**Lab Report**

**Exception Handling and Templates**

By: Ishaa Jain, Nipun Chandra, Shivam Kharangate

**Introduction:**

The aim of the lab is to explore and implement two essential concepts in C++ programming: exception handling and templates. Exception handling and templates are important for writing efficient, maintainable, and error-free code. Exception handling allows programs to deal with unexpected situations, while templates enable writing generic and reusable code pieces. These foundational principles are key to developing software solutions that are flexible and versatile, which are vital skills for pursuing a career in computer science or engineering. They are also important in Data Structures and Algorithms; exception handling guards against operational errors and preserves data integrity. Templates further enhance this by allowing data structures and algorithms to operate on any data type, increasing their utility and reusability.

**Task 1: Design and Implementation of a Movie Shelf Class:**

**Implementation:**

A class named **Movie\_Shelf** was created to manage a collection of movies, with the feature to add and remove movies from the shelf. The class utilizes an array to store movies and includes a counter to track the number of movies currently on the shelf. The shelf's maximum capacity is set as a constant value.

**Exception Handling:**

Two custom exception classes, **Full\_exception** and **Empty\_exception**, were introduced to handle scenarios where if the shelf is full the Full\_execption comes to play when attempting to add a movie and Empty\_exception when trying to remove a movie, from an already empty shelf.

**Task 2: Testing the Movie\_Shelf Class:**

A simple console application was developed to interact with the **Movie\_Shelf** class. It allowed adding and removing movies and displayed the current number of movies on the shelf. The class's functionality and its exception handling mechanisms were confirmed by testing.

**Task 3: Introduction of Templates with EntertainmentCollection:**

A template class **EntertainmentCollection<T>** was introduced, instead of specific classes like Movie\_shelf etc. This class was designed to manage collections of any entertainment items, that showed the flexibility of templates.

**VideoGame Class:**

To test the **EntertainmentCollection** template, a new **VideoGame** class was created, that represented video games with title and genre attributes. This showed the template’s ability to handle different types of items.

**Template Functionality and Exception Handling:**

Like the **Movie Shelf** class, **EntertainmentCollection<T>** implemented custom exceptions for collection overflow and collection underflow conditions, to ensure error-handling across different collection types.

**Concepts Explained:**

**Exception Handling:** Exception handling proved essential for managing errors that would have been caused if there was an overflow or underflow of items in a particular class, which allowed our program to respond to exceptional conditions without crashing. The custom exceptions provided clear feedback on the nature of the error, which made the code more maintainable.

**Templates:** The use of templates significantly improved the code’s reusability and flexibility. By abstracting the collection's item type, the same class could be used to manage any type of entertainment item, reducing code duplication, and increasing the software's adaptability to future requirements.

**Conclusion:**

This lab successfully demonstrated the implementation and utility of exception handling and templates. Through the design of a movie shelf and an entertainment collection, it showed how these concepts contributed to creating more flexible, and maintainable software. These practices are important for a career in Computer Science and/or engineering, as these skills necessary to tackle a wide range of programming challenges.

**Screenshots:A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A computer screen with a black screen

Description automatically generated**

**A computer screen shot of a black screen

Description automatically generated**