

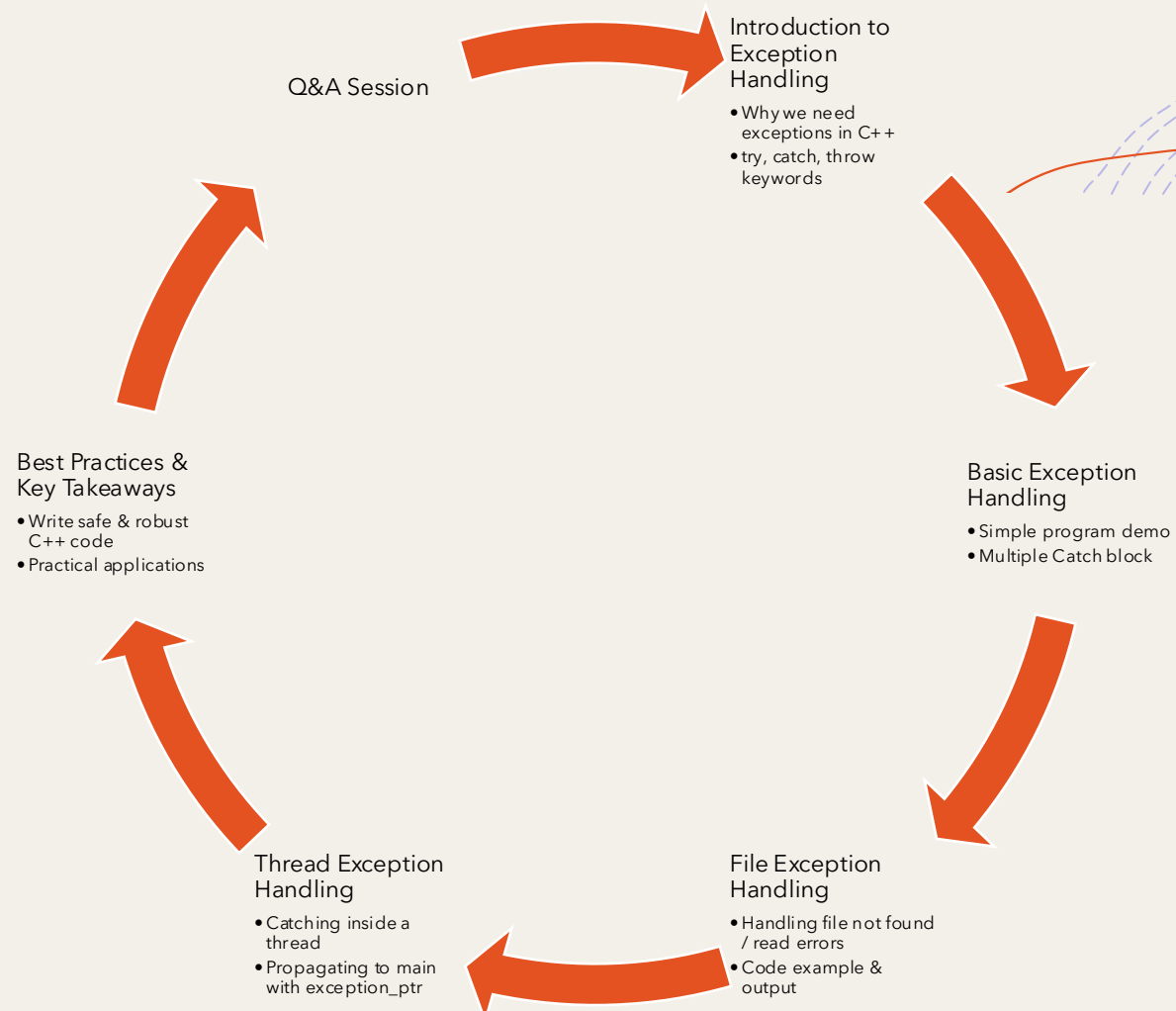
# Introduction

**Topic:** Exception Handling in C++

**Keywords:** try, catch, throw



# Agenda



## Slide 3 - Introduction to Exception Handling

### Definition:

**Exception Handling is a mechanism in C++ that allows a program to respond to runtime errors** (like invalid input, file not found, divide by zero) in a **controlled way**.

