|  |  |  |
| --- | --- | --- |
| American University of SharjahCollege of Engineering Department of Computer Science & Engineering  P. O. Box 26666  Sharjah, UAE |  | Instructor: Dr. Aliaa Moualla **Lab Instructor**: Eng. Sameer Alawnah  **Office**: EB1-0012C  **Phone**: 971-6-515-4940  **e-mail**: salawnah@aus.edu  **Semester**: Spring 2024 |

**CMP 220L – Programming II**

**Lab 6: Operator overloading**

Given the following class definition:

#include<iostream>

#include<sstream>

using namespace std;

class Color

{

private:

int red, green, blue;

int checkAndTruncate(int)const ;

public:

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

Color(int x) { red = green = blue = checkAndTruncate(x); }

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

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

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

int getRed() const { return red; }

int getGreen() const { return green; }

int getBlue() const { return blue; }

istream& input(istream& in);

ostream& output(ostream& out);

};

int Color::checkAndTruncate(int x)const

{

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

return x;

else if (x < 0)

{

return 0;

}

else

{

return 255;

}

}

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.

**Question 1:**

Implement the following two member functions:

1. input : this function will read the color’s components; red, green and blue from the input stream.
2. output : this function will print the color’s components to the output stream

Test your code with following main (you are not allowed to change it):

int main()

{

stringstream sin("300 -100 134 140 145 150");

Color col;

cout<<"The colors are :\n";

while(col.input(sin))

{

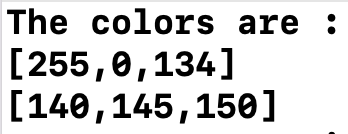
col.output(cout)<<endl;

}

return 0;

}

A sample run is shown below:



#include <iostream>

#include <sstream>

**using** **namespace** std;

**class** Color

{

**private**:

**int** red, green, blue;

**int** checkAndTruncate(**int**) **const**;

**public**:

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

Color(**int** x) { red = green = blue = checkAndTruncate(x); }

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

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

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

**int** getRed() **const** { **return** red; }

**int** getGreen() **const** { **return** green; }

**int** getBlue() **const** { **return** blue; }

istream &input(istream &in);

ostream &output(ostream &out);

};

**int** Color::checkAndTruncate(**int** x) **const**

{

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

**return** x;

**else** **if** (x < 0)

{

**return** 0;

}

**else**

{

**return** 255;

}

}

istream &Color::input(istream &in)

{

**int** r, g, b;

**if** (in >> r >> g >> b)

{

setRed(r);

setGreen(g);

setBlue(b);

**return** in;

}

**return** in;

}

ostream &Color::output(ostream &out)

{

**return** out << "[" << getRed() << ", " << getGreen() << ", " << getBlue() << "]";

}

**int** main()

{

stringstream sin("300 -100 134 140 145 150");

Color col;

cout << "The colors are :\n";

**while** (col.input(sin))

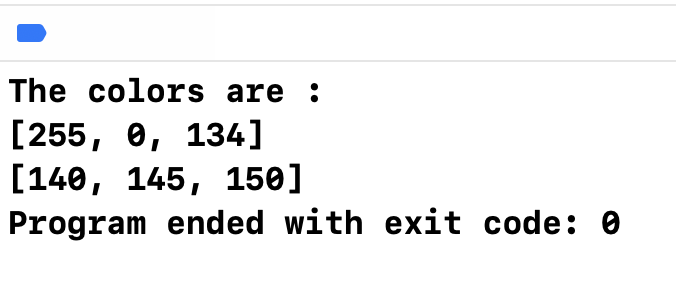
{

col.output(cout) << endl;

}

**return** 0;

}



**Question 2**

Add the following operators:

1 + (binary Add), as member function

2 - (binary Subtract), as friend function

1. ++ (pre-increment), add 1 to every color component, make sure not to exceed 255 for each, as member function
2. §– (unary operators), red=255-red, green=255-green and blue=255-blue, as friend function
3. ==, 1-to-1 equality comparison for red, green, blue in both inputs, as member function
4. !=, 1-to-1 inequality comparison for red, green, blue in both inputs, as friend function

Test your code with following main (you are not allowed to change it):

int main()

{

Color col1,col2;

cout<<"Please enter Red Green and Blue for Color1: ";

col1.input(cin);

cout<<"Please enter Red Green and Blue for Color2: ";

col2.input(cin);

cout<<"Color1 == Color2 => "<<(col1==col2)<< endl;

cout<<"Color1 != Color2 => "<<(col1!=col2)<< endl;

Color sum = col1 + col2;

col1.output(cout) << " + "; col2.output(cout) << " = " ;sum.output(cout) << endl;

Color sub = col1 - col2;

col1.output(cout) << " - ";col2.output(cout) << " = " ;sub.output(cout) << endl;

Color col3 = -col1;

cout << "-";col1.output(cout) << " = ";col3.output(cout) << endl;

cout << "++"; col2.output(cout) << " = ";

++col2;

col2.output(cout) << endl;

Color col5 = col1 + col2 - col1;

col1.output(cout) << " + "; col2.output(cout) << " - "; col1.output(cout) << " = "; col5.output(cout) << endl;

