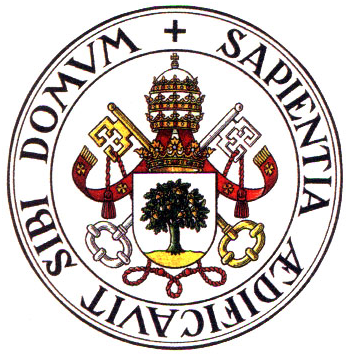

Universidad de **Valladolid**

Departamento de informática

Campus de Segovia

Estructura de datos

Tema 2:
Tipos Abstractos de Datos (TADs)

Prof. Montserrat Serrano Montero

ÍNDICE

- Tipos de datos
- El tipo abstracto de datos (TAD)
- Metodología de la programación de un TAD
- Desarrollo de programas con TADs.

TIPOS DE DATOS

- Un tipo de datos es una colección de valores.
- Utilizando distintos criterios, podemos clasificar los tipos de datos de Pascal:
  - Según si modifican su **valor en ejecución**: constantes (const) y variables (var).
  - Según su **composición**:
    1. Elementales:
    a) *Predefinidos*: enteros (integer), lógicos (boolean), caracteres (char), reales (real).
    b) *Definidos por el usuario*: enumerados y subrangos.
    2. Estructurados:
    a) *Predefinidos:* matrices (array), cadena de caracteres (string), registros (record), ficheros (file), ficheros de texto (text), conjuntos (set), punteros (pointer) y objetos (object).
    b) *Definidos por el usuario:* procedimientos (procedure, function) y *otros* (TADs).

TIPOS DE DATOS

- Según la **naturaleza de los valores**: numéricos (integer, real) y caracteres (char, string).

- Según su **almacenamiento en memoria** (¿Cómo? y ¿Dónde?): estáticos (simples y estructurados) y dinámicos (punteros, variables dinámicas).

- Según su **ordenación**: ordinales (integer, boolean, char, enumerados, subrango) y escalares (integer, boolean, char, real)

TIPOS DE DATOS

CRITERIO	TIPO DATO	PASCAL
Valor en ejecución	Constantes	Const
	Variables	Var
Naturaleza de los valores	Numéricos	Integer, Real
	Caracteres	Char, String
Almacenamiento memoria	Estáticos	Simples, Estruct
	Dinámicos	Pointer, var dina
Ordenación	Escalares	Int, boo, char, re
	Ordinales:Pred,Succ	I, c, b, e, sub
Composición	Elementales: predefinidos	Integer
		Boolean
		Char
		Real
	Elementales: definido por usuario	Enumerados
		Subrangos
	Estructurados	Array
		String
		Record
		File
		Text
		Set
		Object
		Pointer
		Procedure,Func.

TIPOS DE DATOS ABSTRACTOS

- Una abstracción es la simplificación de un objeto o de un proceso de la realidad en la que sólo se consideran los aspectos más relevantes.

- La abstracción se utiliza por los programadores para dar sencillez de expresión al algoritmo.

- La abstracción tiene dos puntos de vista en programación:
    1. Funcional
    2. De datos

TIPOS DE ABSTRACCIÓN

- **La abstracción funcional:**

  - Permite dotar a la aplicación de operaciones que no están definidas en el lenguaje en el que se está trabajando.

  - Se corresponden con el mecanismo del subprograma (acción que se realiza y argumentos a través de los cuales toma información y devuelve resultados).

  - Es irrelevante cómo realiza la acción y no importa su tiempo de ejecución.

- **Las abstracciones de datos (= Clase):**

  - Permiten utilizar nuevos tipos de datos que se definirán especificando sus posibles valores y las operaciones que los manipulan.

  - Cada operación constituye una abstracción funcional.

DEFINICIÓN

- Un tipo abstracto de datos (TAD) es un tipo definido por el usuario que:

  - Tiene un conjunto de valores y un conjunto de operaciones.

  - Cumple con los principios de abstracción, ocultación de la información y se puede manejar sin conocer la representación interna.

- Es decir, los TADs ponen a disposición del programador un conjunto de objetos junto con sus operaciones básicas que son independientes de la implementación elegida.

PREGUNTA

- ¿Los tipos de datos básicos son TDA?

