Ejemplo de cómo resolver un problema

Una vez que has comprendido la importancia de contar con una metodología para hacer programas, te mostraré un ejemplo de cómo los debes resolver. Recuerda que esto es un entrenamiento, ya que debes cambiar los hábitos en la manera de hacer los programas.

Preparemos nuestras herramientas de trabajo: 3 hojas en blanco y nuestro lápiz bien afilado. A continuación veremos el problema que debes de resolver y para el cual debemos llegar a resolverlo en todos los casos, ó sea que nos dé 100 puntos. Sin más preámbulo, vamos a leerlo.

Problema: Los canales del lago

Descripción

Como es bien sabido por todos, el lago de Xochimilco es un hermoso lago que año con año, y desde hace muchísimo tiempo, atrae una gran cantidad de turistas, con el propósito de poder dar un viaje en el lago en una de las trajineras.

Durante el recorrido, uno puede comprar comida, contratar a los mariachis, y disfrutar del paisaje; todo eso sobre las barcazas.

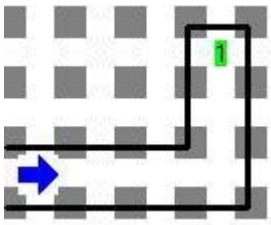
El viaje en trajinera consiste en recorrer los canales que han sido construidos sobre del lago, empezando en un lugar y terminando en otro.

Problema

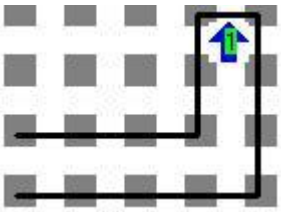
El día de hoy Karel va a visitar el lago, y tú, como dueño de la trajinera Morpheus, tienes que llevarlo por el recorrido.

Consideraciones

- Karel inicia al principio del recorrido orientado hacia donde éste se dirige.
- Karel no tiene zumbadores en la mochila.
- Los canales del lago son muy angostos, y se representan con un camino de ancho de 1.
- Mientras vas en el recorrido, recuérdale a Karel no tirar basura en el lago.
- Puedes saber que llegaste al final del recorrido, porque encontrarás un beeper.
- El camino sólo se cierra al final del recorrido, junto al beeper.
- Karel debe de terminar en el final del recorrido.

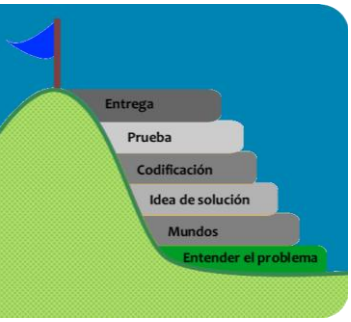


Mundo de ejemplo



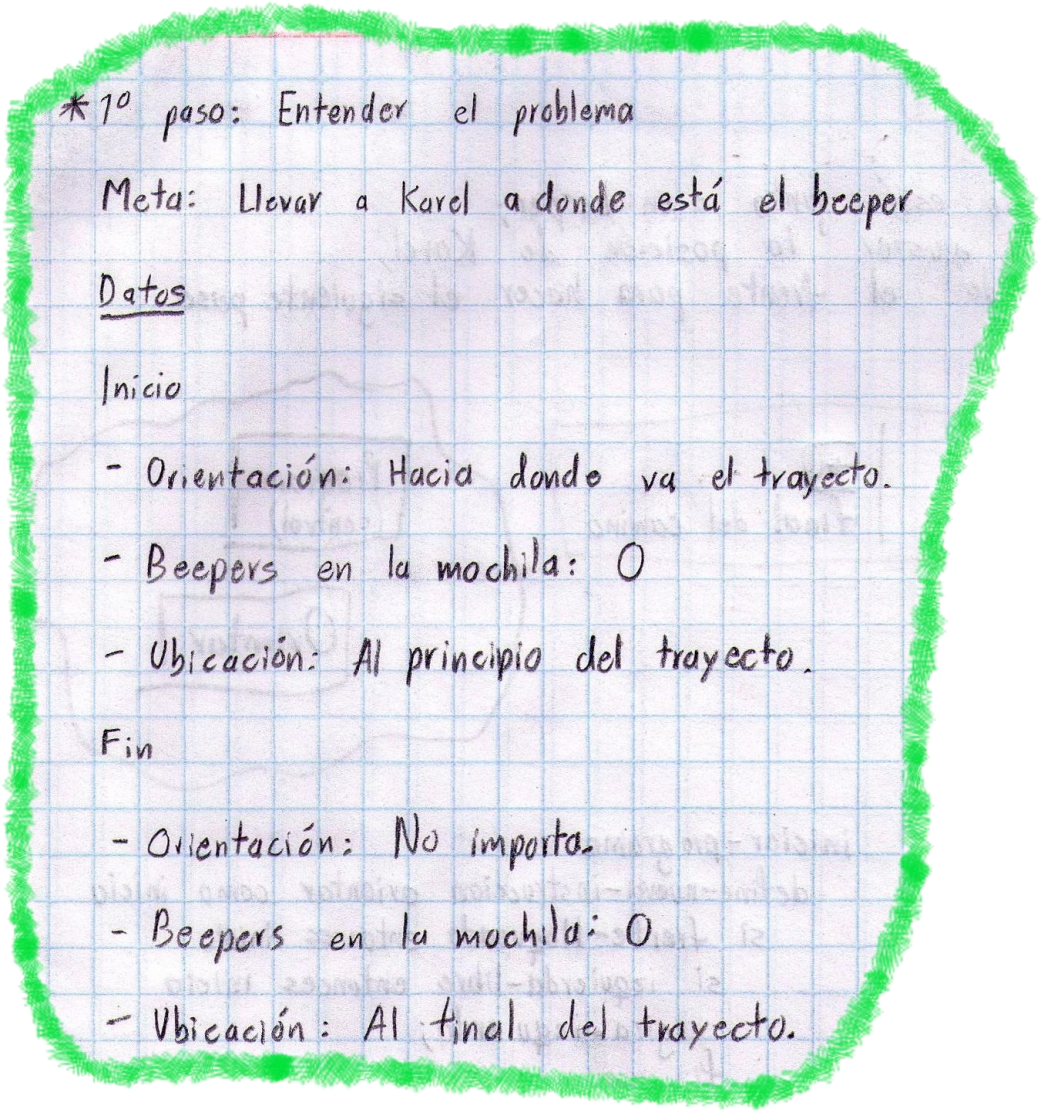
Solución al mundo de ejemplo

1° Paso: Entender el problema



Lo primero que debemos hacer es analizar nuestro problema; para esto lo debemos leer hasta que nos quede claro, si queda alguna duda, la debemos preguntar.

