studocu

Actividades en Gobstones

Programación Imperativa (Universidad Nacional del Chaco Austral)

Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university
Downloaded by Ian Chen (chenian317@gmail.com)

Actividades en Gobstones

A continuación presentamos actividades y ejercicios para trabajar en Gobstones. Las primeros son ejercicios cortos y que buscan reforzar la parte teórica que se intenta dar. Los últimos son ejercicios integradores y por ende más largos. También recomendamos el siguiente blog, perteneciente a Pablo Barenbaum (profesor de la Universidad Nacional de Quilmes), el cual posee actividades completas y desafiantes pensadas para Gobstones.

Además proveemos un documento que complementa la parte teórica de estas actividades.

Enunciado: escribir el siguiente programa Gobstones y correrlo

```
procedure Main()
{
  Poner(Azul)
  Poner(Rojo)
}
```

Ideas involucradas: sintaxis abstracta, valores, comandos

Enunciado: Dibujar un cuadrado lleno de 3x3 de color rojo, utilizando la idea de subtareas para simplificar el código.

Ideas involucradas: valores, comandos, subtareas, precondiciones

Posible solución:

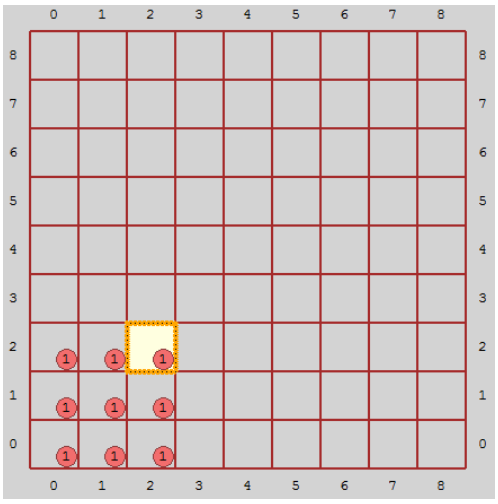
```
// Precondición: Se necesitan 2 celdas al este

procedure DibujarFilaRoja()
```

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

```
{
  Poner(Rojo)
  Mover(Este)
  Poner(Rojo)
  Mover(Este)
  Poner(Rojo)
}

procedure Volver2YSubir1()
{
  Mover(Oeste)
  Mover(Oeste)
  Mover(Norte)
}

// Precondición: Se necesitan 2 celdas este y 2 al norte
procedure DibujarCuadradoLlenoRojo()
{
  DibujarFilaRoja()
  Volver2YSubir1()

  DibujarFilaRoja()
  Volver2YSubir1()

  DibujarFilaRoja()
}
```

Downloaded by Ian Chen (chenian317@gmail.com)



Enunciado: cambiar el siguiente programa para que dibuje un cuadrado de color azul. Luego volver a cambiarlo para que dibuje uno de color verde.

```
procedure Linea()

{

Poner(Rojo)

Mover(Este)

Poner(Rojo)

Mover(Este)

Poner(Rojo)

Mover(Este)


Mover(Oeste)

Mover(Oeste)
```

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Mover(Oeste)

}

procedure Cuadrado()

{

Linea()

Mover(Norte)

Linea()

Mover(Norte)

Linea()

Mover(Norte)

}

Ideas involucradas: valores, comandos, subtareas

Enunciado: Parametrizar el siguiente programa para que dibuje cuadrados de colores arbitrarios (que puedan ser elegidos por el programador al momento de querer dibujar el cuadrado). Se deberían poder escribir los siguientes llamandos a procedimientos: Cuadrado(Rojo), Cuadrado(Negro), Cuadrado(Azul), Cuadrado(Verde).

procedure Linea()

{

Poner(Rojo)

Mover(Este)

Poner(Rojo)

Downloaded by Ian Chen (chenian317@gmail.com)

Mover(Este)

Poner(Rojo)

Mover(Este)

Mover(Oeste)

Mover(Oeste)

Mover(Oeste)

}

procedure Cuadrado()

{

Linea()

Mover(Norte)

Linea()

Mover(Norte)

Linea()

Mover(Norte)

}

Ideas involucradas: valores, comandos, subtareas, parámetros.

Enunciado: Definir un procedimiento que dibuje el perímetro de un cuadrado de 3 celdas por lado, utilizando la subtarea LineaDe2Hacia(dir, color), que toma una dirección y un color y dibuja una linea de 2 celdas hacia la dirección indicada pintando del color indicado por parámetro.

