

2ej
18

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS



**CONSTRUCCION DE UN INTERPRETE
PARA KAREL**

T E S I S
QUE PARA OBTENER EL TITULO DE:
A C T U A R I O
P R E S E N T A :
SUSANA DEL CARMEN FUENTES HERRERA

MEXICO, D. F.

1987.



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

I N D I C E

Introducción.....	1
I) Presentación y entorno de Karel.....	3
II) Karel como instrumento educativo.....	8
III) El Lenguaje de Karel.....	13
A) Comando Ejecuta.	
A.1) Instrucciones simples.....	14
A.2) Palabras condicionales.....	15
A.3) Instrucciones de control.....	16
B) Comando Acepta.....	19
C) Comando Modifica.....	20
D) Ejemplos.....	21
IV) Descripción de la estructura del intérprete de Karel.	
A) Registros asociados a la graficación del entorno de Karel.....	27
B) Estructura y registros asociados al intérprete.	
B.1) Directorio central.....	29
B.2) Registros asociados al ensamblador.....	30
C) Compilador.....	31
D) Definición de la estructura gramatical.....	35
E) El intérprete de Karel.	
E.1) El ciclo "SI".....	40
E.2) El ciclo "RM".....	41
E.3) El ciclo "RP".....	42
E.4) Estructura general del autómata.....	44
E.5) Tabla de transiciones y salidas asociadas al autómata de control.....	46
E.6) Procesos asociados al autómata de control.....	47
V) Conclusiones.....	61

INTRODUCCION.

La aparición de herramientas de cómputo para aprender a programar se remonta a hace más de 15 años; entre ellas, se puede mencionar al lenguaje LOGO desarrollado por Bolt, Beranek y Newman en la Universidad de Cambridge, Massachusetts, usado al inicio en laboratorios de investigación.

Fue Seymour Papert el primero en concebir a dicho lenguaje como un instrumento educacional visto de la siguiente forma:

Los usuarios de LOGO pueden controlar un robot externo llamado "Tortuga", la cual dibuja sobre el video de una computadora generando gráficos animados. El usuario utiliza para ello instrucciones básicas, tales como: muévete n unidades, gira m grados, etc. la "Tortuga" al moverse va dejando su trayectoria dibujada, aunque también es posible indicarle que no dibuje al moverse. La utilización de estas instrucciones permite que el usuario relacione sus razonamientos abstractos y los evalúe mediante la experiencia visual al momento de ejecución.

Las ventajas que ofrecen este tipo de lenguajes han apoyado al desarrollo de Programas para el aprendizaje como, por ejemplo, el robot llamado "KAREL", del cual me ocuparé en esta tesis.

"KAREL" es un robot concebido por el Dr. Richard E. Pattis de la Universidad de Stanford, para la enseñanza básica de la programación. El Robot se maneja a con un conjunto de instrucciones que conforman un lenguaje totalmente imperativo, con el que el estudiante aprende rápidamente a escribir programas bien estructurados.

El Dr. Pattis menciona que un argumento a favor de KAREL está dado en base al síndrome del "aislado a la computación", debido a la gran relación que tiene la programación con los números, mientras que en KAREL los fundamentos para programar están planteados de una forma diferente; dicha diferencia radica en el hecho de que se trabaja con un mundo más intuitivo y de más fácil comprensión como se verá más adelante.

Se le dió el nombre de KAREL a este robot, en memoria al escritor Checoslovaco Karel Capek, primer autor que manejara el término "ROBOT" para una máquina capaz de substituir al hombre en la realización de ciertas actividades, recurriendo, quizá, a la raíz de la palabra robot que proviene del checo "ROBOTA" que significa "trabajos forzados".

En la presente tesis se plantea la construcción de un programa intérprete para el conjunto de instrucciones que manejan a KAREL, usando un autómata.

La construcción inicial del intérprete está programada en lenguaje BASIC para una microcomputadora LNW-80, aun cuando el programa puede fácilmente adaptarse a microcomputadoras más pequeñas como la VIC-20.

En el primer capítulo.- "el entorno de Karel", se describirá detalladamente ese mundo intuitivo con el que KAREL trabaja y que constituye el medio ambiente en el que el robot vive. En él nos encontramos dos tipos de objetos:

- Ajenos a Karel.- como son los trompos y las barreras.
- Los de Karel.- como son los instrumentos con los que cuenta para la realización de las tareas, estos son: una brújula para orientarse en su medio ambiente, un detector de sonidos para identificar a los trompos, y un saco a prueba de zumbidos donde irá guardando los trompos que recoja.

En el segundo capítulo.- "Karel como instrumento educativo", se plantea la idea de como mediante la utilización de Karel, el alumno aprende y qué tipo de problemas se pueden resolver o plantear con el robot.

En el tercer capítulo.- "El lenguaje de Karel", se explica detalladamente cada una de las instrucciones que el alumno puede incluir en la programación de "KAREL".

En el cuarto capítulo.- "La estructura de Karel", se explica la formación del autómata que hace funcionar a KAREL, los registros y estructuras asociadas al intérprete, así como cada uno de los procesos asociados al autómata.

En el quinto capítulo, se dan algunas conclusiones sobre la utilización de KAREL como instrumento de enseñanza en escuelas de educación primaria y secundaria.

I EL ENTORNO DE KAREL.

El entorno de KAREL es el medio ambiente en el cual se mueve el robot (éste deberá ser construido por el alumno al inicio de la sesión).

El Dr. Pattis menciona a este mundo como una ciudad en la que existen calles perfectamente bien trazadas con barreras: por ello es posible representar a este mundo como una malla.

Para el KAREL que se construyó, se consideró a esta malla formada de 20 renglones y 21 columnas (420 celdas), representado de la siguiente manera:

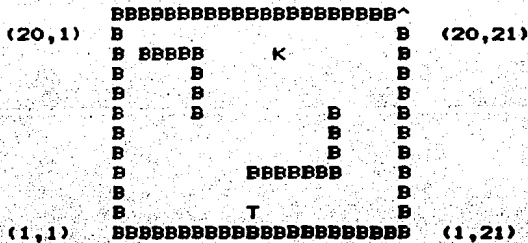


Figura 1.

En donde:

- K ----- Es el robot "Karel".
- T ----- Es el trofeo.
- B ----- Son las barreras.

Los límites del mundo de KAREL están hechos del mismo material de las barreras.

Esta malla se puede apreciar en un video de microcomputador ó en una pantalla de televisión.

La localización de una posición en la malla, se hace por medio de coordenadas, en donde cada celda tiene asignada una pareja de números comprendidos desde la posición (1,1) hasta la posición (20,21).

Los elementos que componen el entorno de KAREL son:

- 1.- Barreras.
- 2.- Trompos.

Estos elementos pueden ser contruidos mediante el uso de algunas teclas de la microcomputadora, en este caso existiran cuatro teclas que permiten el movimiento del cursor sin dibujar, las cuales son:

TECLA	FUNCION
W	Movimiento hacia arriba.
Z	Movimiento hacia abajo.
S	Movimiento a la derecha.
A	Movimiento a la izquierda.

Cuando se quiere construir una barrera, primero hay que situarse en la celda en la cual se iniciará ésta y después mover el cursor con las siguientes teclas, las cuales a la vez que lo mueven también dibujan:

TECLA	FUNCION
I	Dibuja hacia arriba.
M	Dibuja hacia abajo.
K	Dibuja a la derecha.
J	Dibuja a la izquierda.

Al dibujar las barreras hay que tomar en cuenta que el cursor también se ve dibujado en la pantalla y se podría confundir con el extremo de una barrera en construcción.

Para dibujar un trompo, se indica con las siguientes teclas, las que, a la vez que construyen, mueven el cursor hacia la posición deseada:

TECLA	FUNCION
TI	Dibuja trompo y mueve el cursor hacia arriba.
TM	Dibuja trompo y mueve el cursor hacia abajo.
TK	Dibuja trompo y mueve el cursor a la derecha.
TJ	Dibuja trompo y mueve el cursor a la izquierda.

Para dibujar a KAREL, se tiene que colocar el cursor en la celda donde se desea construirlo; a continuación, con las teclas 1, 2, 3 ó 4, marcamos la dirección a la cual KAREL estará viendo, en ese momento aparece en el extremo superior derecho un símbolo que indica la dirección de KAREL:

TECLA	FUNCION
1	Mira hacia el NORTE.
2	Mira hacia el SUR.
3	Mira hacia el OESTE.
4	Mira hacia el ESTE.

La tecla "F", indica el fin de la construcción del entorno de KAREL.

El orden en el cual se pueden construir barreras y trompos del entorno de KAREL no importa, pero de cualquier forma es más conveniente construir primero las barreras, ya que de esta manera se podrá ubicar a los trompos y a KAREL en la posición más adecuada.

Las características del entorno de KAREL son:

i) Para las barreras.

En lo referente a éstas, KAREL no puede pasar a través de ellas, las podrá detectar en cualquiera de las cuatro direcciones: enfrente, atrás, a la derecha ó a la izquierda, pero esta capacidad puede usarla solamente estando a un cuadro de distancia de ellas, las palabras "atrás", "derecha", "izquierda" ó "enfrente", dependen de la dirección a la cual está apuntando KAREL en ese momento. Por ejemplo si KAREL está apuntando hacia el oeste, su derecha será el norte, su izquierda el sur, su frente será el este y atrás será el este. En la pantalla de la microcomputadora, el norte es el extremo superior, el sur el extremo inferior, el este el lado derecho y el oeste el lado izquierdo.

ii) Para los Trompos.

En cuanto a éstos, KAREL puede pasar sobre ellos, los detecta mediante un zumbido que emiten, este zumbido lo puede oír solamente cuando se encuentra "parado en él", esto es, en la misma celda.

iii) Para Karel.

De entre las actividades que éste ejecuta se señalan las siguientes:

Puede levantar trompos y guardarlos en una bolsa, que impide salir el zumbido que emiten.

