|  |  |  |
| --- | --- | --- |
| American University of SharjahCollege of Engineering Department of Computer Science & Engineering  P. O. Box 26666  Sharjah, UAE |  | **Lab Instructor:** Eng. Sameer Alawnah  **Office:** EB2-101  **Phone**: 971-6-5152974  **e-mail**: salawnah@aus.edu  **Semester**: Fall 16 |

**CMP 220L – Introduction to Computer Science II**

**Lab 6**

**Note: The good programmer adds comments to his/her code. Add comments to your code.**

**Question 1:**

Given the following class definition:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

American University of Sharjah ( AUS )

Dept of Computer Science & Eng.

P.O.Box 26666

Sharjah, UAE

Fall 2016

CMP220L

LAB5 – Q1

Revision: 04.

Date: 9th of October, 2015. 12:23 pm.

By: Eng. Sameer A. Alawnah

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

class Color

{

private:

int red, green, blue;

public:

Color(){ red = 0; green = 0; blue = 0; }

Color(int r,int g,int b):red(r),green(g),blue(b){}

void setRed(int r){ red = r; }

void setGreen(int g){ green = g; };

void setBlue(int b){ blue = b; }

int getRed(){ return red; }

int getGreen(){ return green; }

int getBlue(){ return blue; }

};

Color addition or subtracting is done by adding/subtracting each component of the first color with/from the corresponding component of the second color, red with red, green with green and blue with blue.

Since the color components ranges from 0 to 255, if you got a result greater than 255 then you have to truncate it to 255, if the result is less than 0 then you have to truncate it back to 0.

1. Add the following operators to the code (using friend functions):
   1. Add operator (+) to add two colors.
   2. Add operator (-) to subtract two colors.
   3. Unary (-) operator. Assume that the (-colorComponent is equal to 255 – colorComponent, ex. if the color is [R:122,G:245,B:100] then –color is [R:133,G:10,B:155].
   4. == Operator. (Returns true if all colors components of both colors are equal, false otherwise.
   5. The >> operator to read the color components from the input stream
   6. The << operator to print the color component to the output stream in this format [red, green, blue]

Test your class using the main() function included at the end of this question.

#include<iostream>

using namespace std;

class Color

{

friend Color operator+(Color col1, Color col2);

friend Color operator-(Color c1, Color c2);

friend bool operator==(Color col1, Color col2);

friend Color operator-(Color c1);

friend istream &operator>>(istream &cin, Color &col1);

friend ostream &operator<<(ostream &cout, Color &col1);

private:

int red, green, blue;

int checkAndTruncate(int x);

public:

Color(){ red = 0; green = 0; blue = 0; }

void setRed(int r){ red = checkAndTruncate(r); }

void setGreen(int g){ green = checkAndTruncate(g); };

void setBlue(int b){ blue = checkAndTruncate(b); }

int getRed(){ return red; }

int getGreen(){ return green; }

int getBlue(){ return blue; }

};

int Color::checkAndTruncate(int x)

{

if (x >= 0 && x <= 255)

return x;

else if (x < 0)

{

return 0;

}

else

{

return 255;

}

}

istream &operator>>(istream &cin, Color &col1)

{

cin >> col1.red;

cin >> col1.green;

cin >> col1.blue;

return (cin);

}

ostream &operator<<(ostream &cout, Color &col1)

{

cout << "R[" << col1.red << "] G[" << col1.green << "] B[" << col1.blue << "]";

return (cout);

}

Color operator+(Color col1, Color col2)

{

Color res;

res.setRed(col1.red + col2.red);

res.setGreen(col1.green + col2.green);

res.setBlue(col1.blue + col2.blue);

return res;

}

Color operator-(Color col1, Color col2)

{

Color res;

res.setRed(col1.red - col2.red);

res.setGreen(col1.green - col2.green);

res.setBlue(col1.blue - col2.blue);

return res;

}

Color operator-(Color col1)

{

Color res;

res.setRed(255 - col1.red);

res.setGreen(255 - col1.green);

res.setBlue(255 - col1.blue);

return res;

}

bool operator==(Color col1, Color col2)

