Exception Handling

Errors -

Errors can be broadly categorized into two types. We will discuss them one by one.

- Compile Time Errors
- Run Time Errors

- **Compile Time Errors –** Errors caught during compiled time is called Compile time errors. Compile time errors include library reference; syntax error or incorrect class import.

- **Run Time Errors -** They are also known as exceptions. An exception caught during run time creates serious issues.

Exception Handling –

- Exception Handling in C++ is a process to handle runtime errors. We perform exception handling so the normal flow of the application can be maintained even after runtime errors.

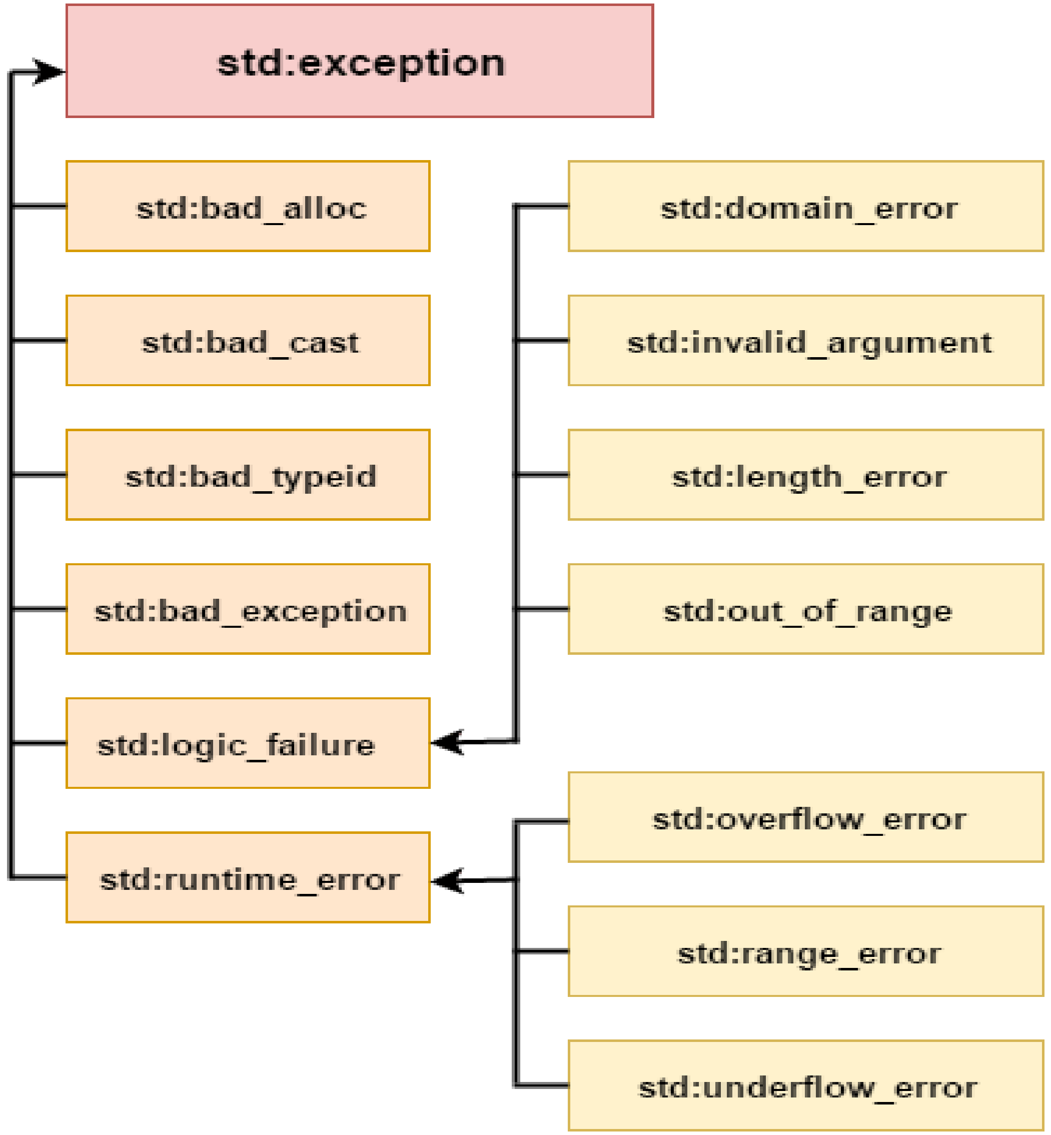
- In C++, exception is an event or object which is thrown at runtime.

- All exceptions are derived from ***std::exception*** class. It is a runtime error which can be handled. If we don't handle the exception, it prints exception message and terminates the program.

Advantage - It maintains the normal flow of the application. In such case, rest of the code is executed even after exception.

Exception Classes -

In C++ standard exceptions are defined in **<exception>** class that we can use inside our programs. The arrangement of parent-child class hierarchy is shown below –



Exception	Description
std::exception	It is an exception and parent class of all standard C++ exceptions.
std::logic_failure	It is an exception that can be detected by reading a code.
std::runtime_error	It is an exception that cannot be detected by reading a code.
std::bad_exception	It is used to handle the unexpected exceptions in a c++ program.
std::bad_cast	This exception is generally be thrown by dynamic_cast.
std::bad_typeid	This exception is generally be thrown by typeid.
std::bad_alloc	This exception is generally be thrown by new.

❑ **NOTE:** In C++, we use 3 keywords to perform exception handling -

1) try
2) catch
3) throw

Try/catch Example -

```
#include<iostream.h>
#include<conio.h>

float division(int x, int y)
{
    if( y == 0 )
    {
        throw "Attempted to divide by zero!";
    }
    return (x/y);
}
```

```
void main ()
{
    int i = 25;
    int j = 0;
    float k = 0;
    try
    {
        k = division(i, j);
        cout << k << endl;
    }
    catch (const char* e)
    {
        cerr << e << endl;
    }
}
```