También es posible que KAREL deposite los trompos en la celda que se le indique, siempre y cuando en esa celda no se encuentre construida una barrera u otro trompo.

Ejecuta tareas básicas, tales como avanzar una celda, dar vuelta a la derecha ó a la izquierda, recoger, dejar ó pedir trompos.

Ejemplo de una construcción del entorno de KAREL.

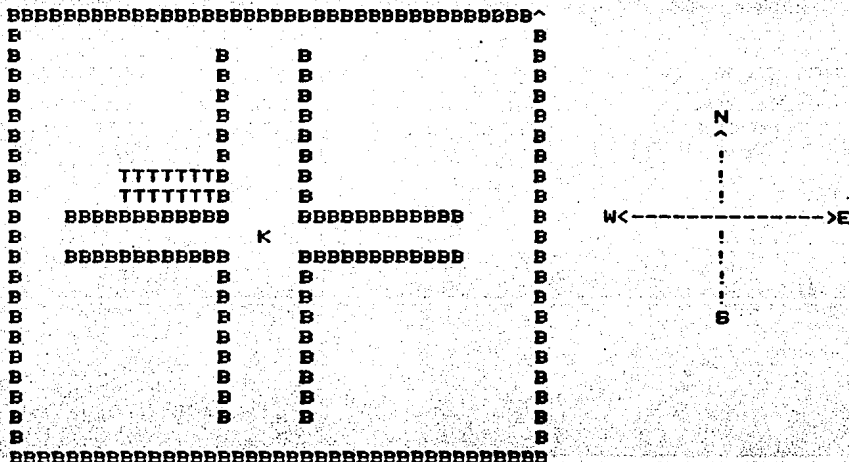


Figura 2

En esta figura observamos a KAREL en el centro de cuatro barreras con forma de escuadra, dispuestas estas una enfrente de la otra, la primera escuadra tiene en su interior dos hileras de tropezos, cada una formada por siete de ellos. En el extremo superior derecho observamos que KAREL está apuntando hacia la dirección norte.

II Karel como instrumento educativo.

La educación ha tenido siempre un papel muy importante en los cambios de las estructuras sociales y los hábitos de conducta, es por ello que la evolución de ésta nos afecta de una manera directa.

A través del tiempo se han ido agregando innovaciones en los métodos de enseñanza; una gran aportación fue la aparición del libro, con lo que fué posible transmitir el conocimiento a una gran cantidad de personas reduciendo el tiempo de aprendizaje, que antes solamente se hacía con la elaboración de trabajos manuales y conocimiento verbal.

Más adelante hubo cambios radicales en la educación con el advenimiento del cine y la televisión, dos medios que originalmente, hicieron concebir la idea de utilizarlos como instrumentos educativos para grandes núcleos de población; vehículos que harían llegar la cultura y conocimientos de manera rápida a amplios sectores; empero, desarrollo comercial de estos medios impidió la concreción de dicha idea. Es así que, aún cuando esta se hubiera logrado, el libro mostraría la ventaja que posee sobre el cine y la televisión: la libertad de imaginar que otorga al lector.

En la incorporación de nuevas técnicas e ideas para el sistema educativo, nos encontramos con la llegada de las computadoras, que tienen las características necesarias para revolucionar a la educación. Desde luego, para que esto suceda, es necesario realizar grandes esfuerzos y experimentar durante muchos años. Las computadoras tienen la capacidad de crear escenarios que despierten el interés y la imaginación en los niños y jóvenes, ya sea transportándolos a una nave espacial ó a un mundo imaginario como es el mundo de Karel el robot.

En cualquier tipo de escenario el estudiante queda involucrado como el personaje central. Simulando las acciones que él tomaría en la realidad y transmitiéndolas al computador, asumiendo la responsabilidad, por ejemplo, en el caso del módulo lunar, éste se estrellaría si el alumno no da correctamente las instrucciones, ó en el caso de Karel el robot, la misión fallaría.

Es por medio de esta simulación que el estudiante **aprende**, ya que él se ve en la necesidad real que tendría un profesionalista cuando se le plantean ciertos problemas.

Los programas educativos que involucran robots, tienen un papel muy importante en la educación, siendo la principal el hecho conocido como "el maestro aprende más que el alumno", esto se da debido a que si el maestro quiere enseñar al alumno, aquel debe dominar el tema que va a exponer. El maestro no puede dejar dudas cuando el alumno pregunta, porque la función es, precisamente, resolver la problemática planteada por el alumno, por ello, tiene la obligación de conocer a fondo el tema a enseñar.

En el caso de los robots, esto se incrementa aún más, ya que la persona que enseñe al robot hará las veces de maestro, pero el alumno, en este caso el robot, no es un alumno común y corriente, es un alumno más bien obediente, que realiza las tareas tal y como se le indican, sin tener un poder de razonamiento propio; por lo tanto, el maestro deberá no sólo dominar el tema, además saber explicarlo paso por paso sin ambigüedad de ningún tipo, porque, si la explicación es incorrecta, él mismo se dará cuenta de los errores que cometió y tratará de remediarlos.

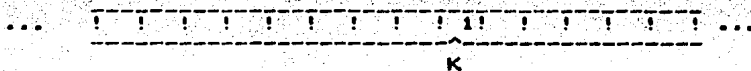
Los robots son entes que carecen de iniciativa propia, pero difieren mucho de las máquinas tradicionales.

Pongamos como ejemplo a una planta envasadora de leche; en ella, hay varias etapas para llegar a un producto final:

- Un proceso abre los envases, que vienen en forma plana.
- Una vez abiertos pasan por un proceso que los acomoda en cajas especiales.
- Los envases pasan por una máquina que los llena con una determinada cantidad de leche.
- Por último, pasan por otra máquina que los sella.

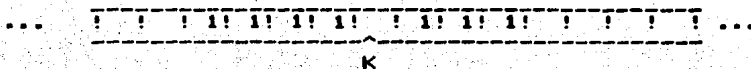
Todos estos pasos son realizados por máquinas que son realmente absurdas, ningún proceso de los mencionados anteriormente es verificado, es decir, si al proceso de llenado de líquido le llega una caja rota, la máquina de cualquier manera intentará llenarlo con leche, ó si el proceso que se encarga de sellar las cajas, no le llega alguna, la máquina de cualquier forma intentará unir la caja, corriendo el riesgo de desajustarse al chocar las dos placas metal contra metal.

Si Karel se mueve hacia la derecha hasta que observe un trompo, nuestra cinta se vería de la siguiente forma:



Ahora supongamos que el problema es un poco más complicado que la anterior.

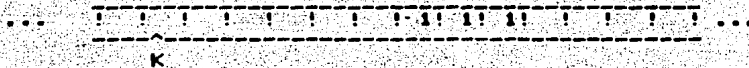
Supongamos que se quiere unir dos conjuntos de trompos. El esquema se vería de la siguiente forma:



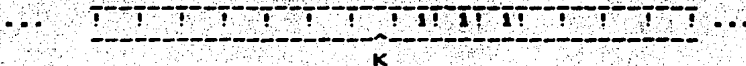
La tarea consistiría en colocar a la derecha de Karel los trompos de los dos conjuntos, sin dejar espacios entre ellos.

El proceso sería el siguiente:

- Mover a Karel hacia la izquierda guardando los trompos que encuentre a su paso hasta que llegue a una celda vacía.



- Mover a Karel a la derecha hasta que se encuentre un trompo.



III EL LENGUAJE DE KAREL.

El Lenguaje de KAREL es el conjunto de instrucciones que el alumno podrá dar al robot, para que éste realice sus tareas.

Existen tres comandos maestros:

- A.- Comando EJECUTA.
- B.- Comando ACEPTA
- C.- Comando MODIFICA.

A.- COMANDO EJECUTA (EJ) .- Con este comando se le indica a KAREL que realice una tarea, la cual puede ser una instrucción o una serie de ellas, que deberán ir entre paréntesis.

La construcción de la instrucción se hace de la siguiente forma:

EJ (<instrucción>/<sucesión de instrucciones>).

Por ejemplo:

EJ(AV).

Con este comando le indicamos a KAREL que avance una celda, lo que realizará considerando la dirección a la cual está apuntando en ese momento.

La instrucción:

EJ(AV AV AV).

Le está indicando a KAREL que avance tres celdas.

Si enfrente de KAREL se encuentra una barrera y se le da la instrucción de que avance, KAREL chocará ya que como se explicó anteriormente, este no puede atravesar barreras.

Las instrucciones que KAREL puede ejecutar las dividimos en tres grupos:

- A.1) Instrucciones simples.
- A.2) Palabras condicionales.
- A.3) Instrucciones de control.

A.1) INSTRUCCIONES SIMPLES.

Estas instrucciones cambian la posición de KAREL dentro de la malla, también a través de ellas podemos modificar la cantidad de trompos y la posición de cada uno de ellos.

Se cuenta con seis instrucciones básicas:

A.1.1) Avanza (AV) .- Con esta instrucción, KAREL se mueve una celda hacia la dirección a la cual esté apuntando; si en la celda hacia la cual se va a colocar existe una barrera, KAREL chocará y fracasará en su tarea; si en esa celda se encuentra un trompo, únicamente se colocará encima de él y en ese momento puede accionar su detector de zumbidos.

A.1.2) Vuelta derecha (VD) .- KAREL gira hacia la derecha 90 grados, por ejemplo, si KAREL apunta hacia el NORTE, al dar esta instrucción estará apuntando hacia el ESTE.

A.1.3) Vuelta izquierda (VI) .- KAREL gira hacia la izquierda 90 grados, por ejemplo si KAREL apunta hacia el NORTE, después de realizar esta instrucción estará apuntando hacia el OESTE.