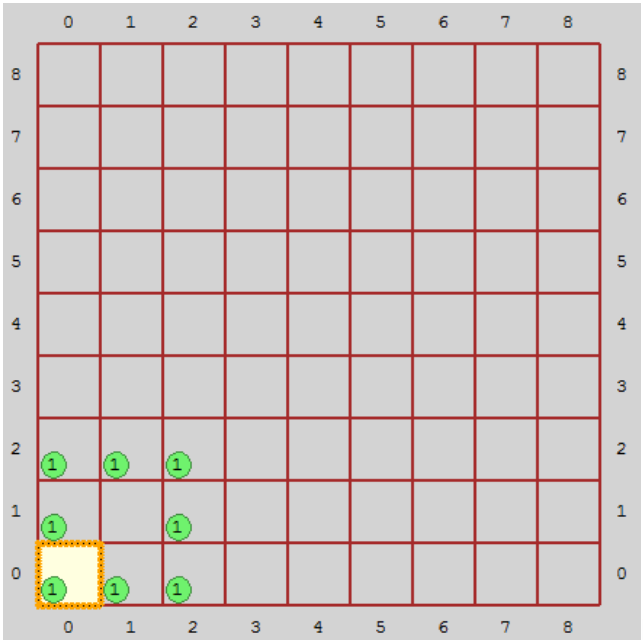
This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Ideas involucradas: parámetros, subtareas

Posible solución:

```
procedure LineaDe2Hacia(dir,color)
{
 Poner (color)
 Mover (dir)
 Poner (color)
 Mover (dir)
}

procedure PerimetroCuadradoLado3(color)
{
 LineaDe2Hacia(Norte, color)
 LineaDe2Hacia(Este, color)
 LineaDe2Hacia(Sur, color)
 LineaDe2Hacia(Oeste, color)
}
```



PerimetroCuadradoLado3(Verde)

Downloaded by Ian Chen (chenian317@gmail.com)

Enunciado: Definan los siguientes procedimientos:

- PonerX(cantidad, color), que pone “cantidad” bolitas de color “color”.

- SacarX(cantidad, color), que saca “cantidad” bolitas de color “color”.

- MoverX(cantidad, dir), , que mueve el cabezal “cantidad” veces hacia la dirección “dir”.

Ideas involucradas: parámetros, repetición

Posible solución:

```
procedure MoverX(cant,dir)

{ repeatWith i in 1..cant { Mover (dir) } }


procedure PonerX(cant,color)

{ repeatWith i in 1..cant { Poner(color) } }


procedure SacarX(cant,color)

{ repeatWith i in 1..cant { Sacar (color) } }
```

Enunciado: Definir los procedimientos Línea(tam, dir, color), PerímetroCuadrado(tam, color) y CuadradoLleno(tam, color). Utilizando todas las ideas que se vieron hasta el momento para simplificar código:
- Repetición
- Subtareas
- Parámetros

Ideas involucradas: parámetros, repetición, subtareas

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Posible solución:

```
procedure Linea(tam, dir, color)
{
 repeatWith i in 1..tam
 {
   Poner (color)
   Mover(dir)
 }
}

procedure PerimetroCuadrado(tam, color)
{
 repeatWith d in minDir()..maxDir()
 {
  Linea(tam, d, color)
 }
}

procedure CuadradoLleno(tam, color)
{
 repeatWith n in 1..tam
 {
  Linea(tam, Este, color)
  MoverX(tam, Oeste)
  Mover(Norte)
 }
}
```

Enunciado: Definir el procedimiento Progresión(min,max,dir,col) que dibuja una linea que empieza poniendo “min” bolitas de color “col” y continúa moviéndose hacia “dir” hasta llegar a

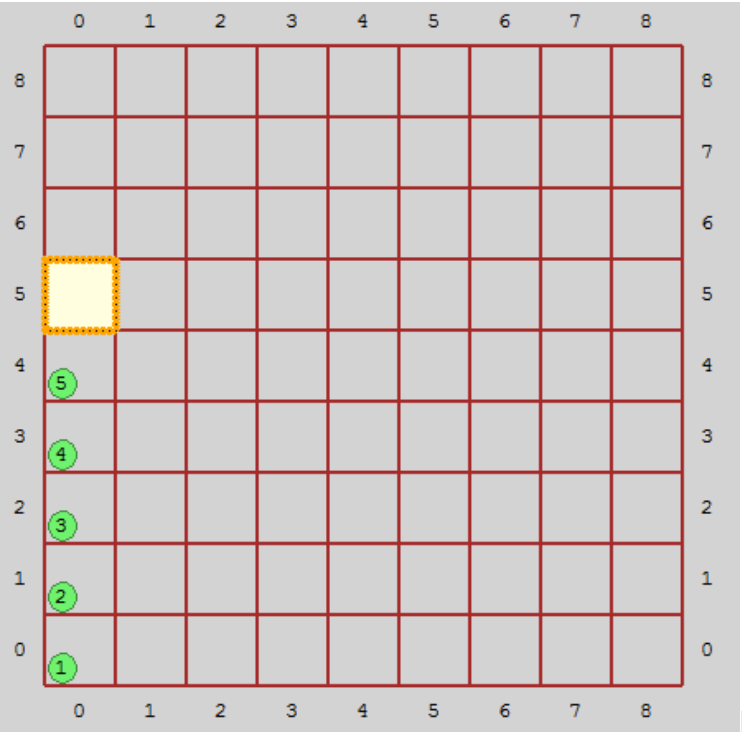
Downloaded by Ian Chen (chenian317@gmail.com)

