**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

Department of Computer Engineering/Computer Science & Engineering/

Information Technology

**Subject Name: Object Oriented Programming with C++**
**Semester: II**
**Subject Code: CE144**
**Academic year: 2020-21**

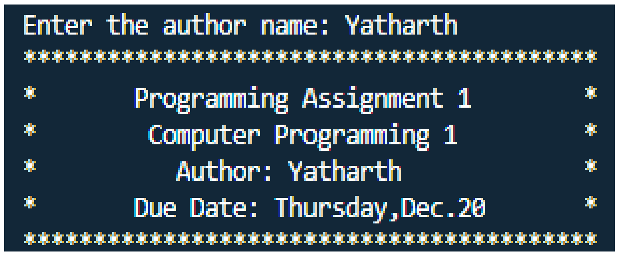
**ID: 20DCE019**

# Practical List

| No. | Aim of the Practical |
|---|---|
| 1. | **Write a C++ program that will print output in the following form. Make sure your output looks exactly as shown here (including spacing, line breaks, punctuation, and the title and author). Use cout and cin objects and endl manipulator.**<br>`**********************************`<br>`*      Programming Assignment 1       *`<br>`*       Computer Programming 1        *`<br>`*              Author: ???             *`<br>`*      Due Date: Thursday,Dec.20       *`<br>`**********************************`<br>**Question: Difference between \n and endl.**<br><br>**PROGRAM CODE :**<br><br>#include<iostream> |

```
using namespace std;
int main()
{
   string author;

   cout<<"Enter the author name: ";
   cin>>author;

   cout<<"*****************************************"<<"\n";
   cout<<"*\tProgramming Assignment 1\t*"<<"\n";
   cout<<"*\t Computer Programming 1\t\t*"<<"\n";
   cout<<"*\t   "<<"Author: "<<author<<"\t\t*"<<"\n";
   cout<<"*\tDue Date: Thursday,Dec.20\t*"<<"\n";
   cout<<"*****************************************";
}
```

**<u>OUTPUT:</u>**



```
Enter the author name: Yatharth
*****************************************
*       Programming Assignment 1        *
*        Computer Programming 1         *
*          Author: Yatharth             *
*       Due Date: Thursday,Dec.20       *
*****************************************
```

**<u>QUESTION:</u> Difference between \n and endl.**

\n is a string of length 1 that gets appended to stdout.

endl is an object that will cause to append the newline character ("\n") and to flush stdout buffer.

**<u>CONCLUSION</u>:** In this practical we Learn how to use cin and cout.

| **No.** | **Aim of the Practical** |
|---|---|

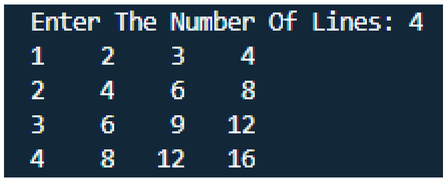
**2.** **Write a program to create the following table.**
**Use endl and setw manipulator.**

```
1   2   3   4
2   4   6   8
3   6   9  12
4   8  12  16
```

**PROGRAM CODE :**

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int i, j, n;
    cout << "Enter The Number Of Lines: ";
    cin >> n;

    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            cout << i * j;
            cout << setw(5);
        }
        cout << "\n";
    }
}
```

**OUTPUT:**



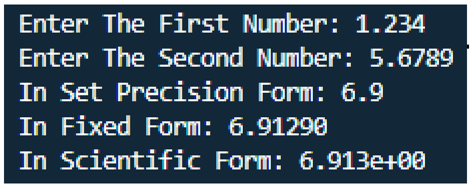
```
Enter The Number Of Lines: 4
1    2    3    4
2    4    6    8
3    6    9   12
4    8   12   16
```

**CONCLUSION:** Learnt how to use setw and the syntax of setw .

| No. | Aim of the Practical |
|---|---|
| 3. | **Write a C++ program to add two floating numbers using pointer. The result should contain only two digits after the decimal. Use fixed, scientific and setprecision() manipulators for controlling the precision of floating point numbers.** |

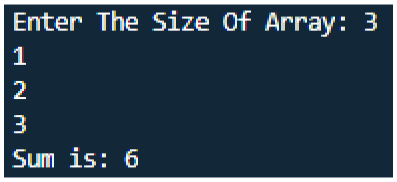
**PROGRAM CODE :**

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   float n1, n2, sum;
   float *p1, *p2;

   p1 = &n1;
   p2 = &n2;

   cout << "Enter The First Number: ";
   cin >> *p1;
   cout << "Enter The Second Number: ";
   cin >> *p2;
```

|  |  |
|---|---|
|  | `sum = *p1 + *p2;`<br><br>`cout << "In Set Precision Form: " << setprecision(2) << sum << endl;`<br>`cout << "In Fixed Form: " << fixed << setprecision(5) << sum << endl;`<br>`cout << "In Scientific Form: " << scientific << setprecision(3) << sum << endl;`<br>`}`<br><br>**<u>OUTPUT:</u>**<br><br>`Enter The First Number: 1.234`<br>`Enter The Second Number: 5.6789`<br>`In Set Precision Form: 6.9`<br>`In Fixed Form: 6.91290`<br>`In Scientific Form: 6.913e+00`<br><br>**<u>CONCLUSION</u>:** Learnt how to use setprecision(), fixed, scientific in c++ along their syntax. |

| No. | Aim of the Practical |
|---|---|
| 4. | **Write a C++ program to add two floating numbers using pointer. The result should contain only two digits after the decimal. Use fixed, scientific and setprecision() manipulators for controlling the precision of floating point numbers.**<br><br>**<u>PROGRAM CODE :</u>** |

```
#include <iostream>
using namespace std;

int arr_add(int a[], int n)
{
   static int sum = 0;
   static int i = 0;

   if (i < n)
   {
      sum = sum + a[i];
      i++;

      arr_add(a, n);
   }
   return sum;
}
int main()
{
   int n;

   cout << "Enter The Size Of Array: ";
   cin >> n;

   int  i, sum, a[n];

   for (i = 0; i < n; i++)
   {
      cin >> a[i];
   }
   sum = arr_add(a, n);
   cout << "Sum is: " << sum;
}
```

| | |
|---|---|
| | **<u>OUTPUT:</u>**<br><br>`Enter The Size Of Array: 3`<br>`1`<br>`2`<br>`3`<br>`Sum is: 6`<br><br>**<u>QUESTION</u>: Show stepwise solution of winding and unwinding phase of recursion.**<br><br>**Winding phase**: In it the recursive function keeps calling itself. This phase ends when the base condition is reached.<br>**Unwinding phase**: When the base condition is reached, unwinding phase starts and control returns back to the original call.<br><br>**<u>CONCLUSION</u>:** In this practical we learn code using recursion and also winding and unwinding phase of recursion. |

| **No.** | **Aim of the Practical** |
|---|---|
| **5.** | **Find error in the following code and give reasons for each error: Can we declare an array of references? Can we assign NULL value to reference variable? Is Reference variable a pointer variable? Can we declare a reference variable without initializing it? Does Reference Variable change the original value of variable?**<br><br>**#include &lt;iostream&gt;**<br>**using namespace std;**<br>**int main()**<br>**{**<br>   **int no1 = 10, no2 = 12;**<br>   **int &x = no1;**<br>   **int &r;**<br>   **int &c = NULL;** |

```
    int &d[2] = {no1, no2};
    cout << "x = " << x + 20;
    cout << "no1=" << no1 + 10;
    return 0;
}
```

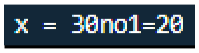
**Error:** int & r;
**Reason:** reference operator must be initialized.

**Error:** int & c = NULL;
**Reason:** reference operator cannot be initialized to null.

**Error:** int & d[2] = {no1,no2};
**Reason:** reference operators cannot be combined to form an array.

**<u>PROGRAM CODE :</u>**

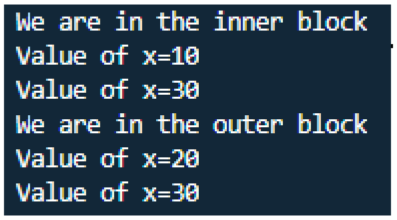
```
#include <iostream>
using namespace std;

int main()
{
    int no1 = 10, no2 = 12;

    int &x = no1;
    int &r = no2;
    int &c = no1;

    int d[2] = {no1, no2};

    cout << "x = " << x + 20;
    cout << "no1=" << no1 + 10;

    return 0;
```

| | |
|---|---|
| | } <br><br> **OUTPUT:** <br><br>  <br> `x = 30no1=20` <br><br> **CONCLUSION:** Learnt that a reference operator cannot be initialized to null and there cannot be an array of reference operator and a reference operator must be initialized. |

| No. | Aim of the Practical |
|---|---|
| 6. | **Find output of the following code: Explain how scope Resolution operator is used to access global version of a variable.** <br><br> **#include <iostream.h>** <br> **#include <conio.h>** <br> **int m = 30;** <br> **int main()** <br> **{** <br>    **int m = 20;** <br>    **{** <br>       **int m = 10;** <br>       **cout <<”we are in inner block”<< endl;** <br>       **cout <<”value of m =”<< m <<”\n”;** <br>       **cout <<”value of ::m =”<< ::m <<”\n”;** <br>    **}** <br>    **cout <<”we are in outer block”<< endl;** <br>    **cout <<”value of m =”<< m <<”\n”;** <br>    **cout <<”value of ::m =”<< ::m <<”\n”;** <br>    **getch();** <br>    **return 0;** |

```
}
```

**OUTPUT:**



```
We are in the inner block
Value of x=10
Value of x=30
We are in the outer block
Value of x=20
Value of x=30
```

**EXPLANATION:** The scope operator defines the scope of a variable and also if can be defined as if the global variable name is same as local variable name, the scope resolution operator will be used to call the global variable. It is also used to define a function outside the class and used to access the static variables of class and hence here we used that just to symbolize its effectiveness in a c++ program.

