**Object Oriented Programming**

**Lab 8**

**Submitted To:**

Ma’am Amber Madeeha Zeb

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

**Create a Class named base which has two data members. Then derive a class derived1 from base class which has one data members. Derive a class derived2 from derived1.**

**i. Write functions for each class to get and display values.**

**ii. Write main() function to create object of derived2 and through that object access the data member of base class and derived1 class.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 using namespace std;

5

6 class base

7 {

8 public:

9 void get\_basedata();

10 void put\_basedata();

11

12 private:

13 int data1, data2;

14 };

15

16 void base::get\_basedata()

17 {

18 cout<< "Enter first data: " << endl;

19 cin>> data1;

20 cout<< "Enter second data: " << endl;

21 cin>> data2;

22 }

23

24 void base::put\_basedata()

25 {

26 cout<< "First Data: " << data1 << endl;

27 cout<< "Second Data: " << data2 << endl;

28 }

29

30 class derived1: public base

31 {

32 public:

33 void get\_derived1data();

34 void put\_derived1data();

35

36 private:

37 int data;

38 } ;

39

40 void derived1::get\_derived1data()

41 {

42 base:get\_basedata();

43

44 cout<< "Enter data for Derived 1: " << endl;

45 cin>> data;

46 }

47

48 void derived1::put\_derived1data()

49 {

50 base:put\_basedata();

51

52 cout<< "Data for Derived 1: " << data << endl;

53 }

54

55 class derived2: public derived1

56 {

57 public:

58 void get\_derived2data();

59 void put\_derived2data();

60

61 private:

62 int data;

63 } ;

64

65 void derived2::get\_derived2data()

66 {

67 derived1:get\_derived1data();

68

69 cout<< "Enter data for Derived 2: " << endl;

70 cin>> data;

71 }

72

73 void derived2::put\_derived2data()

74 {

75 derived1::put\_derived1data();

76 cout<< "Data for Derived 2: " << data << endl;

77 }

78

79 int main()

80 {

81 derived2 d2;

82

83 d2.get\_derived2data();

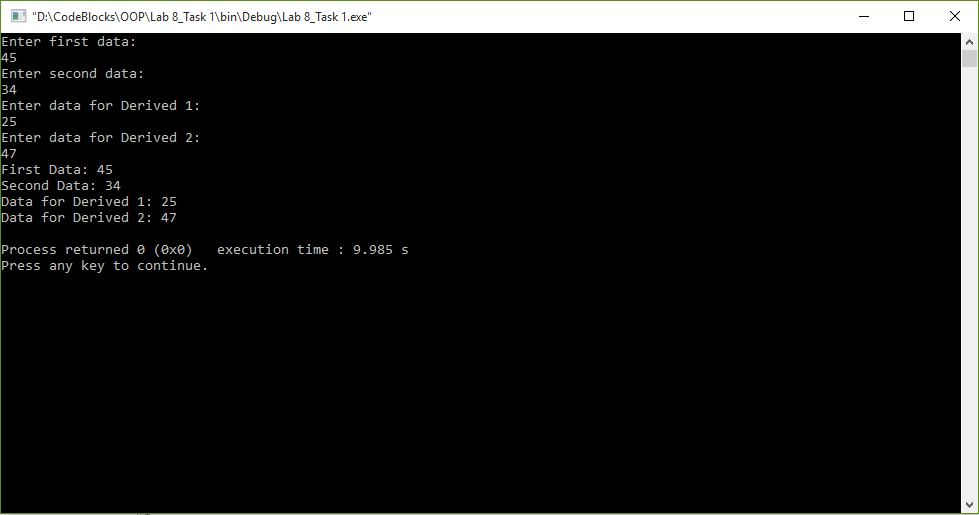
84 d2.put\_derived2data();

85

86 return 0;

87 }

**Output:**



**Task 2:**

**Create a class Person having *name, age* and *gender* as its data members. Create another class Employee which has *employername* and *dailywages* as it data member. From these two classes derive another class teacher which contains teacher grade as data member.**

**i. Write set and get functions to enter and display the data members.**

**ii. Write main function to implement these classes. Enter the teacher data to show multiple inheritance.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Person

7 {

8 **public**:

9 **void** get\_persondata();

10 **void** set\_persondata();

11 **void** set\_persondata(**string** a, **int** b, **char** c);

12

13 **private**:

14 **string** name;

15 **int** age;

16 **char** gender;

17 };

18

19 **void** Person::get\_persondata()

20 {

21 **cout**<< "Name: " << name << **endl**;

22 **cout**<< "Age: " << age << **endl**;

23 **cout**<< "Gender: " << gender << **endl**;

24 }

25

26 **void** Person::set\_persondata()

27 {

28 **cout**<< "Enter Name: " << **endl**;

29 **cin**>> name;

30 **cout**<< "Enter Age: " << **endl**;

31 **cin**>> age;

32 **cout**<< "Enter Gender: " << **endl**;

33 **cin**>> gender;

