Trabajos Practicos

Paradigmas de Programación (Universidad Nacional de Tucumán)

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 1 Fecha: 23/08/17

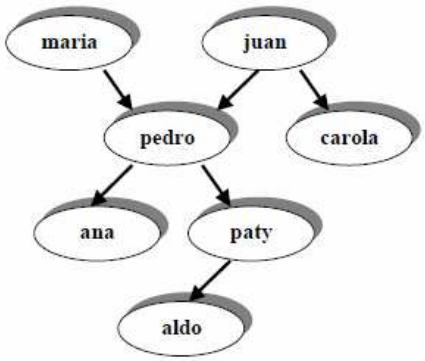
Tema: Programación Lógica - Conceptos básicos de PROLOG

Todos los parámetros facilitados a los predicados en Prolog deben comenzar por una letra minúscula y no deben contener espacios

COMANDOS:
- `listing.` : realiza un listado de las cláusulas consultadas.
- `write('…').` : escribe en el flujo de salida.
- `nl ó '\n'` : escribe una nueva línea en el flujo de salida.
- `change_directory('…').` : cambia el directorio actual.
- `consult(nombre_archivo).` : compila y carga en memoria la información del archivo.
- `halt.` : termina una sesión Prolog.
- `%` : sirve para crear comentarios en Prolog.

Ejercicios

Teniendo en cuenta la guía anterior, resuelva los siguientes ejercicios.



1. Dada la relación *progenitor* que posee los siguientes hechos:

```
progenitor( maria, pedro ).
progenitor( juan, pedro ).
progenitor( juan, carola ).
progenitor( pedro, ana ).
progenitor( pedro, paty ).
progenitor( paty, aldo ).
```

a) Indique que responde Prolog a las siguientes preguntas:

```
?- progenitor(pedro, X).
?- progenitor(X, pedro).
?- progenitor(maria, X), progenitor(X, paty).
?- progenitor(juan, X), progenitor(X, Y), progenitor(Y, aldo).
```

En cada caso, analice como Prolog resuelve la consulta.

b) ¿Cómo formularía las siguientes preguntas en Prolog?
- ¿Quién es el progenitor de Paty?
- ¿Tiene Pedro algún hijo?
- ¿Quiénes son los abuelos de Aldo?

c) Agregar las siguientes cláusulas:

```
femenino(maria).
femenino(carola).
femenino(ana).
femenino(paty).

masculino(juan).
masculino(pedro).
masculino(aldo).
```

Teniendo en cuenta que las relaciones padre e hijo se pueden expresar con las siguientes reglas en Prolog:

```
padre(Y, X) :- progenitor(Y, X), masculino(Y)
hijo(X, Y) :- progenitor(Y, X), masculino(X)
```

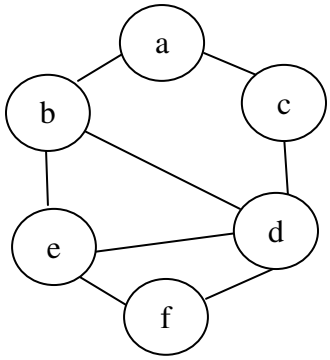
PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 1 Fecha: 23/08/17

Expresar en término de reglas las relaciones:

- madre
- hija
- abuelo/abuela
- hermanos
- primo/prima
- tío/tía

2.Dado el siguiente grafo:



a). Definir los asertos necesarios para declarar el grafo de la figura en Prolog. Para ello, tenga en cuenta la relación `arco(v1, v2)`, donde v1 y v2 corresponden a los vértices extremos del arco.

b). Formular las siguientes preguntas en Prolog:

i. ¿Existe arco entre **b** y **c**?

ii. ¿Cuales son los vértices alcanzables desde el vértice **b**?

c). Expresar en término de reglas de Prolog las relaciones de:

- Adyacencia entre dos vértices.
- Camino de longitud 2 entre dos vértices dados.
- Camino de cualquier longitud entre dos vértices.

