//////////////////PONER//////////////////

**procedure Poner\_DeColor\_(cantidadAPoner, colorAPoner)** { //guía 4 - parametros

/\* PROPOSITO: Pone \*cantidadAPoner\* bolitas de color \*colorAPoner\* en la celda actual.

PARAMETROS: \*cantidadAPoner: Numero - La cantidad de bolitas a poner.

\*colorAPoner: Color - El color de las bolitas a poner.

PRECONDICIONES: Ninguna. \*/

repeat (cantidadAPoner) {

Poner(colorAPoner)

}

}

//////////////////SACAR//////////////////

**procedure Sacar\_DeColor\_(cantidadASacar, colorASacar)** { //guía 4 - parametros

/\* PROPOSITO: Saca \*cantidadASacar\* bolitas de color \*colorASacar\* de la celda actual.

PARAMETROS: \*cantidadASacar: Numero - La cantidad de bolitas a sacar.

\*colorASacar: Color - El color de las bolitas a sacar.

PRECONDICIONES: Debe haber al menos \*cantidadASacar\* bolitas de color \*colorASacar\*

en la celda actual. \*/

repeat (cantidadASacar) {

Sacar(colorASacar)

}

}

**procedure SacarTodasLasDeColor\_(colorASacar)** { //Guia 5 - Expresiones y tipos

/\* PROPOSITO: Saca de la celda actual todas las bolitas de color \*\*colorASacar\*\*.

PARAMETROS: \*colorASacar: Color - El color de las bolitas a sacar.

PRECONDICIONES: Ninguna\*/

while (hayBolitas(colorASacar)) {

Sacar(colorASacar)

}

}

procedure VaciarCelda(){

/\*PROPÓSITO: Vaciar la celda actual.

PRECONDICIONES: Ninguna.\*/

SacarTodasLasDeColor\_(Rojo)

SacarTodasLasDeColor\_(Verde)

SacarTodasLasDeColor\_(Azul)

SacarTodasLasDeColor\_(Negro)

}

//////////////////MOVER//////////////////

procedure Mover\_VecesAl\_(cantidadAMover, direcciónAMover) { //Guia 4 - Parametros

/\*PROPOSITO: Mueve el cabezal \*cantidadAMover\* posiciones en direccion \*direccionAMover\* de la celda actual. El cabezal termina en dicha celda.

PARAMETROS: \*cantidadAMover: Numero - La cantidad de celdas a mover.

\*direcciónAMover: Direccion - El sentido a mover del cabezal.

PRECONDICIONES: \*Debe haber al menos \*cantidadAMover\* celdas hacia \*direccionAMover\* de la celda actual. \*/

repeat(cantidadAMover) {

Mover(direcciónAMover)

}

}

/////////////////////////// CONDICIONALES //////////////////////////

//////////////////PONER//////////////////

procedure Poner\_Si\_(color, condición) { //Guia 6 - Alternativa condicional

/\*PROPOSITO: Pone una bolita de color \*color\* en la celda actual si se cumple la condicion \*condicion\*

PRECONDICIONES: Ninguna.

PARAMETROS:

\*color: Color - El color de la bolita a poner.

\*condicion: Booleano - La condicion que se debe cumplir para poner una bolita \*/

if(condicion){

Poner(color)

}

}

//////////////////SACAR//////////////////

procedure Sacar\_Si\_(color, condición) { //Guia 6 - Alternativa condicional

/\*PROPOSITO: Saca una bolita de color \*color\* en la celda actual si se cumple la condicion \*condicion\*

PRECONDICIONES: Debe haber al menos una bolita de \*\*color\*\* si se cumple la condicion \*\*condicion\*\*

PARAMETROS:

\*color: Color - El color de la bolita a poner.

\*condicion: Booleano - La condicion que se debe cumplir para poner una bolita. \*/

if(condicion){

Sacar(color)

}

}

//////////////////MOVER//////////////////

procedure Mover\_Si\_(dirección, condición) { //Guia 6 - Alternativa condicional

/\*PROPOSITO: Se mueve el cabezal en direccion \*\*direccion\*\* si se cumple la

condicion \*\*condicion\*\*.