34 }

35

36 **void** Person::set\_persondata(**string** a, **int** b, **char** c)

37 {

38 name=a;

39 age=b;

40 gender=c;

41 }

42

43 **class** Employee

44 {

45 **public**:

46 **void** get\_empdata();

47 **void** set\_empdata();

48 **void** set\_empdata(**string** a, **int** b);

49

50 **private**:

51 **string** employername;

52 **int** dailywages;

53 };

54

55 **void** Employee::get\_empdata()

56 {

57 **cout**<< "Employer's Name: " << employername << **endl**;

58 **cout**<< "Dailywages: " << dailywages << **endl**;

59 }

60

61 **void** Employee::set\_empdata()

62 {

63 **cout**<< "Enter Employer's Name: " << **endl**;

64 **cin**>> employername;

65 **cout**<< "Enter Dailywages: " << **endl**;

66 **cin**>> dailywages;

67 }

68

69 **void** Employee::set\_empdata(**string** a, **int** b)

70 {

71 employername=a;

72 dailywages=b;

73 }

74

75 **class** Teacher: **public** Person, **public** Employee

76 {

77 **public**:

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

79 **void** set\_data();

80 **void** set\_data(**int** a);

81

82 **private**:

83 **int** grade;

84 };

85

86 **void** Teacher::get\_data()

87 {

88 Person:get\_persondata();

89 Employee:get\_empdata();

90

91 **cout**<< "Teacher's Grade:: " << grade << **endl**;

92 }

93

94 **void** Teacher::set\_data()

95 {

96 **cout**<< "Enter Teacher's Grade: " << **endl**;

97 **cin**>> grade;

98 }

99

100 **void** Teacher::set\_data(**int** a)

101 {

102 grade=a;

103 }

104

105 **int** main()

106 {

107 Teacher t1;

108

109 t1.set\_persondata();

110 t1.set\_empdata();

111 t1.set\_data();

112

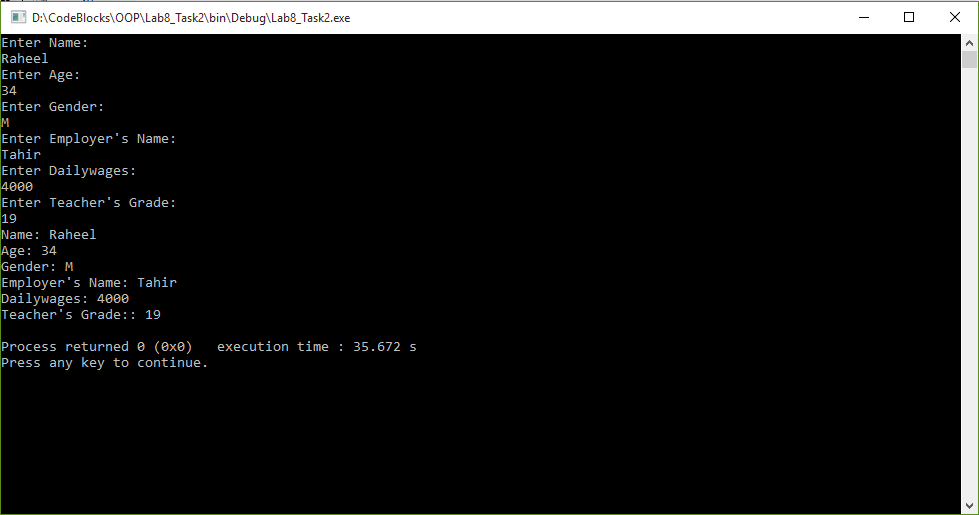
113 t1.get\_data();

114

115 **return** 0;

116 }

**Output:**



**Task 3:**

**Consider the following classes :**

#include <iostream.h>

class Date

{

int day;

int month;

int year;

public:

Date();

~Date();

void display(); // displays the date

Date get(); // accesses the date members

void set(); // sets the date members

};

class Time

{

int hour;

int minute;

int second;

public:

Time();

~Time();

void display(); // displays the time

Time get(); // accesses the time members

void set(); // sets the time members

};

class DateAndTime : public Date, public Time

{

int digital;

public:

void display(); // prints date and time

};

**i. Define an instance object of class DateTime called Watch.**

**ii. Write a main () function that would initialize the values through the constructor functions, and then allows them to be reset through the set () functions. Be sure and display the results following the constructor before you use the set functions.**

**iii. Through the use of the display () function, the time and date are to be displayed. Note that the display () functions in all three classes need to be defined, as well as the constructor and all the access functions.**

**iv. Turn in the .CPP file(s) and screen output.**

**Code:**

1 #include <iostream>

2

3 **using namespace std**;

4

5 **class** Date

6 {

7 **int** day;

8 **int** month;

9 **int** year;

10 **public**:

11 Date(): day(0), month(0), year(0){}

12 Date(**int** d, **int** m, **int** y): day(d), month(m), year(y){}

13 ~Date(){};

14

15 **void** display(); // displays the date

16 **void** get(); // accesses the date members

17 **void set**(**int** d, **int** m, **int** y); // sets the date members

18 };