A.1.4) Recoge trompo (RT) .- Sólomente si KAREL se encuentra encima de un trompo lo podrá recoger y depositar en una bolsa que tiene a prueba de zumbidos, de otra forma si el trompo no está debajo de KAREL y le damos esta instrucción, el programa marcará error.

A.1.5) Deja trompo (DT) .- KAREL tomará un trompo de su bolsa y lo depositará en la celda en la que se encuentre en ese momento, si su bolsa está vacía marcará error.

A.1.6) Pide trompo (PT) .- Aparecerá en la bolsa de KAREL un trompo más. Esta instrucción le sirve cuando queremos que realice tareas más complicadas, tales como la multiplicación de dos conjuntos, lo cual se verá en los ejemplos del inciso III-D.

A.2) PALABRAS CONDICIONALES.

Con estas palabras Karel puede tomar decisiones, que dependen del valor de la condicional en un momento dado; por ejemplo, la realización de una tarea puede estar sujeta a la condición de encontrarse encima de un trospo, enfrente de una barrera o verificando la existencia de trospos. También cuenta KAREL con una brújula, que le indica hacia qué dirección apunta ó mira (la parte superior de la pantalla es el NORTE, la parte inferior es el SUR).

Las Condicionales son:

- A.2.1) AE: Algo enfrente?
- A.2.2) NE: Nada enfrente?
- A.2.3) AI: Algo a la izquierda?
- A.2.4) NI: Nada a la izquierda?
- A.2.5) AD: Algo a la derecha?
- A.2.6) ND: Nada a la derecha?
- A.2.7) AS: Algo suena?
- A.2.8) NS: Nada suena?
- A.2.9) DN: Dirección NORTE?
- A.2.10) DS: Dirección SUR?
- A.2.11) DE: Dirección ESTE?
- A.2.12) DO: Dirección OESTE?
- A.2.13) NT: Ningún trospo?
- A.2.14) AT: Algún trospo?

Al verificar cualquiera de estas condicionales, el valor emitido es falso o verdadero (0 y 1 respectivamente).

A.3) INSTRUCCIONES DE CONTROL.

Existen tres instrucciones que permiten anidamiento, una de decisión y dos de repetición.

A.3.1) Instrucción "SI" (Condicional).

Con esta instrucción KAREL verificará la condicional en cuestión; si esta condicional se cumple ejecutará el primer grupo de instrucciones, de otra manera (DM), i.e. si la condicional no se cumple, KAREL ejecutará el segundo grupo de instrucciones.

La construcción de esta instrucción es:

```
SI <condicional> (< instrucciones DM (< instrucciones  
primer-grpo>)                segundo-grpo>).
```

La segunda parte de esta instrucción es opcional, i.e. la realización del segundo grupo de instrucciones si la condicional no se cumple.

Ejemplos:

```
SI AE (VI AV) DM (VD AV).
```

Con esta instrucción se le indica a KAREL que si tiene algo enfrente (una barrera), entonces de vuelta a la izquierda y avance una celda, en caso contrario, KAREL dará vuelta a la derecha y avanzará una celda.

A.3.2) Instrucción "repite mientras" (RM) .- Con esta instrucción KAREL ejecutará varias veces una serie de instrucciones mientras se cumpla la condición que se le indica.

La construcción de esta instrucción es:

```
RM <condicional> (< serie de instr. a  
realizar si se  
cumple la condición>).
```

La instrucción "repite mientras", primero verifica la condición y después realizará las instrucciones. Una vez realizadas estas, se regresará a verificar la condicional para ejecutar en un segundo ciclo las instrucciones. El proceso continuará hasta que deje de cumplirse la condicional.

EJEMPLO:

Si deseamos que KAREL se mueva hasta la barrera más cercana frente a él, esto puede lograrse mediante la siguiente instrucción:

EJ (RM NE (AV)).

Lo cual significa que repita la instrucción "avanza" mientras no encuentra nada enfrente.

No hay posibilidad de que KAREL choque, ya que la instrucción verifica primero que no haya nada enfrente para después avanzar.

EJEMPLO:

También es posible anidar una instrucción "SI" dentro de una instrucción "RM".

Si queremos recoger todos los trompos que se encuentran en una misma línea, daríamos la siguiente instrucción:

EJ (RM NE (SI AS (RT AV) DM (AV))).

Esta instrucción repetirá la instrucción (SI AS(RT AV) DM (AV)) mientras KAREL no tenga nada enfrente. Con la instrucción "SI", KAREL recogerá un trompo solo si está parado encima de alguno y siempre avanzará una celda.

```
BBBBBBBBBBBBBBBBBBBBBB>
B                               B
B                               B
B                               B
B                               B
B                               B
B                               B
B                               B
B K           T T T           B
B                               B
B                               B
BBBBBBBBBBBBBBBBBBBBBB
```

Antes de ejecutar la instrucción
KAREL apunta a la dirección ESTE

```

BBBBBBBBBBBBBBBBBBBB>
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
B                      B
BBBBBBBBBBBBBBBBBBBB>

```

Después de ejecutar la instrucción la bolsa de Karel tiene tres trompos, y Karel se para en el límite "ESTE" de su mundo, que fue en el momento en el que se encontró algo enfrente.

A.3.3) Instrucción "repite" (RE) .- Esta instrucción es similar a la anterior, pero la diferencia consiste en que en ésta, primero se ejecutarán las instrucciones que se indican y después se verifica la condicional, mientras (MT) esta se cumpla, se volverán a ejecutar la sucesión de instrucciones y así continuará el ciclo hasta que la condicional deje de cumplirse.

La construcción de esta instrucción es:

RE (<instrucciones>) MT <condición>.

Ejemplo:

RE (AV) MT NE.

La instrucción repite la acción de avanzar una celda, mientras (MT) se cumpla la condicional "nada enfrente".

La ejecución de esta sucesión de instrucciones a la postre produce un resultado erróneo, ya que Karel avanza antes de preguntar si puede hacerlo o no, por lo tanto, hay que tener cuidado al utilizar esta instrucción.

B.- COMANDO ACEPTA .- (AC) .- Con este comando se pueden definir tareas a Karel, las memorizará con el nombre que se le indique y podrá ejecutarlas posteriormente si así lo deseamos, únicamente refiriéndonos al nombre (identificador) de la tarea definida. La construcción de la instrucción se hace de la siguiente forma:

```
AC <nombre de la tarea a definir> = (<instrucción>/<sucesión de instrucciones>).
```

Ejemplo B.1:

```
AC T1 = (AV AV AV VI AV AV AV).
```

Con este comando quedan definidas con el nombre de T1 una serie de nuevas instrucciones que son: avanza tres celdas, da vuelta a la izquierda y después avanza otras tres celdas.

Ejemplo B.2:

Haciendo uso del comando AC, podemos definir nuevas instrucciones y utilizarlas en la definición y ejecución de otras:

```
AC T1 = (SI ND (VD)).  
AC T2 = (SI AE (VI) DM (AV T1)).  
AC T3 = (RE (T2) MT NS).  
EJ (T3).
```

En este conjunto de instrucciones estamos definiendo tres nuevas instrucciones, T1, T2 y T3. La instrucción T1 la usamos para definir la T2 y esta a su vez para definir la T3.

Al ejecutar T3, se ejecutará el siguiente grupo de instrucciones una vez substituidas T1 y T2:

```
(RE (SI AE (VI) DM (AV (SI ND (VD)))) MT NS).
```

C .- COMANDO MODIFICA .- (MD) .- Con este comando se puede modificar una tarea definida anteriormente; al darle la nueva serie de instrucciones olvidará las anteriores. La construcción del comando es similar a la construcción del comando aceptar:

MD <nombre de la tarea a modificar> = (<instrucción>/<sucesión de instrucciones>).

Ejemplo C.1:

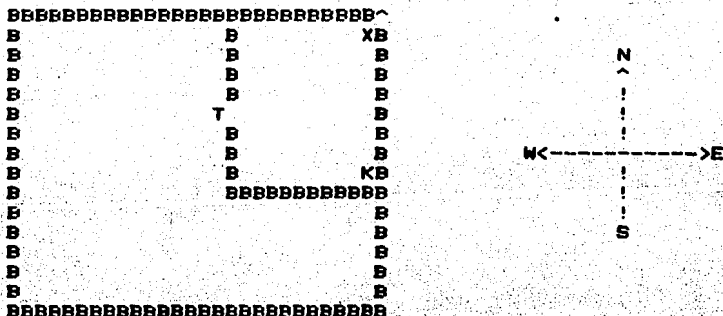
MD T1 = (AV VD AV AV VI AV).

Ahora la tarea T1 tiene definida una nueva serie de instrucciones: un avanza, vuelta a la derecha, dos avanza, vuelta a la izquierda y un avanza.

D) EJEMPLOS.

Ejemplo D.1:

En la siguiente gráfica podemos ver a Karel en una posición dentro de un rectángulo, hecho de barreras y apuntando hacia el norte. Con el siguiente programa le indicamos que se coloque en la primera celda que está fuera del rectángulo. Si existe un trompo lo colocará en la posición marcada con "X" y regresará a su posición original, en caso contrario, únicamente regresará a su posición original. En cualquiera de los dos casos Karel apuntará hacia el sur.



- K----- Karel
- T----- Trompo
- X----- Posición en la cual depositará Karel el trompo.
- B----- Barrera

Programa:

```

AC T0 = (VI VI).
AC T1 = (AV AV AV).
AC T2 = (BI AB (RT TO T1 T1 T1 AV AV VI T1 AV DT TO T1 T1 AV)
        DM (TO T1 T1 T1 AV AV VD T1)).
EJ (T1 VI T1 T1 T1 AV AV T2).
  
```


El tercer paso consiste únicamente en que Karel se dirija al compartimento del cual proviene la hilera mayor, indicando con esto la solución al problema o bien no realizar ninguna acción si los conjuntos son iguales.