3. Dentro de la siguiente base de datos se desea encontrar a todas las personas que están en condiciones de acceder a la jubilación.

a) Dada la siguiente tabla con la información de clientes, escriba en Prolog las reglas necesarias para llevar a cabo la tarea:

Nombre	Edad	Estado Civil
Luis	20	Soltero
Juan	72	Viudo
Pedro	40	Casado
Julia	17	Casada
María	46	Soltera
Andrés	50	Casado
Rocío	67	Soltera

Nombre	Edad	Estado Civil
Joaquín	15	Soltero
Cecilia	35	Soltera
Felicia	60	Soltera
Santiago	45	Casado
Verónica	34	Casada
Eugenia	70	Viuda
Carlos	73	Casado

b) Escriba una regla en Prolog que le permita establecer la relación “X es mayor que Y”.

4. ASESINATO EN LA MANSION DREADBURY (Pelletier F.J. - 1986)

- Un asesino siempre odia a su víctima y nunca es más rico que ella. Además, el asesino vive en la mansión Dreadbury.
- Tía Agatha, el mayordomo y Charles son las únicas personas que viven en la mansión Dreadbury.
- Charles odia a todas las personas de la mansión que no son odiadas por la tía Agatha.
- Agatha odia a todos los que viven en la mansión, excepto al mayordomo.
- Quien no es odiado por el mayordomo y vive en la mansión, es más rico que tía Agatha
- El mayordomo odia a las mismas personas que odia tía Agatha.

i) Realizar las siguientes consultas:
   a. ¿Quién es el asesino de la tía Agatha?
   b. ¿A quién odia Charles?
   c. ¿Quién odia a Agatha?
   d. Todos los que odian y sus odiados.
   e. ¿Quién es el potencial asesino del mayordomo?

ii) Mostrar las consultas utilizadas para conseguir lo anterior, junto con las respuestas obtenidas.

5. a. Implemente en PROLOG la siguiente especificación algebraica del ADT LISTA(item).

Sintaxis:
LISTAVACIA: → LISTA
INSERTAR: LISTA x item → LISTA
ESLISTAVACIA: LISTA → bool
BORRAR: LISTA → LISTA
PRIMERO: LISTA → item U {indefinido}
ULTIMO: LISTA → item U {indefinido}
PERTENECE: LISTA x item → bool

Semántica: ∀ L Є LISTA, ∀ x, k Є item
ESLISTAVACIA(LISTAVCIA) ≡ true
ESLISTAVACIA(INSERTAR(L, x)) ≡ false

BORRAR(LISTAVACIA) ≡ LISTAVACIA
BORRAR(INSERTAR(L, x)) ≡ L

PRIMERO(LISTAVACIA) ≡ indefinido
PRIMERO(INSERTAR(L, x)) ≡ x

ULTIMO(LISTAVACIA) ≡ indefinido
ULTIMO(INSERTAR(L, x)) ≡ SI ESLISTAVACIA(L) ENTONCES
x
SINO
ULTIMO(L)

PERTENECE(LISTAVACIA, k) ≡ false
PERTENECE(INSERTAR(L, x), k) ≡ SI x = k ENTONCES
true
SINO
PERTENECE(L, k)

b. Agregue la operación longitud, que cuente la cantidad de elementos que hay en una lista.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 2 Fecha: 30/08/17

Tema: Programación funcional. Recursión. Funciones high order. Haskell.

1. Explique qué hace el siguiente código Haskell y de un nombre más adecuado a la función.

```
misterio  x y
    | (x < y)   =  0
    | otherwise =  1 + misterio (x - y) y
```

2. Implemente una función que, dados a y b calcule a^b.

3. Escriba en Haskell una función que cuente cuantas veces se repite un elemento en una lista.

4. Implemente una función que devuelva una lista formada por el segundo argumento tantas veces como lo indique el primer argumento. Por ejemplo:

```
> replicar 1 3
[3]
> replicar 3 1
[1,1,1]
> replicar 0 2
[]
> replicar 3 "hola"
["hola", "hola", "hola"]
>replicar (-2) "chau"
[]
```

