Week 4

Exercise 26

../26/main.cc

```
#include "main.ih"

int main(int argc, char const **argv)
{
  Base o_base;
  Derived o_derived("text that I typed");
  Message o_message(o_base);
  o_message.show();
  Message o_message2(o_derived);
  o_message2.show();
}

//One Vtable per type that is used virtually for classes with
//virtual functions. Base and Derived are both polymorphic.
//So since we have one base object this one has 1 vtables with 1 vpointer
//to this table, we have 1 derived object which has 1 vtable and 1 vpointer
//to this vtable.The other two objects are objects that are not virtual and
//their data member is a reference to an already existing object so no new
//pointers or vtables are created.
```

../26/base/base.h

```
#ifndef INCLUDED_BASE_
#define INCLUDED_BASE_

#include <iostream>

class Base
{
  public:
    Base();
    virtual ~Base();

    void hello(std::ostream &out)
    {
      vHello(out);
    };

  private:
    virtual void vHello(std::ostream &out);
};

#endif
```

../26/base/base.ih

```
#include "base.h"

using namespace std;
```

../26/base/c_base.cc

```
#include "base.ih"

Base::Base()
//:
{
}
```

../26/base/destructor.cc

```
1  #include "base.ih"
2
3  Base::~Base()
4  {
5  }
```

../26/base/vhello.cc

```
1  #include "base.ih"
2
3  void Base::vHello(std::ostream &out)
4  {
5    out << "Hello from Base \n";
6  }
```

../26/derived/derived.h

```
1  #ifndef INCLUDED_DERIVED_
2  #define INCLUDED_DERIVED_
3
4  #include "../base/base.h"
5
6  class Derived: public Base
7  {
8          std::string d_string = 0;
9
10     public:
11         Derived(std::string input);
12
13     private:
14         void vHello(std::ostream &out) override
15         {
16           out << d_string << '\n';
17         }
18 };
19
20
21
22 #endif
```

../26/derived/derived.ih

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

../26/derived/c_derived.cc

```
1  #include "derived.ih"
2
3  Derived::Derived(string input)
4  :
5    d_string(input)
6  {
7  }
```

../26/message/message.h

```
1  #ifndef INCLUDED_MESSAGE_
2  #define INCLUDED_MESSAGE_
3
4  #include "../base/base.h"
```

```
5
6   class Message
7   {
8           Base *d_base = 0;
9       public:
10          Message(Base &input);
11          void show();
12
13      private:
14  };
15
16  #endif
```

../26/message/message.ih

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

../26/message/message.h

```
1   #ifndef INCLUDED_MESSAGE_
2   #define INCLUDED_MESSAGE_
3
4   #include "../base/base.h"
5
6   class Message
7   {
8           Base *d_base = 0;
9       public:
10          Message(Base &input);
11          void show();
12
13      private:
14  };
15
16  #endif
```

../26/message/c_message.cc

```
1   #include "message.ih"
2
3   Message::Message(Base &input)
4   :
5     d_base(&input)
6   {
7   }
```

../26/message/show.cc

```
1   #include "message.ih"
2
3   void Message::show()
4   {
5     (*d_base).hello(cout);
6   }
```

Exercise 27

../27/main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char const **argv)
4  {
5
6    Base **bp = derivedFactory(10);
7
8    delete[] bp;
9
10 }
```

../27/derivedFactory.cc

```
1  #include "main.ih"
2
3  Base **derivedFactory(size_t size)
4  {
5
6    Base **base = new Base *[size];
7    Derived *derived = new Derived [size];
8
9
10   for (size_t idx = 0; idx < size; ++idx)
11   {
12     base[idx] = &derived[idx];
13   }
14
15   delete[] derived;
16
17   return base;
18
19 }
```

../27/derived/derived.h

```
1  #ifndef INCLUDED_DERIVED_
2  #define INCLUDED_DERIVED_
3
4  #include "../base/base.h"
5
6  class Derived: public Base
7  {
8          std::string d_string = 0;
9
10     public:
11         Derived();
12         Derived(std::string input);
13
14     private:
15         void vHello(std::ostream &out) override
16         {
17           out << d_string << '\n';
18         }
19 };
20
21
22
23 #endif
```

../27/derived/c_derived2.cc

```
1  #include "derived.ih"
2
3  Derived::Derived(string input)
4  :
5    d_string(input)
6  {
7  }
```