2. INSTRUCCIONES DE REPETICION

Objetivo. Este capítulo termina la discusión de las instrucciones construidas en el vocabulario del lenguaje de programación del robot. Las dos nuevas instrucciones que se aprenderán son while{...}. Las instrucciones se pueden ejecutar en varias ocasiones, en cualquier instrucción que entienda el robot. Estás adiciones realzan ampliamente la concisión y energía del lenguaje de programación del robot.

Somos ya programadores experimentados del robot. Una vista abreviada del mundo del robot será utilizada para algunas figuras. Reducir alboroto visual, las etiquetas de la calle y de la avenida y, de vez en cuando, los muros meridionales u occidentales no serán mostrados. Como de costumbre, en nuestros ejemplos se utilizará un solo robot, a menudo nombrado Karel. Puesto que la mayoría de nuestros ejemplos implicarán la manipulación de pitos. Suponemos que seguimos enriqueciendo la clase Robot.

2.1 INSTRUCCIÓN WHILE

Karel está en el origen mirando al este. Entre dos avenidas n y n+1 (n no se conoce) hay un muro de altura 1 en la calle 1. Sobre la calle 1 y hay una fila de pitos que se extiende desde el origen hasta la avenida n (uno en cada esquina). Karel debe recoger estos pitos y terminar frente al muro.

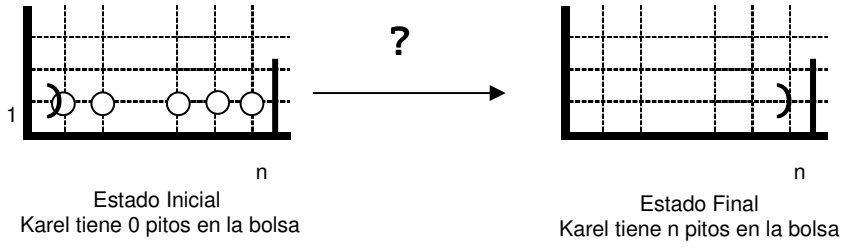


Figura 20. Posibles estados inicial y final del problema propuesto

Note que el número de esquinas con pito no es fijo como en el problema anterior. Hay una cantidad desconocida de esquinas con pito. No se puede usar loop(n) porque no se conoce el valor de n. Se sabe que el robot debe repetir el paso hasta que encuentre un muro al frente (frente bloqueado), es decir, debe hacerlo mientras su frente esté despejado.

El problema es que se debe repetir un paso "varias" veces, pero no se sabe exactamente cuántas. Sin embargo, hay una condición en el mundo que le indica a Karel cuando terminar la repetición. En estos casos se utiliza la instrucción repetitiva while.
Está instrucción tiene la siguiente estructura general:

```
While <condición>
{
  <instrucciones>
}
```

Karel ejecuta las <instrucciónes> mientras la <condición> sea verdadera. Esto es, evalúa la condición en el estado en que se encuentra el mundo y si se cumple ejecuta la instrucción (modificando el mundo) y vuelve a evaluarla en este nuevo estado. Esto se repite hasta que la condición se deje de cumplir (termina cuando se cumple lo contrario). Cada ejecución de las <instrucciónes> se denomina una iteración. La <condición> se llama la condición de entrada. Una instrucción while se llama un ciclo. Si el ciclo tiene una sola instrucción, no podemos suprimir el {...}.

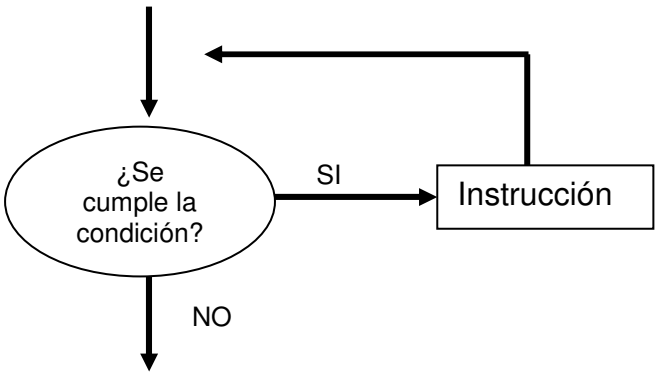


Figura 21. Estructura de la instrucción while

En el problema del ejemplo anterior, se debe repetir el paso hasta que Karel llegue al muro, luego la condición de entrada al ciclo es frontIsClear(), y se obtiene un segmento del programa:

```
while ( frontIsClear() )
{
  pickBeeper();
  move();
}
pickBeeper();
```

2.2 DESARROLLO DE CICLOS while

En el momento cuando decidimos utilizar un while como parte de la solución de un problema se debe tratar de identificar varios factores:

1. El estado del mundo antes de comenzar a ejecutar el ciclo.
2. El estado del mundo en el momento de terminación de ejecutar el ciclo.
3. La transformación gradual que se desea aplicar al mundo: ¿Cómo acercarnos a la solución en cada paso?. ¿Cuál es la regularidad que existe en el problema que permite resolverlo repitiendo varias veces un conjunto de instrucciones?. Para determinar esto es necesario tener en cuenta que:

- Se debe garantizar que en algún momento se llegue a la condición de salida (la negación de la condición de entrada) para que while termine.
- La ejecución del cuerpo del ciclo debe llevar a un estado que se vaya "acercando" al estado final deseado.
- Cada vez que se entre al ciclo se debe encontrar un estado similar, ya que en el problema existe una regularidad.

Como se verá más adelante, una manera de formalizar la esencia de este proceso es mediante un invariante, y así es más fácil desarrollar después el cuerpo del while.

La condición de salida: cuando no interesa volver a ejecutar el ciclo es porque ya llegamos al estado deseado. La negación de está condición es siempre la condición de entrada (la guarda del while). En el ejemplo anterior, en el momento en que el frente esté bloqueado no interesa que Karel vuelva a repetir el cuerpo del while, luego la condición de entrada, será frente desbloqueado (o sea, la negación del frente bloqueado).

El Invariante

Una invariante es una condición que satisface cualquier estado intermedio ("después de i iteraciones"). Dado que existe una regularidad en el proceso (transformación gradual), el invariante dice cómo es el estado del mundo en todo instante durante la ejecución del ciclo: antes de entrar la primera vez, después de cada iteración y al finalizar. Un invariante es útil para el proceso de desarrollo, si muestra el estado de todos los elementos involucrados en el problema, en un punto intermedio de la solución.

Observe el problema anterior:

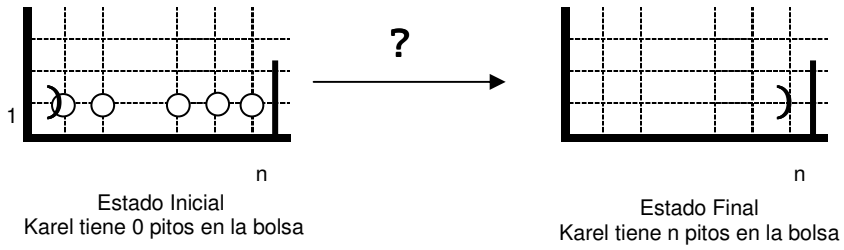


Figura 22. Posibles estados inicial y final de un variante del problema

Para solucionar el problema necesitamos que Karel vaya recogiendo uno a uno los pitos sobre la calle. El invariante que nos "explica" está transformación es el siguiente:
Karel ha recogido los primeros (i -1) pitos y se encuentra sobre la avenida i'ésima, donde $1 \leq i \leq n$. Gráficamente se tiene el mismo invariante así:

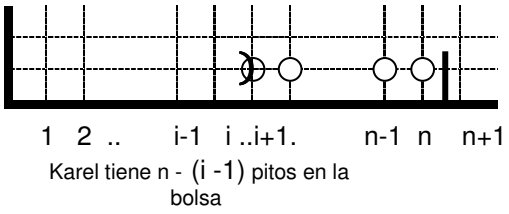


Figura 23. Estado invariante del problema

Lo cual describe claramente el proceso y muestra la forma cómo éste "avanza" hacia la solución del problema.

Se sabe que Karel no puede seguir avanzando cuando encuentra el muro, por lo tanto, la condición de entrada al ciclo debe ser:

```
frontIsClear()
```

Si se supone que el invariante se cumple en el estado i'ésimo para que se conserve en el siguiente estado, el cuerpo del ciclo debe ser tal que se llegue a:

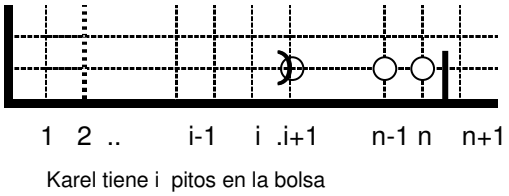


Figura 24. Estado invariante del problema

Es fácil ver que Karel debe recoger el pito y avanzar una avenida:

```
pickBeeper();
move();
```

El ciclo completo es entonces:

```
while ( frontIsClear() )
{
  pickBeeper();
```

```
  move();
}
```

El estado final es:

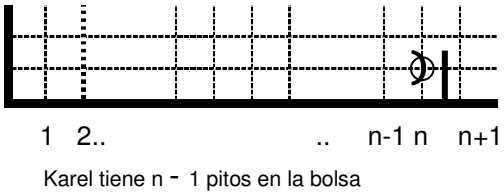


Figura 25. Estado final del problema

Como todavía falta el último pito, el programa completo es el siguiente:

```
while ( frontIsClear() )
{
  pickBeeper();
  move();
}
pickBeeper();
```