poner “max” bolitas en la última celda. El largo de la línea será “max - min”. Utilizar una repetición indexada de min a max y el índice de esta repetición como argumento de la subtarea PonerX.

Ideas involucradas: parámetros, repetición, subtareas

Posible solución:

```
procedure Progresion (min,max,dir, col)
{
 repeatWith n in min..max
 {
  PonerX (n, col)
  Mover (dir)
 }
}
```



Progresion (1,5, Norte, Verde)

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Enunciado: Escribir un procedimiento PonerTodosLosColores() que, usando repeatWith, deposite una bolita de cada color en la celda actual. Recuerde que minColor() devuelve el mínimo color del rango de colores y que maxColor() devuelve el máximo de ese rango.

Ideas involucradas: repetición indexada

Enunciado: Escribir un procedimiento PonerXDeCadaUna(n) que usando un repeatWith sobre los colores y la subtarea PonerX, pone "n" bolitas de cada color. ¿Puede utilizar este procedimiento para definir PonerTodosLosColores()? ¿Cómo?

Ideas involucradas: repetición indexada, parámetros

Enunciado: Cambiar el procedimiento Progresión para que en vez de usar "PonerX" utililce un procedimiento llamado "PonerXDeCada(cant)" que toma una cantidad y pone esa cantidad de bolitas de cada color posible.

Ideas involucradas: parámetros, repetición, subtareas

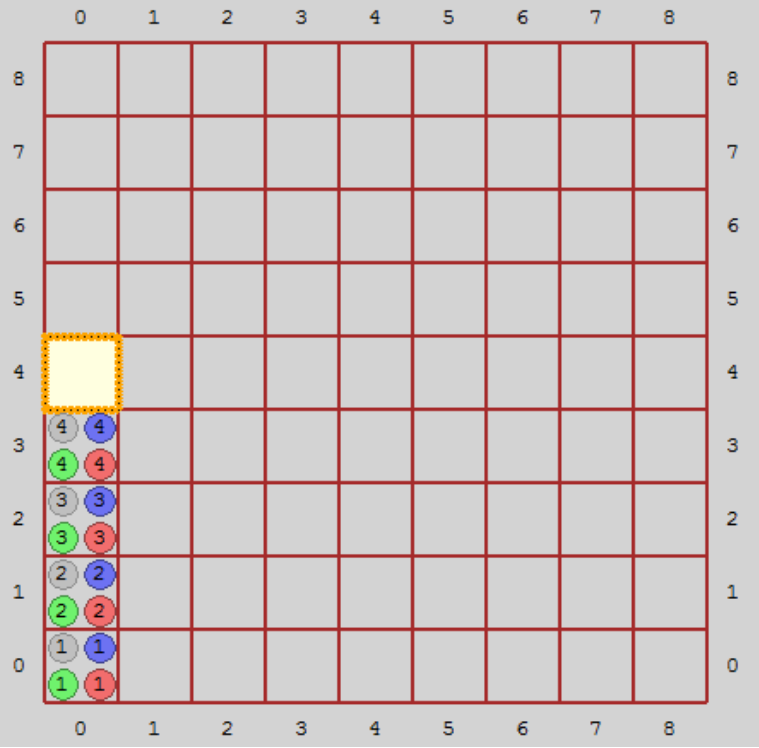
Posible solución:

```
procedure PonerXDeCada (cant)
{
 repeatWith i in 1..cant
 {
  Poner (Rojo)
  Poner (Verde)
  Poner (Azul)
  Poner (Negro)
 }
```

Downloaded by Ian Chen (chenian317@gmail.com)

```
}

procedure ProgresionMulticolor (min,max,dir)
{
 repeatWith n in min..max
 {
  PonerXDeCada (n)
  Mover (dir)
 }
}
```

ProgresionMulticolor(1,4,Norte)



Enunciado: Definir el procedimiento VaciarCeldaDe(color), que saque todas las bolitas de un determinado color que haya en la celda actual. Recuerdo que nroBolitas(color) devuelve el número de bolitas que hay para un determinado color en la celda actual.

