

Re: is such exception handling approach good?

Re: is such exception handling approach good?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-12/msg00827.html>

- *From:* George <George@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 22 Dec 2007 06:59:01 -0800
-

Hi Alex,

In Strostrup's suggestion, he thinks both methods are ok (good pattern)?
Thanks.

regards,
George

"Alex Blekhman" wrote:

"ajk" wrote:

Say if a class initializes some memory, grabs some resource and opens an oracle database in the ctor then what should you do with the object if one of those three failed?

It depends on the nature of a class. Sometimes I need none of them in constructor; sometimes all of them.

The main purpose to do this in constructor is exactly in order to avoid undefined state of the object. When you construct it in phases, then complexity of class members increases insanely. You just can't trust yourself inside class member anymore and bound to perform tedious error prone checks for everything. It harms both performance and stability of the code, not to mention high maintenance costs.

Suppose you have the class:

```
class AllInOne
{
...

```

Re: is such exception handling approach good?

```
private:  
BYTE* m_pBuf;  
DBConn* m_pConn;  
FILE* m_fLog;  
};
```

Now you have two options:

1. Establish defined state of the object in constructor:

```
AllInOne::AllInOne() :  
m_pBuff(new BYTE[42]),  
m_pConn(OpenDB(...))  
m_fLog(fopen(...))  
{  
// do other stuff  
}
```

Then you can reliably use its methods:

```
BYTE AllInOne::GetData(int i)  
{  
return m_pBuff[i];  
}
```

```
void AllInOne::RetrieveData()  
{  
m_pConn->FillBuff(m_pBuff, 42);  
}
```

etc.

2. Create "simple" object first, then create necessary part separately:

```
AllInOne::AllInOne() :  
m_pBuff(NULL),  
m_pConn(NULL)  
m_fLog(NULL)  
{  
}
```

```
AllInOne::InitBuff()  
{  
if(!m_pBuff)  
m_pBuff = new BYTE[42];  
else  
assert("Cannot init the buffer twice!");  
}
```

Re: is such exception handling approach good?

```
AllInOne::InitDBConn()
{
if(!m_pConn)
m_pConn = OpenDB(...);
else
assert("Cannot init DB twice!");
}
```

etc.

Now all of class' methods must paranoically check internal state and hope for the best that user won't forget to call relevant initializer functions:

```
BYTE AllInOne::GetData(int i)
{
if(m_pBuff)
return m_pBuff[i];
else
// deal with error
}
```

```
void AllInOne::RetrieveData()
{
if(m_pConn)
{
if(m_pBuff)
m_pConn->FillBuff(m_pBuff, 42);
else
// deal with error
}
else
{
// deal with error
}
}
```

In my other reply I posted a link to the excerpt from Strostrup's TC++PL. He discusses the topic in details there.

Alex