function kiosco(){}

function almacen(){}

function restaurante(){}

function jugueteria(){}

procedure AgruparTiendasHogareñasBajoColor\_(color){

/\* Proposito: Agrupar en cada sector del centro comercial las tiendas hogareñas

bajo el color \*color\* y si no hay tiendas hogareñas convertir

el sector en un shopping

Precondición: Ninguna

Parametros: \*Color\* - color - representa el color bajo el que se agrupan

las tiendas hogareñas.

Observación: es un recorrido de procesamiento sobre los sectores del centro

comercial, agrupando tiendas hogareñas o convirtiendo en shopping

- las tiendas hogareñas son los kioscos y los almacenes \*/

IrAlInicioDeUnRecorridoAl\_Y\_(Este, Norte)

AgruparTiendasHogareñasBajoColor\_OConvertirEnShopping(color)

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

AgruparTiendasHogareñasBajoColor\_OConvertirEnShopping(color)

}

}

procedure AgruparTiendasHogareñasBajoColor\_OConvertirEnShopping(color){

/\* Proposito: Agrupar en el sector actual las tiendas hogareñas

bajo el color \*color\* y si no hay tiendas hogareñas convertir en un shopping

Precondición: Ninguna

Parametros: \*color\* - color - representa el color bajo el que se agrupan

las tiendas hogareñas.

Observación: las tiendas hogareñas son los kioscos y los almacenes \*/

if(hayTiendasHogareñasAcá()){

AgruparTiendasHogareñasBajoColor\_Acá(color)

}else{

ConvertirEnShopping()

}

}

function hayTiendasHogareñasAcá(){

/\* Proposito: Indica si en el sector actual hay al menos un kiosco o un almacen

Precondicion: ninguna

Tipo: Booleano \*/

return (hayBolitas(kiosco()) || hayBolitas(almacen()))

}

procedure AgruparTiendasHogareñasBajoColor\_Acá(color){

/\* Proposito: Agrupar en el sector actual las tiendas hogareñas

bajo el color \*color\*

Precondicion: Tiene que existir al menos una tienda hogareña en el sector actual

Parametros: \*color\* - color - representa el color bajo el que se agrupan

las tiendas hogareñas. \*/

cantTiendasHogareñas := cantTiendasHogareñasAca()

VaciarCelda()

Poner\_DeColor\_(cantTiendasHogareñas, color)

}

function cantTiendasHogareñasAca(){

/\* Proposito: Describe la cantidad de tiendas hogareñas en el sector actual

Precondicon: ninguna

Tipo: Número \*/

return (nroBolitas(kiosco())+ nroBolitas(almacen()))

}

procedure ConvertirEnShopping(){

/\* Proposito: Convierte en shopping el sector actual

Precondicion: No debe existir tiendas hogareñas en el sector actual \*/

VaciarCelda()

Poner(Verde)

}

//---------------------------------------- Ejercicio 2 ------------------------------------------------

procedure IrACentroDeVentasEnColumnaDeTiendaDe\_(clase){

/\* Proposito: Ubica el cabezal en el centro de ventas de las tiendas de la clase \*clase\*

de la columna actual

Precondicion: Hay un centro de ventas de las tiendas del tipo dado en la columna actual

Parametro: \*clase\* -color - describe el color de la clase de tienda a buscar su centro de ventas

Observación: -un centro de ventas es el sector que agrupa la mayor cantidad de tiendas

del tipo dado en la columna, agrupando la cantidad de tiendas de si mismo

y las de los sectores lindantes en las 4 direcciones ortogonales y si es que hubiese

- es un recorrido de busqueda sobre la columna actual, buscando el centro de venta de la clase dada \*/

IrAlBorde(Sur)

while(not esCentroDeVentaDeTiendaDe\_(clase)){

Mover(Norte)

}

}

function esCentroDeVentaDeTiendaDe\_(clase){

/\* Proposito: Indica si el sector actual es el centro de ventas de tiendas de la clase \*clase\*

Precondicion: Ninguna

Parametro: \*clase\* -color - describe el color de la clase de tienda a buscar su centro de ventas

tipo: Booleano \*/

return (cantTiendasAgrupadasDe\_Aca(clase) == cantTiendasDeCentroDeVetasDe\_EnColumna(clase))

