UNIVERSIDAD NACIONAL DE HURLINGHAM

Introducción a la Programación - Práctica 11

Tipos de Datos Personalizados

CONSEJOS:

- Leer el enunciado en su totalidad y pensar en la forma de resolverlo **ANTES** de empezar a escribir código
- Si un ejercicio no sale, se puede dejar para después y continuar con los ejercicios que siguen
- Los ejercicios están pensados para ser hechos después de haber mirado la teórica correspondiente
- Algunos de los ejercicios están tomados de las guías prácticas utilizadas en la materia de Introducción a la Programación de la Universidad Nacional de Quilmes por Pablo Ernesto “Fidel” Martínez López y su equipo. También Federico Aloi y Miguel Miloro, a su vez basada en las guías Ejercicios de Introducción a la Programación del CIU General Belgrano, elaboradas por Carlos Lombardi y Alfredo Sanzo, y Fundamentos de la Programación del Proyecto Mumuki. Agradecemos a todos los que nos ayudaron con su inspiración.
- **Realizar en papel los ejercicios que así lo indiquen.**
- **Sí un ejercicio indica BIBLIOTECA significa que podrá ser utilizado en el parcial sin definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

VARIANTES:

1. Días de la semana

Declarar un tipo variante llamado **`DíaDeLaSemana`**, que sirva para identificar los días de la semana. Luego implementar las siguientes funciones (sin olvidar sus contratos):

a. **`díaSiguiente_`** que dado un día de la semana, devuelve el día siguiente.

b. **`díaPrevio_`** que dado un día de la semana, devuelve el día previo.

c. **`esDíaDeFinDeSemana_`**, que indica si el día dado es uno del fin de semana.

Reflexionamos: ¿De qué tipo es el parámetro dado a cada una de estas funciones? ¿Cuál sería un buen nombre para el mismo? ¿Por qué las funciones siguiente y previo no pueden utilizarse para solucionar este problema? ¿Qué beneficios trae tener ahora díaSiguiente y díaPrevio sí necesito saber algo cómo "en qué día ocurrió un suceso"?

2. Partidos políticos

EN PAPEL Se quiere modelar una serie de partidos politicos de un país, y poder realizar consultas sobre los mismos. Se cuenta entonces con la siguiente tabla de información:

Partido
Democracia por la Verdad
Unidos por la República
Liberales por la Libertad
Izquierda de los Obreros

Se quiere entonces crear un tipo de datos variante **`PartidoPolítico`** que modele a los partidos mencionados. Además se les brinda la siguiente función como primitiva:

```
function cantidadDeVotosDe_(unPartido)
/*
    PROPÓSITO: Indica la cantidad de votos que recibió un partido.
    PARÁMETROS
        * unPartido: PartidoPolítico - El partido político del cual saber
                                       su cantidad de votos.
    TIPO: Número
    PRECONDICIÓN: Ninguna
*/
```

Luego escriba las siguientes funciones:

a. **`tieneMásVotantes_Que_`** que dados dos partidos indique si el primero tiene más votos que el segundo.

b. **`elQueTieneMásVotos`** que describe el partido con más votantes.

c. **`habráBallotage`** que indica si en estas elecciones habrá ballotage. Esto se da cuando el partido con más votos no acumula más del 50% de los votos totales y no hay una diferencia de más del 10% sobre entre el primero y el segundo candidato.

REGISTROS:

3. Jugamos con cartas

Declare el tipo variante **Palo** y el tipo registro **Carta** y escriba las siguientes funciones. No se olvide de ir probando en Gobstones cada función que realiza para saber sí el resultado es correcto.