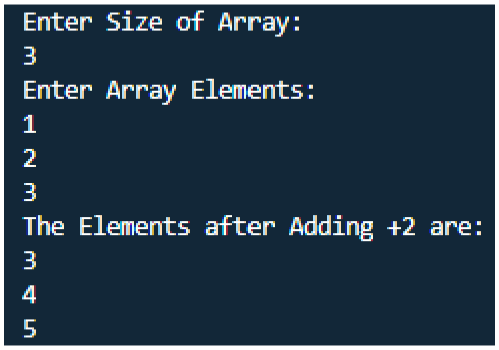
**CONCLUSION:** In this practical we learn about how scope Resolution operator is used to access global version of a variable. It is declared as :: variable.

| No. | Aim of the Practical |
|---|---|
| 7. | **Write a program to enter a size of array. Create an array of size given by user using "new" Dynamic memory management operator (free store operator). Enter the data to store in array and display the data after adding 2 to each element in the array. Delete the array by using "delete" memory management operator.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

int main()
{
    int n;

    cout << "Enter Size of Array: " << endl;
    cin >> n;

    int *ptr = new int;

    cout << "Enter Array Elements: " << endl;
    for (int i = 0; i < n; i++)
    {
        cin >> ptr[i];
    }

    cout << "The Elements after Adding +2 are: " << endl;
    for (int i = 0; i < n; i++)
    {
        cout << ptr[i] + 2 << endl;
    }
    delete[] ptr;
}
```

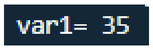
**OUTPUT:**



```
Enter Size of Array:
3
Enter Array Elements:
1
2
3
The Elements after Adding +2 are:
3
4
5
```

**CONCLUSION:** In this practical we learn about ‘new’ and ‘delete’ operator in c++ program and also know their syntax and uses of them.

| No. | Aim of the Practical |
|---|---|
| 8. | **Find Error in the following code of a program and give explanation why these errors exist.**<br><br>**1. //This is an example of constant pointer**<br><br>**#include <iostream>**<br>**using namespace std;**<br>**int main()**<br>**{**<br>   **int var1 = 35, var2 = 20;**<br>   **int *const ptr = &var1;**<br>   **ptr = &var2;**<br>   **cout << "var1= " << *ptr;**<br>   **return 0;**<br>**}**<br><br>**Error:** ptr = &var2;<br>**Reason:** value of constant pointer can not be change.<br>**Solution:** //ptr = &var2;<br><br>**<u>PROGRAM CODE :</u>**<br><br>#include <iostream><br>using namespace std;<br>int main()<br>{<br>   int var1 = 35, var2 = 20;<br>   int *const ptr = &var1;<br>   //ptr = &var2;<br>   cout << "var1= " << *ptr;<br>   return 0;<br>}<br><br>**<u>OUTPUT:</u>** |



```
var1= 35
```

**2. //This is an example of pointer to constant**

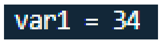
```
#include <iostream>
using namespace std;
int main()
{
   int var1 = 43;
   const int *ptr = &var1;
   *ptr = 1;
   var1 = 34;
   cout << "var1 = " << *ptr;
   return 0;
}
```

**Error:** *ptr = 1;
**Reason:** constant  variable can not be initialized by any constant.
**Solution:** //*ptr = 1;

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
int main()
{
   int var1 = 43;
   const int *ptr = &var1;
   //*ptr = 1;
   var1 = 34;
   cout << "var1 = " << *ptr;
   return 0;
}
```

**<u>OUTPUT:</u>**



```
var1 = 34
```

**3. //This is an example of constant pointer to a**

```
#include <iostream>
using namespace std;
int main()
{
   int var1 = 0, var2 = 0;
   const int *const ptr = &var1;
   *ptr = 1;
   ptr = &var2;
   cout << "Var1 = " << *ptr;
   return 0;
}
```

**Error (1) :** *ptr = 1;
**Error (2) :** *ptr = &var2;

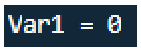
**Reason (1) :** value of constant pointer can not be change.
**Reason (2) :** constant  variable can not be initialized by any constant.

**Solution (1) :** //ptr = &var2;
**Solution (2) : //***ptr = 1

**<u>PROGRAM CODE :</u>**

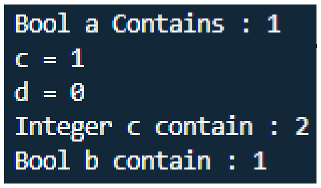
```
#include <iostream>
using namespace std;
int main()
{
   int var1 = 0, var2 = 0;
   const int *const ptr = &var1;
   //*ptr = 1;    //Combination of both 1 and 2 Practical
   //ptr = &var2;
   cout << "Var1 = " << *ptr;
   return 0;
}
```

**<u>OUTPUT:</u>**



```
Var1 = 0
```

|  |  |
|---|---|
|  | **CONCLUSION:** In this practical we know that value of constant pointer can not be change and also constant variable can not be initialized by any constant. |

| No. | Aim of the Practical |
|---|---|
| 9. | **Find the output of following program. Explain the use of bool data type.**<br>`#include <iostream>`<br>`using namespace std;`<br>`int main()`<br>`{`<br>`    bool a = 321, b;`<br>`    cout << "Bool a Contains : " << a << endl;`<br>`    int c = true;`<br>`    int d = false;`<br>`    cout << "c = " << c << endl`<br>`         << "d = " << d;`<br>`    c = a + a;`<br>`    cout << "\nInteger c contain : " << c;`<br>`    b = c + a;`<br>`    cout << "\nBool b contain : " << b;`<br>`    return 0;`<br>`}`<br><br>**OUTPUT:**<br><br>`Bool a Contains : 1`<br>`c = 1`<br>`d = 0`<br>`Integer c contain : 2`<br>`Bool b contain : 1` |

| | |
|---|---|
| | **QUESTION: Explain the use of bool data type.** **ANSWER:** The bool data type is used to represent boolean values - True or False. The data type can't contain any other value. **CONCLUSION:** In this practical we learn the use of Boolean datatype and its true for 1 and false for 0. |

| **No.** | **Aim of the Practical** |
|---|---|
| **10.** | **Define three functions named divide(). First function takes numerator and denominator as an input argument and checks it's divisible or not, Second function takes one int number as input argument and checks whether the number is prime or not and Third function takes 3 float number as argument and finds out average of the numbers. Use concept of Function Overloading / static binding.** |

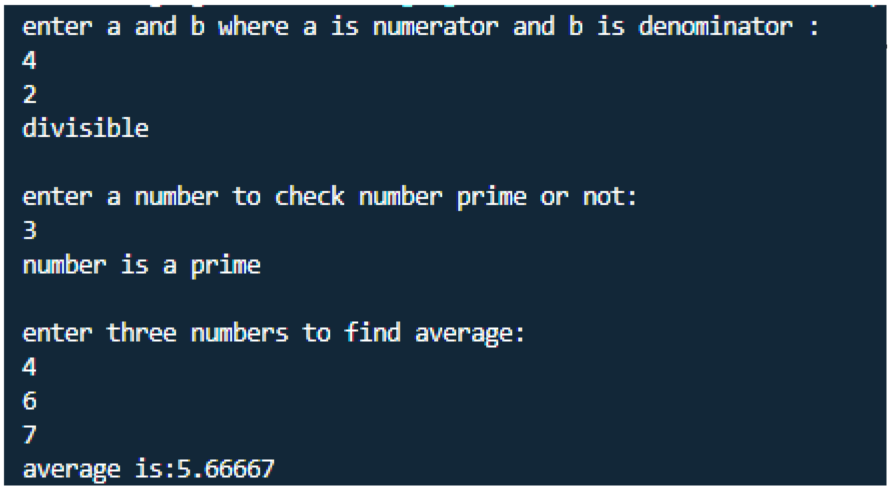
**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

int divide(int a, int b);
int divide(int a);
float divide(float a, float b, float c);

int main()
{
    int a, b;

    cout << "enter a and b where a is numerator and b is denominator : " << endl;
    cin >> a >> b;

    if (divide(a, b))
```

```
        cout << "divisible" << endl;
    else
        cout << "not divisible" << endl;

    int p;

    cout << endl;
    cout << "enter a number to check number prime or not:" << endl;
    cin >> p;

    if (divide(p))
        cout << "number is a prime" << endl;
    else
        cout << "number is not prime" << endl;

    float m, n, o, avg;

    cout << endl;
    cout << "enter three numbers to find average: " << endl;
    cin >> m >> n >> o;

    avg = divide(m, n, o);
    cout << "average is:" << avg << endl;
}

int divide(int a, int b)
{
    if (a % b == 0)
        return 1;
    else
        return 0;
}

int divide(int a)
```

```
{
    int count = 0, i;

    for (i = 2; i <= a; i++)
    {
        if (a % i == 0)
            count++;
    }

    if (count == 1)
        return 1;
    else
        return 0;
}

float divide(float a, float b, float c)
{
    return ((a + b + c) / 3);
}
```

**OUTPUT:**



```
enter a and b where a is numerator and b is denominator :
4
2
divisible

enter a number to check number prime or not:
3
number is a prime

enter three numbers to find average:
4
6
7
average is:5.66667
```

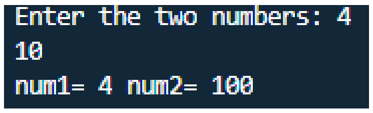
|  |  |
|---|---|
|  | **CONCLUSION:** In this practical we learn about function overloading. And it means same function name can be defined more than one time with different types of argument to perform same or different tasks is called function overloading. |

| No. | Aim of the Practical |
|---|---|
| 11. | **Write a function called tonLarge() that takes two integer arguments call by reference and then sets the larger of the two numbers to 100 using Return by reference. Write a main() program to exercise this function.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

int tonLarge(int &num1, int &num2);
int main()
{
    int num1, num2;

    cout << "Enter the two numbers: ";
    cin >> num1;
    cin >> num2;

    tonLarge(num1, num2);

    cout << "num1= " << num1 << "\tnum2= " << num2;
}

int tonLarge(int &num1, int &num2)
{
    if (num1 > num2)
    {
        return num1 = 100;
    }
    else
    {
        return num2 = 100;
    }
}
```

