UNIDAD IV TDA Conjunto

2.1.1 Descripción del TDA Conjunto.

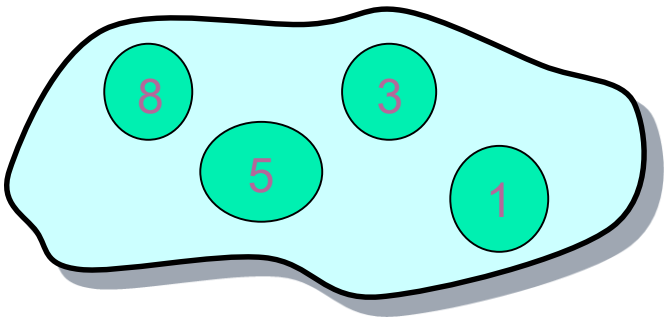
Se define como como conjunto a la agrupación de diferentes elementos que comparten entre sí características y propiedades semejantes. Los elementos de un conjunto pueden ser cualquier cosa, tales como números, canciones, meses, personas, objetos, etcétera.

A la hora de formar un conjunto, la manera y el porqué de como los agrupamos puede variar, dando lugar entonces a los diferentes tipos de conjuntos:

Conjuntos finito. La característica de este conjunto es que sus elementos pueden ser contar o enumerar en su totalidad. Por ejemplo, los meses del año establecen un conjunto finito: enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre y diciembre.
Conjunto infinito. Un conjunto será infinito cuando sus elementos sean imposibles de contar o enumerar en su totalidad, debido a que no tienen fin. Los números son un claro ejemplo de un conjunto infinito.
Conjunto unitario. Aquel que está compuesto por un único elemento. La luna se encuentra dentro de este conjunto, pues es el único satélite natural del planeta tierra.
Conjunto vacío. Se trata de un conjunto el cual no presenta ni tiene elementos.
Conjuntos equivalentes. Serán equivalentes aquellos conjuntos cuya cantidad de elementos sea la misma.
Conjuntos iguales. Podrá decirse que dos o más conjuntos son iguales, cuando estén compuestos por elementos idénticos. eleva

Por lo que los conjuntos contienen elementos no ordenados en el que no existen duplicados, con las operaciones básicas para añadir y eliminar elementos de uno en uno y determinar si un elemento pertenece al conjunto.

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



{8 ,5 ,3, 1}

2.1.2 Especificación del TDA Conjunto.

Partiendo de lo establecido en la Unidad 1. Especificación Informal tenemos lo siguiente:
Elementos que conforman la estructura

TDA Conjunto (**VALORES** son los elementos de cualquier tipo, **OPERACIONES** crea, vacio, pertenece, inserta, suprime, cardinal, muestrea, inserta, suprime)

OPERACIONES

crea (C: conjunto)
Utilidad: Sirve para inicializar el conjunto
Entrada: Conjunto C
Salida: Ninguna
Precondición: Ninguna.
Poscondición: Conjunto inicializado sin términos.

vacio(C:Conjunto) devuelve (booleano)
Utilidad: Verifica si el conjunto es vacio
Entrada: Conjunto C
Salida: Verdadero si esta vacio C, caso contrario falso
Precondición: Ninguna.
Poscondición: Ninguna

Cardinal(C: Conjunto) devuelve (entero)
Utilidad: Cuenta cuantos elementos tiene el conjunto C
Entrada: Conjunto C
Salida: Numero natural
Precondición: Ninguna.
Poscondición: Ninguna

Ordinal(C: Conjunto; E: Elemento) devuelve (entero)
Utilidad: Busca el lugar que ocupa el elemento E en C
Entrada: Conjunto C
Salida: Numero natural
Precondición: Elemento E pertenece a C
Poscondición: Ninguna

Inserta (C:Conjunto, E:Elemento)
Utilidad: Añade el element E al conjunto C
Entrada: Conjunto C
Salida: Numero natural
Precondición: Elemento E no pertenece a C
Poscondición: Conjunto C modificado

Suprime (C: Conjunto, E:Elemento)
Utilidad: Elimina E del conjunto C
Entrada: Conjunto C
Salida: Ninguna
Precondición: Elemento E pertenece a C
Poscondición: Modifica el Conjunto C