19

20 **void** Date::display()

21 {

22 **cout**<< "Days: " << day << **endl**;

23 **cout**<< "Months: " << month << **endl**;

24 **cout**<< "Years: " << year << **endl**;

25 }

26

27 **void** Date::get()

28 {

29 **cout**<< "Enter Days: " << **endl**;

30 **cin**>> day;

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

32 **cin**>> month;

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

34 **cin**>> year;

35 }

36

37 **void** Date::**set**(**int** d, **int** m, **int** y)

38 {

39 day=d;

40 month=m;

41 year=y;

42 }

43

44

45 **class** Time

46 {

47 **int** hour;

48 **int** minute;

49 **int** second;

50 **public**:

51 Time(): hour(0), minute(0), second(0){}

52 Time(**int** h, **int min**, **int** s): hour(h), minute(**min**), second(s){}

53 ~Time(){};

54 **void** display(); // displays the time

55 **void** get(); // accesses the time members

56 **void set**(**int** hr, **int min**, **int** sec); // sets the time members

57 };

58

59 **void** Time::display()

60 {

61 **cout**<< "Hour: " << hour << **endl**;

62 **cout**<< "Minutes: " << minute << **endl**;

63 **cout**<< "Seconds: " << second << **endl**;

64 }

65

66 **void** Time::get()

67 {

68 **cout**<< "Enter Hour: " << **endl**;

69 **cin**>> hour;

70 **cout**<< "Enter Minutes: " << **endl**;

71 **cin**>> minute;

72 **cout**<< "Enter Seconds: " << **endl**;

73 **cin**>> second;

74 }

75

76 **void** Time::**set**(**int** hr, **int min**, **int** sec)

77 {

78 hour=hr;

79 minute=**min**;

80 second=sec;

81 }

82

83 **class** DateAndTime : **public** Date, **public** Time

84 {

85 **int** digital;

86 **public**:

87 DateAndTime():Date(), Time(), digital(0){}

88 DateAndTime(**int** d, **int** m, **int** y, **int** h, **int min**, **int** s,

89 **int** dig):Date(d,m,y), Time(h,**min**,s), digital(dig){}

90 ~DateAndTime(){};

91 **void** display(); // prints date and time

92 **void** get();

93 **void set**(**int** hr, **int min**, **int** sec, **int** d, **int** m, **int** y, **int** dig);

94 };

95

96 **void** DateAndTime::display()

97 {

98 Time::display();

99 Date::display();

100

101 **cout**<< "Digital: " << digital << **endl**;

102 }

103

104 **void** DateAndTime::get()

105 {

106 Time::get();

107 Date::get();

108

109 cout<< "Enter value for Digital: " << endl;

110 cin>> digital;

111 }

112

113 void DateAndTime::set(int hr, int min, int sec, int d, int m, int y, int dig)

114 {

115 Time::set(hr, min, sec);

116 Date::set(d, m, y);

117

118 digital=dig;

119 }

120

121 int main()

122 {

123 DateAndTime Watch;

124

125 Watch.get();

126 Watch.display();

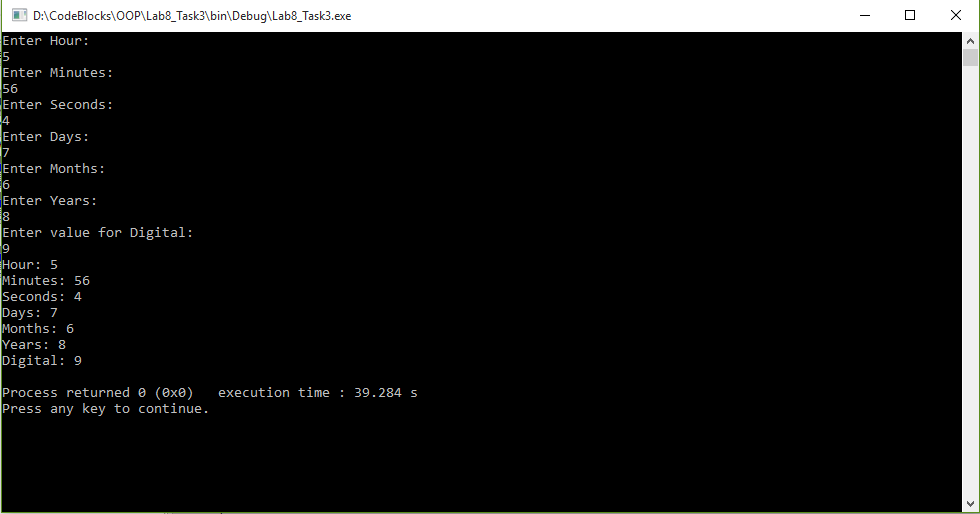
127

128

129 return 0;

