Actividades en Gobstones v0.3

Direcciones válidas: Este, Oeste, Norte, Sur.

Colores válidos: Rojo, Negro, Azul, Verde.

Primitivas de Gobstones útiles

- Poner(<Color>): comando que pone en la celda actual una pelotita del color indicado.
- Sacar(<Color>): comando que saca de la celda actual una pelotita del color indicado.
- Mover(<Dirección>): comando que mueve el cabezal hacia la dirección indicada.
- nroBolitas(<Color>): función que retorna el valor de la cantidad de bolitas del color indicado que hay en la celda actual.
- puedeMover(<Dirección>): función que retorna un valor booleano que indica si se puede mover el cabezal hacia la dirección indicada.
- opuesto(<Dirección>): función que devuelve la dirección opuesta a la pasada por parámetro.

Operaciones aritméticas: +, - , *, div, ^, mod[1]

Ejemplos:

- *“4 div 2”* es equivalente a escribir el número 2
- *“3+2*3”* es equivalente a escribir el número 9
- *“4 mod 2”* es equivalente al número 0. Leer sobre la operación módulo

Operaciones booleanas: ||, &&, not

Ejemplos

- Negación: *“not False”* es equivalente a *“True”*
- Conjunción: *“True && False” es equivalente a “False”*
- Disyunción: *“True || False || False” es equivalente a “True”*

[1] En Gobstones la operación matemática módulo se escribe con la palabra clave *mod.* Esta operación devuelve el resto de dividir el primer número por el segundo. Por ejemplo, 10 mod 3 devuelve 1, 10 mod 5 devuelve 0 y 10 mod 7 devuelve 3.

Problema 1: Recolectar 1 carbón y 1 hierro para 4x5 desde extremo SudOeste

Tópicos: #procedimientos #modularización #legibilidad #repeticiónSimple

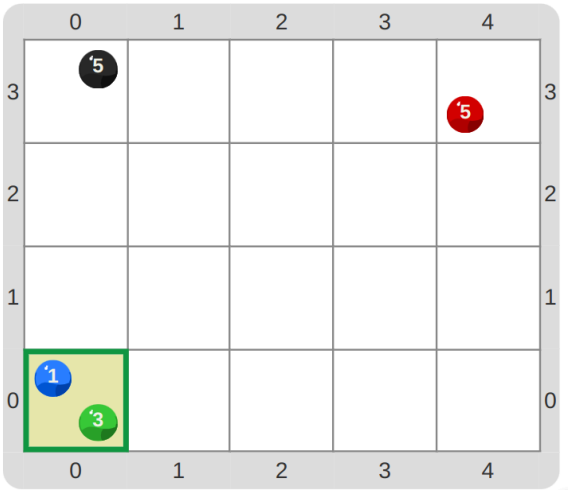
Especificación

Pre:

1) Tablero es 4x5 filas x cols
2) La nave (1Azul), el marciano (3Verdes) y el cabezal están en el extremo SO.
3) Hay mina de carbon (5Negras) en extremo NO y de hierro al NE (5Rojas).

Post:

1) Hay un hierro más y un carbón más en la nave que lo que había inicialmente
2) Hay un carbón menos en la mina y lo mismo para el hierro.
3) Todo lo demás está igual a como estaba inicialmente (alien, cabezal, otros)



Enunciado

Considerando que tiene los siguientes procedimientos disponibles:

```
// Puede copiar y pegar este fragmento de código
procedure SacarNBolitasDeColor(color, n){
    repeat(n){
        Sacar(color)
    }
}

procedure PonerNBolitasDeColor(color, n){
    repeat(n){
        Poner(color)
    }
}

procedure SacarMarciano(){
    SacarNBolitasDeColor(Verde, 3)
}

procedure PonerMarciano(){
    PonerNBolitasDeColor(Verde, 3)
}
```

```
procedure IrASentido(sentido){
    Mover(sentido)
}
```

1) Se pide implementar los siguientes procedimientos:
    a) RecolectarUnCarbón()
    b) RecolectarUnHierro()
2) Hacer un programa que haga que el marciano recolecte todo el hierro y carbón, guardándolo en su nave. Modificar la especificación dada para que ahora contemple este nuevo escenario.

