Previous attempt graded by FB.

Bij de eerste poging zijn er wel punten toegekend aan David, maar niet aan Klaas.

solved

Exercises week 2 - Revision

Klaas Isaac Bijlsma s2394480

David Vroom s2309939

December 6, 2017

9,5

Exercise 16

Learn how to end a program safely

How do you end a program in such a situation?

In de eerste poging genereerden we een `char const *` exception, waardoor het niet zeker was of het programma wel echt stopt.

In main wordt een object geconstrueerd en een functie aangeroepen die throwt. Deze functie doet vervolgens hetzelfde net als de functie daar weer in. In het diepste nested level wordt daadwerkelijk gethrowd, namelijk een lege, globale enum `FatalError`. Dit is om te voorkomen dat een vrij gebruikelijk exception type in tussenliggende code, eventueel externe code zoals van een library, wordt afgevangen. In main wordt deze enum exception gevangen, waardoor alle geconstueerde objecten in tussenliggende levels netjes vernietigd worden en het programma dus veilig stopt. Dit is te zien aan de output die onze main functie genereert, welke helemaal onderaan is gegeven.
Deze methode is niet volledig 'bullet proof', omdat het in theorie mogelijk is dat de enum onderweg al wordt afgevangen, bijvoorbeeld in de code van de externe library. Behalve dat er ongewenste operaties kunnen worden uitgevoerd kan het ook zo zijn dat de exception niet gerethrowd wordt, en dan komt de exception niet aan in main en stopt het programma niet. Bij een zelf gedefinieerde enum is het echter zeer onwaarschijnlijk dat dit het geval is.
Het werkt ook niet als de enum onderweg gevangen wordt door een catch-all die niet rethrowt. Echter zal dit in de praktijk niet moeten voorkomen, dit is namelijk bad practice.

Hieronder volgt onze code,

demo/demo.h

```
1  #ifndef EX16_DEMO_H
2  #define EX16_DEMO_H
3
4  class Demo
5  {
6      public:
7          Demo();
8          ~Demo();
9  };
10
11 #endif
```

demo/demo.ih

```
1  #include "demo.h"
2
3  #include <iostream>
4
5  using namespace std;
```

demo/demo.cc

```
1  #include "demo.ih"
2
3  Demo::Demo()
4  {
5      cout << "Constructor called\n";
6  }
```

demo/destructor.cc

```
1  #include "demo.ih"
2
3  Demo::~Demo()
4  {
5      cout << "Destructor called\n";
6  }
```

main.ih

```
1  #include <iostream>
2
3  #include "demo/demo.h"
4
5  using namespace std;
6
7  void function1();
8  void function2();
9  void function3();
10
11 enum FatalError
12 {};
```

main.cc

```
1  #include "main.ih"
2
3  int main()
4  try
5  {
6      Demo demo1;
7      function1();
8      cout << "Not executed\n";
9  }
10 catch (FatalError fe)
11 {
12     cout << "Program stops\n";
13 }
```

function1.cc

```
1  #include "main.ih"
2
3  void function1()
4  {
5      Demo demo2;
6      function2();
7      cout << "Not executed\n";
```

```
8 |}
```

function2.cc

```
1 |#include "main.ih"
2 |
3 |void function2()
4 |{
5 |    Demo demo3;
6 |    function3();
7 |    cout << "Not executed\n";
8 |}
```

function3.cc

```
1 |#include "main.ih"
2 |
3 |void function3()
4 |{
5 |    Demo demo4;
6 |    throw FatalError{};
7 |    cout << "Not executed\n";
8 |}
```

Output of main.cc

```
|Constructor called
|Constructor called
|Constructor called
|Constructor called
|Destructor called
|Destructor called
|Destructor called
|Destructor called
|Program stops
```