a. **anchoDeEspadas** que describe la carta 1 de Espadas.
b. **anchoDeBastos** que describe la carta 1 de Bastos
c. **laCarta_de_** que dado un número y un palo que describe la carta con dicho número y dicho palo..
d. **esUnAncho_** que indica si la carta dada es un 1.
e. **esFigura_** que dada una carta, indica si la misma es una figura (las figuras son los 10s, los 11s y los 12s).
f. **esDeOro_** que indica si la carta dada es de Oros.
g. **tieneUnNúmeroMásGrande_Que_** que dadas dos cartas indica si la primera carta tiene un número más grande que la segunda.
h. **sonDelMismoPalo_Y_** que dadas dos cartas, indica si estas tienen el mismo palo.
i. valorParaEnvidoDe_ que describa el número que aporta la carta dada en el canto del envido. El número se corresponde al número de la carta, si la misma no es figura, y cero, si fuera figura.
Atención: Sí ya sabe jugar al truco, tenga en cuenta que se está preguntando el valor que aportaría una única carta, no una jugada de multiples cartas.
j. **mayorValorEntre_Y_** que describe el valor más grande entre dos cartas, según lo que aporta cada una para el envido. Por ejemplo, si las cartas son un 7 y un 12, describe 7, pues el 12 no aporta nada para el envido.
k. **sumaParaElEnvidoCon_Y_** que dadas dos cartas, describe el número que tienen las mismas para el envido. El envido se calcula como la suma los valores del envido de cada carta más 20, si las cartas son del mismo palo, o como el mayor valor entre ellas, cuando son de distinto palo.
l. **sonMejores_Y_Que_Y_** que dadas 4 cartas, donde las dos primeras son las cartas del primer jugador para cantar envido, y las segundas dos las del segundo jugador, indica si el envido del primer jugador es más grande que el envido del segundo jugador.

4. Las viejas y queridas celdas, ahora con sabor a datos

Dado el siguiente tipo de registro:

```
type Celda is record {
    /*
        PROPÓSITO: Modelar una celda del tablero
        INV.REP.: Los números son todos >=0
    */
    field cantidadDeAzules  //  tipo: Número
    field cantidadDeNegras  //  tipo: Número
    field cantidadDeRojas   //  tipo: Número
    field cantidadDeVerdes  //  tipo: Número
}
```

Se pide que realice las siguientes funciones y procedimientos:

a. **celdaActual** que lee la información de la celda en donde está el cabezal y retorna un dato de tipo Celda.

b. **`EscribirEnCelda_`** que dado el dato de una calda, escribe la información de este en el tablero en la celda actual.
c. **`tienenMismaCantidadDeRojas_Y_`** que dados dos datos del tipo celda, indica si efectivamente ambos tienen la misma cantidad de bolitas rojas.

5. Mis primeras personas

a. Declarar un tipo de registros llamado **`Persona`**, que contenga el número de DNI y el domicilio representados mediante **`Strings`**, y un **`Booleano`** indicando si la persona es donante de órganos. Implementar las siguientes funciones:

b. **`sonConvivientes_Y_`**, que dadas dos personas indique si comparten domicilio.

c. **`personaNacidaDe_`** que, dada una persona madre, describe a una nueva persona que haya nacido de la misma, y por lo tanto conviva con la madre, no tenga DNI asignado y en principio se indica que no es donante de órganos.
Para registrar el DNI sin asignar, escribir la función **`sinAsignar`**, que denote el string vacío y utilizarla adecuadamente.

d. **`persona_RegistradaCon_`**, que dada una persona con DNI sin asignar y un DNI de registro, describe a la persona pero con el DNI asignado al dado.

e. **`persona_ConDomicilioNuevoEn_`**, que recibe una persona y un nuevo domicilio y describe a la persona con el domicilio cambiado.

f. **`persona_ConSituaciónComoDonanteCambiada`**, que recibe una persona y retorna la persona con su situación como donante cambiada.

