Ejemplos de Tipos de datos Abstractos

LISTAS

Una lista lineal es un conjunto de elementos de un tipo dado que pueden variar en número y donde cada elemento tiene un único predecesor y un único sucesor o siguiente, excepto el primero y último de la lista. Esta es una definición muy general que incluye los ficheros y vectores. Los elementos de una lista lineal se almacenan normalmente contiguos —un elemento detrás de otro— en posiciones consecutivas de la memoria. Las sucesivas entradas en una guía o directorio telefónico, por ejemplo, están en líneas sucesivas, excepto en las partes superior e inferior de cada columna. Una lista lineal se almacena en la memoria principal de una computadora en posiciones sucesivas de memoria; cuando se almacenan en cinta magnética, los elementos sucesivos se presentan en sucesión en la cinta. Esta asignación de memoria se denomina almacenamiento secuencial. Posteriormente se verá que existe otro tipo de almacenamiento denominado encadenado o enlazado. Las líneas así definidas se denominan contiguas. Las operaciones que se pueden realizar con listas lineales contiguas son:

1. Insertar, eliminar o localizar un elemento.
2. Determinar el tamaño —número de elementos— de la lista.
3. Recorrer la lista para localizar un determinado elemento.
4. Clasificar los elementos de la lista en orden ascendente o descendente.
5. Unir dos o más listas en una sola.
6. Dividir una lista en varias sublistas.
7. Copiar la lista.
8. Borrar la lista.

Una lista lineal contigua se almacena en la memoria de la computadora en posiciones sucesivas o adyacentes y se procesa como un array unidimensional. En este caso, el acceso a cualquier elemento de la lista y la adición de nuevos elementos es fácil; sin embargo, la inserción o borrado requiere un desplazamiento de lugar de los elementos que le siguen y, en consecuencia, el diseño de un algoritmo específico. Para permitir operaciones con listas como arrays se deben dimensionar éstos con tamaño suficiente para que contengan todos los posibles elementos de la lista.

LISTAS ENLAZADAS

Los inconvenientes de las listas contiguas se eliminan con las listas enlazadas. Se pueden almacenar los elementos de una lista lineal en posiciones de memoria que no sean contiguas o adyacentes. Una lista enlazada o encadenada es un conjunto de elementos en los que cada elemento contiene la posición —o dirección— del siguiente elemento de la lista. Cada elemento de la lista enlazada debe tener al menos dos campos: un campo que tiene el valor del elemento y un campo (enlace, link) que contiene la posición del siguiente elemento, es decir, su conexión, enlace o encadenamiento. Los elementos de una lista son enlazados por medio de los campos

enlaces. Las listas enlazadas tienen una terminología propia que se suele utilizar normalmente. Primero, los valores se almacenan en un nodo.

Los componentes de un nodo se llaman campos. Un nodo tiene al menos un campo dato o valor y un enlace (indicador o puntero) con el siguiente nodo. El campo enlace apunta (proporciona la dirección o referencia de) al siguiente nodo de la lista. El último nodo de la lista enlazada, por convenio, se suele representar por un enlace con la palabra reservada nil (nulo), una barra inclinada (/) y, en ocasiones, el símbolo eléctrico de tierra o masa.

Un puntero (apuntador) es una variable cuyo valor es la dirección o posición de otra variable. En las listas enlazadas no es necesario que los elementos de la lista sean almacenados en posiciones físicas adyacentes, ya que el puntero indica dónde se encuentra el siguiente elemento de la lista. Por consiguiente, la inserción y borrado no exigen desplazamiento como en el caso de las listas contiguas.

Una lista enlazada sin ningún elemento se llama lista vacía. Su puntero inicial o de cabecera tiene el valor nulo (nil). Una lista enlazada se define por:

• El tipo de sus elementos: campo de información (datos) y campo enlace (puntero o apuntador).

• Un puntero de cabecera que permite acceder al primer elemento de la lista.

• Un medio para detectar el último elemento de la lista: puntero nulo (nil).

LISTAS CIRCULARES

Las listas simplemente enlazadas no permiten a partir de un elemento acceder directamente a cualquiera de los elementos que le preceden. En lugar de almacenar un puntero NULO en el campo SIG del último elemento de la lista, se hace que el último elemento apunte al primero o principio de la lista. Este tipo de estructura se llama lista enlazada circular o simplemente lista circular (en algunos textos se les denomina listas en anillo)

Las listas circulares presentan las siguientes ventajas respecto de las listas enlazadas simples:

• Cada nodo de una lista circular es accesible desde cualquier otro nodo de ella. Es decir, dado un nodo se puede recorrer toda la lista completa. En una lista enlazada de forma simple sólo es posible recorrerla por completo si se parte de su primer nodo.

• Las operaciones de concatenación y división de listas son más eficaces con listas circulares.

Los inconvenientes, por el contrario, son:

• Se pueden producir lazos o bucles infinitos. Una forma de evitar estos bucles infinitos es disponer de un nodo especial que se encuentre permanentemente asociado a la existencia de la lista circular. Este nodo se denomina cabecera de la lista.

El nodo cabecera puede diferenciarse de los otros nodos en una de las dos formas siguientes:

• Puede tener un valor especial en su campo INFO que no es válido como datos de otros elementos.