130 }

**Output:**



**Task 4:**

**Write a class Teacher that contains the attribute *teacher name, age* and *address*. It also contains member function to input and display its attributes. Write another class Author that contains the attributes *author name, address* and *number of books* written by him. It also contains member functions to input and display its attributes. Write a third class Scholar that inherits both Teacher and Author classes. Test these classes from main() by creating objects of derived classes and testing functions in a way that clear concept of multiple Inheritance.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Teacher

7 {

8 **public**:

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

10 **void** set\_data();

11 **void** set\_data(**string** n, **int** a, **string** add);

12

13 **private**:

14 **string** teachername;

15 **int** age;

16 **string** address;

17 };

18

19 **void** Teacher::get\_data()

20 {

21 **cout**<< "Teacher's Name:: " << teachername << **endl**;

22 **cout**<< "Teacher's Age:: " << age << **endl**;

23 **cout**<< "Teacher's Address:: " << address << **endl**;

24 }

25

26 **void** Teacher::set\_data()

27 {

28 **cout**<< "Enter Teacher's Name: " << **endl**;

29 **cin**>> teachername;

30 **cout**<< "Enter Teacher's Age: " << **endl**;

31 **cin**>> age;

32 **cout**<< "Enter Teacher's Address: " << **endl**;

33 **cin**>> address;

34 }

35

36 **void** Teacher::set\_data(**string** n, **int** a, **string** add)

37 {

38 teachername=n;

39 age=a;

40 address=add;

41 }

42

43 **class** Author

44 {

45 **public**:

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

47 **void** set\_data();

48 **void** set\_data(**string** n, **string** add, **int** nb);

49

50 **private**:

51 **string** author\_name;

52 **string** address;

53 **int** num\_books;

54 };

55

56 **void** Author::get\_data()

57 {

58 **cout**<< "Author's Name: " << author\_name << **endl**;

59 **cout**<< "Author's Address: " << address << **endl**;

60 **cout**<< "Number of Books: " << num\_books << **endl**;

61 }

62

63 **void** Author::set\_data()

64 {

65 **cout**<< "Enter Author's Name: " << **endl**;

66 **cin**>> author\_name;

67 **cout**<< "Enter Author's Address: " << **endl**;

68 **cin**>> address;

69 **cout**<< "Enter Number of Books written by him: " << **endl**;

70 **cin**>> num\_books;

71 }

72

73 **void** Author::set\_data(**string** n, **string** add, **int** nb)

74 {

75 author\_name=n;

76 address=add;

77 num\_books=nb;

78 }

79

80 **class** Scholar:**public** Teacher, **public** Author

81 {

82 **public**:

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

84 **void** set\_data();

85 **void** set\_data(**string** t\_n, **int** t\_a, **string** t\_add, **string** a\_n, **string** a\_add, **int** a\_nb, **string** q);

86

87 **private**:

88 **string** qualification;

89 };

90

91 **void** Scholar::get\_data()

92 {

93 Teacher:: get\_data();

94 Author:: get\_data();

95

96 **cout**<< "Employer's Qualification: " << qualification << **endl**;

97 }

98

99 **void** Scholar::set\_data()

100 {

101 Teacher::set\_data();

102 Author::set\_data();

103

104 **cout**<< "Enter Scholar's Qualification: " << **endl**;

105 **cin**>> qualification;

106 }

107

108 **void** Scholar::set\_data(**string** t\_n, **int** t\_a, **string** t\_add, **string** a\_n, **string** a\_add, **int** a\_nb, **string** q)

109 {

110 Teacher::set\_data(t\_n, t\_a, t\_add);

111 Author::set\_data(a\_n, a\_add, a\_nb);

112

113 qualification=q;

114 }

115

116 int main()

117 {

118 Scholar s1;

119

120 s1.set\_data();

121

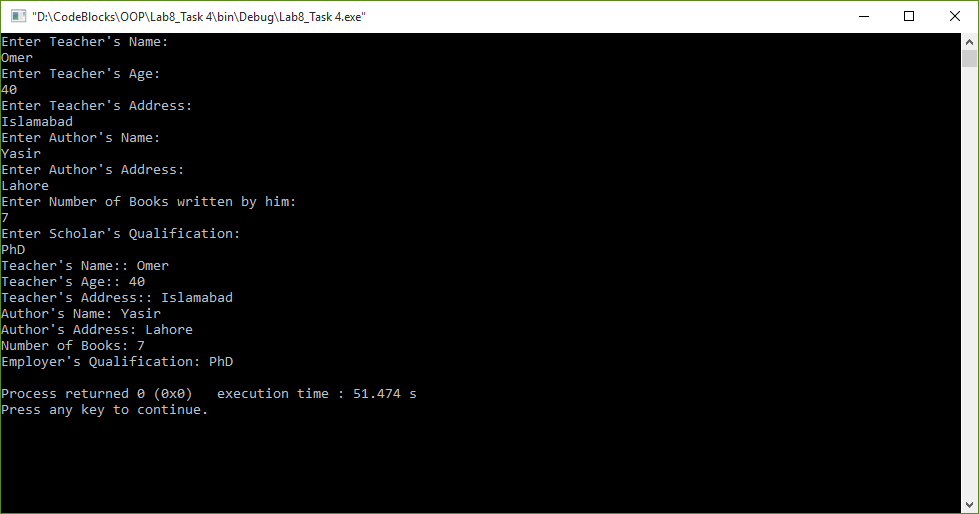
122 s1.get\_data();

