Introducción a la Programación - Práctica 2

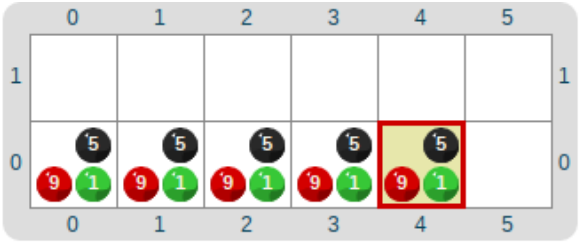
División en subtareas y representación de la información

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, y esta después de las actividades de indagación.
- 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, Introducción a la Programación de la Universidad Nacional de Hurlingham de Alan Rodas Bonjour y su equipo, de ejercicios y actividades realizadas 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.
- **Realizar EN PAPEL los ejercicios que así lo indiquen.**
- **Si un ejercicio indica BIBLIOTECA significa que será útil para la realización de futuros ejercicios tanto en esta guía como en las siguientes, pudiendo ser utilizado sin tener que volver a definirlo. Es útil mantener registro de dichos procedimientos en su carpeta.**

EJERCICIOS:

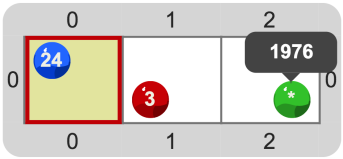
1. Escribir un procedimiento **`PonerGuardaDe5Azulejos()`**, que arme una “guarda” de 5 azulejos (como las que decoran las paredes). Cada azulejo está conformado por 1 bolita verde, 5 negras y 9 rojas. No es necesario que el cabezal termine en la posición inicial.



   Recordar seguir una metodología adecuada para la construcción del código: comenzar por escribir el contrato completo del procedimiento, luego pensar las subtareas necesarias y darles un nombre adecuado, escribir el contrato de las subtareas, LUEGO el código del procedimiento pedido en términos de las subtareas, y FINALMENTE, realizar el código de las subtareas siguiendo la misma metodología (que denominamos metodología de construcción de programas *top-down*).

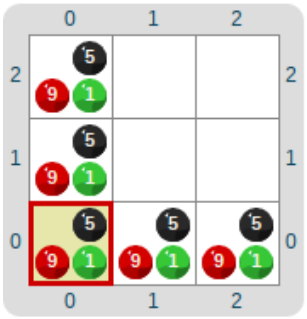
2. Utilizando bolitas pueden representarse diversos elementos; un ejemplo de esto es la posibilidad de representar una fecha. Una fecha que los argentinos deberíamos recordar, para no repetirla jamás, es el 24 de marzo de 1976, hoy constituido como Día de la Memoria por la Verdad y la Justicia en Argentina.

   Hacer un procedimiento **`RegistrarElDíaDeLaMemoria()`** que
   - en la celda actual, ponga 24 bolitas Azules, que representan el día,
   - en la celda lindante al Este de la actual, ponga 3 bolitas Rojas, que representan el mes,
   - en la celda lindante al Este de la anterior, ponga 1976 bolitas Verdes, representando el año, y
   - el cabezal termina en la celda inicial.



   Recordar que la solución completa incluye la documentación: propósito, precondición, los cuales deben ser escritos ANTES de escribir el código.

3. Hacer un procedimiento **`PonerGuardaEnL()`**, que arme una guarda en L como muestra la figura, dejando el cabezal en la posición inicial.



¿Pensaste en reutilizar el procedimiento definido antes, o empezaste a escribirlo de nuevo? Dado que ahora se cuenta con la subtarea definida en el ejercicio anterior, la metodología *top-down* se puede enriquecer con la reutilización de procedimientos ya realizados; esto puede hacer que ciertas divisiones en subtareas, que permiten esa reutilización, sean más sencillas que otras.

4. Escribir un programa para cultivar tomates en todo un lote de 5x3. Los procedimientos primitivos en este ejercicio son **`Sembrar`** y **`Regar`** con los siguientes contratos:

```
procedure Sembrar()
  /* PROPÓSITO: Sembrar una semilla en la celda actual
     PRECONDICIONES:
       * la celda actual está vacía
  */

procedure Regar()
  /* PROPÓSITO: Regar la celda actual
     PRECONDICIONES:
       * hay una semilla sembrada en la celda actual
  */
```

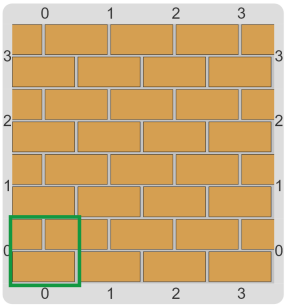
5. Escribir otro programa para sembrar tomates en todo un lote de 5x3 pero ahora los procedimientos primitivos son **`Sembrar`** y **`RegarFila`** con los siguientes contratos:

```
procedure Sembrar()
  /* PROPÓSITO: Sembrar una semilla en la celda actual
     PRECONDICIONES:
       * la celda actual está vacía
  */

procedure RegarFila()
  /* PROPÓSITO: Regar una fila del lote de ancho 5. El cabezal
                termina en la celda inicial
     PRECONDICIONES:
       * hay 4 celdas al Este de la celda actual
```

```
      * hay una semilla sembrada en la celda actual y en
          cada una de las 4 celdas al Este de la actual
  */
```

