procedure **Poner\_DeColor\_(cantidadAPoner, colorAPoner)**{

/\*prop. Pone en el mismo casillero una cantidadAPoner de bolitas de un colorAPoner

parametros. cantidadAPoner : Numero , colorAPoner :Color

precondicion. ninguna\*/

repeat(cantidadAPoner){

Poner(colorAPoner)

}

}

procedure **PonerUnaDeCada()**{

/\*prop. Poner una bolita de cada color en la celda actual

pre. Ninguna. \*/

procedure PonerUnaDeCada() {

Poner(Rojo)

Poner(Azul)

Poner(Negro)

Poner(Verde)

}

procedure **Mover\_VecesAl\_(cantidadAMover , direccionAMover)**{

/\*prop. mueve al automata una cantidadAMover de casilla en una direccionAMover

parametro. cantidadAMover : numero , direccionAMover : direccion

precondicion. Tiene que haber al menos la misma cantidad de casillas de las que se quiere mover \*/

repeat(cantidadAMover){

Mover(direccionAMover)

}

}

procedure **Sacar\_DeColor\_(cantidadASacar, colorASacar)**{

/\*prop. Saca \*cantidad a sacar\*\* de \*colorASacar\* de la misma casilla

parametros . cantidadASacar : numero , colorASacar : color

prec. Tiene que haber al menos la misma cantidad de colores a sacar. \*/

repeat(cantidadASacar){

Sacar(colorASacar)

}

}

procedure **Poner\_Si\_(color,condicion)**{

/\*Prop.: poner una bolita de color \*\*color\*\* en la celda actual si es que \*\*condicion\*\* es verdadero. Si \*\*condicion\*\* es falso, no pondrá la bolita.

Precond.: \*\*condicion\*\* debe ser verdadero.

Parametros:

\*\*color: color - indica el color de la bolita a poner.

\*\*condicion: condición - indica la condición que debe cumplirse para ponerse la bolita.

\*/

if (condicion)

then{

Poner(color)

}

}

function **cantidadDeCeldasConBolitasDeColor\_**(color){

/\*Prop.: describe la cantidad de celdas que contienen al menos una bolita de color \*\*color\*\*.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*color: color - describe el color de la bolita.

\*/

cantidadCeldas:=0

IrALaPrimeraCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

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

cantidadCeldas:= cantidadCeldas + unoSi\_CeroSino(hayBolitas(color))

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

}

cantidadCeldas:= cantidadCeldas + unoSi\_CeroSino(hayBolitas(color))

return(cantidadCeldas)

}

procedure **IrALaPrimeraCeldaEnUnRecorridoAl\_Y\_(dir1, dir2)**{ //BIBLIOTECAAAA

/\*Prop.: lleva el cabezal a la celda más alejada a la dirección al \*\*opuesto(dir1)\*\* y \*\*opuesto(dir2)\*\*.

Precond.:

Parametros:

\*\*dir1: dirección - describe la primer dirección a la que se va a mover el cabezal.

\*\*dir2: dirección - describe la segunda dirección a la que se va a mover el cabezal.

\*/

IrAlBorde\_(opuesto(dir1))

IrAlBorde\_(opuesto(dir2))

}

procedure **IrAlBorde\_(direccion)**{

/\*Prop.: mover el cabezal hasta el borde en dirección al \*\*direccion\*\*

Precond.: ninguna.

Parametros:

\*\*direccion: dirección - describe la dirección a la que se va a mover el cabezal.

\*/

while(puedeMover(direccion)){

Mover(direccion)

}

}

function **haySiguienteCeldaEnUnRecorridoAl\_Y\_(dir1,dir2)**{ //BIBLIOTECAAAAAA

/\*Prop.: indica si hay una celda vecina al \*\*dir1\*\* o al \*\*dir2\*\*.

Precond.: ninguna.

Parametros:

\*\*dir1: direccion - describe la primer direccion a la que se va a mover el cabezal.

\*\*dir2: direccion - describe la segunda direccion a la que se va a mover el cabezal.