Para saber que ya lo entendimos, debemos ser capaces de explicarlo a otra persona y a nosotros mismos. Una manera de saber que estamos bien es sacar los datos del problema de manera similar a como si fuera un problema de matemáticas y anotarlos en nuestra hoja; estos datos serían: zumbadores en la mochila, orientación inicial de Karel, límites del problema, orientación final de Karel, etc. En la siguiente figura se ve un ejemplo para este problema:



* 1° paso: Entender el problema

Meta: Llevar a Karel a donde está el beeper

Datos

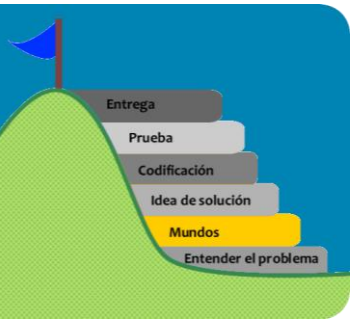
Inicio

- Orientación: Hacia donde va el trayecto.
- Beepers en la mochila: 0
- Ubicación: Al principio del trayecto.

Fin

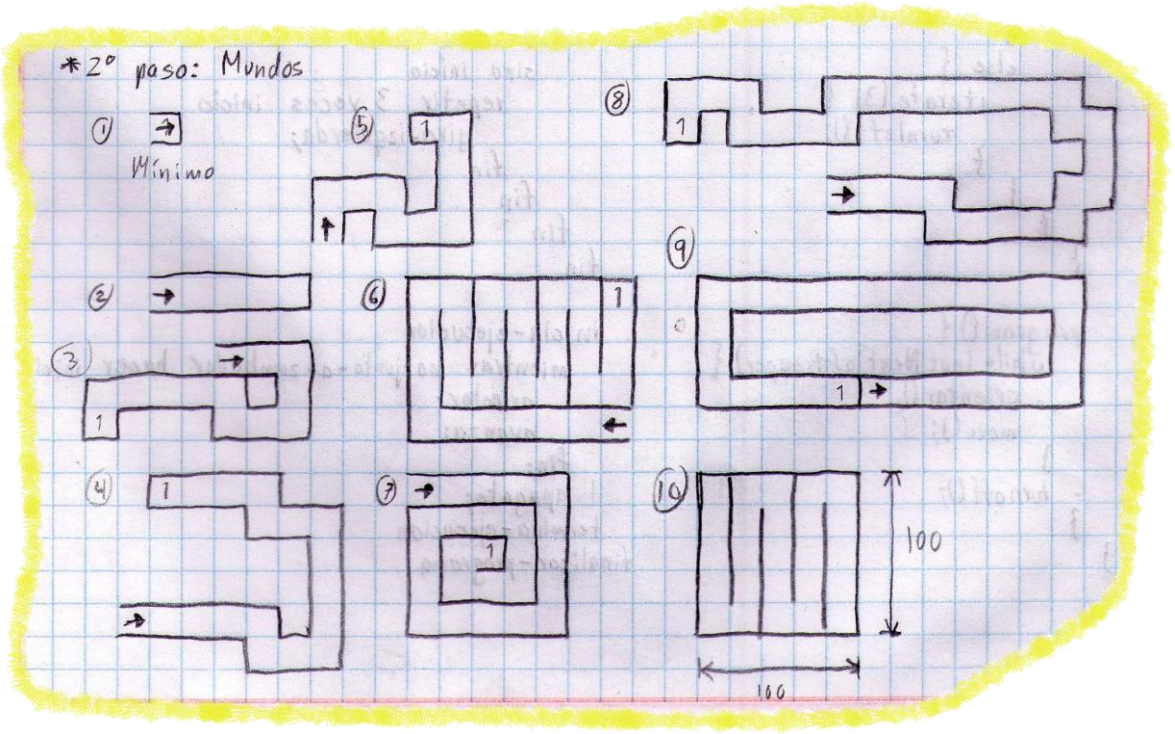
- Orientación: No importa.
- Beepers en la mochila: 0
- Ubicación: Al final del trayecto.

2° Paso: Mundos



Una vez que tenemos una idea más clara de los que nos piden, es bueno pensar en los diferentes mundos a los que aplica la descripción del problema (te recuerdo que realizar esto cuenta entre un 20% y un 40% de tu calificación final), ya que como se mencionó anteriormente, si podemos pensar en las diferentes posibilidades con que nos van a evaluar, seguramente nos llevará a pensar una solución que satisfaga el 100% de los mundos con que nos calificarán.

En la siguiente figura podemos apreciar que fuimos capaces de encontrar 10 casos diferentes y si hacemos un programa que sea capaz de resolver estos 10 casos, ten la seguridad de que tendrás una alta calificación.



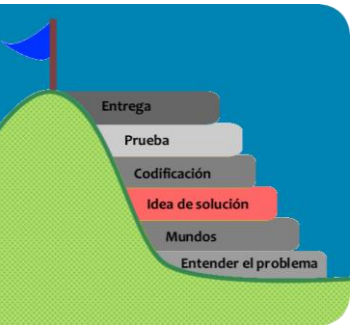
Revisando los casos, puedes ver que no es necesario dibujar perfectamente el mundo de Karel, sino que debemos escribirlos de una forma clara, sencilla y rápida. Por ejemplo, en el caso 10 que mide 100x100, se indica el ancho y largo de 100 con unas señales de acotamiento similares a las que se usan en planos.

A continuación haremos algunas observaciones de los casos desarrollados:

- ✓ **Caso 1**: Es el caso mínimo, ya que no existe un caso más pequeño que este.
- ✓ **Caso 8**: Muestra un caso en el que se tiene que ajustar el camino de Karel en las 4 direcciones posibles: norte, sur, este y oeste. Este caso lo podríamos considerar como un caso máximo en cuanto a las direcciones que debe de tomar Karel para llegar a su objetivo (en este caso llegar al final del canal).
- ✓ **Caso 10**: Muestra un caso máximo en cuanto a longitud del recorrido, ya que debe atravesar todo el mundo (si te da tiempo, constrúyelo; si no, trata de hacer algo que lo iguale en condiciones y características).
- ✓ **Otros casos**: Los podemos considerar como casos típicos o promedio. Siempre en bueno hacer 3 o 4 casos promedio.

Te aseguro que si tienes la suficiente paciencia y dedicación para trabajar de esta forma de ahora en adelante en cada unos de tus problemas de programación, el resultado que obtendrás te llevará a consolidarte como un verdadero ganador.

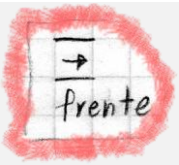
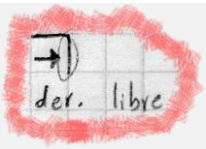
3° Paso: Idea de solución



Una vez que ya hemos comprendido el problema, debemos pensar en alguna solución (más adelante se hablará más de cómo ayudarse a tener ideas de solución). De una manera simple nos podemos basar en las siguientes 3 ideas:

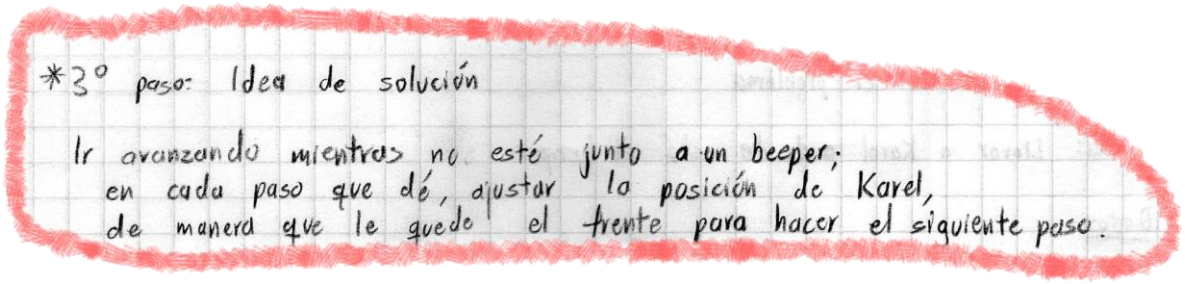
"Otra forma de plantearlo", "subdividir en casos", "se parece a algo ya resuelto".