6. Escribir un programa que construya una pared de ladrillos de ocho ladrillos de alto por cuatro ladrillos de ancho, para lo cual se provee una guía al final del enunciado. Una pared de ladrillos está compuesta por hileras de ladrillos pegadas entre sí con cemento. Las hileras se alternan entre hileras con 4 ladrillos enteros (hilera inferior), e hileras con 3 ladrillos enteros y 2 medios ladrillos en los bordes (hilera superior). Cada fila del tablero puede llevar 2 hileras, una hilera inferior en la parte inferior de las celdas, y una hilera superior arriba de esa. El tablero inicial estará vacío.



Los procedimientos primitivos en este ejercicio son **PonerCemento**, **PonerLadrillo**, **PonerMedioLadrilloEnElBordeOeste** y **PonerMedioLadrilloEnElBordeEste** con los siguientes contratos:

```
procedure PonerCemento()
  /* PROPÓSITO: Poner cemento en la celda actual
     PRECONDICIONES:
       * o bien la celda actual está vacía,
       * o bien la celda actual tiene un ladrillo entero en la
         Parte inferior, sin cemento sobre él
  */

procedure PonerLadrillo()
  /* PROPÓSITO: Poner un ladrillo en la celda actual
     PRECONDICIONES:
      * si no hay ladrillos en la celda actual, hay
        cemento en la misma
      * si hay un ladrillo entero en la celda actual, hay
        cemento sobre el mismo, y existe una celda al Oeste
        que ya contiene un ladrillo entero y medio ladrillo del
        lado izquierdo
  */

procedure PonerMedioLadrilloEnElBordeOeste()
```

```
/* PROPÓSITO: Poner medio ladrillo en la celda actual
   PRECONDICIONES:
    * la celda actual está en el borde Oeste
    * hay un ladrillo entero en la parte inferior de la
      celda actual, y cemento sobre él
*/

procedure PonerMedioLadrilloEnElBordeEste()
  /* PROPÓSITO: Poner medio ladrillo en la celda actual
     PRECONDICIONES:
      * la celda actual esté en el borde Este
      * hay un ladrillo entero en la parte inferior de la
        celda actual, cemento sobre él, y medio ladrillo del lado
        izquierdo de la celda
  */
```

No olvidar seguir la metodología de trabajo *top-down*, y utilizar nombres adecuados para las subtareas. Como guía para la utilización de tal metodología, se proponen los siguientes ítems. Una ayuda: para poder utilizar adecuadamente los procedimientos primitivos, deben analizarse con cuidado las precondiciones y determinar qué tareas deben realizarse primero.

a. Para construir una pared, deben construirse repetidamente hileras de ladrillos en la parte inferior de cada celda de una fila, e hileras en la parte superior, teniendo en cuenta las precondiciones de cada una de las operaciones.
b. Los ladrillos de la hilera superior se colocan con un corrimiento respecto de los de la hilera inferior. Observar con cuidado la precondición del procedimiento de colocar ladrillos enteros y probar su funcionamiento en una hilera superior, luego de cumplir sus precondiciones.

7. **EN PAPEL** Escribir un programa para cumplir con los propósitos de cada uno de los ítems que se indican más adelante. Para ello, deben utilizarse los procedimientos indicados a continuación.

```
procedure DibujarBase()
  /* PROPÓSITO: Dibujar una base de pirámide de 5 celdas de lado
                a partir de la celda actual y hacia el Este.
     PRECONDICIONES:
      * La celda actual está vacía
      * Hay al menos cuatro celdas al Este del cabezal y las
          cuatro más cercanas están vacías
  */

procedure DibujarMedio()
  /* PROPÓSITO: Dibujar un sector del medio de pirámide de 3
```

```
                celdas de lado a partir de la celda actual
                y hacia el Este.
    PRECONDICIONES:
     * La celda actual está vacía
     * Hay al menos dos celdas al Este del cabezal y
         las dos más cercanas están vacías
  */

procedure DibujarPunta()
  /* PROPÓSITO: Dibujar una punta de pirámide en la celda actual.
     PRECONDICIONES:
     * La celda actual está vacía
  */
```

¡Recordar! Para cada ítem que sigue debe comenzarse redactando el contrato correspondiente, y de ser necesario, se deben definir subtareas para construir su solución, expresándolas mediante procedimientos (y en ese caso se deben escribir primero los contratos de los mismos y utilizarlos ANTES de dar su código).

a. Una pirámide.
b. Una pirámide invertida.
c. Una pirámide estirada a lo alto (con dos segmentos de cada uno).
d. Una pirámide gigante (con 11 bloques de base y 6 bloques de altura, donde cada segmento es dos celdas más chico que el que tiene debajo). Para esto deben reutilizarse los procedimientos implementados en los puntos a. y b.
   PISTA: una pirámide gigante se puede conseguir con 4 pirámides comunes, si se las ubica con cuidado.

