y en caso contrario un puntero nulo. Escriba otra función *srchinsrt*(l,x), que agregue x a l si éste no se encuentra y siempre retorna un puntero al nodo que contiene x.

9. Escriba un programa en PASCAL para leer un grupo de líneas de entrada, en la cual cada una contiene una palabra. Imprima cada palabra que aparece en la entrada y el número de veces que ésta aparece.

10. Asuma que una hilera de caracteres se representa por una lista de solamente caracteres, en la forma descrita al final de esta sección. Escriba un conjunto de rutinas para manipular estas listas en la forma como sigue (donde $l1$, $l2$ y *list* son punteros al nodo de encabezamiento de una lista representando una hilera de caracteres, *str* es un arreglo empacado de caracteres e $i1$, e $i2$ son enteros):

(a) *convcl*(*str*)	para convertir la hilera de caracteres *str* a una lista. Esta función retorna un puntero al nodo de encabezamiento.
(b) *convlc*(list,str)	para convertir una lista en una hilera de caracteres.
(c) *posl*(1,$l2$)	para realizar la función *pos* de la Sección 1.2 sobre dos hileras de caracteres representadas por listas. Esta función retorna un entero.
(d) *verifyl*($l1$,$l2$)	determina la primera posición de la hilera representada por $l1$ que no está contenida en la hilera representada por $l2$. Esta función retorna un entero.
(e) *substrl*($p1$,$i1$,$i2$)	para realizar la función *substr* de la Sección 1.2 en una hilera de caracteres representada por la lista $l1$ y los enteros $i1$ e $i2$. Esta función retorna un puntero al nodo de encabezamiento de una lista que representa una hilera de caracteres que viene a ser la subhilera deseada. La lista $l1$ permanece sin cambiar.
(f) *psubstrl*($l1$,$i1$, $i2$, $l2$)	para realizar una asignación de seudo-*substr* a la lista $l1$. Los elementos de la lista $l2$ deben remplazar los $i2$ elementos de $l1$ empezando en la posición $i1$. La lista $l2$ debe permanecer sin cambiar.
(g) *comparel*($l1$,$l2$)	para comparar dos hileras de caracteres representadas por listas. Esta función retorna -1 si la hilera de caracteres representada por $l1$ es menor que la hilera representada por $l2$, 0 si son iguales, y 1 si la hilera representada por $l1$ es mayor.

3. EJEMPLO: SIMULACION UTILIZANDO LISTAS ENCADENADAS

Una de las aplicaciones más útiles de colas y listas encadenadas es en ***simulación***. Un programa de simulación es aquel que intenta modelar una situación del mundo real con el fin de aprender algo acerca de él. Cada objeto y acción en la situación real tiene su contraparte en el programa. Si la simulación es correcta, es decir si el programa puede representar en forma exitosa el mundo real, entonces los resultados del programa deben mostrar los resul-

tados de las acciones que han sido simuladas. Es decir, es posible entender qué ocurre en las situaciones del mundo real sin que éstas ocurran realmente.

Miremos el siguiente ejemplo. Considere un banco con cuatro ventanillas. Un cliente entra al banco en una hora específica $t1$ y desea llevar a cabo una transacción en cualquier ventanilla. Se espera que la transacción tome el tiempo necesario $t2$ antes de ser completada. Si una de las ventanillas está libre, ésta puede procesar la transacción del cliente inmediatamente y el cliente sale del banco al tiempo $t1 + t2$ tan pronto como se ha completado la transacción. El tiempo total gastado en el banco por el cliente $t2$ es exactamente igual a la duración de la transacción.

Sin embargo, es posible que ninguna de las ventanillas esté desocupada; todas están ocupadas dando servicio a otros clientes que llegaron con anterioridad. En este caso existiría una línea de espera en cada una de las ventanillas. La línea para una ventanilla en particular puede estar formada por una sola persona, aquella que está realizando la transacción con esa ventanilla, o puede ser una línea muy larga. El cliente se dirigirá a la línea más corta y espera hasta que todos los clientes que están antes que él hayan completado su transacción y salgan del banco. En este momento el cliente puede realizar su transacción o negocio. El cliente sale del banco en un tiempo $t2$ unidades después de que ha llegado al frente de la línea de la ventanilla. En este caso el tiempo gastado en el banco es $t2$ más el tiempo gastado en la línea de espera.