- Definición de los tipos de datos predefinidos como TDA:

  - El TDA entero tiene como posibles valores de los datos el conjunto $\{-1, -2, ..., \infty\} \cup \{0, 1, 2, ..., \infty\}$, y como operaciones la suma, la resta, la multiplicación y la división entera.
  - El TDA real tiene como tipo el conjunto de los puntos de la recta real (Z+Q+I) y como operaciones la suma, la resta, la multiplicación y la división.
  - El TDA Booleano tiene como conjunto de valores {True, False} y como operaciones las definidas por el álgebra de Boole (AND, OR, NOT).
  - El TDA carácter tiene como tipo el conjunto de caracteres definido por un alfabeto dado y como operaciones todos los operadores relacionales ($<$, $>$, $=$, $<>$, $\geq$, $\leq$).

TD Y TDA

- Definición de algunos tipos de datos estructurados como TDA:

  - El TDA matriz es una colección homogénea de longitud fija tal que cada una de sus componentes pueden ser accedidas individualmente mediante uno o varios índices, que serán de tipo ordinal y que indican la posición de la componente dentro de la colección. Las operaciones asociadas: AsignarElemento, Sumar, Restar, Negar, ProductoEscalar, ProductoMatricial, Determinante, Inversa y Transponer.
  - El TDA registro es un tipo de datos heterogéneo compuesto por un número fijo de componentes denominadas campos a las que se accede mediante un selector de campo.

OBSERVACIONES

- Observar el anidamiento existente entre los TDA. Los registros pueden tener como campos tipos de datos simples, arrays o incluso otros registros.

- Algunos operadores declarados se podrían realizar utilizando un subconjunto del resto de los operadores. Ej: SuprimirDato podría realizarse con el operador Localizar y seguidamente con el Suprimir. Sin embargo, se incorpora a los operadores según la frecuencia de utilización y debido a que su implementación particular mejorará el coste de la misma. Esta estrategia no es nueva, en los enteros se define el operador multiplicación, pese a que podría realizarse por iteración del operador suma. Sin embargo, la multiplicación es una operación muy frecuente y existen implementaciones específicas que mejoran la suma iterada.

METODOLOGÍA DE LA PROGRAMACIÓN DE UN TAD

- Debido al proceso de abstracción, la programación de un TAD deberá realizarse siguiendo tres pasos fundamentales:

  1. Análisis de datos y operaciones.

  2. Especificación del tipo de datos abstracto.

  3. Implementación.

ESPECIFICACIÓN DE UN TAD

- Describe el comportamiento del TAD.

- El lenguaje idóneo para poder definir adecuadamente las especificaciones de un TAD es el matemático.

- Hay dos formas de realizar la especificación de un TAD: informal y formal.

- En la especificación informal intervienen cuatro componentes:
  1. Un conjunto de objetos.
  2. La declaración de los encabezamientos de las operaciones.
  3. La descripción de cada operación bajo el epígrafe de efectos.
  4. Las excepciones que pueden producirse en cada operación (si se producen).

ESPECIFICACIÓN DE UN TAD

- En la especificación formal intervienen tres componentes:

  1. Un conjunto de objetos.

  2. Un conjunto de descripciones sintácticas de operaciones: declaración de encabezamientos de operaciones con sus tipos de argumentos de entrada y de salida.

  3. Un descripción semántica, esto es, un conjunto suficientemente completo de relaciones que especifiquen el funcionamiento de las operaciones para describir su comportamiento sin ningún tipo de ambigüedad.

ESPECIFICCIÓN DE TADs

- Descripción de los dos tipos:

1. Informal

TAD Nombre del tipo de datos (VALORES: valores que toman los datos del tipo; OPERACIONES: nombre de las operaciones que los manipulan)

Nombre de operación (tipo de argumento) → tipo de resultado

Efecto: Descripción de la operación

Excepción: Posibles excepciones

2. Formal

TAD Nombre del tipo de datos (VALORES: valores que toman los datos del tipo; OPERACIONES: nombre de las operaciones que los manipulan)

Sintaxis: {Forma de las operaciones}

Nombre de operación (tipo de argumento) → tipo de resultado

Semántica: {Significado de las operaciones.- Expresión que refleja, para unos determinados argumentos, el resultado que se puede obtener}

Nombre de operación (valores particulares) ⇒Expresión resul

CATEGORIAS DE LAS OPERACIONES APLICABLES A UN TAD

- **Operaciones de creación**: (constructoras o primitivas). Obtienen nuevos objetos del tipo (crearconj). También se suele incluir alguna operación de lectura.

- **Operaciones de consulta:** Realizan funciones tal que empleando un argumento del tipo devuelven un valor de otro tipo (pertenece, estarvacioconj) o tareas frecuentes (Escribirconj).