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Ideas involucradas: parámetros, repetición indexada

Enunciado: Escribir un procedimiento BolitasEnLindantes() que, usando repeatWith, deposite una bolita de un color dado en cada una de las celdas lindantes (al Norte, Este, Sur, Oeste) de la celda actual. Explicitar la precondición de este procedimiento en forma de comentario. Recuerde que minDir() devuelve la mínima dirección del rango de direcciones y maxDir() devuelve la máxima dirección de ese rango.

Ideas involucradas: parámetros, repetición indexada, subtareas

Enunciado: Definir el procedimiento MoverSiSePuede(dir) que dada una dirección se mueva hacia ella sólo si existe una celda en dicha dirección. Dicha condición puede comprobarse con Íla función puedeMover(dir) que dada una dirección dice si es posible moverse hacia ella. Esa función ya existe en Gobstones.

Ideas involucradas: parámetros, alternativa condicional, funciones primitivas

Enunciado: Definir el procedimiento SacarSiHay(color), que saque una bolita de ese color sólo en el caso de haber al menos una.¿Este procedimiento posee precondiciones? ¿Por qué?

Ideas involucradas: parámetros, alternativa condicional, precondiciones, funciones primitivas

Enunciado: Definir el procedimiento PonerVerdeSiHayRojaYAzul() que ponga una bolita de color verde si existe al menos una bolita de color rojo y al menos una de color azul en la celda actual.

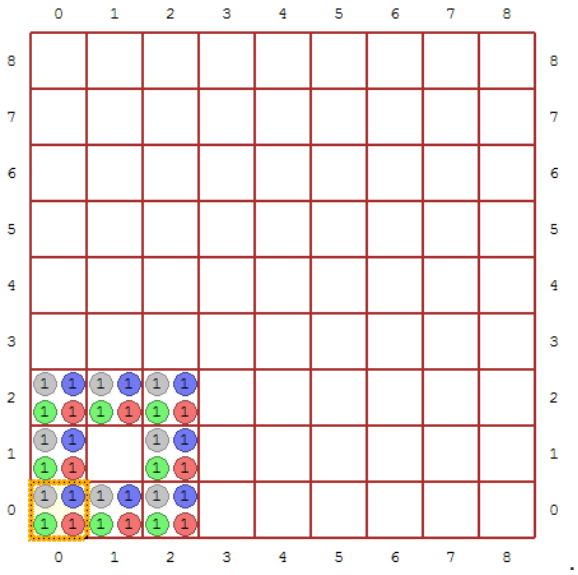
Ideas involucradas: alternativa condicional, expresiones booleanas, funciones primitivas

Downloaded by Ian Chen (chenian317@gmail.com)

Enunciado: Definir el procedimiento Poner20BolitasRojasSiHayMenosDe35Azules(), que en el caso que haya menos de 35 bolitas azules en la celda actual coloque 20 bolitas rojas. De yapa (no es obligatorio) puede definir PonerXBolitasDeC1SiHayMenosDeYBolitasDeC2(x, c1, y, c2).

Ideas involucradas: alternativa condicional, expresiones booleanas, funciones primitivas

Enunciado: Definir el procedimiento PerímetroCuadradoMulticolor(tam) utilizando un repetWith sobre los colores. Como ejemplo, PerímetroCuadradoMulticolor(3) define el siguiente perímetro cuadrado:

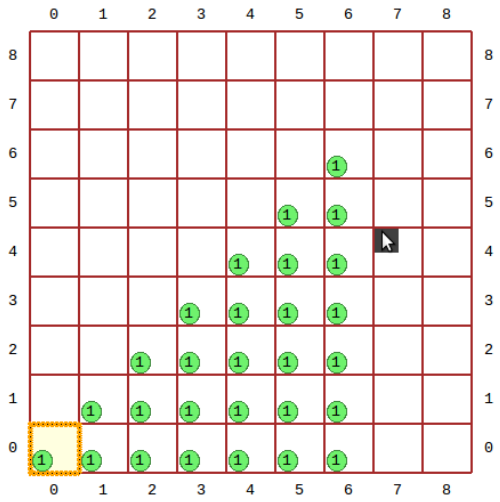


Ideas involucradas: parámetros, subtareas, repetición indexada

Enunciado: Escribir el procedimiento ColumnasEnEscalera(tam, color) que utiliza el índice del repeatWith para dibujar columnas en las que su tamaño se va incrementando. Recuerde reutililzar procedimientos ya vistos (se recomienda utililzar MoverX para volver hacia abajo luego de dibujar una columna). Por ejemplo, ColumnasEnEscalera(7, Verde) resulta en el siguiente tablero:

This document is available free of charge on studocu

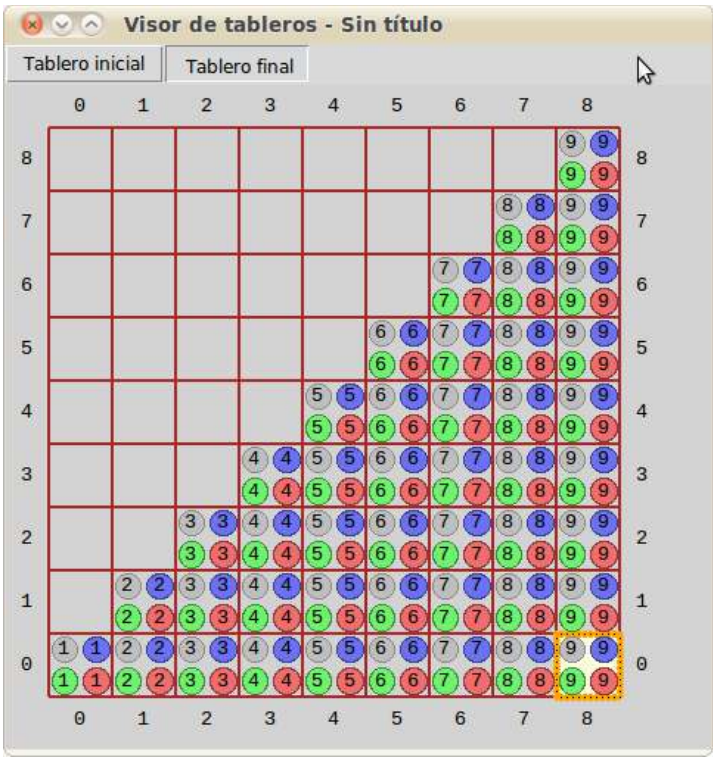
Downloaded by Ian Chen (chenian317@gmail.com)



Ideas involucradas: parámetros, subtareas, repetición indexada

Enunciado: Definir el procedimiento EscaleraMulticolor(tam) que dibuje una escalera como la de la imagen al final del enunciado. En este caso el ejemplo hace uso de EscaleraMulticolor(9). Observe que tanto el alto de la columna como la cantidad de bolitas por celda de la columna va incrementándose. La pista está en utilizar en utilizar PonerXDeCadaUna(num).

Downloaded by Ian Chen (chenian317@gmail.com)



Ideas involucradas: parámetros, subtareas, repetición indexada

Enunciado: Escribir un procedimiento que duplique la cantidad de bolitas de la celda actual.

Ideas involucradas: subtareas, repetición indexada, funciones primitivas

Enunciado: Definir el procedimieto CambiarRojasPorVerdes que coloque exactamente tantas bolitas verdes como la cantidad de bolitas rojas haya, y luego saque todas las bolitas rojas. Puede utilizar la función nroBolitas(color) para saber la cantidad de bolitas que hay de un color determinado.

Ideas involucradas: subtareas, repetición indexada, funciones primitivas

Enunciado: Definir el procedimiento CambiarC1PorC2(c1, c2), que parametriza el ejercicio

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

anterior (CambiarRojasPorVerdes) de tal manera que dados dos colores c1 y c2, se colocan tantas bolitas de color c2 como bolitas de c1 haya, y luego saque todas las bolitas de color c1.

Ideas involucradas: parámetros, subtareas, repetición indexada, funciones primitivas

Enunciado: Definir el procedimiento IrAlBorde(dir), que se mueva hacia “dir” mientras pueda moverse hacia esa dirección.

Ideas involucradas: parámetros, repetición condicional

Enunciado: Escribir un procedimiento que deposite una bolita verde en la primera celda que esté al Norte de la celda actual y que tenga bolitas rojas.

Ideas involucradas: repetición condicional

Enunciado: Escribir un procedimiento que ponga una bolita verde en cada una de las celdas de la columna actual.