Pertenece (C: Conjunto;E Elemento) devuelve (Boleano)
Utilidad: Verifica si elemento E pertence al conjunto C
Entrada: Conjunto C
Salida: Verdadero o Falso

Precondición: : Conjunto no vacío
Poscondición: Ninguna

Muestrea(C:conjunto) devuelve (elemento)
Utilidad: Busca un elemento E al azar que pertenece al conjunto C
Entrada: Conjunto C
Salida: Elemento
Precondición: : Conjunto no vacío
Poscondición: Ninguna

2.1.3 Aplicaciones con Conjunto.

En esta sección se puede apreciar que ya estamos en condiciones para plantear algoritmos usando el TDA Conjunto, abstrayéndonos de la forma como esta implementado.

Ej1: Implementar un algoritmo que Intersecte los conjuntos A,B y coloque el resultado a retornar en C, considerando que A,B,C son parámetros de algoritmo.

```
Unión( Conjunto A,B, ES Conjunto 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
```

Ej2: Implementar un algoritmo que determine si dos conjuntos son equivalentes, por lo que el algoritmo deberá tener como parámetros A, B de tipo conjunto.

```
Boolean Equivalente(Conjunto A,B)
Inicio
  Retornar (a.cardinal = b.cardinal)
fin
```

2.1.4.1 Implementación con Listas.

Para la implementación del TDA Conjunto se utilizara una lista ELEM que contendrá los elementos del conjunto, se hace notar que la cantidad de elementos estará dada por la longitud de la Lista.

Definiendo la clase Conjunto

```
Tipo de Datos
        Clase Conjunto
          Atributos
                Elem : Lista
          Metodos
                Crear()
                booleano vacío ()
                booleano pertenece (Elemento e)
                inserta (Elemento e)
                entero cardinal ()
                entero ordinal (elemento e)
                suprime (Elemento e)
                Elemento muestrea ()
           fin definición clase
```

```
Conjunto.Crear
inicio
   elem.crear()
 fin
```

https://youtu.be/c04RswgL5ws

```
booleano  Conjunto.vacio
inicio
   retornar (elem.longitud=0)
 fin
```

https://youtu.be/imVWhYKsbTU

```
booleano conjunto.pertenece( elemento e)
inicio
     si  no (elem.vacia())
     entonces
     inicio
        p= elem.primero()
        mientras p<> Nulo
          inicio
          ele=elem.recupera(p)
          si e= ele entonces retornar VERDADERO
          p = elem.siguiente(p)
         fin
       fin
   caso contrario
     // exception error conjunto no tiene elementos
  fin
```

https://youtu.be/q6aAO1vU6Ug

Publico conjunto.inserta(elemento e)

```
Inicio
  Si no pertenece( e  ) entonces
       elem.inserta(elem.primero(),e)
fin
```
https://youtu.be/CktQF3en_K4

entero Conjunto.cardinal()

```
 inicio
    retornar elem.longitud()
  fin
```
https://youtu.be/ldXeh0ImxHg

numero conjunto.ordinal(elemento e)

```
 inicio
      si  no elem.vacia()
      entonces
      inicio
         cont=0
         p= elem.primero()
         mientras p<> Nulo
           inicio
           cont = cont +1
           ele=elem.recupera(p)
           si e= ele entonces retornar cont
           p = elem.siguiente(p)
          fin
       fin
    caso contrario
     // exception error conjunto no tiene elementos
   Fin
```
https://youtu.be/QzU7-bWa_e4

Conjunto.suprime(elemento e)

```
inicio
       si  pertenece( e )
      entonces
      inicio
         dir = nulo
         p= elem.primero()
         mientras p<> Nulo
           inicio
              ele=elem.recupera(p)
              si e= ele entonces retornar dir=p
              p = elem.siguiente(p)
           fin mientras
          si dir<>nulo entoces elem.suprime(dir)
       fin
    caso contrario
     // exception error conjunto no  dicho elemento
 fin
```

https://youtu.be/6eptpFk2Ce0

elemento conjunto.muestrea()

inicio

```
lug = // número al azar( >=1 y <= elem.longitud)
dir = elem.primero
direlem=nulo
cont =0
para cada i=1 hasta elem.longitud
    cont= cont +1
    si cont=lug entonces dirElem=dir
    dir=elem.siguiente(dir)
fin para
retornar elem.recupera(dirElem)
```

fin

https://youtu.be/kDOpz3IX-o4

2.1.4.2 Implementación con Vector.

Para la implementación del TDA Conjunto se utilizara un vector V y un Atributo denominado cant, donde el vector será el que contendrá los elementos del conjunto, se hace notar que cant determinara el número de elementos que contiene el conjunto.

