**Name:** *Muhammad Kaleem Ullah* **Reg. No.#:** *FA19-BCE-007*

**In Lab 05 – Operator Overloading (Contd.)**

1. **In-Lab Tasks**

**5.1** Use the Arithmetic Assignment operator (+=) for a Distance class to add one distance to a second, leaving the result in the first. This is similar to example shown earlier, but there is a subtle difference.

**Solution:**

* **Code:**

#include<iostream>

using namespace std;

class Distance

{

private:

int feet;

float inches;

public:

Distance():feet(0),inches(0.0)

{

}

Distance(int ft,float in):feet(ft),inches(in)

{

}

void getDist()

{

cout<<"\n Enter Feet : ";

cin>>feet;

cout<<"\n Enter inches : ";

cin>>inches;

}

void showDist()const

{

cout<<feet<<"\'"<<inches<<"\"";

}

Distance operator += (const Distance);

};

Distance Distance:: operator +=(const Distance d2)

{

feet+=d2.feet;

inches+=d2.inches;

if(inches>=12.0)

{

inches-=12.0;

feet++;

}

}

int main()

{

Distance dist1;

dist1.getDist();

cout<<"dist 1 = ";

dist1.showDist();

cout<<endl;

Distance dist2(11,6.25);

dist1+=dist2;

cout<<"dist 2 = ";

dist2.showDist();

cout<<endl;

cout<<"dist1+=dist2 : ";

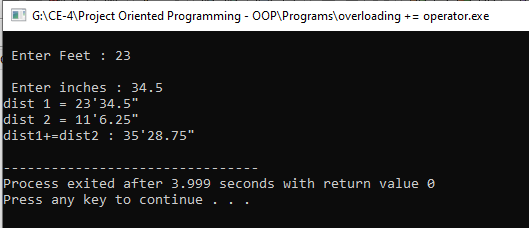
dist1.showDist();

cout<<endl;

return 0;

}

* **Output:**



**5.2** Write a program that substitutes an overloaded += operator for the overloaded + operator in the example given above. This operator should allow statements like s1 += s2; where s2 is added (concatenated) to s1 and the result is left in s1. The operator should also permit the results of the operation to be used in other calculations, as in s3 = s1 += s2;

**Solution:**

* **Code:**

#include<iostream>

using namespace std;

#include<string.h>

#include<stdlib.h>

class String

{

private:

enum

{

SZ=80

};

char str[SZ];

public:

String()

{

strcpy(str,"");

}

String(char s[])

{

strcpy(str,s);

}

void display()const

{

cout<<str;

}

String operator += (String ss)

{

if(strlen(str)+strlen(ss.str)<SZ)

{

strcat(str,ss.str);

}

else

{

cout<<"\n String overflow "; exit(1);

}

}

};

int main()

{

String s1="\n Merry Christmas ! ";

String s2="Happy New Year ! ";

s1.display();

s2.display();

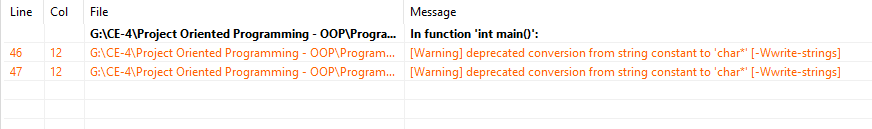
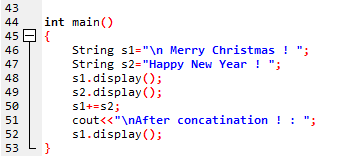
s1+=s2;

cout<<"\nAfter concatination ! : ";

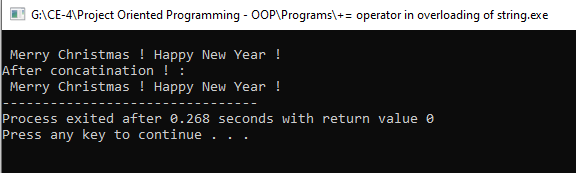
s1.display();

}

* **Things to be answered:**



* **Output:**

****

**5.3** Write a program for class String that uses an overloaded == operator for comparing two strings together. This operator should allow statements like s1 == s2; and display the results showing whether the strings entered by user are same or not.

**Solution:**

* **Code:**

/\*5.3 Write a program for class String that uses an overloaded == operator for comparing two strings together.

This operator should allow statements like s1 == s2; and display the results showing whether the strings entered by user are same or not.\*/

#include<iostream>

using namespace std;

#include<string.h>

#include<stdlib.h>

class String

{

private:

enum

{

SZ=80

};

char str[SZ];

public:

String()

{

strcpy(str,"");

}

String(char s[])

{

strcpy(str,s);

}

void display()const

{

cout<<str;

}

bool operator == (String ss)

{

if(strcmp(str,ss.str))

{

return 0;

}

else

{

return 1;

}

/\*

int count;

for(int i=0;i<SZ;i++)

{

if(str[i]==ss.str[i])

{

count=1;

}

else

{

count=0;

break;

}

}

if(count=1)

{

return true;

}

else

{

return false;

}

\*/

}

};

int main()

{

String s1="\n Merry Christmas ! ";

String s2="Happy New Year ! ";

s1.display();

s2.display();

if(s1==s2)

{

cout<<"Both are equal ! ";

}

else

{

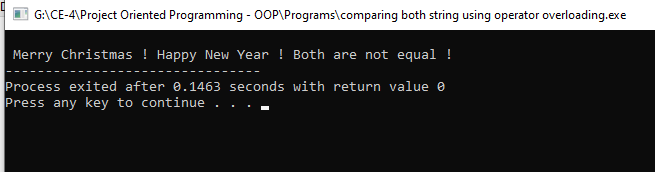
cout<<"Both are not equal ! ";

}

}

**Question:**

* **Check my loop logic please !**
* **Output:**

****

**Conclusion:**

* *I want to try all these with simple strings not with array strings*
* *There are warning while declaring the string kindly clear that*
* *Mam check my for loop compare logic, and clear me the strcmp execution process why it’s return 0 instead of return 1*