File I/O operations

C++ for Developers

Exceptions

When our program runs into an error, it will throw an exception.

This is to avoid undefined behaviour like an empty username, wrong input format or wrong password for wifi.

This will “crash” the program.

try-catch

To control these exceptions we use try-catch statements. These are mostly used around I/O code to control the exceptions.

Example without try-catch:

```
std::vector<int> values = {3, 9, 11, 12};

// this will cause an out of range exception
std::cout << values.at(4) << std::endl;
```

As the vector only has the elements 0, 1, 2 and 3 - trying to access element at [4] will cause an exception.

Example with try-catch:

```
try { // body - sensetive code goes here
    std::cout << values.at(4) << std::endl;
} catch (const std::exception& e) {
    std::cout << e.what() << std::endl;
}
```

As the vector only has the elements 0, 1, 2 and 3 - trying to access element at [4] will cause an exception.

By enclosing the sensitive code in the try {} block and handling the exception thrown in the cath(){} block - we avoid crashing the program!

When to use?

Always wrap sensitive code that might throw an exception!

- Taking in user-input
- Reading a file
- Using fetched data


lafftale
for developers