Arboles

- Estructura de datos, ramas
- Se puede definir recursivamente
- Puede ser un unico elemento, la nada, o una estructura de multiples nodos de los cuales salen multiples ramas

Vamos a usar arboles binarios, con solo dos ramas por nodo que pueden o no ser vacias, pero tienen que estar definidas. (un poco invent eso de que tiene que estar definidas).

En general los arboles se definen en relacion a la cantidad de subarboles que pueden soportar, por ejemplo un arbol unario es una lista, un arbol binario soporta hasta dos subarboles, y asi.

Propiedades interesantes de los arboles

- Cantidad de nodos
- Cuanto crecen vertical y horizontalmente

En haskell: (osea matematica y recursivamente)

- altura(nil)=0
- altura(<n,i,d>)=1+max{altura i, altura d}
- elementos(nil)=[]
- elementos(<n,i,d>)=[n] ++ elementos i ++ elementos d

Una implementacion del arbol binario se puede implementar de una forma similar a una lista doblemente enlazada de la forma *|puntero1|valor|puntero2|*
Viendo a los arboles como tads, que operaciones deberia tener?

TAD ArbolBinario {
obs :

proc vacio (Parametros) : Resultado
requiere {$true$}
asegura {$true$}

proc altura (Parametros) : Resultado
requiere {$true$}
asegura {$true$}

proc bin (Parametros) : Resultado
requiere {$true$}
asegura {$true$}

Como el tad de arbol binario no es muy util, directamente vamos a trabajar sobre la implementacion en nodos
}

Tipo especial de arbol: Arbol Binario de Busqueda Para todo nodo, los valores de su subarbol izquierdo son menores que el valor del nodo y los valores del subarbol derecho son mayores.

modulo ConjuntoConABB **implementa** ArbolBinario{
var e: ArbolBinario
pred InvRep (e){
esABB(e)=true
}
pred Abs (e){
c.elems={n:N | n in elementos(e.raiz)}
}
}

↑ tomar con pinzas porque la slide tiene mas typos que otra cosa

La idea importante es la del principio, a la izquierda todos menores, a la derecha todos mayores, para cualquier nodo que agarre.

Algoritmos para Arbol de Busqueda Binario

- vacio
- Busqueda
- Insercion
- Eliminar

Nodo = Struct <dato: N, izq: Nodo, dar: Nodo> (opcionalmente un padre)

```
modulo AB implementa Conjunto{
      var raiz: Nodo
      pred InvRep (args){
          cuerpo
      }
      pred Abs (args){
          cuerpo
      }
      impl vacio():ABB{
            a= new ABB();
            a.raiz=null;
            return a;
         }
      impl Busqueda(a: ABB, k: N):{
            return busqueda(a.raiz, k)
         }
      impl busqueda(n,k):nodo{
            if n==null || k==n.dato then
               return n
            else
               skip
            end if
            if k<n.dato then
               busqueda(n.izq, k)
            else
               busqueda(n.der, k)
            end if
         }
      impl insercion mia(a: ABB, n: ℕ):ABB{
            if a.dato==n then
               skip
            else
               if a.dato<n then
                  insercion(a.izq, n)
               else
                  insercion(a.der, n)
               end if
            end if
            a.dato=new Nodo(null,null,k) medio pelo quedo, esta mal
         }
```

```
    impl insercion(inout a: ABB, k: ℕ):{
          n=a.raiz
          padre= raiz;
          while a! = null do
             padre=n;
             if k<n.dato then
                n = n.izq
             else
                n = n.der
             end if
          end while  newnodo= new Nodo(k, nil,nil, padre)
          if padre==null then
             a.raiz=newnodo
          else
             if k<padre.dato then
                padre.izq=newnodo
             else
                padre.der=newnodo
             end if
          end if
       }
}
```