**OUTPUT:**



```
Enter the two numbers: 4
10
num1= 4 num2= 100
```

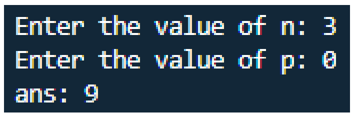
**CONCLUSION:** This way we can use reference variable to pass the values as well as we can use reference variable to update the values in function.

| No. | Aim of the Practical |
|---|---|
| 12. | **Write a function called power () that takes two arguments: a double value for n and an int for p, and returns the result as double value. Use default argument of 2 for p, so that if this argument is omitted, the number will be squared. Write a main () function that gets values from the user to test this function.** |

**PROGRAM CODE :**

```
#include <iostream>
#include <math.h>
using namespace std;

double power(double n, int p = 2)
{
    double ans;
    ans = pow(n, p);

    cout << "ans: " << ans;
}

int main()
{
    double n;
    int p;

    cout<<"Enter the value of n: ";
    cin>>n;

    cout<<"Enter the value of p: ";
    cin>>p;

    if(p==0)
    {
        power(n);
    }
    else
    {
        power(n,p);
    }
}
```

**OUTPUT:**



```
Enter the value of n: 3
Enter the value of p: 0
ans: 9
```

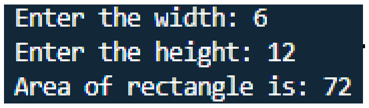
**CONCLUSION:** we can use the concept of default argument here and can give square value in answer even without passing argument with use of default argument.

| No. | Aim of the Practical |
|---|---|
| 13. | **Define a C++ Structure Rectangle with data member's width and height. It has get_values() member functions to get the data from user and area() member functions to print the area of rectangle. Also create a C++ Class for the above program. Define both functions inside the class. Member function defined inside the class behaves like an inline function and illustrate the difference between C++ Structure and C++ Class.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

struct Rectangle
{
    float width, height;
    void get_values()
    {
        cout << "Enter the width: ";
        cin >> width;
        cout << "Enter the height: ";
        cin >> height;
    }
    void area()
    {
        float area;
        area = width * height;
        cout << "Area of rectangle is: " << area;
    }
};
int main()
{
    Rectangle r;
    r.get_values();
    r.area();
}
```

**OUTPUT:**



```
Enter the width: 6
Enter the height: 12
Area of rectangle is: 72
```

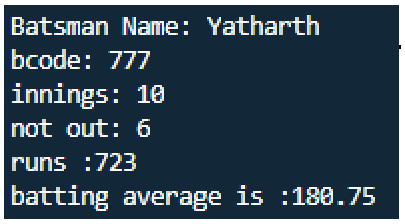
**CONCLUSION:** In this practical we learn the main difference between class and structure is that members of class are private and members of structure are public. Also we can define methods inside a class but not in structure.

| No. | Aim of the Practical |
|---|---|
| 14. | **Write a C++ program having class Batsman. It has private data members: batsman name, bcode (4 Digit Code Number), innings, not out, runs, batting average. Innings, not out and runs are in integer and batting average is in float. Define following function outside the class using scope resolution operator. 1) Public member function getdata() to read values of data members. 2) Public member function putdata() to display values of data members. 3) Private member function calcavg() which calculates the batting average of a batsman. Also make this outside function inline. Hint : batting average = runs/( innings – notout )**<br><br>**PROGRAM CODE :**<br><br>#include <iostream> |

```
using namespace std;

class batsman
{
private:
   string batsman_name;
   int bcode, innings, not_out, runs;
   float batting_average;

public:
   void getdata();
   void putdata();
   float calavg();
} d1;

void batsman::getdata()
{
   cout << "Enter Batsman Name: ";
   cin >> batsman_name;
   cout << "Enter bcode: ";
   cin >> bcode;
   cout << "Enter innings: ";
   cin >> innings;
   cout << "Enter not out: ";
   cin >> not_out;
   cout << "Enter runs: ";
   cin >> runs;
}
void batsman::putdata()
{

   cout << "Batsman Name: " << batsman_name << endl;
   cout << "bcode: " << bcode << endl;
   cout << "innings: " << innings << endl;
```

```
    cout << "not out: " << not_out << endl;
    cout << "runs :" << runs << endl;
    cout << "batting average is :" << d1.calavg() << endl;
}

float batsman::calavg()
{
    return (float(runs) / (innings - not_out));
}
main()
{
    d1.getdata();
    d1.putdata();
    d1.calavg();
}
```

**OUTPUT:**



```
Batsman Name: Yatharth
bcode: 777
innings: 10
not out: 6
runs :723
batting average is :180.75
```

**CONCLUSION:** In this Practical we learn  about private and public specifier we define the access privileges of variables and method.

| No. | Aim of the Practical |
|---|---|
| 15. | **Define class Currency having two integer data members rupee and paisa. A class has member functions enter() to get the data and show() to print the amount in 22.50 format. Define one member function sum() that adds two objects of the class and stores answer in the third object i.e. c3=c1.sum (c2). The second member function should add two objects of type currency passed** |

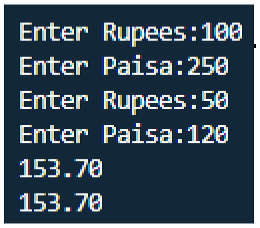
**as arguments such that it supports c3.add (c1, c2); where c1, c2 and c3 are objects of class Currency. Also Validate your answer if paisa >100. Write a main( ) program to test all the functions. Use concepts of Object as Function Arguments, function returning object and function overloading.**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

class Currency
{
   int rupee, paisa;

public:
   enter()
   {
      cout << "Enter Rupees:";
      cin >> rupee;
      cout << "Enter Paisa:";
      cin >> paisa;
   }
   show()
   {
      while (paisa >= 100)
      {
         rupee = rupee + 1;
         paisa = paisa - 100;
      }
      cout << rupee << "." << paisa << endl;
   }
   Currency sum(Currency c)
   {
      Currency t;
```

```
        t.rupee = rupee + c.rupee;
        t.paisa = paisa + c.paisa;
        return t;
    }
    Currency sum(Currency c, Currency d)
    {
        Currency t;
        t.rupee = c.rupee + d.rupee;
        t.paisa = c.paisa + d.paisa;
        return t;
    }
};
main()
{
    Currency c1, c2, c3;
    c1.enter();
    c2.enter();
    c3 = c1.sum(c2);
    c3.show();
    c3 = c3.sum(c1, c2);
    c3.show();
}
```

**OUTPUT:**



```
Enter Rupees:100
Enter Paisa:250
Enter Rupees:50
Enter Paisa:120
153.70
153.70
```

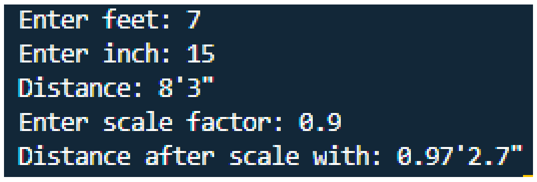
**CONCLUSION:** In this program we used concept of object as function arguments, function returning object and function overloading.

| No. | Aim of the Practical |
|---|---|
| 16. | **Define a class Dist with int feet and float inches. Define member function that displays distance in 1’-2.5” format. Also define member function scale ( ) function that takes object by reference and scale factor in float as an input argument. The function will scale the distance accordingly. For example, 20’-5.5” and Scale Factor is 0.5 then answer is 10’-2.75”** |

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
class dist
{
    int feet;
    float inch;

public:
    void display()
    {
        cout << "Enter feet: ";
        cin >> feet;
        cout << "Enter inch: ";
        cin >> inch;

        while (inch > 12)
        {
            feet = feet + 1;
            inch = inch - 12;
        }
        cout << "Distance: " << feet << "'" << inch << "\"" << endl;
    }
    void scale(dist &d)
    {
        float sf;
        cout << "Enter scale factor: ";
        cin >> sf;

        feet = feet * sf;
        inch = inch * sf;

        while (inch > 12)
        {
            feet = feet + 1;
            inch = inch - 12;
        }
        cout << "Distance after scale with is: " << sf << feet << "'" << inch << "\""
<< endl;
    }
};
int main()
{
    dist d;
    d.display();
    d.scale(d);
}
```

**OUTPUT:**



```
Enter feet: 7
Enter inch: 15
Distance: 8'3"
Enter scale factor: 0.9
Distance after scale with: 0.97'2.7"
```

**CONCLUSION:** In this Practical we learn  about how to scale the given distance into feet and inch scale.

| No. | Aim of the Practical |
|---|---|
| **17.** | **Create a Class Gate for students appearing in Gate (Graduate Aptitude test for Engineering) exam. There are three examination center Vadodara, Surat, and Ahmedabad where Gate exams are conducted. A class has data members: Registration number, Name of student, Examination center. Class also Contains static data member ECV_Cnt, ECS_Cnt and ECA_Cnt which counts the number of students in Vadodara, Surat and Ahmedabad exam center respectively. Class Contains two Member function getdata() which gets all information of students and counts total students in each exam center and pudata () which prints all information about the students. Class also contains one static member function getcount() which displays the total number of students in each examination center. Write a program for 5 students and display the total number of students in each examination center. Use static data member, static member function and Array of Objects.** |

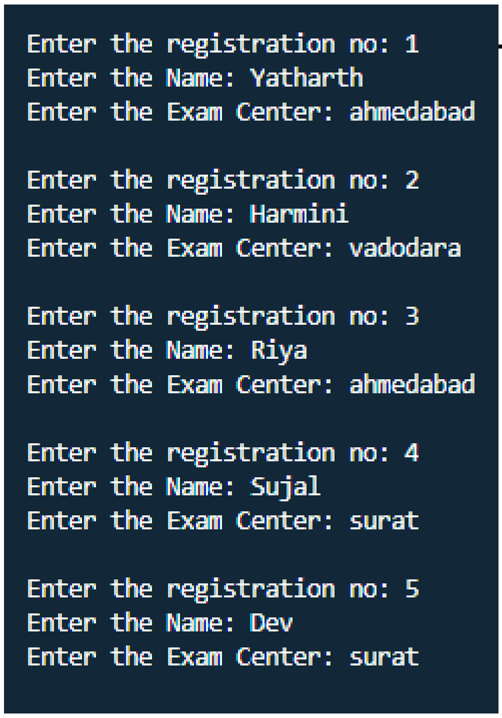
**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
class gate
{
public:
    string roll_no, name, exam_center;
    static int ECA, ECV, ECS;

    void getdata();
    void putdata();

    static void getcount()
    {
        cout << "Student appear in Ahmedabad center: " << ECA << endl;
        cout << "Student appear in Vadodara center: " << ECV << endl;
        cout << "Student appear Surat center: " << ECS << endl;
    }
```

