TDA CONJUNTO

Ing. Mario Milton López Winnipeg

Capitulo 2 TDA Conjunto

2 Conjunto

2.1 Descripción del TDA Conjunto.

2.2 Especificación del TDA Conjunto.

2.3 Ejemplos de uso.

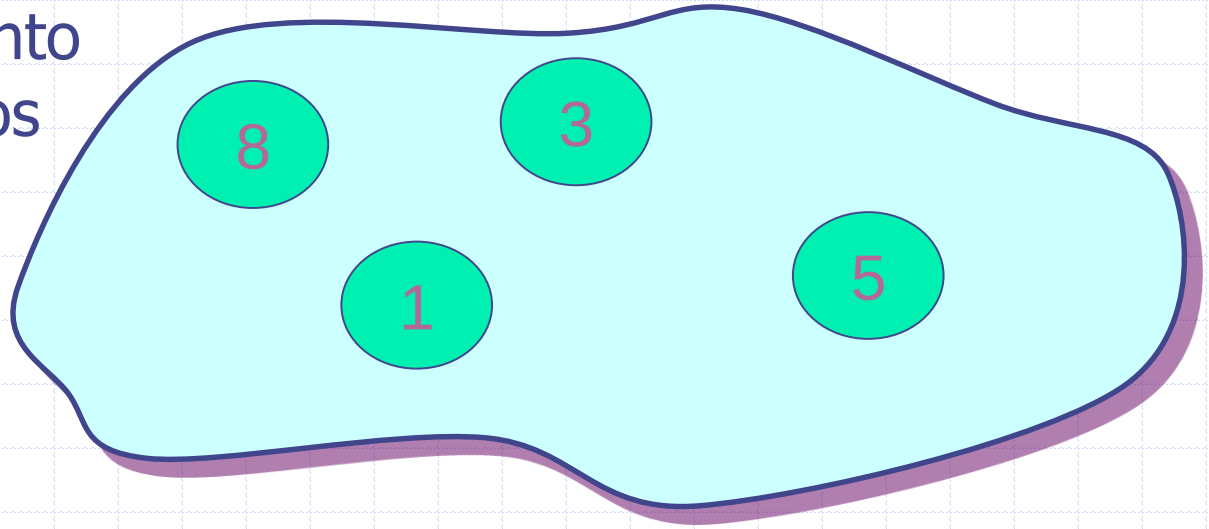
2.4 Implementaciones del TDA Conjunto.

- 2.4.1 Implementación con vectores.
- 2.4.2 Implementación con apuntadores
- 2.4.3 Implementación basada en el TDA Lista.

2.1 Descripción del TDA Conjunto

- **Conjunto:** Colección no ordenada de elementos (o miembros) distintos
- **Elemento:** Cualquier cosa, puede ser un elemento primitivo o, a su vez, un conjunto

C: Conjunto de enteros



2.2 Especificación del TDA Conjunto

Especificación Informal

- **Conjunto = TDA con operaciones crea, vacio, pertenece, inserta, suprime, cardinal, muestra**
- **DESCRIPCIÓN:Los valores del TDA Conjunto son conjuntos de elementos del tipo Elemento. El TDA Conjunto es mutable: las operaciones inserta y suprime, añaden y suprimen elementos del conjunto respectivamente**

- **OPERACIONES:**
  - `crea() devuelve (C:Conjunto)`
    - **efecto: Devuelve la Conjunto vacio `C`.**

2.2 Especificación del TDA Conjunto

vacio(C:Conjunto) devuelve (booleano)

efecto: Devuelve cierto si C es vacío y falso en caso contrario

Cardinal(C: Conjunto) devuelve (entero)

Efecto: Devuelve el numero de elementos de C

Ordinal(C: Conjunto; E: Elemento) devuelve (entero)

Requerimiento: Elemento E pertenece a C

Efecto: Devuelve el lugar que ocupa el elemento en C

Inserta (C:Conjunto, E:Elemento)

Requerimiento: Elemento E no pertenece a C

Modifica : Conjunto C

Efecto : Añade E al conjunto C

2.2 Especificación del TDA Conjunto

Suprime (C:Conjunto, E:Elemento)

Requerimiento: Elemento E pertenece a C

Modifica : Conjunto C

Efecto : Elimina E del conjunto C

Pertenece(C: Conjunto;E Elemento) devuelve (Boleano)