Programa para la comparación de dos conjuntos:

RESTRICCIONES:

- i) Los compartimentos no pueden contener más de 16 trompos.
- ii) Inicialmente Karel se encuentra en la posición (11,17) y apuntando al norte; en ese momento su bolsa de trompos está vacía.

PROGRAMA:

Instrucción para avanzar en una línea:

AC T0 = (RM NE (AV)).

Se sitúa en la puerta del primer compartimento.

AC T1 = (VI TO VD AV AV).

Instrucción para recoger trompos en una hilera.

AC T2 = (RM NE (SI AS(RT AV) DM (AV)) SI AS(RT)).

Instrucción para recoger los trompos del compartimento izquierdo (conjunto "A").

AC T3 = (T2 RM ND (VD AV VD TO VD VD T2)).

Instrucción para salir del compartimento "A" y situarse en la posición (1,1) con dirección ESTE.

AC T4 = (VI TO VI TO VI).

Instrucción para depositar los trompos del conjunto "A" en una hilera.

AC T5 = (RM AT (DT AV)).

Instrucción para situarse en la puerta del conjunto "A".

AC T6 = (TO VI RM NI (AV) AV).

Instrucción para recoger trompos del compartimento "B".

AC T7 = (T2 RM NI (VI AV VI TO VI VI T2)).

Instrucción para situarse en la posición (1,2) con dirección ESTE.

AC T8 = (VD TO VD TO VD VD AV VI TO VD VD).

Instrucción para situarse y comparar.

AC T9 = (VD AV VD AV VD VD).

Primera comparación, para saber si el conjunto "B" es mayor que el conjunto "A".

Se dirige al compartimento "B".

AC T12 = (TO VI TO).

AC T10 = (SI NS (T12) DM (T11)).

Segunda comparación para saber si el conjunto "A" es mayor que el conjunto "B", de otra forma, no hay ninguna acción.

Se dirige al compartimento "A".

AC T13 = (VD VD TO VD TO).

AC T11 = (AV SI AS (T13)).

EJECUCION:

EJ (T1 T3 T4 T5 T6 T7 T8 T5 T9 T10 T11).

Ejemplo D.3a

Multiplicación de dos conjuntos de trompos.

La solución a este problema se plantea mediante las siguientes figuras:

```

BBBBBBBBBBBBBBBBBBBBB^
BTTT B      B TTTT
B TTT B      B T T B
B   B      B TT B
B BBBB      BBBB B
B
B Conj."A"  Conj."B" B
B
B
B
B
B      K      B
B
B
B
B
BBBBBBBBBBBBBBBBBBBBB
    
```

I) Inicio del problema.

```

BBBBBBBBBBBBBBBBBBBBB>
B   B      B TTTT
B   B      B T T B
B   B      B T B
B BBBB      BBBB B
B
B Conj."A"  Conj."B" B
B
B
B
B
B
B
B
BTTTTTTK      B
BBBBBBBBBBBBBBBBBBBBB
    
```

II) El conjunto A se copia en la primera hilera y se quita un troppo del conjunto B.

```

BBBBBBBBBBBBBBBBBBBB>
B   B           B   B
B   B           B   B
B   B           B   B
B BBBB         BBBB B
B Conj. "A"   Conj. "B" B
BTTTTTT      K   B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BTTTTTT      B
BBBBBBBBBBBBBBBBBBBB

```

III) Se repite la misma hilera por cada trompo que exista en el conjunto "B", quedando finalmente de esta forma.

El primer paso consistió en que Karel recogiera los trompos del conjunto "A" y formara con ellos una hilera en el extremo inferior izquierdo, después se dirigió al compartimento "B" y quitó un trompo de este.

El segundo paso fue que Karel repitiera el siguiente ciclo:

- 1.- Ir al compartimento "B" y quitar un nuevo trompo.
- 2.- Localizar la esquina inferior izquierda.
- 3.- Construir una nueva hilera de trompos.
- 4.- Identificar la última hilera ya construida.
- 5.- Pedir trompos cada vez que le hiciera falta.





Cuando en el compartimento "B" ya no quedaran trompos, entonces, se detiene el ciclo y la tarea se da por terminada.

IV. DESCRIPCIÓN DE LA ESTRUCTURA DE KAREL.

A.- Registros asociados a la graficación del entorno de Karel.

La representación del entorno de Karel se hace mediante la graficación de los elementos que lo componen, cada uno de ellos tiene un símbolo distinto que lo distingue en la pantalla.

Como ya se dijo anteriormente, el entorno de Karel está representado por una malla, en donde a cada celda de esta malla le está asociada una pareja de coordenadas; a cada una de estas celdas se le asigna un valor que depende de su contenido, por ejemplo, para la Micro LNW-80 los valores asignados son:

REGISTRO	REPRESENTACION DE	VALOR NUMERICO PARA GRAFICAR	SIMBOLO EN LA PANTALLA
BA	Barreras	166	-----> 
TR	Tropos	140	-----> 
KR	Karel	191	-----> 
EV	Posición en Blanco	128	-----> 

Así, cuando se quiere formar una barrera horizontal de la posición (5,8) a la posición (5,12), se le asigna a las celdas (5,8), (5,9), (5,10), (5,11) y (5,12) el número 166, que es valor que se transforma en el símbolo de barreras en la pantalla.

Si queremos un tropo en la celda (10,10), le asignamos a esta pareja el valor 140 con el cual al graficar, aparecerá en la pantalla el símbolo que representa a los tropos. De igual forma puede graficarse a Karel.

Como puede observarse a cada una de las celdas de la malla le corresponde un valor numérico, también las celdas que se encuentran en blanco tienen asignadas el valor de 128. Es por medio del almacenamiento de todos estos valores que es posible determinar en qué momento Karel se encuentra encima de un tropo ó enfrente de una barrera.

En el caso de los tropos primero guardamos el valor de la celda en la cual se parará Karel, para determinar si (algo suena) (AS), preguntamos si el valor que guardamos es igual a 140, lo que nos indicaría que Karel se paró en un tropo. Si queremos saber que Karel está frente a una barrera, entonces preguntamos por el valor de la siguiente celda, según la dirección a la que apunta Karel, y si este valor es igual a 166, entonces podemos decir que Karel tiene enfrente una barrera.

Los registros asociados a la posición, dirección e instrumentos que posee Karel son:

REGISTRO

FUNCION

- | | |
|---------|---|
| KX y KY | Son las coordenadas de la posición de Karel en un momento dado. |
| KD | Es el registro en el cual se guarda la dirección a la que apunta Karel. |
| KN | Es el registro que está asociado a la bolsa de tropos que lleva Karel. Este registro se modifica con las instrucciones recoge tropo y deja tropo. |

B.- ESTRUCTURA Y REGISTROS ASOCIADOS AL INTERPRETE.

El intérprete de Karel es una rutina escrita en BASIC, que simula el comportamiento de Karel y se apoya en los siguientes registros:

B.1) DIRECTORIO CENTRAL .- El directorio central es una estructura en la que se almacenan los nombres de las instrucciones y de las palabras condicionales, así como también el tipo a la que pertenecen cada uno de ellos.

DIRECTORIO CENTRAL

I	NB(I)	TF(I)	
1	AV	1	Avanza
2	VD	1	Vuelta a la derecha
3	VI	1	Vuelta a la izquierda
4	RT	1	Recoje Trospo
5	DT	1	Deja Trospo
6	(4	
7)	5	
8	SI	3	Si
9	DM	6	De otro modo
10	RP	7	Repita
11	MT	8	Mientras
12	RM	9	Repite Mientras
13	PT	1	Pide trospo
14	AE	2	Algo enfrente
15	NE	2	Nada enfrente
16	AI	2	Algo a la izquierda
17	NI	2	Nada a la izquierda
18	AD	2	Algo a la derecha
19	ND	2	Nada a la derecha
20	AS	2	Algo suena
21	NS	2	Nada suena
22	DN	2	Dirección Norte
23	DE	2	Dirección Este
24	DS	2	Dirección Sur
25	DO	2	Dirección Oeste
26	AT	2	Algún Trospo
27	NT	2	Ningún Trospo
28	T1		Tarea T1
29	T2		Tarea T2
...	
36	T9		Tarea T9
37	EJ		Comando de Ejecución

En donde:

I.- Indica el código asociado a cada instrucción. Este índice nos va a servir para guardar la definición de una instrucción no básica en función de los códigos de las instrucciones que la forman.

NB(I).- Tabla de nombres de instrucciones básicas o instrucciones definidas.

TF(I).- Es la tabla de filtros, en ella se asocia a cada instrucción un valor, el cual indica el tipo de instrucción de que se trata. Por ejemplo, a las palabras condicionales se les asigna el valor dos. La tabla de filtros se hace con el objeto de identificar fácilmente las entradas al autómata de control, que se explicará más adelante.

B.2.- Registros asociados al ensamblador.

Para almacenar la definición de una instrucción generada mediante el comando (AC), se utiliza una rutina que ensambla la instrucción. Para realizar esto, localiza en el directorio central cada una de las instrucciones que componen el AC y coloca en el vector IX los códigos correspondientes a cada una de ellas, guardando en una tabla de apuntadores APX el número de celda en el cual se encuentra la primera instrucción que ensamblamos. Esta rutina se auxilia de los siguientes registros:

IXX .- Tabla de códigos de instrucciones. En este vector se van ensamblando los códigos asociados de cada una de las instrucciones que definen una nueva instrucción con el comando acepta.

APX .- Tabla de apuntadores .- Para cada instrucción nueva, no básica, se tiene un valor correspondiente en APX, que indica a partir de qué celda en el vector IXX comienza la definición de esa instrucción.

C) COMPILADOR

Un compilador es un programa que traduce un programa escrito en un lenguaje de programación de alto nivel, llamado lenguaje fuente, y produce como salida un programa en otro lenguaje de programación de nivel más bajo, llamado lenguaje objeto.