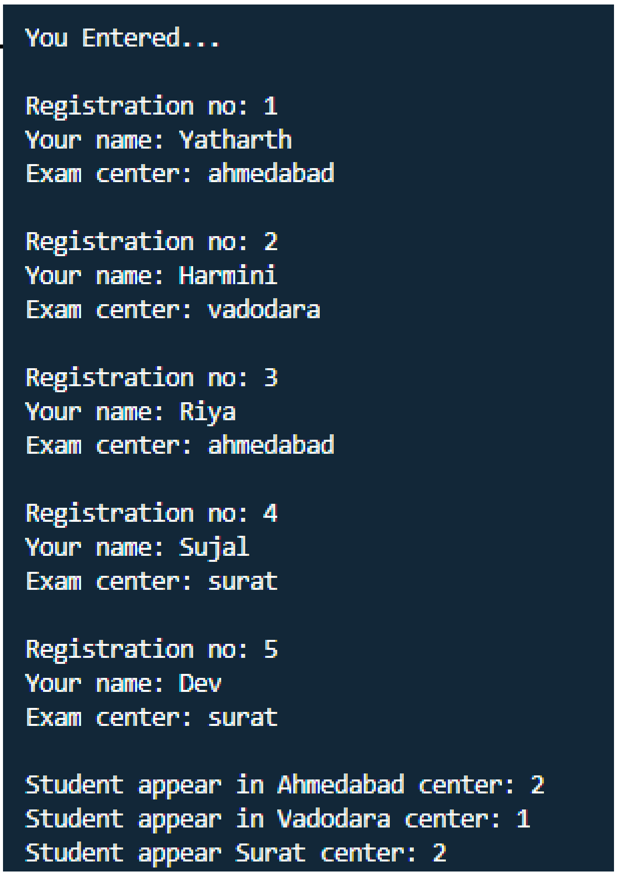
```
};
int gate::ECA = 0, gate::ECV = 0, gate::ECS = 0;

void gate::getdata()
{
    cout << endl
         << "Enter the registration no: ";
    cin >> roll_no;
    cout << "Enter the Name: ";
    cin >> name;
    cout << "Enter the Exam Center: ";
    cin >> exam_center;

    if (exam_center == "ahmedabad")
    {
        ECA++;
    }
    else if (exam_center == "vadodara")
    {
        ECV++;
    }
    else if (exam_center == "surat")
    {
        ECS++;
    }
}
void gate::putdata()
{
    cout << endl
         << "Registration no: " << roll_no;
    cout << endl
         << "Your name: " << name;
    cout << endl
         << "Exam center: " << exam_center;
```

```
}
int main()
{
   gate student[5];

   for (int i = 0; i < 5; i++)
   {
      student[i].getdata();
   }
   cout << endl << "You Entered...";

   for (int i = 0; i < 5; i++)
   {
      student[i].putdata();
   }
   gate::getcount();
}
```

**OUTPUT:**



```
Enter the registration no: 1
Enter the Name: Yatharth
Enter the Exam Center: ahmedabad

Enter the registration no: 2
Enter the Name: Harmini
Enter the Exam Center: vadodara

Enter the registration no: 3
Enter the Name: Riya
Enter the Exam Center: ahmedabad

Enter the registration no: 4
Enter the Name: Sujal
Enter the Exam Center: surat

Enter the registration no: 5
Enter the Name: Dev
Enter the Exam Center: surat
```



```
You Entered...

Registration no: 1
Your name: Yatharth
Exam center: ahmedabad

Registration no: 2
Your name: Harmini
Exam center: vadodara

Registration no: 3
Your name: Riya
Exam center: ahmedabad

Registration no: 4
Your name: Sujal
Exam center: surat

Registration no: 5
Your name: Dev
Exam center: surat

Student appear in Ahmedabad center: 2
Student appear in Vadodara center: 1
Student appear Surat center: 2
```

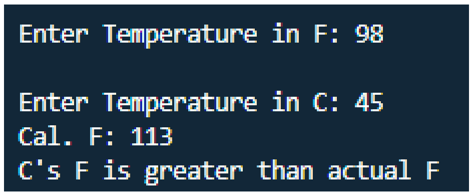
**CONCLUSION**: in this practical we learnt how to use static data member, static member function and array of objects.

| No. | Aim of the Practical |
|---|---|
| 18. | **Define a class *Fahrenheit* with float temp as data member. Define another class *Celsius* with float temperature as data member. Both classes have member functions to input and print data. Write a non-member function that receives objects of both the classes and declare which one is higher than another according to their values. Also define main() to test the function. Define all member functions outside the class. (Formula for converting Celsius to Fahrenheit is F = (9C/5) + 32). Use the concept of friend function.**<br><br>**PROGRAM CODE :**<br><br>#include <iostream> |

```
using namespace std;
class c;
class f
{
private:
    float temp;

public:
    void getdata();
    friend void compare(f, c);
} f1;

class c
{
private:
    float temperature;

public:
    void data();
    friend void compare(f, c);
} c1;

void f::getdata()
{
    cout << endl
         << "Enter Temperature in F: ";
    cin >> temp;
}
void c::data()
{
    cout << endl
         << "Enter Temperature in C: ";
    cin >> temperature;
}
```

```
void compare(f f1, c c1)
{
    float f;
    f = (9 * c1.temperature) / 5 + 32;
    cout << "Cal. F: " << f;

    if (f > f1.temp)
    {
        cout << endl
             << "C's F is greater than actual F";
    }
    else
    {
        cout << endl
             << "C's F is not greater than actual F";
    }
}
int main()
{
    f1.getdata();
    c1.data();
    compare(f1, c1);
}
```

**OUTPUT:**



```
Enter Temperature in F: 98

Enter Temperature in C: 45
Cal. F: 113
C's F is greater than actual F
```

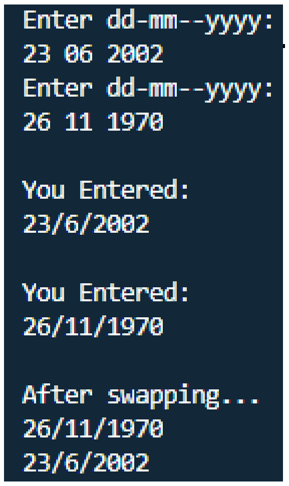
**CONCLUSION:** In this Practical we learnt how to use the concept of friend function.

| No. | Aim of the Practical |
|---|---|
| 19. | **Create a Class Date having data members: int dd, mm, yyyy. Class has one member function to input the dates and another member function which prints the dates. Write a main() function which takes two dates as input. Write a friend function swapdates() which takes two objects by reference of type Date and swaps both the dates. Use the concept of Friend function which takes objects by reference.** |

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
class date
{
private:
    int date_, month, year;

public:
    void getdata()
    {
        cout << "Enter dd-mm--yyyy: " << endl;
        cin >> date_ >> month >> year;
    }
    void putdata()
    {
        cout << "You Entered: " << endl;
        cout << date_ << "/" << month << "/" << year;
    }

    friend void swap(date &d1, date &d2)
    {
        date temp;
        temp=d1;
        d1=d2;
        d2=temp;

        cout<<"After swapping..."<<endl;

cout<<"d1.date_"<<"/"<<d1.month<<"/"<<d1.year<<endl<<d2.date_<<"/"<<d2.
month<<"/"<<d2.year;
    }
};
int main()
{
    date d1,d2;
    d1.getdata();
    d2.getdata();
    d1.putdata();
    d2.putdata();
    swap(d1,d2);

