Exceptions with arguments

- Exception handling in C++ consist of three keywords: **try , throw and catch** : The try statement allows you to define a block of code to be tested for errors while it is being executed.
- The throw keyword throws an exception when a problem is detected, which lets us create a custom error.

1) Separation of Error Handling code from Normal Code: In traditional error handling codes, there are always if-else conditions to handle errors. These conditions and the code to handle errors get mixed up with the normal flow. This makes the code less readable and maintainable. With try/catch blocks, the code for error handling becomes separate from the normal flow.

2) Functions/Methods can handle only the exceptions they choose: A function can throw many exceptions, but may choose to handle some of them. The other exceptions, which are thrown but not caught, can be handled by the caller. If the caller chooses not to catch them, then the exceptions are handled by the caller of the caller.
In C++, a function can specify the exceptions that it throws using the throw keyword. The caller of this function must handle the exception in some way (either by specifying it again or catching it).

```
int main(int argc, char** argv)
{
  try
  {
    // ... read image
    if (image.empty)
    {
      throw MyEmptyImageException(folderName, imageName);
    }
    // ... remained code
  }
  catch (MyEmptyImageException& meie)
  {
    std::cout << meie.what() << std::endl;
  }
  catch (std::exception& e)
  {
    std::cout << e.what() << std::endl;
  }
  return 0;
}
```

created a custom exception:

```
class MyIOException : public std::exception
{
public:
  virtual const char* what() const throw()
  {
    return "IO Exception";
  }
};
```