

Práctica 3

- 1) La semántica se refiere al significado/interpretación de las instrucciones en un programa.
- 2) Compilar significa traducir el código fuente de un program lenguaje de alto nivel a un lenguaje de bajo nivel que puede ser comprendido y ejecutado x una computadora.

Los pasos para compilar un programa son divididos en 2 etapas:

• Etapa de análisis de código fuente: incluye ciertas etapas

- 1) Análisis léxico: divide el cód. en tokens q' son componentes básicos como words, elimina comentarios y espacios en blanco.
- 2) " sintáctico: verifica que los token concidan con la sintaxis. y que cumpla con las reglas gramaticales.
- 3) " semántico: " cumpla con la semántica, tipos, asignaciones, uso adecuado de variables y llamadas a fx

• Etapa de síntesis: produce el cód. objeto. Se hace lo optimizar, asignar de registros.

La semántica interviene en el análisis semántico, la importancia radica en que si no se cumplen las reglas semánticas, el programa no podrá ser ejecutado.

3) No, no es lo mismo. Diferencias:

	Intérprete	Compilador
Ejecución	Ejecuta el programa de entrada	Compila y ejecuta el cód. objeto.
Orden	Sigue orden lógico	Sigue orden físico.
Tiempo de ejecución	Lento	Rápido en el pto. de vista de MW
Eficiencia	P/ r/ sentencia hace proc. de decodificar	Pasa x todas las sentencias
Espacio	- memoria.	
Errores.	Como uso el cód. fuente, es + fácil detectarlos.	Se pierde referencia p/ cód. fuente y cód. objeto.

- 4) • Error sintáctico: una intrux no sigue la sintaxis del lenguaje. Son detectados en el análisis semántico en la etapa de compilax.
 ej:

```
if (num1 < num2) {  
    ///  
}
```

 ↘ falta cierre de paréntesis.

• Error semántico: una intrux es gramaticalmente correcta pero no tiene sentido en términos de lógica
 ej:

```
int resultado = num1 / 0;
```

 ↘ se divide x 0, no tiene sentido.

5) Sintácticos

a)

Línea 2: "var s: integer", las variables deben comenzar con una letra.

Línea 5: "for i:=5 to 10 do begin" la variable i nunca fue declarada.

Semántico.

Línea 4: "var a: char" se declara como char pero luego se le realiza una operación de suma.

Línea 7: "a=a+1" la operación de asignar es ":=".

b) Sintáctico.

Línea 0: el nombre de la clase no está declarado.

Línea 1: "arrayList" debe cambiarse a "ArrayList".

Línea 4: variable i no declarada.

Línea 4: i-- debería ser i++ porque nunca llegará a 11.

Semántico.

Línea 3: result debería inicializarse como result="" porque sino no podrá concatenar.

Línea 4: i-- debe ser i++

Línea 6: quiere asignar un valor cuando se está obteniendo un valor de la lista.

En caso de querer asignar algo sería "listado.set(listado.size()-1; numero > 1)"

Lógico

Línea 8: quiere retornar un booleano cuando el valor de retorno es String.

c) Sintáctico - Línea 11: cuadrado() no especifica valor de retorno.
- Debe incluirse la función antes de la primera llamada o sea sobre man()

Semántico: Línea 11: cuadrado() no incluye apertura de llave {}

Línea 14: "numero_cuadrado := numero * numero" en caso de hacer una asignación debe ser ":="

Línea 14: "numero_cuadrado" no se declaró el tipo.

Línea 11: no declara el tipo del parámetro numero.

Línea 11: no se declara tipo de retorno.

Línea 12: ~~numero~~ numero se declara dos veces.

d) Sintácticos: Línea 2: a print le falta "()

Línea 8: ~~idem~~ línea 2

Semánticos: Línea 7: variable N no declarada.

Línea 7: mod no corresponde a Python, debería ser %.

Línea 10: p/ elevar valor debe usarse "xx" y no "A"

e) Semánticos: Línea 1 "puls" en minúscula.

Línea 3: comilla de más en "Juan"

Línea 4: edad no se definió previa

Línea 5: cadena variable sin antes parsearla.

Ejercicio 5 II

Pascal

```

procedure ordenar_arreglo (var: arreglo: arreglo de caracteres,
    cant: integer);
var
    i: integer; ordenando: boolean;
    aux: char;
begin
    repeat
        ordenando := true;
        for j := 1 to cant-1 do
            if ord(arreglo[j]) > ord(arreglo[j+1])
            then begin
                aux := arreglo[j];
                arreglo[j] := arreglo[j+1];
                arreglo[j+1] := aux;
                ordenando := false;
            end;
        until ordenando;
    end;
end;

```

Java

```

public void ordenar_arreglo (ArrayList
    <char> arreglo, int cant) {
    int i;
    boolean ordenado;
    char aux;
do
    ordenado = false;
    for (i = 1; i < cant; i++) {
        if (ord(arreglo[i]) > ord(arreglo[i+1])) {
            aux = arreglo[i];
            arreglo[i] = arreglo[i+1];
            arreglo[i+1] = aux;
            ordenado = false;
        }
    }
    while (!ordenado);
}

```

Ejercicio 6.

self variable que se refiere al objeto actual o receptor del mensaje. Su semántica es la de una referencia al objeto actual en el que se encuentra el código en ese momento. Se puede usar p/ llamar a los métodos de instancia del obj. actual o p/ acceder a variables de instancia.

nil: variable predefinida que se refiere a un objeto único que representa la ausencia de valor o la no existencia de valor un objeto. Es el equivalente a null en otros lenguajes. Se usa en Ruby para representar el resultado de una operación que no tiene valor de retorno válido.

Ejercicio 7 **null** indica explícitamente que una variable o propiedad no tiene un valor, o sea, se ha asignado intencional el valor null

undefined: indica que en a una variable o propiedad no tiene asignado un valor

Ejercicio 8

C = usada en un bucle for/while/do-while para salir del bucle antes de cumplir condic. de finalizar, dentro de la estructura "switch" p/ salir de la estructura desp. de dtda. condic. / al ejecutar la sentencia break, la ejecuc. del programa se f. mueve fuera del bucle y continúa c/ la instruc. siguiente.

PHP idem C
JavaScript
Ruby.

Ejercicio 9 El binding establece una relación entre un identificador y su valor dentro del programa. La importancia radica en que permite al programa referirse a los valores mediante nombres significativos para el programador, con mayor facilidad y comprensión.

- Binding estático: hecho en la etapa de compilación. El identificador se asocia con una dirección de memoria fija y no cambiará durante la ejecución.

```
int a=5; }  
int b=a+3;
```

- Binding dinámico: hecho en tiempo de ejecución. El identificador se asocia con una dirección de memoria que puede cambiar durante la ejecución.

```
x="hola";  
def imprimir();  
  print(x);  
}  
x="adiós"
```