Árboles multicamino

Tema 6

1. DEFINICION DE LOS TIPOS DE DATOS.

```
#define   m      4      /* Orden del árbol. */
#define   Nulo   -1     /* Puntero nulo */
#define   nK     K[0]   /* Macro para acceder al número de claves. */

#define   NPILA  16     /* Tamaño de la pila. */

typedef int  TClave;   /* Tipo de las claves. */
typedef long TPuntero; /* Tipo de los punteros dentro de los nodos */

typedef int  TPosicion; /* Tipo de las posiciones dentro de los arrays. */

typedef struct
{ TClave K[m];          /* Array de claves. En K[0] estará n. */
  TPuntero A[m];        /* Array de punteros a nodos. */
} TNodo;

typedef struct
{ TNodo Nodo;
  TPuntero Direccion;   /* Dirección del nodo en el fichero. */
  TPosicion Posicion;   /* Posición por la que se desciende. */
} TPila[NPILA];
```

2. BÚSQUEDA DE UNA CLAVE DENTRO DE UN NODO

```
Funcion BuscarPosicion(Nodo, X, Pos)
Nodo: nodo donde se va a buscar (E).
X: valor de la clave a buscar (E).
Pos: posición de la clave buscada dentro del nodo (S).
{
  Primero = 1;
  Ultimo = N.nK;
  while (Primero <= Ultimo)
     { Medio = (Primero + Ultimo) / 2;
       if (N.K[Medio] == X)
          { Pos = Medio;
            return TRUE;
          }
       if (X < N.K[Medio]) Ultimo = Medio – 1;
       else                Primero = Medio + 1;
     }
  Pos = Primero – 1;
  return FALSE;
}
```

3. BÚSQUEDA EN ÁRBOL MULTICAMINO

```
Funcion BuscarAMC(R, X, Pila, Top)
R: dirección del nodo raíz (E).
X: valor de la clave a buscar (E).
Pila: guarda info del recorrido hasta el nodo que contiene X (S).
Top: devuelve el top de la pila (S).
{
  Top = -1;
  Dir = R;
  Encontrado = FALSE;
  while ((!Encontrado) && (Dir != Nulo))
     { Leer de la dirección Dir el nodo N;
       Encontrado = BuscarPosicion(N, X, Pos);
       Top++;
       Pila[Top].Nodo = N;
       Pila[Top].Direccion = Dir;
       Pila[Top].Posicion = Pos;
       if (!Encontrado)
          Dir = N.A[Pos];
     }
  return Encontrado;
}
```

4. INSERCIÓN EN UN ARBOL MULTICAMINO

```
Funcion InsertarAMC(R, X)
R: dirección del nodo raíz del árbol (E/S).
X: valor de la clave a insertar (E).
{
  Encontrado = BuscarAMC(R, X, Pila, Top);
  if (!Encontrado)
     { if (Top >= 0)
          { N = Pila[Top].Nodo;
            Dir = Pila[Top].Direccion;
            Pos = Pila[Top].Posicion;
            if (N.nK < m - 1)
               { N.nK++;
                 for (i = N.nK; i > Pos + 1; i--)
                    N.K[i] = N.K[i - 1];
                 N.K[Pos + 1] = X;
               }
            else
               N.A[Pos] = CrearNuevoNodo(X);
            Escribir nodo N en dirección Dir;
          }
       else
          R = CrearNuevoNodo(X);
     }
  return !Encontrado;
}
```

5. RECORRIDO EN INORDEN

```
Procedimiento InordenAMC(R)
R: dirección del nodo raíz (E).
{
  if (R != Nulo)
     { Leer de la dirección R el nodo N;
       InordenAMC(N.A[0]);
       for (i = 1; i <= N.nK; i++)
          { Procesar(N.K[i]);
            InordenAMC(N.A[i]):
          }
     }
  return;
}
```

6. ACCESO SECUENCIAL DIRECTO

```
Funcion SiguienteClave(Pila, Top)
Pila: almacena el recorrido hasta la clave (E/S).
Top: top de la pila (E/S).
{
  N = Pila[Top].Nodo;
  Pos = Pila[Top].Posicion;
  if (N.A[Pos] != Nulo)
     { P = N.A[Pos];
       while (P != Nulo)
          { Leer de la direccion P el nodo N;
            Top++;
            Pila[Top].Nodo = N;
            Pila[Top].Direccion = P;
            Pila[Top].Posición = 0;
            P = N.A[0];
          }
       Pila[Top].Posicion = 1;
       Resultado = N.K[1];
     }
  else
     { if (Pos < N.nK)
          { Pila[Top].Posición = Pos + 1;
            Resultado = N.K[Pos + 1];
          }
       else
          { while ((Top > 0) && (Pos == N.nK))
               { Top--;
                 N = Pila[Top].Nodo;
                 Pos = Pila[Top].Posicion;
               }
            if (Pos == N.nK) Resultado = ClaveNula;
            else
               { Pila[Top].Posición = Pos + 1;
                 Resultado = N.K[Pos + 1];
               }
          }
     }
  return Resultado;
}
```