**Object Oriented Programming**

**Lab Report**

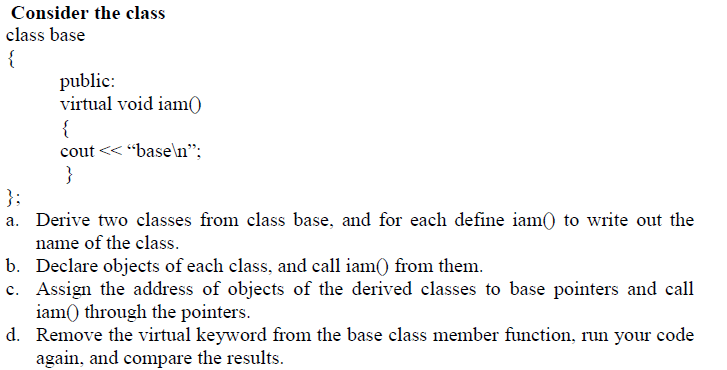
**Lab10**



|  |  |
| --- | --- |
| Group Members Name & Reg #: | **Muhammad Haris Irfan**  **(FA18-BCE-090)** |
|  |  |
| Class | Object Oriented Programming CSC241 (**BCE-4B**) |
| Instructor’s Name | Maam Amber Madeeha Zeb |

**In Lab Tasks**

5.1 Question 1:

****

**Solution:**

The code is given below,

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** base

6 {

7 **public**:

8 **virtual void** iam()

9 {

10 **cout**<<"Base Class"<<**endl**;

11 }

12 };

13

14 **class** derv1 :**public** base

15 {

16 **public**:

17 **void** iam()

18 {

19 **cout**<<"Derived Class 1"<<**endl**;

20 }

21 };

22

23 **class** derv2 :**public** base

24 {

25 **public**:

26 **void** iam()

27 {

28 **cout**<<"Derived Class 2"<<**endl**;

29 }

30 };

31 **int** main()

32 {

33

34 base b1;

35 base \*ptr1;

36 derv1 d1;

37 derv2 d2;

38

39

40 b1.iam();

41 d1.iam();

42 d2.iam();

43

44 **cout**<<**endl**<<"Through pointer:"<<**endl**;

45 ptr1=&d1;

46 ptr1->iam();

47

48 ptr1=&d2;

49 ptr1->iam();