Ejemplo Completo: Otra Cosecha

Karel debe recolectar en un campo de 5 surcos verticales de longitud variable. Los surcos empiezan siempre en la calle 1 y van a lo largo de las avenidas, a partir de las avenidas 2 hasta la 6. Todos los pitos en cada surco están juntos, y hay a lo sumo un pito por esquina. A continuación, se describe un estado inicial posible y el correspondiente estado final:

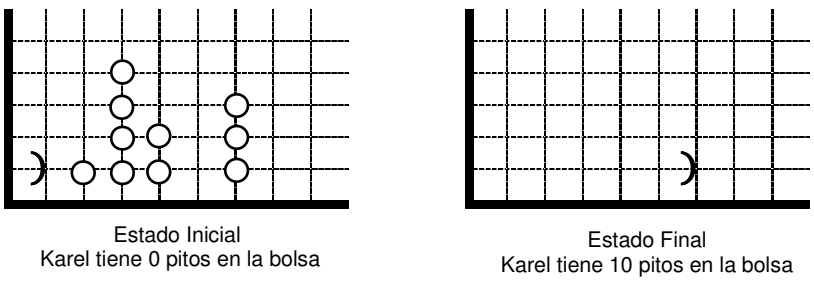


Figura 26. Posibles estados inicial y final del problema recolectar

Plan General:

El enunciado del problema habla de 5 surcos verticales, todos con las mismas características, que deben recibir el mismo tratamiento por parte de Karel, lo que sugiere dividir el problema en 5 subproblemas iguales que se llamarán: recoja-surco. Ahora bien, Karel comienza antes del primer surco y termina a la altura del último, por lo que cada recoja-surco debe comenzar con Karel una avenida antes del surco que va a recoger, mirando al este y terminar sobre la avenida del surco recogido (después de haberlo recogido).

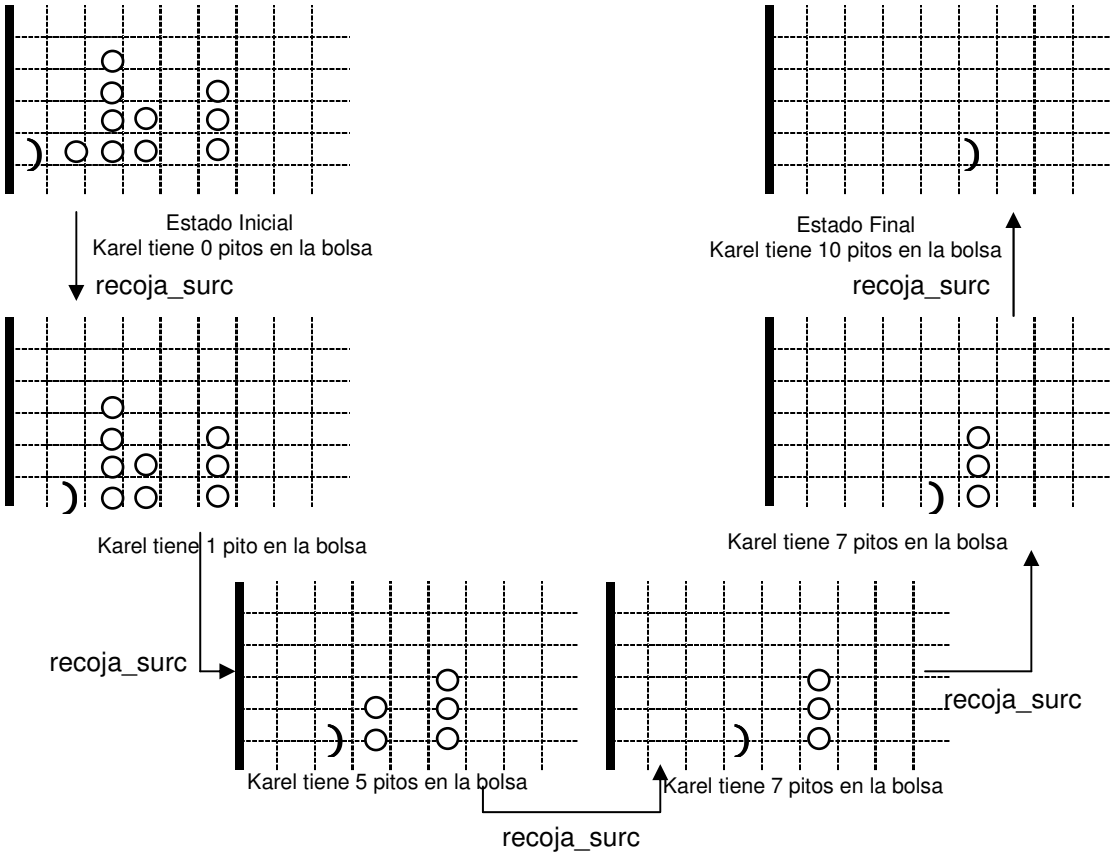


Figura 27. Estado invariante del problema

Bloque principal de ejecución:

```
task
{
 Recolector karel(1,1,East,o);
```

```
loop(5)
    {
       karel.recoja_surco();
       karel.turnOff();
    }
```

Solución del subproblema recoja-surco

La instrucción recoja-surco es todavía bastante compleja, por lo que se debe dividir en subproblemas: en uno que indique a Karel cómo moverse el surco y subir recogiendo los pitos (suba-recogiendo) y en otro para que Karel baje y se localice para el siguiente surco, mirando al este (baje):

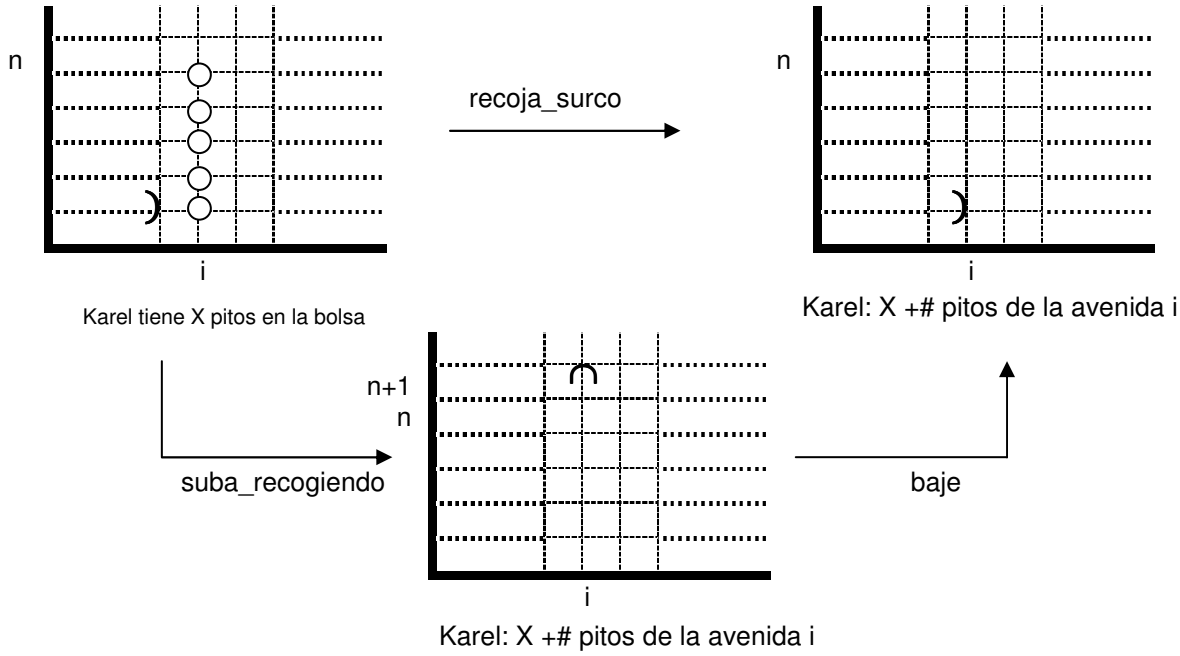


Figura 28. Estado invariante del problema

El bloque de definición de recoja-surco sería entonces:

```
void Recolector:: recoja_surco()
{
   suba_recogiendo();
```

```
   baje();
}
```

Solución del subproblema suba-recogiendo:

Después de un move() y un turnLeft() que localiza a Karel en el surco hacia arriba para comenzar a recoger se tiene:

Figura 29. Estado invariante de suba - recogiendo

No se sabe cuántos pitos hay sobre la calle y Karel debe recogerlos todos. Para poder resolver este problema se requiere un ciclo (while) para que Karel recoja un pito y se mueva, hasta que no haya más pitos. El siguiente estado (j'ésimo) muestra el invariante:

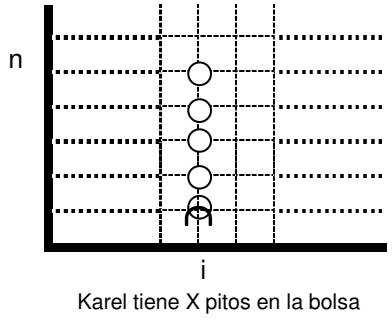


Figura 29. Estado invariante de suba - recogiendo

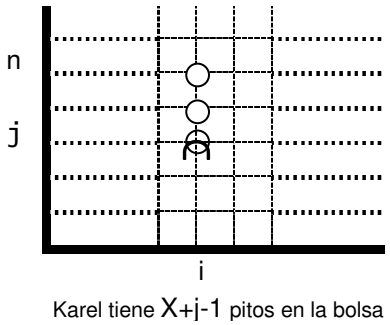


Figura 30. Estado invariante de subproblema

Para que se conserve el invariante en el estado j+1 el cuerpo del ciclo debe ser:

```
pickBeeper();
move();
```