}
```

**OUTPUT:**



```
Enter dd-mm--yyyy:
23 06 2002
Enter dd-mm--yyyy:
26 11 1970

You Entered:
23/6/2002

You Entered:
26/11/1970

After swapping...
26/11/1970
23/6/2002
```

**CONCLUSION:** In this practical we learnt how to use the concept of friend function which takes objects as reference. .

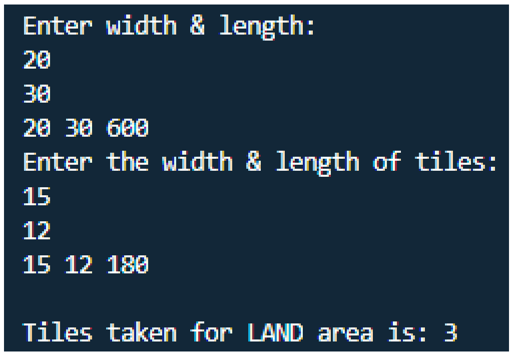
| No. | Aim of the Practical |
|---|---|
| 20. | **Create a class LAND having data members: length, width, area1. Write member functions to read and display the data of land. Also, calculates the area of the land. Create another class TILES having data members: l, w, area2. Write a member function to get the data of tile. Calculate the area of one tile. Class TILE has a member function named number_of_tiles() which is a friend of class LAND and takes the object of class LAND by reference which calculates the number of tiles which can be put over the land area. Write the main function to test all the functions. Use the concept of member function of one class can be a friend function of another class.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class land
{
public:
   int w1, l1, a1;
   void getdata()
   {
      cout << endl
          << "Enter width & length: " << endl;
      cin >> w1 >> l1;
      a1 = w1 * l1;
   }
   void putdata()
   {
      cout << w1 << " " << l1 << " " << a1;
   }
   friend void number_of_titles();
} l;
class tiles
{
```

```
public:
    int w2, l2, a2;

    void getdata()
    {
        cout << endl
             << "Enter the width & length of tiles: " << endl;
        cin >> w2 >> l2;

        a2 = w2 * l2;
    }
    void putdata()
    {
        cout << w2 << " " << l2 << " " << a2 << endl;
    }
    friend void number_of_tiles();
} t;
void number_of_tiles(land &l, tiles &t)
{
    int n;
    n = (l.a1) / (t.a2);
    cout << endl
         << "Tiles taken for LAND area is: " << n;
}
int main()
{
    l.getdata();
    l.putdata();
    t.getdata();
    t.putdata();

    number_of_tiles(l, t);
}
```

**<u>OUTPUT:</u>**



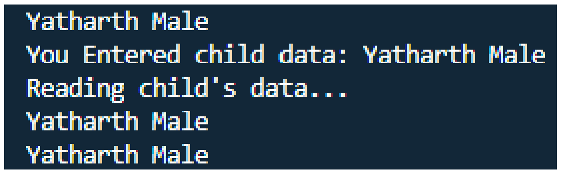
```
Enter width & length:
20
30
20 30 600
Enter the width & length of tiles:
15
12
15 12 180

Tiles taken for LAND area is: 3
```

**<u>CONCLUSION</u>:** In this practical we learnt how to use member function of one class can be a friend function of another class in c++ .

| No. | Aim of the Practical |
|---|---|
| 21. | **Create a class Child having data members: name of the child and gender and a member function to get and print child data. Create another class Parent which is a friend class of child class. Class Parent have member function ReadChildData() which takes child's object by reference as input argument and Reads the childs data and DisplayChildData() which takes childs object as argument and displays childs data. Use the concepts of Friend Class.** |

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
class child
{
private:
    string name, gender;

public:
    void getdata()
    {
        cin >> name >> gender;
    }
    void putdata()
    {
        cout << "You Entered child data: " << name << gender;
    }
    friend class parent;
} c1;

class parent
{
public:
    void ReadChildData(child &c1)
    {
        cout << endl
             << "Reading child's data..." << endl;
        cout << c1.name << " " << c1.gender << endl;
    }
    void DisplayChildData(child *c1)
    {
        cout << c1->name << " " << c1->gender;
    }
} p1;

int main()
{
    child c1;
    c1.getdata();
    c1.putdata();
    p1.ReadChildData(c1);
    p1.DisplayChildData(&c1);
    return 0;
```

|  |  |
|---|---|
|  | } <br><br> **OUTPUT:** <br><br>  <br> `Yatharth Male` <br> `You Entered child data: Yatharth Male` <br> `Reading child's data...` <br> `Yatharth Male` <br> `Yatharth Male` <br><br> **CONCLUSION:** In this practical we learnt how to access the data of a friend child class by parent using the reference of child class object. |

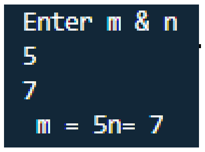
| No. | Aim of the Practical |
|---|---|
| 22. | **Check the following C++ code and find if there is any error in code, give justification for the error, correct the code and write the output:** <br><br> **1. Example of const member functions** <br><br> **#include &lt;iostream&gt; //With Error** <br> **using namespace std;** <br> **class sample** <br> **{** <br> **int m, n;** <br><br> **public:** <br> **void getdata();** <br> **void putdata() const;** <br> **};** <br> **void sample::getdata()** <br> **{** <br> **cout &lt;&lt; "Enter m & n";** <br> **cin &gt;&gt; m &gt;&gt; n;** |

```
}
void sample::putdata() const
{
   m = 12;
   n = 34;
   cout << " m = " << m << "n= " << n;
}
int main()
{
   sample s1;
   s1.getdata();
   s1.putdata();
   return 0;
}
```

**PROGRAM CODE : (Without Error)**

```
#include <iostream>
using namespace std;
class sample
{
   int m, n;

public:
   void getdata();
   void putdata() const;
};
void sample::getdata()
{
   cout << "Enter m & n";
   cin >> m >> n;
}
void sample::putdata() const
{
    // m = 12;
    // n = 34;
    cout << " m = " << m << "n= " << n;
}
int main()
{
    sample s1;
    s1.getdata();
    s1.putdata();
    return 0;
}
```

**OUTPUT:**



```
Enter m & n
5
7
 m = 5n= 7
```

**CONCLUSION:** Error is in this code is m & n are constants & we cannot re-assign the value to m & n. To solve this error we should comment lines which is assigning values of m & n.

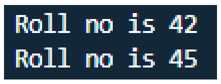
**2. (a) Pointer to data members**

```
#include <iostream>          //With Error
using namespace std;
class student
{
public:
    int roll_no;
};
int main()
{ // declaring pointer to data member
    int student ::*p1 = &student::roll_no;
    student s;
    student *optr = &s;
    s->*p1 = 42;
    cout << "Roll no is " << s->*p1 << endl;
    optr.*p1 = 45;
    cout << "Roll no is " << optr.*p1 << endl;
    return 0;
}
```

**PROGRAM CODE : (Without Error)**

```
#include <iostream>
using namespace std;
class student
{
public:
    int roll_no;
};
int main()
{ // declaring pointer to data member
    int student ::*p1 = &student::roll_no;
    student s;
    student *optr = &s;
    s.*p1 = 42;
    cout << "Roll no is " << s.*p1 << endl;
    optr->*p1 = 45;
    cout << "Roll no is " << optr->*p1 << endl;
    return 0;
}
```

**Output:**



```
Roll no is 42
Roll no is 45
```

**CONCLUSION:** Error is in this code is we cannot use arrow operator to access objects & also cannot use dot operator to access pointers.

**2. (b)Pointer to member functions**

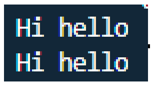
```
#include <iostream>                //With Error
class employee
{
public:
   void hello()
   {
      cout << "Hi hello" << endl;
   }
};
int main()
{ // declaring pointer to member function hello
   void (employee ::*fp)() = &employee::hello;
   employee e;
   employee *optr = &e;
   (e->*fp)();
   (optr.*fp)();
   return 0;
}
```

**PROGRAM CODE : (Without Error)**

```
#include <iostream>
using namespace std;
class employee
{
public:
   void hello()
```

```
    {
        cout << "Hi hello" << endl;
    }
};
int main()
{ // declaring pointer to member function hello
    void (employee ::*fp)() = &employee::hello;
    employee e;
    employee *optr = &e;
    (e.*fp)();
    (optr->*fp)();
    return 0;
}
```

**Output:**



```
Hi hello
Hi hello
```

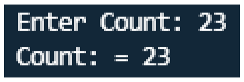
**CONCLUSION:** Error is in this code is using namespace std was missing & arrow operator used to access object.

**3. Example of Local Classes**

```
#include <iostream>          //With Error
using namespace std;
void testlocalclass()
{
    class Test
    {
        static int cnt;

    public:
        void set()
        {
            cout << "Enter Count: ";
            cin >> cnt;
        }
        void get();
    };
    void Test::get()
    {
        cout << "Count: = " << cnt;
    }
    Test t;
    t.set();
    t.get();
}
int main()
{
    testlocalclass();
    return 0;
}
```

**PROGRAM CODE : (Without Error)**

```
#include <iostream>
using namespace std;
void testlocalclass()
{
    class Test
    {
        int cnt; //local class should not have static data member
    public:
        void set()
        {
            cout << "Enter Count: ";
            cin >> cnt;
```

| | |
|---|---|
| | <pre>        }<br>        void get()<br>        {<br>            cout << "Count: = " << cnt;<br>        }<br>    };<br>    Test t;<br>    t.set();<br>    t.get();<br>};<br>int main()<br>{<br>    testlocalclass();<br>    return 0;<br>}</pre> **<u>Output:</u>**  <pre>Enter Count: 23<br>Count: = 23</pre> **<u>CONCLUSION</u>:** Error in this code is local classes does not have static data members. |

| No. | Aim of the Practical |
|---|---|
| 23. | **Write a C++ program having class time with data members: hr, min and sec. Define following member functions.<br>1) getdata() to enter hour, minute and second values<br>2) putdata() to print the time in the format 11:59:59<br>3) default constructor<br>4) parameterized constructor<br>5) copy constructor** |

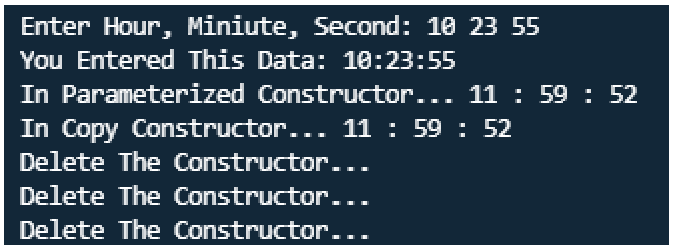
**6) Destructor.**
**Use 52 as default value for sec in parameterized constructor.**
**Use the concepts of default constructor, parameterized constructor, Copy constructor, constructor with default arguments and destructor.**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class time
{
   int hr, min, sec;

public:
   void getdata()
   {
      cout << "Enter Hour, Miniute, Second: ";
      cin >> hr >> min >> sec;
   }
   void putdata()
   {
      cout << "You Entered This Data: ";
      cout << hr << ":" << min << ":" << sec << endl;
   }
   time()
   {
      cout << "This is Default Constructor..." << endl;
   }
   time(int h, int m, int s = 52)
   {
      hr = h;
      min = m;
      sec = s;
      cout << "In Parameterized Constructor..."
```

```
            << " " << hr << " : " << min << " : " << sec << endl;
    }
    time(const time &t2)
    {
        hr = t2.hr;
        min = t2.min;
        sec = t2.sec;
        cout << "In Copy Constructor..."
            << " " << hr << " : " << min << " : " << sec << endl;
    }
    ~time()
    {
        cout << "Delete The Constructor..." << endl;
    }
};
int main()
{
    time t;
    t.getdata();
    t.putdata();
    time t1(11, 59);
    time t2 = t1;
}
```