- Prevents program crashes
- Keeps code clean and readable
- Separates error-handling code from main logic

## Syntax:

```
try {  
    // Code that may throw an exception  
} catch (exception_type e) {  
    // Handle exception  
} catch (...) {  
    // Handle any type of exception  
}
```

# Basic Exception Example

```
#include <iostream>
#include <stdexcept>
using namespace std;
class Divider
{
private:
    int a, b;
public:
    void input()
    {
        cout << "Enter two numbers: ";
        cin >> a >> b;
    }
    void divide()
    {
        try
        {
            if (b == 0)
                throw runtime_error("Division by zero is not valid");

            cout << "Result: " << a / b << endl;
        }
        catch (runtime_error &e)
        {
            cerr << "Error: " << e.what() << endl;
        }
    }
};
int main()
{
    Divider d;
    d.input();
    d.divide();

    return 0;
}
```

## Explanation:

- If user enters 0 as divisor → throws exception
- Catch block prints: *"Error: Division by zero!"*

```

#include <iostream>
#include <fstream>
#include <stdexcept>
using namespace std;
class FileReader {
private:
    string filename;
public:
    FileReader(string fname) {
        filename = fname;
    }
    void readFile() {
        ifstream file(filename);
        try {
            if (!file)
                throw runtime_error("File not found!");
            string line;
            while (getline(file, line)) {
                cout << line << endl;
            }
            file.close();
        }
        catch (runtime_error &e) {
            cerr << "Exception: " << e.what() << endl;
        }
    }
};

int main() {
    FileReader reader("data.txt");
    reader.readFile();



    return 0;
}

```

Here are some typical file-related issues:

- **File not found** - trying to open a file that doesn't exist.
- **Permission denied** - insufficient permission to access a file.
- **Disk full / write error** - cannot write because disk is full.
- **File in use** - another program is using the file.
- **Invalid path** - incorrect file directory or name.

### Demo Result:

-  If file exists → prints file contents.
-  If not → "Exception: File not found!"

### C++ Library Used:

<fstream> + try-catch + runtime\_error

**Thread Exception Handling** means detecting and managing errors that occur inside threads during multithreaded program execution – so that one thread's failure doesn't crash the entire program.

A **thread** is the smallest unit of execution within a process.

- A program can run **multiple threads** simultaneously (multithreading).
- Each thread runs its own code but shares memory and resources with other threads.

**Example:**

One thread reads a file

Another thread processes data

Another thread writes the output

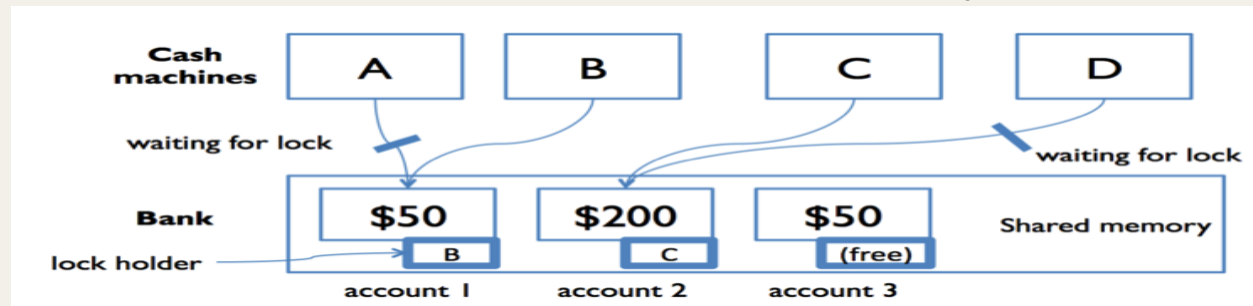
What is a Thread Exception?

A **thread exception** is an error that occurs inside a thread's execution block, such as:

- Accessing invalid memory
- Dividing by zero
- File operation failure
- Synchronization issues (race conditions, deadlocks)

If not handled properly, it can cause:

- Program crash
- Undefined behavior
- Data corruption





This program shows how to **catch exceptions inside threads**.

Each thread handles its own error using try-catch.

If  $x == 0$ , an exception is thrown and caught within the thread.

