**Object Oriented Programming**

**Lab 4**

**Submitted To:**

Ma’am Amber Madeeha Zeb

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Code the example of complex class given above and include the functions for addition, subtraction and multiplication of objects of complex class and return the object containing result. Test all the functions in main.**

**Code:**

1 #include<iostream>

2

3 **using namespace std**;

4

5 **class** Complex

6 {

7 **private**:

8 **double** real, imag;

9 **public**:

10 Complex() // Default Constructor

11 {

12 real = 0.0;

13 imag = 0.0;

14 }

15 // Two argument Constructor

16 Complex (**double** r, **double** im)

17 {

18 real = r;

19 imag = im;

20 }

21 **void** show()

22 {

23 **cout**<<real<<"+"<<imag<<"i"<<**endl**;

24 }

25 Complex Add (Complex& b)

26 { **return** Complex(real + b.real, imag+ b.imag);}

27

28 Complex Sub (Complex& b)

29 { **return** Complex(real - b.real, imag- b.imag);}

30

31 Complex Mul (Complex& b)

32 { **return** Complex(real \* b.real, imag\* b.imag);}

33 };

34

35 **int** main()

36 {

37 Complex C1(11, 2.3);

38 Complex C2(9, 2.3);

39 Complex C7;

40 Complex C3(11, 2.3);

41 Complex C4(9, 2.3);

42 Complex C8;

43 Complex C5(11, 2.3);

44 Complex C6(9, 2.3);

45 Complex C9;

46

47 C7 = C1.Add(C2);

48 C8 = C3.Sub(C4);

49 C9 = C5.Mul(C6);

50 **cout**<<"Addition: "<<**endl**;

51 C7.show();

52 **cout**<<"Subtraction: "<<**endl**;

53 C8.show();

54 **cout**<<"Multiplication: "<<**endl**;

55 C9.show();

56

57 **return** 0;

58 }

**Output:**



**Task 2:**

**Modify the above task such that; define the member function show() outside the class and also define it as inline.**

**Code:**

1 #include<iostream>

2

3 **using namespace std**;

4

5 **class** Complex

6 {

7 **private**:

8 **double** real, imag;

9 **public**:

10 Complex() // Default Constructor

11 {

12 real = 0.0;

13 imag = 0.0;

14 }

15 // Two argument Constructor

16 Complex (**double** r, **double** im)

17 {

18 real = r;

19 imag = im;

20 }

21

22 **int**\* show();

23

24 Complex Add (Complex& b)

25 { **return** Complex(real + b.real, imag+ b.imag);}

26

27 Complex Sub (Complex& b)

28 { **return** Complex(real - b.real, imag- b.imag);}

29

30 Complex Mul (Complex& b)

31 { **return** Complex(real \* b.real, imag\* b.imag);}

32 };

33 **inline int**\* Complex :: show()

34 {

35 cout<<real<<"+"<<imag<<"i"<<endl;

36 return 0;

37 }

38 int main()

39 {

40 Complex C1(11, 2.3);

41 Complex C2(9, 2.3);

42 Complex C7;

43 Complex C3(11, 2.3);

44 Complex C4(9, 2.3);

45 Complex C8;

46 Complex C5(11, 2.3);

47 Complex C6(9, 2.3);

48 Complex C9;

49

50 C7 = C1.Add(C2);

51 C8 = C3.Sub(C4);

52 C9 = C5.Mul(C6);

53 cout<<"Addition: "<<endl;

54 C7.show();

55 cout<<"Subtraction: "<<endl;

56 C8.show();

57 cout<<"Multiplication: "<<endl;

58 C9.show();

59

60 return 0;

61 }

**Output:**

****

**Task 3:**

**Modify the task (5.1) by defining all the member functions outside the class definition.**

**Code:**

1 #include<iostream>

2

3 **using namespace std**;

4

5 **class** Complex

6 {

7 **private**:

8 **double** real, imag;

9 **public**:

10 Complex() // Default Constructor

11 {

12 real = 0.0;

13 imag = 0.0;

14 }

15 // Two argument Constructor

16 Complex (**double** r, **double** im)

17 {

18 real = r;

19 imag = im;

20 }

21

22 Complex Add (Complex& b);

23 Complex Sub (Complex& b);

24 Complex Mul (Complex& b);

25

26 **int**\* show();

27

28

29 };

30 **inline int**\* Complex :: show()

31 {

32 **cout**<<real<<"+"<<imag<<"i"<<**endl**;

33 **return** 0;

34 }

35

36 Complex Complex :: Add (Complex& b)

37 { **return** Complex(real + b.real, imag+ b.imag);}

38

39 Complex Complex :: Sub (Complex& b)

40 { **return** Complex(real - b.real, imag- b.imag);}

41

42 Complex Complex :: Mul (Complex& b)

43 { **return** Complex(real \* b.real, imag\* b.imag);}

44 **int** main()

45 {

46 Complex C1(11, 2.3);

47 Complex C2(9, 2.3);

48 Complex C7;

49 Complex C3(11, 2.3);

50 Complex C4(9, 2.3);

51 Complex C8;

52 Complex C5(11, 2.3);

53 Complex C6(9, 2.3);

54 Complex C9;

55

56 C7 = C1.Add(C2);

57 C8 = C3.Sub(C4);

58 C9 = C5.Mul(C6);

59 **cout**<<"Addition: "<<**endl**;

60 C7.show();

61 **cout**<<"Subtraction: "<<**endl**;