**OUTPUT:**



```
Enter Hour, Miniute, Second: 10 23 55
You Entered This Data: 10:23:55
In Parameterized Constructor... 11 : 59 : 52
In Copy Constructor... 11 : 59 : 52
Delete The Constructor...
Delete The Constructor...
Delete The Constructor...
```

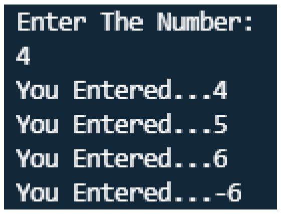
| | |
|---|---|
| | **CONCLUSION:** In this practical we learnt about the concepts of default constructor, parameterized constructor, Copy constructor, constructor with default arguments and destructor. |

| No. | Aim of the Practical |
|---|---|
| 24. | **Create a class Number having int num as member. The class has input and output functions. Overload unary operator (++) such that it supports N1=N2++ and N3=++N1 and Overload unary (-) such that it supports N3 = -N3. Also define default, parameterized and copy constructor for the class. Also explain use of nameless object in operator overloading.** <br> **Use the concept of Overloading Unary Operators.** <br><br> **PROGRAM CODE :** |

```
#include <iostream>
using namespace std;
class number
{
   int num;

public:
   void getdata()
   {
      cout << "Enter The Number: " << endl;
      cin >> num;
   }
   void putdata()
   {
      cout << "You Entered..." << num << endl;
   }
   void operator++(int)
```

```
    {
        num++;
    }
    void operator++()
    {
        ++num;
    }
    void operator-()
    {
        num = -num;
    }
};
int main()
{
    number n2;
    n2.getdata();
    n2.putdata();
    n2.operator++();
    n2.putdata();
    n2.operator++();
    n2.putdata();
    n2.operator-();
    n2.putdata();
}
```

**OUTPUT:**



```
Enter The Number:
4
You Entered...4
You Entered...5
You Entered...6
You Entered...-6
```

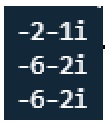
**CONCLUSION:** In this practical learnt about the concept of Overloading Unary Operators.

| No. | Aim of the Practical |
|---|---|
| 25. | **Create a class complex having data members int real , img and member function to print data. Overload Unary operator (-) using friend function such that it supports – C1 where C1 is the object of class complex. Also define default, parameterized and copy constructor for the class.**<br>**Use the concept of Overloading Unary Operators with friend function.** |

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;
class complex
{
    int real, img;

public:
    void putdata()
    {
        cout << real << img << "i" << endl;
    }
    complex()
    {
        real = 2;
        img = 1;
    }
    complex(int r, int i)
    {
        real = r;
        img = i;
    }
    complex(const complex &c)
    {
        real = c.real;
        img = c.img;
    }
    friend complex operator-(complex &c3);
};
complex operator-(complex &c3)
{
    complex c;
    c.real = -c3.real;
    c.img = -c3.img;
    return c;
}
int main()
{
    complex c1, c2(6, 2), c3(c2), c4;
    c4 = operator-(c1);
    c4.putdata();
    c4 = operator-(c2);
    c4.putdata();
    c4 = operator-(c3);
    c4.putdata();
}
```

**OUTPUT:**



```
-2-1i
-6-2i
-6-2i
```

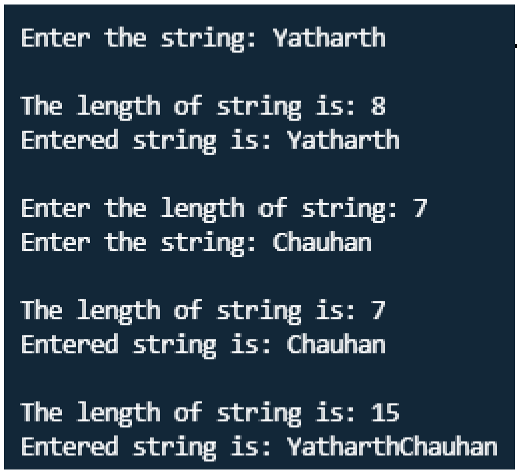
**CONCLUSION:** In this practical we learnt about the concept of Overloading Unary Operators with friend function.

| No. | Aim of the Practical |
|---|---|

**26.** **Create a class String having character array. Class includes constructor and required member functions to get and display the object. Overload the operators +(s3=s1+s2), ==(s1<s2), +=(s1+=s2) for the class.**
**Use the concept of Overloading Binary Operators**

**PROGRAM CODE :**

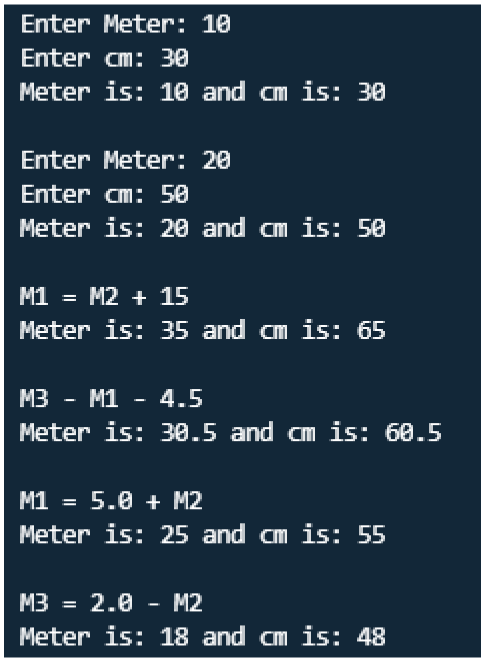
```
#include <iostream>
using namespace std;

class String
{
public:
    char a[50];
    int len;
    void getdata()
    {
        cout << "Enter the length of string:  " << endl;
        cin >> len;
        cout << "Enter the string: " << endl;
        for (int i = 0; i < len; i++)
        {
            cin >> a[i];
        }
    }
    void putdata()
    {
        cout << "The length of string is: " << len << endl;

        cout << "Entered string is: " << endl;
        for (int i = 0; i < len; i++)
        {
            cout << a[i];
        }
        cout << endl;
    }
    String operator+(String s1)
    {
        String temp;
        int i, j, length = 0;
        for (i = 0; i < len; i++)
        {
            temp.a[i] = a[i];
            length++;
        }
        j = i;
        for (i = 0; i < s1.len; i++)
        {
            temp.a[j] = s1.a[i];
            j++;
            length++;
        }
        temp.len = length;
        return temp;
    }
};

int main()
{
    String s1;
    s1.getdata();
    s1.putdata();
    String s2;
    s2.getdata();
    s2.putdata();
    String s3;
    s3 = s1 + s2;
    s3.putdata();
```

| | |
|---|---|
| | return 0;<br>}<br>**OUTPUT:**<br><br>`Enter the string: Yatharth`<br><br>`The length of string is: 8`<br>`Entered string is: Yatharth`<br><br>`Enter the length of string: 7`<br>`Enter the string: Chauhan`<br><br>`The length of string is: 7`<br>`Entered string is: Chauhan`<br><br>`The length of string is: 15`<br>`Entered string is: YatharthChauhan`<br><br>**CONCLUSION:** In this practical we learnt how to use overloading binary operators. |

| No. | Aim of the Practical |
|---|---|
| 27. | **Create a class Measure having members: meter and cm. The class has get( ) and put( ) functions. Overload operator + and – such that they support M1=M2+15 and M3=M1 – 4.5. Also overload + and – such that they support M1=5.0+M2 and M3=2.0 – M4. Write a main( ) to test the class.**<br>**Use the concept of Overloading Binary Operators with friend function.**<br><br>**PROGRAM CODE :**<br><br>#include &lt;iostream&gt;<br>using namespace std;<br>class Measure<br>{<br>private:<br>    float meter, cm; |