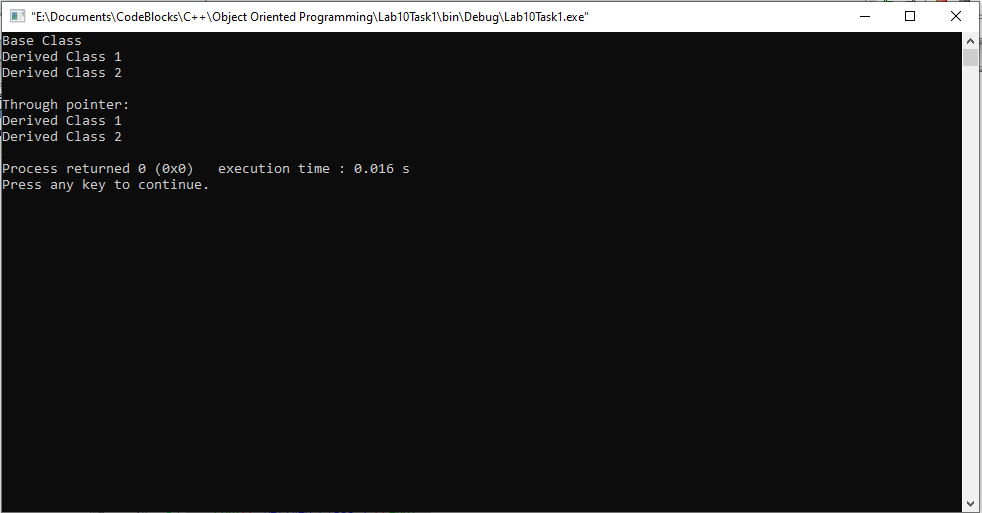
50

51

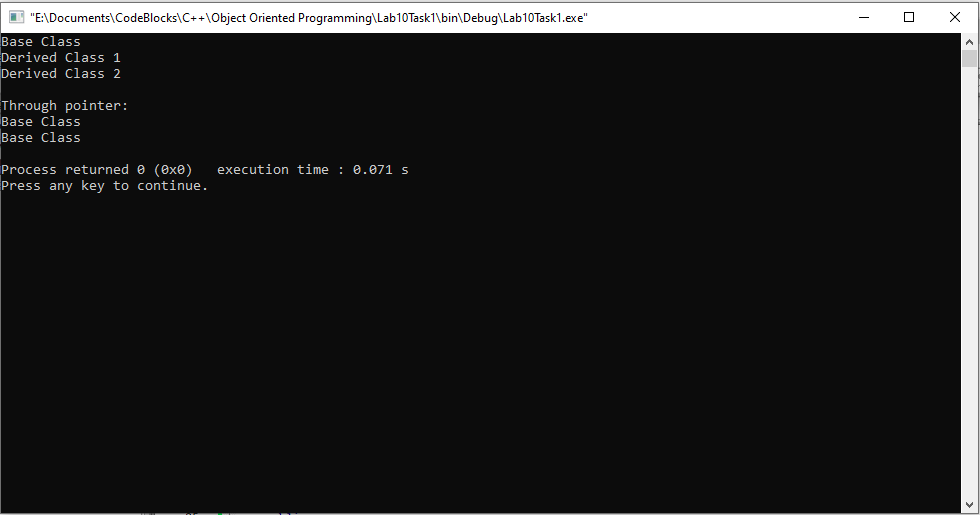
52 **return** 0;

53 }

**Console Output is shown below.**

****

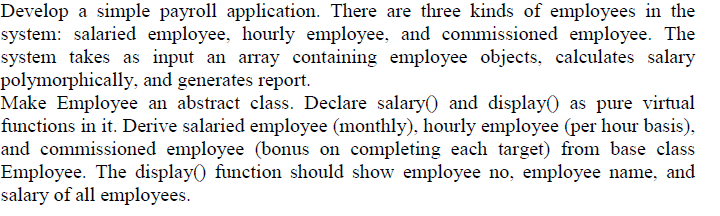
**Now after removing the keyword virtual the output is as follows:**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\

5.2 Question 2:

****

**Solution:**The code is given below,

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** employee

6 {

7 **private**:

8

9 **int** empno;

10 **string** name;

11

12 **public**:

13 **virtual void** salary()=0;

14 **virtual void** display()=0;

15

16 employee()

17 {

18 **cout**<<**endl**<<"Enter Employee number?"<<**endl**;

19 **cin**>>empno;

20 **cout**<<"Enter name of Employee?"<<**endl**;

21 **cin**>>name;

22

23 }

24

25 **void** putdata()

26 {

27 **cout**<<"----------------------------------------------------------------------"<<**endl**;

28 **cout**<<"Employee Name: "<<name<<**endl**;

29 **cout**<<"Employee Number: "<<empno<<**endl**;

30 }

31

32 };

33

34 **class** monthEmploy : **public** employee

35 {

36 **float** sal;

37 **public**:

38 **void** salary()

39 {

40 **cout**<<"Enter Salary?"<<**endl**;

41 **cin**>>sal;

42 }

43 **void** display()

44 {

45 employee::putdata();

46 **cout**<<"Salary: "<<sal<<**endl**;

47 **cout**<<"Employee type: Monthly"<<**endl**;

48 **cout**<<"----------------------------------------------------------------------"<<**endl**;

49 }

50 };

51

52 **class** hourlyEmploy : **public** employee

53 {

54 **float** sal;

55 **public**:

56 **void** salary()

57 {

58 **cout**<<"Enter Salary?"<<**endl**;

59 **cin**>>sal;

60 }

61 **void** display()

62 {

63 employee::putdata();

64 **cout**<<"Salary: "<<sal<<**endl**;

65 **cout**<<"Employee type: Hourly"<<**endl**;

66 **cout**<<"----------------------------------------------------------------------"<<**endl**;

67 }

68 };

69

70 **class** commisEmploy : **public** employee

71 {

72 **float** sal;

73 **public**:

74 **void** salary()

75 {

76 **cout**<<"Enter Salary?"<<**endl**;

77 **cin**>>sal;

78

79 }

80

81 **void** display()

82 {

83 employee::putdata();