La solución completa es entonces:

```
void Recolector:: suba_recogiendo()
{
   move();
   turnLeft();
    While ( nextToABeeper() )
    {
      pickBeeper();
     move();
   }
}
```

Solución del subproblema baje:

Está instrucción debe orientar a Karel hacia el sur con dos turnLeft(), hacerlo que se mueva hasta un muro (con un while) y colocarlo hacia el este, con otro turnLeft():

```
void Recolector:: baje()
{
   turnLeft();
   turnLeft();

    While ( ( frontIsClear() )
    {
      move();
    }
     turnLeft();
}
```

El programa completo tendría que tener la definición de estás instrucciones en el orden contrario al que se definieron en la solución del problema; es decir, debe definirse primero baje y suba-recogiendo y luego recoja-surco.

```
class Recolector: robot
{
  void recoja_surco();
  void baje();
  void  suba_recogiendo()
}
void Recolector:: baje()
{
   turnLeft();
turnLeft();

    While ( ( frontIsClear() )
    {
      move();
    }
     turnLeft();

}

void Recolector:: suba_recogiendo()
{
   move();
   turnLeft();
    While ( nextToABeeper() )
    {
      pickBeeper();
     move();
   }
}
}

void Recolector:: recoja_surco()
{
   suba_recogiendo();
```

```
   baje();
}

task
{
 Recolector karel(1,1,East,0);
   loop(5)
       {
          karel.recoja_surco();
          karel.turnOff();
       }
}
```

2.3 EJERCICIOS PROPUESTOS

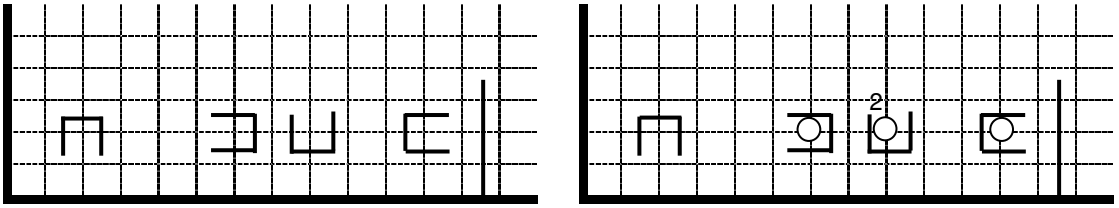
2.3.1. Karel se encuentra en un mundo lleno de cajas cuadradas (cajas formadas con tres muros de un bloque, que encierran una esquina y tiene una entrada por el lado norte). Estás cajas se encuentran todas sobre la calle 2 (encierran una esquina de la calle 2), separadas a una distancia irregular: el número de cajas es variable, pero al final de la última caja existe un muro vertical de altura 3 que corta las calles 1, 2 y 3 y que debe servirle a Karel para reconocer el final de la fila de cajas. Karel parte del origen, mirando al este, con suficientes pitos en su bolsa.

a) Prográmelo para que coloque un pito en la esquina interna de cada una de las cajas.

b) Modifique el programa para que funcione si las cajas están orientadas en cualquier dirección, es decir, si su entrada puede estar hacia el norte, el sur, el este o el oeste.

c) Modifique la solución anterior para que ahora Karel, en un mundo con las mismas especificaciones, coloque 0,1, o 2 pitos en cada caja así: si la caja tiene su entrada hacia el sur, Karel no debe colocar pitos (pues se saldrían), si la entrada está hacia el este o hacia el oeste debe colocar 1 pito y si está

hacia el norte debe colocar en ella dos pitos. A continuación, se muestra un posible estado inicial y el correspondiente estado final para este último caso:



Estado Inicial
Karel tiene 30 pitos en la bolsa

Estado Final
Karel tiene 26 pitos en la bolsa

Figura 31. Estado inicial y final problema de las cajas cuadradas

2.3.2 Revise las siguientes parejas de códigos y diga sí son o no equivalentes

Segmento A

```
pickBeeper();
while ( frontIsClear() )
{
  pickBeeper();
  move();
}
```

Segmento B

```
while (nextToABeeper() )
{
  pickBeeper();
  move();
}
pickBeeper();
```

2.3.3 Revise las siguientes parejas de códigos y diga sí son o no equivalentes

Segmento A

```
while (anyBeepersInBeeperBag() )
{
   putBeeper();
   if (frontIsClear())
   {
     move();
    }
}
```

Segmento B

```
while (frontIsClear()() )
{
   if (anyBeepersInBeeperBag())
   {
     putBeeper();
     move();
    }
}
```

2.3.4 Diga si los siguientes segmentos de programa son o no equivalentes:

Segmento A

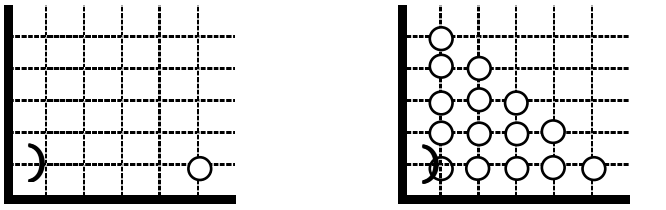
```
loop(3)
{
    move();
    while (facingNorth())
   {
     turnLeft();
     pickBeeper();
   }
      turnLeft();
```

```
}
```

Segmento B

```
while (facingNorth())
{
     move();
     loop(3)
     {
     turnLeft();
     pickBeeper();
    }
      turnLeft();
}
```

2.3.5 Karel se encuentra en un mundo en el que hay un pito en la esquina de la calle 1 con una avenida x. Programe a Karel para que construya un triángulo isósceles, de lado x, relleno de pitos, de forma que su vértice superior quede en el origen. Karel parte del origen, mirando al este y debe terminar en la misma posición. Suponga que el robot tiene suficientes pitos en su bolsa. A continuación, se presenta un posible estado inicial y el correspondiente estado final:



Estado Inicial
Karel tiene 16 pitos en la bolsa

Estado Final
Karel tiene 2 pitos en la bolsa

Figura 32. Condiciones iniciales y finales del problema construir isósceles

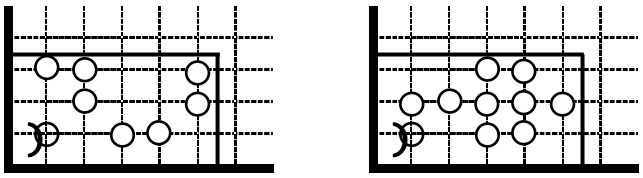
2.3.6 Modifique el programa anterior para que Karel construya únicamente el perímetro de un triángulo isósceles, sin rellenarlo.

2.3.7 Karel ha sido contratado por artesanías de Colombia para elaborar sus nuevos diseños de tapetes. Para construir el arte del tapete, Karel debe trabajar dentro de un cuarto rectangular, colocando un pito en algunas esquinas del cuarto. Karel parte de la esquina inferior izquierda del cuarto, mirando al este, con suficientes pitos en su bolsa. Para seguir el tejido, Karel debe obedecer las siguientes reglas:

a) El diseño (pitos) que está sobre la calle más al sur del cuarto, debe quedar intacto.
b) Para decidir el patrón del tejido para una calle del cuarto es necesario haber resuelto el patrón para las calles que están al sur de éstas.
c) Para decidir el patrón final en una calle dada, se hace lo siguiente:

- Si sobre la esquina en consideración no hay pito, pero en la inmediatamente al sur de ésta no hay pito, se deja tal como está.
- Si en la esquina en consideración no hay pito, pero en la inmediatamente al sur sí hay, Karel debe colocar uno en la esquina en consideración.
- Si en la esquina de Karel hay pito y en la de más al sur no hay, Karel deja la esquina intacta.
- Si tanto en la que está, como en la de más al sur hay pito, Karel recoje el de su esquina.

Programe a Karel para que resuelva está tarea. A continuación, se da un ejemplo:



Estado Inicial
Karel tiene 16 pitos en la bolsa

Estado Final
Karel tiene 0 pitos en la bolsa

Figura 33. Posibles estados iniciales del problema diseño de tapetes

2.3.8 Karel está sembrando flores (pitos) en las colinas de su mundo (una serie de montes de altura 4 pegados uno a otro M). Programe a Karel para que, partiendo del origen, mirando al este, coloque todas las flores que tiene en su

bolsa en las colinas. Karel debe terminar sobre la calle 1, al final de la última colina en la que colocó alguna flor. Note que a Karel se le pueden acabar las flores en la mitad de una colina, en tal caso debe terminar de recorrer la colina (sin colocar flores). A continuación, se ilustra un estado posible y el correspondiente estado final:

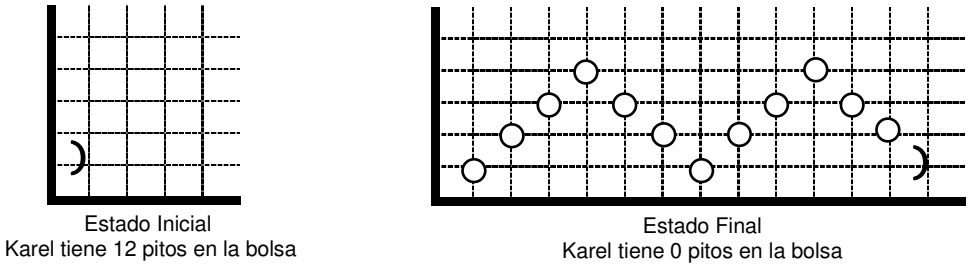


Figura 34. Posibles estados inicial y final del problema sembrando flores

2.3.9 Karel parte del origen (mirando al este) y debe recoger una serie de diagonales de pitos que parten de la calle 1 y se extienden hacia el nordeste, las diagonales pueden ser de cualquier tamaño; la primera diagonal parte del origen; a partir de ésta, el primer pito de cada diagonal se encuentra en la calle 1, sobre la avenida siguiente a la última avenida ocupada por la diagonal anterior. Por ejemplo, en el caso particular que se presenta a continuación, la primera diagonal termina en la avenida 4 y la segunda comienza en la 5:

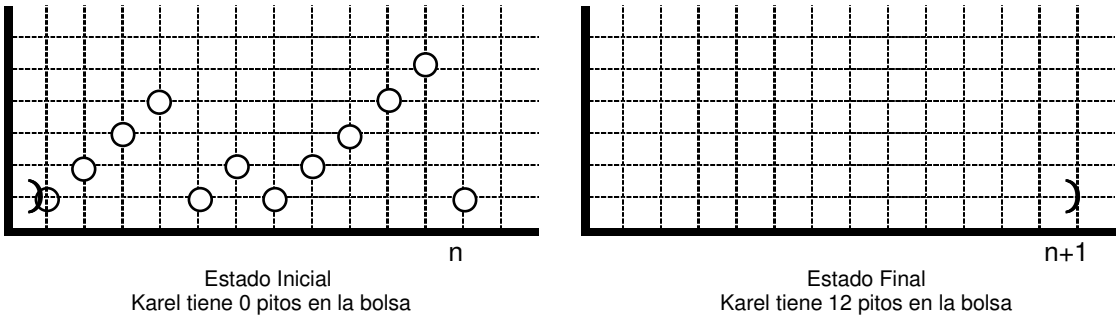


Figura 35. Estados inicial y final del problema haciendo diagonales

2.3.10 Karel se encuentra sobre la calle 1, mirando al este, frente a un obstáculo vertical de longitud x, Karel tiene en su bolsa n pitos ($n \geq x$).

Prográmelo para que coloque x de sus n pitos delante del obstáculo (al oeste del muro) y el resto (n-x) del lado este del obstáculo (ambos grupos sobre la calle 1). Karel debe terminar en su posición inicial. <<

A continuación, se da un ejemplo del problema:

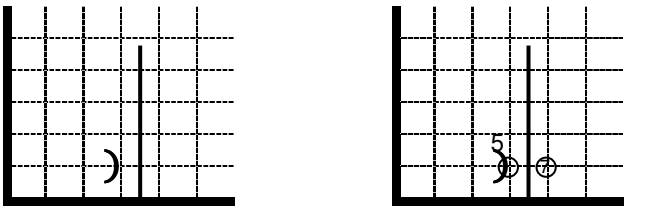


Figura 36. Estados inicial y final del problema en colocar pitos

2.3.11 En el mundo de Karel se tienen dos pitos: uno sobre la calle 1 en alguna avenida "x" y otro en la avenida 1 sobre la calle "z". Programe a Karel para que coloque un pito en la esquina de la calle "z", avenida "x". Karel comienza sin pitos en la bolsa; note que no se especifica la posición en la que debe terminar Karel (puede escoger la que más le sirva para su algoritmo). A continuación, se presenta un posible estado inicial y el correspondiente estado final.

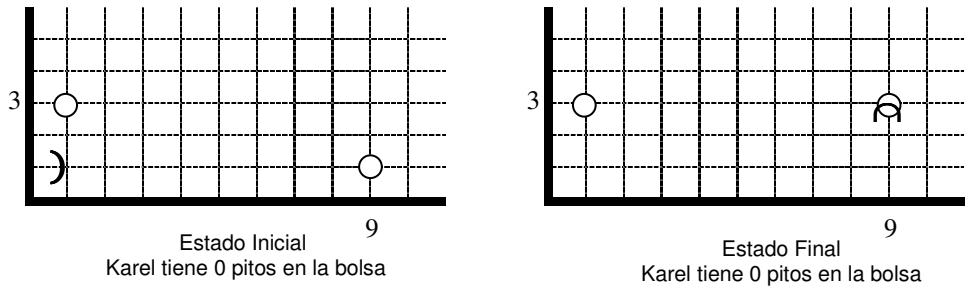
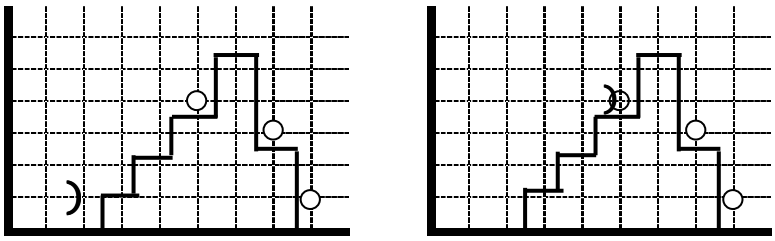


Figura 37. Estados inicial y final del problema "colocando pito calle z, avenida x"

2.3.12 Escriba un programa en lengua Karel para que el robot suba una montaña irregular (de cualquier forma, pegada al muro infinito del sur del mundo), hasta encontrar un pito (que se encuentra contra la montaña). Karel

parte del origen, mirando al este, frente a la montaña. A continuación, se muestra un caso posible.

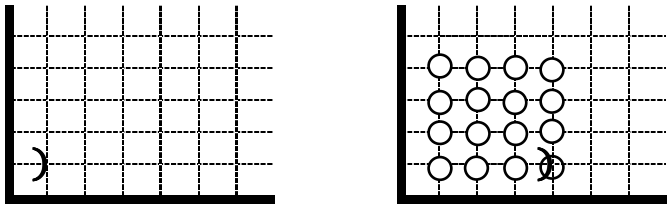


Estado Inicial
Karel tiene 0 pitos en la bolsa

Estado Final
Karel tiene 0 pitos en la bolsa

Figura 38. Estados inicial y final del problema suba montaña irregular

2.3.13 Karel se encuentra en el origen, mirando al este con N^2 pitos en su bolsa. Prográmelo para que coloque todos sus pitos en un cuadrado de lado N a partir del origen. A continuación, se da un posible estado inicial y su correspondiente estado final.

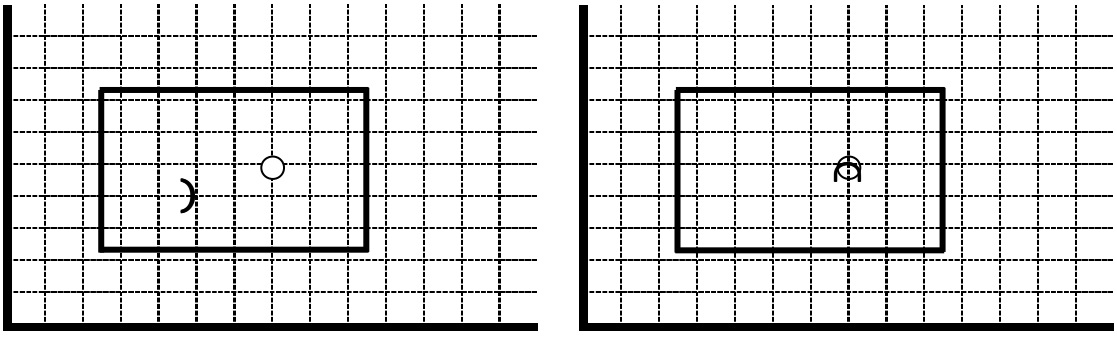


Estado Inicial
Karel tiene 16 pitos en la bolsa

Estado Final
Karel tiene 0 pitos en la bolsa

Figura 39. Estados inicial y final del problema colocar pitos en un cuadrado

2.3.14 Karel se encuentra en un cuarto rectangular cerrado (sin muros interiores), en el que se le ha perdido un pito. Programe a Karel para que localice el pito. (Nota: en el estado inicial tanto Karel como el pito pueden estar en cualquier posición dentro del rectángulo). A continuación, se presenta un posible estado inicial y el correspondiente estado final.



Estado Inicial
Karel tiene 0 pitos en la bolsa

Estado Final
Karel tiene 0 pitos en la bolsa

Figura 40. Estados inicial y final del problema localice el pito

2.3.15 Karel se encuentra en una oficina rectangular que tiene varias puertas y ventanas y ha perdido un documento importante. Ayude a Karel a encontrar el documento (pito), sin salirse de su oficina. Ayuda: dentro de la oficina no hay muros. A continuación, se presenta un posible estado inicial y el correspondiente estado final.

