**Object Oriented Programming**

**Lab 12**

**Submitted To:**

Ma’am Amber Madeeha Zeb

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Create a counter class, overload ++ operator for counter post and pre increment, use the object of counter class as a loop counter for printing a table in main function.**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Counter

6 {

7 **private**:

8 **int count**;

9 **public**:

10 Counter() : **count**(0)

11 { }

12 Counter(**int** c) : **count**(c)

13 { }

14 **int** get\_count() **const**

15 { **return count**; }

16 Counter **operator** ++ ()

17 {

18 **return** Counter(++**count**);

19 }

20 Counter **operator** ++ (**int**)

21 {

22 **return** Counter(**count**++);

23 }

24 };

25

26 **int** main()

27 {

28 Counter c1, c2;

29 **cout** << "\nc1=" << c1.get\_count();

30 **cout** << "\nc2=" << c2.get\_count();

31 ++c1;

32 c2 = ++c1;

33 **cout** << "\nc1=" << c1.get\_count();

34 **cout** << "\nc2=" << c2.get\_count();

35 c2 = c1++;

36 **cout** << "\nc1=" << c1.get\_count();

37 **cout** << "\nc2=" << c2.get\_count() << **endl**;

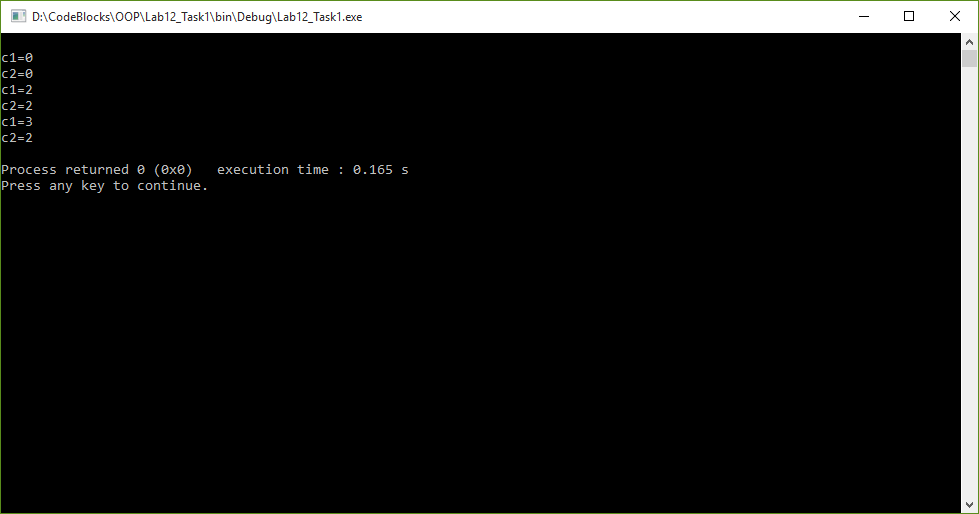
38

39 **return** 0;

40 }

41

**Output:**



**Task 2:**

**A complex number is a number which can be put in the form a + bi. Create a class for complex numbers, which handle real and imaginary part separately. Class should consist minimum required constructors, get and show methods also Overload the + operator for this class which work like this formula.**



**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Distance

6 {

7 **private**:

8 **int** feet, inches;

9 **public**:

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

11 {}

12 Distance(**int** f, **int** i):feet(f), inches(i)

13 {}

14

15 **void** get\_data();

16 **void** display();

17

18 **int operator** %(**int** val)

19 {

20 Distance t1;

21 **int** temp;

22 t1.inches=(feet\*12)+inches;

23 temp=t1.inches%val;

24

25 **return** temp;

26 }

27 };

28

29 **void** Distance::get\_data()

30 {

31 **cout**<< "Enter Feet: " << **endl**;

32 **cin**>> feet;

33 **cout**<< "Enter Inches: " << **endl**;

34 **cin**>> inches;

35 }

36

37 **void** Distance::display()

38 {

39 **cout**<< "Feet: " << feet << **endl**;

40 **cout**<< "Inches: " << inches << **endl**;

41 }

42

43 **int** main()

44 {

45 Distance d1;

46 **int** r1;

47

48 d1.get\_data();

49

50 r1=d1%3;