- **Operaciones de modificación:** Permiten modificar un objeto del TAD (añadir y eliminar)

- **Operaciones propias del tipo:** Operaciones propias del tipo que se emplean en las anteriores operaciones descritas.

CARACTERÍSTICAS DE LA ESPECIFICACIÓN

- También se le llama Interface. Responde al QUÉ compone un TAD.

- Debe ser precisa. No debe dar información redundante o inservible.

- Debe ser lo más general posible.

- Debe ser legible.

- Debe evitar ambigüedades.

IMPLEMENTACIÓN DE UN TAD

- Para la implementación del TAD es fundamental disponer de mecanismos que permitan ENCAPSULAR, el tipo de dato y sus operaciones, y OCULTAR la información al programador usuario.

- Estas dos características son fundamentales para poder llevar a cabo esta abstracción.

  - Encapsulación: Ocultar los detalles de implementación del comportamiento de una clase. Permite implementar cambios sin afectar a otros componentes del programa. Implica reutilización.
  - Ocultación: Restringir el acceso a la información del estado de un objeto. Esta restricción afecta al acceso a los atributos y a algunas de las operaciones del objeto. Implica no manipulación y por tanto un mejor mantenimiento del sistema.

IMPLEMENTACIÓN DE UN TAD

- Turbo Pascal 7.0 provee de **unidades** que permiten, en cierta medida, cumplir con ambas premisas.

- Modularización:

  - Un módulo es una unidad de programa donde están implementados una colección de recursos que pueden ser utilizados por uno o más programas.

  - Cuando un programa necesita un recurso importa un módulo.

  - Dependiendo del lenguaje se importa toda la biblioteca o recursos sueltos.

  - Las bibliotecas no son programas, por lo que no pueden ser ejecutadas independientemente.

  - Los módulos se compilan por separado y se caracterizan por su interfaz y su implementación.

RESUMEN UNIDADES

UNIT <identificador>; {cabecera obligatoria}

{El identificador debe ser utilizado como nombre de archivo}

INTERFACE

USES <lista de unidades>; {opcional}

CONST.....

TYPE.....

VAR.....

PROCEDURE..... {sólo cabecera}

FUNCTION..... {sólo cabecera}

IMPLEMENTATION

{Declaraciones privadas: locales a la unidad}

USES <lista de unidades>; {opcional}

CONST.....

TYPE.....

VAR.....

PROCEDURE..... {cabecera y cuerpo}

FUNCTION..... {cabecera y cuerpo}

BEGIN {Selección de inicialización: opcional}

.....

END. {Obligatorio, fin de implementación}

DESARROLLO DE PROGRAMAS CON TADs

- El primer paso a seguir es definir un TAD (colección de objetos junto con el conjunto de operaciones definidas para dichos objetos).

- Una vez definido el TAD y sin necesidad de conocer cómo ha sido implementado, ya se puede utilizar de una forma abstracta, siendo suficiente la información suministrada por su definición.

ETAPAS EN LA ELABORACIÓN DE UN TAD

- Reconocimiento de los objetos candidatos a elementos del nuevo tipo de datos
- Identificación de las operaciones del tipo de datos.
- Especificación de las operaciones de forma precisa, sin ambigüedad.
- Selección de una buena implementación

CONJUNTO DE OBJETOS

- Tipo Conjunto:

  Colección no ordenada de elementos 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.

ESPECIFICCIÓN DE LAS OPERACIONES

1. Informal

TAD Conjunto (VALORES: conjunto de elementos sin repetición; OPERACIONES: ConjuntoVacío, EsVacío, Añadir, Borrar, Pertenece)

ConjuntoVacío → Conjunto

Efecto: Crea un conjunto vacío.

Añadir (Conjunto, Elemento) → Conjunto

Efecto: Añade a un conjunto un elemento si no lo tiene ya y no lo añade si ya está en el conjunto.

Excepción: Error al llegar al tamaño máximo del conjunto.

Borrar (Conjunto, Elemento) → Conjunto

Efecto: Elimina de un conjunto el elemento si está en él y no hace nada en otro caso.

EsVacío (Conjunto) → Boolean

Efecto: Informa sobre si el conjunto está o no vacío.

Pertenece (Conjunto, Elemento) → Boolean

Efecto: Indica si un elemento está o no en el conjunto.