PARAMETROS: \*dirección: Direccion - La direccion a la que se movera el cabezal.

\*condición: Booleano - La condicion que se debe cumplir para mover el cabezal.

PRECONDICIONES:

\*Debe haber al menos una celda en direccion \*\*direccion\*\* si se cumple la condicion

\*\*condicion\*\*. \*/

if(condición){

Mover(dirección)

}

}

////////////////// FUNCIONES BOOLEANAS //////////////////

function esCeldaVacía() { //Guia 7 - Funciones simples

/\*

PROPÓSITO: Indica si al celda actual se encuentra vacía.

PRECONDICIONES: Ninguna.

TIPO: Booleano.

\*/

return (not (hayBolitas(Azul) || hayBolitas(Negro) || hayBolitas(Rojo) || hayBolitas(Verde)))

}

function tieneUnaDeCada() { //Guia 7 - Funciones simples

/\*

PROPÓSITO: Indica si la celda actual tiene al menos una bolita de cada color.

PRECONDICIONES: Ninguna.

TIPO: Booleano.

\*/

return (hayBolitas(Azul) && hayBolitas(Negro) && hayBolitas(Rojo) && hayBolitas(Verde))

}

function esCeldaConBolitas() { //Guia 7 - Funciones simples

/\*

PROPÓSITO: Indica si la celda actual tiene al menos una bolita, de cualquier color.

PRECONDICIONES: Ninguna.

TIPO: Booleano.

\*/

return (not esCeldaVacía())

}

//////////////////REPETICIÓN CONDICIONAL//////////////////

procedure IrAlBorde\_(dirección) { //Guia 8

/\*PROPÓSITO: Llevar el cabezal al borde \*\*dirección\*\*.

PARÁMETROS:

\*dirección: Dirección - La dirección del borde.

PRECONDICIONES: Ninguna. \*/

while (puedeMover(dirección)) {

Mover(dirección)

}

}

/////////////////// PARA RECORRIDO POR CELDAS //////////////////

procedure IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria) { //Guia 8

/\*PROPOSITO: Mueve el cabezal a la primera celda del tablero en dirección

\*\*dirPrincipal\*\* y en direccion \*\*dirSecundaria\*\* desde la celda actual.

PARAMETROS:

\*dirPrincipal: Direccion - La direccion principal a la que se movera el cabezal.

\*dirSecundaria: Direccion - La direccion secundaria a la que se movera el cabezal.

PRECONDICIONES:

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser iguales.

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser opuestas. \*/

IrAlBorde\_(opuesto(dirPrincipal))

IrAlBorde\_(opuesto(dirSecundaria))

}

function haySiguienteCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria) { //Guia 8

/\*PROPOSITO: Indica si hay siguiente celda en direccion \*\*dirPrincipal\*\* y en direccion

\*\*dirSecundaria\*\* desde la celda actual.

PARAMETROS:

\*dirPrincipal: Direccion - La direccion principal a la que se movera el cabezal.

\*dirSecundaria: Direccion - La direccion secundaria a la que se movera el cabezal.

PRECONDICIONES:

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser iguales.

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser opuestas. \*/

return (puedeMover(dirPrincipal) || puedeMover(dirSecundaria))

}

procedure IrASiguienteCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria) {

/\*PROPOSITO: Mueve el cabezal a la siguiente celda desde la celda actual.

PARAMETROS:

\*dirPrincipal: Direccion - La direccion principal a la que se movera el cabezal.

\*dirSecundaria: Direccion - La direccion secundaria a la que se movera el cabezal.

PRECONDICIONES:

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser iguales.

\*Las direcciones \*\*dirPrincipal\*\* y \*\*dirSecundaria\*\* no deben ser opuestas. \*/

if (puedeMover(dirPrincipal)) {

Mover(dirPrincipal)

} else {

IrAlBorde\_(opuesto(dirPrincipal))

Mover(dirSecundaria) } }

//////////////////FUNCIONES CON PROCESAMIENTO//////////////////

function tieneBolitas\_Al\_(colorAMirar,direcciónAMirar){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Indica si la celda lindante hacia direcciónAMirar tiene bolitas

del color colorAMirar

PARÁMETROS:

\* colorAMirar: Color - El color de las bolitas.