Ideas involucradas: repetición condicional, recorridos

Enunciado: Escribir un procedimiento que coloque una bolita verde en todas las celdas al Norte de la celda actual. Puede asumir que el tablero se encuentra vacío.

Ideas involucradas: repetición condicional, recorridos

Enunciado: Escribir un procedimiento que ponga una bolita verde en cada celda vacía del tablero.

Ideas involucradas: repetición condicional, alternativa condicional, recorridos

Downloaded by Ian Chen (chenian317@gmail.com)

Enunciado: Escribir una función hayBolitaAlDeColor, que reciba una dirección dir y un color col y devuelva True si hay bolitas de color col en la celda lindante en la dirección dir.

Ideas involucradas: funciones, parámetros

Enunciado: Escribir una función que reciba una dirección y devuelva la cantidad de celdas hasta el final del tablero en esa dirección.

Ideas involucradas: funciones, parámetros, recorridos

Enunciado: Escribir una función que devuelva el color del que haya más bolitas en la celda actual.

Ideas involucradas: funciones, repetición indexada

Enunciado: Escribir una función que devuelva la dirección en la que haya más bolitas rojas (revisando las bolitas de las celdas lindantes)

Ideas involucradas: funciones, repetición indexada

Enunciado: Utilizando funciones ya definidas, escribir un procedimiento que mueva el cabezal en la dirección en la que haya más bolitas rojas.

Ideas involucradas: funciones

Enunciado: Utilizando funciones ya definidas, escribir un procedimiento que elimine de la celda actual las bolitas del color que haya mayor cantidad.

Ideas involucradas: funciones

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Enunciado: Utilizando funciones ya definidas, escribir un procedimiento que iguale la cantidad de bolitas de todos los colores en la celda actual, agregando bolitas de los distintos colores hasta completar la cantidad de la que tiene mayor ocurrencias.

Ideas involucradas: funciones

Enunciado: Aprovechando las funciones y/o procedimientos anteriores, escribir un procedimiento que mueva el cabezal en la dirección en la que haya más bolitas rojas y repita esa acción tantas veces como sea necesario mientras alguna de las celdas lindantes tenga más bolitas rojas que aquella en la que está posicionado el cabezal.

Ideas involucradas: funciones, subtareas

Enunciado: Escribir una función que retorne la cantidad de bolitas verdes en el tablero.

Ideas involucradas: funciones, recorridos, variables

Enunciado: Escribir una función que retorne el ancho y alto del tablero. ¿Cómo puede calcular la cantidad celdas?

Ideas involucradas: funciones, recorridos

Enunciado: Escriba una función que retorne las coordenadas de la celda donde se posiciona el cabezal.

Ideas involucradas: funciones

Enunciado: Escribir un procedimiento que reciba dos parámetros (X,Y) y mueva el cabezal a la celda con coordenadas X e Y. Asuma que la coordenada X igual a 0 es la primera columna

Downloaded by Ian Chen (chenian317@gmail.com)

y la coordenada Y igual a 0 es la primera fila.

Ideas involucradas: funciones

Enunciado: Escribir un procedimiento que dadas dos coordenadas, copie las bolitas de la celda de la primera coordenada a la celda de la segunda coordenada.

Ideas involucradas: variables

Enunciado: Escribir una función que devuelva las coordenadas de la celda que tiene más bolitas rojas en el tablero.

Ideas involucradas: recorridos, variables, funciones

Enunciado: Escribir un procedimiento MoverNAbsoluto que reciba un número n y una dirección dir y que mueva el cabezal de la siguiente forma: si el número es positivo, mueve el cabezal n lugares hacia la dirección dir y si es negativo, mueve el cabezal n lugares hacia la dirección apuesta a dir.

Ideas involucradas: funciones, parámetros, subtareas, alternativa condicional

Enunciado: Realizar un procedimiento que cuente la cantidad de bolitas que haya en el tablero de un color dado.

Ideas involucradas: funciones, variables, recorridos

El Tablero de la Batalla Naval

Se implementarán en el lenguaje Gobstones partes del juego Batalla Naval (simplificado). Las reglas de este juego son las siguientes:

