Guía de Actividad 4.1

Identificación

Asignatura: Estructura de los Lenguajes
Unidad: Identificadores, enlaces, chequeo de tipos y ámbito

Objetivos

- Implementar programas que permitan interactuar con conceptos estudiados en la unidad

Plazo

Fecha y hora de entrega: *indicada en la plataforma.*

Evaluación

La actividad será evaluada por el método de calificación simple, en una escala de 0 a 100, con las siguientes consideraciones:

a. Se entrega el conjunto de fuentes completo en un archivo ZIP	5%
b. Se respetan las convenciones de formato (indentado, comentado)	5%
c. Se entrega el proyecto sin retraso alguno	5%
d. Todos los programas entregados compilan sin errores	5%
e. Se resuelven correctamente los ejercicios	80%

Actividad

En esta actividad, se debe desarrollar la solución a ejercicios indicados a continuación, todas las soluciones debe incluirlas en un solo archivo ZIP para la entrega.

Adjunto a esta guía instruccional, encontrará un archivo ZIP con código fuente preliminar para ejecutar esta actividad, descargue el archivo y luego de descomprimirlo, continúe con la actividad:

Ejercicio 1: Manejo de Direcciones en Memoria
Para entregar: archivo fuente dangling.c corregido

Considerando el archivo fuente proveído llamado *dangling.c,* termine el programa propuesto, el cual calcula la diferencia en días entre 2 fechas.

Por simplicidad asuma que un mes tiene siempre 30 días y un año 365 días.

Adicionalmente, verifique si en el código se produce alguna referencia pendiente (dangling references), si es así, corrija el problema.

Si ejecutamos el programa se genera inicialmente este resultado:

```
Lectura de una fecha:
        Ingrese dia: 6
        Ingrese mes: 3
        Ingrese anio: 2020

        Fecha leida en funcion leerFechaDelTeclado:  6/3/2020
Lectura de una fecha:
        Ingrese dia: 5
        Ingrese mes: 7
        Ingrese anio: 2021

        Fecha leida en funcion leerFechaDelTeclado:  5/7/2021

Fechas sobre las que se calculara la dif en funcion main:  -411498448/32766/-1724841898  -411498448/32766/-1724841898
```

Ejercicio 2: Sobrecarga de Operadores en C++
Para entregar: solución escrita en archivo fuente cpp

El ejemplo presentado en el archivo adjunto *Complejo.cpp*, se muestra como en C++ se puede sobrecargar operadores.

Escriba un programa en C++ que permita manipular NumerosRacionales con todos sus operadores aritméticos básicos sobrecargados. Un NumeroRacional es un nro. que representa la relación entre 2 enteros A y B, tal que B no pueda ser cero. Ej. NroA = ⅗, NroB = ⅘

Y se pueden realizar estas operaciones mínimas: suma, resta, producto, division.

Muestre con un main de prueba su correcto funcionamiento.

Ejercicio 3: Manejo de Sobrecarga en ADA vs Java

Este ejercicio no se entrega

Usando un intérprete online de ADA, transcriba el código del archivo *overload.adb*. Ejemplo: https://www.jdoodle.com/execute-ada-online/

Ejecute y verifique que funciones son invocadas

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure overload is
function max(x: integer; y: integer) -- max #1
return integer is
begin
    Put("Max v1 invocado..."); new_line;
    if x > y then return x;
    else return y;
    end if;
end max;

function max(x: integer; y: integer) -- max #2
return float is
begin
    Put("Max v2 invocado..."); new_line;
    if x > y then return float(x);
    else return float(y);
    end if;
end max;

a: integer;
b: float;
begin -- max_test
a := max(2,3); -- cual version de max llamara?
b := max(2,3); -- cual version de max llamara?
```

Si se escribe un programa equivalente en Java, qué resultado arrojará? reflexione el porqué se daría esto:

```
public class overload {

    public static int max(int a, int b) {
        System.out.println("Max v1 invocado...");
        if (a > b) return a;
        else return b;
    }

    public static double max(int a, int b) {
        System.out.println("Max v2 invocado...");
        if (a > b) return a;
        else return b;
    }

     public static void main(String []args){
        int a;
        double b;

        a = max(2,3);
        b = max(2,3);
     }
}
```

Ejercicio 4: Scoping estático/dinámico en Perl

Este ejercicio no se entrega

Usando un intérprete online de Perl, transcriba el código de la figura. Ejemplo: https://www.tutorialspoint.com/execute_perl_online.php

Ejecute y verifique si la resolución de scoping es estática o dinámica.

```
1  $x = 10;
2  sub f {
3      return $x;
4  }
5
6  sub g {
7      local $x = 20;
8      return f();
9  }
10
11 print g()."\n";
```