```
public:
   void getdata()
   {
      cout << "Enter Meter: ";
      cin >> meter;

      cout << "Enter cm: ";
      cin >> cm;
   }
   void putdata()
   {
      cout << "Meter is: " << meter << " and cm is: " << cm << endl;
      cout << endl;
   }
   Measure operator+(float i)
   {
      Measure m;
      m.meter = meter + i;
      m.cm = cm + i;
      return m;
   }
   Measure operator-(float i)
   {
      Measure m;
      m.meter = meter - i;
      m.cm = cm - i;
      return m;
   }
   Measure friend operator+(float i, Measure m1);
   Measure friend operator-(float i, Measure m1);
};
Measure operator+(float i, Measure m1)
{
    Measure m;
    m.meter = m1.meter + i;
    m.cm = m1.cm + i;
    return m;
}
Measure operator-(float i, Measure m1)
{
    Measure m;
    m.meter = m1.meter - i;
    m.cm = m1.cm - i;
    return m;
}
int main()
{
    Measure m1, m2, m3;

    m1.getdata();
    m1.putdata();

    cout << endl;

    m2.getdata();
    m2.putdata();

    cout << "M1 = M2 + 15" << endl;
    m1 = m2 + 15;
    m1.putdata();

    cout << "M3 - M1 - 4.5" << endl;
    m3 = m1 - 4.5;
    m3.putdata();

    cout << "M1 = 5.0 + M2" << endl;
    m1 = 5.0 + m2;
    m1.putdata();

    cout << "M3 = 2.0 - M2" << endl;
    m3 = 2.0 - m2;
    m3.putdata();

    return 0;
}
```

**OUTPUT:**



```
Enter Meter: 10
Enter cm: 30
Meter is: 10 and cm is: 30

Enter Meter: 20
Enter cm: 50
Meter is: 20 and cm is: 50

M1 = M2 + 15
Meter is: 35 and cm is: 65

M3 - M1 - 4.5
Meter is: 30.5 and cm is: 60.5

M1 = 5.0 + M2
Meter is: 25 and cm is: 55

M3 = 2.0 - M2
Meter is: 18 and cm is: 48
```

**CONCLUSION:** In this practical we learnt how to use overloading binary operators with friend function.

| No. | Aim of the Practical |
|---|---|
| 28. | **Create a class Celsius with float. Define appropriate member functions such that it support the statements: C1=30.5F; float temperature; temperature=C2;** |

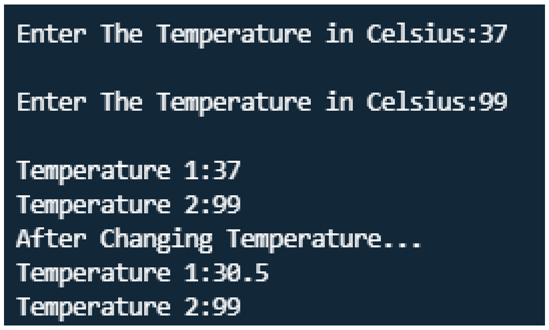
**Use the concept of Type conversion from basic type to class type and class type to basic type.**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class celsius
{
   float t;

public:
   void get()
   {
      cout << "\nEnter The Temperature in Celsius:";
      cin >> t;
   }
   void put(int x)
   {
      cout << "\nTemperature " << x << ":" << t;
   }
   float operator=(float a)
   {
      t = a;
      return t;
   }
   operator float()
   {
      return t;
   };
};
int main()
{
   celsius c1, c2;
```

```
    float temperature;
    c1.get();
    c2.get();
    c1.put(1);
    c2.put(2);
    temperature = c2;
    c1 = 30.5F;
    cout << "\nAfter Changing Temperature...";
    c1.put(1);
    cout << "\nTemperature 2:" << temperature;
}
```

**OUTPUT:**



```
Enter The Temperature in Celsius:37

Enter The Temperature in Celsius:99

Temperature 1:37
Temperature 2:99
After Changing Temperature...
Temperature 1:30.5
Temperature 2:99
```

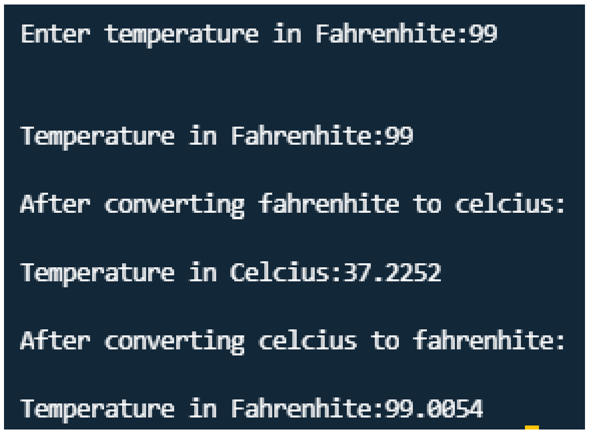
**CONCLUSION:** In this practical we learnt how to use type conversion from basic type to class type and class type to basic type.

| No. | Aim of the Practical |
|---|---|
| 29. | **Create classes Celsius and Fahrenheit with float. Define appropriate member functions such that they support the statements in main( ): Celsius C1, C2=5.0; Fahrenheit F1, F2; F1=C2; C1=F2;Use the concepts of Type conversion from class type to class type. Write this Program in two ways. Define appropriate member function in class Celsius. Define appropriate member function in class Fahrenheit.** |

**PROGRAM CODE :**

```
#include <iostream>
#include <conio.h>
using namespace std;
class fahrenhite;
class celcius
{
protected:
    float c;

public:
    celcius()
    {
    }
    void getdata()
    {
        cout << "enter temperature in celcius:";
        cin >> c;
    }
    void display()
    {
        cout << "\n\nTemperature in Celcius:" << c;
    }
    float setc()
    {
        return c;
    }

    celcius(fahrenhite f1);
};
class fahrenhite
{
protected:
    float f;

public:
    fahrenhite()
    {
    }
    void get_data()
    {
        cout << "\n\nEnter temperature in Fahrenhite:";
        cin >> f;
    }
    void display()
    {
        cout << "\n\nTemperature in Fahrenhite:" << f;
    }
    float setf()
    {
        return f;
    }
    fahrenhite(celcius c1)
    {
        f = (1.8 * c1.setc()) + 32;
    }
};
celcius::celcius(fahrenhite f1)
{
    c = ((f1.setf()) - 32) * (0.5556);
}
int main()
{
    celcius c1;
    fahrenhite f1;
    c1.getdata();
    c1.display();
    f1.get_data();
    f1.display();
    c1 = f1;
    cout << "\n\nAfter converting fahrenhite to celcius:";
    c1.display();
    f1 = c1;
    cout << "\n\nAfter converting celcius to fahrenhite:";
    f1.display();
}
```

**OUTPUT:**



```
Enter temperature in Fahrenhite:99


Temperature in Fahrenhite:99

After converting fahrenhite to celcius:

Temperature in Celcius:37.2252

After converting celcius to fahrenhite:

Temperature in Fahrenhite:99.0054
```

**CONCLUSION:** In this practical we learnt how to use type conversion from class type to class type.

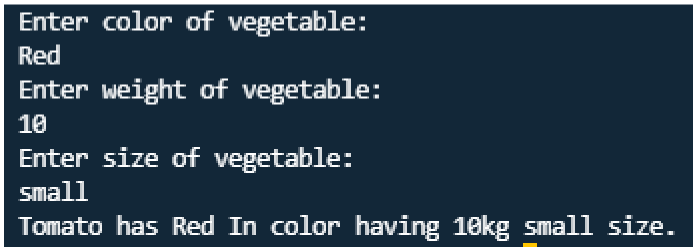
| **No.** | **Aim of the Practical** |
|---|---|
| **30.** | **Define a Base Class Vegetable having data member Color and member function getdata() which takes color as an input and putdata() which print the color as an output. Vegetable Class has one subclass named Tomato having data members weight and size and member function gtdata() which takes weight and size as an input and ptdata() which prints weight and size as output. Write a C++ Program which inherits the data of Vegetable class in Tomato class using Single Inheritance.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class vegetable
{
   string color;

public:
   ;
   void getdata()
   {
      cout << "Enter color of vegetable: " << endl;
      cin >> color;
   }
   void putdata()
   {
      cout << "Tomato has " << color << " In color";
   }
};
class tomato : public vegetable
{
   int wt;
   string size_;

public:
   void gtdata()
   {
      cout << "Enter weight of vegetable: " << endl;
      cin >> wt;
      cout << "Enter size of vegetable: " << endl;
      cin >> size_;
   }
```

```
    void ptdata()
    {
        cout << "having"
             << " " << wt << "kg ";
        cout << size_ << " size.";
    }
};
int main()
{
    tomato t;
    t.getdata();
    t.gtdata();
    t.putdata();
    cout << " ";
    t.ptdata();
    cout << endl;
    return 0;
}
```

**OUTPUT:**