Other threads run normally without affecting the program.

```
Task running with value: 10
Thread error: Zero not allowed!
```

Catch Exception Inside Thread

```
#include <iostream>
#include <thread>
#include <stdexcept>
using namespace std;
class Task {
private:
    int value;
public:
    Task(int x) : value(x) {}
    void execute() {
        try {
            if (value == 0) {
                throw runtime_error("Zero not allowed!");
            }
            cout << "Task running with value: " << value << endl;
        } catch (exception &e) {
            cout << "Thread error: " << e.what() << endl;
        }
    }
    int getValue() const {
        return value;
    }
};

int main() {
    Task task1(10);
    Task task2(0);
    thread t1(&Task::execute, &task1);
    thread t2(&Task::execute, &task2);
    t1.join();
    t2.join();
    return 0;
}
```

```

#include <iostream>
#include <thread>
#include <exception>
#include <vector>
using namespace std;
class TaskManager {
private:
    vector<thread> threads;
    vector<exception_ptr> exceptions;
public:
    TaskManager(int count) {
        exceptions.resize(count);
    }
    void runTask(int id, exception_ptr &eptr) {
        try {
            if (id == 2)
                throw runtime_error("Thread " + to_string(id) + " failed!");
            cout << "Thread " << id << " running successfully." << endl;
        }
        catch (...) {
            eptr = current_exception();
        }
    }
    void startTasks(int count) {
        for (int i = 1; i <= count; ++i)
            threads.emplace_back(&TaskManager::runTask, this, i, ref(exceptions[i - 1]));
    }
    void finishTasks() {
        for (auto &t : threads) t.join();

        for (auto &ep : exceptions) {
            try {
                if (ep) rethrow_exception(ep);
            }
            catch (exception &e) {
                cout << "Caught in main (via TaskManager): " << e.what() << endl;
            }
        }
        cout << "All threads completed safely.\n";
    }
};

int main() {
    TaskManager manager(3);
    manager.startTasks(3);
    manager.finishTasks();
}

```

- We create **3 threads**, each running task(id, eptr).
- Thread with id == 2 deliberately throws an exception.
- exception\_ptr safely captures the exception.
- After joining, the main thread rethrows and handles all exceptions.

```

Thread 1 running successfully.
Thread 3 running successfully.
Caught in main: Thread 2 failed!
Program finished safely.

```



```

#include <iostream>
#include <stdexcept>
using namespace std;
class AgeException : public exception {
    string message;
public:
    AgeException(string msg) : message(msg) {}
    const char* what() const noexcept override {
        return message.c_str();
    }
};
class Person {
    int age;
public:
    void setAge(int a) {
        try {
            if (a < 0)
                throw AgeException("Age cannot be negative!");
            else if (a > 120)
                throw AgeException("Age seems unrealistic!");
            age = a;
            cout << "Age set to: " << age << endl;
        }
        catch (AgeException &e) {
            cout << "Custom Exception: " << e.what() << endl;
        }
    }
};
int main() {
    Person p1;
    p1.setAge(25);
    p1.setAge(-3);
    p1.setAge(150);
}

```

## •Custom Exception: AgeException

- A **user-defined exception** used to handle invalid age inputs.
- Inherits from the **std::exception** class.
- Stores a **custom error message** using a string variable.
- Overrides the **what()** function to display the error message.
- Helps make the program **more readable, reusable, and specific** in error handling.

# Comparison

Concept	File Exception	Thread Exception
Purpose	Handle file read/write errors	Handle exceptions in multithreading
Common Errors	File not found, access denied	Race conditions, bad input, Sync
Key Tools	<code>ifstream</code> , <code>ofstream</code> , <code>runtime_error</code>	<code>std::thread</code> , <code>exception_ptr</code>
Goal	Prevent file I/O crash	Prevent thread termination

# Best Practices

- + Always **validate file and input** before use
- + Use **custom exception classes** for clarity
- + For threads, **catch locally** or **use exception\_ptr**
- + Never ignore exceptions – handle gracefully
- + Log errors for debugging



# Question and Answer session?

Do you have any questions about files or thread exceptions?

**Presenter Name: Ashraful, Momin, Shoaib.**



**GitHub: shoaib3375**

# Thank You! ❤️

**If you're out of your mind...  
then a glass of fresh juice for you!" 🍹  
Stay cool, stay curious, and keep coding! 💻 ✨**