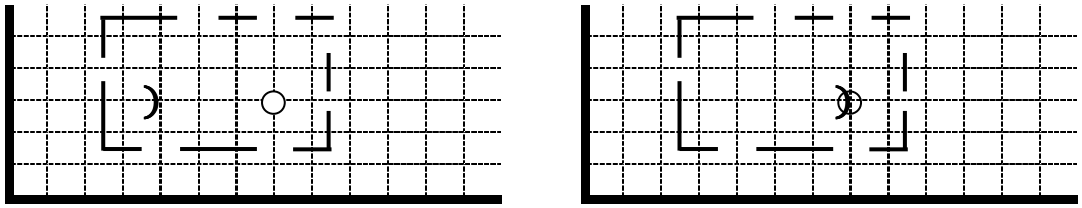


Figura 41. Estados inicial y final del problema encontrar documento

2.3.16 En el mundo de Karel hay una columna vertical (de longitud impar) de pitos sobre alguna avenida del mundo, a partir de la calle 1. Programe a Karel para que construya otra columna de pitos, de igual tamaño que se cruce con la original en el centro, formando una cruz perfecta. Suponga que la columna original está suficientemente separada del muro infinito vertical del mundo para que quepa la cruz y que Karel parte del origen con suficientes pitos en su bolsa. Por ejemplo:

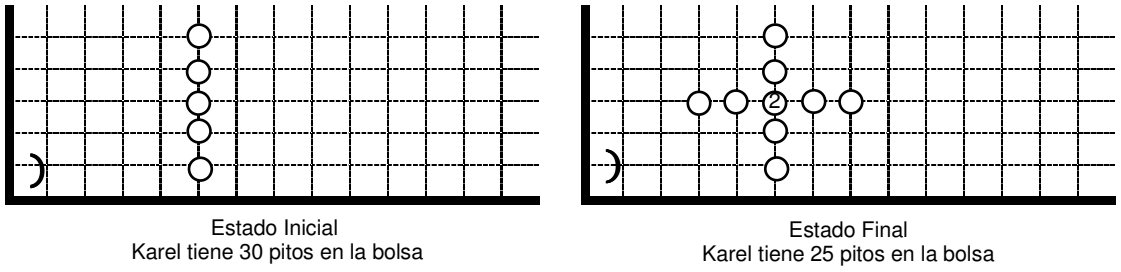
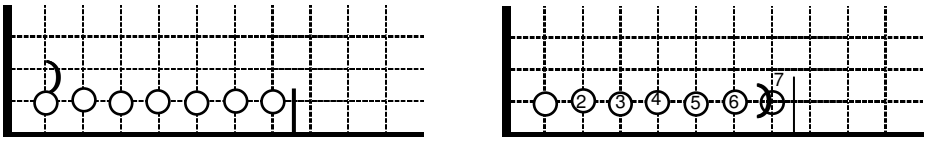


Figura 42. Estados inicial y final del problema “Construya columna de pitos”

2.3.17. En el mundo de Karel hay un muro que corta la calle 1 a una distancia desconocida. Programe a Karel para que genere los números naturales, desde 1 hasta el muro, es decir que coloque en el origen un pito, en la segunda avenida 2, en la siguiente 3, etc. Hasta llegar al muro. Suponga que Karel parte

del origen, mirando al este, con suficientes pitos en su bolsa. A continuación, se muestra un ejemplo posible:

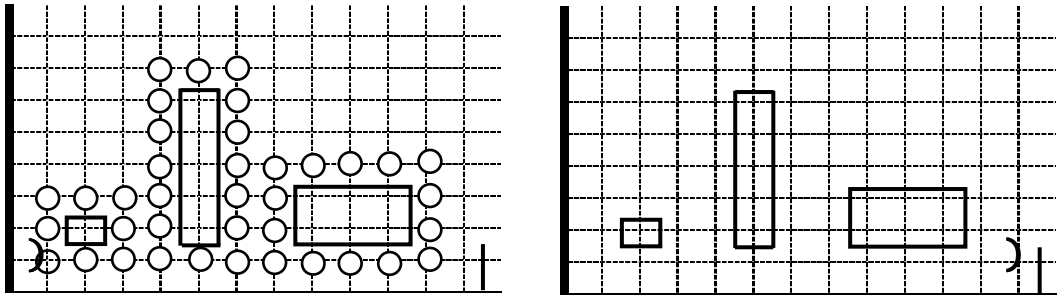


Estado Inicial
Karel tiene 30 pitos en la bolsa

Estado Final
Karel tiene 2 pitos en la bolsa

Figura 43. Estados inicial y final del problema "Genere los números naturales"

2.3.18. Karel está muy preocupado por el problema de las basuras del Mundo-Karel. En su candidatura para la alcaldía ofreció resolver este problema y ahora nos solicita que lo ayudemos. En el Mundo-Karel hay varias construcciones (rectángulos de segmentos de muro: ™ de cualquier tamaño) localizados encima de la calle 1, a partir de la avenida 1. Estás construcciones están separadas entre sí por 2 avenidas. Dos avenidas después de la última construcción, hay un muro (de altura 1 bloque), que corta la calle 1. En cada una de las esquinas que rodean una construcción hay una caneca de basura (pito). Haga un programa para que Karel, partiendo del origen, mirando al este, recoja todas las canecas de basura (pitos) que rodean las construcciones y termine frente al muro que corta la calle 1, mirando al este. A continuación, se muestra un posible estado inicial y el correspondiente estado final:



Estado Inicial
Karel tiene 0 pitos en la bolsa

Estado Final
Karel tiene 38 pitos en la bolsa

Figura 44. Estados inicial y final del problema "Basuras"

2.3.19 Karel parte del origen, mirando al este. A una distancia desconocida del origen, hay un muro vertical de una altura mayor que 6. Sobre la calle 1, entre el origen y este muro vertical hay "montones" de pitos (cualquier cantidad entre 0 y 6 en cada esquina). Karel debe ordenar estos montones de mayor a menor (de acuerdo con el número de pitos que haya en cada uno) a partir del origen, y terminar contra el muro vertical, mirando al este. A continuación, se presenta un estado inicial posible y el estado final correspondiente y se da un plan de solución para el problema general.

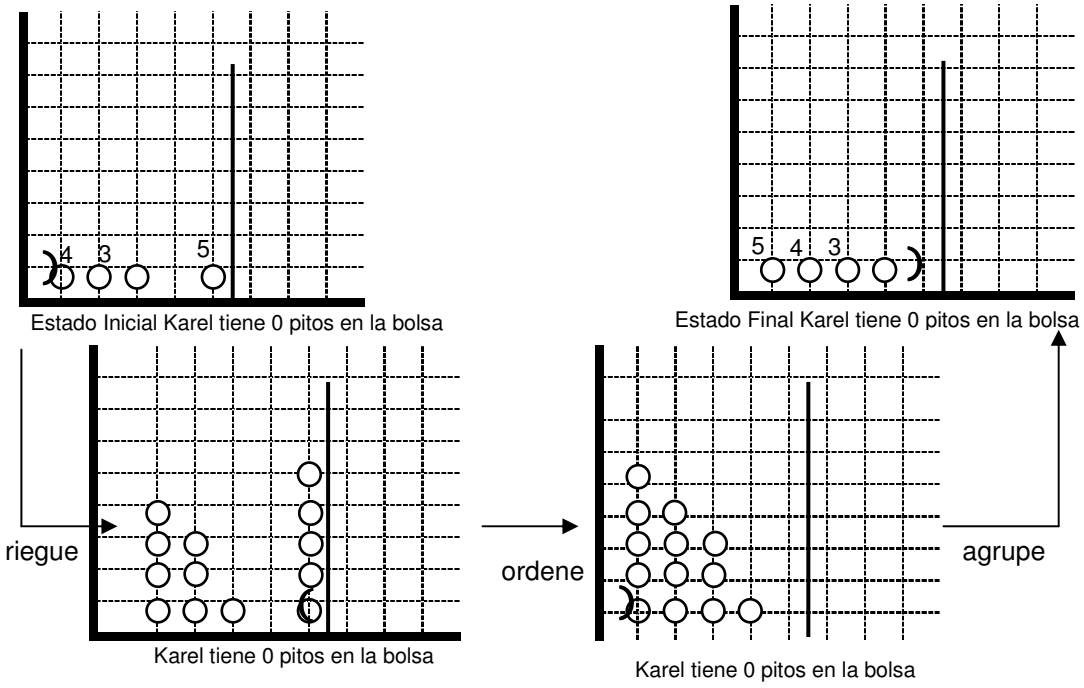


Figura 45. Invariante del problema propuesto

De acuerdo con el plan presentado anteriormente, el bloque de ejecución del programa resulta de la siguiente manera:

```
task
{   ur_Robot  karel(1,1,East,0);
      karel.riegue();
      karel.ordene();
      karel.agrupe();
      karel.turnOff();
}
```

La instrucción ordene consiste en repetir 6 veces (1 para cada calle i a partir de la 1): recoger todos los pitos de la calle i, colocarlos a partir de la avenida 1 y subir a la calle siguiente (i+1)

A continuación, se describen gráficamente estos tres pasos para la calle i:

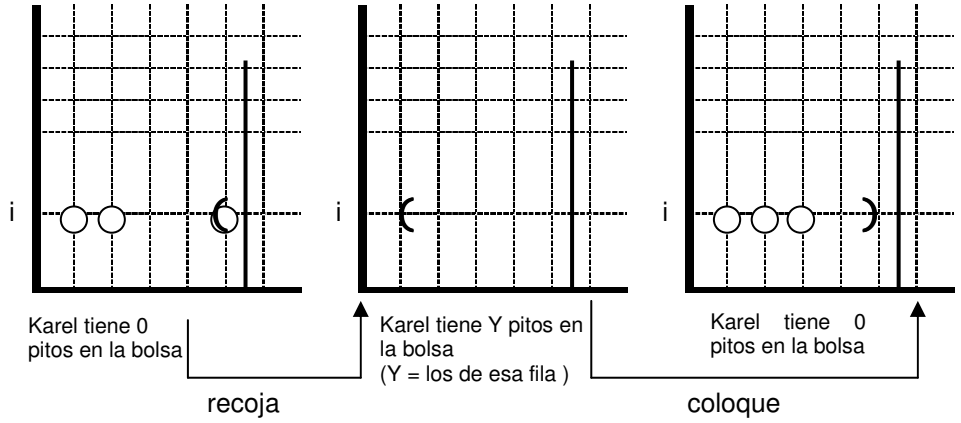


Figura 46. Invariante del problema enunciado

Después de repetir 6 veces estos pasos, la instrucción ordene debe hacer que karel vuelva al origen (vuelvaorigen) y quede mirando al este (para seguir con agrupe). La definición de ordene es:

```
void clase::ordene()
{
  loop(6)
   {
    recoja();
    coloque();
    suba();
   }
 vayaorigen();
}
```

Termine de escribir el programa de solución, definiendo las instrucciones que faltan (para definir estás instrucciones se puede requerir más instrucciones nuevas).

a) Defina la instrucción riegue

b) Defina la instrucción agrupe

c) Defina las instrucciones recoja, coloque, suba y vuelvaorigen

d) En el problema se supuso que el máximo número de pitos en cada esquina era 6. Modifique el programa para que resuelva correctamente el problema para cualquier cantidad de pitos en cada esquina, suponiendo que el muro vertical finito que limita los pitos es de una altura mayor que el máximo número de pitos que hay en una esquina.

