Unidad 1 – Tipos Abstractos de Datos

Los POOsexuales

Escuela de Ingeniería en Sistemas

poo@ues.edu.sv

Comunismo

Universidad de El Salvador

comunismo@ues.edu.sv

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent convallis orci arcu, eu mollis dolor. Aliquam eleifend suscipit lacinia. Maecenas quam mi, porta ut lacinia sed, convallis ac dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse potenti.

2012 ACM Subject Classification Replace ccsdesc macro with valid one, e.g., 'Information systems → Question answering'

Keywords and phrases keyword one, keyword 2, three

Digital Object Identifier 10.4230/LIPIcs.ESD115.2023.1

Tabla de Contenidos

1 Definiciones de TAD's **2**

1.1 Abstracción . . . 2

1.2 Proceso de abstracción . . . 2

1.3 Abstracción en informática . . . 2

2 Implementación tradicional vs implementación de TAD's **3**

2.1 Algoritmo . . . 3

2.2 Ventajas de los TAD's . . . 3

2.3 Especificaciones de los TAD . . . 4

2.4 ¿Cómo diseñar un TAD? . . . 4

2.5 Clasificaciones de las operaciones . . . 4

2.5.1 Operaciones para crear objetos . . . 4

2.5.2 Operaciones para transformar objetos de TAD . . . 4

2.5.3 Operaciones para analizar los elementos del TAD . . . 4

2.6 ¿Cómo diseñar un TAD? . . . 5

2.6.1 Análisis del problema . . . 5

2.6.2 Resultado de análisis del problema . . . 5

2.7 Definición de valor y operador . . . 5

2.8 Diseño y/o espificicación de las operaciones . . . 6

3 Especificación semiformal de los TAD – Patricia **7**

Videos de apoyo en ESD115.

Editor: Camilo Medrano; Article No. 1; pp. 1:1–1:8

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Definiciones de TAD's

Cuando alguien nos habla de abstracto, a la mayoría de nosotros se nos viene a la mente algo relacionado con el dibujo o la pintura (Arte abstracto?). £Cierto o Falso?

▶ **Definición 1** (Dibujo Abstracto)**.** *Se define por dibujos de lineas, figuras y colores. No muestra la realidad sino lo que el pintor siente en ese momento.*

1.1 Abstracción

Hay muchas definiciones de lo que es abstracción, revisemos algunas de estas:

▶ **Definición 2** (Abstracción – Filosofía)**.** *Un acto mental el que se aísla conceptualmente un objeto o una propiedad de un objeto.*

▶ **Definición 3** (Abstracción – Psicología)**.** *Un proceso que implica reducir los componentes fundamentales de información de un fenómeno para conservar sus rasgos más relevantes.*

En informática, el énfasis en el "£Qué hace?" más que en el "£Cómo lo hace?".

▶ **Ejemplo 4.** Caja negra: Sabemos que tiene entradas y salidas, más no nos importa como lo hace.

▶ **Definición 5** (Abstracción)**.** *Técnica de quitarle a una idea o a un objeto todos los acompañamientos innecesarios hasta que los deja en una forma esencial y mínima.*

Una buena abstracción elimina todos los detalles poco importantes y permite enfocarse y concentrarse en los detalles importantes.

1.2 Proceso de abstracción

Analizando la definición de abstracción, podemos decir que para lograr abstraer, necesitamos seguir 2 pasos muy concretos, los cuales son:

1. Enfocarse en los aspectos más relevantes del objeto.
2. Ignorar aspectos irrelevantes del mismo.

La irrelevancia depende del nivel de abstracción ya que si se pasa a niveles más concretos, es posible que ciertos aspectos pasen a ser relevantes o irrelevantes

1.3 Abstracción en informática

La ciencia de la computación es la ciencia de abstracción.

Se puede pensar en el tamaño de un objeto sin conocer cómo está constituido ese objeto.

La abstracción es un mecanismo fundamental para la compresión de fenómenos o situaciones que implican gran cantidad de detalles, es considerada, como uno de los conceptos más potentes en el proceso de resolución de problemas.

Se entiende por abstracción la capacidad de manejar un tema o idea como un concepto general, sin considerar los detalles.

▶ **Ejemplo 6.** Saber conducir un automóvil. No nos interesa la marca o el color, solamente saber conducir.

Un programa es una abstracción en dos niveles.

Si consideramos una computadora como un dispositivo concreto, el programa es una abstracción por que se aleja de las instrucciones que brinda la máquina y se acerca al problema.

Si consideramos la resolución de un problema, el programa es una abstracción por que sólo modela sus aspectos relevantes.

DIAGRAMA 8:38

2 Implementación tradicional vs implementación de TAD's

Empezamos con la ecuación de ????

Datos + Algoritmos = Programa

2.1 Algoritmo

▶ **Definición 7** (Algoritmos de Datos)**.** *Son todos los métodos, funciones y controles que ejercen sobre algún dato en específico.*

▶ **Ejemplo 8.** La multiplicación es un algoritmo, un proceso que le pertenece a todos los números, pero es un algoritmo que puede ser común en muchas soluciones.

