Computer Science 2510 - Lab 3

Readings

- Class Notes
- Textbook: Chapters 8 and 9

Objectives

- To become familiar with functions, namespaces, classes and enumerations.

Notes

- Some of the exercises in this lab are broadly based on the "Drill" and "Exercises" sections of Chapters 8 and 9 of the textbook (Bjarne Stroustrup, *Programming - Principles and Practice Using C++*, Second edition, Addison-Wesley, 2014, ISBN 978-0-321-99278-9.)

Lab Exercises

1. Header Files

We will be using three files for this exercise: `my.h`, `my.cpp`, and `use.cpp`.

Download the header file `my.h` which contains this code:

my.h (click to download)

```
#ifndef __MY_H
#define __MY_H
void print(double);
double area(double, double);
#endif
```

Create a source code file called `my.cpp`, which needs to:

- `#include "my.h"`
- `#include "std_lib_facilities.h"`
- define `print(double d)` to print the value of `d` using `cout`
- define `area(double a, double b)` to return the value `a * b`

Create a source code file called `use.cpp`, which needs to:

- `#include "my.h"`
- define `main()` to:
    - declare a variable `myArea`
    - use `area()` to set `myArea` using input parameters (5, 20.5)
    - print the value of `myArea` using `print()`

Note that `use.cpp` does not `#include std_lib_facilities.h` as it doesn't directly use any of those facilities.

Compile the files using the command `c++14 -o areaprinter my.cpp use.cpp`. Then run `areaprinter` and confirm that the result is `102.5`.

Show Solution

2. Namespaces

Download the source code file `namespaces.cpp` which contains this code:

namespaces.cpp (click to download)

```
#include "std_lib_facilities.h"
#include "thisiswrong.h"

int main()
{
    X::var = 7;
    X::print(); // print X's var
    using namespace Y;
    var = 9;
    print(); // print Y's var
    {
        using Z::var;
        using Z::print;
        var = 11;
        print(); // print Z's var
    }
    print(); // print Y's var
    X::print(); // print X's var
}
```

If you try to compile this file right now it will fail, because the namespaces have not been defined.

So now create a file called `namespaces.h`. Create 3 namespaces in that file called `X`, `Y` and `Z`.

Each namespace needs to define:

- an integer variable called `var`
- a function called `print()` that outputs the appropriate `var` using `cout`.

See the *Jack and Jill* example in the class notes as a guide.

Compile the code using the command `c++14 -o namespaces namespaces.cpp` and run your code. *Didn't compile?* Check the `#include` statements carefully and make any necessary correction.

Show Solution

3. Creating a class: `Date`

We are going to build a `Date` class in several stages. Download these source files to begin:

chrono.h (click to download)

```
#ifndef __DATE_H
#define __DATE_H

#include "std_lib_facilities.h"

namespace Chrono
{
    class Date
    {
        int y, m, d;

    public:
        Date(int yy, int mm, int dd)
        {
            y=yy;
            m=mm;
            d=dd;
        }

        int year() { return y; }
        int month() { return m; }
        int day() { return d; }

        void addDay() { d++; }

        string toString(); // TIP: use to_string(a) to convert integer a to a string
    };
}
#endif
```

dateTester.cpp (click to download)

```
#include "std_lib_facilities.h"
#include "chrono.h"
using namespace Chrono;

int main()
{
    Date today {2019,5,30};
    Date tomorrow = today;
    tomorrow.addDay();
    cout << tomorrow.toString();
    return 0;
}
```

As you complete each stage, test your code:

- define a `Date` called `today` initialized to July 31, 2018

- define a `Date` called `tomorrow`
    - give it a value by copying `today` into it
    - increase its day by one using `addDay()`
- output `today` and `tomorrow` using `toString()`
- output `today.year()`, `today.month()`, `today.day()`

Stages:

1. implement a member function `toString()` that returns the date as a string in "YYYY-MM-DD" format
2. add a default constructor that creates a `Date` of 1970-01-01
3. rewrite `year()`, `month()` and `day()` as *const member functions*
4. create an enum called `Month`
    a. modify the `Date` class to use this enum
5. create an `InvalidDate` exception
    a. implement an `isValid()` member function that checks that a date is valid and if it isn't throws an `InvalidDate` exception
    b. use the `isValid()` function in the constructor to prevent non-valid dates being created
    c. modify the `main()` method to include a try-catch block to handle the exception
6. implement a `leapYear()` member function that checks if a given year is a leap year and returns true or false
    a. modify the `isValid()` function to include `leapYear()`
    b. try to create non-valid dates!

Show Solution

Author: Department of Computer Science, MUN (BE220531)