Object-oriented Programming

Week 11 | Lecture 1

Exception

- Exceptions are run-time anomalies or abnormal conditions that a program encounters during its execution
- Can cause your program to crash
- Examples:
  - Divide by zero
  - Bad memory allocation
  - Using an out of range index
  - Stack overflow

Exception Handling

- In order to prevent our program from abnormal termination, we can handle an exception

- To handle an exception, C++ uses the keywords ***try***, ***catch*** and ***throw***

Try Block

- We place code where an exception may (possibly) occur inside a try block
- A try block can contain a single line or as much as an entire body of a function
- It should always be immediately followed by one or more catch blocks

Catch Block

- A catch block represents a block of code that is executed when a particular exception is thrown

- As soon as the exception occurs, the program execution flows into the appropriate catch block

Catch Block

- There can be more than one catch blocks associated with a single try block
- As soon as the exception occurs, the first catch block capable of catching exception will occur
- Once the catch block finishes execution, the code following all the catch blocks is executed

Throw keyword

- We can explicitly throw an exception object from within a try block using the keyword throw

- The exception object can be a value of any primitive data type (like int or char*), a standard exception object or even an object of a user-defined class

Throw keyword

- Your code can also throw some exceptions implicitly (without using the throw keywords)
- Implicit exception objects belong to standard exceptions category
- The first catch block that is capable of accepting thrown exception object gets executed

Putting this all together

```
try
{
    // code where an exception may occur
}
catch(exceptionObType e)
{
    // code to handle exception
}
// other catch blocks (if any)
```

Example

```
try
{
    int a = 1, b = 0;
if(b == 0)
    throw 1;
else
        b = a/b;
}
catch(int e)
{
    cout << “Divide by zero exception” << endl;
}
```

Example

```
 int divide(int a, int b)
{
 if(b == 0)
      throw 1;
 else
      return a/b;
}
int main( )
{
      try { divide(50, 0); }
      catch(int e) { cout << “Divide by zero” << endl; }
}
```

Default catch block

- A default catch block can catch any type of exception object

- Must always be declared as the last catch block if there are more than one

- Syntax:

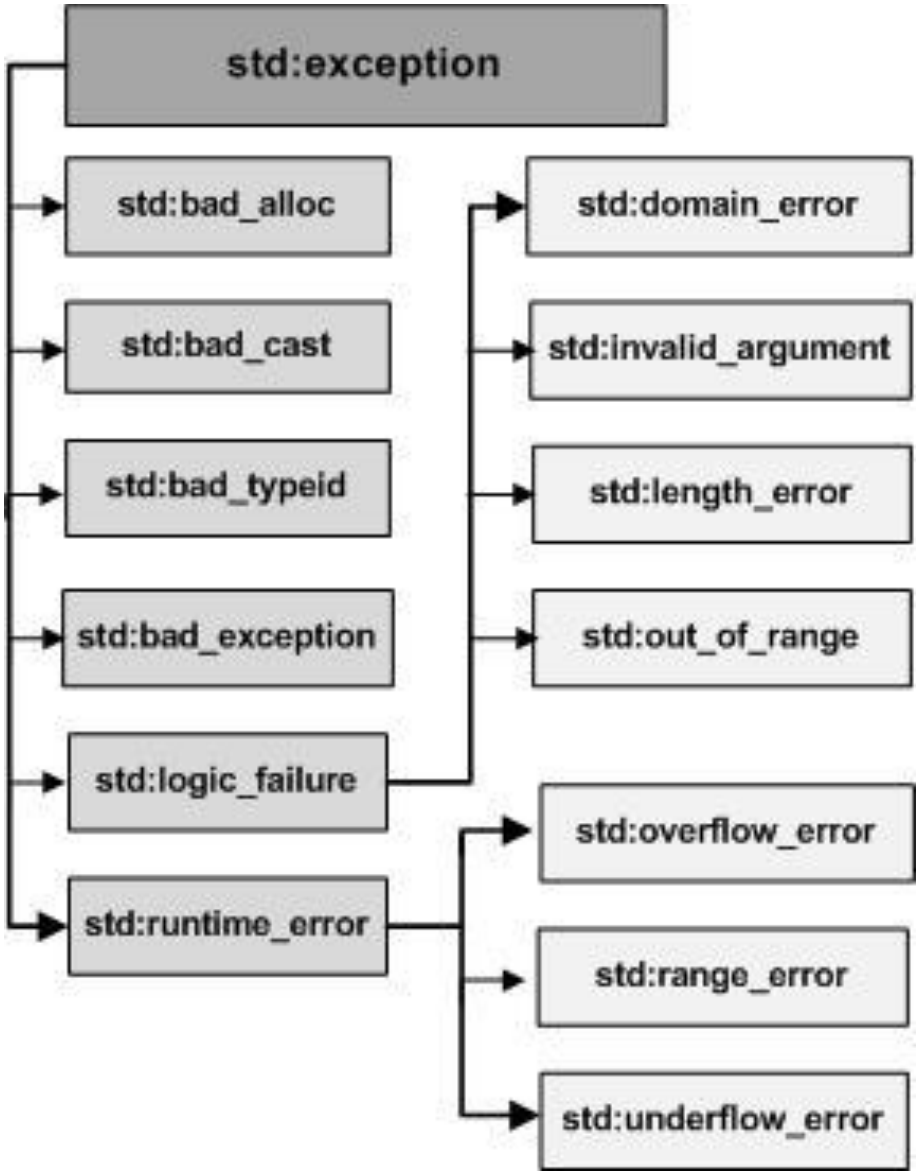
```
catch(...)
{
}
```

C++ Standard Exceptions

- C++ provides a list of standard exceptions defined in **<exception>** which we can use in our programs

- Each of the classes that inherit from parent class exception represent a different type of exception

C++ Standard Exceptions



Example

```
 int arr[10];
int index;
try
 {
 cin >> index;
 if(index < 0 || index > 10)
      throw out_of_range("Invalid index");
 else
      arr[index]  = index;
}
```

Example (cont'd)

```
 catch(out_of_range& e)
 {
     cout << e.what( ) << endl;
}

 //what( ) is a member function defined in
//exception and overriden in all standard
//exeption classes. It returns a description
//about the particular type of exception
```

Catch block with parent object

- Since exception is the parent class of all other standard exception types, we can catch any type of standard exception using a catch with *exception&* object as argument

- Example:

```
 catch(exception& e)
{
    // can catch every standard exception
}
```

Custom Exception Classes

- We can define our own custom exception class by deriving it from the class exception

- Whenever we define custom exception classes, we have to throw its exception object using the throw keyword inside try

Example

```
class MyException: public exception
{
public:
    const char* what() const throw()
    {
        return “Some special error occurred";
    }
};
    //const throw() is an exception specifier and is optional
   as //long as we are sure that an exception of type
   //MyException won’t ever occur inside the function what
```

Example (cont'd)

```
int main( )
{
    try
    {
        throw MyException( );
}
    catch(MyException& e)
{
    cout << e.what( ) << endl;
    }
}
```

Exercise

- Write a program that takes student details (ID, name, section, marks, credit hours as input. The program must allow the user to search for a student using ID and calculate its GPA. However, unless each student ID begins with K19, your program should throw an exception.

- Use a custom exception class (if you can).