5. Implemente una función que permita retornar los primeros n elementos de una lista. Por ejemplo:

```
> retener -3 ['h','a','s','k','a']
[]
> retener 0 ['h','a','s','k','a']
[]
> retener 3 []
[]
> retener 3 ['h','a','s','k','a']
['h', 'a', 's']
> retener 42 ['h','a','s','k','a']
['h', 'a', 's', 'k', 'a']
```

6. Escriba una función que reciba otra función como primer parámetro y un segundo parámetro que actúe como argumento de la función recibida. Defina la función para que el resultado sea una doble aplicación de la función pasada por parámetro con el segundo argumento.

7. Escriba una función que combine dos listas aplicando otra función que reciba como parámetro un elemento de cada lista a combinar por vez. Utilice la siguiente definición de tipo para su función:

```
combinarCon :: (a -> b -> c) -> [a] -> [b] -> [c]
```

8. En un párrafo explique qué hace el siguiente código Haskell y dé un nombre más adecuado a la función. También explique cómo se usa dicha función mediante un ejemplo y muestre la aplicación de la función en cada paso de recursión.

```
misterio2 :: Int -> [a] -> ([a], [a])
misterio2 _ [] = ([], [])
misterio2 n l@(x : xs)
    | n > 0     = (x : ys, zs)
    | otherwise = ([], l)
    where (ys,zs) = misterio2 (n - 1) xs
```

9. Defina una función **m** que reciba como parámetros una función **f** de un argumento y una lista y devuelva como resultado la lista recibida en la que cada uno de sus elementos haya sido transformado con la función **f**.

10. Explique qué hace la siguiente función. Rescribirla con nombres de identificadores significativos.

```
f :: (a -> Bool) -> [a] -> [a]
f _ [] = []
f p (x:xs)
    | p x = x : f p xs
    | otherwise = f p xs
```

11. Estudie cómo se aplica la función foldl en Haskell y escriba una versión propia.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 3 Fecha: 06/09/17

Tema: Diseño Orientado a Objetos. UML

En los siguientes apartados identifique las clases que intervienen en el problema, las relaciones entre clases, las responsabilidades de cada clase y diseñe el diagrama de clases del problema.

1) Se desea diseñar un tipo de dato HORA que permita mantener información sobre horas, minutos y segundos. La clase debe responder a los mensajes que soliciten dicha información. Debe ser posible crear objetos del tipo hora con una hora determinada por el usuario, incrementar una hora con otra dada, escribir de forma adecuada objetos del tipo HORA. El tipo de dato HORA es responsable de mantener la validez de sus datos.

2) Se desea modelar un sistema para el manejo interno de los productos de una farmacia. Dichos productos pueden ser medicamentos o cosméticos. De cada medicamento se sabe el nombre, el laboratorio y el precio base (sin el IVA). De los productos cosméticos se tiene la misma información que de los medicamentos. Para el cálculo del IVA, se debe tener en cuenta que a los medicamentos les corresponde el 10.50 %, mientras que a los productos cosméticos el 21 %. Muestre en el diagrama de clases cómo sería cálculo del precio de los productos.

3) Se desea modelar una Empresa con empleados. Una empresa conoce a todos sus empleados, y estos pueden ser de planta permanente o temporaria. Cuando un empleado es de planta permanente cobra la cantidad de horas trabajadas por $30, más antigüedad ($10 por año de antigüedad), más salario familiar. Cuando es de planta temporaria, no cobra antigüedad y cobra la cantidad de horas trabajadas por $20, más salario familiar. El salario familiar es $20 por cada hijo, los empleados casados además cobran $10 por su esposa/o. En el diagrama modele el método *montoTotal* en la clase Empresa, que retorna el monto total que la empresa debe pagar en concepto de sueldos a sus empleados.

