


ESTRUCTURAS DE DATOS

NOTAS SOBRE C++

# Manejo de excepciones



Manuel Montenegro Montes  
Departamento de Sistemas Informáticos y Computación  
Facultad de Informática – Universidad Complutense de Madrid



# Lanzar y capturar excepciones

Dentro de los TADS hay algunas "operaciones parciales" que no se aplican en cualquiera de los casos. Por ejemplo, en el caso de las pilas, la operación de desapilar o pop de las pilas es una operación parcial porque no se puede aplicar a cualquier tipo de pila. Utilizábamos assert, ahora pasamos a las excepciones.

No se aplica por ejemplo a las pilas vacías

# Lanzamiento de excepciones

- Se utiliza la palabra clave **throw**.
- Recibe como argumento la excepción a lanzar.
  - Puede ser un objeto (*recomendado*) o un valor básico.
- No es necesario declarar los tipos de excepciones lanzadas.

```
class division_por_cero { };  
  
double dividir(double x, double y) {  
    if (y == 0) {  
        throw division_por_cero();  
    } else {  
        return x / y;  
    }  
}
```

NO es necesario ahí poner el throws al lado de los parámetros de la función.

Esto es lo mismo que ocurre en java.

# Captura de excepciones

Esto funciona igual que en Java.

- Se utilizan bloques `try/catch`, con sintaxis similar a la de Java.
- Se permiten varios bloques `catch`, cada uno capturando un tipo distinto.
- No existe bloque `finally`.

```
try {  
    dividir(1, 0);  
} catch (division_por_cero &e) {  
    std::cout << "División por cero!" << std::endl;  
}
```

**La excepciones  
se capturan por  
referencia**

Para evitar copias del objeto  
excepción.

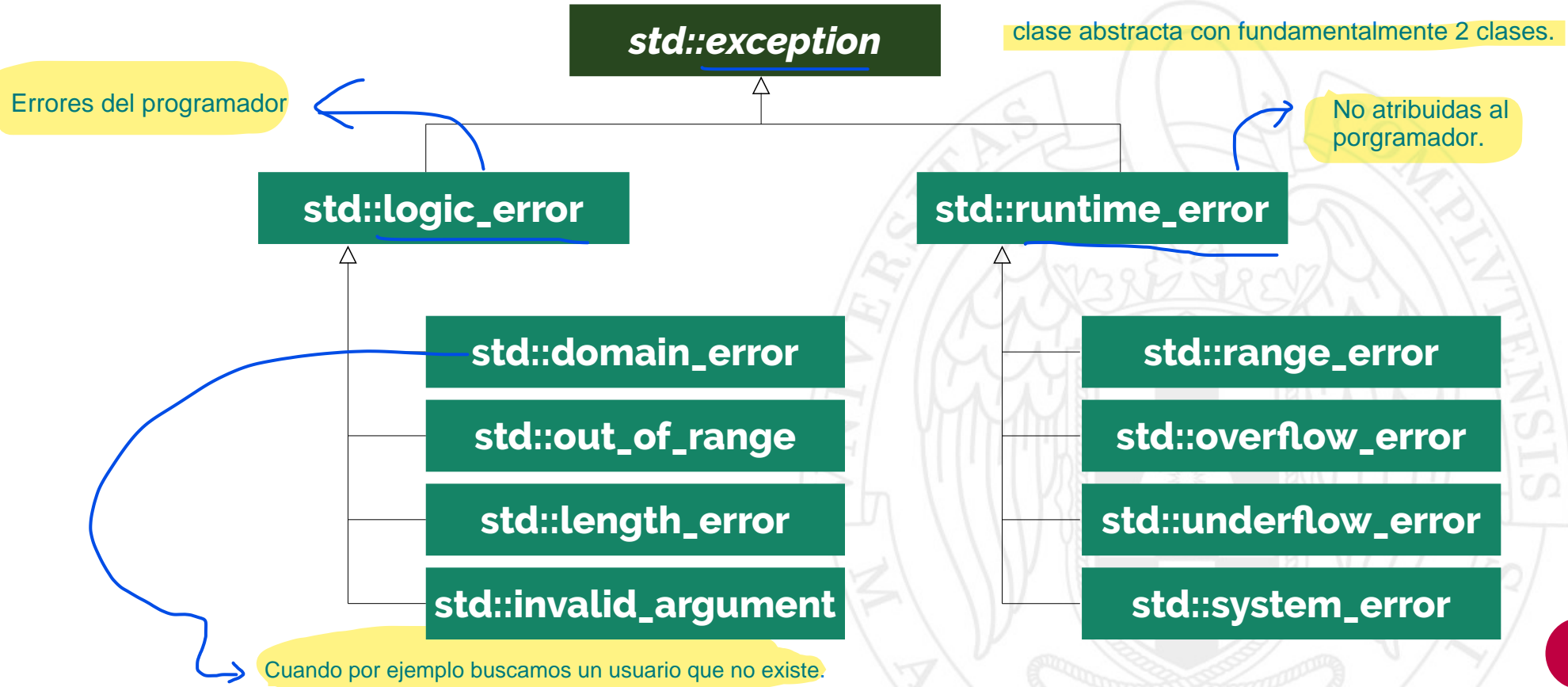
# **Jerarquía de excepciones estándar**



# Excepciones estándar

- Ficheros de cabecera `<exception>` y `<stdexcept>`.

Esto es igual que en Java.



# Excepciones estándar

- Ficheros de cabecera `<exception>` y `<stdexcept>`.

***std::exception***

const char \*what() Descripción de la excepción

Esto es similar al `getMessage()` de Java.

# Ejemplo

```
double dividir(double x, double y) {  
    if (y == 0) {  
        throw std::domain_error("división por cero");  
    } else {  
        return x / y;  
    }  
}  
  
int main() {  
    try {  
        dividir(1, 0);  
    } catch (std::exception &e) {  
        std::cout << e.what() << std::endl;  
    }  
}
```

Lanzamos una de las excepciones estándar de C++.

Mensaje de texto.

Imprime el mensaje de error.



# Heredar de excepciones estándar

```
class division_por_cero: public std::logic_error {  
public:  
    division_por_cero(): std::logic_error("división por cero") { }  
};
```

creamos esta excepción.

```
double dividir(double x, double y) {  
    if (y == 0) {  
        throw division_por_cero();  
    } else {  
        return x / y;  
    }  
}
```

```
int main() {  
    try {  
        dividir(1, 0);  
    } catch (division_por_cero &e) {  
        std::cout << e.what() << std::endl;  
    }  
}
```

**std::logic\_error**



**division\_por\_cero**