• Puede tener un indicador o bandera (flag) que señale cuando es nodo cabecera. El campo de la información del nodo cabecera no se utiliza.

LISTAS DOBLEMENTE ENLAZADAS

En las listas lineales estudiadas anteriormente el recorrido de ellas sólo podía hacerse en un único sentido: de izquierda a derecha (principio a final). En numerosas ocasiones se necesita recorrer las listas en ambas direcciones. Las listas que pueden recorrerse en ambas direcciones se denominan listas doblemente enlazadas. En estas listas cada nodo consta del campo INFO de datos y dos campos de enlace o punteros: ANTERIOR(ANT) y SIGUIENTE(SIG) que apuntan hacia adelante y hacia atrás. Como cada elemento tiene dos punteros, una lista doblemente enlazada ocupa más espacio en memoria que una lista simplemente enlazada para una misma cantidad de información. La lista necesita dos punteros CABECERA y FIN que apuntan hacia el primero y último nodo. La variable CABECERA y el puntero SIG permiten recorrer la lista en el sentido normal y la variable FIN y el puntero ANT permiten recorrerla en sentido inverso.

PILAS

Una pila (stack) es un tipo especial de lista lineal en la que la inserción y borrado de nuevos elementos se realiza sólo por un extremo que se denomina cima o tope (top). La pila es una estructura con numerosas analogías en la vida real: una pila de platos, una pila de monedas, una pila de cajas de zapatos, una pila de camisas, una pila de bandejas, etc. Dado que las operaciones de insertar y eliminar se realizan por un solo extremo (el superior), los elementos sólo pueden eliminarse en orden inverso al que se insertan en la pila. El último elemento que se pone en la pila es el primero que se puede sacar; por ello, a estas estructuras se les conoce por el nombre de LIFO (last-in, first-out, último en entrar, primero en salir). Las operaciones más usuales asociadas a las pilas son:

"push" Meter, poner o apilar: operación de insertar un elemento en la pila.

"pop" Sacar, quitar o desapilar: operación de eliminar un elemento de la pila.

Idealmente una pila puede contener un número ilimitado de elementos y no producir nunca desbordamiento. En la práctica, sin embargo, el espacio de almacenamiento disponible es finito. La codificación de una pila requiere un cierto equilibrio, ya que si la longitud máxima de la pila es demasiado grande se gasta mucha memoria, mientras que un valor pequeño de la longitud máxima producirá desbordamientos frecuentes. Para trabajar fácilmente con pilas es conveniente diseñar subprogramas de poner (push) y quitar (pop) elementos. También es necesario con frecuencia comprobar si la pila está vacía; esto puede conseguirse con una variable o función booleana VACIA, de modo que cuando su valor sea verdadero la pila está vacía y falso en caso contrario.

COLAS

Las colas son otro tipo de estructura lineal de datos similar a las pilas, diferenciándose de ellas en el modo de insertar/eliminar elementos. Una cola (queue) es una estructura lineal de datos

var array [1..n] de : C

en la que las eliminaciones se realizan al principio de la lista, frente (front), y las inserciones se realizan en el otro extremo, final (rear). En las colas el elemento que entró el primero sale también el primero; por ello se conoce como listas FIFO (first-in, first-out, "primero en entrar, primero en salir"). Así, pues, la diferencia con las pilas reside en el modo de entrada/salida de datos; en las colas las inserciones se realizan al final de la lista, no al principio. Por ello las colas se usan para almacenar datos que necesitan ser procesados según el orden de llegada.

DOBLE COLA

Existe una variante de la cola simple estudiada anteriormente y que es la doble cola. La doble cola o bicola es una cola bidimensional en la que las inserciones y eliminaciones se pueden realizar en cualquiera de los dos extremos de la lista.

Existen dos variantes de la doble cola: • Doble cola de entrada restringida: acepta inserciones sólo al final de la cola. • Doble cola de salida restringida: acepta eliminaciones sólo al frente de la cola.

ÁRBOLES

El árbol es una estructura de datos fundamental en informática, muy utilizada en todos sus campos, porque se adapta a la representación natural de informaciones homogéneas organizadas y de una gran comodidad y rapidez de manipulación. Esta estructura se encuentra en todos los dominios (campos) de la informática, desde la pura algorítmica (métodos de clasificación y búsqueda...) a la compilación (árboles sintácticos para representar las expresiones o producciones posibles de un lenguaje) o incluso los dominios de la inteligencia artificial (árboles de juegos, árboles de decisiones, de resolución, etc.). Las estructuras tipo árbol se usan principalmente para representar datos con una relación jerárquica entre sus elementos, como son árboles genealógicos, tablas, etc. Un árbol A es un conjunto finito de uno o más nodos, tales que:

1. Existe un nodo especial denominado RAIZ(v1) del árbol.

2. Los nodos restantes (v2, v3, ..., vn) se dividen en m >= 0 conjuntos disjuntos denominado A1, A2, ..., Am, cada uno de los cuales es, a su vez, un árbol. Estos árboles se llaman subárboles del RAIZ.

La definición de árbol implica una estructura recursiva. Esto es, la definición del árbol se refiere a otros árboles. Un árbol con ningún nodo es un árbol nulo; no tiene raíz.

Bibliografía

Joyanes, L. Fundamentos de Programación. Algoritmos y estructura de datos. McGraw-Hill.

México. 1990.