\* direcciónAMirar: Dirección - La dirección de la celda lindante.

PRECONDICIONES: Debe haber al menos una celda lindante hacia direcciónAMirar

TIPO: Booleano. \*/

Mover(direcciónAMirar)

return (hayBolitas(colorAMirar))

}

function hayBolitas\_Al\_(colorAMirar,direcciónAMirar){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*

PROPÓSITO: Indica si hay una celda lindante hacia direcciónAMirar y la misma

tiene bolitas del color colorAMirar

PARÁMETROS:

\* colorAMirar: Color - El color de las bolitas.

\* direcciónAMirar: Dirección - La dirección de la celda lindante.

PRECONDICIONES: Ninguna.

TIPO: Booleano

\*/

return (puedeMover(direcciónAMirar) && tieneBolitas\_Al\_(colorAMirar,direcciónAMirar))

}

function hayBolitas\_Hacia\_(colorAMirar,direcciónABuscar){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Indica si en alguna de las celdas hacia direcciónABuscar hay bolitas

de color colorAMirar.

PARÁMETROS:

\* colorAMirar: Color - El color de las bolitas a mirar.

\* direcciónABuscar: Direccion - La dirección a buscar.

PRECONDICIONES: En la celda actual no hay bolitas de color colorAMirar.

TIPO: Booleano.\*/

while (not hayBolitas(colorAMirar) && puedeMover(direcciónABuscar)) {

Mover(direcciónABuscar)

}

return (hayBolitas(colorAMirar))

}

function distanciaAlBorde\_(direcciónDelBorde){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPOSITO: Describe la cantidad de celdas que hay entre la celda actual y el borde \*\*direccion\*\*

PARAMETRO:

\*direccion: Direccion - La direccion a la que se evaluara la distancia.

PRECONDICIONES: Ninguna.

TIPO: Numerico. \*/

cantidadDeCeldasContadas := 0

while (puedeMover(direcciónDelBorde)) {

Mover(direcciónDelBorde)

cantidadDeCeldasContadas := cantidadDeCeldasContadas + 1

}

return (cantidadDeCeldasContadas)

}

function coordenadaX(){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la coordenada de la columna actual.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (distanciaAlBorde\_(Oeste))

}

function coordenadaY(){ //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la coordenada de la fila actual.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (distanciaAlBorde\_(Sur))

}

function nroFilas() { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la cantidad de filas del tablero.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (distanciaAlBorde\_(Sur) + distanciaAlBorde\_(Norte) + 1)

}

function nroColumnas() { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la cantidad de columnas del tablero.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (distanciaAlBorde\_(Oeste) + distanciaAlBorde\_(Este) + 1)

}

function nroBolitasTotalDeColor\_(colorAContar) { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la cantidad de bolitas de color colorAContar que hay en el tablero.

PARÁMETROS:

\* colorAContar: Color - El color de las bolitas a contar.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

cantidadDeBolitasContadas := 0

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

cantidadDeBolitasContadas := cantidadDeBolitasContadas + nroBolitas(colorAContar)

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

}

cantidadDeBolitasContadas := cantidadDeBolitasContadas + nroBolitas(colorAContar)

return (cantidadDeBolitasContadas)

}

function nroVacías() { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe la cantidad de celdas vacías que hay en total en el tablero.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

cantidadDeVacíasContadas := cantidadDeVacíasContadasAcá()

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

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

cantidadDeVacíasContadas := cantidadDeVacíasContadas + cantidadDeVacíasContadasAcá()

}

return (cantidadDeVacíasContadas)

}

function cantidadDeVacíasContadasAcá() {

/\*PROPÓSITO: Describe la cantidad de celdas vacías en la posición actual.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (

choose

1 when (esCeldaVacía())

0 otherwise

)

}

function cantidadDeBolitasTotales(){

/\*PROPÓSITO: Describe la cantidad de bolitas de todos los colores

que hay en la celda actual.

TIPO: Número.

