Tipos Abstractos de Datos

1.1 Conceptos, terminología y ejemplos

1.1.1 Concepto de abstracción

ABSTRACCIÓN: Método de resolución de problemas que consiste en destacar los detalles importantes y dejar a un lado los irrelevantes

- Ejemplos de abstracción en ciencia de la computación:
  - ☑ Lenguajes de alto nivel respecto lenguaje ensamblador
  - ☑ Procedimientos y funciones con parámetros
  - ☑ Librerías de los lenguajes de programación
  - ☑ Lenguajes visuales (DELPHI, Visual Basic, Visual C, etc.)

Tipos Abstractos de Datos

- Distinguimos 2 formas de abstracción

  - ☑ Abstracción de acciones o funcional
    - Procedimientos y funciones
    - Se le asigna un nombre y se parametriza
    - Se oculta información
    - Separación entre ***qué hace*** **(especificación)** y ***cómo se hace*** **(implementación)**

  - ☑ Abstracción de Datos
    - Tipos básicos o estándar: entero, carácter, booleano, etc.
    - Tipos simples definidos por el programador: enumerado, subrango
    - Tipos estructurados: array, registro
    - Tipo Abstracto de Datos (TAD)

Tipos Abstractos de Datos

1.1.2 Tipos Abstractos de Datos

Definición de TAD

Colección de valores y de operaciones definidas sobre ellos mediante una especificación independiente de cualquier representación

☑ La programación con TAD requiere 2 pasos que generan 2 piezas de documentación

- PASO 1: Definición o especificación del tipo
  - Visible al usuario
  - Precisa, legible y no ambigua
  - Nombre del tipo + sintaxis y semántica de las operaciones

- PASO 2: Implementación del tipo
  - Oculta al usuario
  - Estructurada, eficiente y legible
  - Elección de la representación más adecuada para los valores del tipo

Tipos Abstractos de Datos

- Métodos para especificar un tipo abstracto de datos
  - ☑ Especificación formal (especificación algebraica)
  - ☑ Especificación informal (lenguaje natural)
  - ☑ Especificación semi-formal
    - Para el tipo
      - Nombre
      - Descripción
      - Características
      - Valores no admitidos
    - Para las operaciones
      - Parámetros
      - Valor de retorno
      - Precondiciones
      - Efecto
      - Excepciones

Tipos Abstractos de Datos

- Ejemplo. Un TAD para almacenar fichas de personas con una serie de operaciones:

Documento de definición del TAD Ficha de personas

Definición del TIPO

Nombre: ficha

Descripción: Este tipo almacena información sobre personas. Se puede almacenar el nombre, la edad y el número de hijos

Características: Permite nombres iguales. Admite cualquier cadena de caracteres como nombre, etc.

Valores no admitidos: Los descritos en la operación *Crear*

Tipos Abstractos de Datos

Definición de las OPERACIONES

function Crear (nombre: cadena; edad, hijos: byte; **var** f: ficha): byte;

{

Parámetros:

nombre: cadena de caracteres donde se almacena el nombre de la persona

edad: número natural que indica la edad de la persona

hijos: número natural que indica el número de hijos que tiene la persona

f: variable de tipo ficha que se crea al ejecutarse con éxito la operación

Devuelve:

0 ó 1. Si (edad < 16 e hijos > 0), devuelve 0. En caso contrario, la función devuelve 1

Precondiciones:

0 < edad £ 110

0 £ hijos £ 15

Efecto:

Crea una ficha con los valores indicados en los argumentos

Excepciones:

Si (edad < 16 e hijos > 0) no se crea la ficha f y se muestra un mensaje de error

}

Tipos Abstractos de Datos

...

function Descuento (f: ficha): Real;

{

Parámetros:

f: la ficha a la que se le aplica el descuento

Devuelve:

El valor del descuento

Precondiciones:

ficha f creada correctamente (no vacía)

Efecto:

Calcula el descuento de una persona dependiendo del número de hijos que tenga, según esta tabla:

Un 25% por hijo si tiene 1 ó 2 hijos

Un 50% por hijo si tiene entre 3 y 5 hijos

Un 75% por hijo si tiene entre 6 y 10 hijos

Un 100% por hijo si tiene entre 11 y 15 hijos

Excepciones:

}

Tipos Abstractos de Datos

En PASCAL, los TAD se crean en unidades independientes

```
unit tadFicha;
interface
uses ...
type cadena = String[50];
type ficha = record
                nombre: cadena;
                edad: 1..110;
                hijos: 0..15;
             end;

function Crear (nombre: cadena; edad, hijos: byte; var f: ficha): byte;
procedure Ver (f: ficha);
function Descuento (f: ficha): Real;

implementation
uses ...
function Crear (nombre: cadena; edad, hijos: byte; var f: ficha): byte;
{ codificación de la operación }

end.    { Fin de la unidad }
```