4) El Gobierno de Tucumán necesita realizar un censo de todas las salas de Jardín de 5 años que funcionan en la provincia para adoptar medidas de crecimiento de acuerdo a las necesidades de la población. De cada sala se necesita saber la localidad a la que pertenece, su capacidad máxima, información del docente a cargo de la misma, alumnos inscriptos para el próximo año, información de la institución a la que pertenece la sala como ser el nombre de la institución, si es pública o privada, información sobre el director de dicha institución, etc. Con la información recavada, el gobierno necesita determinar cuántas vacantes se registran por localidad.

5) Se desea modelar un sistema para una empresa telefónica a efectos de administrar su facturación. De cada línea telefónica se tiene el número y la información relativa al consumo en forma de llamadas realizadas. Las llamadas pueden ser nacionales o internacionales. De cada llamada se tiene la fecha y la duración en minutos. Cada línea se registra a nombre de un cliente. De un cliente se sabe el nombre y dirección. De las llamadas se tiene el precio por minuto. Este precio depende del tipo de llamada. Los valores son:
   a. Llamada Nacional 0.30
   b. Llamada Internacional 0.70

   En el diagrama de clases modele además el método que realiza el cálculo del monto a abonar de una línea telefónica. El monto se calcula como el abono básico de la línea -común a todas las líneas telefónicas- más el costo de las llamadas realizadas.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 4 Fecha: 25/09/17

Tema: Clases en C++

1)

i. Agregue a la clase HORA diseñada en el práctico anterior un constructor por defecto y un constructor copia e implemente la misma en C++. Para ello:

a. Escriba un archivo .h con la definición de la clase.
b. Escriba un archivo .cpp con el desarrollo de cada una de las operaciones de la clase.
c. Escriba un programa de prueba de su tipo de dato HORA.

ii. Como usuario de la clase HORA, **sobrecargue el operador de inserción de flujo (<<)** para escribir la hora en formato de 12 hs (AM/PM). Pruebe la misma.

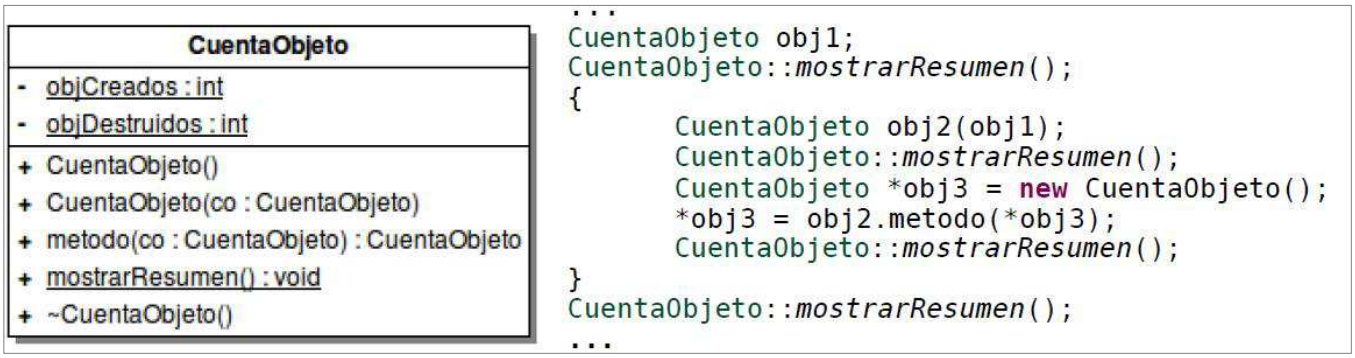
2)

i. Crear una clase Cuenta que mantenga la información sobre una determinada cuenta bancaria (n° de cuenta, titular, saldo). Agregar las operaciones necesarias para:

- **Crear** cuentas vacías y cuentas con un determinado monto inicial
- **Depositar** dinero en la cuenta
- **Extraer** dinero de la cuenta.
- **Determinar** cuál es el saldo de la misma.

ii. Como usuario de la clase Cuenta implemente una función que permita realizar la transferencia de dinero entre dos cuentas.

