio} \rangle ::= \langle \text{cadena} \rangle \{ . \langle \text{cadena} \rangle \}^+$

$\langle \text{cadena} \rangle ::= \{ \langle \text{letra} \rangle \}^+$

$\langle \text{letra} \rangle ::= (a | \dots | z | A | \dots | Z)$

$\langle \text{digito} \rangle ::= (0 | \dots | 9)$

$\}$

B.- En el caso del valor apuntado por un puntero. Antes de hacerse un new() este valor no tiene tiempo de vida pero si alcance:

Program prueba;

type = pi: ^integer;


```

var
    puntero: pi;
begin
    writeln('Acá puntero^ tiene alcance pero no tiene tiempo de vida');
    new(p);
    writeln('Acá puntero^ tiene alcance y tiempo de vida');
end.

```

C.- R-valor: es el valor codificado de una variable. Este es el valor que se encuentra en una celda de memoria direccionada por el l-valor de la variable. Este valor se encuentra en binario. La interpretación de su valor varía dependiendo el tipo de la variable.

L-valor: es la dirección en memoria donde se encuentra el valor de la variable. Es el área de memoria ligada a la variable durante la ejecución.

|                                                                                                                                      |                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pregunta</b>                                                                                                                      |                                                                                                                                                                                                                                                                                              |
| <pre> program ideone; ... var while:integer; ... begin ...     cad:="hola";     while:=20/2;     writeln(while,cad); ... end. </pre> | <p>¿En el proceso de compilación del siguiente programa escrito en Pascal, qué tipo de error podría llegar a detectar el compilador, en la sentencia (while:= 20/2). Especifique el error que podría detectarse y, en qué proceso del compilador se detectaría. Justifique la respuesta.</p> |
| <b>Respuesta</b>                                                                                                                     |                                                                                                                                                                                                                                                                                              |
|                                                  |                                                                                                                                                                                                                                                                                              |

El error que podría detectarse es el hecho de que se está haciendo uso de la palabra reservada while como identificador de una variable. Este error se detectaría en la etapa de análisis del programa fuerte, más concretamente en la etapa de análisis sintáctico, en la que interviene el parser.

## Pregunta

Sea el el siguiente ejemplo escrito en dos lenguajes distintos, indique las diferencias en cuanto a la sintaxis concreta y pragmática y determine si la sintaxis abstracta es coincidente o no lo es.

```
while a < n do  
  begin  
    a:=b-1;  
    b:= a+b;  
  end;
```

```
while a < n  
  a = b-1  
  b = a+b
```

Sintaxis concreta:

- En el primer caso se requiere de las palabras claves 'do', 'begin' y 'end;' para delimitar el bloque de código referente al while, mientras que en el segundo caso, el bloque de código referente al while se delimita usando indentación.
- En el primer caso son necesarios los ';', mientras que en el segundo caso no lo son.
- En el primer caso para realizar una asignación se usa ':=' mientras que en el segundo caso se usa '='.

Sintaxis pragmática:

- Las asignaciones en el segundo caso son más fáciles de leer ya que solo utilizan el caracter '=' (en el primer caso se usa ':=').
- El hecho de delimitar el bloque de código referente al while usando las palabras claves 'do', 'begin' y 'end;', hace que sea más fácil saber dónde comienza y acaba dicho bloque de código en el primer caso, que en el segundo caso, en el cual se usa solamente la indentación.

Sintaxis abstracta: la sintaxis abstracta es coincidente ya que en los dos casos lo que se hace es un loop mientras el valor de la variable 'a' sea menor que el valor de la variable 'n', y dentro del bloque se le asigna a la variable 'a' el valor de la variable 'b' menos 1 y luego se le asigna a la variable 'b' el valor de la variable 'a' menos el valor de la variable 'b'.

## Pregunta

Lea el siguiente fragmento de texto sacado de un libro de un lenguaje de programación XX y diga qué característica del mismo le hace pensar en alguno de los criterios de evaluación de lenguajes. Justifique la respuesta. ¿Qué otro concepto podría tenerse en cuenta para evaluar este criterio?

```
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops! That was no valid number. Try again...")
... 
```

La sentencia try funciona de la siguiente manera.

- Primero, se ejecuta la cláusula try (la(s) línea(s) entre las palabras reservadas try y la except).
- Si no ocurre ninguna excepción, la cláusula except se omite y la ejecución de la cláusula try finaliza.
- Si ocurre una excepción durante la ejecución de la cláusula try, se omite el resto de la cláusula. Luego, si su tipo coincide con la excepción nombrada después de la palabra clave except, se ejecuta la cláusula except, y luego la ejecución continúa después del bloque tryexcept.
- Si ocurre una excepción que no coincide con la indicada en la cláusula except se pasa a los try más cercanos; si no se encuentra un gestor, se genera una unhandled exception (excepción no gestionada) y la ejecución se interrumpe con un mensaje como el que se muestra arriba.

## Respuesta

Tiene manejo de excepciones por lo que aumenta la confiabilidad del lenguaje.

Es legible, puesto que se puede comprender lo que hace el programa solamente con ver el ejemplo.

Es simple, ya que se pudo expresar lo que el programador quería en pocas líneas de código concisas (y no por ello poco legibles).

Presenta un gran nivel de abstracción, esto se debe a que las instrucciones usadas permiten al programador obviar detalles de la implementación de las mismas. Por ejemplo en el caso de la instrucción input se entiende que permite al usuario ingresar un número, antes mostrando un mensaje pasado como argumento.

## [EMT 1 - 2023 \(agusnfr\)](#)

La característica del lenguaje que me hace pensar que el mismo no es ortogonal es el hecho de que la función no pueda devolver arrays o funciones. Esto implicaría que el lenguaje no sea ortogonal ya que por más que el lenguaje permita trabajar con arrays y funciones, no se pueden devolver en una función.