Exercises week 2

Klaas Isaac Bijlsma
s2394480

David Vroom
s2309939

November 28, 2017

Exercise 11

Exercise 12

Exercise 13

Exercise 14

Exercise 15

Exercise 16

Exercise 17

Learn to understand how throw lists and noexept work

A small class `ShowExcepts` is made. It has a constructor and two functions `asAthrowList()` and `asNoexcept()`, that behave as if `throw (int, std::string)` or the keyword `noexcept` is specified, respectively, while this is not actually specified. When this would be speciefied, the compiler generates additional code, and this code is now already provided by us, thus simualiting the use of the specifications. The two member functions call a function `test()`, which prints that it is called and throws an exception (below the code, the output is discussed when this exception is an int or a double).

showexcepts/showexcepts.h

```
#ifndef INCLUDED_SHOWEXCEPTS_
#define INCLUDED_SHOWEXCEPTS_

class ShowExcepts
{
    void (*d_fp)();

    public:
        ShowExcepts(int value, void (*fp)());
        void asAthrowList() const ;
        void asNoexcept() const;
};

#endif
```

showexcepts/showexcepts.ih

```
#include "showexcepts.h"
#include <string>
#include <exception>

using namespace std;
```

showexcepts/showexcepts1.cc

```
#include "showexcepts.ih"

ShowExcepts::ShowExcepts(int value, void (*fp)())
:
    d_fp(fp)
{}
```

showexcepts/asathrowlist.cc

```
#include "showexcepts.ih"

void ShowExcepts::asAthrowList() const
try
{
    // function code throwing exceptions, e.g. :
    (*d_fp)();
}
catch (int)
{
    throw;
}
catch (string)
{
    throw;
}
catch (...)
{
    throw bad_exception{};
}
```

showexcepts/asnoexcept.cc

```
#include "showexcepts.ih"

void ShowExcepts::asNoexcept() const
try
{
    (*d_fp)();
}
catch (...)
{
    terminate();
}
```

main.ih

```
#include <iostream>
#include "showexcepts/showexcepts.h"
#include <exception>

using namespace std;

void test();
```

main.cc

```
#include "main.ih"

int main()
{
    ShowExcepts object(1, &test);
    try
    {
        object.asAthrowList();      // throws an exception
    }
    catch (bad_exception bad)
    {
        cout << bad.what() << '\n';
    }
    catch (...)
    {
        cout << "Caught exception in main\n";
    }
    try
    {
        object.asNoexcept();        // terminates program
    }
    catch (...)
    {
        cout << "Will not be reached\n";
    }
}
```

test.cc

```
#include "main.ih"

void test()
{
    cout << "test called\n";
    throw 1.5;          // case 1
    //throw 1;          // case 2
}
```

If `test()` throws a double (e.g. 1.5), the output shown below is produced. In `asathrowlist`, the double is not caught by the int or string catcher, but by the catch-all, throwing a `bad_exception`.

In the second try block, `asnoexcept` is called, which calls `std::terminate()`, terminating the program. Therefore, the cout statement in `main`'s final catch clause is not reached.

Output when `test()` throws a double (1.5)

```
test called
std::bad_exception
test called
terminate called after throwing an instance of 'double'
Aborted (core dumped)
```

If `test()` throws an int (e.g. 1), the output shown below is produced. In `asathrowlist`, the int is caught by the int catcher, and then rethrown. In `main`, it is then caught by the catch-all handler, giving the shown output.

Output when `test()` throws an int (1)

```
test called
Caught exception in main
```

```
test called
terminate called after throwing an instance of 'int'
Aborted (core dumped)
```

Exercise 18

Learn to identify points where exceptions may be thrown See 10.10 Annotations ;-)

Exercise 19