▶ **Definición 9** (Algoritmos de Control)**.** *Son la parte restante (la que representa en sí la lógica de solución del problema, independiente hasta cierto punto de las estructuras de datos seleccionadas).*

Algoritmo de Datos + Algoritmo de Control = Algoritmo

(Datos + Algoritmo de Datos) + Algoritmo de Control = Programa

Implementación del TAD + Algoritmo de Control = Programa

2.2 Ventajas de los TAD's

- Permite una mejor conceptualización y modelización del mundo real
- Mejora la robustez del sistema. Los TAD's permiten la comprobación de tipos para evitar errores de tipo de tiempo de ejecución.
- Mejora el rendimiento (prestaciones). Para sistemas tipificados, el conocimiento de los objetos permite la optimización del tiempo de compilación.
- Separa la implementación de la especificación. Permite la modificación y mejora de la implementación sin afectar a la interfaz pública del TAD.
- Permite la extensabilidad del sistema. Los componentes de software reutilizables son más fáciles de crear y mantener.
- Recoge mejor la semántica del tipo. Los TAD agrupan o localizan las operaciones y la representación de atributos.

2.3 Especificaciones de los TAD

La especificación de un TAD consta de dos partes, la **descripción** matemática del conjunto de datos, y las **operaciones** definidas en ciertos elementos de ese conjunto de datos. El objetivo de la especificación es describir el comportamiento del TAD

Estas especificaciones pueden ser:

▶ **Definición 10** (Informal). *Se describen los datos y la operaciones relacionadas al uso del lenguaje natural.*

▶ **Definición 11** (Semiformal). *Donde se combinan elementos del lenguaje natural y axiomas o lenguaje de programación.*

▶ **Definición 12** (Formal). *A través del cual se suministra un conjunto de axiomas que describen las operaciones en su aspecto sintáctico y semántico.*

2.4 ¿Cómo diseñar un TAD?

- Se establece el nombre del TAD y los datos que lo forman.
  - Nombre del TAD.
  - Valores y su descripción.
- Posteriormente se especifica cada una de las operaciones con sus argumentos, y una descripción funcional en lenguaje natural.
  - Operación (argumentos).
  - Descripción funcional.

2.5 Clasificaciones de las operaciones

2.5.1 Operaciones para crear objetos

▶ **Definición 13** (Iniciales). *Se utilizan para crear objetos del TAD, en cuya creación no se requiere ningún objeto abstracto del mismo tipo.*

▶ **Definición 14** (Constructores). *Utilizadas para crear objetos del TAD cuya creación depende de objetos del mismo tipo.*

2.5.2 Operaciones para transformar objetos de TAD

▶ **Definición 15** (Simplificadoras). *Son operaciones cuyo codominio es el TAD que se define, pero que dan como resultado objetos que pueden ser descritos utilizando únicamente operaciones iniciales y constructoras.*

2.5.3 Operaciones para analizar los elementos del TAD

▶ **Definición 16** (Analizadoras). *Son operaciones cuyo dominio no es el TAD que se define, si no otro ya conocido. El propósito es obtener información de los objetos de tipo abstracto.*

2.6 ¿Cómo diseñar un TAD?

▶ **Ejemplo 17.** Realizar una especificación informal de un TAD para un "Conjunto Numérico" con las operaciones Crear Conjunto, Es vacío, Añadir un elemento al conjunto, Pertenece un elemento al conjunto, Retirar un elemento del conjunto, Unión de dos conjuntos, Intersección de dos conjuntos e Inclusión de conjuntos.

2.6.1 Análisis del problema

Definición del tipo de datos

TAD Conjunto (especificación de elementos sin duplicidad, pueden estar en cualquier orden, se usa para representar los conjuntos matemáticos de números enteros positivos con sus operaciones).

```
TAD <Nombre> (<descripción>)
```

Descripción de las operaciones

`CrearConjunto()`: Crea un conjunto sin elementos
`Añadir(Conjunto, elemento)`: Comprueba si el elemento forma parte del conjunto, en caso negativo es añadido. La función modifica al conjunto
`Retirar(Conjunto, elemento)`: En el caso de que el elemento pertenezca al conjunto es eliminado de este. La función modifica al conjunto.
`Pertenece(Conjunto, elemento)`: Verifica si el elemento forma parte del conjunto, en cuyo caso devuelve cierto.
`EsVacio(Conjunto)`: Verifica si el conjunto no tiene elementos, en cuyo caso devuelve cierto.
`Cardinal(Conjunto)`: Devuelve el número de elementos del conjunto.
`Union(Conjunto, Conjunto)`: Realiza la operación matemática de la unión de dos conjuntos. La operación devuelve un conjunto con los elementos comunes y no comunes a los dos argumentos.
`Interseccion(Conjunto, Conjunto)`: Realiza la inclusión matemática de la intersección de dos conjuntos. La operación devuelve un conjunto con los elementos comunes a los dos argumentos.
`Inclusión(Conjunto, Conjunto)`: Verifica si el primer conjunto está incluido en el conjunto especificado en el segundo argumento, en cuyo caso devuelve cierto.