62 C8.show();

63 **cout**<<"Multiplication: "<<**endl**;

64 C9.show();

65

66 return 0;

67 }

**Output:**

****

**Task 4:**

**Test the distance class example given above.**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** add //Specifies the class

6 {

7 **private**:

8 **int** iNum1, iNum2, iNum3; //Member data

9

10 add(**int** a=0, **int** b=0)

11 {

12 iNum1=a;

13 iNum2=b;

14 }

15 **public**:

16 **void** input(**int** iVar1, **int** iVar2) //Member function

17 {

18 **cout**<<"Functions to assign values to the member data"<<**endl**;

19 iNum1=iVar1; iNum2=iVar2;

20 }

21 **void** sum(**void**) //Member function

22 {

23 **cout**<<"Functions to find the sum of two numbers"<<**endl**;

24 iNum3=iNum1+iNum2;

25 }

26 **void** disp(**void**) //Member function

27 {

28 **cout**<<"The sum of the two numbers is "<<iNum3<<**endl**;

29 }

30 };

31

32 **int** main()

33 {

34 add A1;

35 **int** iX, iY;

36 **cout**<<"Input two numbers"<<**endl**;

37 **cin**>>iX;

38 **cin**>>iY;

39 A1.input(iX, iY);

40 A1.sum();

41 A1.disp();

42 //system("pause");

43

44

45 **return** 0;

46 }

**Output:**

****

**Conclusion:**

In this task the addition of two distances in not done properly because an **if-statement** executes and only check once if feet are greater than 12. This approach does not lead us to proper addition of two distances.

**Post Lab:**

**Task 1:**

**Modify the Distance class example of lab task (5.4) by including functions for subtraction and multiplication of distance class objects like addition.**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Distance //English Distance class

6 {

7 **private**:

8 **int** feet;

9 **float** inches;

10 **public**:

11 //constructor (no args)

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

13 { }

14 //constructor (two args)

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

16 { }

17

18 **void** getdist() //get length from user

19 {

20 **cout**<<"\nEnter feet: ";

21 **cin**>>feet;

22 **cout**<<"Enter inches: ";

23 **cin**>>inches;

24 }

25

26 **inline void** showdist(); //display distance

27

28 **void** add\_dist( Distance, Distance ); //declaration

29 **void** sub\_dist( Distance, Distance ); //declaration

30 **void** mul\_dist( Distance, Distance ); //declaration

31 };

32 //--------------------------------------------------------------

33 //definition of inline function which display distance

34

35 **inline void** Distance::showdist() { **cout**<<feet<<"\""<<inches<<"\'"; }

36

37 //add lengths d2 and d3

38 **void** Distance::add\_dist(Distance d2, Distance d3)

39 {

40 **float** n1, n2, add;

41

42 n1=(d2.feet\*12)+d2.inches;

43 n2=(d3.feet\*12)+d3.inches;

44

45 add=n1+n2;

46

47 feet=**int**(add)/12;

48 inches=add-feet\*12;

49 }

50

51 //subtracts lengths d2 and d3

52 **void** Distance::sub\_dist(Distance d2, Distance d3)

53 {

54 **float** n1, n2, sub;

55

56 n1=(d2.feet\*12)+d2.inches;

57 n2=(d3.feet\*12)+d3.inches;

58

59 sub=n1-n2;

60

61 feet=**int**(sub)/12;

62 inches=sub-feet\*12;

63 }

64

65 //multiplies lengths d2 and d3

66 **void** Distance::mul\_dist(Distance d2, Distance d3)

67 {

68 **float** n1, n2, mul;

69

70 n1=(d2.feet\*12)+d2.inches;

71 n2=(d3.feet\*12)+d3.inches;

72

73 mul=n1\*n2;

74

75 feet=**int**(mul)/12;

76 inches=mul-feet\*12;

77

78 }

79 ////////////////////////////////////////////////////////////////

80 **int** main()

81 {

82 Distance dist1, dist3; //define two Distance Objects

83 Distance dist2(11, 6.25); //define and initialize dist2

84 dist1.getdist(); //get dist1 from user

85

86 **cout**<<"\nAddition: ";

87 dist3.add\_dist(dist1, dist2); //dist3 = dist1 + dist2

88 //display all lengths

89 **cout**<<"\ndist1 = ";

90 dist1.showdist();

91 **cout**<<"\ndist2 = ";

92 dist2.showdist();

93 **cout**<<"\ndist3 = ";

94 dist3.showdist();

95 **cout**<<**endl**;

96

97 **cout**<<"\nSubtraction: ";

98 dist3.sub\_dist(dist1, dist2); //dist3 = dist1 - dist2

99 //display all lengths

100 **cout**<<"\ndist1 = ";

101 dist1.showdist();

102 **cout**<<"\ndist2 = ";

103 dist2.showdist();

104 **cout**<<"\ndist3 = ";

105 dist3.showdist();

106 **cout**<<**endl**;

107

108 **cout**<<"\nMultiplication: ";

109 dist3.mul\_dist(dist1, dist2); //dist3 = dist1 \* dist2

110 //display all lengths

111 **cout**<<"\ndist1 = ";

112 dist1.showdist();

113 **cout**<<"\ndist2 = ";

114 dist2.showdist();

115 **cout**<<"\ndist3 = ";

116 dist3.showdist();

117 **cout**<<**endl**;

118

119 **return** 0;

120 }

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**