Introducción a la Programación - Práctica 8

Registros y variantes

CONSEJOS:

- Leer el enunciado en su totalidad y pensar en la forma de resolver el ejercicio 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 ejercicios están tomados de la guía complementaria realizada por 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.

EJERCICIOS:

1. 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.
2. Implementar las siguientes funciones utilizando el tipo **Carta** definido en la clase teórica
    a. **esFigura_**, que dada una Carta, describa verdadero solo si es una figura. Se dice que una carta española es una figura, porque sus dibujos representan diferentes personajes. La carta con número 10 representa un paje medieval, número 11 un caballero, número 12 el rey.
    b. **envidoCon_Y_**, que dadas 2 cartas, describa el valor del envido. Tener en cuenta la función realizada en clase teórica para el envido simple y contemplar los nuevos casos. Cuando el envido tiene alguna figura no suma 20, y las figuras no suman nada.
    c. **envidoCon_Y_mayorA_,** que dadas 2 cartas y una cantidad, indique si repreesenta ese valor de envido o mayor
3. Implementar las siguientes funciones utilizando el tipo **Celda**
    a. **celdaActual** , que describa la celda actual.
    b. cantidadDeAzules_, que describie la cantidad de azules de la celda dada.

c. cantidadTotalDeBolitas_, que describe la cantidad total de bolitas de los 4 colores de la celda dada.

d. la_tieneMasBolitasQue_, que dadas dos celdas, indica si la primera tiene más bolitas que la segunda.

```
type Celda is record {
  /* PROPÓSITO: modelar una celda del tablero
     INV.REP.: los números son todos >=0
  */
  field cantidadDeAzules  //  un número
  field cantidadDeNegras  //  un número
  field cantidadDeRojas   //  un número
  field cantidadDeVerdes  //  un número
}
```

4. Declarar un tipo de registros llamado **Persona**, que contenga el número de DNI y el domicilio representados mediante Strings, la fecha de nacimiento utilizando el tipo Fecha del ejercicio 2 y un booleano indicando si la persona es de donante de órganos. Implementar las siguientes funciones:

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

   b. **personaNacidaDe_El_**, que, dada una persona madre y una fecha, describe a una nueva persona que haya nacido en la fecha dada, 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.

   c. **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.

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

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

   f. **edadDe_Al_**, que dada una persona y una fecha, describe su edad en esa fecha. AYUDA: utilizar alguna de las funciones del ejercicio 4.

5. Discutir en clase con los docentes y sus compañeros acerca de la conveniencia de representar el domicilio y el DNI como un String, y la situación de donante como Booleano. Considerar las siguientes preguntas como guía:
   - ¿Por qué el DNI debe ser un String y no un Número? ¿Cómo se puede detectar si el String dado es un DNI válido?
   - ¿Qué situaciones anómalas pueden aparecer en la representación de domicilios como String? ¿Cómo saber si es un domicilio válido? ¿Cómo

distinguir entre diferentes formas de escribir la misma dirección? ¿Cómo distinguir las partes de las direcciones (calle, número, etc.)
    - ¿La condición de donante se puede resolver simplemente como “es donante” o “no es donante”? ¿Hay otras condiciones de donantes que no estén en uno de estos extremos? Buscar más información en la web (por ejemplo, http://www.trasplantes.net/index.php/men-ser-donante/tipos-de-donante).

6. 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.

7. Declarar un tipo de registros llamado **`Alumno`**, que contenga los siguientes datos básicos del alumno: su identidad modelada con el registro **`Persona`**, su número de legajo, la fecha de ingreso a la Universidad, la cantidad de materias que rindió y la cantidad que aprobó. Implementar las siguientes funciones:

    a. **`alumno_IngresanteEn_Con_`**, que recibe una persona, una fecha de ingreso y un número de legajo y describe un alumno que ingresó en la fecha dada con el legajo dado, y no cursó aún ninguna materia (no rindió ninguna, y por lo tanto no aprobó ninguna).

    b. **`materiasDesaprobadasPor_`**, que describe el número de materias desaprobadas por el alumno dado.

    c. **`alumno_ConLegajoNuevo_`**, que dado un alumno y un número de legajo describe el alumno que resulta de modificar el legajo del alumno dado como argumento al nuevo legajo.

    d. **`alumno_ConDomicilioNuevo_`**, que dado un alumno y un nuevo domicilio describe el alumno que resulta de modificar el domicilio del alumno dado como argumento al domicilio dado.

    e. **`edadDelAlumno_AlIngresar`**, que describe la edad que tenía el alumno dado en el momento de ingresar a la Universidad.