{

return (col1.red == col2.red) && (col1.green == col2.green) && (col1.blue == col2.blue);

}

int main()

{

Color col1;

Color col2;

cout<<"Please enter the first color (red green blue):";

cin>>col1;

cout<<"Please enter the second color (red green blue):";

cin>>col2;

Color col3 = col1+col2;

cout<<col1<<" + "<<col2<<" = "<<col3<<endl;

Color col4 = col1 - col2;

cout<<col1<<" - "<<col2<<" = "<<col4<<endl;

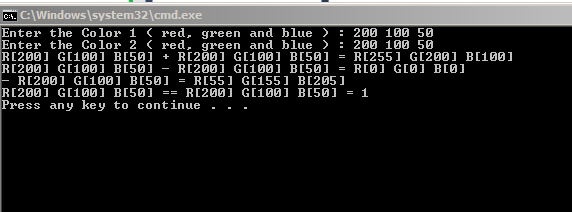
Color col5 = -col1;

cout<<"-"<<col1<<" = "<<col5<<endl;

cout<<col1<<" == "<<col2<<" = "<<(col1==col2)<<endl;

return 0;

}



1. Implement the operators in points a.1, a.2, a.3 and a.4 using member functions (keep << and >> as it is) and test your class using the following main() function.

int main(){

Color col1;

Color col2;

cout<<"Please enter the first color (red green blue):";

cin>>col1;

cout<<"Please enter the second color (red green blue):";

cin>>col2;

Color col3 = col1+col2;

cout<<col1<<" + "<<col2<<" = "<<col3<<endl;

Color col4 = col1 - col2;

cout<<col1<<" - "<<col2<<" = "<<col4<<endl;

Color col5 = -col1;

cout<<"-"<<col1<<" = "<<col5<<endl;

cout<<col1<<" == "<<col2<<" = "<<(col1==col2)<<endl;

return 0;

}

#include<iostream>

using namespace std;

class Color

{

friend istream &operator>>(istream &cin, Color &col1);

friend ostream &operator<<(ostream &cout, Color &col1);

private:

int red, green, blue;

int checkAndTruncate(int x);

public:

Color(){ red = 0; green = 0; blue = 0; }

void setRed(int r){ red = checkAndTruncate(r); }

void setGreen(int g){ green = checkAndTruncate(g); };

void setBlue(int b){ blue = checkAndTruncate(b); }

int getRed(){ return red; }

int getGreen(){ return green; }

int getBlue(){ return blue; }

Color operator+(Color col1)

{

Color c3;

c3.red = red + col1.red;

c3.green = green + col1.green;

c3.blue = blue + col1.blue;

return (c3);

}

Color operator-(Color col1)

{

Color c3;

c3.red = red - col1.red;

c3.green = green - col1.green;

c3.blue = blue - col1.blue;

return (c3);

}

Color operator-()

{

Color c3;

c3.red = 255 - red;

c3.green = 255 - green;

c3.blue = 255 - blue;

return (c3);

}

bool operator==(Color col1)

{

return ((red == col1.red) && (blue == col1.blue) && (green == col1.green));

}

};

int Color::checkAndTruncate(int x)

{

if (x >= 0 && x <= 255)

return x;

else if (x < 0)

{

return 0;

}

else

{

return 255;

}

}

istream &operator>>(istream &cin, Color &col1)

{

cin >> col1.red;

cin >> col1.green;

cin >> col1.blue;

return (cin);

}

ostream &operator<<(ostream &cout, Color &col1)

{

cout << "R[" << col1.red << "] G[" << col1.green << "] B[" << col1.blue << "]";

return (cout);

}

int main(){

Color col1;

Color col2;

cout << "Please enter the first color (red green blue):";

cin >> col1;

cout << "Please enter the second color (red green blue):";

cin >> col2;

cout << col1 << " + " << col2 << " = " << col1 + col2 << endl;

cout << col1 << " - " << col2 << " = " << col1 - col2 << endl;

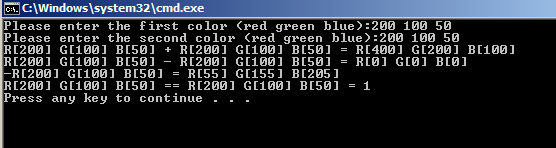
Color col5 = -col1;

cout << "-" << col1 << " = " << col5 << endl;

cout << col1 << " == " << col2 << " = " << (col1 == col2) << endl;

return 0;

}



Note: you are supposed to write 2 different programs (one for a and the other for b)

**Question 2:**

This question is based on Question 2 of lab5, you have to copy the solution from the solution posted on iLearn and do the following modifications.