Definiendo la Conjunto

Constante max = 100

Definiendo la Conjunto

Tipo de Datos

Clase Conjunto
Atributos
V Arreglo(MAX) // Elementos
cant entero // Determina la cantidad de elementos
Metodos
Crear()
booleano vacío ()
booleano pertenece (Elemento e)
inserta (Elemento e)
entero cardinal ()
entero ordinal (elemento e)
suprime (Elemento e)
Elemento muestrea ()
fin definición clase

Conjunto.Crear

inicio
para cada I = 1 hasta MAX
v[i] = 0
fin para

fin

https://youtu.be/_ows9BhsLFs

Conjunto.inserta(E : Elemento)
Inicio
si no pertenece (e) entonces
v[E]=1
cant = cant +1
Fin
https://youtu.be/H9UkMIGc7l0

Conjunto.suprime(E : Elemento)
Inicio
si pertenece (e) entonces
v[E]=0
cant = cant - 1
Fin
https://youtu.be/iBpfnI8VsKo

booleano Conjunto.vacio ()
Inicio
retornar (cant=0)
Fin
https://youtu.be/BTQ2qbkDfLM

entero Conjunto.cardinal()
Inicio
retornar cant
fin
https://youtu.be/T_YdxmnnG1g

booelano conjunto.pertenece(E : Elemento)
inicio
retornar (v[e]=1)
fin
https://youtu.be/-GmuD9NnSnU
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
https://youtu.be/H5eLCqX1wnk

elemento Conjunto.muestrea()

```
Inicio
   resp=0     elemento=0
   lug = // numero al azar >=1 y <= cant
   para cada I = 1 hasta max
      inicio
         si V[I ] <>0 entonces
                  inicio
                     resp=resp +1
                     si resp = lug  entonces elemento = v[i]
                  fin
      fin
   retornar elemento
Fin
```

https://youtu.be/RPXCXK3G7MY

2.1.4.3 Implementación con Simulación de Memoria (usando la clase CSMemoria)

Esta forma de implementación es netamente académica en virtud a que lo que busca es una mejor comprensión sobre los punteros, para ello se entiende que se usara como Memoria nuestra clase CSmemoria implementada en la unidad uno.

Definiendo la clase Conjunto
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
            Crear()
            booleano vacío ()
            booleano pertenece (Elemento e)
            inserta (Elemento e)
            entero cardinal ()
            entero ordinal (elemento e)
            suprime (Elemento e)
            Elemento muestrea ()
fin definición clase
```

Implementación clase conjunto utilizando Simulador de Memoria CSmemoria.

Conjunto.Crear()

```
Inicio
     Cant=0
```

```
      ptrConj = Nulo
fin
```
https://youtu.be/sTxglnp6s14

booleano Conjunto.vacio ()

```
Inicio
  retornar (cant=0) // prtConj = -1
Fin
```
https://youtu.be/5uAX964hVbo

Publico entero Conjunto.pertenece(E : Elemento)

```
Inicio
    resp=false
    pc = prtConj
    mientras pc<>Nulo hacer
          si m.obtenerdato(pc,'->dato')= E
                entonces
                  resp=true
                  pc = Nulo
                caso contrario
                  pc=m.obtenerdato(pc, '->sig')
    fin mientras
   retornar resp
Fin
```
https://youtu.be/KmOMUn3cTSw

Conjunto.inserta(Elemento e)

```
Inicio
    si no pertenece( E ) entonces
          dir=m.new_respacio(´dato,sig')
          si dir <> Nulo 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
```
https://youtu.be/F1Z8syYtdEY

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
```
https://youtu.be/yS3adTkZmAs

entero Conjunto.cardinal()

```
Inicio
  retornar cant
fin
```
https://youtu.be/RL3nDJ0Gtvs

