Tipos Abstractos de Datos

5. Introducción a los TDA
6. El TDA Lista
7. El TDA Pila
8. El TDA Cola

5. Introducción al concepto de TDA

1. Ventajas de la programación modular
2. El concepto de Tipo de Dato Abstracto
3. Ejemplos de TDA
   - 3.1. TDA conjunto
   - 3.2. TDA matriz

Bibliografía:

- Weiss
- Aho, Hopcroft, Ullman

5.1. Ventajas de la programación modular

Programación modular:

- realización de un programa mediante módulos,
- cada módulo resuelve un sub-problema del programa

Ventajas del enfoque modular:

- permite desarrollo independiente de módulos (en grupo),
- facilita la depuración,
- encapsulación

Hasta ahora:

programación módular = uso de algoritmos o procedimientos independientes que manipulan tipos de datos propios o predefinidos

Los algoritmos:

- generalizan operadores sobre datos (no elementales),
- facilitan la encapsulación

5.2. El concepto de Tipo de Dato Abstracto (TDA)

- Modelo matemático (tipo de dato complejo) + conjunto de operadores del modelo
- Son diferentes:
  - tipo de dato básico (operadores+ valores),
  - dato estructurado o compuesto,
  - tipo de dato abstracto
- TDA generaliza los tipos de datos
- Facilita la encapsulación / depuración

UNA VEZ DEFINIDO EL TDA:

- **sólo se puede acceder a los objetos del TDA mediante sus operadores,**
- **ya no es necesario conocer cómo se implementan,**
- **no debe hacerse referencia a los detalles de implementación**

5.3. Ejemplos de TDA

5.3.1. TDA conjunto

Colección de elementos distintos, definidos dentro de un cierto Dominio

Operaciones del TDA Conjunto

Deben definirse en función de la aplicación.

- **Crear** (C: Conjunto, OK: lógico)
- **Borrar** (C: Conjunto, OK: lógico)
- **Unión** (C1: Conjunto, C2: Conjunto, C3: Conjunto, OK: lógico)
- **Intersección** (C1: Conjunto, C2: Conjunto, C3: Conjunto, OK: lógico)
- **Diferencia_positiva** (C1: Conjunto, C2: Conjunto, C3: Conjunto, OK: lógico)
- **Pertenece** (E: T, C1: Conjunto, resp: lógico)

Estas operaciones son independientes de la implementación.

Posible implementación

{Tipo}
Conjunto = **registro** de
    tamaño: numérico;
    nombres: vector[1..N] de cadena;
    elementos: vector[1..N] de lógico;
fin_registro

5.3.2. TDA matriz

Estructura de más de dos dimensiones de elementos de un tipo base, por ejemplo, matriz de 2 dimensiones de numérico.

Operaciones del TDA Matriz

- **Crear** (M: Matriz, Filas: numérico, Columnas: numérico, OK: lógico)
- **Borrar**(M: Matriz, OK: lógico)
- **Sumar** (M1: Matriz, M2: Matriz, M3: Matriz, OK: lógico)
- **Restar** (M1: Matriz, M2: Matriz, M3: Matriz, OK: lógico)
- **Multiplicar** (M1: Matriz, M2: Matriz, M3: Matriz, OK: lógico)
- **División** (M1: Matriz, M2: Matriz, M3: Matriz, OK: lógico)
- **Determinante** (M: Matriz, d: numérico, ok: lógico)
- **Compatibles?** (M1: Matriz, M2: Matriz, operación: Carácter, Ok: lógico)

Posible implementación

{Tipo}
  Matriz = registro de
    num_fil, num_col: numérico;
    datos: Vector [1..M, 1..N] de T; {num_fil <= M, num_col <= N}
  fin_registro

Cada una de las operaciones suele llevar una cabecera donde se especifica:
- condiciones para utilizar el operador,
- efecto/s del operador (tanto sobre los parámetros como sobre el entorno)

5.3.3 Otros ejemplos

Enteros grandes (que no se puedan representar en un tipo de dato entero)