UNIVERSIDAD
DE GRANADA

Relación 2

Abstracción

MAURICIO LUQUE JIMÉNEZ

21 de noviembre de 2025

Estructuras de Datos

1. Definir el T.D.A Servidor de red. Un servidor es un punto de red que se encuentra identificado por una dirección ip. Una dirección ip viene definida por cuatro dígitos que pueden tener valores que van desde 0 a 255. Se pide:

- **Dar la especificación del tipo Servidor. Además establecer las operaciones que manejan al T.D.A**

    - **Especificación**: representación de un servidor de red identificado por una dirección IP compuesta por cuatro dígitos leídos consecutivamente, donde cada dígito es un número entero positivo menor a 255.

    - **Operaciones:**
        - Constructor por defecto (dirección IP 0.0.0.0)
        - Constructor con parámetros
        - Constructor de copia a partir de otro servidor
        - Operador de consulta de la dirección IP completa
        - Operador de consulta de un dígito concreto de la dirección IP
        - Modificador de la dirección IP completa
        - Modificador de un dígito concreto de la dirección IP
        - Operador de entrada
        - Operador de salida

- **Definir al menos dos tipo rep**

```C/C++
class Servidor
{
	private:
		int d1, d2, d3, d4; // Ejemplo: 192, 168, 1, 0
}
```

```C/C++
class Servidor
{
	private:
		string ip; // Ejemplo: "192.168.1.0"
}
```

- **Escoger uno de los tipo rep y para este establecer la función de abstracción e invariante de representación**
    - **Tipo rep elegido:**

```C/C++
class Servidor
{
	private:
		int d1, d2, d3, d4; // Ejemplo: 192, 168, 1, 0
}
```

    - **Función de abstracción**
        - $f_A(r) = (r.d1, r.d2, r.d3, r.d4)$
        - Representa la dirección IP d1.d2.d3.d4

    - **Invariante de representación:**
        - 0 < d1, d2, d3, d3 < 255

2. Definir el T.D.A Subred. Este T.D.A es una colección de servidores (s1,s2,...,sn), según se ha definido el T.D.A Servidor en el ejercicio anterior. En el T.D.A. Subred también se almacena si dos servidores están conectados (existe un enlace directo) entre ellos. Se pide:

- **Dar la especificación del tipo Subred. Además establecer las operaciones que manejan a T.D.A**

    - **Especificación:** representación de una subred formada por uno o más servidores definidos mediante el T.D.A Servidor, que además indica si dos servidores están conectados

    - **Operaciones**
        - Constructor por defecto (subred sin servidores)
        - Constructor de copia a partir de otra subred
        - Operador de consulta de la subred completa
        - Operador de consulta para devolver un servidor de una subred
        - Operador de consulta para comprobar si dos servidores están conectados
        - Operador de consulta del tamaño de la subred (nº de servidores)
        - Modificador para añadir un servidor a la subred
        - Modificador para eliminar un servidor de la subred
        - Modificador para crear una conexión entre dos servidores
        - Modificador para eliminar una conexión entre dos servidores
        - Operador de entrada
        - Operador de salida

- **Definir al menos dos tipo rep**

```C/C++
class Subred
{
	private:
		int max_servidores;
		Servidor servidores[max_servidores];
		bool conectados[max_servidores][max_servidores];

}
```

```C/C++
class Subred
{
	vector<pair<Servidor, list<Servidor>>> subred;
}
```

- **Escoger uno de los tipo rep y para este establecer la función de abstracción e invariante de representación**

```C/C++
class Subred
{
	private:
		int max_servidores;
		vector<Servidor> servidores;
		bool conectados[max_servidores][max_servidores];
}
```

  - **Función de abstracción**
    - $f_A(r) = (\{r.servidores[i] \mid 0 \leq i \leq r.max\_servidores\},$

      $\{(r.servidores[i],\ r.servidores[j])\} \mid 0 \leq i, j < r.n,\ r.conectado[i][j]\})$
  - **Invariante de representación**
    - 0 <= servidores.size() <= max_servidores
    - Todos los elementos de *servidores* son distintos
    - *conectados[i][i]* es falso para todo i
    - *conectados[i][j]* == *conectados[j][i]* para todo i, j

