

Exceptions Workshop

- The BufferManager class is shown on the following slides
- Write a program to test the class
 - This should exercise the copy constructor and assignment operator
 - It should also have a suitable handler for any exceptions that might be thrown
- Check that your program runs as expected in the absence of exceptions

buffer_manager.h

```
class BufferManager {  
    private:  
        int size;  
        char *buffer;  
    public:  
        BufferManager(int size=10);  
        ~BufferManager();  
        BufferManager(const BufferManager &other);  
        BufferManager& operator =(const BufferManager& other);  
};
```

```
BufferManager::BufferManager(int size) : size(size) {  
    buffer = new char[size];  
}  
BufferManager::~~BufferManager() {  
    delete [] buffer;  
}  
BufferManager::BufferManager(const BufferManager &other) {  
    size = other.size;  
    buffer = new char[size];  
    for (int i = 0; i < size; ++i)  
        buffer[i] = other.buffer[i];  
}
```

buffer_manager.cc Contd

```
BufferManager& BufferManager::operator =(const BufferManager& other) {  
    if (&other != this) {  
        size = other.size;  
        buffer = new char[size];  
        for (int i = 0; i < size; ++i)  
            buffer[i] = other.buffer[i];  
        delete [] buffer;  
    }  
    return *this;  
}
```

Constructor

- Alter your program so that the constructor throws an `std::bad_alloc` exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
 - In C++, an object does not exist until the constructor has successfully completed
 - If the constructor throws, no object exists, and there is nothing to destroy. The destructor is not called
- Add print statements to the program and check the results

Copy Constructor

- Alter your program so that the copy constructor throws an `std::bad_alloc` exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
 - The destructor will not be called for the constructed object, because it does not exist
 - The destructor will be called for the copied object and any other objects that exist in the scope
- Add print statements to the program and check the results

Assignment Operator

- Alter your program so that the assignment operator throws an `std::bad_alloc` exception
- Check that the exception is handled correctly
- Is the class destructor called? Explain your answer
 - The destructor will be called for all objects that exist in the scope, including the ones involved in the assignment
- Add print statements to the program and check the results