Tipos Abstractos de Datos

❑ El usuario sólo podrá crear y manipular variables de tipo *ficha* con las operaciones que se ofrecen

```
program usuario;
uses tadFichas;
var  f1, f2, f3, f4: ficha;
     d1: Real;
     res: Integer;

f1 := ¿?
res:= Crear ('José Luís … ', 30, 3, f2);   { llamada correcta }
res:= Crear ('Antonio … ', 5, 2, f3);      { llamada incorrecta, salta una excepción }
res:= Crear ('Isabel … ', 26, -5, f4);     { llamada incorrecta, no cumple las precondiciones. Resultado impredecible }

d1 := Descuento (f2);
```

La **encapsulación** u **ocultación de la información** consiste en:

☑ Privacidad de la representación. El usuario no conoce los detalles de cómo se representan los datos en la memoria del ordenador.

☑ Protección del tipo. el usuario sólo puede usar las operaciones definidas en la especificación.

Tipos Abstractos de Datos

1.2 Programación con TAD

1.2.1 Los TAD como base del diseño modular

- **Modularidad.** Mecanismo que permite descomponer el código de un proyecto software
- **Módulo.** Unidad de programa que puede ser desarrollada independientemente del resto

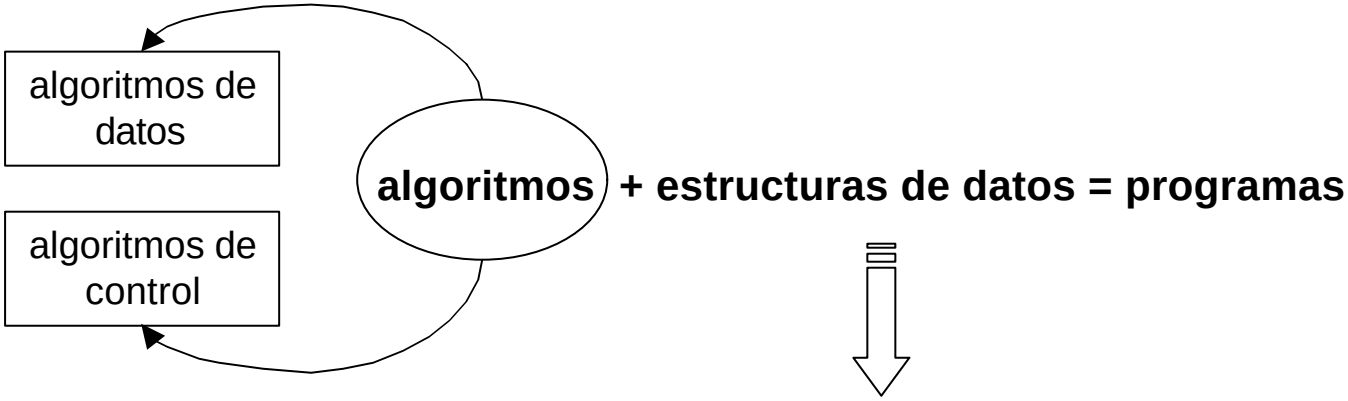
- Un diseño modular correcto desde el punto de vista de la *Ingeniería de la Programación* o *Ingeniería del Software*, debe cumplir una serie de requisitos:

  - ☑ Facilidad de descomponer un problema en subproblemas menos complejos
  - ☑ Mínima conexión entre los módulos
  - ☑ Los cambios y mejoras del proyecto deben afectar sólo a un número pequeño de módulos

Tipos Abstractos de Datos

1.2.2 La programación en gran escala

- El diseño descendente junto con la abstracción funcional no es suficiente:
  - ☑ Desequilibrio entre acciones abstractas o de alto nivel y tipos de datos concretos o de bajo nivel
  - ☑ Decisiones sobre la representación de los datos se toman desde el principio



algoritmos de control + TAD = programas

Tipos Abstractos de Datos

- **Ejemplo:** Programa que lea una secuencia de enteros desde fichero y escriba en pantalla cada entero distinto leído junto con su frecuencia de aparición, en orden decreciente de frecuencias

ENTRADA: 5 15 25 5 7 5 7 15 7 5

SALIDA: 5 4
7 3
15 2
25 1

```
algoritmo estadística
importa tadFrecuencias
var
  t: tablaF;  nombre: cadena;  dato, orden, frec: entero;
fvar
inicio
   escribir ('Nombre del fichero:   ');
   leer (nombre);  asociar (f, nombre);
   iniciarLectura (f);
   inicializar (t);
   mientras ¬ fin (f) hacer
            leer (f, dato);
            añadir (t, dato);
   fmientras
   para orden:= 1 hasta total (t) hacer
            info (t, orden, dato, frec);
            escribir ('Número:   ', dato, 'Frecuencia:    ', frec);
   fpara
falgoritmo
```