2.3.20. Karel se encuentra en un campo de práctica de 1 milla (8 bloques) de largo, con obstáculos de 1, 2 o 3 bloques de alto colocados aleatoriamente entre dos esquinas de la pista. Para ayudar a un amigo principiante, Karel desea indicar el tamaño de cada obstáculo. Programe a Karel para que coloque delante de cada obstáculo (al oeste del muro), tantos pitos como bloques de alto tenga el obstáculo. Karel comienza en el origen mirando al este con el número exacto de pitos que necesita colocar en su recorrido y debe terminar nuevamente en el origen, mirando al oeste con 0 pitos en la bolsa. A continuación, se presenta un posible estado inicial y el correspondiente estado final:

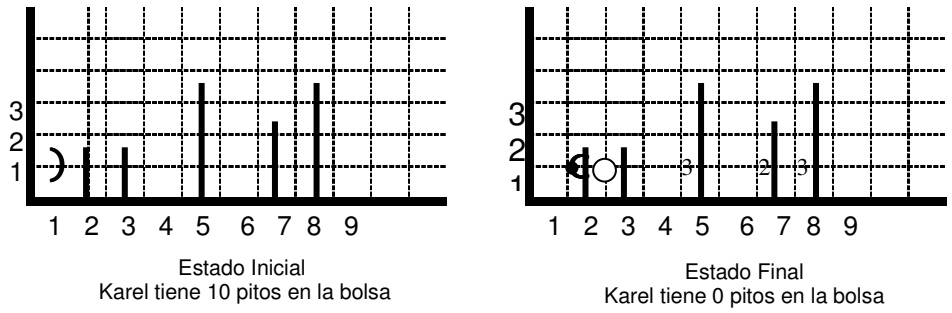


Figura 47. Estados inicial y final del problema de obstáculos

2.3.21. Karel es el portero de un edificio de oficinas. Entre sus funciones está la de lavar los parqueaderos que se encuentran en el sur de la calle principal del conjunto (la calle Alamos). Los parqueaderos se extienden verticalmente; cada

parqueadero tiene un bloque de ancho y cualquier longitud. Los parqueaderos están localizados a partir de la avenida 2. El final de los parqueaderos está marcado por un muro que bloquea la calle Alamos. Karel parte de la calle Alamos con avenida 1, mirando al este, con suficientes baldes de agua (pitos) en su bolsa. Prográmelo para que lave todos los parqueaderos (1 balde-pito por esquina de parqueadero) y termine frente al muro que rodea la calle Alamos, mirando al este. Una posible estado inicial y el correspondiente estado final se muestra a continuación.

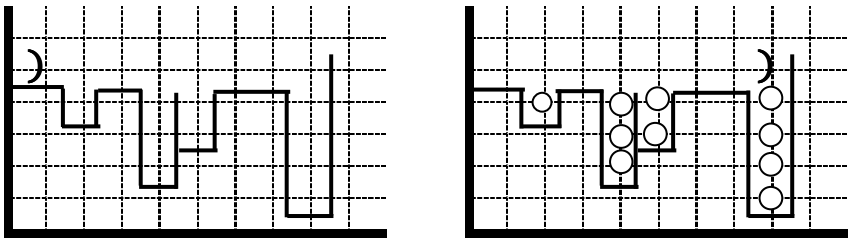


Figura 48. Estados inicial y final del problema "Lavar el parqueadero"

Note que no en todas las avenidas hay parqueaderos y que puede haber dos parqueaderos contiguos.

2.3.22. Programe a Karel para que recoja todas las flores de la montaña (escalera de muros). Karel comienza y termina en el origen, mirando al este con la bolsa vacía y debe terminar igual. A continuación, se presenta un posible estado inicial y su correspondiente estado final.

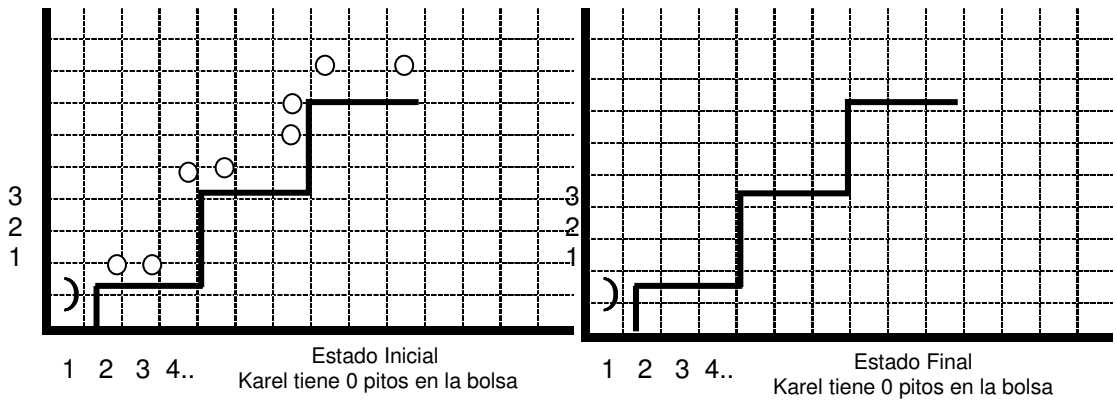


Figura 49. Estados inicial y final del problema "Recoger todas las flores"

2.3.23. En el mundo de Karel hay dos filas verticales de pitos, que parten de la calle 1 se extienden hacia el norte (1 pito por esquina). Karel parte del origen, mirando al este, con 1 pito en su bolsa y debe terminar sobre la base (calle) de la fila más larga de pitos, mirando al norte, con 1 pito en su bolsa. A continuación, se muestra un posible estado inicial y el correspondiente estado final.

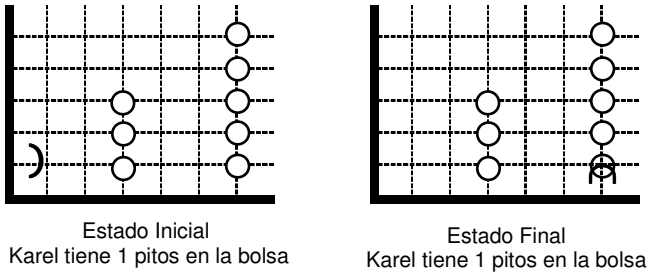


Figura 50. Estados inicial y final del problema "Identificar la fila más larga"

En los siguientes problemas se da la descripción de un estado inicial para Karel y un programa. Usted debe describir el estado final al que llega Karel después de ejecutar ese programa, si parte del estado inicial dado (recuerde que describe el estado final e incluye la descripción de todos los elementos del mundo). Si Karel termina por un error de ejecución, explique claramente la causa de ese error.

2.3.24. Karel se encuentra en la calle 1 con avenida 7 mirando al este con 0 pitos en la bolsa, frente a una montaña. A continuación, se presenta el estado inicial:

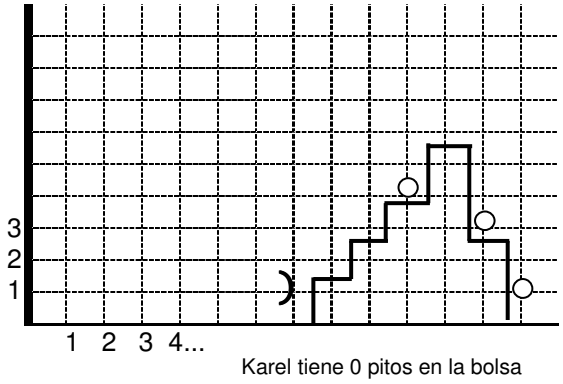


Figura 51. Estado inicial del problema propuesto

Karel debe ejecutar el siguiente programa:

```
class alpinista : Robot
{
  void turnRight();
}
void alpinista::turnRight()
{
  loop(3)
  {  turnLeft();
  }
}

task
{  alpinista karel(1,7,East,0);
  while (!nextToABeeper())
  {
    if (rightIsBlocked())
     {
        if (frontIsBlocked())
        {
             turnLeft();
        }
         else
              {
                   move();
              }
     }
       else
          {
            turnRight();
            move();
          }
  }
    turnOff();
}
```

Describa claramente el estado final al que llega Karel:

2.3.25. Karel se encuentra en el origen, mirando al este, con 15 pitos en su bolsa y ejecuta el siguiente programa:

```
class Desconocido: Robot
{
       void turnRight();
       void ponga();
       void escala();
}
void Desconocido::turnRight()
{
  loop(3)
  {  turnLeft();
  }
}
void Desconocido::ponga()
{
     if(anyBeepersInBeeperBag())
     {
          putBeeper();
      }
}
void Desconocido::escala()
{
     loop(3)
     {
          move();
          turnLeft();
          move();
          turnRight();
          ponga();
      }
}

task
```

```
{
    Desconocido karel(1,7,East,15);
   while(anyBeepersInBeeperBag())
   {
      karel.escala();
      karel.turnRight();
      karel.escala();
      karel.turnLeft();
   }
   karel.turnOff();
}
```