En este caso propongo "otra forma de plantearlo" debido a que podemos ver el problema como hacer que Karel camine hasta que encuentre un zumbador. La idea es hacer un programa que haga que Karel camine mientras no esté junto a un zumbador. Debemos considerar que Karel siempre inicia en la posición en que debe caminar, por lo cual se asegura que no debemos ajustar su posición al inicio; sin embargo, como el camino presenta algunos quiebres (variaciones de orientación hacia donde caminará Karel), cada vez que Karel camine una posición, se debe ajustar la orientación de Karel con la finalidad de lograr una posición en que pueda moverse (de ser posible, una posición de frente libre). De los casos que construimos, podemos apreciar que cada vez que avanza Karel pueden existir las siguientes 4 situaciones:

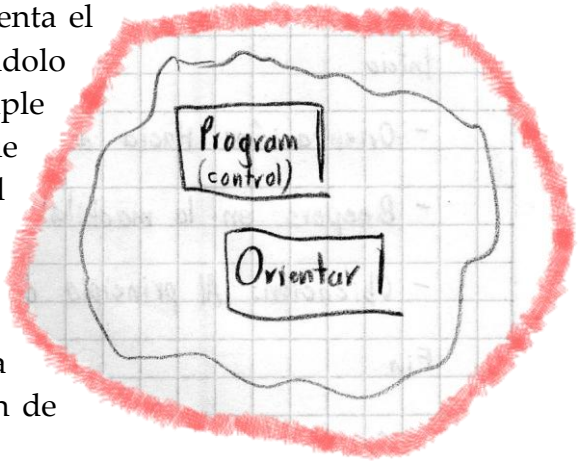
1. Si está junto a un zumbador, ahí termina. No debemos hacer nada porque ahí termina el programa.	
2. Que el frente esté libre. Para este caso no debemos hacer ningún ajuste.	
3. Que el frente esté bloqueado y la izquierda libre. En este caso debemos ajustar la orientación de Karel haciéndolo girar una vez a la izquierda.	
4. Que el frente esté bloqueado y la derecha libre. En este caso debemos ajustar la orientación de Karel haciéndolo girar a la derecha (esto se hace con 3 giros a la izquierda de Karel).	

Con lo que vemos, el problema se resume a un problema sencillo de hacer: que Karel camine y ajuste su orientación mientras no esté junto a un zumbador.

Es importante que te des cuenta que todo esto se hace en la mente y, ayudado de nuestras herramientas exclusivas (papel y lápiz),; nosotros debemos anotar la idea para que no se nos olvide como se muestra a continuación:

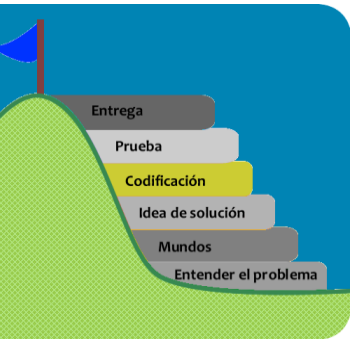


Nuestro programa se puede ver muy simple. Nos ayudaremos de un diagrama de bloques (llamado de bloques porque usamos cuadros para representar las ideas generales). Como podemos apreciar, la nube que rodea nuestros bloques representa el problema que una vez habiéndolo dividido en 2, resulta más simple resolver 2 problemas menores que uno mayor. Esta figura muestra el *bloque de control* que hará que "Karel camine mientras no está junto a un zumbador", y el bloque de la nueva instrucción llamada *orientar* que "ajustará la orientación de Karel en caso de ser necesaria".



En este momento, ya estamos listos para planificar el problema de acuerdo a la idea de la sección anterior, es decir, construir la escalera para llegar nuestra meta que se ubica en la cima. Esto no lo debemos hacer; sin embargo, en este apunte lo mostraré para ver cómo se unen los conceptos teóricos de elaboración de un programa con la práctica de hacer un programa. Como sólo se muestra de una manera ilustrativa, utilizaremos los recuadros redondeados. Y el tipo de letra en cursivas.

4° Paso: Codificación



Ahora lo que debemos hacer es desarrollar el programa de control y cada una de las nuevas instrucciones que vamos a necesitar. En este caso son el *programa principal* que tiene la rutina de control y la nueva instrucción *orientar*.

Las siguientes figuras muestran el desarrollo en papel de las rutinas más importantes, ya que las más pequeñas y sencillas se pueden hacer directamente en el teclado cuando ya se tiene algo de experiencia. Mientras adquieres la experiencia, te aconsejo que trates de escribirlas todas en el papel antes de escribirlas en el Karel.

El programa principal quedaría de la siguiente manera en Java y/o Pascal; si lo observamos sigue nuestra idea: mientras Karel no esté junto a un zumbador, avanza y se orienta.

```
program () {
  while (notNextToABeeper) {
    orientar();
    move();
  }
  turnoff();
}
```

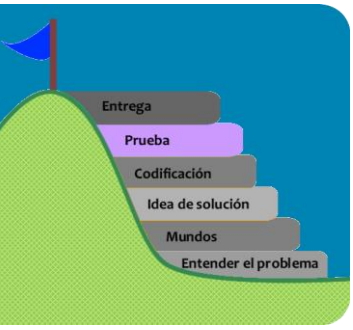
```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
    avanza;
  fin;
  apagate;
termina-ejecucion
```

En la parte de la orientación quedaría como lo muestra la siguiente figura. Una buena práctica es hacerle una prueba de escritorio y ver que funciona en todos los casos.

```
program {
void orientar () {
  if (frontIsBlocked) {
    if (leftIsClear) {
      turnleft();
    }
    else {
      iterate (3) {
        turnleft();
      }
    }
  }
}
```

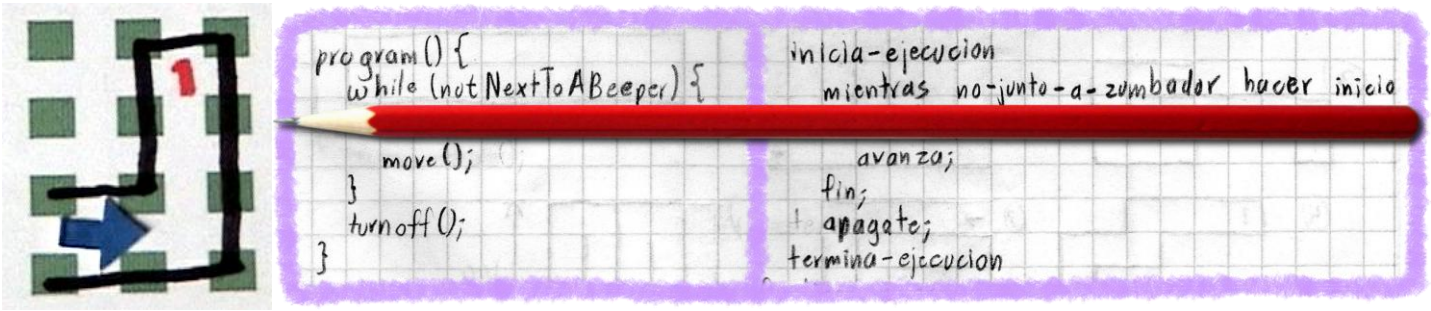
```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
    fin
    sino inicio
      repetir 3 veces inicio
        gira-izquierda;
      fin
    fin
  fin
fin
```