\*/

return(puedeMover(dir1) || puedeMover(dir2))

}

procedure **IrASiguienteCeldaEnUnRecorridoAl\_Y\_(dir1,dir2)**{ //BIBLIOTECAAAAA

/\*Prop.: mover el cabezal a la siguiente celda en un recorrido al \*\*dir1\*\* y \*\*dir2\*\* si es posible.

Precond.:

Parametros:

\*\*dir1: direccion - describe la primer direccion a la que va a moverse el cabezal.

\*\*dir2: direccion - describe la segunda direccion a la que se va mover el cabezal.

\*/

if(puedeMover(dir1))

then{

Mover(dir1)

}

else{

Mover(dir2)

IrAlBorde(opuesto(dir1))

}

}

function **unoSi\_CeroSino(condicion)**{

/\*Prop.: describe con un 1 si se cumple la condición \*\*condicion\*\* y con un 0 si no se cumple en la celda actual.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*condicion: booleano - indica la expresión a evaluar.

\*/

return(choose 1 when (condicion) 0 otherwise)

}

function **esCeldaVacia()**{

/\*Prop.: indica si la celda actual está vacía.

Precond.: ninguna.

Tipo: booleano.

\*/

return (cantidadTotalDeBolitas()==0)

}

function **cantidadTotalDeBolitas()**{

/\*Prop.: describe la cantidad total de bolitas que hay en la celda actual.

Precond.: ninguna.

Tipo: número.

\*/

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

}

function **coordenadaX()**{

/\*Prop.: describe la coordenada de la fila de la celda actual.

Precond:: ninguna.

Tipo: número.

\*/

return(distanciaAlBorde\_(Oeste))

}

function **coordenadaY()**{

/\*Prop.: describe la coordenada de la columna de la celda actual.

Precond:: ninguna.

Tipo: número.

\*/

return(distanciaAlBorde\_(Sur))

}

function **distanciaAlBorde\_(direccion)**{

/\*Prop.: describe la cantidad de celdas que hay hasta el borde en dirección al \*\*direccion\*\*.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*direccion: dirección - describe la dirección a la que se moverá el cabezal.

\*/

cantidadDeCeldasDeDistancia:=0

while(puedeMover(direccion)){

Mover(direccion)

cantidadDeCeldasDeDistancia:= cantidadDeCeldasDeDistancia + 1

}

return(cantidadDeCeldasDeDistancia)

}

function **esCeldaConBolitas()**{

/\*Prop.: indica si la celda actual tiene al menos una bolita, de cualquier color.

Precond.: ninguna.

Tipo: booleano.

\*/

return (nroBolitas>=1)

}

function **hayBolitas\_Al\_(color,direccion)**{

/\*Prop.: indica si hay bolitas de color \*\*color\*\* en la celda en direccion al \*\*direccion\*\*. Si no hay celda, indicará falso.

Precond.: ninguna.

Tipo: booleano.

Parametros:

\*\*color: color - describe el color de la o las bolitas.

\*\*direccion: direccion - describe la dirección a la que revisará el cabezal.

\*/

if(puedeMover(direccion))

then{

Mover(direccion)

}

return(hayBolitas(color))

}

function **máximoEntre\_Y\_(valor1,valor2)**{

/\*Prop.: indica si \*\*valor1\*\* es más grande que \*\*valor2\*\* o viceversa.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*valor1: número - describe el primer valor a comparar.

\*\*valor2: número - describe el segundo valor a comparar.

\*/

return (choose (valor1) when (valor1 > valor2) (valor2) otherwise)

}

function **mínimoEntre\_Y\_(valor1,valor2)**{

/\*Prop.: indica si \*\*valor1\*\* es más chico que \*\*valor2\*\* o viceversa.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*valor1: número - describe el primer valor a comparar.

\*\*valor2: número - describe el segundo valor a comparar.