PRECONDICIONES: Ninguna. \*/

return(nroBolitas(Azul) + nroBolitas(Negro) + nroBolitas(Rojo) + nroBolitas(Verde))

}

//////////////////ALTERNATIVA CONDICIONAL EN EXPRESIONES (CHOOSE)//////////////////

function mínimoEntre\_Y\_(valor1, valor2) { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe el valor más chico entre valor 1 y valor 2.

PARÁMETROS:

\*valor1: Dirección, número o color - El primer valor a comparar.

\*valor2: Dirección, número o color - El segundo valor a comparar.

PRECONDICIONES: Ninguna.

TIPO: Depende del tipo del valor que describa. \*/

return (

choose

valor1 when (valor1 <= valor2)

valor2 otherwise

)

}

function máximoEntre\_Y\_(valor1, valor2) { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe el valor más grande entre valor 1 y valor 2.

PARÁMETROS:

\*valor1: Dirección, número o color - El primer valor a comparar.

\*valor2: Dirección, número o color - El segundo valor a comparar.

PRECONDICIONES: Ninguna.

TIPO: Depende del tipo del valor que describa. \*/

return (

choose

valor1 when (valor1 >= valor2)

valor2 otherwise

)

}

function unoSi\_ceroSino(condiciónACumplir) { //GUIA 9 - VARIABLES Y FUNCIONES CON PROCESAMIENTO

/\*PROPÓSITO: Describe 1 cuando se cumple \*\*condición\*\* o 0 cuando no se cumple.

PARÁMETROS:

\*condiciónACumplir: Booleano - La condición a cumplir.

PRECONDICIONES: Ninguna.

TIPO: Número. \*/

return (

choose

1 when (condiciónACumplir)

0 otherwise

)

}

function elegir\_Si\_Sino\_(valor1, condiciónACumplir, valor2) {

/\*PROPÓSITO: Describe \*\*valor1\*\* cuando se cumple \*\*condiciónACumplir\*\* o \*\*valor2\*\*

cuando no se cumple.

PARÁMETROS:

\*valor1: Tipo del valor - El valor a elegir cuando se cumple \*\*condiciónACumplir\*\*

\*condiciónACumplir: Booleano - La condición a cumplir.

\*valor2: Tipo del valor - El valor a elegir cuando no se cumple \*\*condiciónACumplir\*\*

PRECONDICIONES: \*\*valor1\*\* y \*\*valor2\*\* son del mismo tipo.

TIPO: Número. \*/

return (

choose

valor1 when (condiciónACumplir)

valor2 otherwise)

}

//////////////////RECORRIDOS//////////////////

\*\*\*Por celdas

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //inicia el recorrido

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){ //mientras que (queden elementos por procesar)

procesa el elemento

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //pasa al siguiente elemento

}

Finaliza el recorrido

\*\*\*Por filas

IrAPrimeraFila //inicia el recorrido

while(queden elementos por procesar){ //mientras que (queden elementos por procesar)

procesa el elemento

IrAProximaFila //pasa al siguiente elemento

}

Finaliza el recorrido

\*\*\*De Busqueda

En caso de saber con seguridad de que existe el elemento:

IrAPrimeraCelda //inicia el recorrido

while(not hayElementoABuscar)){ //mientras que (no encontré lo que buscaba)

procesa el elemento

IrASiguienteCelda //pasa al siguiente elemento

}

En caso de no saber si existe el elemento para evitar un bucle infinito:

IrAPrimeraCelda //inicia el recorrido

while(not hayElementoABuscar && hayProximoElemento)){ //mientras que (no encontré lo que buscaba y )

IrASiguienteCelda //pasa al siguiente elemento

}

\*\*\*Acumulacion

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //inicia el recorrido

contador := 0 //inicia el contador con una variable

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){ //mientras que (queden elementos por procesar)

contador := contador + 1 //suma elementos al contador

procesa el elemento

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //pasa al siguiente elemento

}

Finaliza el recorrido

\*\*\*Por direcciones (es para cuando tengo que hacer cosas en todas las direcciones)

direccion := minDir() //Inicia el recorrido en este caso seria Norte

while(direccion /= maxDir()){ //mientras que dirección sea distinto a Oeste

procesa el elemento

direccion := siguiente(direccion) //pasa a la siguiente dirección (ej: norte a este)

}

Finaliza el recorrido

}