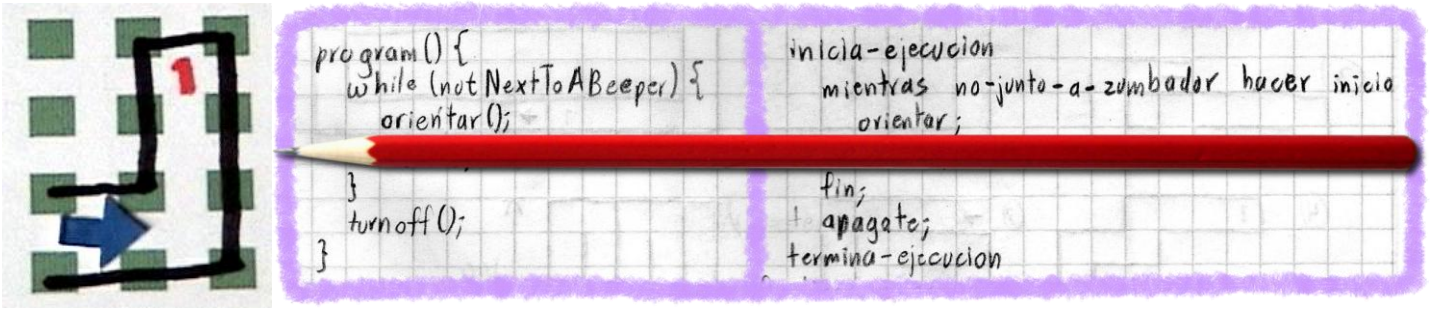
5° Paso: Prueba

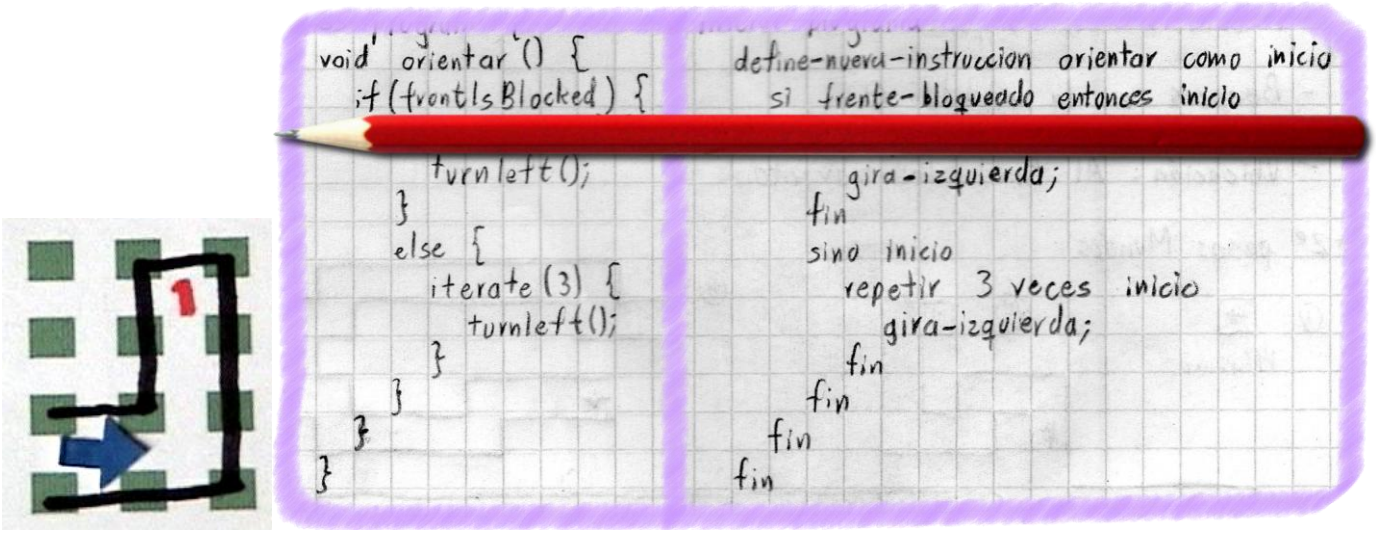


Haremos la prueba de escritorio (recuerda que se llama así porque la hacemos en el escritorio de trabajo utilizando un Karel y nuestras hojas de papel).

Haremos esta prueba para el caso de que Karel deba ajustar su orientación a la izquierda como se muestra a continuación. Nuestra rutina de control haría que Karel comprobara su posición y caminara un paso:

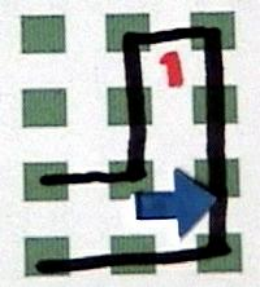




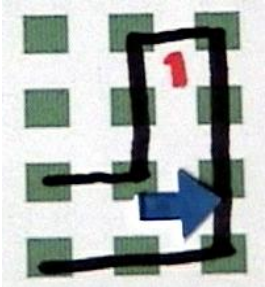


```
program() {
  while (notNextToABeeper) {
    orientar();
    move();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
    avanza;
  fin;
  apagate;
termina-ejecucion
```

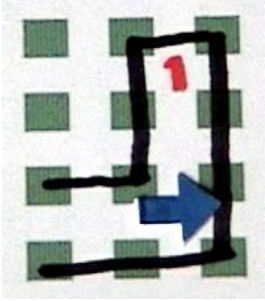


Dejándonos en esta posición, se llamará a la rutina *orientar*, la cual pregunta si el frente está bloqueado, cumpliéndose en este caso: por lo tanto, pregunta si la izquierda está libre, siendo afirmativamente, haciendo que Karel haga un giro a la izquierda.



