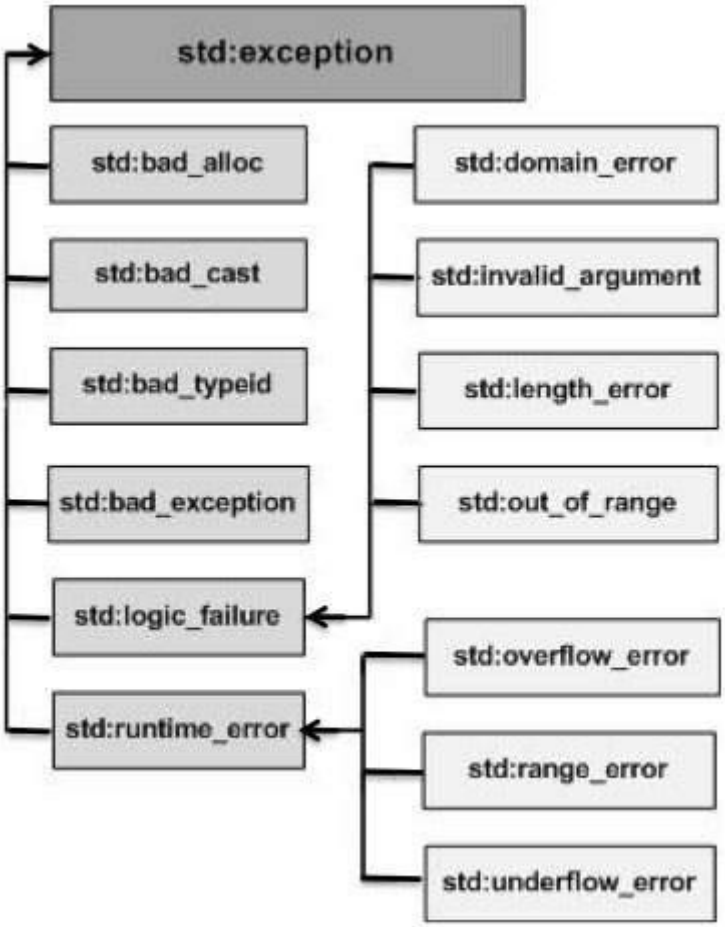
Exception Handling

C++ Standard Exceptions

C++ provides a list of standard exceptions defined in **<exception>** which we can use in our programs. These are arranged in a parent-child class hierarchy shown below −



Here is the small description of each exception mentioned in the above hierarchy –

Sr.No	Exception & Description
1	**std::exception** An exception and parent class of all the standard C++ exceptions.
2	**std::bad_alloc** This can be thrown by **new**.
3	**std::bad_cast** This can be thrown by **dynamic_cast**.
4	**std::bad_exception** This is useful device to handle unexpected exceptions in a C++ program.
5	**std::bad_typeid** This can be thrown by **typeid**.
6	**std::logic_error** An exception that theoretically can be detected by reading the code.
7	**std::domain_error** This is an exception thrown when a mathematically invalid domain is used.
8	**std::invalid_argument** This is thrown due to invalid arguments.
9	**std::length_error** This is thrown when a too big std::string is created.
10	**std::out_of_range** This can be thrown by the 'at' method, for example a std::vector and std::bitset<>::operator[]().
11	**std::runtime_error** An exception that theoretically cannot be detected by reading the code.
12	**std::overflow_error** This is thrown if a mathematical overflow occurs.
13	**std::range_error** This is occurred when you try to store a value which is out of range.
14	**std::underflow_error** This is thrown if a mathematical underflow occurs.

Define New Exceptions

You can define your own exceptions by inheriting and overriding **exception** class functionality. Following is the example, which shows how you can use std::exception class to implement your own exception in standard way −

```
#include <iostream>
#include <exception>
using namespace std;

struct MyException : public exception {
  const char * what () const throw () {
    return "C++ Exception";
  }
};

int main() {
  try {
    throw MyException();
  } catch(MyException& e) {
    std::cout << "MyException caught" << std::endl;
    std::cout << e.what() << std::endl;
  } catch(std::exception& e) {
    //Other errors
  }
}
```

This would produce the following result −

```
MyException caught
C++ Exception
```

Here, **what()** is a public method provided by exception class and it has been overridden by all the child exception classes. This returns the cause of an exception.