84 **cout**<<"Salary: "<<sal<<**endl**;

85 **cout**<<"Employee type: Commissioned"<<**endl**;

86 **cout**<<"----------------------------------------------------------------------"<<**endl**;

87 }

88 };

89 **int** main()

90 {

91 employee \*ptr[3];

92 **char** op;

93 **for**(**int** i=0;i<3;i++)

94 {

95 **cout**<<**endl**<<"Enter Monthly, Hourly or Commissioned Employee? (Enter M/H/C)"<<**endl**;

96 **cin** >>op;

97 **if**(op=='M')

98 {

99 ptr[i]= **new** monthEmploy;

100 ptr[i]->salary();

101 ptr[i]->display();

102

103 }

104 **if**(op=='H')

105 {

106 ptr[i]= **new** hourlyEmploy;

107 ptr[i]->salary();

108 ptr[i]->display();

109

110 }

111 **if**(op=='C')

112 {

113 ptr[i]= **new** commisEmploy;

114 ptr[i]->salary();

115 ptr[i]->display();

116

117 }

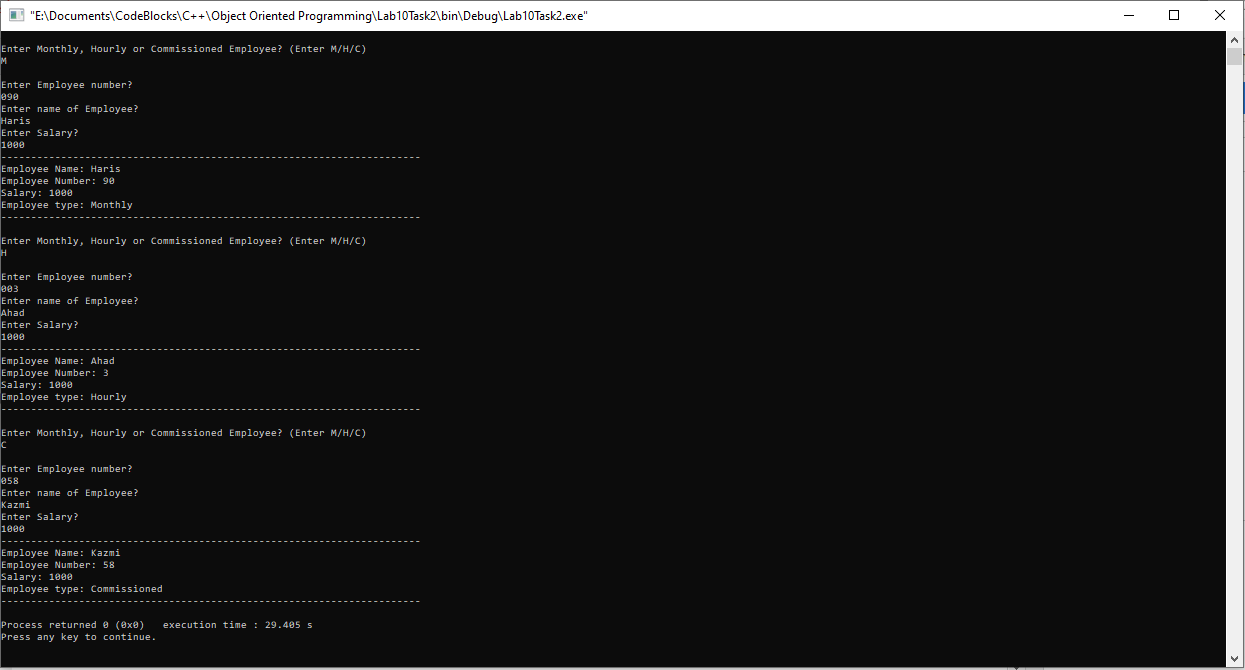
118

119 }

120 **return** 0;

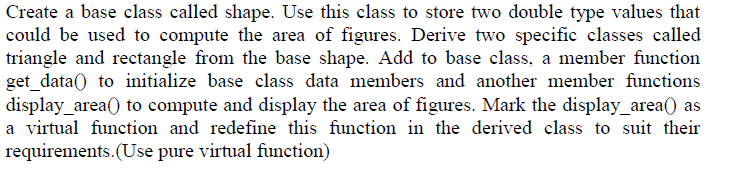
121 }

**Console Output is shown below.**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5.3 Question 3:

****

**Solution:**The code is given below,

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** base

6 {

7 **private**:

8 **double** width;

9 **double** length;

10

11 **public**:

12 **virtual void** display\_area()=0;

13 **double** getlength()

14 {

15 **cout**<< **endl**;

16 **cout**<<"---------------------------------------------------------"<<**endl**;

17 **cout**<<"Enter length ?"<<**endl**;

18 **cin**>>length;

19 **return** length;

20 }

21 **double** getwidth()

22 {

23 **cout**<<"Enter length ?"<<**endl**;

24 **cin**>>width;

25 **return** width;

26 }

27

28 };

29

30 **class** triangle :**public** base

31 {

32 **public**:

33 **void** display\_area()

34 {

35 **double** areat;

36 areat=0.5\*getlength()\*getwidth();

37 **cout**<<"Area of triangle is: "<<areat<<**endl**;

38 **cout**<<"---------------------------------------------------------"<<**endl**;

39 }

40

41 };

42

43 **class** rectangle :**public** base

44 {

45 **public**:

46 **void** display\_area()

47 {

48 **double** arear;

49 arear=getlength()\*getwidth();

50 **cout**<<"Area of rectangle is: "<<arear<<**endl**;

51 **cout**<<"---------------------------------------------------------"<<**endl**;

52 }

53

54 };

55 **int** main()

56 {

57 triangle t1;

58 rectangle r1;

59

60 t1.display\_area();

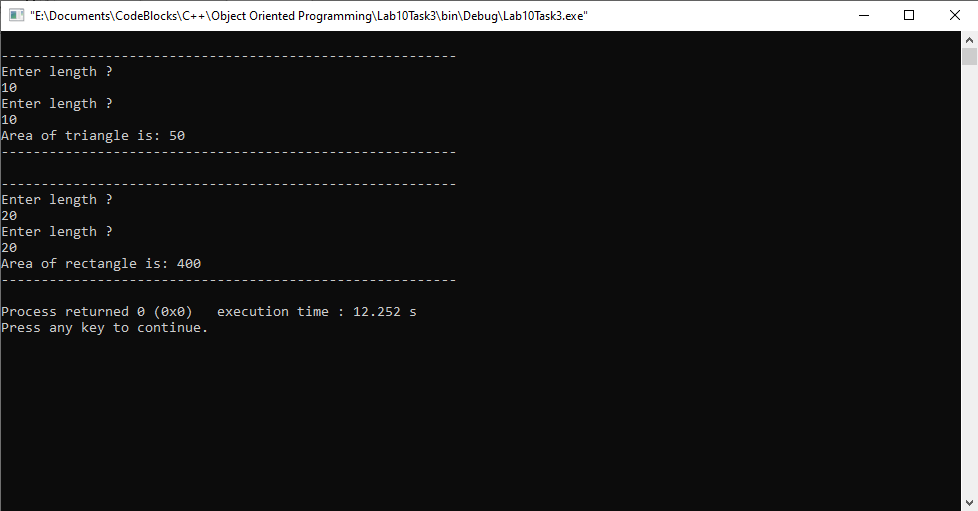
61 r1.display\_area();

62

63 **return** 0;

64 }

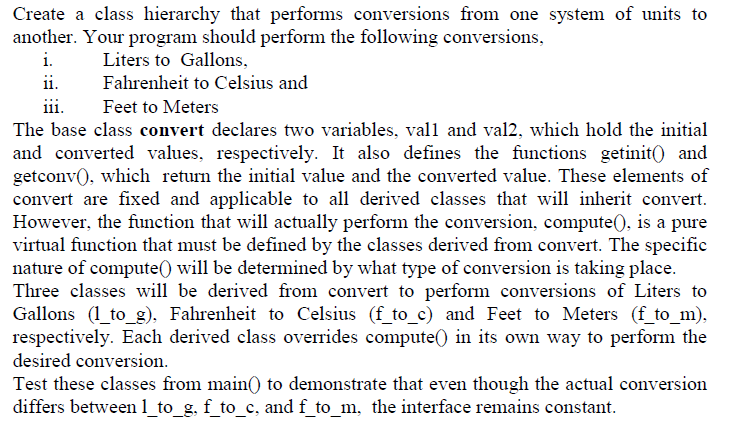
**Console Output is shown below.**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