3. Definir el T.D.A Punto Geográfico. Un punto geográfico se define por una latitud y longitud. La latitud es la distancia en grados desde la línea del Ecuador a los Polos. Su rango va desde -90º a 90º. La longitud es la distancia desde el meriadiano 0 al punto donde estamos. El rango de valores que adopta va desde -180º a 180º. Se pide:

- **Dar la especificación del tipo Punto Geográfico. Además establecer las operaciones que manejan a T.D.A**

    - **Especificación:** representación de las coordenadas de un punto geográfico almacenando su latitud y su longitud.

    - **Operaciones**
        - Constructor por defecto (latitud y longitud 0)
        - Constructor de copia a partir de otro punto geográfico
        - Operador de consulta para devolver la latitud
        - Operador de consulta para devolver la longitud
        - Modificador de la latitud
        - Modificador de la longitud
        - Operador de entrada
        - Operador de salida

- **Definir al menos dos tipo rep**

```C/C++
class PuntoGeografico
{
	double latitud;
	double longitud;
}
```

```C/C++
class PuntoGeográfico
{
	pair<double, double> punto;
}
```

- **Escoger uno de los tipo rep y para este establecer la función de abstracción e invariante de representación**

```C/C++
class PuntoGeografico
{
	double latitud;
	double longitud;
}
```

  - **Función de abstracción**
    - $f_A(r) = (\{r.latitud, r.longitud\}$

  - **Invariante de representación**
    - -90 <= latitud <= 90
    - -180 <= longitud <= 180

4. Definir el T.D.A Ruta. Una ruta es una secuencia de puntos geográficos (ver ejercicio anterior). Se pide:

- **Dar la especificación del tipo Ruta. Además establecer las operaciones que manejan a T.D.A**
  - **Especificación:** sucesión de puntos geográficos representados mediante el T.D.A *PuntoGeográfico.*

  - **Operaciones**
    - Constructor por defecto (ruta sin puntos)
    - Constructor de copia a partir de otra ruta
    - Operador de consulta para devolver un punto
    - Operador de consulta para comprobar si un punto está en la ruta
    - Operador de consulta para devolver el tamaño de la ruta
    - Modificador para añadir un punto a la ruta
    - Modificador para eliminar un punto de la ruta
    - Operador de entrada
    - Operador de salida

- **Definir al menos dos tipo rep**

```
C/C++
class Ruta
{
	private:
		vector<PuntoGeografico> puntos;
}
```

```
C/C++
class Ruta
{
	private:
		// Cada punto tiene asociado el siguiente punto de la ruta
		vector<pair<PuntoGeografico>, PuntoGeografico>> puntos;
}
```

- **Escoger uno de los tipo rep y para este establecer la función de abstracción e invariante de representación**

```C/C++
class Ruta
{
	private:
		vector<PuntoGeografico> puntos;
}
```

  - **Función de abstracción**
    - $f_A(r) = (\{r.puntos[i] \mid 0 \leq i \leq r.ountos\})$

  - **Invariante de representación**
    - Para todo *i* tal que 0 <= i <= *r.puntos.size()*, el objeto *r.puntos[i]* cumple el invariante de representación del T.D.A *PuntoGeográfico*.
    - *puntos.size* >= 1

5. Dar una especificación para la función que permite derivar un polinomio. Suponiendo que tenemos el TDA Polinomio, la cabecera de la función derivada sería asi: void Derivar(const Polinomio & p_origen, Polinomio & p_derivada);

C/C++

```
/**
 * @brief Calcula la derivada de un polinomio
 *
 * @param p_origen   polinomio que se desea derivar (no se modifica)
 * @param p_derivada polinomio donde se almacena la derivada previamente calculada
 *
 * @pre  p_origen es un polinomio válido de grado mayor o igual que 0
 * @post Para todo x real se cumple Q(x) = P'(x)
 * @post Si P tiene grado 0 (es constante) entonces Q es el polinomio 0
 */
void Derivar(const Polinomio & p_origen, Polinomio & p_derivada);
```