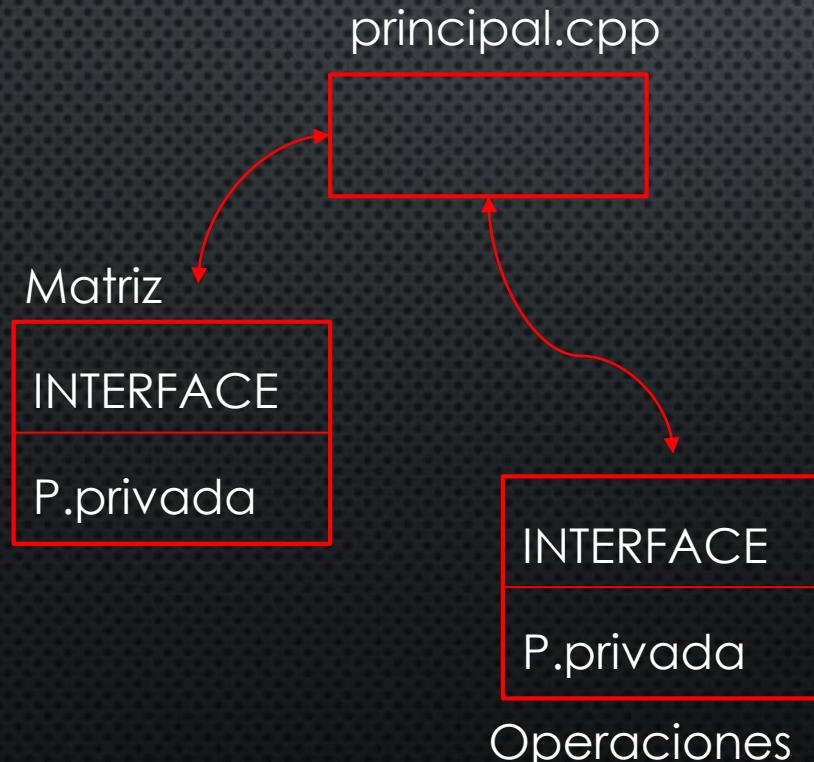


TIPOS DE DATOS ABSTRACTOS

Abstraer: Crear una capa sobre otras u otras

Objetivos.- 1.- Concepto de Abstracción
2.- Aprender a abstraer tipos de datos abstractos: TDA

Ejemplo.- Crear un programa que dadas dos matrices obtenga la matriz suma de las matrices de entrada



TIPOS DE DATOS ABSTRACTOS

Ejemplo.-



```
//Matriz.h  
class Matriz{  
private:  
    ....  
public:  
};
```

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

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

TIPOS DE DATOS ABSTRACTOS

Ejemplo.-



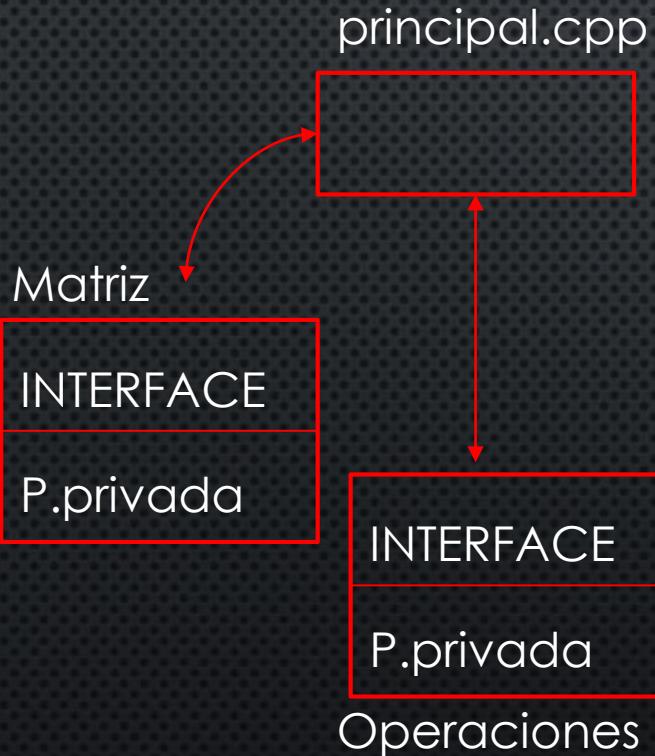
```
//Matriz.h  
class Matriz{  
private:  
    ....  
public:  
};
```