Describa claramente el estado final al que llega Karel:

2.3.26. En el mundo de Karel se tiene un sembrado rectangular de pitos formado por un número par de calles y cualquier cantidad de avenidas. El sembrado está cercado por muros y tiene una entrada de un bloque en la esquina más al este de la pared sur. En cada esquina del sembrado puede haber una mata pequeña (1 pitos), grande (2 pitos) o puede estar sin cultivar (0 pitos). Programe a Karel para que, partiendo de la esquina que queda frente a la entrada de sembrado, mirando al norte y con suficientes pitos en la bolsa, cultive el sembrado y vuelva a su posición inicial. Cultivar el sembrado significa regar las matas pequeñas (donde hay 1 pito deben quedar 2), talar las grandes (donde hay 2 pitos quitarlos) y sembrar (donde no hay pitos debe colocar uno). Fíjese que Karel no puede pasar dos veces por la misma esquina cultivando. Un estado inicial posible y el correspondiente estado final se muestran a continuación:

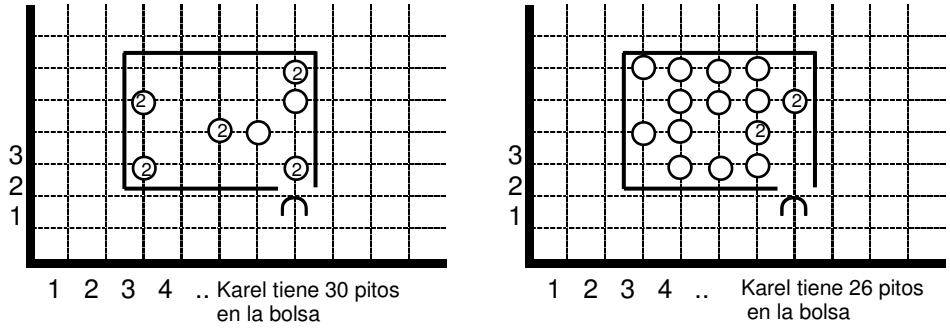


Figura 52. Condiciones iniciales y finales del problema “Sembrado”

2.3.27. Karel está de visita en una gran ciudad y se le ha encargado que ponga bombillos (pitos) en el techo de los 4 edificios de la ciudad. Un edificio está construido con tres muros: dos verticales y uno horizontal; puede ser de 1, 2 o 3 pisos y cubrir 1, 2 o 3 manzanas; los edificios están separados entre sí por una avenida (una sola esquina). Karel se encuentra en el origen mirando al este con un número suficiente de pitos. Prográmelo para que ponga un pito en cada esquina de la parte superior de cada edificio y termine después del último edificio, mirando al este. A continuación, se presenta un posible estado inicial y el correspondiente estado final.

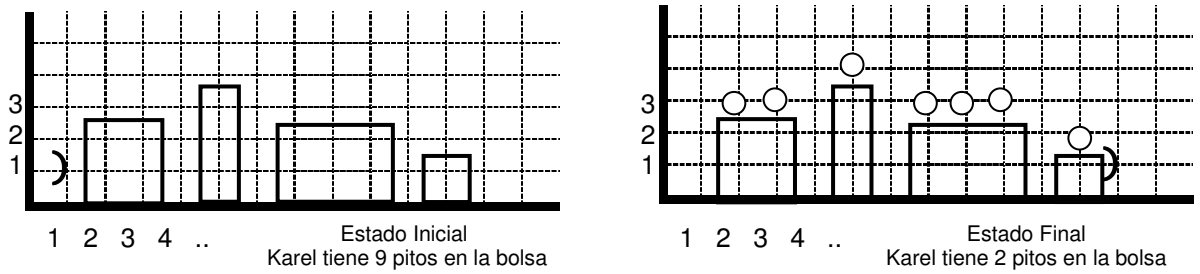


Figura 53. Estados inicial y final del problema “Ponga bombillos”

2.3.28. Karel encontró la pista que lo conduce al famoso tesoro de Morgan (varios pitos, más de uno); el pirata, al enterrar el tesoro, dejó un rastro de monedas (pitos) que parte del origen del mundo. Programe a Karel para que, partiendo del origen, mirando al este, siga el rastro de monedas y recoja el tesoro. El rastro de monedas es un camino; entre un pito del camino y el siguiente hay una calle o una avenida de diferencia; recorriendo el camino hacia delante, Karel sólo encuentra en cada paso un movimiento posible. A continuación, se presenta un posible estado inicial y el correspondiente estado final.

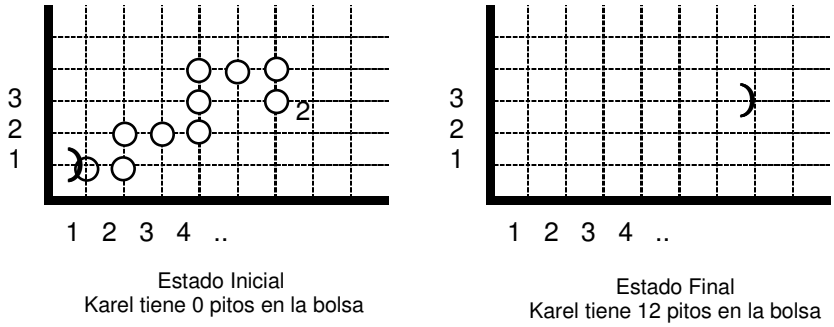


Figura 54. Condiciones iniciales y finales del problema "Tesoro de morgan"

2.3.29. Karel se encuentra en un cuarto cerrado (sin muros interiores), en el que se le ha perdido un pito. Programe a Karel para que localice el pito y lo recoja. El pito perdido se encuentra contra una pared del cuarto. A continuación se presenta un posible estado inicial y el correspondiente estado final.

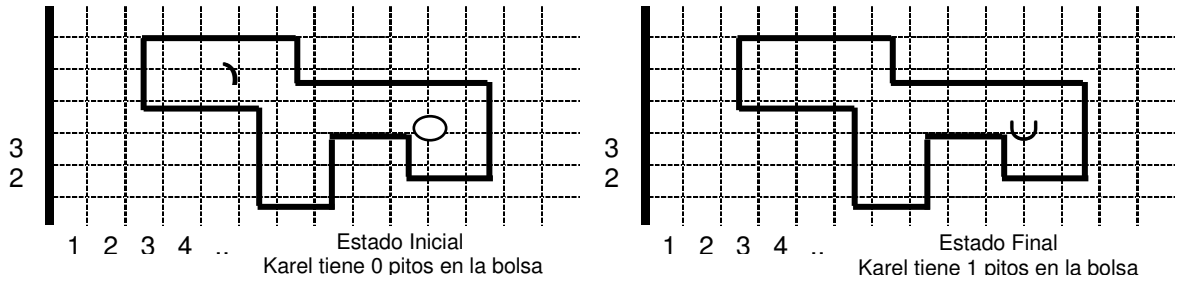


Figura 55. Condiciones iniciales y finales del problema "Cuarto cerrado"

2.3.30. Programe a Karel para que busque un pito (en un mundo sin muros) y coloque alrededor de éste, más pitos. Tenga en cuenta que el pito puede estar pegado a una de las dos paredes infinitas del mundo; Karel comienza con 8 pitos en su bolsa. Fíjese que al final Karel puede quedar en cualquier posición.

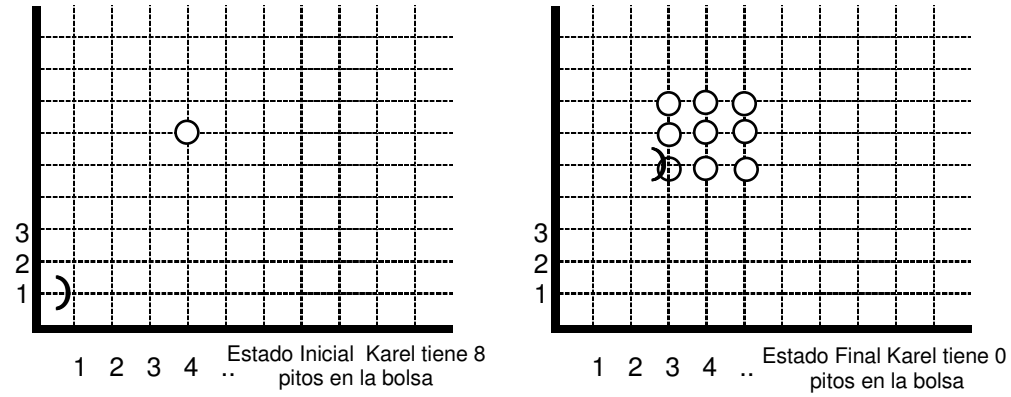


Figura 56. Condiciones iniciales y finales del problema "Buscar y rodearse"

2.3.31. Karel se encuentra en un túnel en forma de cruz. Los brazos del túnel son de un bloque de ancho y de longitud mayor que uno. Karel se encuentra en el centro del túnel mirando en cualquier dirección y desea colocar una marca, en este sitio (el centro) que indique por dónde es la salida y luego ir hasta la salida. Programe a Karel para que coloque en el centro del túnel un pito a la salida si está en el brazo oeste, dos si está en el brazo norte, tres si está en el

brazo este y cuatro si está por el lado norte y para que termine en la salida. A continuación, se presenta un posible estado inicial y el correspondiente estado final.