Color col6 = col5 + 1;

col5.output(cout) << " + 1 = "; col6.output(cout) << endl;

cout<<"Testing non-default constructor : \n";

cout<<"Color col3(250,240,0);\nColor col4 = col3+col2 = ";

Color col4 = col3+col2;

col4.output(cout)<<endl;

cout<<"Testing setters and getters :\ncol4 is filled with 0 0 0 using setters\n";

col4.setRed(0);

col4.setGreen(0);

col4.setBlue(0);

cout<<"Testing getters, printing col4 using the getters\ncol4 = ";

cout<<"["<<col4.getRed()<<"] ["<<col4.getGreen()<<"] ["<<col4.getBlue()<<"]\n";

return 0;

}

A sample run is shown below:

**Please enter Red Green and Blue for Color1:** 145 154 133

**Please enter Red Green and Blue for Color2:** 160 145 87

**Color1 == Color2 => 0**

**Color1 != Color2 => 1**

**[145,154,133] + [160,145,87] = [255,255,220]**

**[145,154,133] - [160,145,87] = [0,9,46]**

**-[145,154,133] = [110,101,122]**

**++[160,145,87] = [161,146,88]**

**[145,154,133] + [161,146,88] - [145,154,133] = [110,101,88]**

**[110,101,88] + 1 = [111,102,89]**

**Testing non-default constructor :**

**Color col3(250,240,0);**

**Color col4 = col3+col2 = [255,247,210]**

**Testing setters and getters :**

**col4 is filled with 0 0 0 using setters**

**Testing getters, printing col4 using the getters**

**col4 = [0] [0] [0]**

#include <iostream>

#include <string>

**using** **namespace** std;

**class** Color

{

**private**:

**int** red, green, blue;

**int** checkAndTruncate(**int**)**const**;

**public**:

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

Color(**int** x) { red = green = blue = checkAndTruncate(x); }

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

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

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

**int** getRed() **const** { **return** red; }

**int** getGreen() **const** { **return** green; }

**int** getBlue() **const** { **return** blue; }

istream& input(istream& in);

ostream& output(ostream& out);

**friend** **bool** **operator**!=(**const** Color& col1, **const** Color& col2) {

**return** !(col1 == col2);

}

**friend** Color **operator**-(**const** Color& c1, **const** Color& c2) {

Color sub;

sub.setRed(c1.red - c2.red);

sub.setGreen(c1.green - c2.green);

sub.setBlue(c1.blue - c2.blue);

**return** sub;

}

Color **operator**+(**const** Color& c1) **const** {

Color sum;

sum.setRed(red + c1.red);

sum.setGreen(green + c1.green);

sum.setBlue(blue + c1.blue);

**return** sum;

}

**friend** Color **operator**-(**const** Color& col1) {

Color un;

un.setRed(255 - col1.red);

un.setGreen(255 - col1.green);

un.setBlue(255 - col1.blue);

**return** un;

}

**void** **operator**++() {

setRed(red + 1);

setGreen(green + 1);

setBlue(blue + 1);

}

**bool** **operator**==(**const** Color& col1) **const**

{

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

}

};

istream& Color::input(istream& in)

{

**int** r, g, b;

**if** (in >> r >> g >> b)

{

setRed(r);

setGreen(g);

setBlue(b);

}

**return** in;

}

ostream& Color::output(ostream& out)

{

out << "[" << getRed() << "," << getGreen() << "," << getBlue() << "]";

**return** out;

}

**int** Color::checkAndTruncate(**int** x)**const**

{

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

**return** x;

**else** **if** (x < 0)

{

**return** 0;

}

**else**

{

**return** 255;

}

}

**int** main()

{

Color col1, col2;

cout << "Please enter Red Green and Blue for Color1: ";

col1.input(cin);

cout << "Please enter Red Green and Blue for Color2: ";

col2.input(cin);

cout << "Color1 == Color2 => " << (col1 == col2) << endl;

cout << "Color1 != Color2 => " << (col1 != col2) << endl;

Color sum = col1 + col2;

col1.output(cout) << " + "; col2.output(cout) << " = "; sum.output(cout) << endl;

Color sub = col1 - col2;

col1.output(cout) << " - "; col2.output(cout) << " = "; sub.output(cout) << endl;

Color col3 = -col1;

cout << "-"; col1.output(cout) << " = "; col3.output(cout) << endl;

cout << "++"; col2.output(cout) << " = ";

++col2;

col2.output(cout) << endl;

Color col5 = col1 + col2 - col1;

col1.output(cout) << " + "; col2.output(cout) << " - "; col1.output(cout) << " = "; col5.output(cout) << endl;

Color col6 = col5 + 1;

col5.output(cout) << " + 1 = "; col6.output(cout) << endl;

cout << "Testing non-default constructor : \n";

cout << "Color col3(250,240,0);\nColor col4 = col3+col2 = ";

Color col4 = col3 + col2;

col4.output(cout) << endl;

cout << "Testing setters and getters :\ncol4 is filled with 0 0 0 using setters\n";

col4.setRed(0);

col4.setGreen(0);

col4.setBlue(0);

cout << "Testing getters, printing col4 using the getters\ncol4 = ";

cout << "[" << col4.getRed() << "] [" << col4.getGreen() << "] [" << col4.getBlue() << "]\n";

**return** 0;

}

Good Luck ☺