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

```
//principal.cpp  
1. #include "Matriz.h"  
2. #include "operaciones.h"  
3. int main(){  
4.     Matriz m1,m2,m3;  
5.     cin>> m1 >> m2;  
6.     m3=m1+m2;  
7.     cout <<m3;  
}
```

TIPOS DE DATOS ABSTRACTOS

Ejemplo.-



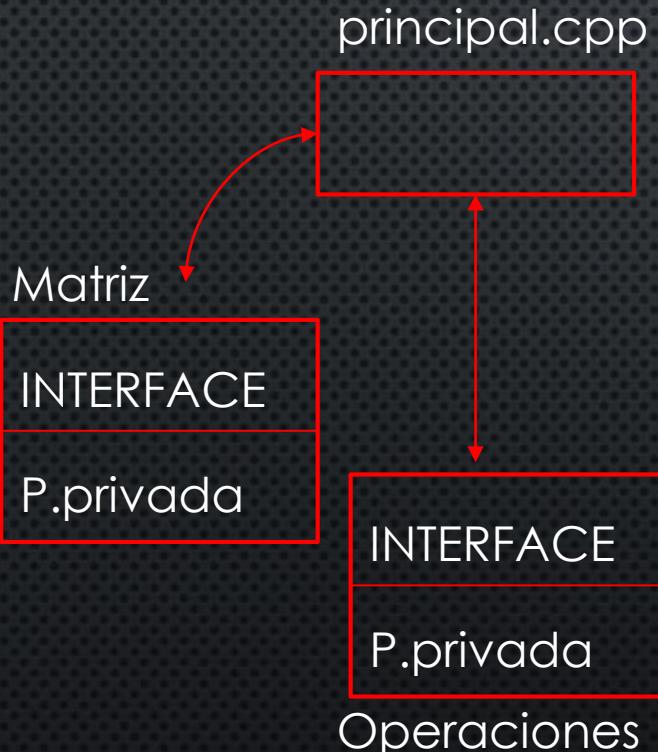
```
//Matriz.h
class Matriz{
private:
.....
public:
    Matriz ();
    Matriz(int nf, int nc);
    Matriz (const Matriz &M);
    ~Matriz();
    Matriz & operator=(const Matriz & M);
    int & operator(int i,int j);
    const int & operator(int i, int j)const;
    int getFilas()const;
    int getCols()const;
    friend ostream & operator<<
        (ostream &os, const Matriz &M);
    friend istream & operator>>
        (istream &is, Matriz &M);
};
```

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

```
//principal.cpp
1. #include "Matriz.h"
2. #include "operaciones.h"
3. int main(){
4.     Matriz m1,m2,m3;
5.     cin>> m1 >> m2;
6.     m3=m1+m2;
7.     cout <<m3;
}
```

TIPOS DE DATOS ABSTRACTOS

Ejemplo.-



```
//Matriz.h
class Matriz{
private:
    ....
public:
    Matriz ();
    Matriz(int nf, int nc);
    Matriz (const Matriz &M);
    ~Matriz();
    Matriz & operator=(const Matriz & M);
    int & operator(int i,int j);
    const int & operator(int i, int j)const;
    int getFilas()const;
    int getCols()const;
    friend ostream & operator<<
        (ostream &os, const Matriz &M);
    friend istream & operator>>
        (istream &is, Matriz &M);
};
```

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

const operator-(const Matriz &M1, const Matriz &M2);
```

```
//principal.cpp
1. #include "Matriz.h"
2. #include "operaciones.h"
3. int main(){
4.     Matriz m1,m2,m3;
5.     cin>> m1 >> m2;
6.     m3=m1+m2;
7.     cout <<m3;
}
```

TIPOS DE DATOS ABSTRACTOS

Ejemplo.-



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

Abstracción por especificación. Se da los detalles independientes de la implementación. Importa qué pero no cómo.

Ejemplo:

```
const operator-(const Matriz &M1, const Matriz &M2);  
/** Obtenemos la matriz diferencia de dos matrices**/
```

NO es válido el siguiente comentario.

