

CSC3002: Introduction to Computer Science

Assignment 2

Assignment description:

You should write your code for each question in several .cpp and .h file. Please pack your **whole project (or source files are acceptable) into a single .zip file**, name it using your **student ID** (e.g. if your student ID is 123456, then the file should be named as 123456.zip), and then submit the .zip file via BB. For this assignment, **it is not necessary to** use the sample project. You can create an empty project with QT Creator to start programming. Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we may check whether your program is **too similar** to your fellow students' code using BB. **Please refer to the BB system for the assignment deadline.** For each day of late submission, you will obtain late penalty in the assignment marks. If you submit more than three days later than the deadline, you will receive zero in this assignment. **Detailed description on assignment requirement is stated in the last few pages.**

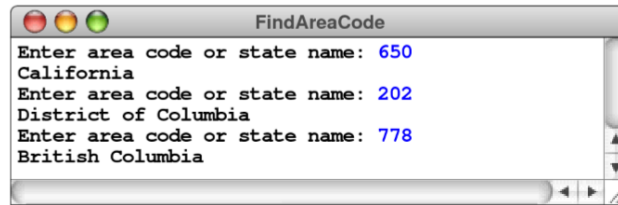
Exercise 1: Area Codes

P258(266) No.19, 20

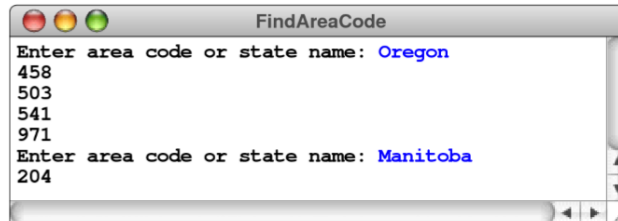
1.1 Telephone numbers in the United States and Canada are organized into various three-digit **area codes**. A single state or province will often have many area codes, but a single area code will not cross a state boundary. This rule makes it possible to list the geographical locations of each area code in a data file. For this problem, assume that you have access to the file **AreaCodes.txt**, which lists all the area codes paired with their locations as illustrated by the first few lines of that file:

```
AreaCodes.txt
201-New Jersey
202-District of Columbia
203-Connecticut
204-Manitoba
205-Alabama
206-Washington
```

Using the `AirportCodes` program as a model, write the code necessary to read this file into a `Map<int,string>`, where the key is the area code and the value is the location. Once you've read in the data, write a main program that repeatedly asks the user for an area code and then looks up the corresponding location, as illustrated in the following sample run:



As the prompt suggests, however, your program should also allow users to enter the name of a state or province and have the program list all the area codes that serve that area, as illustrated by the following sample run:



1.2 When you wrote the `FindAreaCode` program for the previous exercise, it is likely that you generated the list of area codes for a state by looping through the entire map and printing out any area codes that mapped to that state. Although this strategy is fine for small maps like the area code example, efficiency will become an issue in working with much larger data maps.

An alternative approach is to `invert` the map so that you can perform lookup operations in either direction. You can't, however, declare the inverted map as a `Map<string, int>`, because there is often more than one area code associated with a state. What you need to do instead is to make the inverted map a `Map<string, Vector>` that maps each state name to a vector of the area codes that serve that state. Rewrite the `FindAreaCode` program so that it creates an inverted map after reading in the data file and then uses that map to list the area codes for a state.

Requirement for 1.1:

```
/*
 * File: FindAreaCode.cpp
 * -----
 * This program looks up a numeric area codes for the United States
 * and Canada. The program works in both directions. If the user
 * enters a number, the program prints out the state or province to
 * which that code is assigned. If the user enters a name, it prints
 * out all the area codes assigned to that name.
 */
#include <iostream>
#include <fstream>
#include <string>
#include "error.h"
#include "map.h"
#include "simpio.h"
#include "strlib.h"
// TODO
using namespace std;
```

```

/* Function prototypes */
void readCodeFile(string filename, Map<int,string> & map);

/* Main program */
int main() {
    Map<int,string> areaCodeToState;
    readCodeFile("AreaCodes.txt", areaCodeToState);
    // TODO
    return 0;
}

/*
 * Function: readCodeFile
 * Usage: readCodeFile(filename, map);
 * -----
 * Reads a data file representing area codes and locations into the map,
 * which must be declared by the client. Each line must consist of the
 * area code, a hyphen, and the name of the state/province.
 */
void readCodeFile(string filename, Map<int,string> & map) {
    // TODO
}