ESPECIFICCIÓN DE LAS OPERACIONES

1. Formal (* Constructores)

TAD Conjunto (VALORES: conjunto de elementos sin repetición; OPERACIONES: ConjuntoVacío, EsVacío, Añadir, Borrar, Pertenece)

Sintaxis:

*ConjuntoVacio	→	Conjunto
*Añadir (Conjunto, Elemento)	→	Conjunto
Borrar (Conjunto, Elemento)	→	Conjunto
EsVacio (Conjunto)	→	Boolean
Pertenece (Conjunto, Elemento)	→	Boolean

Semántica: para todo e, e1: elemento y todo C de tipo conjunto

Añadir (Añadir (C, e), e)	⇒	Añadir (C, e)
Añadir (Añadir (C, e), e1)	⇒	Añadir (Añadir (C, e1), e)
Borrar (ConjuntoVacio, e)	⇒	ConjuntoVacío
Borrar (Añadir (C, e1), e)	⇒	**si** Igual (e, e1) **entonces** Borrar (C, e) **si-no** Añadir (Borrar (C, e),e1)
Pertenece (ConjuntoVacio, e)	⇒	False
Pertenece (Añadir (C, e1), e)	⇒	**si** Igual (e, e1) **entonces** True **si-no** Pertenece (C, e)
EsVacio (ConjuntoVacio)	⇒	True
EsVacio (Añadir (C, e))	⇒	False

IMPLEMENTACIÓN DEL TAD

```
unit TADconj;
  interface
    uses tadelem;
    type
     conjunto= record
       numelem: 0..100;
       L: array[1..100] of elemento {definido en la unidad
     tadelem}
     end;
    procedure ConjuntoVacio (var c: conjunto);
    function Pertenece (c: conjunto; e: elemento): boolean;
    procedure Añadir (var c: conjunto; e: elemento);
    function EsVacio (c: conjunto): boolean;
    procedure Borrar (var c: conjunto; e: elemento);
    var
     errorconj: boolean;
  implementation
    procedure ConjuntoVacio;
     begin
       c.numelem := 0;
     end;
```

IMPLEMENTACIÓN DEL TAD

```
{implementation ... Continúa}
 function EsVacio;
  begin
   EsVacio := c.numelemen = 0
  end;
 procedure Dicotomi (var c: conjunto; e:elemento; var p:
  integer; var encontrado: boolean);
  var
   izq, der, central: integer;
  begin
   izq := 1; der := c.numelem;
   while izq <= der do
    begin
     central := (izq + der) div 2;
     if Mayor (e, c.L[central]) then
      izq := central + 1
     else
      der := central –1
    end;
   p := izq; encontrado := Igual (e, c.L[p])
  end;
```

IMPLEMENTACIÓN DEL TAD

```
{implementation ... Continúa}
 function Pertenece;
  var
   encontrado: boolean; p: integer;
  begin
   Dicotomi (c, e, p, encontrado);
   Pertenece := encontrado
  end;
 procedure Añadir;
  var
   p, i: integer;
   encontrado: boolean;
  begin
   Dicotomi (c, e, p, encontrado); errorconj := false;
   if not encontrado then
    if c.numelem = 100 then errorconj := true
    else
     begin
      for i := c.numelem downto p do c.L[i+1] := c.L[i];
      c.L[p] := e;
      c.numelem := c.numelem + 1
     end
  end;
```

IMPLEMENTACIÓN DEL TAD

```
{implementation ... Continúa}
  prcedure Borrar;
    var
      encontrado: boolean;
      p, i: integer;
    begin
      Dicotomi (c, e, p, encontrado);
      if encontrado then
        begin
          for i :=p to c.numelem - 1 do c.L[i] :=
    c.L[i+1];
          c.numelem := c.numelem – 1
        end
    end;
begin
  errorconj := false;
end.
```

IMPLEMENTACIÓN DEL TAD

- La unidad TADelem proporcionaría:

```
unit TADelem;
  interface
    type
      elemento = char;
    function Mayor (a, b: elemento): boolean;
    function Igual (a, b: elemento): boolean;
    ...
  implementation
    function Mayor;
      begin
        Mayor := a > b
      end;
    function Igual;
      begin
        Igual := a = b
      end;
    ....
end.
```

OTROS CONCEPTOS

- Una estructura de datos es la implementación física de un tipo abstracto de datos. Se reserva esta definición a la representación física de los datos.

- La tendencia actual es la programación orientada a objetos, que no dejan de ser los actuales TADs

  Comunicación: paso de mensajes.

  Mecanismos: herencia y polimorfismo.

- Beneficios de POO: fiabilidad, productividad, reutilización de código, reflejan mejor la realidad.