Problema 2: Minas en los extremos

Tópicos: #función #repeticiónCondicional #variable #modularización #procedimientos

Especificación

Pre:

1) La nave (1Azul) y el marciano (3Verdes) están en el extremo SO.
2) Hay mina de carbón (negras) en extremo NO y de hierro al NE (rojas).

Post:

1) No queda nada de las minas, están vacías.
2) Todo lo que había en las minas está en la nave.
3) El marciano y la nave están en el extremo SO, todo lo demás sigue igual (cabezal, otras cosas en el mapa).

Enunciado

Se desea ahora que el marciano recolecte todo el hierro y carbón que haya en las minas, donde no se sabe a priori cuánto hay de cada mineral.
Considere que ahora no se sabe el tamaño del mapa posible, por lo que su procedimiento debe andar para cualquier tablero.

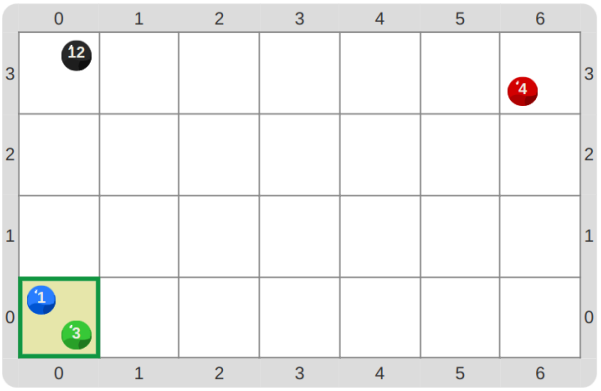
Se pide implementar:

a) Las funciones tamañoDeMinaDeCarbon y tamañoDeMinaDeHierro que debe devolver los tamaños las minas correspondientes (suponiendo que el cabezal está en la mina correspondiente).
b) Los procedimientos RecoletarTodoCarbonDeMina y RecoletarTodoHierroDeMina que debe hacer que el marciano recolecte todo, suponiendo que ya está posicionado en la mina.
c) El procedimiento RecolectarTodosLosMinerales que resuelva el problema mencionado.

