Examen Final de PRED

FIB-UPC

7 de enero de 2011

---

1. **(2.5 puntos)** Se pide realizar la especificación algebraica del tipo de datos **`Diccionario`** con las operaciones usuales de `diccionario_vacio`, `actualizar`, `consultar` y `suprimir` la información asociada a una clave o índice en un diccionario. La semántica de estos operadores es la siguiente:
   - `diccionario_vacio`: como sugiere el identificador de este operador, devuelve un diccionario en el que no hay ninguna información.
   - `actualizar`: dado un `diccionario`, una `clave` y la `información` asociada a ella, inserta o actualiza la `información` en el `diccionario`.
   - `consultar`: dado un `diccionario` y una `clave` devuelve la `información` asociada a ella en el `diccionario`.
   - `suprimir`: dado un `diccionario` y una `clave`, `devuelve` un diccionario donde se ha eliminado la `información` asociada a la `clave` en el `diccionario`.

2. **(2 puntos)** Comentar en no más de 15 líneas las ventajas e inconvenientes del *hashing* abierto que requiere función de *rehashing* (en el libro de Ricardo Peña se denomina tabla dispersa cerrada) y el *hashing* encadenado.

3. **(2.5 puntos)** Se desea implementar el tipo abstracto de datos **`Bosque_arboles`** de árboles generales de enteros y las operaciones constructoras `bosque_vacio` y `añadir` un árbol del bosque.
   Se pide utilizar la representación de árboles generales y bosques de árboles mediante árboles binarios.
   En concreto, se pide lo siguiente:
   a) Definir la representación del tipo abstracto **`Bosque_arboles`**, su invariante y la equivalencia de la representación, y dar un ejemplo gráfico de bosque de árboles generales representado como un árbol binario.
   b) Implementar la operación de añadir un árbol como primer árbol del bosque.

Examen Final de PRED

FIB-UPC

7 de enero de 2011

4. (3 puntos) Dado el siguiente programa en C++, decid los números de las líneas del programa que producen un error de compilación justificando el motivo. A continuación, dar el resultado que produciría el programa sin las líneas erróneas.

```
1  #include <iostream>
2  using namespace std;
3
4
5  class A  {
6
7     private:
8          int x;
9
10    protected:
11         int y;
12
13     public:
14
15         int z;
16
17         A(){
18          x= 2; y=2; z=2;
19           }
20
21   int  g(int z) {
22          x = x + z;
23        return x;}
24 };
25
26 class B: public  A {
27
28  private:
29         int x1;
30  protected:
31         int y1;
32  public:
33   int z1;
34
35   B(){x1=4;
36       y1=4;
37       z1=4;
38        }
39
40   int  g(int a)    {
41         x1 = x1 + 2*z + a;
42    return x1;}
43 };
44 class  C: protected  A {
45
46  private: int x2;
47  protected: int y2;
48  public: int z2;
49
50  C(){ x2=6;
51       y2=6;
52       z2= 6;
53     }
54
55
56  C   h(C c, A a)    {
57
58       c.x = c.x + 2;
59
60       a.y = a.y + 2;
61       c.y = c.y + 2;
62
63       c.z  = c.z +2;
64          return c ;}
65 };
66
67 int main() {
68           A a;
69           B b;
70           C c;
71           int res=0;
72           res=a.g(a.z);
73           cout << res <<  endl;
74           a=b;
75           res= a.g(a.z);
76           cout << res << endl;
77           cout << c.x << endl;
78           cout << c.y << endl;
79           cout << c.z << endl;
80           c= c.h(c,a);
81           cout << c.x << endl;
82           cout << c.y << endl;
83           cout << c.z << endl;
84 }
85
86
```

Examen Final de PRED

FIB-UPC

7 de enero de 2011

---

1 **Especificacion** Diccionario
usa diccionario, indice, valor
operaciones

diccionario_vacio: -> Diccionario
actualizar: Diccionario x indice x valor -> Diccionario
consultar: Diccionario x indice -> valor (parcial)
suprimir: Diccionario x indice -> Diccionario

ecuaciones

actualizar(actualizar(d,i,v),i,w) = actualizar(d,i,w)
i≠j => actualizar(actualizar(d,i,v),j,w) = actualizar(actualizar(d,j,w),i,v)
consultar(diccionario_vacio) indef.
consultar(actualizar(d,i,v),i) = v
i≠j => consultar(actualizar(d,i,v),j) = consultar(d,j)
suprimir(diccionario_vacio)= diccionario_vacio
suprimir(actualizar(d,i,v),i) = suprimir(d,i)
i≠j => suprimir(actualizar(d,i,v),j) = actualizar(suprimir(d,j),i,v)

fespecificacion

2.

Una desventaja del hashing abierto en cuanto a espacio se requiere es que la tabla tiene que tener muchas más posiciones que el total de elementos se quieren insertar. Esto evita que la operación de búsqueda sea ineficiente cuando la tabla esta próxima a llenarse. Además en este hashing es conveniente saber con cierta exactitud el número de elementos a insertar en la tabla.
Otra desventaja del hashing abierto es que la operación suprimir puede hacer menos eficiente la búsqueda.

Como ventaja del hashing abierto tenemos que la tabla se almacena en la pila y no el heap, mientras que el hashing encadenado utiliza el heap para almacenar la información.
Como ventajas del hashing encadenado tenemos que la tabla no degenera con la operación suprimir y la memoria utilizada es proporcional al número de claves almacenadas.

3

```
modulo Bosque_arboles
    usa entero, arbol
    ops
        accion bosque_vacio (sal b:Bosque_arboles)
        accion insertar_arbol (ent a:arbol, ent/sal b:Bosque_arboles)
    representación
      tipo
        bosque =↑nodo_ar;
        nodo_ar = tupla
                    ari:bosque;
                    i:entero;
                    ard:bosque;
                  ftupla
```

Invariante de representación

B es un bosque si y sólo si

a) B = NIL o

b) B↑ = <NIL,i,NIL> o

c) B↑ = <ari,i,ard> y ari y ard son bosques

Equivalencia de representación

b1 ≡ b2 si y sólo si

a) b1 y b2 son vacíos o

b) Si b1↑= <b1i,i,b1d> y b2↑ = <b2i,j,b2d> entonces i = j,
b1i ≡ b2i y b1d ≡ b2d

```
accion añadir_arbol(ent arg: arbol, ent/sal b:bosque)
    {Pre: a ≠ NIL}
    {Post: añadimos arg como primer árbol del bosque}
    arg ↑.ard:= b;
    b:=arg;
faccion
```

4. Los errores serían en las siguientes líneas:

58 c.x privado de A no es heredado por C

60 a.y atributo protegido de A

77 c.x no es heredado por C

78 c.y es un atributo protegido de C

79 c.z es un atributo protegido de C

81 c.x no es heredado por C

82 c.y es un atributo protegido de C

83 c.z es un atributo protegido de C

Después de borrar estas líneas del programa y después de compilarlo y ejecutarlo el resultado sería 4 4