Dado este sistema, estamos interesados en calcular el tiempo promedio gastado por un cliente en el banco. Una forma de hacer esto consiste en pararse en la puerta del banco, preguntarle a los clientes que salen el tiempo de su llegada y registrar el tiempo de su salida, luego restar el primero del segundo, y tomar el promedio para todos los clientes. Sin embargo, esto no sería muy práctico. Sería muy difícil el de asegurarse de que todos los clientes son realmente revisados al salir del banco. Además existe la duda de que la mayoría de los clientes recuerdan el tiempo exacto de su llegada.

En su lugar, podemos escribir un programa para simular la acción de los clientes. Cada parte de la situación del mundo real tiene su contraparte en el programa. Cada línea de entrada al programa representa un cliente. Las acciones en el mundo real de un cliente que llega son modeladas como una línea de entrada que es leída. Cuando llega un cliente, se presentan dos hechos: el tiempo de su llegada y la duración de su transacción (puesto que se supone que en el momento de la llegada la gente sabe qué es lo que desea hacer en el banco). De tal manera que cada línea de entrada contiene dos números: el tiempo (en minutos desde que el banco abrió) de llegada del cliente y el tiempo necesario (nuevamente en minutos) para su transacción. Estas líneas de entrada son ordenadas en forma ascendente con respecto al tiempo de llegada. Asumimos por lo menos una línea de entrada.

Las cuatro líneas en el banco se representan por medio de cuatro colas. Cada nodo de las colas representa un cliente esperando en una línea, y el nodo en el frente de la cola representa al cliente que está recibiendo el servicio en la ventanilla.

Supongamos que en algún instante de tiempo las cuatro líneas contienen cada una un número específico de clientes. ¿Qué puede suceder para que se altere el estado de la línea? Puede ser que llegue un nuevo cliente al banco,

en cuyo caso una de las líneas adicionará un nuevo cliente, o que el primer cliente en una de las cuatro líneas completa su transacción, en cuyo caso la línea tendrá un cliente menos. Es decir que hay un total de cinco acciones (un cliente entra, más cuatro casos de clientes saliendo) que pueden cambiar el estado de las líneas. Cada una de estas cinco acciones se llaman un ***evento.***

La simulación se hace encontrando el número de eventos que ocurren y efectuando el cambio en las colas que representen el cambio en las líneas del banco debido a estos eventos. Para tener el registro de los eventos, el programa utiliza una ***lista de eventos***. Esta lista contiene al menos cinco nodos, cada uno representando la ocurrencia de uno de los cinco tipos de eventos. Es decir que la lista de eventos contiene un nodo que representa la llegada del cliente siguiente y cuatro nodos que representan cada uno de los cuatro clientes en la cabeza de una línea que ha completado su transacción y salen del banco. Por supuesto es posible que una o más de estas líneas en el banco estén vacías o que las puertas del banco se hayan cerrado por ese día, de tal manera que no hay más llegada de clientes. En tales casos la lista de eventos contiene menos de cinco nodos.

Un nodo de eventos que representa la llegada de un cliente se denomina ***nodo de llegada***, y el nodo que representa la salida se denomina ***nodo de salida***. En cada punto en la simulación, es necesario conocer el evento siguiente que va a suceder. Por esta razón, la lista de eventos son ordenados de acuerdo al incremento de tiempo de la ocurrencia del evento de tal manera que el primer nodo de eventos en la lista representa el evento siguiente a ocurrir.

El primer evento que sucede es la llegada del primer cliente. La lista de eventos es por consiguiente inicializada leyendo la primera línea de entrada y colocando un nodo de llegada que representa la llegada del primer cliente en la lista de eventos. Por supuesto, inicialmente las cuatro colas están vacías. Entonces la simulación continúa como sigue. Se retira el primer nodo de la lista de eventos y los cambios que este evento ocasiona se hacen en las colas. Como veremos más adelante, estos cambios pueden también causar que sucedan otros eventos adicionales en la lista de eventos. El proceso de remover el primer nodo de la lista de eventos y efectuar los cambios que ella representa se repite hasta que la lista de eventos queda vacía.

