COEN-244 Tutorial #8

March 16th, 2023

Errors and Exceptions

Exception: an event that disrupts the normal flow of the program's instructions, when the program is executing.

Error: A problem that arises due to lack of resources. Errors should be illegal during runtime (while the program is executing).

- Exceptions are left to the programmer to deal with
    - Arithmetic exceptions
    - Run-time memory allocation
    - Unprivileged access
    - Exceptions in user-defined classes
    - Illegal Inputs
- Errors are not handled during runtime so no handling:
    - They should be directly fixed in the code (Bug-free coding)

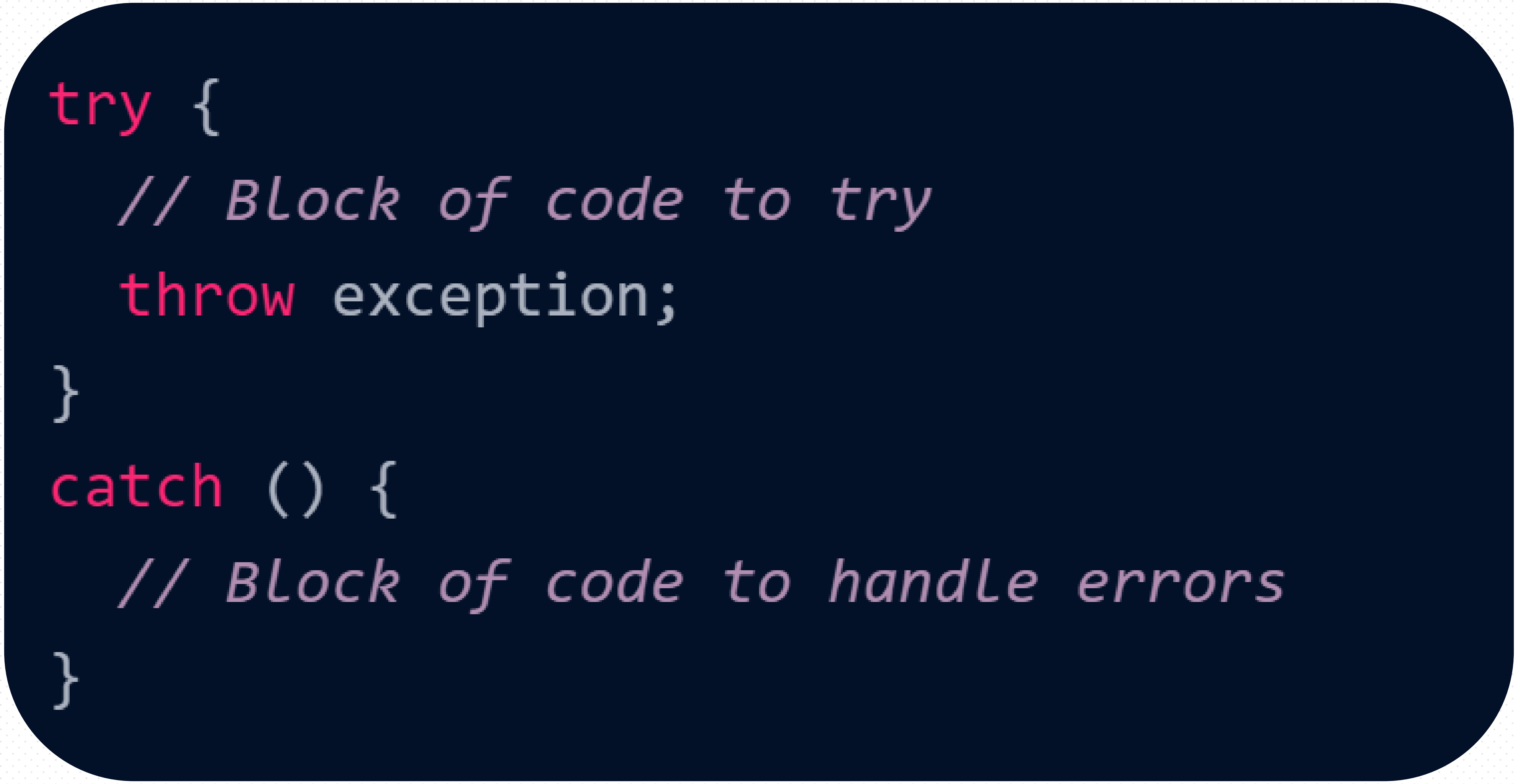
Exception Handling

- **Exception Handling** was introduced in C++ to deal with abnormal behaviour and anomalies caused during run-time.
- What happens when there is an exception but the code is fine?
- Usually the compiler has some exception handling integrated into it, **but is this ideal?**
- There are some high-level exceptions unknown to the compiler!