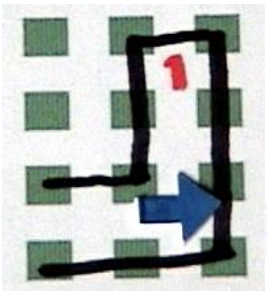
```
program() {
  while (notNextToABeeper) {
    orientar();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
  fin;
  apagate;
termina-ejecucion
```



```
void orientar() {
  if (frontIsBlocked) {
    turnleft();
  }
  else {
    iterate(3) {
      turnleft();
    }
  }
}
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    gira-izquierda;
  fin
  sino inicio
    repetir 3 veces inicio
      gira-izquierda;
    fin
  fin
fin
fin
```

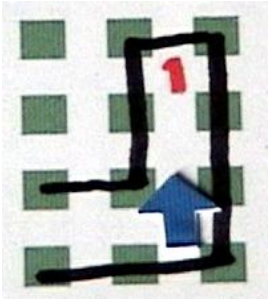



```
void orientar() {
  if(frontIsBlocked) {
    if(leftIsClear) {
      }
      else {
        iterate(3) {
          turnleft();
        }
      }
    }
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      fin
      sino inicio
        repetir 3 veces inicio
          gira-izquierda;
        fin
      fin
    fin
fin
```

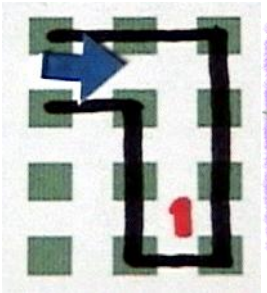
```
void orientar() {
  if(frontIsBlocked) {
    if(leftIsClear) {
      turnleft();
    }
    else {
      iterate(3) {
        turnleft();
      }
    }
  }
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
      sino inicio
        repetir 3 veces inicio
          gira-izquierda;
        fin
      fin
    fin
fin
```



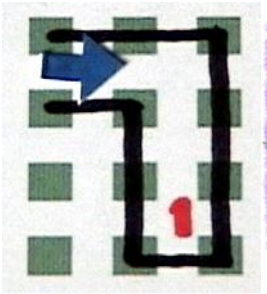
Dejándonos, finalmente, en la posición correcta para que siga su recorrido.

Ahora haremos la prueba que cumpla en el caso de que Karel se deba mover hacia la derecha. Como se muestra a continuación, nuestra rutina de control hace que Karel camine un paso.



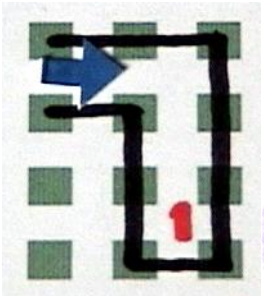
```
program() {
  while (notNextToABeeper) {
    move();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    avanza;
  fin;
  apagate;
termina-ejecucion
```



```
program() {
  while (notNextToABeeper) {
    orientar();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
  fin;
  apagate;
termina-ejecucion
```

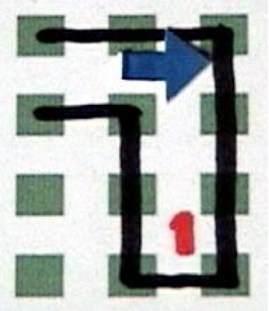


```
void orientar() {
  if (frontIsBlocked) {
    turnleft();
  }
  else {
    iterate (3) {
      turnleft();
    }
  }
}
}
```

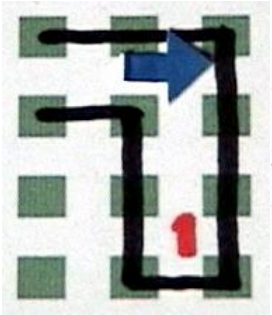
```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    gira-izquierda;
  fin
  sino inicio
    repetir 3 veces inicio
      gira-izquierda;
    fin
  fin
fin
fin
```

```
program() {
  while (notNextToABeeper) {
    orientar();
    move();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
    avanza;
  apagate;
termina-ejecucion
```



Dejándonos en esta posición, se llama a nuestra rutina *orientar*, la cual pregunta si el frente está bloqueado, esto siendo afirmativo; entonces pregunta si la izquierda está libre, siendo negativo para este caso; entonces, Karel hará 3 giros a la izquierda:

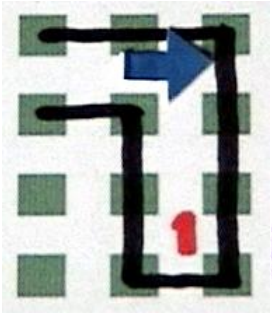


```
program() {
  while (notNextToABeeper) {
    orientar();
  }
  turnoff();
}
```

```
inicia-ejecucion
  mientras no-junto-a-zumbador hacer inicio
    orientar;
  fin;
  apagate;
termina-ejecucion
```

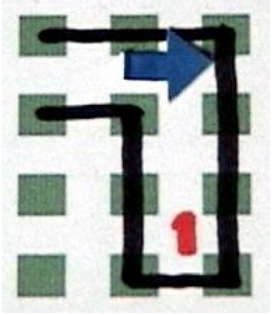
```
void orientar() {
  if (frontIsBlocked) {
    turnleft();
  }
  else {
    iterate(3) {
      turnleft();
    }
  }
}
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    gira-izquierda;
  fin
  sino inicio
    repetir 3 veces inicio
      gira-izquierda;
    fin
  fin
fin
fin
```

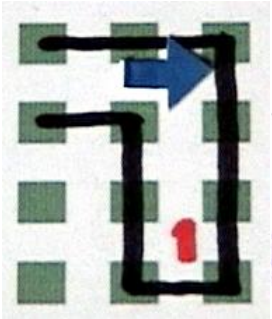


```
void orientar() {
  if (frontIsBlocked) {
    if (leftIsClear) {
    }
    else {
      iterate(3) {
        turnleft();
      }
    }
  }
}
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      fin
      sino inicio
        repetir 3 veces inicio
          gira-izquierda;
        fin
      fin
    fin
  fin
fin
```

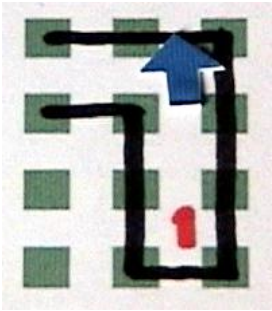


Al final, nos dejará en la posición correcta.



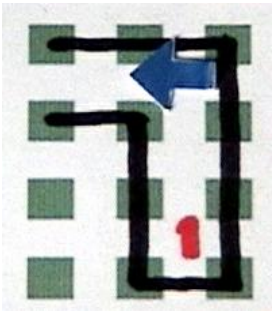

```
void orientar() {
  if (frontIsBlocked) {
    if (leftIsClear) {
      turnleft();
    }
    else {
      iterate (3) {
      }
    }
  }
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
    fin
    sino inicio
      repetir 3 veces inicio
      fin
    fin
  fin
fin
```




```
void orientar() {
  if (frontIsBlocked) {
    if (leftIsClear) {
      turnleft();
    }
    else {
      iterate (3) {
        turnleft();
      }
    }
  }
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
    fin
    sino inicio
      repetir 3 veces inicio
        gira-izquierda;
      fin
    fin
  fin
fin
```

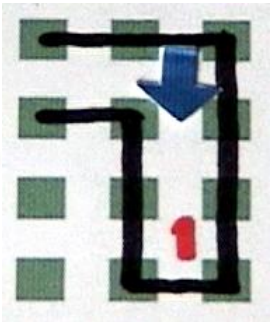



```
void orientar() {
  if (frontIsBlocked) {
    if (leftIsClear) {
      turnleft();
    }
    else {
      iterate (3) {
        turnleft();
      }
    }
  }
}
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
    fin
    sino inicio
      repetir 3 veces inicio
        gira-izquierda;
      fin
    fin
  fin
fin
```

```
void orientar() {
  if (frontIsBlocked) {
    if (leftIsClear) {
      turnleft();
    }
    else {
      iterate (3) {
        turnleft();
      }
    }
  }
```

```
define-nueva-instruccion orientar como inicio
  si frente-bloqueado entonces inicio
    si izquierda-libre entonces inicio
      gira-izquierda;
    fin
    sino inicio
      repetir 3 veces inicio
        gira-izquierda;
      fin
    fin
  fin
```



En este punto ya estamos seguros que nuestro programa va a funcionar bastante bien.