3) Implemente una clase que sea capaz de registrar la cantidad de objetos instanciados, la cantidad de objetos destruidos, y la cantidad de objetos vivos. Tome como referencia la clase del diagrama, pruebe el código a continuación y saque conclusiones.



OBS:

- Los datos y funciones miembros subrayados son miembros de clase (static).
- El método mostrarResumen(), en el momento de su invocación, debe mostrar por pantalla la cantidad de objetos creados, destruidos, y almacenados en memoria.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 5 Fecha: 02/10/17

Tema: Clases en C++. Asociación, Composición y Agregación

1) Se desea modelar el Ticket de los estacionamientos. Cada vez que ingresa un vehículo al estacionamiento se emite un ticket con un número que no se repite, la información del vehículo y la hora de ingreso. Cuando el vehículo se retira del estacionamiento, el sistema debe calcular el importe que debe pagar en función del tiempo de estadía en el lugar. El calculo de dicho importe se realiza de la siguiente manera: se paga por la primera hora el importe completo (haya o no permanecido la hora) y a partir de ahí se cobra el excedente fraccionando el importe cada 15 minutos. Diseñe una solución al problema, implemente y pruebe la misma.

2) Teniendo en cuenta el problema 4 del TP3, implemente su diseño y pruebe la solución obtenida.

3) Modifique el diseño de la clase Cuenta desarrollado en el TP4 para que el titular de la cuenta corresponda a un objeto del tipo persona. Agregue a su diseño la clase Banco. Tenga en cuenta que un Banco puede, entre otras operaciones:

- Crear cuentas asignándole un número que no debe repetirse en ningún momento.
- Dar de baja una determinada cuenta.
- Depositar y extraer dinero a una determinada cuenta.
- Transferir dinero entre dos cuentas del mismo banco.
- Imprimir el resumen de una determinada cuenta (datos de la cuenta y saldo) dado su número o de todas las cuentas que contiene dicho banco (operaciones homónimas).
- Obtener la suma total de los saldos de las cuentas que posee el banco.

Implemente su diseño, pruebe que las clases funcionan correctamente y escriba un programa principal en el cual deberá probar la creación de personas y objetos del tipo Banco. Crear cuentas, realizar depósitos, extracciones y transferencias a través de un objeto Banco y presentar el resumen completo por pantalla.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 6 Fecha: 20/10/17

Tema: Herencia, clases abstractas y polimorfismo.

1) Teniendo en cuenta los diseños realizados en el TP3 para los siguientes problemas:

   A. Se desea modelar un sistema para el manejo interno de los productos de una farmacia. Dichos productos pueden ser medicamentos o cosméticos. De cada medicamento se sabe el nombre, el laboratorio y el precio base (sin el IVA). De los productos cosméticos se tiene la misma información que de los medicamentos. Para el cálculo del IVA, se debe tener en cuenta que a los medicamentos les corresponde el 10.50 %, mientras que a los productos cosméticos el 21 %. Muestre en el diagrama de clases cómo sería cálculo del precio de los productos.

   B. Se desea modelar una Empresa con empleados. Una empresa conoce a todos sus empleados, y estos pueden ser de planta permanente o temporaria. Cuando un empleado es de planta permanente cobra la cantidad de horas trabajadas por $30, más antigüedad ($10 por año de antigüedad), más salario familiar. Cuando es de planta temporaria, no cobra antigüedad y cobra la cantidad de horas trabajadas por $20, más salario familiar. El salario familiar es $20 por cada hijo, los empleados casados además cobran $10 por su esposa/o. En el diagrama modele el método *montoTotal* en la clase Empresa, que retorna el monto total que la empresa debe pagar en concepto de sueldos a sus empleados.

   a. Identifique clases abstractas y clases concretas, funciones virtuales y funciones virtuales puras.
   b. Modifique sus diseños para que reflejen estas características.
   c. Implemente el diseño obtenido.
   d. Escriba un programa que cree objetos de las distintas clases y pruebe cada una de ellas.
   e. Identifique donde se espera obtener un comportamiento polimórfico dentro de su programa.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 6 (continuación) Fecha: 25/10/17

Tema: Herencia, clases abstractas y polimorfismo.