```
/** Para obtener la matriz diferencia en cada punto  
de la matriz de salida recorremos con dos bucles  
anidados punto a punto de las dos matrices y  
obtenemos la diferencia.**/
```

TIPOS DE DATOS ABSTRACTOS

Abstracción por especificación. Se da los detalles independientes de la implementación. Importa qué pero no cómo. Se obtiene mediante un conjunto de precondiciones y postcondiciones.

De un Módulo (p.ej: Matriz)

- **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 Procedimental. 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, y no importa como se obtiene. Si importa qué hace pero no cómo lo hace.

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- Dar la especificación de la función que obtiene el mínimo de un vector.

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Elementos para especificar una función. Doxygen. Se hace en el .h del módulo si tiene en otro caso en el .cpp

- Breve descripción de lo que hace **@brief**

```
/**  
 *  • @brief Obtiene la posición de un elemento x en un array 1-D  
 */
```

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Elementos para especificar una función. Doxygen. Se hace en el .h del módulo si tiene en otro caso en el .cpp

- Argumentos **@param <nombre_variable>**
 - Especifica cada uno de los argumentos no el tipo
 - Se indican las restricciones sobre cada parámetro
 - Se indica si es modificado o no el parámetro en caso de que se pase por referencia

```
* /**
 * @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 a
 * buscar.
 * @param n: numero de elementos en el array
 * @param x: elemento que se busca en el vector.
 */
```

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Elementos para especificar una función. Doxygen. Se hace en el .h del módulo si tiene en otro caso en el .cpp

- Devolución. Describe que valores devuelve la función. **@return**

```
/**  
 * @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 a  
 buscar.  
 * @param n: número de elementos en el vector  
 * @param x: elemento que se busca en el vector.  
 * @return La posición i del elemento x en el array v. Donde i cumple que 0<=i<n  
 */
```

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Elementos para especificar una función. Doxygen. Se hace en el .h del módulo si tiene en otro caso en el .cpp

- Precondiciones: condiciones que deben cumplirse antes de la ejecución de la función.
@pre

```
/**  
 * @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 a  
 buscar.  
 * @param n: numero de elementos en el array  
 * @param x: elemento que se busca en el vector.  
 * @return La posición i del elemento x en el array v. Donde i cumple que 0<=i<n  
 * @pre  
 * n>0  
 * v tiene al menos n elementos  
 */
```

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Elementos para especificar una función. Doxygen. Se hace en el .h del módulo si tiene en otro caso en el .cpp

- Postcondiciones: condiciones que deben cumplirse tras la ejecución de la función.
@post
- Excepciones: resultados extraños para entradas concretas. **@exception**

```
/**  
 * @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 a  
 buscar.  
 * @param n: numero de elementos en el array  
 * @param x: elemento que se busca en el vector.  
 * @return La posición i del elemento x en el array v. Donde i cumple que 0<=i<n  
 * @pre  
 * n>0  
 * v tiene al menos n elementos  
 * @exception: devuelve -1 si el elemento x no se encuentra en v.  
 */
```

```
int Busqueda(const int *v,int n,int x);
```

TIPOS DE DATOS ABSTRACTOS

Abstracción de tipos de datos abstractos (T.D.A)

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

Ejemplo de tipos de operaciones:

- **Constructores: por defecto, por copia, por parámetros.**
- **Destructor**
- **Modificadores**
- **Consultores**
- **Operaciones E/S**

TIPOS DE DATOS ABSTRACTOS

Abstracción de tipos de datos abstractos (T.D.A)

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

Ejemplo de tipos de operaciones:

- **Constructores: por defecto, por copia, por parámetros.**
- **Destructor**
- **Modificadores**
- **Consultores**
- **Operaciones E/S**