123

124 return 0;

125 }

**Output:**

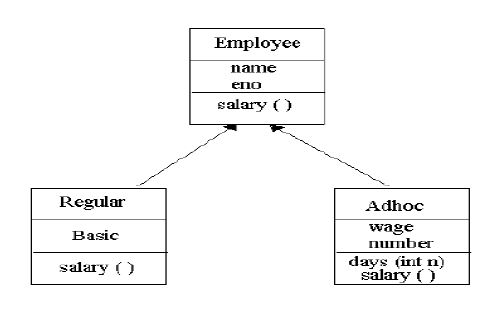


**Post Lab:**

**Task 1:**

**An organization has two types of employees: regular and adhoc. Regular employees get a salary which is basic + DA + HRA where DA is 10% of basic and HRA is 30% of basic. Adhoc employees are daily wagers who get a salary which is equal to Number \* Wage.**

**(i) Define the classes shown in the following class hierarchy diagram:**



**(ii) Define the constructors. When a regular employee is created, basic must be a parameter.**

**When adhoc employee is created wage must be a parameter.**

**(iii) Define the destructors.**

**(iv) Define the member functions for each class. The member function days ( ) updates**

**number of the Adhoc employee.**

**(v) Write a test program to test the classes.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Teacher

7 {

8 **public**:

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

10 **void** set\_data();

11 **void** set\_data(**string** n, **int** a, **string** add);

12

13 **private**:

14 **string** teachername;

15 **int** age;

16 **string** address;

17 };

18

19 **void** Teacher::get\_data()

20 {

21 **cout**<< "Teacher's Name:: " << teachername << **endl**;

22 **cout**<< "Teacher's Age:: " << age << **endl**;

23 **cout**<< "Teacher's Address:: " << address << **endl**;

24 }

25

26 **void** Teacher::set\_data()

27 {

28 **cout**<< "Enter Teacher's Name: " << **endl**;

29 **cin**>> teachername;

30 **cout**<< "Enter Teacher's Age: " << **endl**;

31 **cin**>> age;

32 **cout**<< "Enter Teacher's Address: " << **endl**;

33 **cin**>> address;

34 }

35

36 **void** Teacher::set\_data(**string** n, **int** a, **string** add)

37 {

38 teachername=n;

39 age=a;

40 address=add;

41 }

42

43 **class** Author

44 {

45 **public**:

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

47 **void** set\_data();

48 **void** set\_data(**string** n, **string** add, **int** nb);

49

50 **private**:

51 **string** author\_name;

52 **string** address;

53 **int** num\_books;

54 };

55

56 **void** Author::get\_data()

57 {

58 **cout**<< "Author's Name: " << author\_name << **endl**;

59 **cout**<< "Author's Address: " << address << **endl**;

60 **cout**<< "Number of Books: " << num\_books << **endl**;

61 }

62

63 **void** Author::set\_data()

64 {

65 **cout**<< "Enter Author's Name: " << **endl**;

66 **cin**>> author\_name;

67 **cout**<< "Enter Author's Address: " << **endl**;

68 **cin**>> address;

69 **cout**<< "Enter Number of Books written by him: " << **endl**;

70 **cin**>> num\_books;

71 }

72

73 **void** Author::set\_data(**string** n, **string** add, **int** nb)

74 {

75 author\_name=n;

76 address=add;

77 num\_books=nb;

78 }

79

80 **class** Scholar:**public** Teacher, **public** Author

81 {

82 **public**:

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

84 **void** set\_data();

85 **void** set\_data(**string** t\_n, **int** t\_a, **string** t\_add, **string** a\_n, **string** a\_add, **int** a\_nb, **string** q);

86

87 **private**:

88 **string** qualification;

89 };

90

91 **void** Scholar::get\_data()

92 {

93 Teacher:: get\_data();

94 Author:: get\_data();

95

96 **cout**<< "Employer's Qualification: " << qualification << **endl**;

97 }

98

99 **void** Scholar::set\_data()

100 {

101 Teacher::set\_data();

102 Author::set\_data();

103

104 **cout**<< "Enter Scholar's Qualification: " << **endl**;

105 **cin**>> qualification;

106 }

107

108 **void** Scholar::set\_data(**string** t\_n, **int** t\_a, **string** t\_add, **string** a\_n, **string** a\_add, **int** a\_nb, **string** q)

109 {

110 Teacher::set\_data(t\_n, t\_a, t\_add);

111 Author::set\_data(a\_n, a\_add, a\_nb);

112

113 qualification=q;

114 }

115

116 int main()

117 {

118 Scholar s1;

119

120 s1.set\_data();