Conjunto.Suprime(E : Elemento)

```
Inicio
   Dir=Nulo
   pc = prtConj
   mientras pc<>Nulo hacer
   inicio
     si m.obtenerdato(pc,'->dato')= E
            entonces
              Dir=pc
              pc = Nulo
            caso contrario
              pc=m.obtenerdato(pc, '->sig')
   fin
   m.delete_espacio(Dir)
  Fin
```
https://youtu.be/8HQXbL7AP-w

Elemento Conjunto.Muestrea()

```
Inicio
   Si Cant>0 entonces
   Lugar=0
   lugarBuscado= // numero al azar >=1 y <= cant
   pc = prtConj
   mientras pc<>-1 hacer
     Lugar = Lugar + 1
     si lugarBuscado=Lugar
            entonces
```

```
                Elemento=m.obtenerdato(pc,'->dato')
                pc = -1
            caso contrario
              pc=m.obtenerdato(pc, '->sig')
  fin mientras
  retornar elemento
caso contrario
      caso contrario // exception error no existe elemento
Fin
```

https://youtu.be/fp9iZNPyU8M

2.1.4.4 Implementación con punteros.

En esta forma de implementación planteada lo que se resalta son los cambios que tienen que hacerse al código de la implementación con el simulador de memoria considerando que ahora se está trabajando con punteros reales, es así que se tiene de color rojo los cambios fundamentales en los algoritmos ya vistos quedando por resolver las definiciones formales en c++.

Definiendo la clase Conjunto

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
Crear()
booleano vacío ()
booleano pertenece (Elemento e)
inserta (Elemento e)
entero cardinal ()
entero ordinal (elemento e)
suprime (Elemento e)
Elemento muestrea ()
fin definición clase

Implementación clase conjunto utilizando Simulador de Memoria CSmemoria.

Conjunto.Crear()

```
Inicio
    Cant=0
    ptrConj = Nulo
fin
```

booleano Conjunto.vacio ()

```
Inicio
```

```
  retornar (cant=0) // prtConj =Nulo
Fin

Publico entero Conjunto.pertenece(E : Elemento)
Inicio
    resp=false
    pc = prtConj
    mientras pc<>Nulo hacer
          si pc→dato=E
                entonces
                  resp=true
                  pc = Nulo
                caso contrario
                  pc=pc→sig
    fin mientras
   retornar resp
Fin

entero Conjunto.cardinal()
Inicio
  retornar cant
fin

Conjunto.inserta(Elemento e)
Inicio
    si no pertenece( E ) entonces
         dir= new Nodo
         si dir <> Nulo entonces
            dir→dato =e
           dir→sig=ptrconj
           ptrConj= dir
           cant = cant +1
         caso contrario
           // error no existe expacio memoria
     caso contrario
        // error elemento ya existe
Fin

entero Conjunto.ordinal(E : Elemento)
Inicio
    resp=0
    pc = prtConj
    mientras pc<>Nulo hacer
    inicio
       resp = resp + 1
       si pc→dato = E
                entonces
                 retornar resp
```

```
                pc = Nulo
              caso contrario
                pc=pc→sig
  fin
Fin
```

Conjunto.Suprime(E : Elemento)

```
Inicio
  Dir=Nulo
  pc = prtConj
  mientras pc<>Nulo hacer
  inicio
    si pc→dato=E  = E
              entonces
                Dir=pc
                pc = Nulo
              caso contrario
                pc= pc→sig
  fin
  delete Dir
 Fin
```

Elemento Conjunto.Muestrea()

```
Inicio
  Si Cant>0 entonces
  Lugar=0
  lugarBuscado= // numero al azar >=1 y <= cant
  pc = prtConj
  mientras pc<> Nulo hacer
    Lugar = Lugar + 1
    si lugarBuscado=Lugar
              entonces
                 Elemento= pc→dato
                 pc = Nulo
              caso contrario
                pc=pc→sig
  fin mientras
  retornar elemento
 caso contrario
        caso contrario // exception error no existe elemento
fin
```

LABORATORIO

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