El compilador de Karel es un traductor muy simple escrito en el lenguaje BASIC para realizar dos funciones básicas:

- a) Verificar que las instrucciones estén construidas correctamente en cuanto al número de parentesis izquierdos y derechos (que deben coincidir) y en cuanto a que los nombres de las instrucciones básicas estén bien escritos.
- b) Traduce las instrucciones básicas en sus códigos de ejecución de acuerdo con el contenido de un directorio en el que se encuentran registrados comandos, instrucciones básicas y cada nueva instrucción definida por el usuario.

El compilador es entonces, una rutina mucho más simple que los compiladores utilizados usualmente en cuanto a que nuestro programa no realiza ningún tipo de análisis sintáctico (adicional al de los parentesis) de acuerdo a las reglas de una gramática; adicionalmente, el compilador tampoco cuenta con generadores semánticos completos ya que esta función es solo realizada con el copiado de los códigos en la tabla.

Es importante hacer notar que, tanto la verificación sintáctica cuanto los aspectos semánticos son realizados por el intérprete de Karel que se constituye así en la rutina central del sistema.

Las acciones específicas del compilador se describen a continuación:

- 1) La primera cuerda de caracteres deberá ser un comando, i.e. AC, EJ ó MD.
- 2) Verifica en el directorio la existencia de las instrucciones básicas, ó instrucciones definidas anteriormente.
- 3) Valida igual número de parentesis derechos e izquierdos.

Este Compilador, a la vez que valida las instrucciones, va llenando el directorio y el vector de códigos.

Si se trata de un ACEPTA (AC):

- a) Inserta el nuevo nombre en el directorio, esto es, en el vector NB\$(I).
- b) Va colocando en el vector IXX los códigos de las instrucciones que forman la nueva instrucción.
- c) Guarda en APX(I) un apuntador hacia la celda en el vector IXX, indicando exactamente a partir de que celda comienzan los códigos que definen a la instrucción.

Ejemplo:

Supongamos que definimos con el comando **accepta (AC)** las siguientes instrucciones:

AC T1 = (AV AV VD AV).

AC T2 = (AV VD AV).

La Tabla de Apuntadores (AP)%, quedaria de la siguiente forma:

I	NB%	AP%
1	AV	-
2	VD	-
...
27	NT	-
28	T1	1.....
29	T2	7.....

Vector de códigos
de instrucciones (IXX)

Num. celda

1 2 3 4 5 6 7 8 9 10 11

(Código)

6 1 1 2 1 7 6 1 2 1 7 ...

(Instrucciones)

(AV AV VD AV) (AV VD AV)

Si se trata de un EJECUTA (EJ):

- a) Capta en el vector `NR$(I)` los caracteres "EJ".
- b) A continuación va colocando en el vector `IX%` los códigos de las instrucciones que se van a ejecutar.

Si se trata de un MODIFICA (MO):

- a) Va colocando en el vector `IX%`, los códigos que forman la instrucción a modificar.
- b) Guarda en el apuntador de la instrucción `AP$(I)` el nuevo apuntador hacia la primera celda en el vector `IX%`, donde comienza la definición de la instrucción modificada.

D).- DEFINICION DE LA ESTRUCTURA GRAMATICAL.

Toda gramática la podemos definir con los siguientes elementos:

- 1).- El vocabulario, el cual se divide a su vez en vocabulario terminal y terminos sintácticos.
- 2).- Las reglas de producción ó reglas gramaticales.
- 3).- El tipo sintáctico fundamental (meta de la gramática).

El lenguaje de Karel puede definirse mediante las siguientes reglas de producción:

```
PROG <----- DEF-LISTA EJEC.  
DEF-LISTA <----- DEF  
DEF-LISTA <----- DEF-LISTA DEF  
EJEC <----- EJ (LISTAI).  
DEF <----- AC NOMBRE = (LISTAI).  
LISTAI <----- INST  
LISTAI <----- LISTAI INST  
LISTAI <----- DEF  
LISTAI <----- LISTAI DEF  
INST <----- AV/VD/VI/RT/DT/PT  
INST <----- IDENT  
INST <----- BI COND (LISTAI) DM (LISTAI).  
INST <----- RM COND (LISTAI).
```

INST <----- RP (LISTAI) MT COND.
INST <----- BI COND (LISTAI).
COND <----- AE/NE/AI/NI/AD/ND/AS/NS/AT/NT/DN/DE/DS/DO
NOMBRE <----- LETRA DIGITO
LETRA <----- A/B/C.../Y/Z
DIGITO <----- 0/1/2.../8/9

El vocabulario terminal está en letras más oscuras.

Los términos sintácticos son las palabras COND, INST, LISTAI, DEF, DEF-LISTA, EJEC, PROG.

Y el tipo sintáctico fundamental ó meta de la gramática es el programa (PROG).

Para poder determinar si un conjunto de instrucciones pertenecen al lenguaje o no, es necesario realizar un análisis lexicográfico (llamado "SCANNER") y un análisis sintáctico (llamado "PARSER").

En el análisis lexicográfico, se separan los caracteres de las instrucciones formando términos o vocablos y en el PARSER aplicamos las reglas de producción de la gramática del lenguaje, reduciendo las instrucciones hasta llegar a la meta.

E).- EL INTERPRETE DE KAREL.

Un intérprete de un lenguaje es un programa que a la vez que va reconociendo la sintaxis de las instrucciones, también las va ejecutando.

El intérprete de Karel realiza una serie de operaciones de reconocimiento sintáctico para asegurarse que las instrucciones que ejecuta están bien construidas antes de ejecutarlas.

En virtud de la complejidad de la gramática del lenguaje descrita en la sección anterior, las operaciones de reconocimiento requieren que el intérprete esté formado por un autómata con "push-downs".

El autómata funciona de manera muy similar a los autómatas secuenciales y su comportamiento puede ser descrito en términos de estos con muy leves modificaciones, excepto que en algunos momentos hace uso de los "push-downs" para tomar algunas decisiones relacionadas con posibles problemas de anidamiento.

Así por ejemplo en la estructura:

```
SI cond (listai) DM (listai2).
```

es posible encontrar una segunda estructura SI cond (listai) DM (listai) dentro de listai1, que no podría ser resuelta sin un "push-down".

Principiaremos por describir el autómata secuencial para posteriormente discutir los "push-downs" y sus conexiones al autómata.

Un autómata secuencial es la representación gráfica de una máquina capaz de reconocer la sintaxis de un lenguaje, almacenando a través de cada uno de sus estados la sucesión de entradas a ella.

En el proceso de control del intérprete de Karel se distinguen los tres ciclos de operación siguientes:

- E.1) Ciclo "SI".
- E.2) Ciclo "REPITE MIENTRAS".
- E.3) Ciclo "REPITE".

Cada uno de estos ciclos tiene asociado un determinado número de estados, dependiendo de la complejidad de cada instrucción. A cada uno de estos estados les podemos asociar una o varias ternas: (E, P, Q), en donde:

- E .- Es la entrada.
- P .- Es el proceso a ejecutar.
- Q .- Es el estado siguiente al que se pasa dentro del autómata.

Nota: El símbolo "*" indica cualquier otra entrada no mencionada anteriormente dentro del estado. La palabra "cond" significa la entrada de cualquier palabra condicional.

E.1) El ciclo "SI".

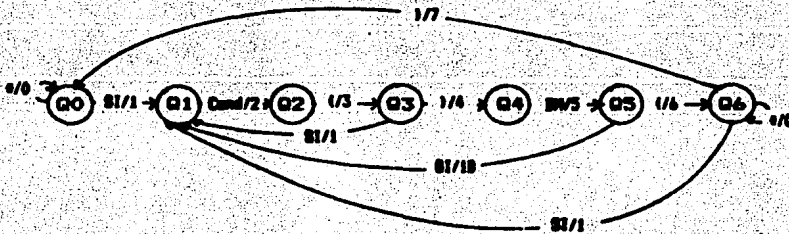
Cuenta con siete estados:

Q0 (SI,P1,Q1), (*,P0,Q0)
Q1 (cond,P2,Q2)
Q2 ("(",P3,Q3)
Q3 ("(",P4,Q4), (SI,P1,Q1), (*,P0,Q3)
Q4 (DM,P5,Q5), (SI,P1B,Q1), (*,P0,Q0)
Q5 ("(",P6,Q6)
Q6 (SI,P1,Q1), ("",P7,Q0), (*,P0,Q6)

En este ciclo puede observarse que el proceso uno contempla el anidamiento de una instrucción "SI", dentro de otra instrucción "SI", lo cual no ocurre con el proceso 1B, el cual da por terminado el primer "SI".

La terminación del ciclo puede hacerse tanto en el estado Q4 cuanto en el estado Q6; este hecho dependerá de que se use o no se use la segunda parte de la instrucción.

Figura III.1



Ciclo "SI"

E.2) El ciclo "REPITE MIENTRAS" (RM).

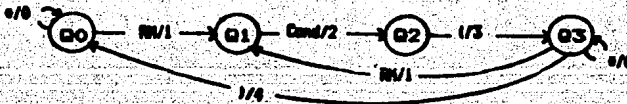
Cuenta con cuatro estados:

Q0 (RM,P1,Q1), (*,P0,Q0)
Q1 (Cond,P2,Q2)
Q2 ("(",P3,Q3)
Q3 (")",P4,Q0), (RM,P1,Q1), (*,P0,Q3)

El proceso uno contempla en el estado Q3 el anidamiento de un RM dentro de otro RM. En este mismo estado pueden llegar, como entradas, cualquier tipo de instrucciones simples.

El fin de este ciclo está dado por el paréntesis izquierdo encontrado en el estado Q3.

Figura III.2



Ciclo "RM"

E.2) El ciclo "REPITE" (RP).