1) En el siguiente problema identifique clases, relaciones entre clases, responsabilidades de cada clase y modele una solución del problema.

Un estacionamiento desea automatizar el registro diario de vehículos para simplificar la tarea del administrador. En una jornada de trabajo, cada vez que ingresa un vehículo al estacionamiento se emite un Ticket con un número único de identificación por día, información del vehículo y hora de ingreso. Cuando el vehículo se retira del estacionamiento, el sistema debe calcular el importe que debe pagar en función del tiempo de estadía en el lugar y el tipo de vehículo. Si el vehículo es:

- **Motocicleta**: se cobra por hora la mitad de la tarifa básica y el monto a pagar corresponde a la estadía en el lugar teniendo en cuenta que se fracciona el tiempo cada media hora.
- **Automóviles**: se cobra por hora la tarifa básica completa y se paga por la primera hora el importe completo (haya o no permanecido la hora) y a partir de ahí se cobra el excedente fraccionando el importe cada 15 minutos.
- **Camionetas**: se cobra por hora 1.5 de tarifa básica y se paga por hora, sin fraccionar el tiempo de estadía.

Cada día se inicia una nueva jornada de trabajo en el estacionamiento. De cada Jornada se conoce la fecha, los tickets emitidos durante la misma, la tarifa básica de estacionamiento por hora, la capacidad máxima de estacionamiento disponible ese día y el monto recaudado al momento de realizar la consulta. Al finalizar la jornada, se debe realizar un cierre de caja. Cuando se solicita el cierre de caja, el estacionamiento debe estar vacío o se debe realizar la salida de todos los vehículos que queden en el lugar.

Su trabajo consiste en crear las clases necesarias que permitan obtener la información del monto recaudado por el estacionamiento en una determinada jornada de trabajo.

** El monto recaudado por una jornada de trabajo corresponde a la sumatoria de los importes correspondientes a los tickets emitidos durante la misma. Este valor se actualiza con el importe a pagar de cada vehículo que se retira del estacionamiento.*

2) Implemente su diseño en C++.

3) Escriba un programa principal en el que deberá probar la creación de objetos según su diseño, y realizar todas las pruebas necesarias para determinar que las operaciones de las distintas clases funcionan correctamente.

PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 7 Fecha: 30/10/17

Tema: Plantilla de clases, tipos genéricos, STL.

1) Implemente una plantilla de clase que almacene elementos de un tipo genérico T en un vector generado dinámicamente. La plantilla de clase debe contar con las operaciones necesarias para:

- Construir vectores con el tamaño suministrado por el usuario de la clase, o de tamaño 10 en su defecto.
- Construir un vector como copia de otro vector dado.
- Seleccionar la capacidad del vector.
- Seleccionar un elemento del vector dada su posición.
- Insertar un elemento al principio, al final ó en una posición determinada del vector.
- Escribir por pantalla el contenido del vector.
- Destruir un vector.

Escriba un programa de prueba que haga uso de cada método implementado para la clase genérica.

2) Implemente las siguientes plantillas de funciones como usuario de la clase VectorGenérico:

- Función **pertenece** para determinar si un elemento T se encuentra en un vector del tipo T.
- Función **max** que selecciona un máximo elemento T de un vector del tipo T.

Pruebe las plantillas de función implementadas con tipos de datos incorporados en el lenguaje y para los tipos Cuenta y Hora de trabajos prácticos anteriores. Haga las modificaciones necesarios para llevar a cabo esta tarea.

3) Realice una especialización explícita de la función miembro escribir para el tipo Cuenta para que, de cada objeto Cuenta incluido en el vector, se escriba su número de cuenta y saldo.

4) Busque información (sintaxis, operaciones, etc.) sobre la plantilla de clase Vector disponible en la Standard Template Library (STL) de C++, e implemente el mismo programa de prueba del ejercicio 1 utilizando objetos de este tipo.