```
class Matriz{  
private:  
    ....  
public:  
    Matriz ();  
    Matriz(int nf, int nc);  
    Matriz (const Matriz &M);  
    ~Matriz();  
    Matriz & operator=(const Matriz & M);  
    int & operator(int i,int j);  
    int & operator(int i, int j) const;  
    int getFilas() const;  
    int getCols() const;  
    friend ostream & operator<<  
        (ostream &os, const Matriz &M);  
    friend istream & operator>>  
        (istream &is, Matriz &M);  
};
```

TIPOS DE DATOS ABSTRACTOS

Abstracción de tipos de datos abstractos (T.D.A)

Implementación de un T.D.A:

1. Elegir una representación (tipo rep)
2. Basándonos en la representación escogida dar una implementación de las operaciones

TDA-> Dos tipos

Dado en la especificación

Tipo REP: tipo a usar para representar el TDA sobre el que se implementan las operaciones

TIPOS DE DATOS ABSTRACTOS

Abstracción de tipos de datos abstractos (T.D.A)

Ejemplo.- T.D.A Fecha

Especificación.- Representa a una fecha en el calendario occidental

Tipo Rep.-

TIPOS DE DATOS ABSTRACTOS

Abstracción de tipos de datos abstractos (T.D.A)

Ejemplo.- T.D.A Racional

Especificación.- Representa a los números racionales. Tal que si num es el numerador y den el denominador el racional es num/den.

Tipo Rep.-

TIPOS DE DATOS ABSTRACTOS

Función de Abstracción.- Conecta el T.D.A dado en la especificación con el tipo rep.

$$f_A: rep \rightarrow T.D.A (\text{definido en la especificación})$$

Invariante de la representación.- Condiciones que hacen el tipo rep válido para representar el T.D.A.

Ejemplo.- T.D.A Racional

Especificación.- Representa a los números racionales. Tal que si n es el numerador y d es el denominador diremos que un racional es n/d.

Operaciones:

- Constructores
- Consultas
- Modificadores.
- Simplificación
- Operaciones de E/S

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Racional

Especificación.- Representa a los números racionales. Tal que si n es el numerador y d es el denominador diremos que un racional es n/d.

Tipo rep

```
class Racional{  
    private:  
        int n,d;  
    ....  
}
```

Función de Abstracción.- $f_A: rep \rightarrow T.D.A$ (definido en la especificación)

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Racional

Especificación.- Representa a los números racionales. Tal que si n es el numerador y d es el denominador diremos que un racional es n/d.

Tipo rep

```
class Racional{  
    private:  
        int n,d;  
    ....  
}
```

Invariante de la Representación

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Fecha

Especificación.-representa fechas del calendario occidental

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Fecha

Especificación.-representa fechas del calendario occidental

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Polinomio

Especificación.- Sucesión de reales $\{a_0, a_1, \dots, a_n\}$ que representan polinomios con coeficientes reales de tipo $a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$

Operaciones:

Constructores: por defecto (inicia al polinomio 0), por parámetros, por copia

Destructor

Consultores:

Grado del polinomio

Coeficiente del monomio de grado i

Modificadores:

Cambiar el coeficiente del monomio de grado i

Operaciones E/S

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Polinomio

Especificación.- Sucesión de reales $\{a_0, a_1, \dots, a_n\}$ que representan polinomios con coeficientes reales de tipo $a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$

Tipo rep

Class Polinomio{

 float * coef;

 int grado;

 int maxgrado;

...

}

Función de Abstracción. Sea p un objeto de tipo polinomio

$$f_A(p) = p.coef[0] + p.coef[1]x^1 + \dots + p.coef[p.grado]x^{p.grado}$$

Invariante de la representación

$$p.coef[p.grado]! = 0$$

$$p.coef[i] \forall i \text{ cumpliendo que } p.maxgrado > i > p.grado$$

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Conjunto de enteros

Especificación.- Es una colección de enteros que no pueden repetirse y están ordenados.

Operaciones:

- Constructor: por defecto, por parámetros y copia
- Destructor
- Modificadores.
 - Añadir un elemento al conjunto
 - Borrar un elemento
- Consultores:
 - Numero de elementos del conjunto
 - Saber si un determinado elemento esta
- Operaciones E/S

TIPOS DE DATOS ABSTRACTOS

Ejemplo.- T.D.A Conjunto

Especificación.- es una colección de elementos de un determinado tipo base que no pueden repetirse y están ordenados.

Tipo rep:

```
class Conjunto {  
    int * v;  
    int n;  
....  
}
```

Invariante de la representación.

- C.v tiene al menos n elementos
- C.v[i]<C.v[j] para todo i<j

Función de Abstracción

Sea C un objeto de tipo conjunto

$$f_a(C) = \{C.v[i]\}_{i=0}^{n-1}$$