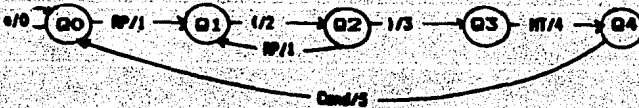
Cuenta con cinco estados:

Q0 (RP,P1,Q1), (*,P0,Q0)
Q1 ("(",F2,Q2)
Q2 (")",P3,Q3), (RP,P1,Q1), (*,P0,Q2)
Q3 (MT,P4,Q4)
Q4 (Cond,P5,Q0)

Como en los ciclos anteriores, en este también es posible anidar una instrucción "RP" dentro de otra instrucción "RP".

En este ciclo la condicional es la que marca el fin.

Figura III.3



Ciclo "RP"

La composición de los tres ciclos de control en un solo modelo de autómata, permite anidar un ciclo de cualquiera de los tres tipos mencionados dentro de otros en el nivel que se desee.

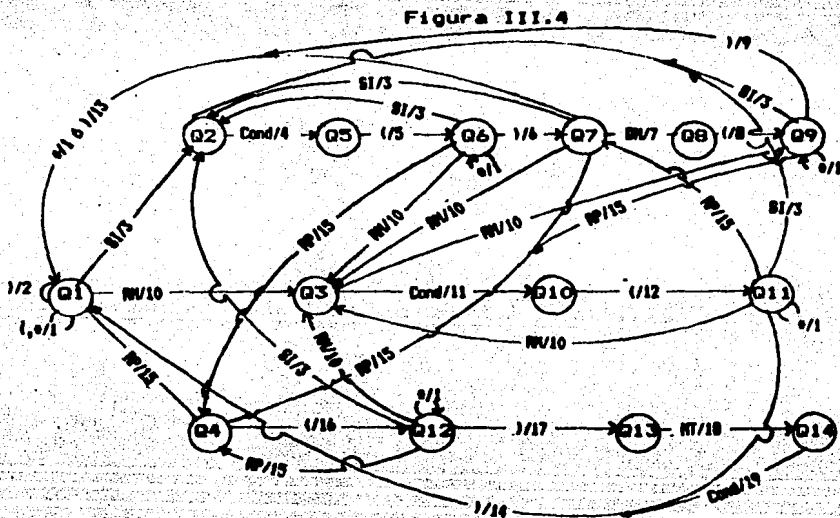
Para auxiliarnos en la selección de las entradas al autómata general establecemos un filtro cuyas salidas se especifican en la siguiente tabla:

TABLA DE FILTRADO PARA EL AUTOMATA

Entrada	Salida	Código Salida CS
SI	SI	1
AE NE AD ND AI NI AT DD AS NS DS DE DN	Cond	2
((3
))	4
RP	RP	5
MT	MI	6
RM	RM	7
DM (De otro modo)	DM	8
AV VI VD DT PT RT ó cualquier otro Término	*	9

Estas salidas nos permiten recorrer el autómata, reconociendo cada uno de los términos de entrada, al mismo tiempo que los seleccionamos.

E.4) ESTRUCTURA GENERAL DEL AUTOMATA.



El autómata de control de Karel se apoya en cuatro "stacks" "push-down", un "stack" es una lista en la cual se va almacenando información, en donde la entrada de un nuevo elemento al "stack" se hace al principio de la lista y la salida de un elemento se hace tomando el primer elemento de la lista, i.e. el último elemento que entra es el primero en salir.

Para anidar los ciclos de control el intérprete cuenta con los siguientes "stacks" y apuntadores:

- BP Es el stack de programas, en su celda de salida tiene la dirección de la siguiente instrucción a ejecutar.

- KP Apuntador del "stack" SP.
- SQ Es el "stack" de estados. En él se registra el "estado" del autómata en cada nivel de ejecución, de acuerdo a los ciclos de control explicados anteriormente.
- KQ Apuntador del "stack" SQ.
- SC Es el "stack" para la instrucción condicional de autómata en cada nivel de ejecución. Cuando el valor de la celda superior del "stack" es uno, esto significa que el grupo de instrucciones que continúa serán ejecutadas. Cuando el "stack" es cero las ignorará hasta terminar su nivel de ejecución.
- KC Apuntador del "stack" SC.
- SW Es el stack de apuntadores al vector IX%, para cada nivel de ejecución.
- KW Apuntador del "stack" SW.

D.5) TABLA DE TRANSICIONES Y SALIDAS ASOCIADAS AL AUTOMATA DE CONTROL.

Por medio de esta tabla se ejecutan cada uno de los procesos asociados al autómata de control.

La primera columna nos indica el estado en el cual nos encontramos y el primer renglón nos indica la entrada al autómata. La intersección de esta columna con el renglón nos da una pareja de números (q,p), en donde "q" es el estado siguiente dentro del autómata y "p" es el proceso que se realizará en ese momento.

Q	RE	*	Cond	SI	()	DM	RP	MT	RM	
1		1,1	0,0	2,3	1,1	1,2	0,0	4,15	0,0	3,10
2		0,0	5,4	0,0	0,0	0,0	0,0	0,0	0,0	0,0
3		0,0	10,11	0,0	0,0	0,0	0,0	0,0	0,0	0,0
4		0,0	0,0	0,0	12,16	0,0	0,0	0,0	0,0	0,0
5		0,0	0,0	0,0	6,5	0,0	0,0	0,0	0,0	0,0
6		6,1	0,0	2,3	0,0	7,6	0,0	4,15	0,0	3,10
7		1,1	0,0	2,3	0,0	1,13	8,7	4,15	0,0	3,10
8		0,0	0,0	0,0	9,8	0,0	0,0	0,0	0,0	0,0
9		9,1	0,0	2,3	0,0	1,9	0,0	4,15	0,0	3,10
10		0,0	0,0	0,0	11,12	0,0	0,0	0,0	0,0	0,0
11		11,1	0,0	2,3	0,0	1,14	0,0	4,15	0,0	3,10
12		12,1	0,0	2,3	0,0	13,17	0,0	4,15	0,0	3,10
13		0,0	0,0	0,0	0,0	0,0	0,0	0,0	14,18	0,0
14		0,0	1,19	0,0	0,0	0,0	0,0	0,0	0,0	0,0

D.6) PROCESOS ASOCIADOS AL AUTOMATA DE CONTROL.

Para la ejecución de los procesos del autómata es necesario auxiliarnos de los siguientes REGISTROS:

- RI Contiene el código de filtro, dependiendo del valor de entrada al autómata.
- NP Contiene en número de proceso que se está ejecutando de acuerdo al recorrido del autómata.
- PC Contiene el código de la siguiente instrucción a ejecutar.
- NQ Contiene el número de estado en el cual se encuentra el autómata.
- TE Indica la existencia de un trompo. Cuando Karel se encuentra encima de él, Si $TE = 1$ entonces existe un trompo, si $TE = 0$ entonces no hay trompo. Este registro es necesario, debido a que no es posible almacenar dos valores distintos en una misma celda, en este caso Karel y el trompo.

DESCRIPCION DE LOS PROCESOS.

PROCESO P1.

Estado inicial: Q1
Entradas : AV ó VD ó VI ó RT ó DT ó PT ó "(" ó cualquier nombre nuevo de instrucción.
Estado final : Q1

Descripción:

i) Si el "stack" de condición (SC) es igual a 2 ó 3, esto indica que hasta ese momento se ha detectado la primera parte de una instrucción "SI", si esto sucede, se decrementa el "stack" de condición y el de estados.

ii) Si la entrada es una instrucción definida por el usuario es necesario asignar al registro PC el valor del apuntador que nos indica a partir de dónde se encuentra la definición de la instrucción. Se incrementa el "stack" de la siguiente instrucción a ejecutar.

iii) Si la entrada es un avanza (AV), entonces, dependiendo de la dirección que tenga Karel, se modifican los registros XO, YO.

iv) Si la entrada es vuelta a la derecha (VD) ó vuelta a la izquierda (VI), se modifica el registro KD, incrementándolo o decrementándolo, según la instrucción, cuidando que el resultado sea entre 1 y 4, para lo cual se utiliza la función módulo.

v) Si la entrada es recoge trompo (RT), primero se verifica que el registro TE, sea igual a 1. Se le asigna a TE el valor cero. Se incrementa el valor del registro KN, que es el que simula la bolsa en donde Karel guarda sus trompos.

vi) Si la entrada es deja trompo (DT), primero se verifica que el registro KN sea mayor que cero. Se le asigna al registro TE el valor de 1.

vii) Si la entrada es pide trompo (PT), se incrementa el registro KN.

viii) Si la entrada es "(", no hay ninguna acción.

PROCESO P2.

Estado inicial: Q1
Entradas : ")"
Estado final : Q1

Descripción:

i) Decrementa el "stack" de siguiente instrucción a ejecutar, y el "stack" de estados, asignándole al registro NO el nuevo estado.

PROCESO P3.

Estado inicial: Q1
Entradas : "SI"
Estado final : Q2

Descripción:

i) Si el "stack" de condición es igual a 2 ó 3, se decrementa el "stack" de condición y el "stack" de estados, asignando al registro NO el nuevo estado, indicando con ello el término de una instrucción "SI".

ii) Se decrementa el "stack" de estados y el "stack" de condición. Al primero se le asigna el estado inicial Q1 y al segundo el valor de 1.

PROCESO P4.

Estado inicial: Q2
Entradas : Cualquier palabra condicional.
Estado final : Q5

Descripción:

i) Si se trata de las condicionales, algo enfrente? (AE) ó nada enfrente? (NE).

- Asignamos al registro KA el valor de la siguiente celda, de acuerdo con la dirección a la cual apunta Karel, comparamos con el valor del registro BA, el cual contiene el valor de una barrera.

Instrucción	Karel	Valor "stack" de condición
Algo enfrente? (AE)	KA=BA	SC(KC)=1
Algo enfrente? (AE)	KA#BA	SC(KC)=0
Nada enfrente? (AE)	KA=BA	SC(KC)=0
Nada enfrente? (AE)	KA#BA	SC(KC)=1

ii) Si se trata de las condicionales, algo a la izquierda? (AI) ó nada a la izquierda? (NI).