8. El bosque, parte 1

En este ejercicio, se usará el tablero para ***representar un bosque***. Cada celda representa a una ***parcela***. Cada ***bolita verde representa un árbol***. Cada **b*olita roja representa una semilla***. Una ***bolita negra representa una bomba***. Una ***bolita azul representa una unidad de nutrientes***.

Escribir los siguientes procedimientos de representación, que hacen lo que su nombre indica. Todos trabajan siempre sobre la celda actual.

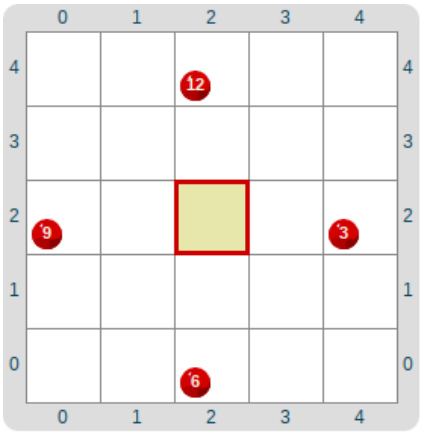
a. **`PonerUnaSemilla()`**
b. **`PonerUnÁrbol()`**
c. **`PonerUnaBomba()`**
d. **`PonerUnNutriente()`**
e. **`SacarUnaSemilla()`**
f. **`SacarUnÁrbol()`**

g. **`SacarUnaBomba()`**

h. **`SacarUnNutriente()`**

9. Dibujar un reloj analógico de agujas en un tablero cuadriculado puede ser un desafío. Una simplificación posible sería representar solamente algunos de los números que aparecen en el mismo:
   - el 12 arriba,
   - el 3 a la derecha,
   - el 9 a la izquierda y
   - el 6 abajo.

   Construir un procedimiento **`DibujarRelojAnalógicoSimplificado()`**, que ponga los números del reloj tal como se indicó, alrededor del casillero actual. El tamaño del reloj por el momento, será de 2 celdas de "radio" (suponiendo que miramos al reloj como un círculo).
   El efecto de utilizar el comando **`DibujarRelojAnalógicoSimplificado()`** en un tablero inicial vacío de 5x5 con la celda inicial en el centro del mismo, obtiene el siguiente tablero final:



   Recordar escribir el contrato en primer lugar, y en caso de utilizar división en subtareas, seguir la metodología *top-down* para las mismas (primero su nombre y su contrato, usarlas, y recién luego definirlas con la misma metodología).

10. **EN PAPEL** En el equipo de programación empezó a trabajar un programador que no tiene formación profesional, y al que todo el equipo apodó Nova[1]. En el código que escribió Nova se encontró un procedimiento que no aplica la técnica de división en subtareas, y tampoco tiene contrato. Para poder integrar el código con el resto, en este equipo se requiere que los programas sigan buenas prácticas de programación. Por eso, se pide corregir este código para que cumpla con las mismas.

```
procedure ConstruirEscaleraAzulDe4Escalones() {
```

[1] Al principio creían que era porque era un Novato, pero en realidad es porque su código suele tener eventos cataclísmicos inesperados, tal como el fenómeno astronómico del mismo nombre…

```
    Poner(Azul) Mover(Este) Poner(Azul) Mover(Norte)
    Poner(Azul) Mover(Este) Poner(Azul) Mover(Norte)
    Poner(Azul) Mover(Este) Poner(Azul) Mover(Norte)
    Poner(Azul) Mover(Este) Poner(Azul)
}
```

a. Comenzar por redactar el propósito y la precondición correspondientes.

b. Luego aplicar la técnica de división en subtareas expresadas con procedimientos. Dado que no se trata de hacer el código de nuevo, se propone identificar partes de código que se repitan, darles un buen nombre y extraer esa parte de la estrategia en procedimientos auxiliares. Por supuesto, cada procedimiento auxiliar debe tener un nombre adecuado, y debe documentarse su propósito y precondición.

c. ¿Puede apreciar las ventajas de contar con el contrato y de expresar la estrategia como parte del código en forma de procedimientos? Hay más de una forma de dividir en subtareas. Comparar la solución propuesta con la de otros compañeros, y discutir las ventajas y desventajas de la dada con las que resulten diferentes.