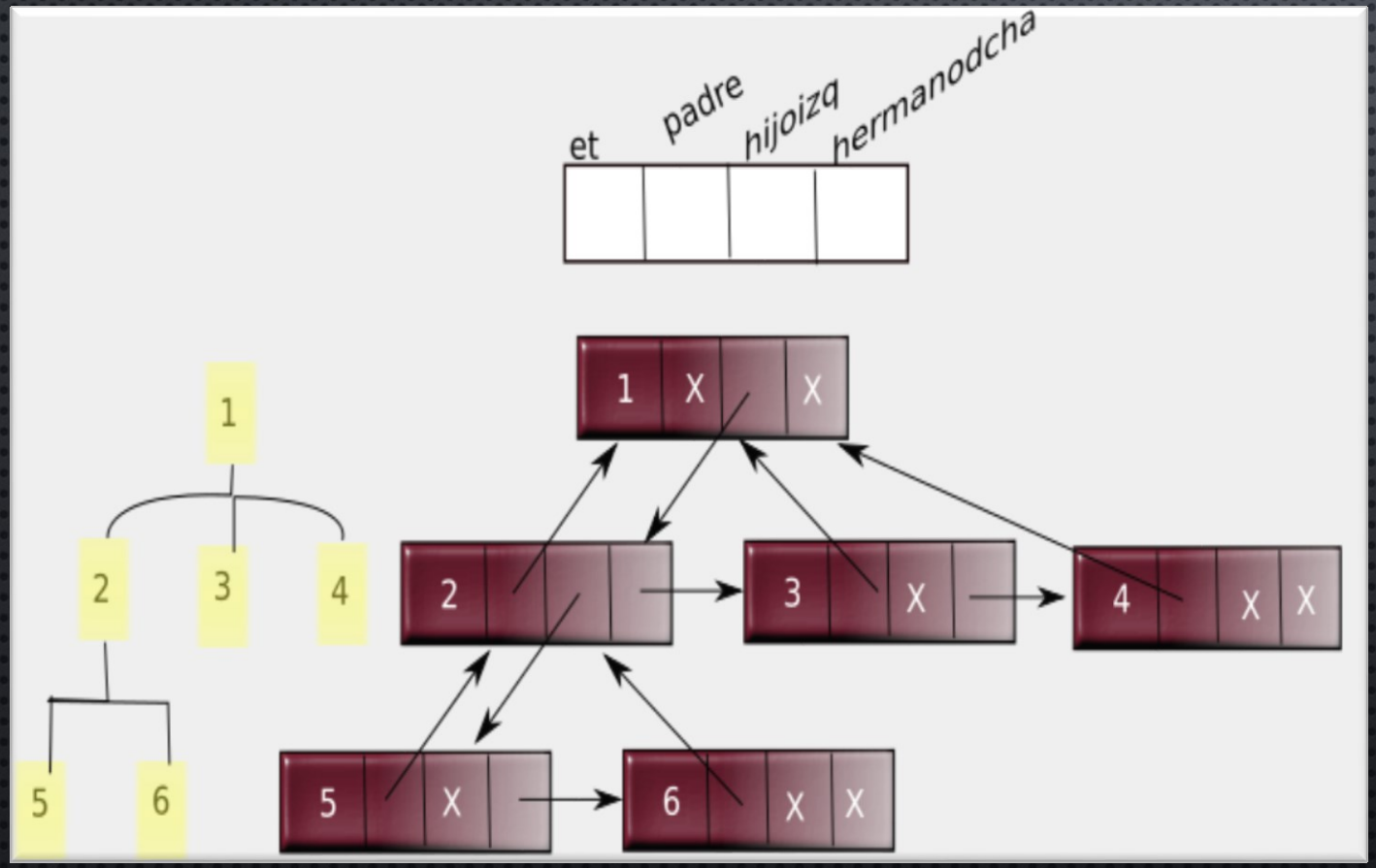
ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.




```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
    }
     info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
     }
}
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
          * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
info_nodo<T>* CrearRaiz(const T &e){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
void Copiar (info_nodo<T> *s, info_nodo<T>* &d){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
           * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
void Borrar(info_nodo<T> *s){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
void InsertarHijoIzq(info_nodo<T>* n, info_nodo<T>* &T2){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
void InsertarHermanoDrcha(info_nodo<T>* n, info_nodo<T>* &T2){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
info_nodo<T>* PodarHijoIzq(info_nodo<T>* n){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
           * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
info_nodo<T>* PodarHermanoDrcha(info_nodo<T>* n){
```

ARBOLES: ARBOLES GENERALES

Arbol Generales.- Pueden contener cualquier número de hijos.

```
template <class T>
struct info_nodo {
  T et
  info_nodo<T> * padre,
               * hijoizq,
              * hermanodcha;

   info_nodo() {
          padre=
          hijoizq=
          hermanodcha = 0;
   }
    info_nodo(const T & e) {
           et = e;
           padre =
           hijoizq =
          hermanodcha = 0;
    }
}
```

```
template <class T>
int Altura(info_nodo<T>* n){
```