Efecto devuelve cierto si E pertenece al conjunto C y falso en caso contrario

Muestrea(C:conjunto) devuelve (elemento)

Requerimiento: Conjunto no vacio

Efecto : Devuelve un elemento E aleatorio del conjunto C

2.2 Especificación del TDA Conjunto

La interface del TDA Conjunto de acuerdo a esta especificación puede definirse de la siguiente forma:

Clase Conjunto

class Conjunto

Constructores

crear()

Métodos

boolean vacio()

int cardinal()

int ordinal(elemento e)

void inserta(Elemento e)

void suprime(Elemento e)

boolean pertenece(Elemento e)

Elemento muestrea()

2.3 Ejemplo de uso

```
publico Unión( Conjunto A,B,C)
  inicio
    mientras a.cardinal<> 0
        inicio
            elem= a.muestrea()
            c.inserta(elem)
            a.suprime(elem)
        fin
        mientras b.cardinal<> 0
         inicio
            elem= b.muestrea()
            c.inserta(elem)
            b.suprime(elem)
        fin
  fin
```

Conjunto A { }

Conjunto B { }

Conjunto C { }

2.4 Implementaciones del TDA Conjunto

- En esta sección mostraremos tres implementaciones para el TDA Conjunto:
  - Implementación con vectores
  - Implementación con apuntadores
  - Implementación basada en el TDA Lista

5.4.1 Implementacion basada en vectores

Definición básica de la clase Conjunto cuya implementacion es usando vectores

```
Clase Conjunto
    Atributos
        V       Arreglo(MAX)     // Elementos
       cant     entero             //  Determina la cantidad de elementos
     Metodos
          boolean vacio()
          int cardinal()
          int ordinal(elemento e)
          void inserta(Elemento e)
          void suprime(Elemento e)
          boolean pertenece(Elemento e)
          Elemento muestrea()
Fin
Constructor  Conjunto.Crear
 inicio
     para cada I = 1 hasta MAX
            v[i] = 0
  fin
```

V

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0

cant=0

2.4.1 Implementacion basada en vectores

Definición básica de la clase Conjunto cuya implementacion es usando vectores

```
Publico Conjunto.inserta(E : Elemento)
Inicio
     si no pertenece(e ) entonces
                  v[E]=1
                  cant = cant +1
Fin


Publico booleano Conjunto.vacio  ()
Inicio
   retornar (cant=0)
Fin
Publico entero Conjunto.cardinal()
Inicio
   retornar cant
fin
```

Conjunto { }

V

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0

cant=0

2.4.1 Implementacion basada en vectores

Definición básica de la clase Conjunto cuya implementacion es usando vectores

```
Publico entero Conjunto.ordinal(E : Elemento)
Inicio
     resp=0
     para cada I = 1 hasta max
         inicio
              si V[I ] <>0 entonces
                              inicio
                                   resp=resp +1
                                   si    E    =    I
      entonces retornar resp
                              fin
         fin
     retornar resp
Fin
```

2.4.2 Implementación con apuntadores

Definición básica de la clase Conjunto implementada usando punteros:

```
Tipo de dato
 Nodo
    dato      Entero,
    Sig       Puntero a Nodo
 // fin definicion

 Direccion Puntero a espacio de memoria de tipo Nodo

Clase Conjunto
 Atributos
    cant          Entero // Numero de elementos
    PtrConj       Direccion
 Metodos
       ……..
```

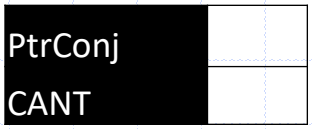
2.4.2 Implementación con apuntadores

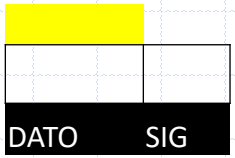
Definición básica de la clase Conjunto cuya implementacion es usando simulacion de memoria

```
Publico entero Conjunto.inserta(Elemento e)
Inicio
     si no pertenece( E ) entonces
            dir=m.new_respacio(´dato,sig')
            si dir <> -1 entonces
               m.ponerdato(dir, '->dato',e)
               m.ponerdato(dir, '->sig', ptrConj)
               ptrConj= dir
               cant = cant +1
            caso contrario
               // error no existe expacio memoria
      caso contrario
           // error elemento ya existe
Fin
```