- Asignamos al registro KA el valor de la celda que se encuentra a la izquierda del lugar que ocupa Karel, tomando como su frente la dirección a la cual apunta.

- El valor del registro KA es comparado con el valor del registro BA (barrera).

Instrucción	Karel	Valor Stack de Condición
Algo a la izquierda?(AI)	KA=BA	SC(KC)=1
Algo a la izquierda?(AI)	KA#BA	SC(KC)=0
Nada a la izquierda?(NI)	KA=BA	SC(KC)=0
Nada a la izquierda?(NI)	KA#BA	SC(KC)=1

iii) Si se trata de las condicionales algo a la derecha? (AD) ó nada a la derecha? (ND).

- El proceso que se realiza es similar al anterior, sólo que en estos casos se asigna al registro KA el valor de la celda que se encuentra a la derecha de Karel.

iv) Si se trata de la condicional, algo suena? (AS).

- Le asigna al "stack" de condición el valor del registro TE, el cual tiene el valor de 1 si Karel está parado encima de un trompo, o el valor de cero en caso contrario.

v) Si se trata de la condicional, nada suena? (NS).

- Se le asigna al "stack" de condición el valor de 1-TE, esto es, el valor de cero cuando Karel se encuentra encima de un trompo y el valor de 1 en caso contrario.

vi) Si se trata de las condicionales dirección norte? (DN), dirección sur? (DS), dirección oeste? (DO) ó dirección este? (DE).

- Se pregunta por el valor del registro KD, que es el que contiene la dirección de Karel:

Para	Valor del "Stack" de condición		
DN?	SC(KC)=1	si	KD=1
	SC(KC)=0	si	KD#1
DE?	SC(KC)=1	si	KD=2
	SC(KC)=0	si	KD#2
DS?	SC(KC)=1	si	KD=3
	SC(KC)=0	si	KD#3
DO?	SC(KC)=1	si	KD=4
	SC(KC)=0	si	KD#4

vii) Si se trata de la condicional, algún trompo? (AT).

- Preguntamos por el registro KN, si este registro es igual a cero, entonces SC(KC)=0, en caso contrario el "stack" de condición toma el valor de 1.

viii) Si se trata de la condicional, ningún trompo? (NT).

- Preguntamos por el registro KN, si el valor es cero, entonces SC(KC)=1 en caso contrario el "stack" toma el valor de cero.

PROCESO P5.

Estado inicial: Q5
Entradas : "("
Estado final : Q6

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómata de control.

PROCESO P6.

Estado inicial: Q6
Entradas : ")"
Estado final : Q7

Descripción:

- Modifica el "stack" de condición con los valores 2 ó 3:

Si $SC(KC)=0$ entonces $SC(KC)=2$
Si $SC(KC)=1$ entonces $SC(KC)=3$

De esta forma, detectamos la terminación de la primera parte de una instrucción "SI", esto se hace con el objeto de poder manejar la parte opcional de esta misma instrucción, ó sea el "de otro modo" (DM).

PROCESO P7.

Estado inicial: Q7
Entradas : "DM"
Estado final : Q8

Descripción:

- Este proceso invierte los valores del "stack" de condicionales, con lo cual indica que se completó una instrucción "SI", incluyendo su parte opcional.

Si $SC(KC)=2$ entonces $SC(KC)=1$
Si $SC(KC)=3$ entonces $SC(KC)=0$

PROCESO P8.

Estado inicial: Q8
Entradas : "("
Estado final : Q9

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómatá de control.

PROCESO P9.

Estado inicial: Q9
Entradas : ")"
Estado final : Q1

Descripción:

- Decrementa el "stack" de condición y el "stack" de estados, asignando al registro NQ el nuevo estado, indicando con esto el término de una instrucción "SI".

PROCESO P10.

Estado inicial: Q1 ó Q6 ó Q7 ó Q9 ó Q11 ó Q12
Entradas : "RM"
Estado final : Q3

Descripción:

i) Si el "stack" de condición es igual a 2 ó 3, se decrementa éste, y también el "stack" de estados, asignando al registro NQ el nuevo estado, indicando con esto el término de una instrucción "SI".

ii) Se incrementa el "stack" de estados, el "stack" de condición con el valor de 1 y el "stack" de apuntadores al vector IXX, con el valor de PC-1, el cual apunta a la misma instrucción "RM". Esto se hace con el fin de conservar el apuntador para repetir la instrucción mientras se cumpla la condicional.

PROCESO P11.

Estado inicial: Q3
Entradas : cualquier palabra condicional
Estado final : Q10

Descripción:

- La descripción de este proceso es igual a la descripción del proceso P4.

PROCESO P12.

Estado inicial: Q10
Entradas : "("
Estado final : Q11

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómata de control.

PROCESO P13.

Estado inicial: Q7
Entradas : ")"
Estado final : Q1

Descripción:

i) Si el "stack" de condición es 2 ó 3: decrementará a éste y al "stack" de estados, asignando al registro NO el nuevo estado. Le asigna al "stack" de siguiente instrucción a ejecutar el valor de PC-1, esto es, vuelve a apuntar a la misma entrada: ")"; esto se hace con el fin de detectar en el siguiente ciclo la terminación de la primera parte de la instrucción "SI".

ii) Si el "stack" de condición es 0 ó 1 únicamente hará el cambio de estados en el autómata de control.

PROCESO P14.

Estado inicial: Q11
Entradas : ")"
Estado final : Q1

Descripción:

i) Si el "stack" de condición es distinto de 1:

- Decrementa el "stack" de estados y el "stack" de condición, asignándole al registro NQ el nuevo estado.
- Decrementa el "stack" que contiene los apuntadores al vector IX% para cada nivel de ejecución, i.e. el registro KW.

ii) Si el "stack" de condición es igual a 1:

- Le asigna al "stack" del programa el valor de la última celda del "stack" de apuntadores al vector IX%.
- Decrementa el "stack" de estados, el "stack" de condición y el "stack" de apuntadores al vector IX%, por último, le asigna al registro NQ el nuevo estado.

PROCESO P15.

Estado inicial: Q1 Q12 Q6 Q11 Q7 Q9
Entradas : RP RP RP RP RP RP
Estado final : Q4 Q4 Q4 Q4 Q4 Q4

Descripción:

i) Si el "stack" de condición es 2 ó 3:

- Le asigna a los "stacks" de condición y de estados el valor de cero.
- Decrementa los "stacks" de condición y de estados, le asigna al registro NQ el nuevo estado.

ii) Si el "stack" de condición es 0 ó 1:

- Incrementa los "stacks" de estado y de condición, le asigna al "stack" de condición el valor de 1.
- Incrementa el "stack" de apuntadores al vector IX%, y le asigna a éste el valor de PC-1, i.e. el número de celda en la cual se encuentra la instrucción de entrada "RP".

PROCESO P16.

Estado inicial: Q4
Entradas : "("
Estado final : Q12

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómata de control.

PROCESO P17.

Estado inicial: Q12
Entradas : ")"
Estado final : Q13

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómata de control.

PROCESO P18.

Estado inicial: Q13
Entradas : "M"
Estado final : Q14

Descripción:

- Este proceso únicamente hace el cambio de estados en el autómata de control.

PROCESO P19.

Estado inicial: Q14
Entradas : cualquier palabra condicional
Estado final : Q1

Descripción:

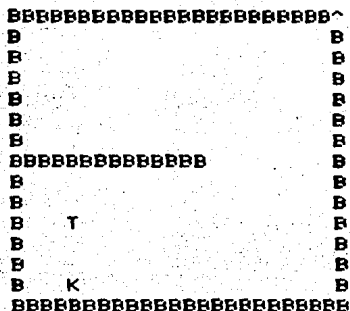
- Verifica la condicional, asignando el valor de 1 al "stack" de condicionales. Si esta se cumple ó el valor de cero en caso contrario (ver los incisos de la descripción del proceso P4).

- Si el "stack" de condición es distinto de el valor 1, decrementa los "stacks" de estados y de condición, decrementa el "stack" que contiene los apuntadores al vector IX% para cada nivel de ejecución. i.e. el registro KW. Le asigna al registro NQ el nuevo estado.

- Si el "stack" de condición es igual a 1, le asigna al "stack" de programa el valor de la última celda del "stack" de apuntadores al vector IX%, decrementa el "stack" de estados, el "stack" de condición y el "stack" de apuntadores al Vector IX%, por último le asigna al registro NQ el nuevo estado.

A continuación se verá mediante un ejemplo el funcionamiento de los "stacks" y registros asociados al autómata de Karel.

Supongamos que el mundo de Karel está construido de la siguiente forma:



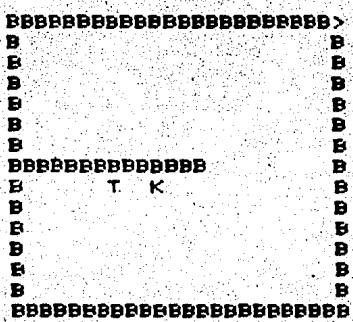
y se desea que Karel avance hacia el norte, recoja el trompo, al encontrar la barrera de vuelta a la derecha, avance tres celdas, deposite el trompo y avance tres celdas; podríamos ejecutar este trabajo mediante las siguientes instrucciones:

```

AC TO = (SI AS (RT) DM (AV)).
AC T1 = (RP (TO) MT NE).
EJ (T1 VD AV AV AV DT AV AV AV).

```

El mundo de Karel quedaría de la siguiente manera, una vez ejecutada la tarea descrita anteriormente.