A continuación, te muestro las hojas de trabajo que se realizaron antes de hacer el programa en la computadora.

Los canales del lago

*1º paso: Entender el problema

Meta: Llevar a Karel a donde está el beeper

Datos

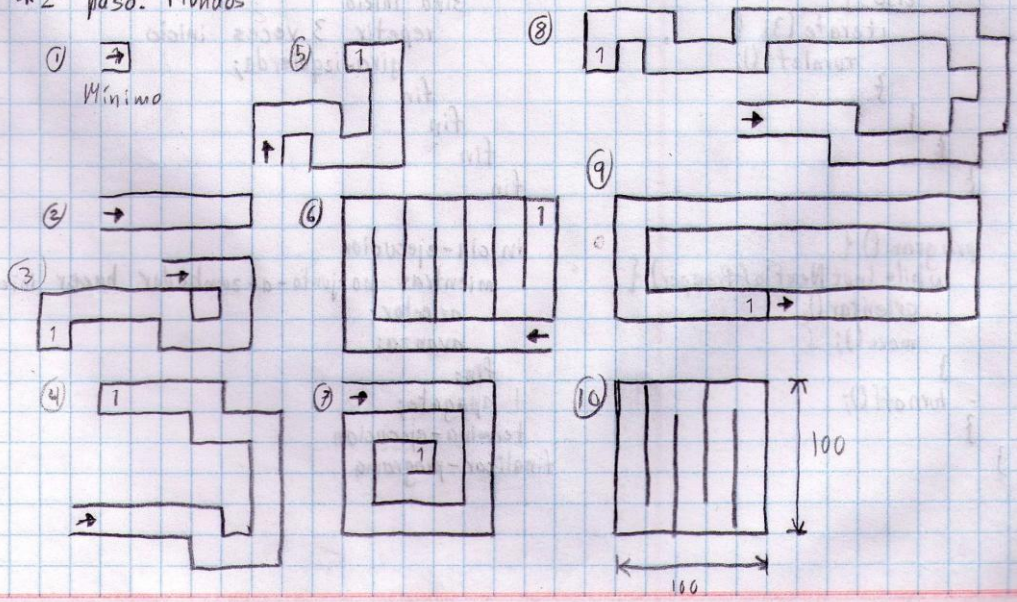
Inicio

- Orientación: Hacia donde va el trayecto.
- Beepers en la mochila: 0
- Ubicación: Al principio del trayecto.

Fin

- Orientación: No importa.
- Beepers en la mochila: 0
- Ubicación: Al final del trayecto.

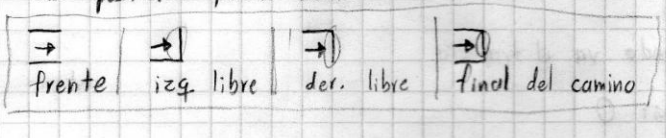
*2º paso: Mundos

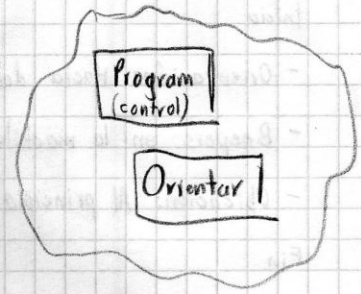


*3° paso: Idea de solución

Ir avanzando mientras no esté junto a un beeper;
en cada paso que dé, ajustar la posición de Karel,
de manera que le quede el frente para hacer el siguiente paso.

Los posibles pasos son:





*4° paso: Código

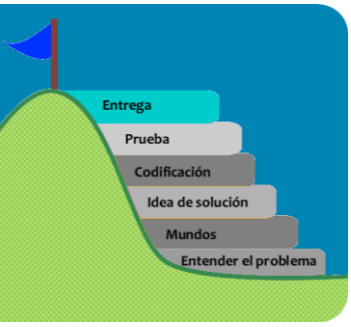
```
class program {
  void orientar() {
    if (frontIsBlocked) {
      if (leftIsClear) {
        turnleft();
      }
      else {
        iterate (3) {
          turnleft();
        }
      }
    }
  }

  program() {
    while (notNextToABeeper) {
      orientar();
      move();
    }
    turnoff();
  }
}
```

```
iniciar-programa
  define-nueva-instruccion orientar como inicio
    si frente-bloqueado entonces inicio
      si izquierda-libre entonces inicio
        gira-izquierda;
      fin
      sino inicio
        repetir 3 veces inicio
          gira-izquierda;
        fin
      fin
    fin
  fin

  inicia-ejecucion
    mientras no-junto-a-zumbador hacer inicio
      orientar;
      avanza;
    fin;
    apagate;
  termina-ejecucion
finalizar-programa
```

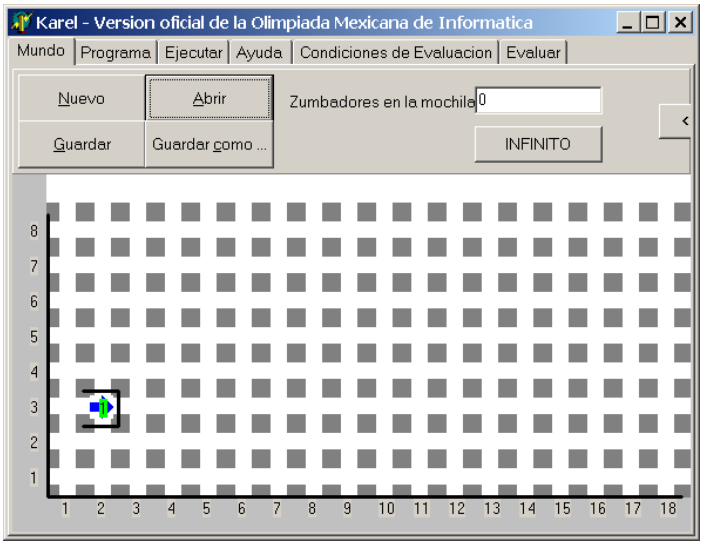
6° Paso: Entrega



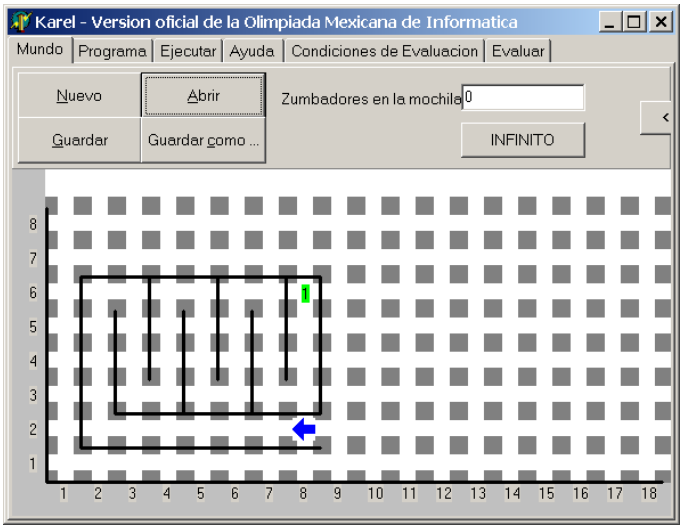
Lo que vamos hacer ahora es escribir los casos en el Karel. Recuerda que es importante guardar los mundos (casos), ya que debemos evitar estar reconstruyendo mundos si nuestro programa presenta una falla.