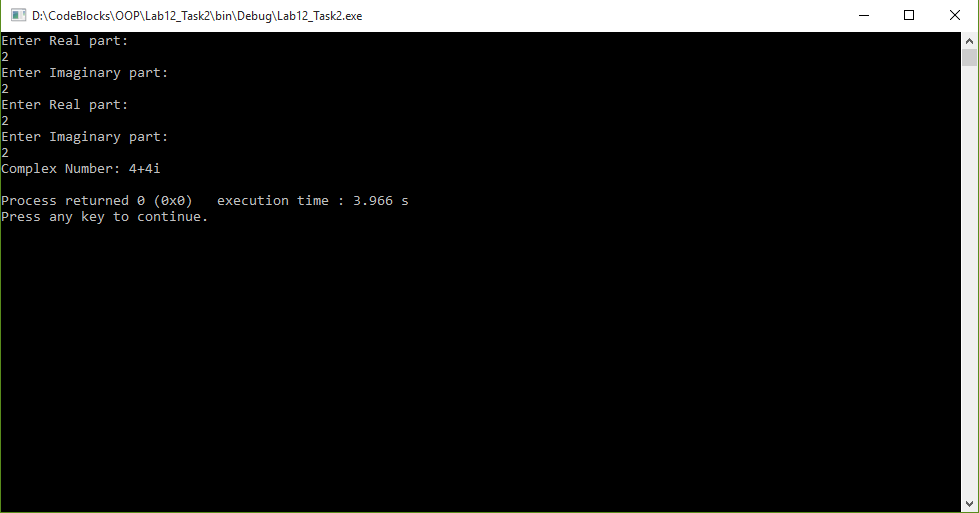
51 **cout**<< "Remainder: " << r1 << **endl**;

52

53 **return** 0;

54 }

**Output:**



**Task 3:**

**Create a class of Distance including feet and inches. Class should consist minimum required constructors, get and show methods also overload the % operator for this class.**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Distance

6 {

7 **private**:

8 **int** feet, inches;

9 **public**:

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

11 {}

12 Distance(**int** f, **int** i):feet(f), inches(i)

13 {}

14

15 **void** get\_data();

16 **void** display();

17

18 **int operator** %(**int** val)

19 {

20 Distance t1;

21 **int** temp;

22 t1.inches=(feet\*12)+inches;

23 temp=t1.inches%val;

24

25 **return** temp;

26 }

27 };

28

29 **void** Distance::get\_data()

30 {

31 **cout**<< "Enter Feet: " << **endl**;

32 **cin**>> feet;

33 **cout**<< "Enter Inches: " << **endl**;

34 **cin**>> inches;

35 }

36

37 **void** Distance::display()

38 {

39 **cout**<< "Feet: " << feet << **endl**;

40 **cout**<< "Inches: " << inches << **endl**;

41 }

42

43 **int** main()

44 {

45 Distance d1;

46 **int** r1;

47

48 d1.get\_data();

49

50 r1=d1%3;

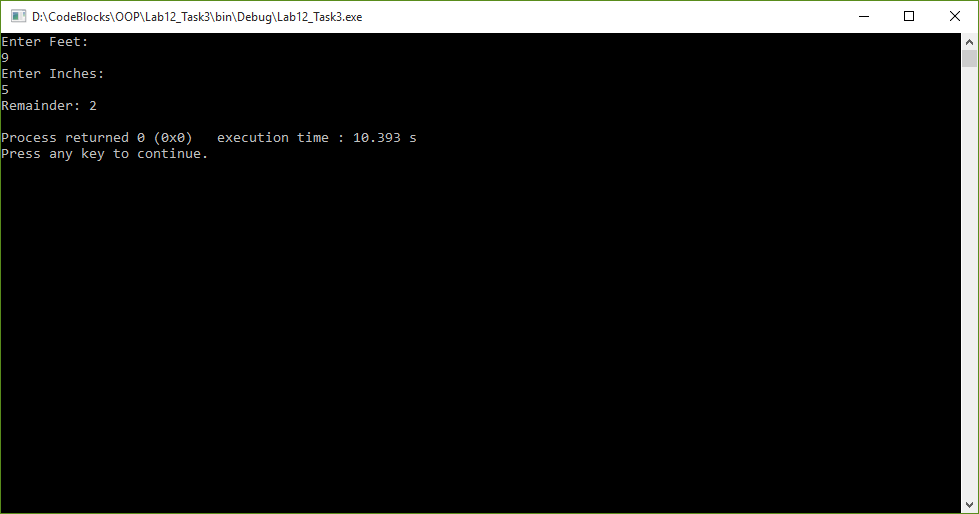
51 **cout**<< "Remainder: " << r1 << **endl**;

52

53 **return** 0;

54 }

**Output:**



**Post Lab:**

**Task 1:**

**Create a calculator for the complex number by creating a class of complex number with overloading all operators in it. (Operators: ++,--,+,-,/,\*, >>, <<).**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Complex

6 {

7 **private**:

8 **int** real, imag;

9 **public**:

