Bloque III: Estructuras de datos no lineales

ESTRUCTURAS DE DATOS

Tema 6: Conjuntos

ESTRUCTURAS DE DATOS

Conjuntos: contenidos

- Introducción
- Especificación algebraica del TAD TipoConjunto
- Implementaciones del TAD TipoConjunto
- Programación con conjuntos
- Bolsas (conjuntos con repetición)

Conjuntos: Introducción

- Es una colección no ordenada de elementos del mismo tipo, donde no hay repeticiones.
- Cardinalidad
- Conjunto vacío
- Operaciones: pertenencia, inclusión, intersección, unión y diferencia
- Conjuntos en Pascal

Especificación

```
ESPECIFICACIÓN Conjunto

PARÁMETROS GENÉRICOS
   TIPOS TipoElemento
FIN PARÁMETROS

TIPOS TipoConjunto

OPERACIONES
   (* constructoras generadoras *)
      CrearConjuntoVacio: → TipoConjunto
      Poner: TipoConjunto x TipoElemento → TipoConjunto

   (* observadoras selectoras *)
      PARCIAL Elegir: TipoConjunto → TipoElemento

   (* observadoras no selectoras *)
      EsConjuntoVacio: TipoConjunto → Booleano
      Pertenece: TipoConjunto x TipoElemento → Booleano
      EsSubconjunto: TipoConjunto x TipoConjunto → Booleano
      Cardinal: TipoConjunto → Natural
```

Especificación

```
(* constructoras no generadoras *)
    Quitar: TipoConjunto x TipoElemento → TipoConjunto
    Union: TipoConjunto x TipoConjunto → TipoConjunto
    Interseccion: TipoConjunto x TipoConjunto → TipoConjunto
    Diferencia: TipoConjunto x TipoConjunto → TipoConjunto

VARIABLES
   conjunto, conjunto2: TipoConjunto;
   i, j, e1, e2: TipoElemento;

ECUACIONES DE DEFINITUD
   DEF(Elegir(Poner(conjunto, i)))

ECUACIONES ENTRE GENERADORAS
(* la operación 'Poner' es conmutativa *)
Poner(Poner(conjunto, e1), e2) = Poner(Poner(conjunto, e2), e1)

(* la operación 'Poner' es idempotente *)
Poner(Poner(conjunto, e1), e1) = Poner(conjunto, e1)
```

Especificación

```
ECUACIONES
(* observadoras selectoras *)
   Elegir(Poner(conjunto, i)) =
<< seleccion aleateoria de un elemento j tal que: Pertenece(conjunto, j) >>

(* observadoras no selectoras *)
   EsConjuntoVacio(CrearConjuntoVacio)= CIERTO
   EsConjuntovacio(Poner(conjunto, i))= FALSO

   Pertenece(CrearConjuntoVacio, i) = FALSO
   Pertenece(Poner(conjunto, i), j) = (i = j) O Pertenece(conjunto, j)

   EsSubconjunto(CrearConjuntoVacio, conjunto2) = CIERTO
   EsSubconjunto(Poner(conjunto, e1), conjunto2) = Pertenece(conjunto2, e1) Y
      EsSubconjunto(conjunto, conjunto2)

   Cardinal(CrearConjuntoVacio) = 0
   Cardinal(Poner(conjunto, e1)) = SI Pertenece(conjunto, e1) →
      Cardinal(conjunto)
   | 1 + Cardinal(conjunto)
```

Especificación

```
(* constructoras no generadoras *)
Quitar(CrearConjuntoVacio,j)= CrearConjuntoVacio
Quitar(Poner(conjunto, i), j) = SI i = j →
   Quitar(conjunto, j)
| Poner(Quitar(conjunto, j), i)

Union(CrearConjuntoVacio, conjunto2) = conjunto2
Union(Poner(conjunto, i), conjunto2) = Poner(Union(conjunto, conjunto2), i)

Interseccion(CrearConjuntoVacio, conjunto2)= CrearConjuntoVacio
Interseccion(Poner(conjunto, i), conjunto2) = SI Pertenece(i, conjunto2) →
   Poner(Interseccion(conjunto, conjunto2), i)
| Interseccion(conjunto, conjunto2)

Diferencia(conjunto, CrearConjuntoVacio) = conjunto
Diferencia(conjunto, Poner(i, conjunto2))=
   Diferencia(Quitar(conjunto, i), conjunto2)
```

FIN ESPECIFICACIÓN

Implementaciones

- Tipos predefinidos del lenguaje (SET en Pascal*)
- Mediante vectores booleanos
- Mediante un vector de elementos
- Realización dinámica
- Comparación de las diversas realizaciones
- Observaciones Generales

* `set` en la STL o C++Std Library sí tiene orden y se implementa mediante ABB

Comparativa de implementaciones

- SET en Pascal: la mas inmediata y eficiente pero está limitada a conjuntos de 256 elementos.
- Vector de booleanos: permite trabajar con cualquier tamaño y su implementación es directa pero requiere un vector del tamaño del conjunto universal.
- Si el rango del conjunto universal es muy grande, deberemos utilizar implementaciones con vector de elementos o con listas, en este caso las operaciones no son tan eficientes, pero se ahorra espacio.

Consideraciones generales

- En las operaciones de unión, intersección y diferencia se ha considerado que los conjuntos tienen el mismo tamaño.
- Hemos considerados que los conjuntos están formados por enteros y que sus valores van desde 1 hasta NUM_MAX

Programación con conjuntos

- Escribir los elementos de un conjunto
- Encontrar el mayor de los factores primos comunes de tres números menores que 100
- Juego del Bingo
- Criba de Eratóstenes para generar los primos menores que un número dado.
- Clasificación de caracteres

Bolsas

- Una bolsa es una colección formada por un número arbitrario de elementos del mismo tipo, donde no hay orden, pero que puede existir repetición de elementos.
- Surge el concepto de multiplicidad de un elemento

Implementaciones

- Las operaciones sobre bolsas son las mismas que sobre conjuntos salvo que aparece una nueva operación:
  - `Multiplicidad: TipoElemento x TipoBolsa → Entero`

- Las implementaciones son las mismas pero ahora se considera la posibilidad de repeticiones.

Ejercicio propuesto

- Dada la realización del TAD `TipoConjunto`, adaptarla para la representación y manipulación de bolsas (TAD `TipoBolsa`)
- Usando el TAD `TipoBolsa`, especificar y codificar en Pascal un algoritmo que permita calcular el mínimo común múltiplo de tres números naturales.