Programación II

Estructuras de Datos y Abstracción de Datos

Universidad Nacional de Rosario.
Facultad de Ciencias Exactas, Ingeniería y Agrimensura.

Introducción

Los tipos de datos definidos en unidades anteriores son **tipos de datos concretos**. Por ejemplo: `Point` (punto) se definió como un par ordenado de flotantes; `Rectangle` (rectángulo) se definio como un punto y dos flotantes.

Introducción

Los tipos de datos definidos en unidades anteriores son **tipos de datos concretos**. Por ejemplo: `Point` (punto) se definió como un par ordenado de flotantes; `Rectangle` (rectángulo) se definio como un punto y dos flotantes.

Otra forma de definir tipos de datos es **mediante sus operaciones**: enumerándolas e indicando su comportamiento (es decir, cuál es su resultado esperado).

Introducción

Los tipos de datos definidos en unidades anteriores son **tipos de datos concretos**. Por ejemplo: `Point` (punto) se definió como un par ordenado de flotantes; `Rectangle` (rectángulo) se definio como un punto y dos flotantes.

Otra forma de definir tipos de datos es **mediante sus operaciones**: enumerándolas e indicando su comportamiento (es decir, cuál es su resultado esperado).

Esta manera de definir datos se conoce como **Tipos Abstractos de Datos** o **TADs** (o *tipo de datos abstracto*, ambas provienen de la traducción del término en inglés *abstract data type*).

Introducción a Tipos Abstractos de Datos

Un **tipo abstracto de datos** o **TAD** es:

Introducción a Tipos Abstractos de Datos

Un **tipo abstracto de datos** o **TAD** es:

- una **colección de datos**

Introducción a Tipos Abstractos de Datos

Un **tipo abstracto de datos** o **TAD** es:

- una **colección de datos**
- acompañada de un **conjunto de operaciones para manipularlos**, de forma tal que queden ocultas la representación interna del nuevo tipo y la implementación de las operaciones, para todas las unidades de programa que lo utilice.

Introducción a Tipos Abstractos de Datos

Un **tipo abstracto de datos** o **TAD** es:

- una **colección de datos**
- acompañada de un **conjunto de operaciones para manipularlos**, de forma tal que queden ocultas la representación interna del nuevo tipo y la implementación de las operaciones, para todas las unidades de programa que lo utilice.

Introducción a Tipos Abstractos de Datos

¿Por qué son **útiles los tipos abstractos de datos**?

- Los programas que los usan hacen referencia a las operaciones que tienen, no a la representación, y por lo tanto ese programa sigue funcionando si se cambia la representación.

Introducción a Tipos Abstractos de Datos

¿Por qué son **útiles los tipos abstractos de datos**?

- Los programas que los usan hacen referencia a las operaciones que tienen, no a la representación, y por lo tanto ese programa sigue funcionando si se cambia la representación.
- Simplifican el desarrollo de algoritmos utilizando las operaciones del tipo abstracto de dato, sin importar cómo las mismas son implementadas.

Introducción a Tipos Abstractos de Datos

¿Por qué son **útiles los tipos abstractos de datos**?

- Los programas que los usan hacen referencia a las operaciones que tienen, no a la representación, y por lo tanto ese programa sigue funcionando si se cambia la representación.
- Simplifican el desarrollo de algoritmos utilizando las operaciones del tipo abstracto de dato, sin importar cómo las mismas son implementadas.
- Dado que una operación puede ser implementada de diferentes formas en un TAD, resulta útil escribir algoritmos que puedan ser usados con cualquiera de sus posibles implementaciones.

Introducción a Tipos Abstractos de Datos

¿Por qué son **útiles los tipos abstractos de datos**?

- Los programas que los usan hacen referencia a las operaciones que tienen, no a la representación, y por lo tanto ese programa sigue funcionando si se cambia la representación.
- Simplifican el desarrollo de algoritmos utilizando las operaciones del tipo abstracto de dato, sin importar cómo las mismas son implementadas.
- Dado que una operación puede ser implementada de diferentes formas en un TAD, resulta útil escribir algoritmos que puedan ser usados con cualquiera de sus posibles implementaciones.
- Algunos TADs utilizados con frecuencia, son implementados en librerías estándares de manera que puedan ser utilizados por cualquier programador.

Introducción a Tipos Abstractos de Datos

¿Por qué son **útiles los tipos abstractos de datos**?

- Los programas que los usan hacen referencia a las operaciones que tienen, no a la representación, y por lo tanto ese programa sigue funcionando si se cambia la representación.
- Simplifican el desarrollo de algoritmos utilizando las operaciones del tipo abstracto de dato, sin importar cómo las mismas son implementadas.
- Dado que una operación puede ser implementada de diferentes formas en un TAD, resulta útil escribir algoritmos que puedan ser usados con cualquiera de sus posibles implementaciones.
- Algunos TADs utilizados con frecuencia, son implementados en librerías estándares de manera que puedan ser utilizados por cualquier programador.
- Las operaciones de los TADs proveen una especie de lenguaje de alto nivel para discutir y especificar otros algoritmos.

Introducción a Tipos Abstractos de Datos

Dentro del **ciclo de vida de un TAD** hay dos fases:

Introducción a Tipos Abstractos de Datos

Dentro del **ciclo de vida de un TAD** hay dos fases:

la programación del TAD: en la cual se elige una representación, y luego se programa cada uno de los métodos sobre esa representación.

Introducción a Tipos Abstractos de Datos

Dentro del **ciclo de vida de un TAD** hay dos fases:

la programación del TAD: en la cual se elige una representación, y luego se programa cada uno de los métodos sobre esa representación.

la construcción de los programas que lo usan: en esta fase no será relevante para el programador que utiliza el TAD cómo está implementado, sino únicamente los métodos que posee.