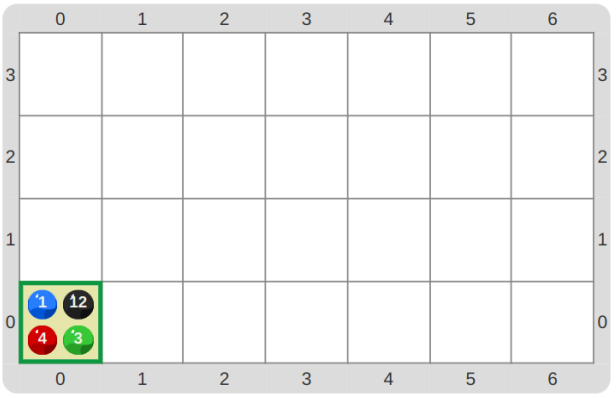
Ejemplos

Caso 1

Tablero inicial

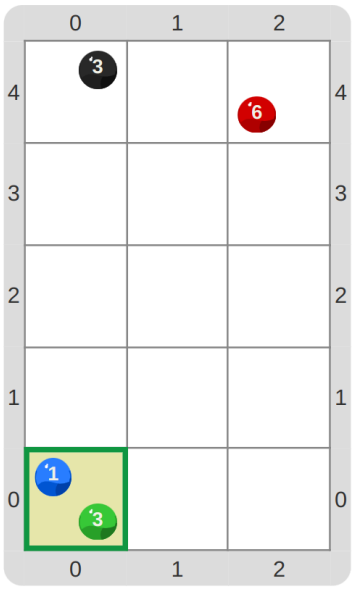


Tablero final

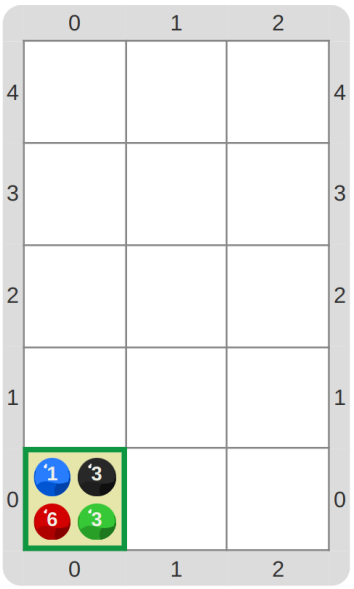


Caso 2

Tablero inicial



Tablero final



Problema 3: Robot que limpia la basura

Tópicos: #alternativaCondicional #repeticiónCondicional #variable #funciones

Enunciado

En la nave, el marciano tiene un robot azul que colabora con la limpieza de los largos pasillos. El largo de los pasillos puede ir cambiando y hay zonas riesgosas donde no puede ir el robot limpiando. De hecho, las zonas accesibles por el robot se marcan con una línea negra.

Considerando que tiene los siguientes procedimientos y funciones dados:

```
// Puede copiar y pegar este fragmento de código
procedure AvanzarRobot(){
    SacarNBolitasDeColor(Azul,3)
```

```
    Mover(Este)
    PonerNBolitasDeColor(Azul,3)
}

procedure LevantarBasuras(cantidadDeBasuraALevantar){
    SacarNBolitasDeColor(Verde, cantidadDeBasuraALevantar)

}
function estaRobotEnPasillo(){
    return(nroBolitas(Negro)>0)
}

function hayBasura(){
    return(nroBolitas(Verde)>0)
}

function obtenerCantidadDeBasuraEnCelda(){
    return(nroBolitas(Verde))
}

procedure SacarNBolitasDeColor(color, n){
    repeat(n){
        Sacar(color)
    }
}

procedure PonerNBolitasDeColor(color, n){
    repeat(n){
        Poner(color)
    }
}
```

(a) Implementar el procedimiento LevantarBasurasDelPasillo que se encargue de hacer que el robot vaya levantando cada basura que encuentra a lo largo del pasillo mientras avanza.

Especificación

Pre:

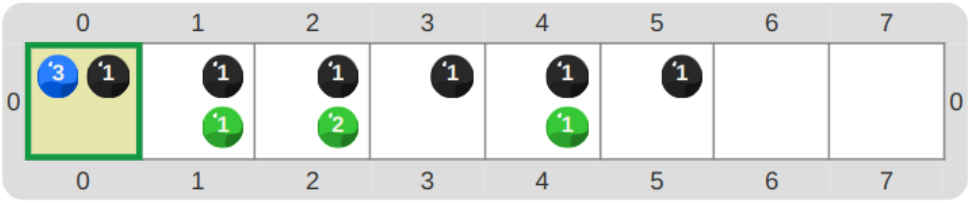
1. El tablero tiene solo una fila y al menos dos columnas.
2. El robot (3Azules) está en algún lugar del tablero junto al cabezal.
3. Desde el robot arranca el pasillo (representado por 1Negra en c/celda), cuyo tamaño es de al menos una celda.
4. Al final del pasillo (al oeste), hay al menos una posición vacía.
5. A lo largo del pasillo puede haber basuras (representada con Verdes).

