Programming in C/C++
Exercises set one: class templates

Christiaan Steenkist
Jaime Betancor Valado
Remco Bos

January 26, 2017

Exercise 3, custom back inserter

In this exercise we make a custom class work with the `back_inserter` iterator so we can use the `copy` generic algorithm.

Code listings

Listing 1: data.ih

```
1  #include "data.h"
2  #include <algorithm>
3  #include <iterator>
4
5  using namespace std;
```

Listing 2: data.h

```
1  #ifndef DATA_H
2  #define DTA_H
3
4  #include <vector>
5  #include <memory>
6  #include <iostream>
7
8  class Data
9  {
10   typedef std::vector<std::shared_ptr<
```

```
11      std::string>> DataVector;
12
13    DataVector d_data;
14
15    public:
16      typedef std::string value_type;
17      void push_back(std::string const &str);
18      void vecOutput();
19  };
20
21  #endif
```

Listing 3: main.cc

```
1  #include "data.ih"
2
3  int main(int argc, char **argv)
4  {
5    Data DataObj;
6    copy(istream_iterator<string>(cin),
7      istream_iterator<string>(),
8      back_inserter(DataObj));
9    DataObj.vecOutput();
10 }
```

Listing 4: pushback.cc

```
1  #include "data.ih"
2
3  void Data::push_back(string const &str)
4  {
5    shared_ptr<string> somePtr =
6      make_shared<string>(str);
7
8    d_data.push_back(somePtr);
9  }
```

Exercise 5, static polymorphism

We made a static polymorphic class that prints things!

Code listings

Listing 5: inserter.ih

```
#include "inserter.h"

using namespace std;
```

Listing 6: inserter.h

```
#ifndef INSERTER_H
#define INSERTER_H

#include <iostream>

template <typename Derived>
class Inserter
{
  private:
    std::ostream &insertInto(std::ostream &out)
    {
      return static_cast<Derived*>(this)->
        insertInto(out);
    }

  template <typename Derivative>
  friend std::ostream &operator<<(std::ostream &out,
    Inserter<Derivative> &base);
};

template <typename Derivative>
std::ostream &operator<<(std::ostream &out,
  Inserter<Derivative> &base)
{
  return base.insertInto(out);
}

#endif
```

Listing 7: main.ih

```
#include "main.h"
```

```
2
3  using namespace std;
```

Listing 8: main.h

```
1  #ifndef MAIN_H
2  #define MAIN_H
3
4  #include "inserter.h"
5
6  class IntValue : public Inserter<IntValue>
7  {
8    int d_int;
9
10   public:
11     IntValue(int someInt);
12
13   private:
14     std::ostream &insertInto(std::ostream &out);
15
16   friend Inserter;
17 };
18
19 class DoubleValue : public Inserter<DoubleValue>
20 {
21   double d_double;
22
23   public:
24     DoubleValue(double someDouble);
25
26   private:
27     std::ostream &insertInto(std::ostream &out);
28
29   friend Inserter;
30 };
31
32 #endif
```

Listing 9: main.cc

```
1  #include "main.ih"
```

```
2
3  int main(int argc, char **argv)
4  {
5    IntValue iv(12);
6    DoubleValue dv(3.14);
7
8    cout << iv << '\n';
9    cout << dv << '\n';
10 }
```

IntValue

Listing 10: intconstructor.cc

```
1  #include "main.ih"
2
3  IntValue::IntValue(int someInt)
4  :
5    d_int(someInt)
6  {
7  }
```

Listing 11: intinserter.cc

```
1  #include "main.ih"
2
3  ostream &IntValue::insertInto(ostream &out)
4  {
5    return out << d_int;
6  }
```

DoubleValue

Listing 12: doubleconstructor.cc

```
1  #include "main.ih"
2
3  DoubleValue::DoubleValue(double someDouble)
4  :
5    d_double(someDouble)
6  {
7  }
```

Listing 13: doubleinserter.cc

```
1  #include "main.ih"
2
3  ostream &DoubleValue::insertInto(ostream &out)
4  {
5    return out << d_double;
6  }
```

Exercise 6, static polymorphism contd.

Now with more inheritence?

Code listings

Listing 14: main.ih

```
1  #include "main.h"
2
3  using namespace std;
```

Listing 15: main.h

```
1  #ifndef MAIN_H
2  #define MAIN_H
3
4  #include "inserter.h"
5
6  class IntValue : public Inserter<IntValue>
7  {
8    int d_int;
9
10   public:
11     IntValue(int someInt);
12     int value();
13
14   private:
15     virtual std::ostream &insertInto(
16       std::ostream &out);
17
18   friend Inserter;
19 };
```

```
20
21  class DoubleValue : public Inserter<DoubleValue>
22  {
23    double d_double;
24
25    public:
26      DoubleValue(double someDouble);
27
28    private:
29      std::ostream &insertInto(std::ostream &out);
30
31    friend Inserter;
32  };
33
34  class LabelledInt : public IntValue
35  {
36    std::string d_label;
37
38    public:
39      LabelledInt(int someInt, std::string label);
40
41    private:
42      std::ostream &insertInto(
43        std::ostream &out) override;
44
45    friend Inserter;
46  };
47
48  #endif
```

Listing 16: main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char **argv)
4  {
5    IntValue iv(12);
6    DoubleValue dv(3.14);
7    LabelledInt li(3, "lithium");
8
9    cout << iv << '\n';
```

```
10    cout << dv << '\n';
11    cout << li << '\n';
12  }
```

LabelledInt

Listing 17: labelconstructor.cc

```
1  #include "main.ih"
2
3  LabelledInt::LabelledInt(int someInt, string label)
4  :
5    IntValue(someInt),
6    d_label(label)
7  {
8  }
```

Listing 18: labelinserter.cc

```
1  #include "main.ih"
2
3  ostream &LabelledInt::insertInto(ostream &out)
4  {
5    return out << d_label << ": " << value();
6  }
```