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

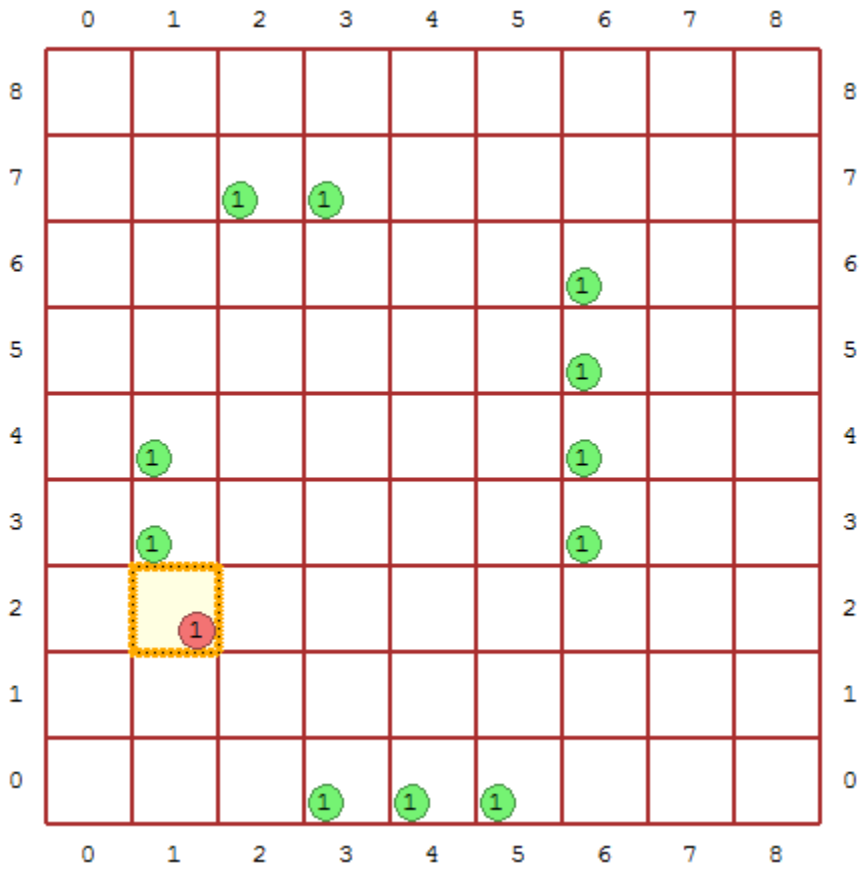
- En el tablero hay barcos en distintas partes
- Existen tres tamaños de barcos: pequeño (2 celdas), mediano (3 celdas) y grande (4 celdas).
- Los barcos son de color verde.
- Los barcos pueden estar en posición vertical u horizontal dentro del mapa.
- Existe una "mira" apuntando siempre a una coordenada dentro del mapa (el mapa en Gobstones sería el tablero y la mira simplemente el cabezal).
- La mira puede moverse una cantidad arbitraria de veces en una dirección.
- La mira puede disparar. Si en la celda en la que dispara la mira hay un pedazo de un barco (una bolita verde en la celda en donde se encuentra) entonces destruye esa parte del barco (saca la bolita verde y la reemplaza por una de color rojo). Si en la celda en donde dispara no hay un barco (celda que no tiene una bolita verde), entonces el disparo falla y no pasa nada.

Para implementar este juego deberá programar los siguientes procedimientos y funciones (en el orden sugerido):

1. Definir el procedimiento DibujarBarco(tam, dir), que dibuja un barco de tamaño *tam* hacia la dirección *dir.* Recuerde que los barcos son de color verde.
2. Definir el procedimiento MoverMira(cant, dir), que mueve la mira hacia la dirección *dir* una cantidad *cant* de veces. MoverMira puede caerse del tablero si no hay suficientes celdas hacia la dirección a la que se mueve, ¿cuál es entonces su precondición?
3. Definir el procedimiento Disparar(), que saca una parte del barco en caso de haber una en la celda actual y poner una bolita roja en su reemplazo (recuerde que una parte del barco simplemente es una bolita de color verde). Para preguntar si la mira está parada o no en una parte del barco utilizar la función "hayBolitas".
4. Definir el procedimiento LocalizarBarcoHacia(dir), que mueve la mira hacia la dirección *dir* MIENTRAS NO haya encontrado un barco. Nuevamente tenga en cuenta utilizar la función "hayBolitas" y en este caso también la función "not".

Una posible situación de este juego es la siguiente:

Downloaded by Ian Chen (chenian317@gmail.com)



Las distintas líneas verdes representan los barcos. El cabezal disparó a una parte de un barco, y hay una bolita roja, porque la mira apuntaba a una parte de este y se reemplazo la bolita verde por esta roja. Es conveniente modelar con el mouse una situación inicial y probar en Main los distintos procedimientos para ver su comportamiento.

GPS: Gobstones Positional System (autor: Elias Filipponi, Universidad Nacional de Quilmes)

Un barco recibe imágenes satelitales simples de la zona donde se encuentra. La imagen es como un mapa que consta de colores que representan tierra (verde), agua (azul), su barco (negro) y otros barcos (rojo).