Post:

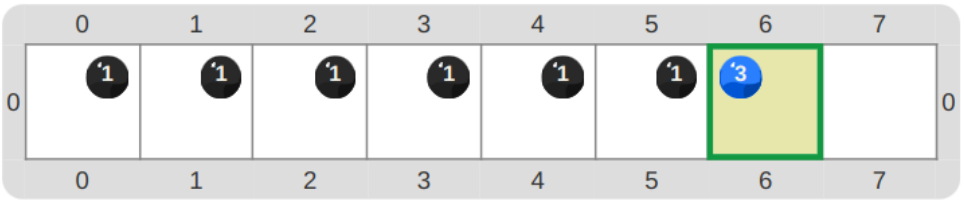
1. El robot y el cabezal están al final del pasillo.
2. Las basuras del pasillo ya no están.
3. El pasillo está igual que al inicio.

Ejemplo

Tablero inicial:



Tablero final:

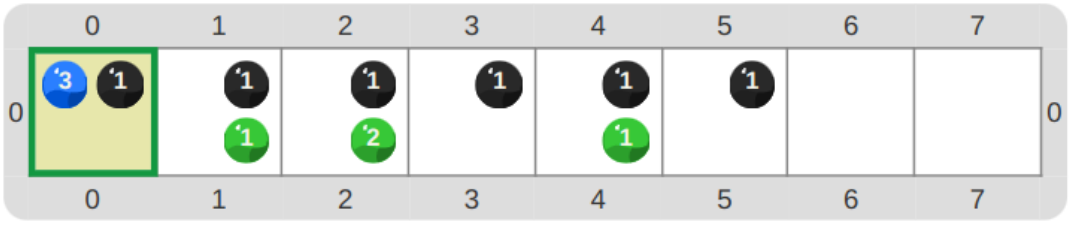


b) Especificar e implementar el procedimiento ContarBasurasDelPasillo, el cual haga que el robot avance por el pasillo mientras que va contando la cantidad de basura que ve tirada. Al finalizar el recorrido, debe poner marcar con rojo la cantidad total que encontró.

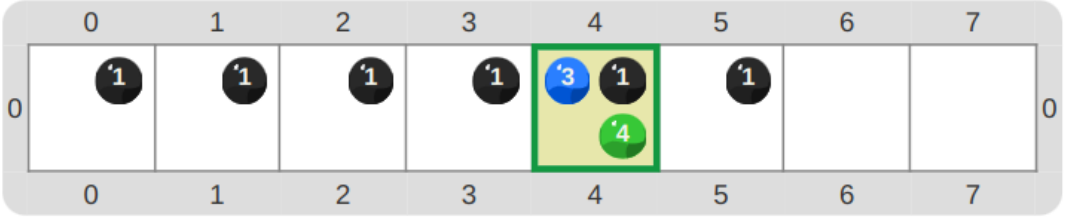
c) Especificar e implementar el procedimiento BarrerBasurasDelPasillo, el cual haga que el robot vaya barriendo toda la basura que encuentra en el pasillo. Al final debe quedar el pilón de toda la basura y además el robot.

Ejemplo

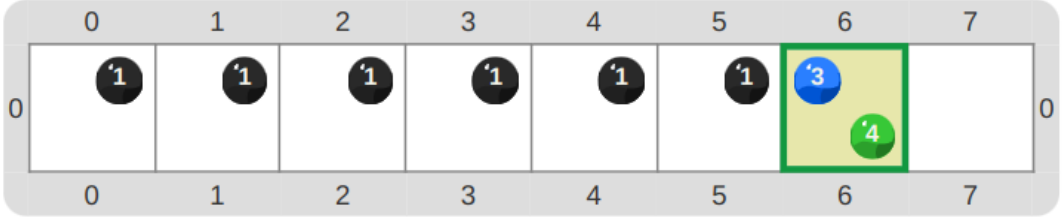
Tablero inicial



Tablero intermedio



Tablero final



Problema 4: Marciano recolectando con capacidad de carga

Tópicos: #variable #repeticiónCondicional

Enunciado

El marciano ahora está tratando de recolectar todo el hierro y carbón posible, tratando de no sobrecargar su nave con el peso de carga y recolectando de la forma más pareja posible (un hierro y un carbón por vez). Para ello cuenta con un letrero fijo que le recuerda cuál es la capacidad de carga de la nave.

