Programming in C/C++
Exercises set three: polymorphism

Christiaan Steenkist
Jaime Betancor Valado
Remco Bos

November 24, 2016

Exercise 15, construct ostream class

We were tasked to construct an ofstream class with our own buffer. The program should work correctly with the syntaxis in the question.

Code listings

Listing 1: main.ih

```
#include "bistream.h"
#include "bistreambuffer.h"

#include <iostream>
#include <fstream>
```

Listing 2: main.cc

```
#include "main.ih"

int main()
{
  std::ofstream one("one");
    std::ofstream two("two");

    BiStream ms(one, two);

    ms << "Hello world" << std::endl << std::flush;
}
```

Listing 3: bistream.h

```
#ifndef BISTREAM_H
#define BISTREAM_H

#include "main.ih"
class BiStream: public std::ostream
{
  public:
    BiStream(std::ofstream &one, std::ofstream &two);
    ~BiStream();
};

#endif
```

Listing 4: bistreambuffer.h

```
#ifndef BISTREAMBUFFER_H
#define BISTREAMBUFFER_H

#include "main.ih"

class BiStreamBuffer: public std::streambuf
{
  std::ostream *d_stream1, *d_stream2;

  public:
    BiStreamBuffer(std::ofstream &one, std::ofstream &
   two);
    std::streamsize xsputn(const char* s, std::
   streamsize n) override;
};

#endif
```

Listing 5: bistreamConst.cc

```
#include "main.ih"

BiStreamBuffer::BiStreamBuffer(std::ofstream &one, std
    ::ofstream &two)
```

```
:
  d_stream1(&one),
  d_stream2(&two)
{
}
```

Listing 6: bistreambufConst.cc

```
#include "main.ih"

BiStream::BiStream(std::ofstream &one, std::ofstream &
   two)
:
  std::ostream(new BiStreamBuffer(one, two))
{
}
```

Listing 7: bistreamDestr.cc

```
#include "main.ih"

BiStream::~BiStream()
{
  delete this->rdbuf();
}
```

Listing 8: xsputn.cc

```
#include "main.ih"

std::streamsize BiStreamBuffer::xsputn(const char* s,
   std::streamsize n)
{
  *d_stream1 << s;
  *d_stream2 << s;
  return 0;
}
```

Exercise 16, design streambuf

We were tasked to design a streambuf class that is called IFdStreamBuff that allows extractions from an FD.

Code listings

Listing 9: ifdstreambuf.ih

```
#include "ifdstreambuf.h"
#include <unistd.h>
#include <memory.h>

using namespace std;
```

Listing 10: ifdstream.h

```
#ifndef IFDSTREAM_H
#define IFDSTREAM_H

#include <iostream>
#include "ifdstreambuf.h"

class IFdStream: public std::istream
{
  public:
    explicit IFdStream(int FD);
    ~IFdStream();
};

#endif
```

Listing 11: ifdstreambuf.h

```
#ifndef IFDSTREAMBUF_H
#define IFDSTREAMBUF_H

#include "mode.h"
#include <streambuf>

class IFdStreambuf: public std::streambuf
{
  int d_FD;
  Mode d_mode;
  std::size_t bufferSize = 100;
  char buffer[100] {0};
  size_t place = 0;

  protected:
    int underflow() override;
    int uflow() override;
    std::streamsize xsgetn(char* s, std::streamsize n) override;

    public:
        explicit IFdStreambuf(Mode mode = KEEP_FD);
        explicit IFdStreambuf(int FD, Mode mode = KEEP_FD);
        ~IFdStreambuf();
        void close(int FD);
        void open(int FD, Mode mode = KEEP_FD);
};

#endif
```

Listing 12: close.cc

```
#include "ifdstreambuf.ih"

void IFdStreambuf::close(int FD)
{
    ::close(FD);
    // code for setting mode to CLOSE_FD here
}
```

Listing 13: cnstr1.cc

```
#include "ifdstreambuf.ih"

IFdStreambuf::IFdStreambuf(Mode mode)
:
  d_mode(mode)
{
}
```

Listing 14: cnstr2.cc

```
#include "ifdstreambuf.ih"

```

```
IFdStreambuf::IFdStreambuf(int FD, Mode mode)
:
  d_FD(FD),
  d_mode(mode)
{
  read(FD, buffer, bufferSize * sizeof(char));
}
```

Listing 15: destructor.cc

```
#include "ifdstreambuf.ih"

IFdStreambuf::~IFdStreambuf()
{
    if (d_mode == CLOSE_FD)
    close(d_FD);
}
```

Listing 16: open.cc

```
#include "ifdstreambuf.ih"

void IFdStreambuf::open(int FD, Mode mode)
{
    d_FD = FD;
    d_mode = mode;
    read(FD, buffer, bufferSize * sizeof(char));
}
```

Listing 17: uflow.cc

```
#include "ifdstreambuf.ih"

int IFdStreambuf::uflow()
{
  char output[1] = {0};
  if (place < bufferSize)
  {
    *output = *(buffer + place);
    ++place;
  }
  else
  {
    read(d_FD, output, 1 * sizeof(char));
  }
    return *output;
}
```

Listing 18: undflow.cc

```
#include "ifdstreambuf.ih"

int IFdStreambuf::underflow()
{
  if (place < bufferSize)
  {
    return *(buffer + place);
  }
  return EOF;
}
```

Listing 19: xsgetn.cc

```
#include "ifdstreambuf.ih"

std::streamsize IFdStreambuf::xsgetn(char* s, std::
    streamsize n)
{
  int size = bufferSize;
  if (n <= size)
    memcpy(s, buffer, n * sizeof(char));
  else
  {
    memcpy(s, buffer, bufferSize * sizeof(char));
    read(d_FD, s + bufferSize, (n - bufferSize) *
   sizeof(char));
  }
  return n;
}
```