Ways to do Exception Handling:

- Try, Throw, and Catch (Formal Exception Handling)
- If…Else statements (Traditional Exception Handling)
- Context Jumps (Register-level Exception Handling)

Formal Exception Handling Syntax



```
try {
   // Block of code to try
   throw exception;
}
catch () {
   // Block of code to handle errors
}
```

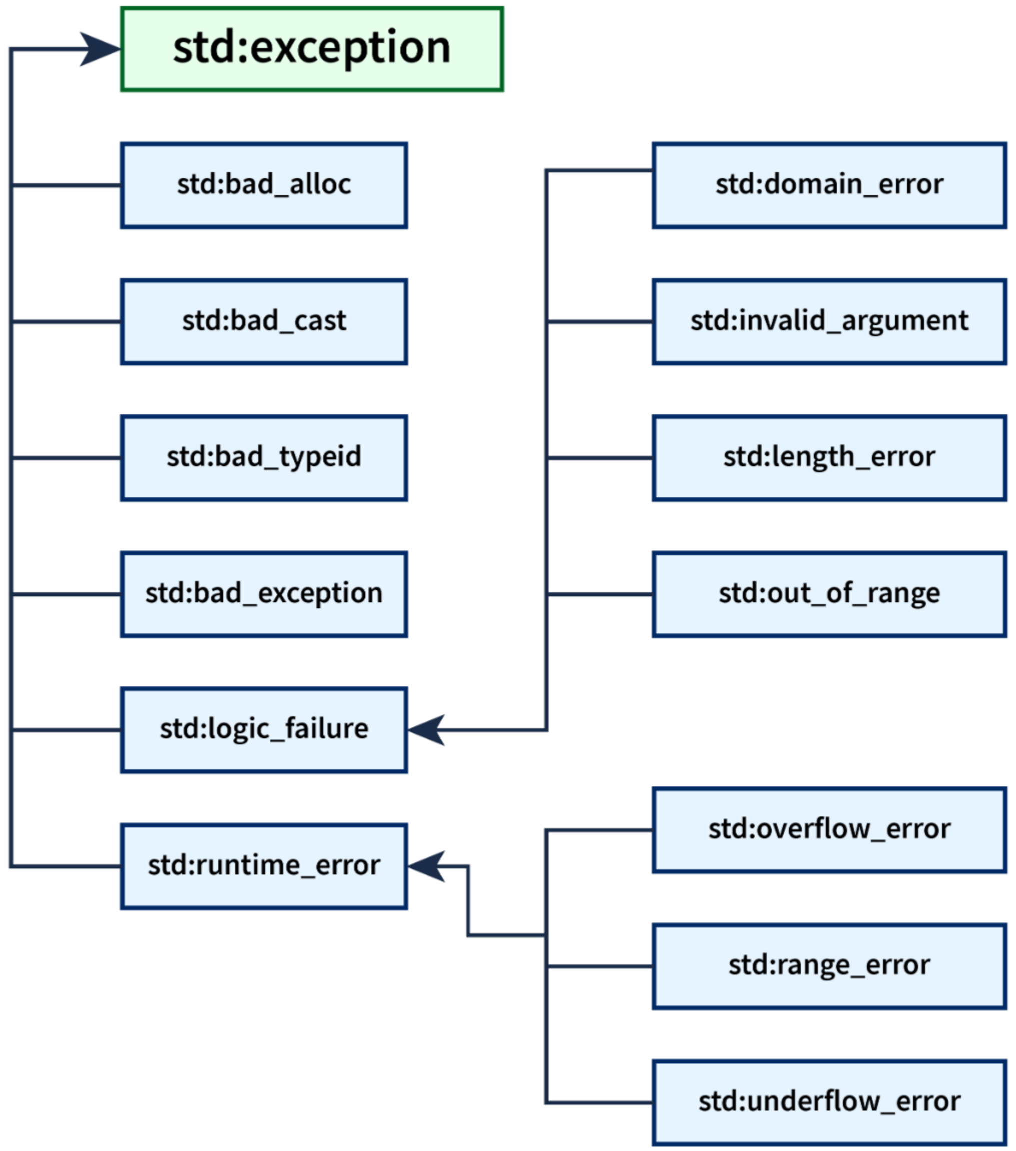
NOTE: A **function** can also throw an **Exception** but that function can only be called inside a **try…catch block**

C++ Standard Exceptions

In C++, some exceptions are given and they can be used as dictionary of exceptions that might arise.

Source:

- https://en.cppreference.com/w/cpp/error/exception
- https://www.scaler.com/topics/cpp/exception-handling-in-cpp/



User-Defined Exception Handling

- In OOP, we are interested in handling illegal behaviours that could happen in user-defined classes.
  - **E.G.,** It is illegal to add a `square` to `circle` if we have a `shape` class
  - **E.G.,** an `irrational` number cannot be a `fraction`
- A program can be used and manipulated in multiple ways:
  - Some operations or inputs cannot be tolerated
  - High-level feedback is needed and should be generated
  - Hence the user is prevented from using the program illegally
- Exceptions are **not** due to syntax errors
- In **OOP**, it is recommended to use exceptions for managing **unexpected events** as discussed above

Exercises

Exercise 1: `Age` class

- Add an exception that handles negative age.
- Add the a test case for this exception

Exercise 2: `OverSpeed` class

- Add a `car` class and throw an overspeed exception from it
- Throw the exception from a method `speed_up`

THANK YOU