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

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;
  }
}
```

Captura de excepciones

- 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`.

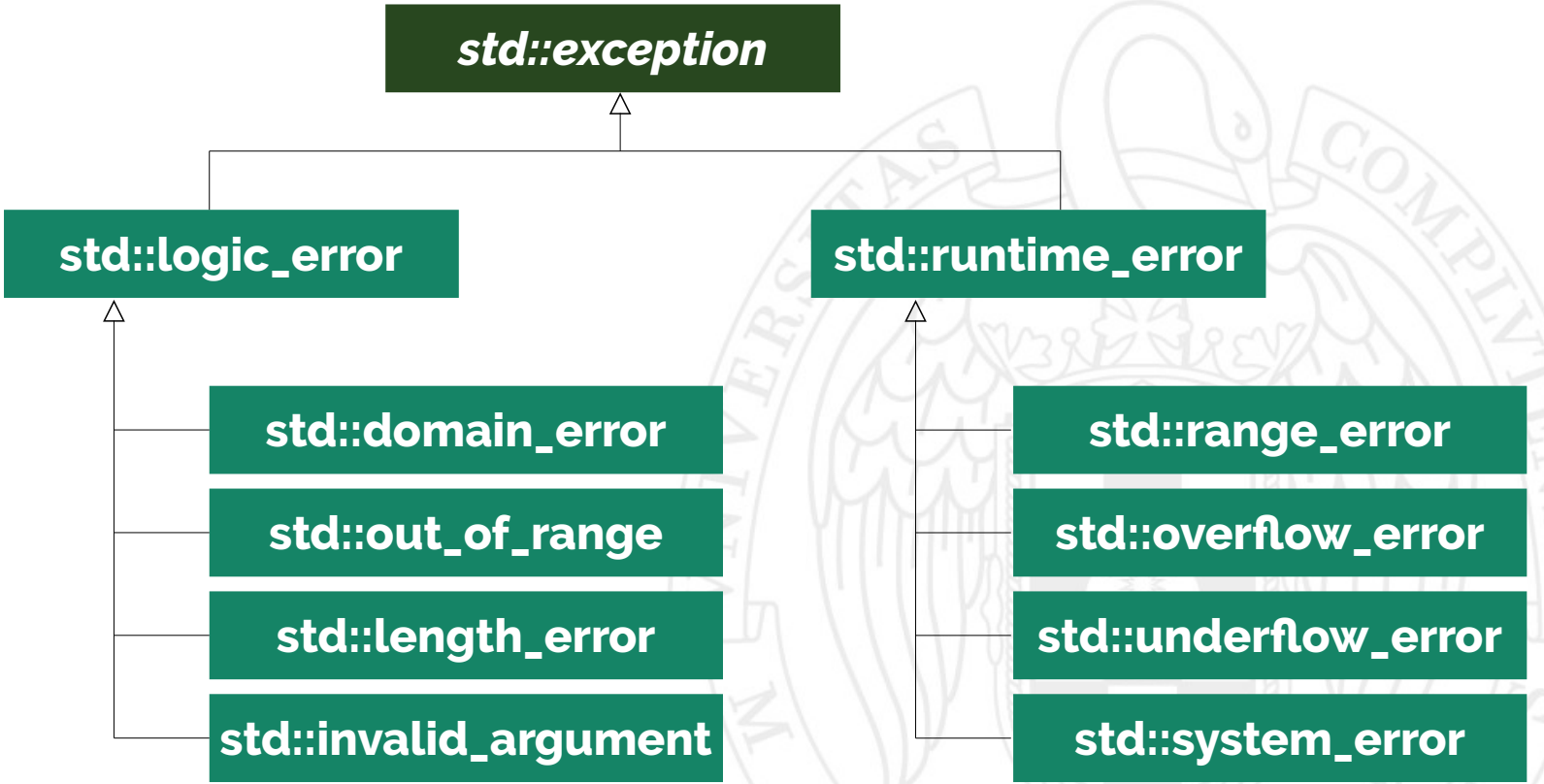
```
try {
  dividir(1, 0);
} catch (division_por_cero &e) {
  std::cout << "División por cero!" << std::endl;
}
```

La excepciones se capturan por referencia

Jerarquía de excepciones estándar

Excepciones estándar

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



Excepciones estándar

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

std::exception	
`const char *what()`	Descripción de la excepción

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;
  }
}
```

Heredar de excepciones estándar

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

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;
  }
}
```