



ID	CS001141	Creation date	October 14, 2008
Platform	S60 3rd Edition, FP2	Tested on devices	Nokia 6220 Classic
Category	Open C/C++	Subcategory	Files/Data

Keywords (APIs, classes, methods, functions): exception, try, catch, throw

Overview

This code snippet shows how to use the try and catch exception handling mechanism in Open C++. The traditional Symbian C++ cleanup mechanism handles exceptions in a completely different way when compared to the standard C++ try and catch mechanism. From Symbian v9.1 onwards also the try and catch mechanism has been supported, although Symbian has recommended to use the cleanup mechanism. However, when working with Open C++, the try and catch mechanism can be used.

The try and catch mechanism allows one section of an application to detect and dispatch an error condition and another section to handle it. These two sections are represented by reserved words `try` and `catch`. The `throw` statement with the data type that matches the parameter of the proper catch handler is used to jump to a catch section. The catch section will catch every type of exception if the ellipsis (...) is used. It is usually preferable to catch exceptions by reference to avoid the thrown variable to be copied onto the stack.

Note: In order to use this code, you need to install the [Open C/C++ plug-in](#).

This snippet can be self-signed.

MMP file

The following libraries are required:

```

STATICLIBRARY    libcrt0.lib

LIBRARY    libstdcpp.lib
LIBRARY    libc.lib
LIBRARY    euser.lib
  
```

Header file

```

#ifndef EXAMPLEEXCEPTION_H_
#define EXAMPLEEXCEPTION_H_

#include <exception>
#include <string>

class ExampleException : public std::exception
{
public:
    ExampleException(const std::string& name, const std::string & msg):
        message(name + "[" + msg + "]") {};
    virtual ~ExampleException() {};
    virtual const char * what() const {return message.c_str();}
private:
    std::string message;
};

class ExampleInputException : public ExampleException
{
public:
    ExampleInputException(const std::string & msg)
        : ExampleException("InputException",msg) {};
    virtual ~ExampleInputException() {};
};

class ExampleArgumentException : public ExampleException
{
public:
    ExampleArgumentException(const std::string & msg)
        : ExampleException("ArgumentException",msg) {};
    virtual ~ExampleArgumentException() {};
};

#endif /*EXAMPLEEXCEPTION_H_*/

```

Source file

```

#include "ExampleException.h"
#include <iostream> //cout, cin
#include <stdio.h> //atoi()

using namespace std;

//the example function that can throw an exception
void throw_exception_if_null(const string* text)
{
    if(!text)
        throw ExampleArgumentException("A null pointer given as a parameter");

    cout << *text << endl;
}

int main()
{

```

CS001141_-_Try_and_catch_exception_handling_in_Open_C++

```
string* str = NULL;

try
{
    int number = 0;
    string input;

    cout<<"Please, enter the number between 1 and 10)";
    getline(cin, input);

    number = atoi(input.c_str());

    if(number < 1 || number > 10)
        throw ExampleInputException("The given number is not between 1 and 10");

    if(number > 5)
        str = new string ("Hello!");

    //the str pointer refers still to NULL if the given
    //number is 1-5 -> the exception will be thrown
    throw_exception_if_null(str);

    //throw exception here anyway (values 6-10)
    throw number;

}
catch(int e) //catch by value
{
    cout << "Exception with the number: " << e << endl;
}
catch (const ExampleException& e) //catch by reference
//...or catch (const exception& e)
{
    cout << e.what() << endl;
}
catch (...)
{
    cout << "Unhandled exception";
}

if(str)
{
    delete str;
    str = NULL;
}

//getchar();

return 0;
}
```

Postconditions

The example application has prompted the user to enter a number between 1 and 10 and depending on the input, an exception has been thrown and caught.