8. 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 centavos.

```
type Puesto is variant {
  /* PROPÓSITO: modelar los diferentes puestos que hay dentro
                de una empresa de software
  */
  case GestorDeProyecto {}
  case LíderDeProyecto  {}
  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).

9. Declarar los tipos necesarios para representar fechas (según el calendario gregoriano utilizado para identificar nuestras fechas).
   AYUDA: declarar un tipo variante llamado **`Mes`** y un tipo registro llamado **`Fecha`**.

   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.

10. Escribir funciones para describir las fechas dadas en el ejercicio 2 de la práctica 2 y en el ejercicio 5 de la práctica 3 como valores del tipo **`Fecha`**.

11. Escribir el procedimiento **`EscribirFecha_`**, que dada una fecha, la represente con bolitas siguiendo la representación utilizada en las prácticas 2 y 3.
    AYUDA: considerar reutilizar el procedimiento **`EscribirFecha___`**[1] definido en el ejercicio 4 de la práctica 3.

12. Escribir las siguientes funciones sobre fechas (puede resultar necesario escribir funciones auxiliares, como por ejemplo **`siguienteDíaDelMes__`**, que reciba un día y un mes y retorne el siguiente día; ¿por qué es necesario saber el mes?):

    a. **`esDelPrimerSemestre_`**, que indica si el día determinado por la fecha dada es uno del primer semestre.

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

    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. **`esMásAntigua_Que_`**, que dadas dos fechas indique si la primera es anterior a la segunda. AYUDA: usar funciones auxiliares **`mismoAño__`** y **`mismoMes__`**, que reciben dos fechas e indican la condición correspondiente, y **`estáAntesQue__`**, que recibe dos meses e indica si el primero aparece antes que el segundo en el mismo año.

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

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

    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. **`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`**.

    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).

[1] Observar que tienen diferente cantidad de parámetros.

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

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

13. Declarar un tipo de registros llamado **Cuenta** que contenga el número de cuenta, el tipo de cuenta (modelado por **TipoDeCuenta**), el nombre del cliente (representado como un String), un tipo de moneda (modelado por **Moneda**) y un saldo (representado como un número de centavos). Implementar las siguientes funciones:

```
type TipoDeCuenta is variant {
  /* PROPÓSITO: modelar los diferentes tipos de cuenta que
                ofrece el banco
  */
  case CuentaCorriente {}
  case CajaDeAhorro    {}
  case CuentaSueldo    {}
}

type Moneda is variant {
  /* PROPÓSITO: modelar los diferentes tipos de moneda con los
                que trabaja el banco
  */
  case Pesos   {}
  case Dólares {}
  case Euros   {}
  case Reales  {}
}
```

a. **cuenta_ConSaldoActualizadoA_**, que dada una cuenta y un nuevo saldo describa la cuenta con el saldo actualizado.

b. **cuenta_ConTipoDeCuentaCambiadoA_**, que dada una cuenta y un nuevo tipo de cuenta bancaria describa la cuenta con el tipo cambiado al dado.

c. Por disposición bancaria todas las cuentas bancarias de tipo **Sueldo** cuyo saldo supera los $25.000 deben cambiar su tipo de cuenta bancaria a Caja de Ahorro.
Implementar **cuenta_ConTipoCambiadoSegúnDisposición**, que dada una cuenta bancaria describe la cuenta con el tipo que corresponda a cada caso.

d. **saldoEnPesosDe_**, que dada una cuenta describe el saldo en pesos de la misma. Asumir los valores cambiarios dados en la siguiente tabla[2].

[2] Cotización hipotética basada en la de mediados de 2019. En 2015 era diferente (un dólar, $8, y un euro, $11). Para reflexionar sobre la importancia de las decisiones tomadas en períodos de elecciones…

Un dólar	$50
Un euro	$60
Un real	$15

e. **`númeroDeCuentaConMayorSaldoEntre_Y_`**, que dadas dos cuentas bancarias describe el número de la cuenta que tiene mayor saldo entre ambas. Tener en cuenta que las cuentas pueden estar expresadas en diferentes monedas.

f. **`cuenta_ActualizadaTrasExtracciónDeCajeroDe_`**, que dada una cuenta y un monto a extraer describa la cuenta con el saldo actualizado. ¿Es necesario establecer alguna una precondición?

g. **`puedeExtraer_PorCajeroDeLaCuenta_`**, que dado un monto a extraer y una cuenta indique si la extracción NO resultaría en un saldo negativo.

h. **`cuenta_ActualizadaTrasDepósitoEnCajeroDe_`**, que dada una cuenta y un monto a depositar describa la cuenta con el saldo actualizado.

14. Declarar el registro **`Coordenada`** que representa una coordenada del tablero con los campos **`fila`** y **`columna`**, donde la coordenada del origen tiene fila 0 y columna 0. Implementar las siguientes operaciones:

    a. **`coordenada__`**, que dado los números **`x`** e **`y`** retorne una nueva coordenada con fila en **`x`** y columna en **`y`**.

    b. **`esCoordenadaOrigen_`**, que denote la coordenada del origen, i.e., la que tiene fila y columna en 0.

    c. **`esLaMayorCoordenadaEntre_Y_`**, que, dadas dos coordenadas determina si la primer aparece después de la segunda en un recorrido que va primero hacia el Norte y luego hacia el Este. Es decir, **`esLaMayorCoordenadaEntre_Y_(c1, c2)`** es verdadero cuando la columna de **`c1`** es mayor que la de **`c2`**, o cuando estando en la misma columna, la fila de **`c1`** es mayor que la de **`c2`**.

    d. **`coordenadaActual`**, que describa la coordenada en la que se encuentra el cabezal.

    e. **`últimaCoordenada`**, que describa la última coordenada del tablero en un recorrido hacia el Noreste.

    f. **`IrACoordenada_`**, que, dada una coordenada posicione el cabezal en la celda correspondiente del tablero.

    g. **`siguienteCoordenadaA_`**, que, dada una coordenada describa la siguiente coordenada en un recorrido que va primero hacia el Norte y luego hacia el Este.

h. **`reflejoDe_`**, que dada una coordenada describa la coordenada que se obtiene de intercambiar la fila y la columna.