\*/

return (choose (valor1) when (valor1 < valor2) (valor2) otherwise)

}

function **nroBolitasTotalDeColor\_(color)**{

/\*Prop.: describe la cantidad de bolitas de color \*\*color\*\* que hay en total en todo el tablero.

Precond.: ninguna.

Tipo: número.

Parametros:

\*\*color: color - describe el color de las bolitas.

\*/

cantidadBolitas:=0

IrALaPrimeraCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

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

cantidadBolitas:= cantidadBolitas + nroBolitas(color)

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

}

cantidadBolitas:= cantidadBolitas + nroBolitas(color)

return(cantidadBolitas)

}

function **nroVacías()**{

/\*Prop.: descirbe la cantidad de celdas vicías del tablero.

Precond.: ninguna.

Tipo: número. \*/

cantidadCeldasVacías:=0

IrALaPrimeraCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

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

cantidadCeldasVacías:= cantidadCeldasVacías + unoSi\_CeroSino(esCeldaVacia())

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este,Norte)

}

cantidadCeldasVacías:= cantidadCeldasVacías + unoSi\_CeroSino(esCeldaVacia())

return(cantidadCeldasVacías)

}

function **tieneUnaDeCada()**{

/\*Prop.: indica si hay, al menos, una bolita de cada color en la celda actual.

Precond.: ninguna.

Tipo: booleano.

\*/

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

}

procedure **Mover\_Si\_(direccion,condicion)**{

/\*Prop.: mover el cabezal en direccion al \*\*direccion\*\* si es que \*\*condicion\*\* es verdadero. Si \*\*condicion\*\* es falsa, entonces no hará nada.

Precond.: \*\*condicion\*\* debe ser verdadero. Debe haber una celda en direccion al \*\*direccion\*\*.

Parametros:

\*\*direccion: direccion - indica la direccion a la que se va a mover el cabezal.

\*\*condicion: condición - indica la condición que debe cumplirse para sacar la bolita.

\*/

if(condicion)

then{

Mover(direccion)

}

}

function **nroFilas()** {

/\*prop.describe la cantidad de filas que hay en el tablero

prec.ninguna

tipo numero

\*/

IrAlBorde\_(Oeste)

cantidadFilas:= 1

while( puedeMover(Este)){

Mover(Este)

cantidadFilas:= cantidadFilas + 1

}

return (cantidadFilas)

}

function **nroColumnas()** {

/\*prop.describe la cantidad de columnas que hay en el tablero

prec.ninguna

tipo numero

\*/

IrAlBorde\_(Sur)

cantidadColumnas:= 1

while( puedeMover(Norte)){

Mover(Norte)

cantidadColumnas:= cantidadColumnas + 1

}

return (cantidadColumnas)

}

procedure **Sacar\_Si\_(color,condicion)**{

/\*Prop.: sacar bolita de color \*\*color\*\* si es que \*\*condicion\*\* es verdadero. Si \*\*condicion\*\* es falsa, entonces no hará nada.

Precond.: \*\*condicion\*\* debe ser verdadero. Debe haber una bolita de color \*\*color\*\* en la celda actual.

Parametros:

\*\*color: color - indica el color de la bolita a sacar.

\*\*condicion: condición - indica la condición que debe cumplirse para sacar la bolita.

\*/

if(condicion)

then{

Sacar(color)

}

}

procedure **SacarTodasLasDeColor\_(colorASacar)**{

/\*Prop.: sacar todas las bolitas de color \*\*color\*\* en la celda actual.

Precond.: debe haber, al menos, una bolita de color \*\*color\*\* en la celda actual.

Parametros:

\*\*colorASacar: color - indica el color de las bolitas a Sacar.

\*/

Sacar\_DeColor\_(nroBolitas(colorASacar),colorASacar)

}

procedure **VaciarCelda()**{

/\*Prop.: vaciar la celda actual.

Precond.: ninguna.

\*/

SacarTodasLasDeColor\_(Azul)

SacarTodasLasDeColor\_(Negro)

SacarTodasLasDeColor\_(Rojo)

SacarTodasLasDeColor\_(Verde)

}