```

The **AreaCode.txt** file is given. Please fill in the **TODO** and rename the file as **FindAreaCode.cpp**

Requirement for 1.2:

```

/*
 * File: InvertMap.cpp
 * -----
 * This program looks up a numeric area codes for the United States
 * and Canada. The program works in both directions. If the user
 * enters a number, the program prints out the state or province to
 * which that code is assigned. If the user enters a name, it prints
 * out all the area codes assigned to that name. This version of the
 * code inverts the map at the beginning so that queries about the
 * state names can be resolved efficiently.
 */

#include <iostream>
#include <cctype>
#include <fstream>
#include <string>
#include "error.h"
#include "map.h"
#include "simpio.h"
#include "strlib.h"
using namespace std;

```

```

// TODO

/* Function prototypes */

void readCodeFile(string filename, Map<int,string> & map);
void invertMap(Map<int,string> & areaCodeToState,
               Map< string,Vector<int> > & stateToAreaCodeList);

/* Main program */

int main() {
    Map<int,string> areaCodeToState;
    Map<string, Vector<int> > stateToAreaCodeList;
    readCodeFile("AreaCodes.txt", areaCodeToState);
    invertMap(areaCodeToState, stateToAreaCodeList);
    // TODO
    return 0;
}

/*
 * Function: readCodeFile
 * Usage: readCodeFile(filename, map);
 * -----
 * Reads a data file representing area codes and locations into the map,
 * which must be declared by the client. Each line must consist of the
 * area code, a hyphen, and the name of the state/province.
 */

void readCodeFile(string filename, Map<int,string> & map) {
    // TODO
}

/*
 * Function: invertMap
 * Usage: invertMap(areaCodeToState, stateToAreaCodeList);
 * -----
 * Fills up the stateToAreaCodeList map by linking each state
 * to a vector of all the area codes that state contains. The
 * stateToAreaCodeList map is created by the client and should
 * be empty when invertMap is called. It is interesting to note
 * that the implementation doesn't need to check whether the state
 * is already in the stateToAreaCodeList. If it isn't, selecting
 * the element creates a default value.
 */

void invertMap(Map<int,string> & areaCodeToState,
               Map< string,Vector<int> > & stateToAreaCodeList) {
    // TODO
}

```

The **AreaCode.txt** file is given. Please fill in the **TODO** and rename the file as **InvertMap.cpp**

Exercise 2: IntArray

P565(573) No.4, 5, 6 -> Treat these three problem as one exercise.

2.1 Design and implement a class called `IntArray` that implements the following methods:

- A constructor `IntArray(n)` that creates an `IntArray` object with n elements, each of which is initialized to 0.
- A destructor that frees any heap storage allocated by the `IntArray`.
- A method `size()` that returns the number of elements in the `IntArray`.
- A method `get(k)` that returns the element at position k . If k is outside the vector bounds, `get` should call `error` with an appropriate message.
- A method `put(k, value)` that assigns value to the element at position k . As with `get`, the `put` method should call `error` if k is out of bounds.

Your solution should be split into separate interface and implementation files in a manner similar to the `CharStack` example from the chapter. In the initial version of the code, you should add the necessary definitions to the `intarray.h` file to prevent clients from copying `IntArray` objects. Design and implement a unit test to check the methods exported by the class.

By keeping track of the array size and checking that index values are inside the array bounds, this simple `IntArray` class already fixes two of the most serious shortcomings of the built-in array type.

