Facultad de Informática

Ingeniería en Informática de Gestión

Estructuras de Datos y de la Información

Grupo A, Mayo 2002

Ejercicios sobre tablas

Ejercicio 1 Los compiladores de los lenguajes de programación con estructura de bloques usan tablas de símbolos estructuradas del mismo modo. Cuando el compilador est'a inspeccionando un cierto punto del programa, la tabla de símbolos contiene una sección correspondiente al bloque actual, y otras secciones creadas anteriormente, correspondientes a bloques textualmente más externos. En cada sección, la tabla registra las declaraciones de diversos atributos asociados a los identificadores del bloque correspondiente, los cuales pueden ser tipos, direcciones de memoria etc. La siguiente figura muestra un pequeo programa en Pascal junto con la tabla de símbolos T correspondiente al momento en que el compilador está inspeccionando la instrucción `writeln(c)`. T se compone de dos secciones: T_2 corresponde al bloque actual y T_1, correponde al bloque textualmente más externo (el del programa principal).

```
program P
var a,b: integer;
procedure Q(a:integer);
  var a,d :real;
  begin writeln(c) end
begin Q(a) end.
```

id	atr		
P	prog	–	10230
a	var	int	1024
b	var	int	1026
Q	proc	–	10500
c	param	int	1028
a	var	real	1030
d	var	real	1032

Las dos reglas principales que rigen el funcionamiento de estas tablas de símbolos son: a) cada sección registra como mucho una declaración de atributos para un mismo identificador; y b) los atributos asociados a un identificador se consultan comenzando por la sección más recientemente creada de la tabla, correspondiente al bloque actual, y retrocediendo hacia las secciones de bloques más externos. Diseña una especificación algebraica de las tablas de símbolos con estructura de bloques, incluyendo las operaciones que se describen informalmente a continuación. Debes suponer disponible una operación booleana de igualdad entre identificadores.

$crear :\rightarrow ts$	crea una tabla vacía (bloque del programa principal)
$entra : ts \rightarrow ts$	crea una nueva sección en la tabla, entrando en un nuevo bloque
$sal : ts \rightarrow ts$	borra la sección actual de la tabla, regresando al bloque exterior
$declara : ts\ id\ atr \rightarrow ts$	inserta en la sección actual de la tabla la declaración de unos atributos para un identificador
$consulta : ts\ id \rightarrow atr$	consulta en la tabla los atributos asociados a un identificador
$declarado : ts\ id \rightarrow bool$	decide si un identificador está declarado en la tabla
$eslocal : ts\ id \rightarrow bool$	decide si un identificador está declarado en la sección más reciente de la tabla (bloque actual)

Suponiendo implementados los tipos *id* y *atr*, define una estructura de datos adecuada para representar las tablas de símbolos con estructura de bloques. Usando la representación elegida, diseña métodos que implementen las operaciones *declara* y *consulta*.

Ejercicio 2 Muestra cual es el resultado de insertar los números 185, 99, 145, 71, 197, 129, 72, 172, 48, 108, 142, 122 en una tabla de tamao 13 cuando se emplea dispersión lineal, cuadrática y doble (con $h'(k) = k\ mod\ 11$), con la función $h(k) = k\ mod\ 13$.

Ejercicio 3 Diseña un método de inserción en una tabla de dispersión en la que cada componente ocupada además de los campos *clave*, *valor* y *ocupada*, tiene otro que dice cuántas claves sinónimas de la clave que ocupa esa componente incluida ella existen en la tabla. Se supone que que la secuencia de prueba emplea hashing lineal y que la tabla no está llena.

Ejercicio 4 El *hash ordenado* inserta una clave en una tabla de la siguiente manera. Se calcula su localización. Si la posición está vacía, se inserta; si está ocupada por una clave menor, se sustituye por ella y se pasa a insertar la que ya estaba; si está ocupada por una mayor, se calcula una nueva localización y se vuelve a repetir el proceso. Diseña un método que implemente este tipo de inserción.

Ejercicio 5 Diseña un método que implemente la llamada *dispersión de Robin Hood* que para insertar una clave *c* procede como en otros tipos de dispersión salvo cuando ésta colisiona con otra clave *c'*. En este caso calcula con cuántas clave colisionó *c'* cuando se insertó y con cuántas ha colisionado hasta ahora *c*; deja entonces la clave que tenga mayor número y pasa a insertar la otra.

Ejercicio 6 El *encadenamiento interno* consiste en guardar las claves sinónimas en listas implementadas estáticamente dentro de la propia tabla. Cuando se necesita una posición vacía se recorre la tabla de abajo a arriba hasta encontrarla. Diseña un método que implemente la inserción en una tabla de este tipo.