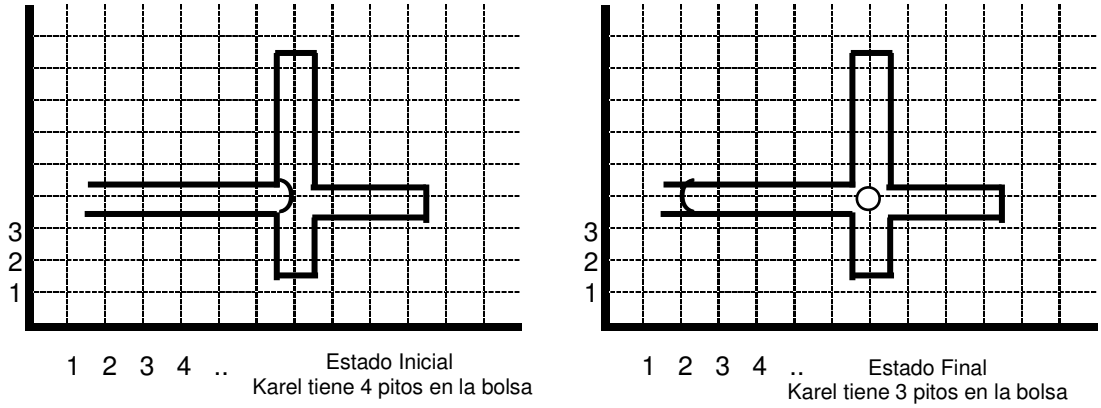


Figura 57. Condiciones iniciales y finales del problema "Túnel"

2.3.32. En el mundo de Karel hay una figura formada por calles de pitos. Las calles de pitos que conforman la figura comienzan a partir de la calle 1 hasta la calle F (arbitraria) y en cada calle los pitos están colocados uno detrás de otro empezando en la avenida 1 (mínimo un pito, puede estar totalmente llena). Entre las avenidas E (arbitraria) y E+1 hay un espejo (muro) que parte de la calle 1 y tiene longitud L (arbitraria) (donde L>=F). Haga un programa para que Karel refleje la imagen de la figura sobre el espejo. A continuación, se muestra un posible estado inicial con su correspondiente estado final. Karel parte del origen mirando al este y también termina en el origen mirando al este.

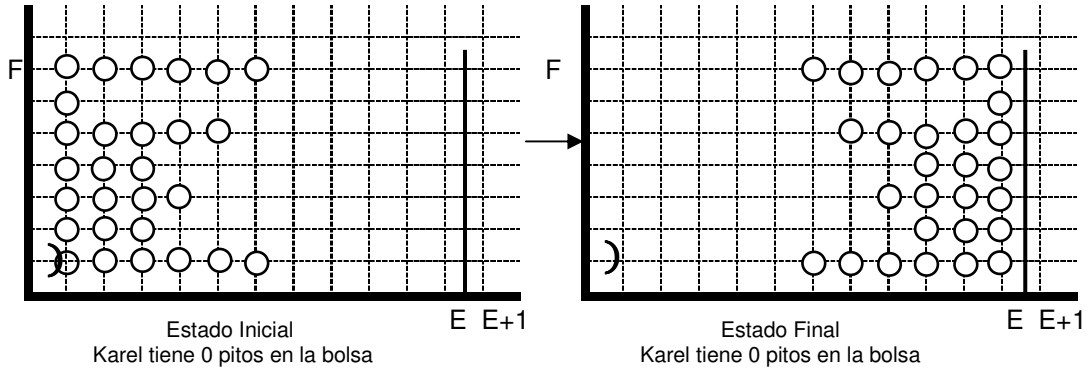


Figura 58. Condiciones iniciales y finales del problema "Imagen"

Plan general:

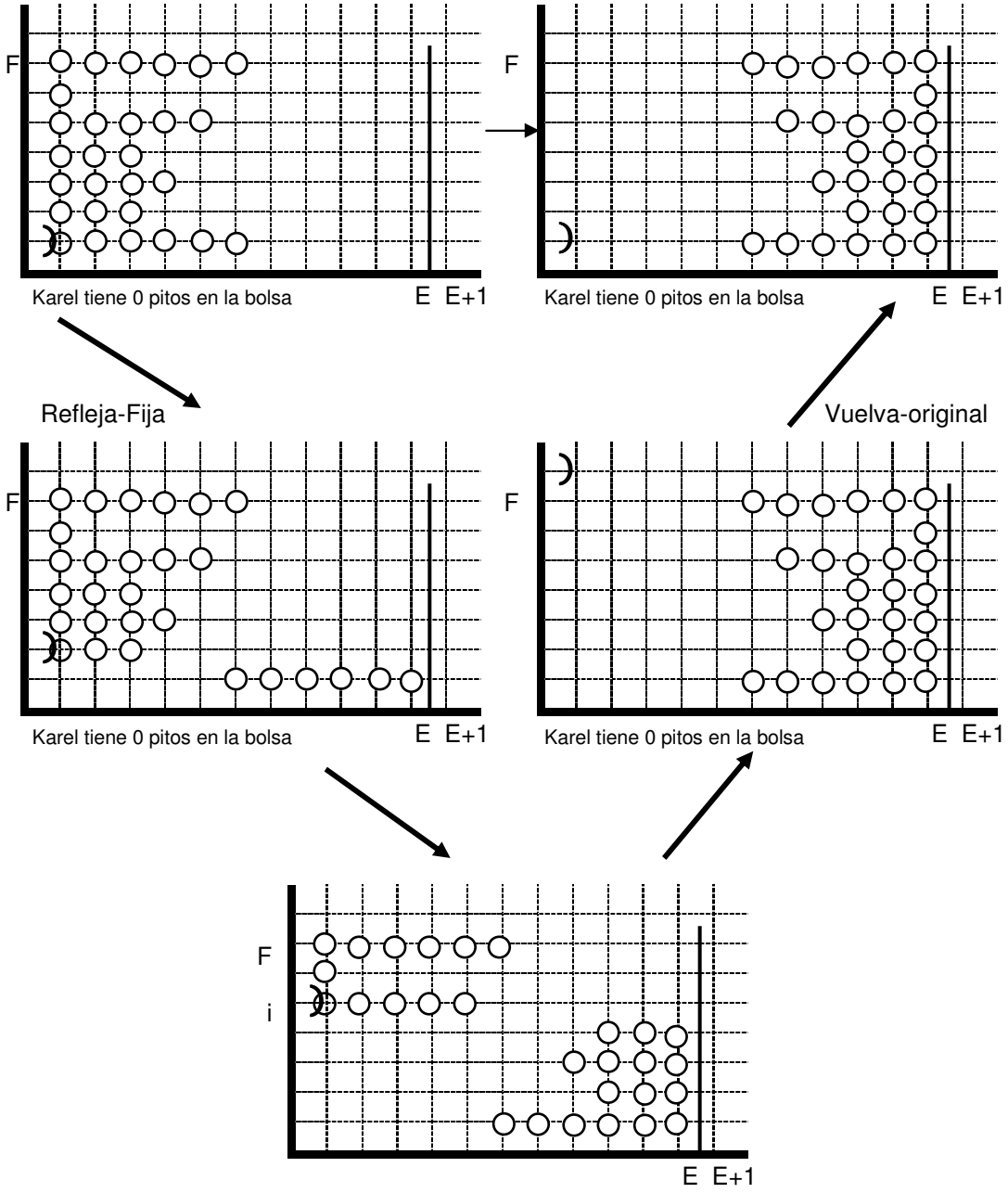


Figura 59. Invariante del problema "Imagen"

Resuelva el problema siguiendo el plan de solución presentado. Este plan muestra que en cada paso Karel debe reflejar una fila de la figura en el espejo (empezando con la fila que está sobre la calle 1). Después de reflejar la última fila que tenía la figura, Karel debe volver al origen.

2.3.33. Karel es un coleccionista de carros antiguos. Cada vez que llueve tiene el problema de guardar todos sus carros en los parqueaderos cubiertos de su

casa. Ayúdele a encontrar la mejor manera de guardar los carros dentro de los parqueaderos. Tenga en cuenta que no necesariamente hay parqueaderos para todos los carros (puede haber más carros que parqueaderos). Un carro lo vamos a representar como un pito y un parqueadero cubierto como una avenida del mundo que tiene muros en sus dos lados verticales. Note que la longitud de los dos muros determina la capacidad de los parqueaderos. A continuación, se da un posible estado inicial con sus correspondientes estados final. Karel siempre comienza en el origen mirando al este

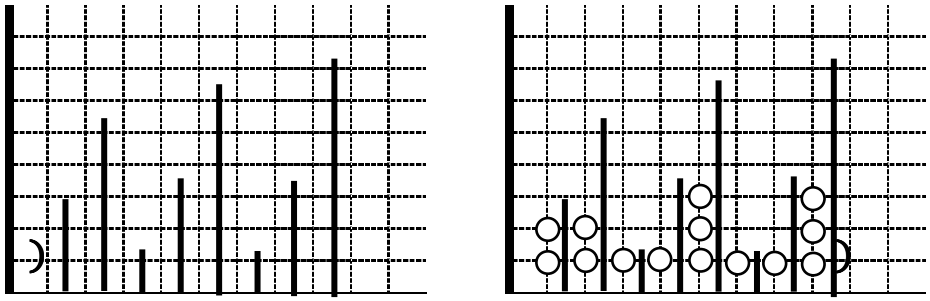


Figura 60. Condiciones iniciales y finales del problema "Coleccionista de carros"