Joseph Les

CS 405 Professor Hodde

Exceptions

In this week module, exception handling principles, including the use of standard exceptions, custom exceptions, and catch-all handlers. It begins by defining a custom exception class, `CustomException`, which inherits from `std::exception` and overrides the `what()` method to return a specific error message. The function `do\_even\_more\_custom\_application\_logic` throws a `std::runtime\_error` to simulate a typical error scenario. In `do\_custom\_application\_logic`, this function is called within a try-catch block that handles `std::exception` and prints the error message, then throws a `CustomException`. The `divide` function showcases handling divide-by-zero errors by throwing a `std::invalid\_argument` exception.

The `main` function is the program's beginning and uses a try-catch block to handle exceptions in a specific order: `CustomException`, `std::exception`, and a catch-all handler. This structure ensures that specific exceptions are addressed before resorting to a general catch-all handler for unforeseen errors. The `do\_division` function, marked with `noexcept`, demonstrates exception handling by catching divide-by-zero errors thrown by `divide`. Through these implementations, the program highlights best practices in error management, ensuring applications handle unexpected conditions gracefully and maintain robustness.

A screen shot of a computer code

Description automatically generated