POST LAB

6.1 Question 4:



Solution:

I am attaching my code below,

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** convert

6 {

7

8 **public**:

9 **double** val1; **///initial value**

10 **double** val2;

11 **virtual void** compute ()=0;

12

13 **double** getint()

14 {

15

16 **cout**<<**endl**<<"Enter the value to be converted ?"<<**endl**;

17 **cin**>>val1;

18 **return** val1;

19 }

20

21 **double** getconv() **const**

22 {

23

24 **return** val2;

25 }

26

27 };

28 **class** litToGal:**public** convert

29 {

30 **public**:

31

32 **void** compute()

33 {

34 val2=getint()\*0.264172;

35 **cout**<<"After conversion: "<<**endl**;

36 **cout**<<val1<< " Liters is equal to "<<val2 <<" Gallons"<<**endl**;

37 **cout**<<"----------------------------------------------------------"<<**endl**;

38 }

39 };

40

41 **class** farToCel:**public** convert

42 {

43 **public**:

44

45 **void** compute()

46 {

47 val2=(getint()-32)\*(0.5555555);

48 **cout**<<"After conversion: "<<**endl**;

49 **cout**<<val1<< " Fahrenheit is equal to "<<val2 <<" Celsius"<<**endl**;

50 **cout**<<"----------------------------------------------------------"<<**endl**;

51 }

52 };

53

54

55 **class** feetToMet:**public** convert

56 {

57 **public**:

58

59 **void** compute()

60 {

61 val2=getint()/3.2808;

62 **cout**<<"After conversion: "<<**endl**;

63 **cout**<<val1<< " Feet is equal to "<<val2 <<" Meters"<<**endl**;

64 **cout**<<"----------------------------------------------------------"<<**endl**;

65 }

66 };

67

68 **int** main()

69 {

70 litToGal g1;

71 farToCel c1;

72 feetToMet m1;

73

74 **cout**<<"----------------------------------------------------------"<<**endl**;

75 **cout**<<"Liters to Gallons Conversion:"<<**endl**;

76 g1.compute();

77

78 **cout**<<"----------------------------------------------------------"<<**endl**;

79 **cout**<<"Fahrenheit to Celsius Conversion:"<<**endl**;

80 c1.compute();

81

82 **cout**<<"----------------------------------------------------------"<<**endl**;

83 **cout**<<"Feet to Meters Conversion:"<<**endl**;

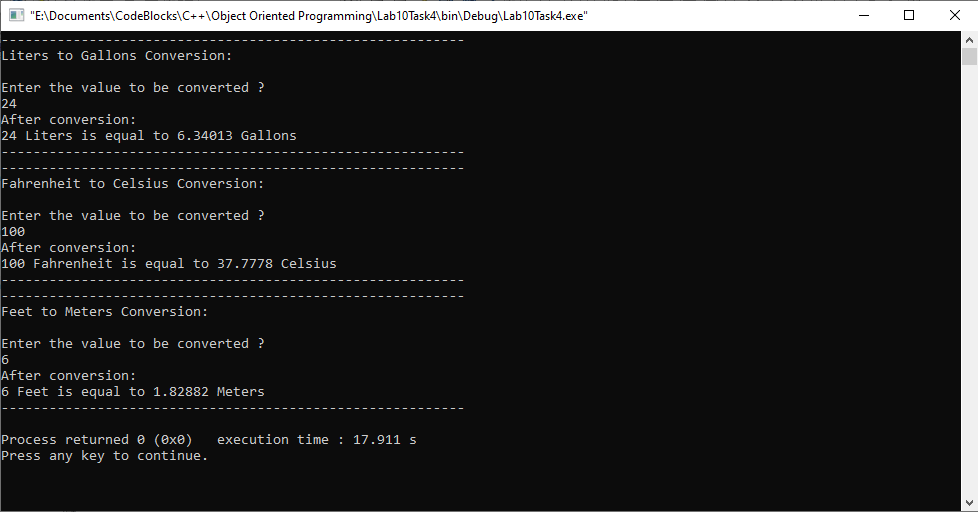
84 m1.compute();

85

86 **return** 0;

87 }

**The result for this program is shown below,**

****

\_\_\_\_\_\_THE END\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_