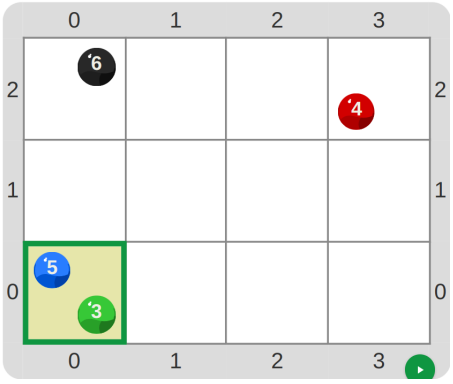
Especificar e implementar el procedimiento RecolectarCantidadMáximaDeMinerales. Emplee la misma representación usada en el problema #2, considerando que la cantidad de bolitas azules en el extremo SO indican ahora la capacidad de carga de la nave (y no solamente la presencia de la nave).

Nota: Puede suponerse que el tamaño de las minas de minerales superan por mucho la capacidad de la nave. No es necesario actualizar el letrero que indica la capacidad de carga.

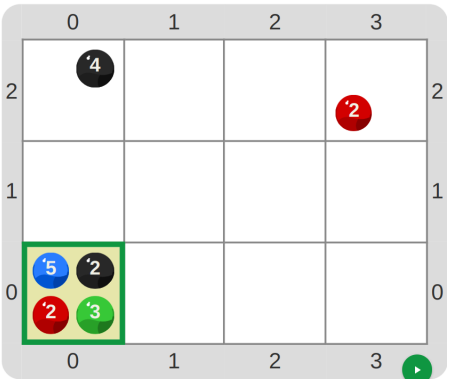
Ejemplos

Caso 1

Tablero inicial

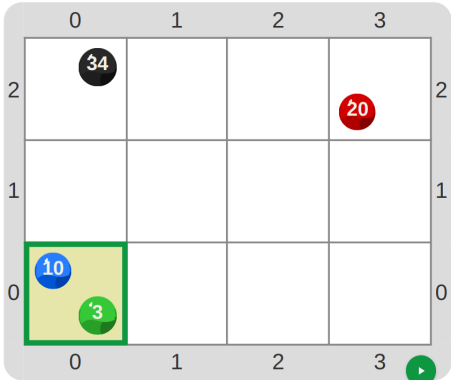


Tablero final

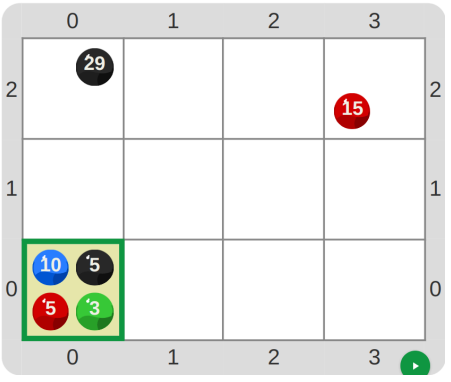


Caso 2

Tablero inicial



Tablero final



Problema 5: Marciano explorando el mapa

Tópicos: #recorridos #repeticiónCondicional #alternativaCondicional

El marciano llegó a una tierra desconocida y quiere poder explorar todo el área en forma de zig-zag (como se indica en la figura). Implementar el procedimiento ExplorarMapa, que haga que el marciano vaya recorriendo todo el mapa de forma ordenada. Recuerde que el mapa puede tener cualquier tamaño.