Reflexionamos: Considere las siguientes preguntas:

- ¿Es conveniente representar el DNI y el domicilio como Strings?
- ¿Puedo validar de alguna forma sí el DNI dado es válido?
- ¿Y para el domicilio?
- ¿Cómo puedo lidiar con distintos formatos de esribir un domicilio?
- ¿Hay forma de distinguir las partes de una dirección (calle, altura, etc.)?
- ¿Y sí representaramos el nombre de una persona?¿Conviene utilizar un String?
- ¿Qué diferencia hay entre la dirección y el nombre de una persona que hace que la primera pueda ser inconveniente para representar con Strings?
- Aquí representamos la situación de un donante como un booleano, ¿Hay situaciones donde esto sea inconveniente?
- En muchas aplicaciones que modelan personas es común incluir un campo para identificar el “sexo” de las personas, típicamente como “varón” o “mujer”, o quizás como “masculino” o “femenino”. . Sin embargo, en la actualidad la condición binaria de este concepto está en discusión, así como la necesidad de que al modelar personas sea necesario identificar esta característica, ya que existen personas que no se identifican necesariamente en forma binaria con lo que históricamente se entendió por “sexo”. Discutir en clase con los docentes y sus compañeros sobre cómo afecta la vida de las personas las decisiones sobre modelado de datos que se toman al escribir programas. Discutir también sobre cómo habría que modelar esta información en caso de considerarlo necesario. Como guía, se puede utilizar este video llamado “¿Qué es la diversidad sexual?” publicado por la Colectiva Antipatriarcal Floreciendo en Fuga:

https://www.youtube.com/watch?v=OqQm8nvEhUw.

6. Mis primeras personas ahora van a laburar

Declarar un tipo de registros llamado **`Empleado`** que contenga la identidad del empleado modelada con el registro **`Persona`**, el puesto dentro de la empresa, representado por un tipo variante llamado **`Puesto`** dado a continuación, y un sueldo modelado como un **`Número`** en pesos (sin centavos).

```
type Puesto is variant {
    /*
        PROPÓSITO: Modelar los diferentes puestos que hay
                   dentro de una empresa de software
    */
    case GestorDeProyecto {}
    case LiderDeProyecto  {}
    case Desarrollador    {}
    case Operador         {}
}
```

Implementar las siguientes funciones:

a. **empleado_ConSueldoActualizadoA_**, que dado un empleado y un nuevo sueldo, describa al empleado con el sueldo actualizado.

b. **empleado_ConNuevoPuesto_**, que dado un empleado y un nuevo puesto, describa al empleado con el puesto actualizado.

c. **categoríaDelPuesto_**, que dado un valor de tipo puesto devuelve su categoría según la siguiente tabla:

GestorDeProyecto	4
LíderDeProyecto	3
Desarrollador	2
Operador	1

d. **empleadoConMayorCargoEntre_Y_**, que dados dos empleados describa el empleado de mayor categoría entre ellos.

e. **tienenIgualSueldo_Y_**, que dados dos empleados indique si ambas cobran lo mismo.

f. **empleado_ConAumentoEn_PorBonoDeFinalizaciónDeProyecto**, que dado un empleado y un porcentaje de aumento describa al empleado con el sueldo actualizado en ese porcentaje. El porcentaje de aumento es un número del 0 al 100. El monto en el que se incrementa depende del porcentaje (i.e. si el porcentaje es 20%, cuando el sueldo es 100, el nuevo sueldo es 120, cuando el sueldo es 200 el nuevo sueldo es 240, y cuando el sueldo es 150, el nuevo sueldo es 180).

g. empleado_ConDomicilioActualizadoA_ que dado un empleado y un nuevo domicilio, describe el empleado en donde la identidad se ha actualizado para contener el nuevo domicilio.

7. Fechas, ahora en sabor datos

Declarar los tipos necesarios para representar **Fechas** (según el calendario gregoriano utilizado para identificar nuestras fechas, componiendolas de una **día** (representado mediante un **Número**, un **mes** (representado mediante un tipo **variante Mes** que modela los meses, y un **año** representado mediante un **Número**).

a. ¿Cuáles deben ser las invariantes de representación?
AYUDA: separar los meses en 3 grupos, y para Febrero considerar los años que sean bisiestos y no lo sean.

b. Escribir funciones para describir las fechas dadas en el ejercicio 5 de la práctica 2 y en el ejercicio 12 de la práctica 4 como valores del tipo **Fecha**.

8. Escribir fechas ahora requiere menos argumentos

Escribir el procedimiento **`EscribirFecha_`**, que dada una fecha, la represente con bolitas siguiendo la representación utilizada en las prácticas 2 y 4.
AYUDA: considerar reutilizar el procedimiento **`EscribirFechaConDía_Mes_Año_`**[1] definido en el ejercicio 12 de la práctica 4.