10 Complex():real(0), imag(0)

11 {}

12 Complex(**int** r, **int** i):real(r), imag(i)

13 {}

14

15 Complex **operator** ++ ()

16 {

17 ++real;

18 ++imag;

19 **return** Complex(real, imag);

20 }

21

22 Complex **operator** ++ (**int**)

23 {

24 real++;

25 imag++;

26 **return** Complex(real, imag);

27 }

28

29 Complex **operator** -- ()

30 {

31 --real;

32 --imag;

33 **return** Complex(real, imag);

34 }

35

36 Complex **operator** -- (**int**)

37 {

38 real--;

39 imag--;

40 **return** Complex(real, imag);

41 }

42

43 Complex **operator** +(Complex c)

44 {

45 Complex temp;

46 temp.real= real + c.real;

47 temp.imag= imag + c.imag;

48 **return** temp;

49 }

50

51 Complex **operator** -(Complex c)

52 {

53 Complex temp;

54 temp.real= real - c.real;

55 temp.imag= imag - c.imag;

56 **return** temp;

57 }

58

59 Complex **operator** \*(Complex c)

60 {

61 Complex temp;

62 temp.real= (real\*c.real) - (imag\*c.imag);

63 temp.imag= (real\*c.imag) + (imag\*c.real);

64 **return** temp;

65 }

66

67 Complex **operator** /(Complex c)

68 {

69 Complex temp;

70 temp.real= ((real\*c.real)+(imag\*c.imag))/((c.real\*c.real)+(c.imag\*c.imag));

71 temp.imag= ((imag\*c.real)-(real\*c.imag))/((c.real\*c.real)+(c.imag\*c.imag));

72 **return** temp;

73 }

74

75 **friend** ostream &**operator**<<( ostream &output, **const** Complex &C)

76 {

77 output << "Complex Number: " << C.real << "+" << C.imag << "i";

78 **return** output;

79 }

80

81 **friend istream** &**operator**>>( **istream** &input, Complex &C )

82 {

83 input >> C.real >> C.imag;

84 **return** input;

85 }

86

87 };

88

89 **int** main()

90 {

91 Complex c1, c2, r1;

92

93 **cout**<< "Enter First Complex Number: " <<**endl**;

94 **cin**>> c1;

95

96 **cout**<< "Enter Second Complex Number: " <<**endl**;

97 **cin**>> c2;

98

99 **cout**<< "Increment: " << **endl**;

100 r1= c1++;

101 **cout**<< r1 << **endl**;

102

103 **cout**<< "Decrement: " << **endl**;

104 r1= c2--;

105 **cout**<< r1 << **endl**;

106

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

108 r1=c1+c2;

109 **cout**<< r1 << **endl**;

110

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

112 r1=c1-c2;

113 **cout**<< r1 << **endl**;

114

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

116 r1=c1\*c2;

117 **cout**<< r1 << **endl**;

118

119 **cout**<< "Division: " << **endl**;

120 r1=c1/c2;

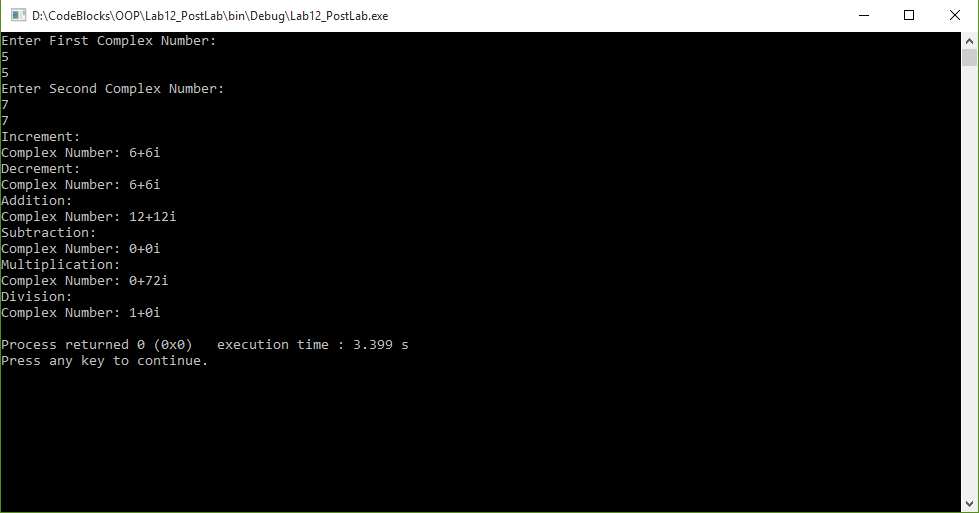
121 **cout**<< r1 << **endl**;

122

123 **return** 0;

124 }

**Output:**



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

**THE END**