1. Remove the ContactList(string filename) constructor from the ContactList class.
2. Remove the print() function from the Contact class.
3. Add the >> operator to the Contact class to read the contact from the input stream.
4. Add the << operator to the Contact class to print the contact information to the output stream.
5. Add the >> operator the ContactList class to read the contacts from the input stream.
6. Redo everything required by question 2 of lab 5 but with the following constraints:
   1. You can only use the >> operators to read from the input stream.
   2. You can only use the << operator to print a contact.

Note: you have to open any required input/output stream in the main().

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

using namespace std;

class Contact{

private:

string fname, lname;

int phone;

public:

string getFirstName(){

return fname;

} // get the first name of the contact and return it.

string getLastName(){

return lname;

} // get the last name of the contact and return it.

int getPhone(){

return phone;

}// get the phone # of the contact and return it.

string getFullName(){

return fname + " " + lname;

}// get the full name of the contact and return it.

friend istream& operator>>(istream &cin, Contact &cnt);

friend ostream& operator<<(ostream &cout, Contact &cnt);

};

istream& operator>>(istream &cin, Contact &cnt)

{

cin >> cnt.fname >> cnt.lname >> cnt.phone;

return(cin);

}

ostream& operator<<(ostream &cout, Contact &cnt)

{

cout << cnt.fname << " " << cnt.lname << " " << cnt.phone << endl;

return (cout);

}

class ContactList{

private:

vector<Contact> list; // vector of contacts

public:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Constructor with parameter ( fileName).

This constructor should read the file

given in fileName and save the content

into the list.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// a function to add contact con to list.

// return the contact at index, exit the program with error if index is out of range

Contact getContactAt(int index);

//return the index of contact in which firstName is equal to target, -1 otherwise.

int searchByFristName(string target);

friend istream& operator>>(istream &cin, ContactList &cnt);

};

istream& operator>>(istream &cin, ContactList &cnt)

{

Contact c;

cin >> c;

while (!cin.eof())

{

cnt.list.push\_back(c);

cin >> c;

}

return (cin);

}

Contact ContactList::getContactAt(int index)

{

if (index < 0 || index >= list.size())

{

cout << "Index out of range";

exit(-1);

}

return list[index];

}

int ContactList::searchByFristName(string target)

{

for (int i = 0; i < list.size(); i++)

{

if (list[i].getFirstName() == target)

return i;

}

return -1;

}

int main()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Task 1: Implement all the functions in the

two classes above.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Task 2: create an object of type contact

list with the data loaded from the file

inputs.txt using the constructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string FileName;

cout << "Enter name of file: ";

cin >> FileName;

ContactList cList;

ifstream in(FileName);

if (in.fail())

{

exit(1);

}

in >> cList;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Task 3: ask the user for the target he want to search for (search by first name ) and print

the found contact if exist, error message otherwise.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string target;

cout << "[Searching by first Name]\nPlease enter the target :";

cin >> target;

int index = cList.searchByFristName(target);

if (index == -1)

{

cout << "No Contact Found \n";

}

else

{

cout << cList.getContactAt(index);

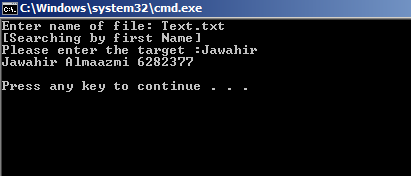
cout << endl;

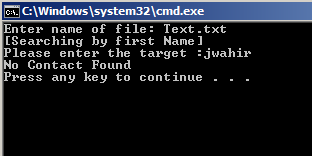
}

return 0;

in.close();

}





Good Luck ☺