121

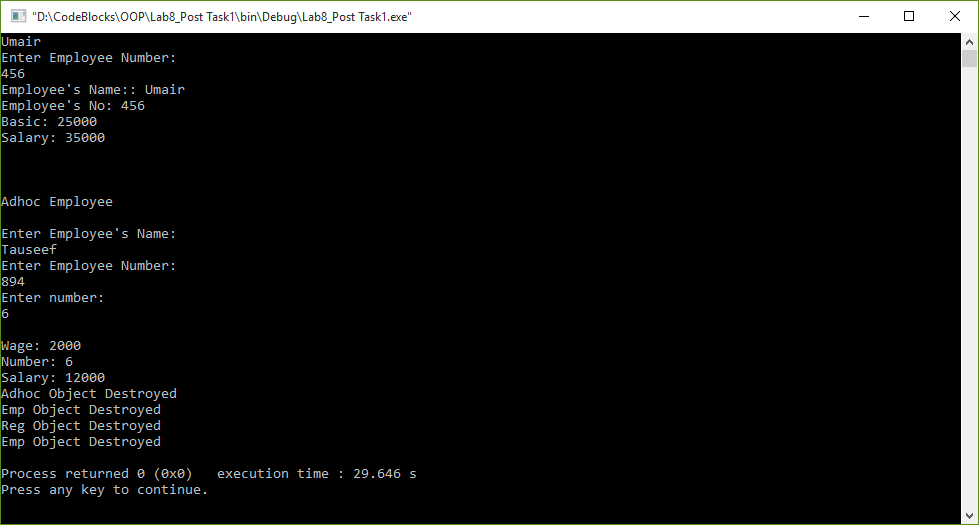
122 s1.get\_data();

123

124 return 0;

125 }

**Output:**



**Task 2:**

**Write a program having a base class Student with data member *rollno* and member functions *getnum*() to input *rollno* and *putnum*() to display *rollno*. A class Test is derived from class Student with data member *marks* and member functions *getmarks*() to input *marks* and *putmarks*() to display *marks*. Class Sports is also derived from class Student with data member *score* and member functions *getscore*() to input *score* and *putscore*() to display *score*. The class Result is inherited from two base classes, class Test and class Sports with data member *total* and a member function *display*() to display *rollno*, *marks*, *score* and the *total*(*marks* + *score*).**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** Student

7 {

8 **public**:

9 **void** get\_num();

10 **void** put\_num();

11

12 **private**:

13 **int** roll\_no;

14 };

15

16 **void** Student::get\_num()

17 {

18 **cout**<< "Enter Student's Roll no: " << **endl**;

19 **cin**>> roll\_no;

20 }

21

22 **void** Student::put\_num()

23 {

24 **cout**<< "Roll no: " << roll\_no << **endl**;

25 }

26

27 **class** Test: **public** Student

28 {

29 **public**:

30 **void** get\_marks();

31 **void** put\_marks();

32

33 **protected**:

34 **float** marks;

35 };

36

37 **void** Test::get\_marks()

38 {

39 Student::get\_num();

40

41 **cout**<< "Enter Test Marks: " << **endl**;

42 **cin**>> marks;

43 }

44

45 **void** Test::put\_marks()

46 {

47 Student::put\_num();

48

49 **cout**<< **endl** << "Marks: " << marks << **endl**;

50 }

51

52 **class** Sports:**public** Student

53 {

54 **public**:

55 **void** get\_score();

56 **void** put\_score();

57

58 **protected**:

59 **float** score;

60 };

61

62 **void** Sports::get\_score()

63 {

64 **cout**<< "Enter Score: " << **endl**;

65 **cin**>> score;

66 }

67

68 **void** Sports::put\_score()

69 {

70 **cout**<< "Score: " << score << **endl**;

71 }

72

73 **class** Result:**public** Test, **public** Sports

74 {

75 **public**:

76 **void** display();

77

78 **private**:

79 **float** total;

80 };

81

82 **void** Result::display()

83 {

84 Test::put\_marks();

85 Sports::put\_score();

86

87 total=marks+score;

88 **cout**<<**endl** << "Total: " << total <<**endl**;

89 }

90

91 **int** main()

92 {

93 Result r1;

94

95 r1.get\_marks();

96 r1.get\_score();

97

98 cout<<endl <<endl;