Cuando se retira un nodo de llegada de la lista de eventos, se coloca un nodo que representa la llegada del cliente en la cola más corta que representa a las cuatro líneas. Si este cliente es el único en su cola, se coloca también un nodo que represente su salida en la lista de eventos, puesto que en ese momento se encuentra en el frente de la cola. Al mismo tiempo, se lee la línea siguiente de entrada y un nodo de llegada que representa la llegada del cliente siguiente se coloca en la lista de eventos. Por tanto, siempre habrá un nodo de llegada en la lista de eventos (siempre y cuando la entrada no esté agotada, en cuyo caso no llegan más clientes), porque tan pronto como se retira el nodo de llegada de la lista de eventos, se agrega otro a dicha lista.

Cuando un nodo de salida es removido de la lista de eventos, el nodo que representa el cliente de salida es removido del frente de una de las cuatro colas. En este momento la cantidad de tiempo que el cliente de salida ha gastado en el banco, es calculado y agregado al total. Al final de la simulación, este total debe ser dividido por el número de clientes para dar el tiempo promedio gastado por un cliente. Después de que un nodo del cliente ha sido retirado del frente de su cola, el cliente siguiente en la cola (si existe alguno)

pasa a ser el que ha de ser servido por esa ventanilla y se agrega a la lista de eventos un nodo de salida para ese nuevo cliente.

Este proceso continúa hasta que la lista de eventos queda vacía, en cuyo momento se calcula y se imprime el tiempo promedio. Observe que la lista de eventos en sí misma no representa ninguna parte de la situación del mundo real. Es utilizada únicamente como parte del programa para controlar el proceso en forma total. Una simulación de este tipo, que funciona al cambiar la situación simulada en respuesta a la ocurrencia de uno de los varios eventos, es denominada una ***simulación de cambio por evento***.

Ahora examinaremos las estructuras de datos necesarias para este programa. Los nodos en las colas representan clientes y por consiguiente deben contener campos para representar el tiempo de llegada, la duración de la transacción, además del campo *next* para enlazar los nodos de la lista. Los nodos en la lista de eventos representan eventos y deben contener el tiempo en el cual ocurre el evento , el tipo de evento y cualquier otra información asociada con el evento al igual que un campo *next*. Por consiguiente se ve que se requiere un registro variante o dos conjuntos de nodos separados para los dos tipos diferentes de nodos. El utilizar un registro variante equivale a ajustar y revisar el campo de identificación antes de insertar o utilizar un nodo. Dos tipos de nodos diferentes equivalen a tener dos rutinas de *getnode* y *freenode* y dos conjuntos de rutinas de manipulación de listas. Para evitar esta duplicación de rutinas, consideremos el uso de un solo tipo de nodo tanto para los eventos como para los clientes.

Podemos declarar este conjunto de nodos como sigue:

```
const numnodes = 500;
type  nodeptr = 0..numnodes;
      nodeinfo = record
                   time: integer;
                   duration: integer;
                   ntype : 0..4
                 end;
      nodetype = record
                   info: nodeinfo;
                   next: nodeptr
                 end;
var   node: array[1..numnodes] of nodetype;
```

Para un cliente, *time* es el tiempo de llegada del cliente y *duration* es la duración de la transacción. *ntype* no es usada en el nodo del cliente. *next* es usada como un puntero para enlazar la cola conjuntamente. Para un nodo de evento, *time* es utilizado para retener el tiempo de la ocurrencia del evento; *duration* es usada para la duración de la transacción del cliente que llega en un nodo de llegada y no es utilizado en un nodo de salida. *ntype* es un entero entre cero y cuatro dependiendo de si el evento es de llegada (*ntype* = 0) o de salida de la línea uno, dos, tres o cuatro (*ntype* = 1, 2, 3, ó 4). *next* contiene un puntero que encadena la lista de eventos conjuntamente.