Una buena práctica es guardarlo con el nombre del problema seguido de un número.

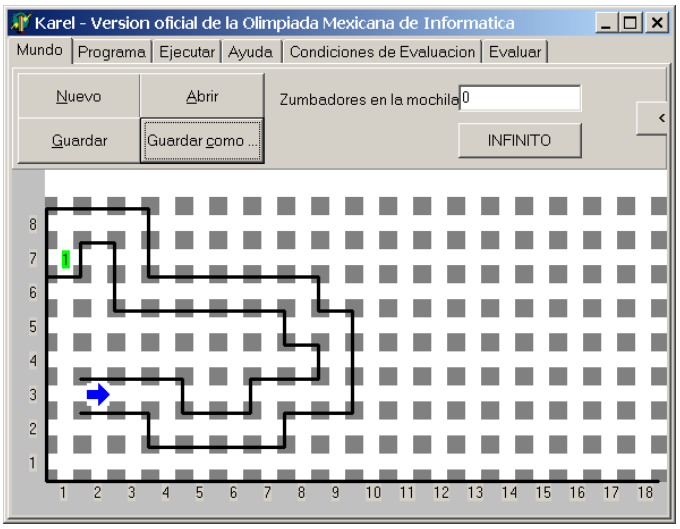
A continuación se muestran las figuras de 3 mundos construidos (recuerda que debes hacer los 10 mundos y guardarlos):



Ejemplo del mundo mínimo, el cual tiene el nombre de archivo `canales_01.mdo`.

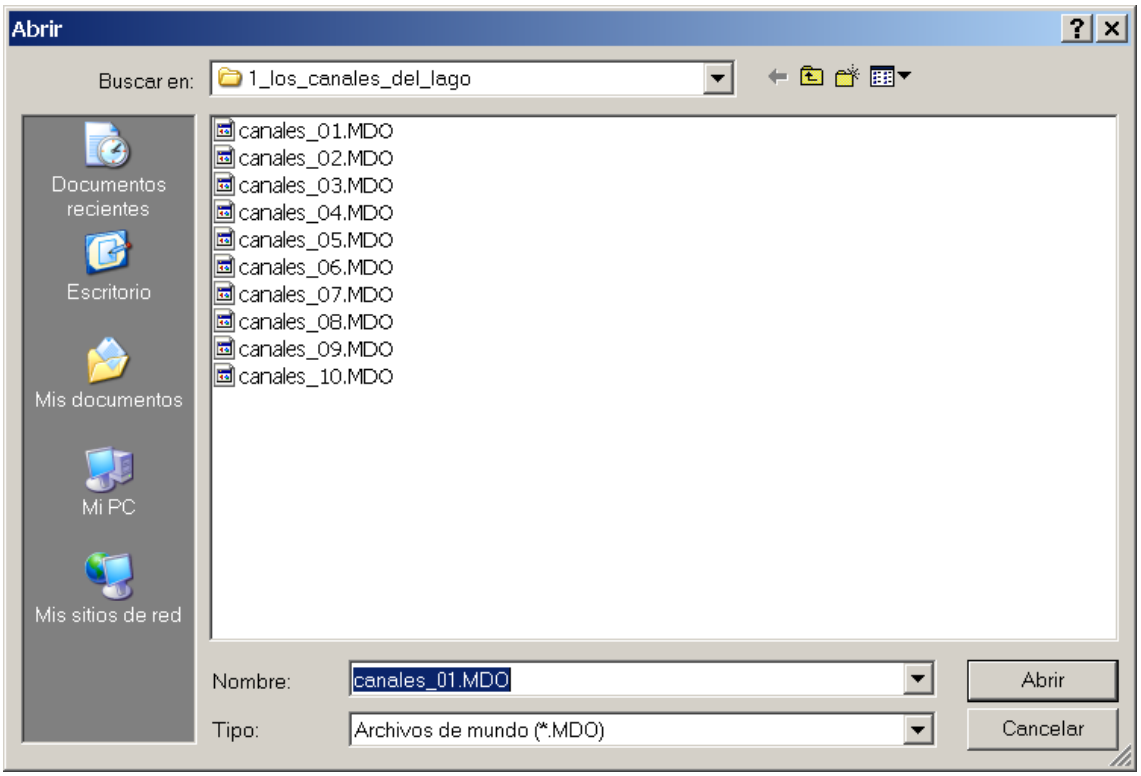


La siguiente imagen representa el caso promedio, representando el mundo número 6 (`canales_06.mdo`).