2.2 You can make the `IntArray` class from the preceding exercise look a little more like traditional arrays by overriding the bracket-selection operator, which has the following prototype:

```
int & operator [] (int k);
```

Like the `get` and `put` methods, your implementation of `operator []` should check to make sure that the index k is valid. If it is, the `operator []` method should return the element by reference so that clients can assign a new value to a selection expression.

2.3 Implement deep copying for the `IntArray` class from exercises 2.2 and 2.3.

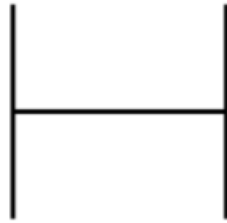
Requirement:

The **TestIntArray.cpp** file is given. Please write your **intarray.h** file and **intarray.cpp** file.

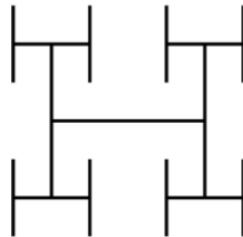
Exercise 3: H-Fractal

P385(393) No.16

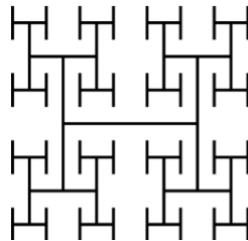
If you search the web for fractal designs, you will find many intricate wonders beyond the Koch snowflake illustrated in this chapter. H-fractal, in which the repeated pattern is shaped like an elongated letter H in which the horizontal bar and vertical lines on the sides have the same length. Thus, the order-0 Hfractal looks like this:



To create the order-1 fractal, all you do is add four new H-fractals—each one half of the original size—at each open end of the order-0 fractal, like this:



To create the order-2 fractal, all you have to do is add even smaller H-fractals (again half the size of the fractal to which they connect) to each of the open endpoints. This process gives rise to the following order-2 fractal:



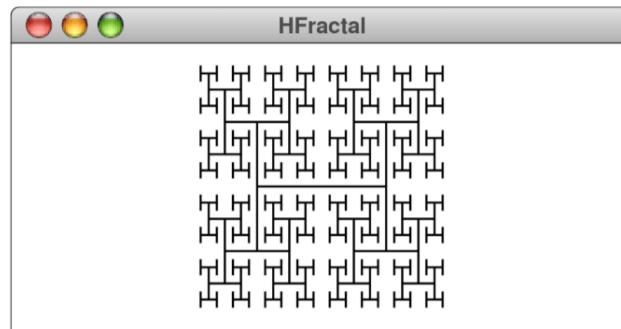
Write a recursive function

```
drawHFractal(GWindow & gw, double x, double y,  
             double size, int order);
```

where x and y are the coordinates of the center of the H-fractal, $size$ specifies the width and the height, and $order$ indicates the order of the fractal. As an example, the main program

```
int main() {  
    GWindow gw;  
    double xc = gw.getWidth() / 2;  
    double yc = gw.getHeight() / 2;  
    drawHFractal(gw, xc, yc, 100, 3);  
    return 0;  
}
```

would draw an order-3 H-fractal at the center of the graphics window, like this:



Requirement

```
/*
 * File: HFractal.cpp
 * -----
 * This program draws an H-fractal on the graphics window.int main() {
 */
#include "gwindow.h"

/* Function prototypes */
void drawHFractal(GWindow & gw, double x, double y, double size, int order);

/* Main program */
int main() {
    GWindow gw;
    double xc = gw.getWidth() / 2;
    double yc = gw.getHeight() / 2;
    drawHFractal(gw, xc, yc, 100, 3);
    return 0;
}

/*
 * Function: drawHFractal
 * Usage: drawHFractal(gw, x, y, size, order);
 * -----
 * Draws a fractal diagram consisting of an H in which each additional
 * fractal layer draws half-size fractals at the four endpoints of each H.
 */
void drawHFractal(GWindow & gw, double x, double y, double size, int order) {
    //TODO
}
```

Please fill the **TODO** and rename the file as **HFractal.cpp**.