Previous attempt graded by FB.

Exercises week 1, revision

Klaas Isaac Bijlsma s2394480
David Vroom s2309939

November 29, 2017

Exercise 1

Attain some familiarity with the way functions are selected from namespaces

In the previous attempt, the answer to the last question was incorrect / incomplete.

We used the following code,

main.cc

```
1  #include <iostream>
2
3  namespace First
4  {
5      enum Enum
6      {};
7
8      void fun(First::Enum symbol)
9      {
10         std::cout << "First::fun called\n";
11     }
12 }
13
14 namespace Second
15 {
16     void fun(First::Enum symbol)
17     {
18         std::cout << "Second::fun called\n";
```

```
19      }
20  }
21
22  int main()
23  {
24      First::Enum symbol;
25
26      fun(symbol);              // First::fun called
27  }
```

Call fun and explain why `First::fun` is called. How would you call `Second::fun` instead?

Als een functie uit een namespace wordt aangeroepen zonder de namespace te speciferen, dan wordt de namespace van het argument van de functie gebruikt om de namespace van de functie te bepalen; het zogenaamde 'Koenig Lookup'. Aangezien het argument is gedeclareerd als type `First::Enum` wordt `First::fun` aangeroepen. Om `Second::fun` aan te roepen moet de namespace expliciet worden genoemd: `Second::fun(symbol)`.

In the namespaces slides (#6) it is stated that `operator<<`'s use is simplified because of the Koenig lookup. Explain.

Zonder Koenig lookup zal de korte versie `std::cout << "Hello"` (net als de lange versie `operator<<(std::cout, "Hello")`) niet gebruikt kunnen worden. De insertion operator functie uit de standard namespace is dan niet bereikbaar zonder explicite functie call `std::operator<<(std::cout, "Hello")` voor zowel de korte als de lange versie.

Now, just above main, declare a function `void fun(First::Enum symbol)`. Compile this program. What happens? Why?

Er ontstaat een foutmelding vanwege ambiguiteit. De compiler ziet zowel de functie uit namespace `First` en de globale functie als kandidaten voor de functie aanroep in main. Koenig lookup wordt hier wel gebruikt: omdat het argument uit namespace `First` komt, wordt daarin gezocht naar een passende functie `fun`, en gevonden. Daarom is deze functie een kandidaat, en de passende functie in namespace `Second` niet. Echter, het resultaat van Koenig lookup wordt gecombineerd met het resultaat van 'gewone' unqualified lookup, welke alle scopes langs gaat. Deze vindt de passende globale functie `fun`, waardoor er in totaal twee mogelijkheden zijn, resulterend in een ambiguiteit.