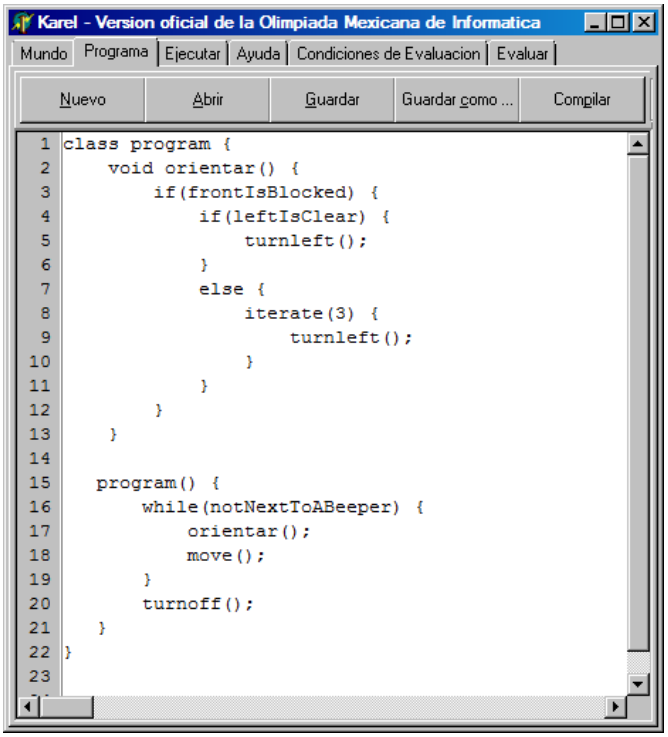


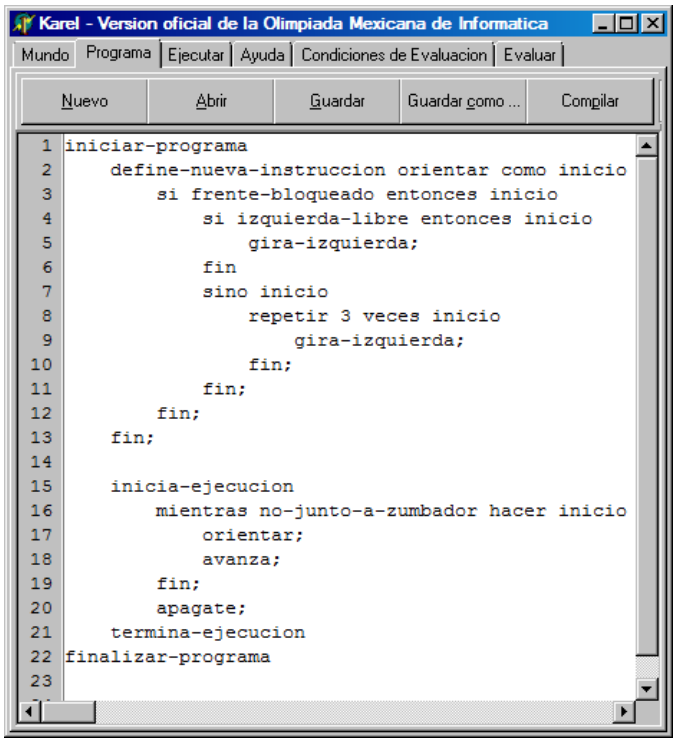
La siguiente figura representa el mundo 8 (`canales_08.mdo`). Este mundo representa todas las posibles variantes que puede tener en cuanto a cambios de orientación: hacia el norte, hacia el este, hacia el oeste y hacia el sur.

A continuación les muestro la imagen de abrir un mundo en donde vemos que ya están guardados los 10 mundos del problema "Los canales del lago".

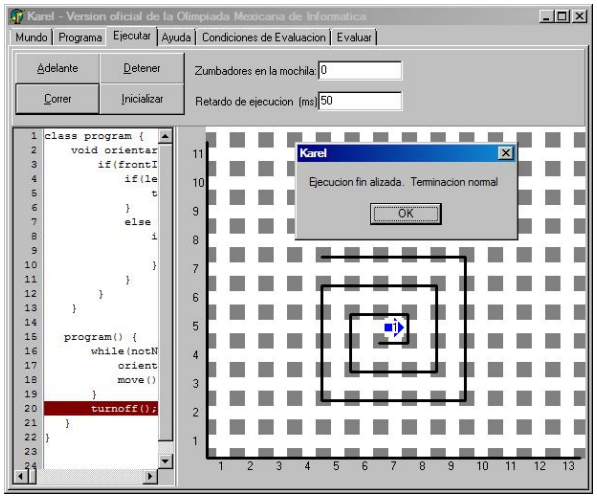


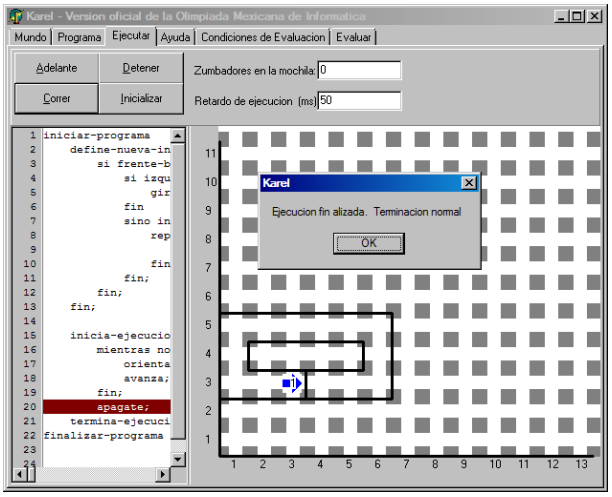
Posteriormente pasamos a escribir el programa en Karel, como se muestra en la siguiente figura. Debes fijarte que el programa se escribe **indentando** las instrucciones y como verás se ve muy claro (yo diría que hasta se ve elegante).





Ahora sólo falta probarlo con todos los mundos (casos) que fuiste capaz de pensar. Si al llegar a este punto descubres un error en tu idea de solución, te darás cuenta del tiempo que te ahorrarás al haber guardado tus mundos, ya que no los volverás hacer cada vez que tengas que probar tu programa. A continuación se muestran 2 mundos que fueron probados satisfactoriamente, tú sólo debes hacer el programa en un solo lenguaje; aquí muestro un ejemplo en Java y el otro en Pascal, porque este libro esta dirigido a los profesores y alumnos de ambos lenguajes.





Por último, como podrás ver, tu programa funciona porque seguramente ya lo probaste con todos los casos y no tuviste error alguno, así que ya lo puedes entregar o continuar haciendo el siguiente problema, con la seguridad de que tu solución te dará los 100 puntos del problema. Te recomiendo que vayas midiendo el tiempo de elaboración de un programa; en cuanto domines el procedimiento para hacer un programa verás que es muy sencillo y rápido hacer un problema.

Tips para alumnos

A continuación te hago una lista en importancia de los tips que nunca debes de brincarte y que te ayudarán a lograr un mejor desempeño.

1. Construir tus casos antes de pensar en una solución; de esta manera tu solución te dará el 100% de los casos.
2. Por cada nueva instrucción que escribas debes hacer la prueba de escritorio, este punto te ayudará en un 30% de tu calificación y te ahorrará mucho tiempo.
3. Escribe tus mundos en Karel y guárdalos en caso de que tu programa falle. Así, no tendrás que reescribir los mundos para volverlo a probar, esto nos ahorrará tiempo y nos dará seguridad.
4. Entregar tu programa hasta que lo hayas probado con todos las variantes de mundos que fuiste capaz de encontrar.
5. Si no tienes experiencia en Karel, escribe todas las nuevas instrucciones en papel.
6. Trata de hacer tu Karel de papel para ayudarte en tu prueba de escritorio.
7. Indentar de manera adecuada tu programa para que sea más fácil de entenderlo en caso de que debas ajustar algunas instrucciones.

Tips para profesores

A continuación te hago énfasis en lo que ayudará a mejorar el desempeño de tus alumnos en las clases.

1. Primero, debes creer en lo que estás haciendo. En caso de no estar totalmente de acuerdo con esta guía, te invito a que la modifiques de acuerdo a tu experiencia y grado escolar, y me dará mucho gusto que la compartas conmigo.
2. Ser muy estricto en cuanto al procedimiento a seguir. Te sugiero que les apagues las pantallas de sus computadoras hasta que te presenten en su cuaderno los datos del problema y los casos y su solución. Una vez que hagan esto, tú mismo les enciendes el monitor.
3. Ser exigente con la identación para que los chicos se hagan de un buen hábito al escribir programas de computadora.
4. Si los chicos no tienen mucha experiencia o son muy pequeños, te sugiero que les entregues en hojas de papel blanco el mundo de Karel y los hagas construir un Karel de papel a fin de poder realizar las pruebas de escritorio de sus programas de una manera sencilla y amena. En el anexo 1 del libro existe una hoja de papel con el la imagen del mundo de Karel y un Karel listo para fotocopiar.
5. Te recomiendo mucho leer los problemas que vas a poner y revisar las soluciones antes de presentarte ante el grupo.

Anexo 1 – Karel y su mundo, listos para recortar