99

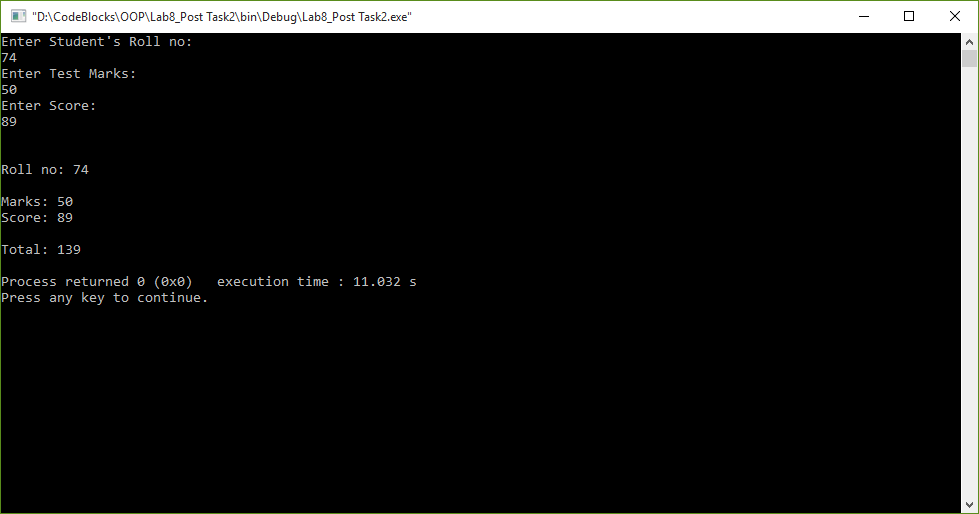
100 r1.display();

101

102 return 0;

103 }

**Output:**



**Task 3:**

**Write a class LocalPhone that contains an attribute *phone* to store a local telephone number. The class contains member functions to input and display phone number. Write a child class NatPhone for national phone numbers that inherits LocPhone class. It additionally contains an attribute to store *city code*. It also contains member functions to input and show the city code. Write another class IntPhone for international phone numbers that inherit NatPhone class. It additionally contains an attribute to store *country code*. It also contains member functions to input and show the country code. Test these classes from main() by creating objects of derived classes and testing functions in a way that clear concept of multi-level Inheritance.**

**Code:**

1 #include <iostream>

2 #include <string>

3

4 **using namespace std**;

5

6 **class** LocalPhone

7 {

8 **public**:

9 **void** get\_num();

10 **void** put\_num();

11

12 **private**:

13 **long** phone;

14 };

15

16 **void** LocalPhone::get\_num()

17 {

18 **cout**<< "Enter Phone no: " << **endl**;

19 **cin**>> phone;

20 }

21

22 **void** LocalPhone::put\_num()

23 {

24 cout<< "Phone no: " << phone << endl;

25 }

26

27 class NatPhone: public LocalPhone

28 {

29 public:

30 void get\_code();

31 void put\_code();

32

33 private:

34 int city\_code;

35 };

36

37 void NatPhone::get\_code()

38 {

39 LocalPhone::get\_num();

40

41 cout<< "Enter City Code: " << endl;

42 cin>> city\_code;

43 }

44

45 void NatPhone::put\_code()

46 {

47 LocalPhone::put\_num();

48

49 cout<< "City Code: " << city\_code << endl;

50 }

51

52 class IntPhone:public NatPhone

53 {

54 public:

55 void get\_country\_code();

56 void put\_country\_code();

57

58 private:

59 int country\_code;

60 };

61

62 void IntPhone::get\_country\_code()

63 {

64 NatPhone::get\_code();

65

66 cout<< "Enter Country Code: " << endl;

67 cin>> country\_code;

68 }

69

70 void IntPhone::put\_country\_code()

71 {

72 NatPhone::put\_code();

73

74 cout<< "Country Code: " << country\_code << endl;

75 }

76

77 int main()

78 {

79 IntPhone ip1;

80

81 ip1.get\_country\_code();

82 ip1.put\_country\_code();

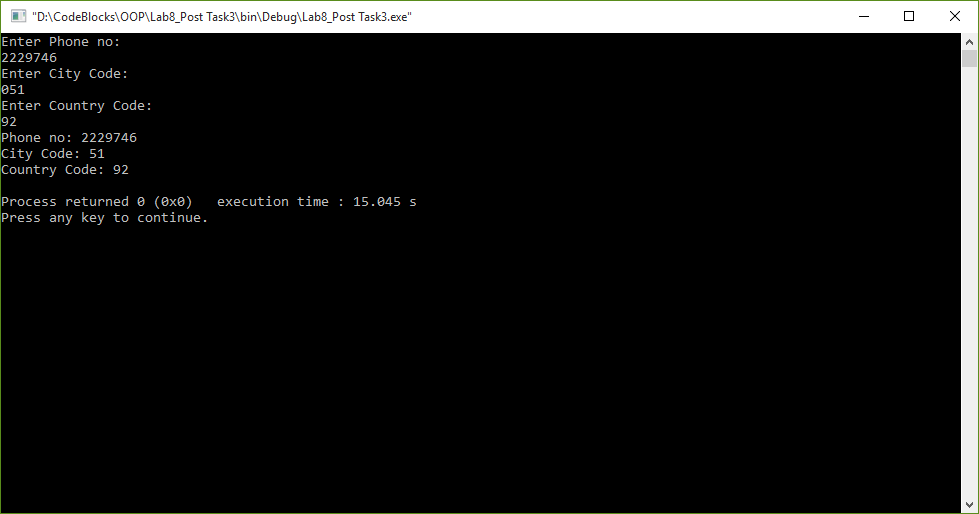
83

84 return 0;

85 }

86

**Output:**



**Task 4:**

**Start with the publication, book and tape classes. Add base class sales that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to get three sale amount from the user and a putdata() function to display the sales figure.**