En el vector IXX encontraremos cada una de las instrucciones almacenada de la siguiente forma:

código		código		código	
IXX (1) = (6	IXX (11) =)	7	IXX (21) = T1	30
IXX (2) = SI	8	IXX (12) = (6	IXX (22) = VD	2
IXX (3) = AS	20	IXX (13) = RP	10	IXX (23) = AV	1
IXX (4) = (6	IXX (14) = (6	IXX (24) = AV	1
IXX (5) = RT	4	IXX (15) = TO	29	IXX (25) = AV	1
IXX (6) =)	7	IXX (16) =)	7	IXX (26) = DT	5
IXX (7) = DM	9	IXX (17) = MT	11	IXX (27) = AV	1
IXX (8) = (6	IXX (18) = NE	15	IXX (28) = AV	1
IXX (9) = AV	1	IXX (19) =)	7	IXX (29) = AV	1
IXX (10) =)	7	IXX (20) = (6	IXX (30) =)	7

Los "stacks" en cada ciclo de operación contendrían lo siguiente:

Ent	IXX	SPX	SCX	SWX	SOX	NP	NO	RE	RI	KN	KD
T1	(21)=30	(1)=21	(1)=1	(1)=20	(1)=1	1	1	1	30	0	1
*RP	(13)=10	(2)=13	(1)=1	(2)=12	(2)=1	15	4	7	10	0	1
((14)=6	(2)=14	(2)=1	(3)=12	(3)=4	16	12	4	6	0	1
TO	(15)=29	(2)=15	(2)=1	(3)=12	(2)=12	1	12	1	29	0	1
SI	(2)=8	(3)=2	(2)=1	(4)=1	(4)=1	3	2	3	8	0	1
AS	(3)=20	(3)=3	(3)=1	(4)=1	(5)=2	4	5	2	20	0	1
((4)=6	(3)=4	(3)=0	(4)=1	(5)=5	5	6	4	6	0	1
RT	(5)=4	(3)=5	(3)=0	(4)=1	(5)=6	1	6	1	4	0	1
)	(6)=7	(3)=6	(3)=0	(4)=1	(5)=6	6	7	5	7	0	1
DM	(7)=9	(3)=7	(3)=2	(4)=1	(5)=7	7	8	6	9	0	1
((8)=6	(3)=8	(3)=1	(4)=1	(5)=8	8	9	4	6	0	1
AV	(9)=1	(3)=9	(3)=1	(4)=1	(5)=9	1	9	1	1	0	1
)	(10)=7	(3)=10	(3)=1	(4)=1	(5)=9	9	1	5	7	0	1
)	(11)=7	(3)=11	(2)=1	(3)=12	(4)=1	2	1	5	7	0	1
)	(16)=7	(2)=16	(2)=1	(2)=12	(3)=12	17	13	5	7	0	1
MT	(17)=11	(2)=17	(2)=1	(2)=12	(3)=13	18	14	8	11	0	1
NE	(18)=15	(2)=18	(2)=1	(1)=20	(3)=14	19	1	2	15	0	1

El autómata repite lo anterior desde el '+', hasta que Karel se encuentra en su casino un trompo, en cuyo caso, ejecuta la instrucción de repetición de la siguiente forma:

Ent	IXX	SPX	SCX	SMX	SOX	NP	NO	RE	RI	KN	KD
RP	(13)=10	(2)=13	(1)=1	(2)=12	(2)=1	15	4	7	10	0	1
((14)=6	(2)=14	(2)=1	(3)=12	(3)=4	16	12	4	6	0	1
TO	(15)=29	(2)=15	(2)=1	(3)=12	(3)=12	1	12	1	29	0	1
SI	(2)=8	(3)=2	(2)=1	(4)=1	(4)=1	3	2	3	8	0	1
AS	(3)=20	(3)=3	(3)=1	(4)=1	(5)=2	4	5	2	20	0	1
((4)=6	(3)=4	(3)=1	(4)=1	(5)=5	5	6	4	6	0	1
RT	(5)=4	(3)=5	(3)=1	(4)=1	(5)=6	1	6	1	4	0	1
)	(6)=7	(3)=6	(3)=1	(4)=1	(5)=6	6	7	5	7	1	1
DM	(7)=9	(3)=7	(3)=3	(4)=1	(5)=7	7	8	6	9	1	1
((8)=6	(3)=8	(3)=0	(4)=1	(5)=8	8	9	4	6	1	1
AV	(9)=1	(3)=9	(3)=0	(4)=1	(5)=9	1	9	1	1	1	1
)	(10)=7	(3)=10	(3)=0	(4)=1	(5)=9	9	1	5	7	1	1
)	(11)=7	(3)=11	(2)=1	(3)=12	(4)=1	2	1	5	7	1	1
)	(16)=7	(2)=16	(2)=1	(2)=12	(3)=12	17	13	5	7	1	1
MT	(17)=11	(2)=17	(2)=1	(2)=12	(3)=13	18	14	8	11	1	1
NE	(18)=15	(2)=18	(2)=1	(1)=20	(3)=14	19	1	2	15	1	1

A continuación se repite nuevamente el primer conjunto de instrucciones varias veces, hasta que Karel se encuentra con la barrera, en este caso, los stacks variarán de la siguiente forma:

Ent	IXX	SPX	SCX	SMX	SOX	NP	NO	RE	RI	KN	KD
RP	(13)=10	(2)=13	(1)=1	(2)=12	(2)=1	15	4	7	10	1	1
((14)=6	(2)=14	(2)=1	(3)=12	(3)=4	16	12	4	6	1	1
TO	(15)=29	(2)=15	(2)=1	(3)=12	(3)=12	1	12	1	29	1	1
SI	(2)=8	(3)=2	(2)=1	(4)=1	(4)=1	3	2	3	8	1	1
AS	(3)=20	(3)=3	(3)=1	(4)=1	(5)=2	4	5	2	20	1	1
((4)=6	(3)=4	(3)=0	(4)=1	(5)=5	5	6	4	6	1	1
RT	(5)=4	(3)=5	(3)=0	(4)=1	(5)=6	1	6	1	4	1	1
)	(6)=7	(3)=6	(3)=0	(4)=1	(5)=6	6	7	5	7	1	1
DM	(7)=9	(3)=7	(3)=2	(4)=1	(5)=7	7	8	6	9	1	1
((8)=6	(3)=8	(3)=1	(4)=1	(5)=8	8	9	4	6	1	1
AV	(9)=1	(3)=9	(3)=1	(4)=1	(5)=9	1	9	1	1	1	1
)	(10)=7	(3)=10	(3)=1	(4)=1	(5)=9	9	1	5	7	1	1
)	(11)=7	(3)=11	(2)=1	(3)=12	(4)=1	2	1	5	7	1	1
)	(16)=7	(2)=16	(2)=1	(2)=12	(3)=12	17	13	5	7	1	1
MT	(17)=11	(2)=17	(2)=1	(2)=12	(3)=13	18	14	8	11	1	1
NE	(18)=15	(2)=18	(2)=1	(1)=20	(3)=14	19	1	2	15	1	1
VD	(22)=2	(1)=22	(1)=1	(1)=20	(2)=1	1	1	1	2	1	1
AV	(23)=1	(1)=23	(1)=1	(1)=20	(2)=1	1	1	1	1	1	2
AV	(24)=1	(1)=24	(1)=1	(1)=20	(2)=1	1	1	1	1	1	2
AV	(25)=1	(1)=25	(1)=1	(1)=20	(2)=1	1	1	1	1	1	2
DT	(26)=5	(1)=26	(1)=1	(1)=20	(2)=1	1	1	1	5	1	2
AV	(27)=1	(1)=27	(1)=1	(1)=20	(2)=1	1	1	1	1	0	2
AV	(28)=1	(1)=28	(1)=1	(1)=20	(2)=1	1	1	1	1	0	2
AV	(29)=1	(1)=29	(1)=1	(1)=20	(2)=1	1	1	1	1	0	2
)	(30)=7	(1)=30	(1)=1	(1)=20	(2)=1	2	1	5	7	0	2

V. CONCLUSIONES

Dada la facilidad de acceso al programa de Karel, los jóvenes estudiantes de programación logran con gran facilidad entender las órdenes que deben darse al robot para que realice un trabajo determinado; al mismo tiempo, les permite el plantear problemas más complejos, tanto como su imaginación se los permita. Para ellos es una gran satisfacción el lograr que Karel realice exactamente lo que le indican a través de su lenguaje, ya que la solución del problema es visual, como si se tratara de un videojuego.

Lo importante es el hecho de que al ir aprendiendo el lenguaje de Karel, el estudiante está capacitado para manejar instrucciones lógicas, imperativas y de repetición, con lo cual el acceso a cualquier lenguaje de programación resulta mucho más sencillo.

Creo que la creación de este tipo de programas, puede ayudar en gran parte a la difícil tarea de alfabetización computacional en nuestro país, la que se hace cada vez más necesaria.

BIBLIOGRAFIA

INTRODUCCION A LA TEORIA MATEMATICA DE LAS
COMPUTADORAS Y DE LA PROGRAMACION
B. A. Trajtenbrot
Siglo XXI, editores, s.a. 1967

FINITE-STATE MODELS FOR LOGICAL MACHINES
Frederick C. Hennie
John Wiley & Sons, inc. 1968

CIENCIAS DE LA COMPUTACION VOL II
Lenguajes, Traductores y Aplicaciones
Presser-Cardenas y Marin
Limusa, s.a. 1972

FUNDAMENTALS ALGORITHMS
THE ART OF COMPUTER PROGRAMMING
Donald E. Knuth
Addison-Wesley 1975

PRINCIPLES OF COMPILER DESIGN
Alfred V. Aho, Jeffrey D. Ullman
Addison-Wesley 1979

KAREL THE ROBOT
A GENTLE INTRODUCTION TO THE ART OF
PROGRAMMING
Richard E. Pattis
John Wiley & Sons 1981

COMPILADORES
Luis Legarreta Garcíadiago
Fundacion Arturo Rosenblueth 1983