**Andrew Tran**

**CSCI 1107 Lab Experience Four**

You are to do the following from your lab manual. If you use a word processor to reproduce the lab manual you need to also include the question with your answer.

*Do Lab 2.2. On pages 23 – 24 you are supposed to create a function called next. Change the function name to nextOne instead of next.*

Do Project 2.2 on pages 29-30. In project 2.2 you can use your overloaded extraction operator for file input since the ifstream class inherits from the istream class.

Copy and paste your program into a word document. Capture the console window for each test run and paste it below your program.

**What to turn in:**

1. Printouts of your word document and your source code.
2. A hard copy of the questions and answers from your lab manual.
3. Compress the word document and the source code from all of your programs and submit them to the D2L assignment folder called Lab Experience Four.

Lab 2.2 – Header

#include <string>

using namespace std;

enum Rock { BASALT = 0, DOLOMITE, GRANITE, GYPSUM, LIMESTONE, MARBLE, OBSIDIAN, QUARTZITE, SANDSTONE, SHALE, ROCK\_OVERFLOW };

Rock nextOne(Rock rockVal);

string kind(Rock rockVal);

Lab 2.2 – Geology

/\*--- geology.cpp ---------------------------------------------------

A program to model rocks.

Written by: Larry R. Nyhoff

Written for: Lab Manual for ADTs, Data Structures, and Problem

Solving with C++, 2E

Lab #2.2 and Project #2.2

Add your name here and other info requested by your instructor.

---------------------------------------------------------------------\*/

#include <iostream>

#include "Rock(lab2.2).h"

using namespace std;

// Represent each rock by a number

/\*const int BASALT = 0;

const int DOLOMITE = 1;

const int GRANITE = 2;

const int GYPSUM = 3;

const int LIMESTONE = 4;

const int MARBLE = 5;

const int OBSIDIAN = 6;

const int QUARTZITE = 7;

const int SANDSTONE = 8;

const int SHALE = 9;

const int ROCK\_OVERFLOW = 10;\*/

// Function prototypes go here

int main()

{

/\*\*\*\*\* PART 1 \*\*\*\*\*\*/

// Declare a variable to store a rock

Rock sample;

// Input and display various rocks

for(;;)

{

cout << "Enter a rock (" << ROCK\_OVERFLOW << " to stop): ";

int temp; cin >> temp;

sample = Rock(temp);

if (sample == ROCK\_OVERFLOW) break;

cout << "Rock is: " << sample << endl;

}

/\*\*\*\*\* PART 2 \*\*\*\*\*\*/

// Display all the rock names

cout << "\nList of rocks:\n";

for (Rock rockVal = BASALT; rockVal < ROCK\_OVERFLOW; rockVal = nextOne(rockVal))

{

cout << rockVal << " ";

cout << kind(rockVal) << endl;

}

cout << endl;

}

Lab 2.2 – Implementation

#include "Rock(lab2.2).h"

#include <string>

Rock nextOne(Rock rockVal)

{

switch (rockVal)

{

case BASALT: return DOLOMITE;

case DOLOMITE: return GRANITE;

case GRANITE: return GYPSUM;

case GYPSUM: return LIMESTONE;

case LIMESTONE: return MARBLE;

case MARBLE: return OBSIDIAN;

case OBSIDIAN: return QUARTZITE;

case QUARTZITE: return SANDSTONE;

case SANDSTONE: return SHALE;

case SHALE: return ROCK\_OVERFLOW;

default: return ROCK\_OVERFLOW;

}

}

string kind(Rock rockVal)

{

switch (rockVal)

{

case BASALT: return "Igneous";

case DOLOMITE: return "Sedimentary";

case GRANITE: return "Igneous";

case GYPSUM: return "Sedimentary";

case LIMESTONE: return "Metamorphic";

case MARBLE: return "Metamorphic";

case OBSIDIAN: return "Igneous";

case QUARTZITE: return "Sedimentary";

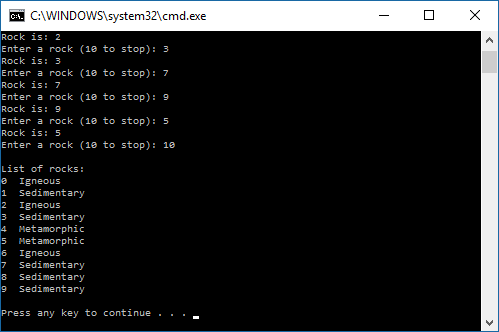
case SANDSTONE: return "Sedimentary";

case SHALE: return "Sedimentary";

default: return "Unknown";