**Alter the book and tape classes, so they are derived from both publication and sales.**

**An object of book or tape should should input and output ans sales data along with other data.**

**Write a main function to create a book and tape object and exercise their input/output capabilities.**

**Code:**

1 #include<iostream>

2 #include<conio.h>

3 #include<string>

4

5 **using namespace std**;

6

7 **class** Sales

8 {

9 **public**:

10 **void** get\_dollar\_sales();

11 **void** put\_dollar\_sales();

12 **private**:

13 **float array**[2];

14 **float** dollar\_sales;

15 };

16

17 **void** Sales::get\_dollar\_sales()

18 {

19 **for**(**int** i=0; i<=2; i++)

20 {

21 **cout**<< "Enter Dollar Sales:" <<**endl**;

22 **cin**>> dollar\_sales;

23

24 array[i]=dollar\_sales;

25 }

26 }

27

28 void Sales::put\_dollar\_sales()

29 {

30 for(int i=0; i<=2; i++)

31 {

32 cout<< endl << "Dollar Sales:" << array[i] <<endl;

33 }

34 }

35

36 class Publication

37 {

38 public:

39 void get\_data();

40 void put\_data();

41 private:

42 string title;

43 int price;

44 };

45

46 void Publication::get\_data()

47 {

48 cout << "Enter the Title of Publication: " << endl;

49 cin >> title;

50 cout << "Enter the Price of Publication:" << endl;

51 cin >> price;

52 }

53

54 void Publication::put\_data()

55 {

56 cout << "Title of Publication: " << title <<endl;

57 cout << "Price of Publication: " << price << endl;

58 }

59

60 class Book:public Publication, public Sales

61 {

62 public:

63 void get\_data();

64 void put\_data();

65 private:

66 int page\_count;

67 };

68

69 void Book::get\_data()

70 {

71 cout << "Enter the Number of Page: " << endl;

72 cin >> page\_count;

73 }

74

75 void Book::put\_data()

76 {

77 cout << "Number of Pages: " << page\_count << endl << endl;

78 }

79

80 class Tape : public Publication, public Sales

81 {

82 public:

83 void get\_data();

84 void put\_data();

85 private:

86 int playing\_time;

87 };

88

89 void Tape::get\_data()

90 {

91 cout << endl << "Enter the playing time in minutes: " << endl;

92 cin >> playing\_time;

93 }

94

95 void Tape::put\_data()

96 {

97 cout << "Playing Time: " << playing\_time << endl << endl;

98 }

99

100 int main ()

101 {

102 Book b1;

103 Tape t1;

104

105 b1.get\_data();

106 b1.put\_data();

107 b1.get\_dollar\_sales();

108 b1.put\_dollar\_sales();

109

110 t1.get\_data();

111 t1.put\_data();

112 t1.get\_dollar\_sales();

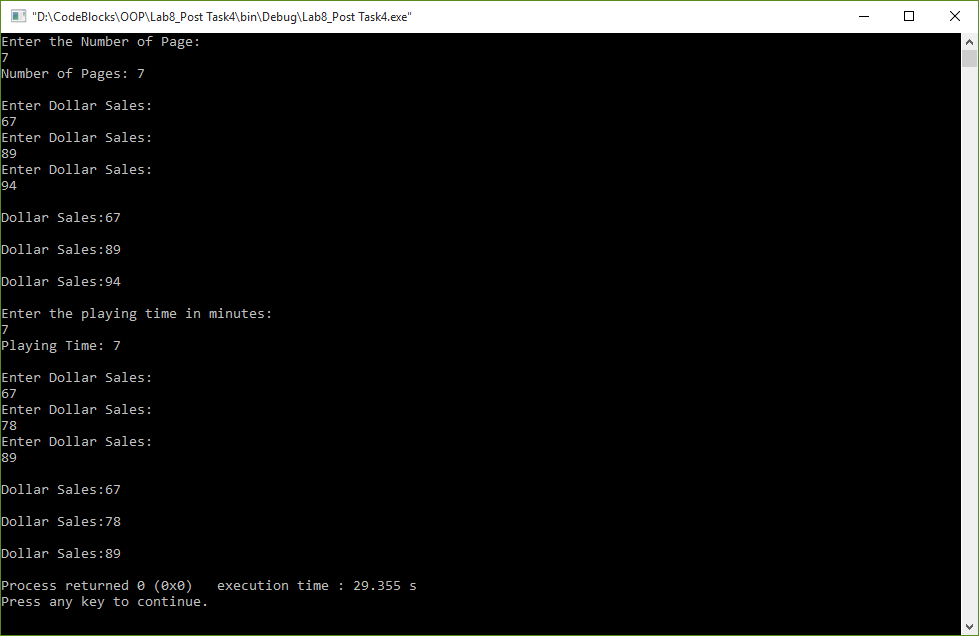
113 t1.put\_dollar\_sales();

114

115 return 0;

116 }

**Output:**



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

**THE END**