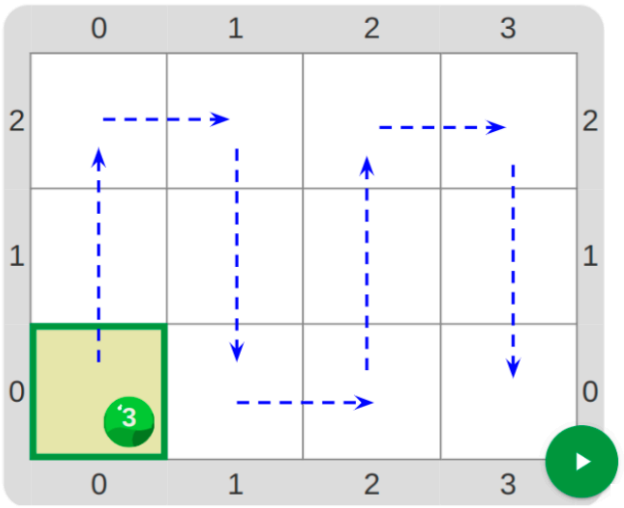
Especificación

Pre:

1) El marciano (3Verde) inicia la exploración en el extremo SO.
2) El mapa tiene al menos 2 columnas y 2 filas.
3) La cantidad de columnas del mapa es par.
4) El cabezal apunta al marciano.

Post:

1) El marciano termina su exploración en el extremo SE.
2) Nada del tablero se ve alterado, el cabezal sigue apuntando al marciano.



Biblioteca de representación

Para comenzar a abordar problemas más complejos, vamos a expandir los procedimientos y funciones primitivos de Gobstones. Para ellos definimos como Biblioteca de Representación a un conjunto de procedimientos y funciones que vamos a suponer como disponibles para la ejercitación.

Biblioteca de Representación:

* procedure SacarNBolitasDeColor(n, color)
* procedure PonerNBolitasDeColor(n, color)

Expansión de la Biblioteca de Representación

Para cada una de las siguientes funciones, cree un programa para poder testear su correcto funcionamiento. Puede que le sea útil revisar sobre cómo se escriben las operaciones booleanas en Gobstones (ver inicio de esta guía).

Expanda la biblioteca implementando las funciones:

a) esCeldaVacía(), que indica si la celda actual se encuentra vacía
b) hayAlMenosUnaDeCada(), que indica si en la celda actual hay al menos una bolita de cada color.
c) esCeldaConBolitas(), que indica si la celda actual tiene al menos una bolita, de cualquier color.
d) hayBolitasDeColorHacia(color, dirección), que devuelva un booleano indicando si hay bolitas del color indicado en la dirección pasada por parámetro. Notar que el cabezal debe mantenerse en la misma posición inicial.
e) hayAlMenosNBolitasDeColorHacia(n, color, dirección), que indica si hay igual o menor cantidad de bolitas del color indicado, en la dirección y del color indicados por parámetro.
f) hayNBolitasDeColorHacia(n, color, dirección), análogamente al ítem anterior pero con la cantidad exacta indicada.
g) hayNBolitasDeColorADistancia(n, color, dirección, distancia), que indica si en la *dirección* pasada por parámetro se encuentran exactamente *n* bolitas del *color* indicado, a exactamente *distancia* celdas de la posición actual.

Problema 6: Moviéndose al planeta cercano

Tópicos: #repeticiónCondicional #alternativaCondicional #funciones

El marciano se encuentra explorando la galaxia con su nave, buscando qué planeta puede serle útil para conseguir nuevos recursos naturales. Su principal limitante de búsqueda es el combustible, ya que gasta una unidad por celda espacial recorrida. La cantidad de combustible se indica por la cantidad de Azules.

(a) Implementar la función hayPlanetaAlcanzableHacia(dirección), que indica si se halla un planeta que sea alcanzable por la nave en la dirección indicada (es decir, si se encuentra a menor o igual cantidad de celdas que la cantidad de combustible disponible).

(b) Implementar el procedimiento MoverHastaPlanetaAlcanzableHacia(), que se encargue de mover la nave hacia el planeta alcanzable en caso de haber uno. Si no hay, la nave se queda en el mismo lugar. Recuerde modificar la cantidad de combustible indicada a medida que avanza la nave.