Week 4

Exercise 32

../32/stringManip.h

```
1  #ifndef INCLUDED_STRINGMANIP_
2  #define INCLUDED_STRINGMANIP_
3
4  class StringManip
5  {
6      std::string d_source;
7
8      private:
9          StringManip(std::string source);
10
11         std::string lc() const;          // return a copy of d_source in
12                                          // lower-case chars
13         std::string uc() const;          // return a copy in upper-case
14                                          // chars
15
16         int compare(std::string &rhs);   // -1: d_source first, 0: equal
17                                          // 1: rhs first, case insensitive
18                                          // comparison.
19
20         std::string copy() const;        // return a copy of d_source
21 };
22
23 // Changes
24 // - Safer make member functions constant
25 // - Include guards
26 // - Safer to keep those functions private
27
28 #endif
```

Exercise 33

../33–34/person/person.h

```
1  // Person class: interface header
2
3  #ifndef INCLUDED_PERSON_
4  #define INCLUDED_PERSON_
5
6  #include <string>
7  #include <iostream>
8
9  class Person
10 {
11   std::string d_name;     // name of person
12   std::string d_address;  // address field
13   std::string d_phone;    // telephone number
14   size_t      d_mass; // the mass in kg.
15
16   public:
17     std::string const &name()     const;
18     std::string const &address()  const;
19     std::string const &phone()    const;
20     size_t mass()                 const;
21     // Getters
22
23     void insert(std::ostream &outputStream); // Storing data
24     void extract(std::istream &inputStream);  // Extracting data
25
26   private:
27     void setName(std::string const &name);
28     void setAddress(std::string const &address);
29     void setPhone(std::string const &phone);
30     void setMass(size_t mass);
31     // Setters
32 };
33
34 // Basic inline functions
35
36 inline void Person::setName(std::string const &name)
37 {
38   d_name = name;
39 }
40
41 inline std::string const &Person::name() const
42 {
43   return d_name;
44 }
45
46 inline void Person::setAddress(std::string const &address)
47 {
48   d_address = address;
49 }
50
51 inline std::string const &Person::address() const
52 {
53   return d_address;
54 }
55
56 // Phone number setter defined seperately
57
58 inline std::string const &Person::phone() const
59 {
60   return d_phone;
61 }
62
```

```
63  inline void Person::setMass(size_t mass)
64  {
65    d_mass = mass;
66  }
67
68  inline size_t Person::mass() const
69  {
70    return d_mass;
71  }
72
73  #endif
```

../33–34/person/person.ih

```
1  // Person class: implementation internal header
2
3  #include "person.h"
4
5  #define CERR std::cerr << __FILE__": "
6
7  using namespace std;
```

../33–34/person/extract.cc

```
1  // Person member function: extract person data from istream
2
3  #include "person.ih"
4
5  void Person::extract(istream &inputStream)
6  {
7    string inputString;
8    for (size_t index = 0; index != 3; ++index)
9    {
10     if (!getline(inputStream, inputString, ','))
11       break;
12     switch (index)                       // Assign the object variables in order
13     {
14       case 0:
15         setName(inputString);
16         break;
17       case 1:
18         setAddress(inputString);
19         break;
20       case 2:
21         setPhone(inputString);
22         break;
23     }
24   }
25   if (!getline(inputStream, inputString, '\n'))
26     return;
27   setMass(stoi(inputString));
28 }
```

../33–34/person/insert.cc

```
1  // Person member function: insert data into ostream
2
3  #include "person.ih"
4
5  void Person::insert(ostream &outputStream)
6  {
7    outputStream   << "NAME:    " << name()     << '\n'
8                   << "ADDRESS: " << address()  << '\n'
9                   << "PHONE:   " << phone()    << '\n'
10                  << "MASS:    " << mass()     << '\n';
```

```
11 }
12 // Inserts all object characteristics into ostream. It was assumed that the
13 // variable identifiers were also desirable.
```

../33–34/person/setPhone.cc

```
1  // Person member function: set phone number after verification
2
3  #include "person.ih"
4
5  void Person::setPhone(string const &phone)
6  {
7    if (phone.empty())
8      d_phone = " - not available -";
9    else if (phone.find_first_not_of("0123456789") == string::npos)
10     d_phone = phone;
11   // Switched the two options above around from the example, as an empty string
12   // will also not contain any non-numerical characters.
13   else
14     cout << "A phone number may only contain digits\n";
15 }
```

Exercise 35