2.6.2 Resultado de análisis del problema

Generar tabla

2.7 Definición de valor y operador

Un TAD consta de dos partes:

▶ **Definición 18** (Definición de valor)**.** *Establece el conjunto de valores para el TAD y consta de dos partes: una cláusula de definición y una cláusula de condición.*

▶ **Definición 19** (Definición de operador)**.** *Cada operador está definido como una función abstracta con cuatro partes: un encabezado, un área de declaración de variables, las condiciones previas [opcionales] y las condiciones posteriores.*

2.8 Diseño y/o espificicación de las operaciones

Tomaremos como estándar que todas las palabras o signos con formato '**negrita**' son inamovibles, son parte de la sintaxis de una especificación de un TAD, y todo lo que no esté en '**negrita**' es lo variable en la especificación de un TAD.

▶ Comentario. Como no estoy programando en word, no puedo poner en negrita abstract y condition.

Listing 1 Definición de Valor

```
abstract typedef <{tipo de dato del TAD}, ... ,{tipo de dato del TAD}>
{nombre del TAD}

condition {Condicion que restringe los valores que puede tomar el TAD}
```

Listing 2 Definición de Operador

```
abstract {tipo de dato del valor de retorno} {Nombre de la operación}
({parámetros divididos por comas})

{Declaración de variable: [tipo] [nombre de la variable] =
[valor inicial_opcional]}

PreCondition {Restricciones necesarias para llevarse a cabo la
operación}

PostCondition
{Explicación de toda la operación y como va a llegar a obtener el
resultado deseado, para esta parte puede optarse por escribirlo en
lenguaje C, o hacerlo con pseudocódigo}
```

▶ **Ejemplo 20.**

Listing 3 Definición de Valor

```
abstract typedef <integer, n> Conjunto

condition Conjunto en los enteros positivos.
```

Listing 4 Definición de Operador

```
abstract int Pertenece(Conjunto MiConjunto, int Elemento)
int i, resultado = 0;

PreCondition MiConjunto != null y
Elementos esta en los enteros positivos

PostCondition
    while MiConjunto[i] != null
        if Miconjunto == Elemento {
            resultado = 1;
            exit; // Rompe el ciclo
        }
return resultado;
```

▶ Nota 21. Entre más detallistas seamos mejor sera para el programa.

3 Especificación semiformal de los TAD – Patricia

▶ Nota 22. Repaso del enfoque semiformal.

Listing 5 Operación Crear

```
/*La siguiente operación tiene como objetivo crear un conjunto vacio
para almacenar números enteros positivos*/

abstract Conjunto CrearConjunto()

Conjunto MiConjunto = null;

PostCondition CrearConjunto = MiConjunto
```

Listing 6 Operación Pertenece

```
/*El objetivo de la operación Pertenece es identificar si un elemento
dado pertenece al conjunto. Un valor de retorno 1 indica que el
elemento ya existe en el conjunto. Una valor 0 indica que el elemento
no se encuentra en el conjunto*/

abstract int Pertenece(Conjunto MiConjunto, int Elem)
int i, Resultado=0;

Precondition MiConjunto != null y Elem en los enteros positivos

Postcondition
while MiConjunto[i] != null
  if MiConjunto[i] == Elem{
    Resultado = 1;
    exit; //rompe ciclo
  }

return Resultado;
```

Listing 7 Operación Añadir

```
/*El objetivo de la operación añadir es agregar nuevos elementos al
conjunto. No se permiten duplicados*/

abstract Conjunto Añadir(Conjunto MiConjunto, int Elem)
int i = 0;

Precondition Elem en los enteros positivos

Postcondition
if !Pertenece(MiConjunto, Elem){
  //identifica la primera posición vacía
  while (MiConjunto[i] != 0)
    i = i+1;
  MiConjunto[i]=Elem;
}
return MiConjunto;
```

Listing 8 Operación Cardinal

```
/*El objetivo de la operación cardinal es identificar el número de
elementos que se encuentran dentro de un conjunto*/

abstract int Cardinal(Conjunto MiConjunto)
int NumElementos, i = 0;

Precondition MiConjunto != null

Postcondition
while (MiConjunto[i] != 0)
  i = i+1;
NumElementos = i;
return NumElementos;
```

Listing 9 Operación Retirar

```
/*El objetivo de la operación retirar es elminar un elemento del
conjunto, en caso que este pertenezca al conjunto*/

abstract Conjunto Retirar(Conjunto MiConjunto, int Elem)
int i = 0;

Precondition MiConjunto != null

Postcondition
if !Pertenece(MiConjunto, Elem){
  while (MiConjunto[i] != Elem)
    i = i+1;
  MiConjunto[i]=0;
}
return MiConjunto;
```

Existen otras operaciones que podemos definir en nuestro TAD pero quedan para nosotros.