

# Informática II

## Funciones amigas

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional  
Facultad Regional Córdoba  
UTN-FRC

– 2024 –

# Funciones y clases amigas

- ▶ Las funciones **friend** de una clase se definen fuera del alcance de la clase (no son miembros) pero tienen acceso a los miembros privados (y protegidos) de la clase

# Funciones y clases amigas

- ▶ Las funciones **friend** de una clase se definen fuera del alcance de la clase (no son miembros) pero tienen acceso a los miembros privados (y protegidos) de la clase
- ▶ Se puede declarar una función o una clase completa como **friend** de otra clase

# Funciones y clases amigas

- ▶ Las funciones **friend** de una clase se definen fuera del alcance de la clase (no son miembros) pero tienen acceso a los miembros privados (y protegidos) de la clase
- ▶ Se puede declarar una función o una clase completa como **friend** de otra clase

Se declara una función amiga de una clase antecediendo la palabra reservada **friend** al prototipo de la función en la definición de la clase.

# Funciones y clases amigas

- ▶ Las funciones `friend` de una clase se definen fuera del alcance de la clase (no son miembros) pero tienen acceso a los miembros privados (y protegidos) de la clase
- ▶ Se puede declarar una función o una clase completa como `friend` de otra clase

Se declara una función amiga de una clase antecediendo la palabra reservada `friend` al prototipo de la función en la definición de la clase.

Para declarar a la `ClaseDos` como amiga de la `ClaseUno`, en la definición de `ClaseUno` debe agregarse:

```
friend class ClaseDos;
```

# Funciones y clases amigas

- La amistad se gana, no se toma. Para que la clase B sea amiga de la clase A, la clase A debe declarar de manera explícita que la clase B es su amiga.

# Funciones y clases amigas

- ▶ La amistad se gana, no se toma. Para que la clase B sea amiga de la clase A, la clase A debe declarar de manera explícita que la clase B es su amiga.
- ▶ La amistad no es simétrica ni transitiva

# Funciones y clases amigas

- ▶ La amistad se gana, no se toma. Para que la clase B sea amiga de la clase A, la clase A debe declarar de manera explícita que la clase B es su amiga.
- ▶ La amistad no es simétrica ni transitiva  
(si la clase A es amiga de la clase B, y la clase B es amiga de la clase C, no se puede inferir que la clase B sea amiga de la clase A (la amistad no es simétrica), que la clase C es amiga de la clase B o que la clase A es amiga de la clase C (la amistad no es transitiva)).



# Funciones y clases amigas

- ▶ La amistad se gana, no se toma. Para que la clase B sea amiga de la clase A, la clase A debe declarar de manera explícita que la clase B es su amiga.
- ▶ La amistad no es simétrica ni transitiva  
(si la clase A es amiga de la clase B, y la clase B es amiga de la clase C, no se puede inferir que la clase B sea amiga de la clase A (la amistad no es simétrica), que la clase C es amiga de la clase B o que la clase A es amiga de la clase C (la amistad no es transitiva)).
- ▶ Aún cuando los prototipos de las funciones amigas aparecen en la definición de la clases, las mismas no son funciones miembro.

# Funciones y clases amigas

- ▶ La amistad se gana, no se toma. Para que la clase B sea amiga de la clase A, la clase A debe declarar de manera explícita que la clase B es su amiga.
- ▶ La amistad no es simétrica ni transitiva  
(si la clase A es amiga de la clase B, y la clase B es amiga de la clase C, no se puede inferir que la clase B sea amiga de la clase A (la amistad no es simétrica), que la clase C es amiga de la clase B o que la clase A es amiga de la clase C (la amistad no es transitiva)).
- ▶ Aún cuando los prototipos de las funciones amigas aparecen en la definición de la clases, las mismas no son funciones miembro.
- ▶ Algunos programadores consideran que la “*amistad*” rompe el ocultamiento de información y debilita el valor del método de diseño orientado a objetos.

# Funciones amigas – friend

---

```
1 #include <iostream>
2 using namespace std;
3
4 class Cuenta { // Clase modificada Cuenta
5     friend void estableceX(Cuenta & , int ); // Declaración de la amiga
6
7     public:
8         Cuenta() { x = 0; } // Constructor
9         void imprime() const { cout << x << endl; } // Salida
10
11     private:
12         int x; // dato miembro
13 };
14
15 void estableceX(Cuenta &c, int val) {
16     c.x = val; // legal: estableceX es una amiga de Cuenta
17 }
18
19 void noPuedeEstablecerX(Cuenta &c, int val) {
20     c.x = val; // ERROR: 'Cuenta::x' no es accesible
21 }
```

---

# Funciones amigas – friend

---

```
1 #include <iostream>
2 using namespace std;
3
4 class Cuenta { // Clase modificada Cuenta
5     friend void estableceX(Cuenta & , int ); // Declaración de la amiga
6
7     public:
8         Cuenta() { x = 0; } // Constructor
9         void imprime() const { cout << x << endl; } // Salida
10
11     private:
12         int x; // dato miembro
13 };
14
15 void estableceX(Cuenta &c, int val) {
16     c.x = val; // legal: estableceX es una amiga de Cuenta
17 }
18
19 void noPuedeEstablecerX(Cuenta &c, int val) {
20     c.x = val; // ERROR: 'Cuenta::x' no es accesible
21 }
```

---

Es recomendable definir funciones **set** como funciones miembro de la clase.

Ver código fuente ejemplo `fig17_05.cpp` y `fig17_06.cpp` (D&D 4° ed.)