9. Y ahora puedo calcular cosas sobre las fechas

Escribir las siguientes funciones sobre fechas:

a. **`esDíaDeÑoquis_`**, que indica si el día dado es un 29.

b. **`esDelPrimerSemestre_`**, que indica si el día determinado por la fecha dada es uno del primer semestre (entre el 1ro de enero y el 31 de julio).
AYUDA: Considere implementar una función auxiliar **`númeroDelMes_`** que describa el número de un mes dado, y otra que le permita determinar si un mes es anterior a otro.

c. **`esFechaDeAñoBisiesto_`**, que dada una fecha, indica si la misma cae dentro de un año bisiesto.
AYUDA: Para saber si un año es bisiesto, hay que verificar que el año sea múltiplo de 4, pero no de 100 a menos que sea múltiplo de 400. Por ejemplo, 1796, 1896 y 1996 son bisiestos (son múltiplos de 4 y no de 100), pero 1800 y 1900 NO lo son (son múltiplos de 4 y de 100, pero no de 400); en cambio 2000 ES bisiesto (es múltiplo de 4, de 100 y de 400).

d. **`primerDíaDelInviernoDe_`**, que describa el primer día del invierno del año dado (en el hemisferio sur).

e. **`últimoDíaDelInviernoDe_`**, que describa el último día del invierno del año dado (en el hemisferio sur).

f. **`esMásAntigua_Que_`**, que dadas dos fechas indique si la primera es anterior a la segunda.
AYUDA: Considere primero comparar por año, sí no se pudiera determinar en base a este pues son iguales, considerar los meses, y sí con los meses no se pudiera determinar, considerar el día.

g. **`esInviernoEl_`**, que dada una fecha indica si el día dado es uno del invierno.
AYUDA: considerar utilizar las funciones de los 3 puntos anteriores.

h. Escriba también **`esVeranoEl_`**, **`esPrimaveraEl_`** y **`esOtoñoEl_`**, siguiendo la misma estrategia que la utilizada para **`esInviernoEl_`**.

i. **`estaciónDe_`**, que dada una fecha retorna la estación del año en la que ocurre la fecha. Para ello, definir un tipo variante **`Estación`** que modela las estaciones del año.

j. **`cuántosAñosPasaronEntre_Y_`**, que dadas dos fechas describa el número de años que hay entre las dos fechas (no tiene en cuenta períodos de menos de 1 año -- o sea, si la distancia es menor a 1 año, describe 0).

k. **`siguienteFecha_`**, que dada una fecha, devuelve la fecha del día siguiente.
AYUDA: usar funciones auxiliares **`siguienteDíaParaDía_YMes_`** y **`siguienteMesParaDía_YMes_`**.

l. **`previoFecha_`**, que dada una fecha, devuelve la fecha del día anterior.
AYUDA: usar funciones auxiliares **`previoDíaParaDía_YMes_`** y **`previoMesParaDía_YMes_`**.

[1] Observar que tienen diferente cantidad de parámetros.

m. **`EscribirCalendarioDeAño_`** que escriba en el tablero en forma vertical todas las fechas del año dado, usando la representación del ejercicio anterior, con la primera más al Norte del tablero, y la última más al Sur. No se olvide sus precondiciones.

n. **EN PAPEL** Dada la siguiente función primitiva:

```
function hayCumpleañosEn_(unaFecha)
/*
    PROPÓSITO: Indica si en la fecha dada hay un cumpleaños de alguno
               de los creadores de Gobstones
    PARÁMETROS:
        * unaFecha : Fecha - La fecha de la cual saber si hay un cumpleaños
    TIPO: Booleano
    PRECONDICIÓN: Ninguna
*/
```

Escriba las funciones **`primerCumpleañosDelAño_`** y **`últimoCumpleañosDelAño_`** que dado un año describen la fecha en la que se da el primero de los cumpleaños del año, y la última, respectivamente.