}

}



PROJECT 2.2 – HEADER

using namespace std;

#include <string>

#include <fstream>

#include <iomanip>

enum Rock { BASALT = 0, DOLOMITE, GRANITE, GYPSUM, LIMESTONE, MARBLE, OBSIDIAN, QUARTZITE, SANDSTONE, SHALE, ROCK\_OVERFLOW};

Rock next(Rock rockVal);

string kind(Rock rockVal);

ostream & operator<<(ostream &out, Rock rockVal);

istream &operator >> (istream &in, Rock &rockVal);

void stringToUpper(string &s);

void getRock(istream &in, Rock &z);

PROJECT 2.2 – GEOLOGY

/\*--- geology.cpp ---------------------------------------------------

A program to model rocks.

Written by: Larry R. Nyhoff

Written for: Lab Manual for ADTs, Data Structures, and Problem

Solving with C++, 2E

Lab #2.2 and Project #2.2

Andrew Tran

---------------------------------------------------------------------\*/

#include <iostream>

#include <fstream>

using namespace std;

#include "Rock.h"

int main()

{

Rock sample;

ifstream inFile("Rockfile.txt");

if (!inFile) //close out if file fails

{

cout << "Error opening file" << endl;

exit(1107);

}

int count[10] = { 0 };

for(;;) //infinite loop to read file

{

inFile >> sample;

count[sample]++; //increment matching array element (the rock) detected in file

cout << "Rock counted is: " << sample << endl;

if (inFile.eof()) //only break if it reaches end of file

break;

}

cout << endl;

for (int i = 0; i < 10; i++) //print histogram

{

int n = count[i];

string s(n, 'X'); //determines number of "x"

cout << static\_cast<Rock>(i) << " " << s << " " << "(" << n << ")" << endl;

}

}

PROJECT 2.2 – IMPLEMENTATION

#include <iostream>

using namespace std;

#include <string>

#include "Rock.h"

// overloaded output operator

ostream &operator<<(ostream &out, Rock rockVal)

{

switch (rockVal)

{

case BASALT: cout << "BASALT";

break;

case DOLOMITE: cout << "DOLOMITE";

break;

case GRANITE: cout << "GRANITE";

break;

case GYPSUM: cout << "GYPSUM";

break;

case LIMESTONE: cout << "LIMESTONE";

break;

case MARBLE: cout << "MARBLE";

break;

case OBSIDIAN: cout << "OBSIDIAN";

break;

case QUARTZITE: cout << "QUARTZITE";

break;

case SANDSTONE: cout << "SANDSTONE";

break;

case SHALE: cout << "SHALE";

break;

default: cout << "Unknown";

}

return out;

}

//overloaded input operator

istream &operator>>(istream &in, Rock &rockVal)

{

getRock(in, rockVal);

return in;

}

//matcher function to read inputs

void getRock(istream &in, Rock &z)

{

void stringToUpper(string &s);

string allowedRock[] = { "BASALT","DOLOMITE","GRANITE","GYPSUM","LIMESTONE","MARBLE","OBSIDIAN","QUARTZITE","SANDSTONE","SHALE"};

string rock;

int location = 0;

bool found = false;

in >> rock;

stringToUpper(rock);

for (; !found && location < 10; location++)

found = rock == allowedRock[location];

location--;

if (found)

z = static\_cast<Rock>(location);

}

//capitalizes user input to read

void stringToUpper(string &s)

{

for (unsigned i = 0; i < s.length(); i++)

s.at(i) = toupper(s.at(i));

}

Rock next(Rock rockVal)

{

switch (rockVal)

{

case BASALT: return DOLOMITE;

case DOLOMITE: return GRANITE;

case GRANITE: return GYPSUM;

case GYPSUM: return LIMESTONE;

case LIMESTONE: return MARBLE;

case MARBLE: return OBSIDIAN;

case OBSIDIAN: return QUARTZITE;

case QUARTZITE: return SANDSTONE;

case SANDSTONE: return SHALE;

case SHALE: return ROCK\_OVERFLOW;

default: return ROCK\_OVERFLOW;

}

}

string kind(Rock rockVal)

{

switch (rockVal)

{

case BASALT: return "Igneous";

case DOLOMITE: return "Sedimentary";

case GRANITE: return "Igneous";

case GYPSUM: return "Sedimentary";

case LIMESTONE: return "Metamorphic";

case MARBLE: return "Metamorphic";

case OBSIDIAN: return "Igneous";

case QUARTZITE: return "Sedimentary";

case SANDSTONE: return "Sedimentary";

case SHALE: return "Sedimentary";

default: return "Unknown";

}

}