DIR	DATO	ID	LINK
1			2
2			3
3			4
4			5
5			6
6			7
7			8
8			9
9			10
10			11
11			12
12			13
13			14
14			15
15			-1

LIBRE=1





2.4.2 Implementación con apuntadores

Definición básica de la clase Conjunto cuya implementacion es usando simulacion de memoria

```
Publico booleano Conjunto.vacio  ()
Inicio
   retornar (cant=0) // prtConj = -1
Fin
Publico entero Conjunto.cardinal()
Inicio
   retornar cant
fin
```

2.4.2 Implementación con apuntadores

```
Publico entero Conjunto.ordinal(E : Elemento)
Inicio
     resp=0
     pc = prtConj
     mientras pc<>-1 hacer
     inicio
         resp = resp + 1
         si m.obtenerdato(pc,'->dato')= E
                        entonces
                          retornar resp
                           pc = -1
                        caso contrario
                           pc=m.obtenerdato(pc, '->sig')
     fin
Fin
```

2.4.3 Implementación con lista

Definición básica de la clase Conjunto cuya implementacion es usando listas

```
Clase Conjunto
    Atributos
        Elem : Lista
     Metodos
        ……..
Fin
 Constructor  Conjunto.Crear
  inicio
      elem.crear()
   fin
Publico conjunto.inserta(elemento e)
Inicio
   elem.inserta(elem.primero(),E)
fin
```

Conjunto { }

LISTA < >

2.4.3 Implementación con lista

Definición básica de la clase Conjunto cuya implementacion es usando listas

```
publico boolean  Conjunto.vacio
 inicio
     retornar elem.vacia()
  fin

Publico entero Conjunto.cardinal
  inicio
      retornar elem.longitud()
   fin
```

COMPLETAR IMPLEMENTACION

2.4.4 Implementación de acuerdo a necesidad

- Definición básica de la clase Conjunto implementada usando un entero para reprecentar conjuntos cuyos elementos son¨{1,2,3,4,5,6,7,8,9}

```
Clase Conjunto
 Atributos
    elem   entero
 Metodos
      ……….
constructor conjunto.crear
inicio
   elem=0
fin
publico booleano conjunto.vacio()
inicio
    retornar elem=0
fin
```

Conjunto { }

Elem = 0

2.4.4 Implementación de acuerdo a necesidad

```
Publico entero conjunto.cardinal()
Inicio
   si no vacia()
          entonces
             retornar [log(elem)] +1
Fin

Publico conjunto.insertar(elemento e)
Inicio
    si no pertence(e) entoces
            elem = (elem * 10 ) + e
        caso contrario
          // elemento ya existe
fin
```

Complementación

Implementar el método Mostrar pertenciente a la clase conjunto: Muestra el conjunto en el siguiente formato {e1,e2,e3…en}

PROYECTO CONJUNTO (usando Smemoria)

1) Crear Memoria
2) Pedir Espacio Memoria
3) Crear Conj A
4) Crear Conj B
5) Crear Conj C
6) Insertar Elemento (que pida a que Conj)
7) Mostrar Conjunto (Que indique qu conj)
8) Unir y poner en C el resultado
9) Intersectar y poner en C el resultado
10) Salir

PROYECTO CONJUNTO(Usando Lista)

1) Crear conj A
2) Crear Conj B
3) Crear Conj C
4) Insertar Elemento (que pida a que Conj)
5) Mostrar Conjunto (Que indique qu conj)
6) Unir y poner en C el resultado
7) Intersectar y poner en C el resultado
8) Salir

PROYECTO CONJUNTO(Usando Vector)

1) Crear conj A
2) Crear Conj B
3) Crear Conj C
4) Insertar Elemento (que pida a que Conj)
5) Mostrar Conjunto (Que indique qu conj)
6) Unir y poner en C el resultado
7) Intersectar y poner en C el resultado
8) Salir

PROYECTO CONJUNTO(Usando Punteros)

1) Crear conj A
2) Crear Conj B
3) Crear Conj C
4) Insertar Elemento (que pida a que Conj)
5) Mostrar Conjunto (Que indique qu conj)
6) Unir y poner en C el resultado
7) Intersectar y poner en C el resultado
8) Salir