LECCIÓN 5: TIPOS DE DATOS ABSTRACTOS

Abstraer → Crear una capa sobre otra u otras

OBJETIVOS

- Clarificar concepto de abstracción y abstracción de datos
- Aprender como abstraer T. D. A
  - Especificación
  - Elección de E.D y operaciones
- Uso de C++ con estos conceptos

Parte privada { - representación; - implementación

Documentación

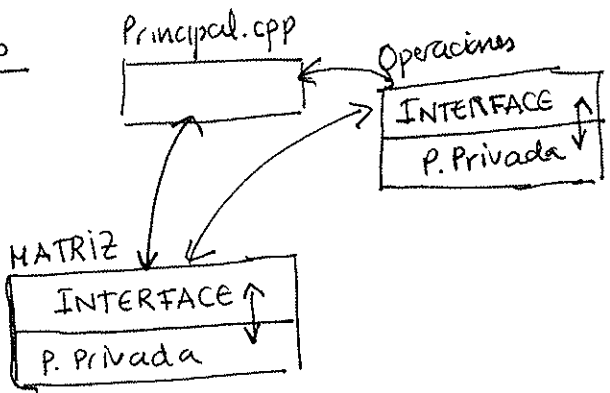
Para que el módulo operaciones se pueda desarrollar el módulo Matriz debe darle una especificación de las operaciones que le ofrece el módulo.

- Abstracción por especificación
  - se dan los detalles independientemente de la implementación.

- DE UN MÓDULO
  - ESPECIFICACIÓN: da las características sintácticas y semánticas de la parte pública. Se especifica en lenguaje natural. Deben ser suficientes para que otros módulos los usen.
  - IMPLEMENTACIÓN (.cpp). Documento que presenta las características internas del módulo.

- ABSTRACCIÓN ⟹ Ocultamiento de Inf.
  - MÓDULO
    - Parte Pública: Visible a los usuarios del módulo
    - Parte Privada: Son ocultos a los usuarios del módulo

Ejemplo




```
//Matriz.h
class Matriz {
  private:
    - -
    - - -
  public: //Interfaca
    Matriz();
    ~Matriz();
    Matriz & operator=(const Matriz &M);
    Matriz (const Matriz &M);
    int GetFilas() const;
    int Get Cols() const;
    void Set (int i, int j, int v);
    int  Get (int i, int j) const;
};
```

```
//operaciones.h
#include "Matriz.h"
Matriz Suma (const Matriz &M1,
             const Matriz &M2)

Matriz Diferencia (const Matriz &M
                   const Matriz &M2)
```

```
//principal.cpp
#include "Matriz.h"
#include "operaciones.h"

int main(){
  Matriz M1, M2;
  .
  M1 = M1 + M2;
  M2 = M1 - M2;
}
```

LECCIÓN 5 - continuación

- ESPECIFICACIÓN de UN MÓDULO
  - usa → Abstracción por especificación
    - ¿qué? sí importa
    - ¿cómo? no importa.

- Abstracción por especificación
  - Precondiciones
  - Postcondiciones

Ejemplo

```
int buscaMinimo(const float *v, int n);
```

```
/**
 * Precondiciones:
 *      . n > 0
 *      . v: es un vector con n elementos
 *
 * Postcondiciones:
 *      devuelve la posición del mínimo elemento
 *      en v
 */
```

(documentación ESPECIFICACIÓN)

ABSTRACCIÓN Procedimental

- función: se agrupa un conjunto de operaciones como si fuera una, de forma que a partir de los datos de entrada se obtiene otros de salida, sin importar cómo los obtiene. Sí importa qué hace.

Para una misma especificación de una función podemos tener diferentes implementaciones.

ESPECIFICACIÓN de UNA FUNCIÓN

ELEMENTOS (en doxygen)

- Descripción de lo que hace
  - @brief <descripción>
- Argumentos
  - Especifica cada uno de los argumentos no el tipo
  - Se indican las restricciones sobre cada parámetro
  - Se indica si el parámetro es modificado o no
  - @param <nombre>: <texto>
- Devuelve: Describe qué valores devuelve la función.
  - @return
- Precondiciones: restricciones impuestas para poder usar la función que no se ha recogido en los argumentos.
  - @pre <precondición>
- Postcondiciones: condiciones que deben cumplirse tras la ejecución de la función.
  - @post
- Excepciones = Resultados extraños para entradas correctas.

(3)

LECCION 5.- continuación (como documentar una función en doxygen)

```
/**
 * @file ej-abstraccion.cp
 *
 **/

/**
 * @brief: Obtiene la posición de un elemento x en un array 1-D
 * @param v: array 1-d que contiene los elementos donde encontrar
 *           el elemento x
 * @param x: elemento que se busca en el vector v
 * @pre
 *   - n >= 0
 *   - v tiene al menos n elementos
 * @return Devuelve la posicion del elemento la x en la v.
 *         Devuelve la i tq v[i]=x para 0<=i<n
 * @exception: Devuelve -1 si el elemento la x no se encuentra
 *             en la v.
 */
int Busqueda (const int *v, int n, int x);
```

Para poder ejecutar doxygen

1) doxygen -g // esto genera Doxyfile

Manual doxygen: www.doxygen.org

LECCIÓN 5 : continuación

. Abstracción de T.D.A

- T.D.A.: conjunto de datos con un conjunto de operaciones asociadas proporcionando una especificación sobre ellos que es independiente de la implementación

tipos de operaciones
- constructores primitivos
- constructores de copia
- observadores o de consulta
- modificadores
- destructores.

IMPLEMENTACIÓN de UN T.D.A
1) Elegir una representación
2) Basándonos en la representación escogida dar una implementación de las operaciones.

DOS TIPOS de DATOS:
- TIPO ABSTRACTO → especificación
- TIPO rep: tipo a usar para representar los objetos de T.D.A y sobre el implementar las operaciones.

Ejemplo: T. D. A FECHA

TIPO ABSTRACTO — ESPECIFICACION
representa una fecha en el calendario occidental.

POSIBLES tipo rep

①
```
class Fecha{
  private:
    int d,m,a;
    :
```

②
```
class Fecha{
  private:
    string a;
    :
```

Ejemplo2: T.D.A Polinomio

ESPECIFICACIÓN

Sucesión de reales $\{a_0, a_1, \ldots, a_n\}$ que representa a polinomios con coeficientes reales de tipo $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

POSIBLES tipo rep

①
```
class Polinomio{
  private:
    struct monomio{
      int grado;
      float coef;
    };
    monomio *p;
    :
```

②
```
class Polinomio{
  private:
    float *p;
    int maxgrado;
    int grado;
    :
};
```