}

function cantTiendasAgrupadasDe\_Aca(clase){

/\* Proposito: Describe la cantidad de tiendas agrupadas de la clase \*clase\* en el sector actual

Precondicion:Ninguna

Parametro:\*clase\* -color - describe el color de la clase de tienda a buscar su centro de ventas

tipo: Número

Observación: es un recorrido de acumulación sobre los sectores lindantes al sector actual \*/

cantTiendasVistas:= cantTiendasDe\_Aca(clase)

dirActual:= minDir()

while(dirActual/=maxDir){

cantTiendasVistas:= cantTiendasVistas + cantTiendasDe\_Al\_(clase, dirActual)

dirActual:= siguiente(dirActual)

}

return(cantTiendasVistas + cantTiendasDe\_Al\_(clase, dirActual)

}

function cantTiendasDe\_Aca(clase){

/\* Proposito: Describe la cantidad de tiendas de clase \*clase\* en el sector actual

Precondicion: Ninguna

Parametro:\*clase\* -color - describe el color de la clase de tienda a buscar su centro de ventas

Tipo: Número \*/

return (nroBolitas(clase))

}

function cantTiendasDe\_Al\_(clase, direccion){

/\* Proposito: Describe la cantidad de tiendas de clase \*clase\* en el sector lindante hacia la direccion \*direccion\*,

si no hay un sector lindante hacia la direccion dada describe 0

Precondicion:NINGUNA

Parametro:\*clase\* -Color - describe el color de la clase de tienda a buscar su centro de ventas

- \*direccion\* - Direccion - Direccion hacia donde sensar

Tipo: Número \*/

return (choose cantTotalTiendasDe\_Al\_(clase, direccion) when (puedeMover(direccion))

0 otherwise )

}

function cantTotalTiendasDe\_Al\_(clase, direccion){

/\*

Proposito: Describe la cantidad de tiendas de clase \*clase\* en el sector lindante hacia la direccion \*direccion\*

Precondicion:Debe existir al menos un sector hacia la direccion dada

Parametro:\*clase\* -Color - describe el color de la clase de tienda a buscar su centro de ventas

- \*direccion\* - Direccion - Direccion hacia donde sensar

Tipo: Número

\*/

Mover(direccion)

return (cantTiendasDe\_Aca(clase))

}

function cantTiendasDeCentroDeVetasDe\_EnColumna(clase){

/\*

Proposito: describe la cantidad de tiendas del centro de venta de tiendas de clase \*clase\* en la columna actual

Precondicion: Debe existir al menos un centro de ventas de la clase dada en la columna actual

Parametro:\*clase\* -Color - describe el color de la clase de tienda a buscar su centro de ventas

Observación: Es un recorrido de maximo/minimo sobre la celda de la columna actual, buscando

el centro de venta de tiendas de la clase dada

\*/

IrAlBorde(Sur)

cantMaximaTiendasAgrupadas:= cantTiendasAgrupadasDe\_Aca(clase)

while(puedeMover(Norte)){

Mover(Norte)

cantMaximaTiendasAgrupadas:= maximoEntre\_Y\_(cantMaximaTiendasAgrupadas, cantTiendasAgrupadasDe\_Aca(clase))

}

return (cantMaximaTiendasAgrupadas)

}

procedure IrANésimaVacía\_(nnn) {

/\* Proposito

Mueve el cabezal a la celda vacia numero \*n\*, si no hay suficientes celdas vacias

deja el cabezal en el borde Noreste

Precondicion

ninguna

Observaciones

es es un reccorido de busqueda sobre las celdas del tablero, que busca la celda

vacia numero \*n\* \*/

celdasVacíasYaVistas := 0

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

celdasVacíasYaVistas := celdasVacíasYaVistas + unoSi\_CeroSino(esCeldaVacía())

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este,Norte) && celdasVacíasYaVistas /= nnn){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

celdasVacíasYaVistas := celdasVacíasYaVistas + unoSi\_CeroSino(esCeldaVacía())

}

}