Al dueño del barco le gustaría tener un radar más inteligente que le resuelva tareas comunes, y por eso ha recurrido al mejor programador que conoce: Usted B-).

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Imagine que los pixeles (o sea, los puntitos que conforman la imagen satelital) son las celdas de nuestro tablero. Imagine también que ya existe el procedimiento EncontrarMiBarco() que deja parado al cabezal donde esta la única bolita negra del tablero (que representa su barco).

Se pide programar:

1. La función hayAguaAl(dir) que indique si hay o no agua en la celda vecina a su barco en la dirección dir. Para ello deberá usar el procedimiento EncontrarMiBarco(), moverse en la dirección indicada, y luego devolver si hay o no agua (o sea, azules).
2. La función hayBarcoAl(dir) que indique si hay o no un barco en la celda vecina a su barco en la dirección dir. Para ello deberá usar el procedimiento EncontrarMiBarco(), moverse en la dirección indicada, y luego devolver si hay o no un barco (o sea, rojas).
3. La función distanciaATierraHaciaEl(dir) que devuelva la distancia que hay desde su barco hasta la tierra más cercana en la dirección dir. Deberá usar una variable para contar esa distancia, cuyo valor irá actualizando con cada paso que dé, hasta encontrar tierra.
4. La función hayPeligroDeChoque() que verifica si hay un barco cerca del suyo. Para ello, encuentre el suyo, y luego use hayBarcoAl(dir) para ver si hay barcos en cada dirección.
5. El procedimiento EncontrarMiBarco() que nos ubique donde esté nuestro barco (o sea, donde haya una bolita negra). Puede asumir que siempre hay solo un barco en el napa.
6. La función hayAguaEnTodasDirecciones() que usando la función hayAguaAl(dir) devuelva True cuando haya agua en las cuatro celdas vecinas a la celda actual.

La Búsqueda del Tesoro (posible exámen) (autor: Elias Filipponi, Universidad Nacional de Quilmes)

La búsqueda del tesoro es un juego sencillo en el que se esconden pistas en un parque. La primer pista conduce a la segunda pista, la segunda pista a la tercera y así hasta la última que conduce hacia el tesoro.
Vamos a implementar ese juego en Gobstones. El tablero representará nuestro “parque”. Las “pistas” estarán representadas en celdas con el código que aparece en instructivo de códigos. Véalo!

Downloaded by Ian Chen (chenian317@gmail.com)

La idea es que el programa que haremos lea las pistas y mueva el cabezal según esas pistas hasta encontrar el tesoro. Consideraremos que hallamos el tesoro cuando una pista nos conduzca a una celda que tenga bolitas del color del tesoro (vea el instructivo de códigos). No se asuste: iremos de a poco.

1. Defina el procedimiento MoverN(dir, n) que dada la dirección dir y el número n, mueva el cabezal n veces en la dirección dir.
2. Defina la función cantidadDePasosADar() que obtenga el número de pasos que debemos dar a partir de la cantidad de bolitas del color que representa los pasos (vea el instructivo de códigos). ¿Qué tipo de dato devuelve esta función?
3. Defina la función direccionDecodificada() que devuelva la dirección que corresponda según el número de bolitas del color que corresponda (vea el instructivo de códigos). Para ello, necesitará usar varios if y una variable. ¿Qué tipo de dato devuelve esta función?
4. Defina el procedimiento SeguirPista() que, interpretando a la celda actual como una pista, mueva el cabezal según esa pista. Ayuda: sólo necesita todo lo que programó anteriormente. Este procedimiento demanda una sola línea.
5. Defina la función encontreElTesoro() que devuelve verdadero o falso dependiendo de si encontré o no el tesoro, según haya o no bolitas del color que lo representa en la celda actual. (vea el instructivo de códigos) ¿Qué tipo de dato devuelve esta función?
6. Defina el procedimiento BuscarTesoro() que siga las pistas, una por una, hasta encontrar el tesoro... o debería decir... mientras no lo encuentre... ;)

El instructivo con los códigos es:

CÓDIGO DE LAS PISTAS					
Nº de Verdes	Dirección				
1	Norte		Cantidad de Pasos:	---->	Nº de Negras
2	Este				
3	Sur				
4	Oeste		Color del tesoro:	---->	Rojo

This document is available free of charge on studocu

Downloaded by Ian Chen (chenian317@gmail.com)

Downloaded by Ian Chen (chenian317@gmail.com)