```
Enter color of vegetable:
Red
Enter weight of vegetable:
10
Enter size of vegetable:
small
Tomato has Red In color having 10kg small size.
```

**CONCLUSION:** In this practical we learnt how to use single inheritance.

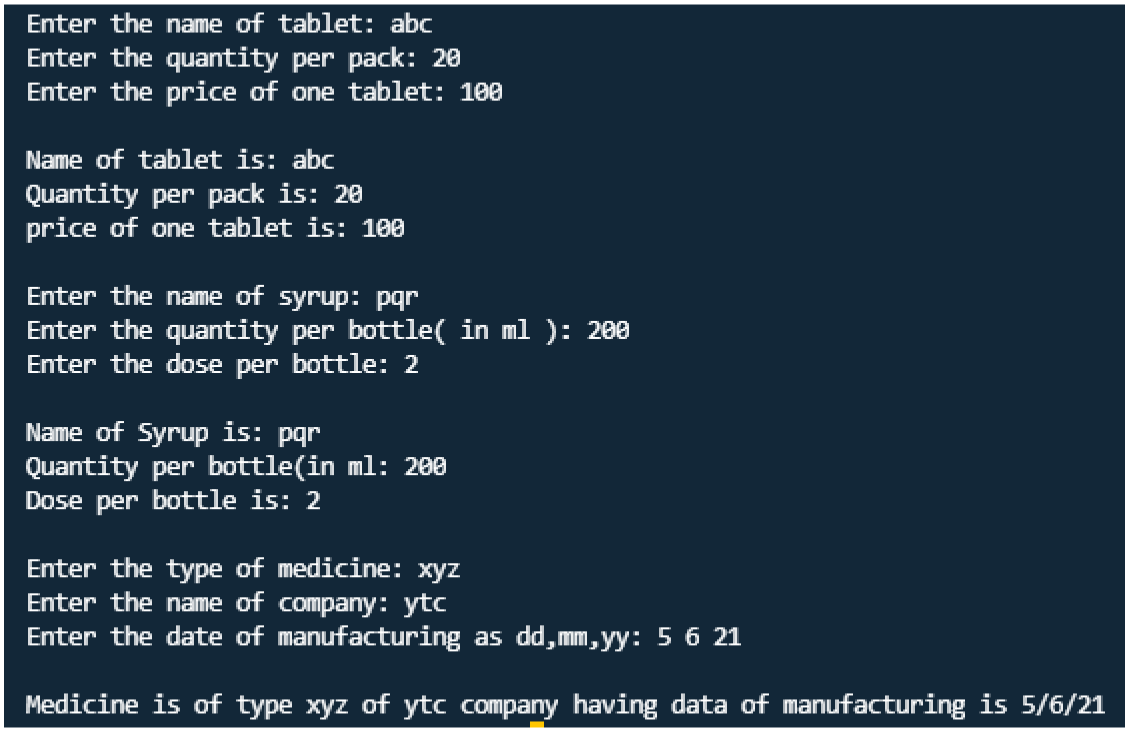
| No. | Aim of the Practical |
|---|---|

**31.** **Write a program to create a class Medicine which stores type of medicine, name of company, date of manufacturing. Class Tablet is inherited from Medicine. Tablet class has name of tablet, quantity per pack, price of one tablet as members. Class Syrup is also inherited from Medicine and it has quantity per bottle, dosage unit as members. Both the classes contain necessary member functions for input and output data. Write a main( ) that enter data for tablet and syrup, also display the data. Use the concepts of Hierarchical Inheritance.**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

class Medicine
{
    string typeOfmedicine;
    string numberOfcompany;
    int dd, mm, yy;

public:
    void getdata()
    {
        cout << "Enter the type of medicine: ";
        cin >> typeOfmedicine;
        cout << "Enter the name of company: ";
        cin >> numberOfcompany;
        cout << "Enter the date of manufacturing as dd,mm,yy: ";
        cin >> dd >> mm >> yy;
        cout << endl;
    }
    void display()
    {
        cout << "Medicine is of type " << typeOfmedicine << " of " << numberOfcompany << " company having data of manufacturing is " << dd << "/" << mm << "/" << yy;
    }
};
class Tablet : public Medicine
{
    string nameOftablet;
    int quantity;
    float price;

public:
    void setdata()
    {
        cout << "Enter the name of tablet: ";
        cin >> nameOftablet;
        cout << "Enter the quantity per pack: ";
        cin >> quantity;
        cout << "Enter the price of one tablet: ";
        cin >> price;

        cout << endl;
    }
    void display()
    {
        cout << "Name of tablet is: " << nameOftablet << endl;
        cout << "Quantity per pack is: " << quantity << endl;
        cout << "price of one tablet is: " << price << endl;
    }
};
class Syrup : public Medicine
{
    string nameOfsyrup;
    int qpb;
    int dpb;

public:
    void gtdata()
    {
        cout << "Enter the name of syrup: ";
        cin >> nameOfsyrup;
        cout << "Enter the quantity per bottle( in ml ): ";
        cin >> qpb;
        cout << "Enter the dose per bottle: ";
        cin >> dpb;

        cout << endl;
    }
    void displaydata()
    {
        cout << "Name of Syrup is: " << nameOfsyrup << endl;
        cout << "Quantity per bottle(in ml: " << qpb << endl;
        cout << "Dose per bottle is: " << dpb << endl;

        cout << endl;
    }
};

int main()
{
    Syrup s;
    Tablet t;

    t.setdata();
    t.display();

    cout << endl;
    s.gtdata();
    s.displaydata();
    s.getdata();
    s.display();

    return 0;
}
```

**OUTPUT:**



```
Enter the name of tablet: abc
Enter the quantity per pack: 20
Enter the price of one tablet: 100

Name of tablet is: abc
Quantity per pack is: 20
price of one tablet is: 100

Enter the name of syrup: pqr
Enter the quantity per bottle( in ml ): 200
Enter the dose per bottle: 2

Name of Syrup is: pqr
Quantity per bottle(in ml: 200
Dose per bottle is: 2

Enter the type of medicine: xyz
Enter the name of company: ytc
Enter the date of manufacturing as dd,mm,yy: 5 6 21

Medicine is of type xyz of ytc company having data of manufacturing is 5/6/21
```

**CONCLUSION:** In this practical we learnt how to use hierarchical inheritance

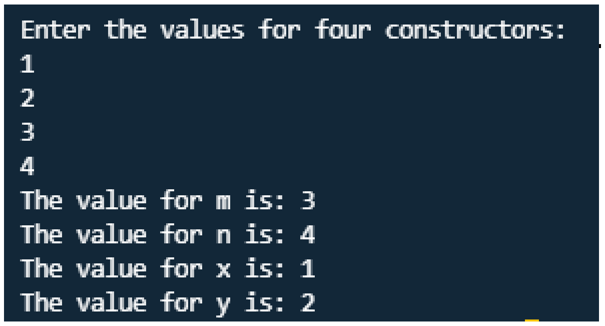
| No. | Aim of the Practical |
|---|---|
| 32. | **Create a Class alpha having data member: int x and one argument constructor which initializes the value of x. It also has member function which displays the value of x. Create another class beta which contains data member: float y and one argument constructor which initializes the value of y. It also has member** |

**function which displays the value of y. Create a Class Gamma which publicly inherits from class alpha and class beta and has two data members: int m, n and a constructor which passes argument to the base class constructor as well as initializes its own data members. Class Gamma also has member function to print the values of m and n. Write main function which creates object of class Gamma which passes values of base class constructor as well as derived class constructor. Use the concept of Multiple Inheritance and Constructor in Derived Class.**

**<u>PROGRAM CODE :</u>**

```
#include <iostream>
using namespace std;

class alpha
{
protected:
    int x;

public:
    alpha(int a)
    {
        x = a;
    }
};
class beta
{
protected:
    float y;

public:
    beta(int b)
    {
        y = b;
    }
};
class gamma : public alpha, public beta
{
    int m, n;

public:
    gamma(int a, int b, int c, int d) : alpha(a), beta(b)
    {
        m = c;
        n = d;
    }
    void putdata()
    {
        cout << "The value for m is: " << m << endl;
        cout << "The value for n is: " << n << endl;
        cout << "The value for x is: " << x << endl;
        cout << "The value for y is: " << y << endl;
    }
};
main()
{
    int a, b, c, d;

    cout << endl
         << "Enter the values for four constructors: " << endl;
    cin >> a >> b >> c >> d;
    gamma q1(a, b, c, d);
    q1.putdata();
}
```

**OUTPUT:**

| | |
|---|---|
| |  |

```
Enter the values for four constructors:
1
2
3
4
The value for m is: 3
The value for n is: 4
The value for x is: 1
The value for y is: 2
```

**CONCLUSION:** In this practical we learnt how to use the concept of multiple inheritance and constructor in derived class.

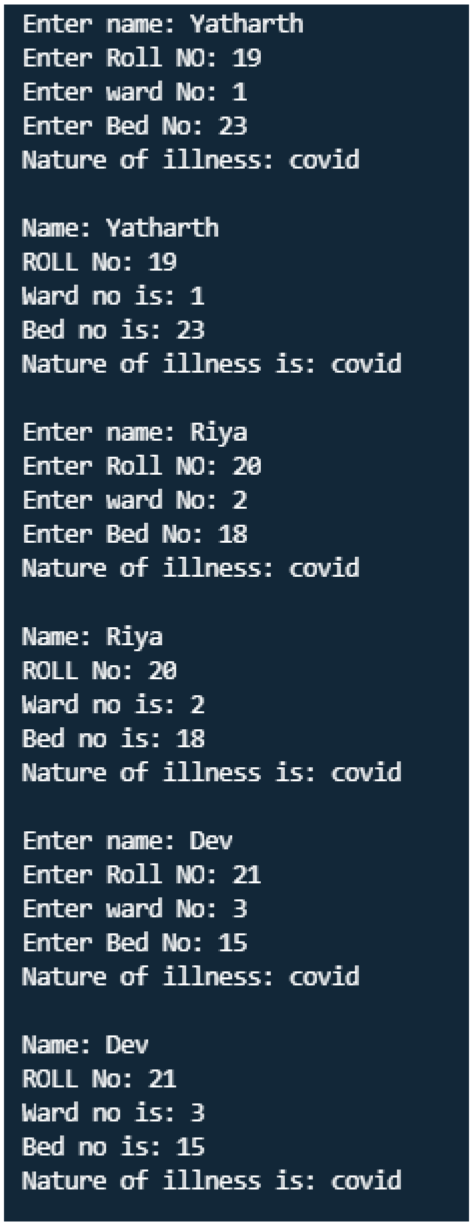
| No. | Aim of the Practical |
|---|---|
| 33. | **Define a class Hospital having rollno and name as data members and member function to get and print data. Derive a class Ward from class Hospital having data members: ward number and member function to get and print data. Derive another class Room from Hospital having data member bed number and nature of illness and member function to get and print data. Derive class Patient from Class Ward and Class Room. In main () declare 5 object of Class Patient and get and display all the information. Use the concept of Virtual Base Class and Hybrid Inheritance.** |

**PROGRAM CODE :**

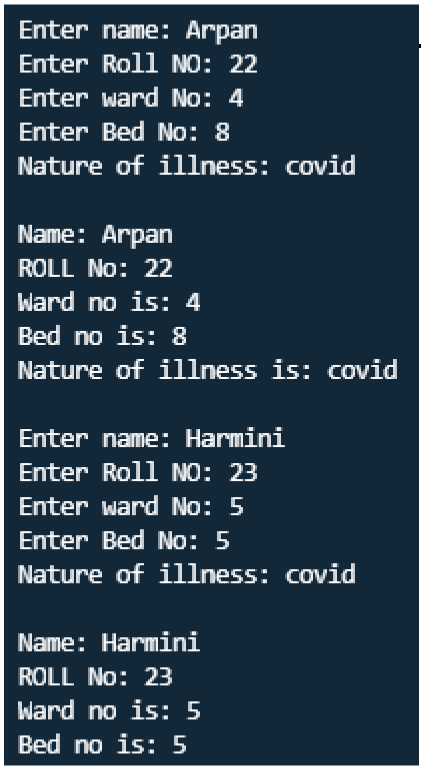
```
#include <iostream>
using namespace std;

class Hospital
{
    string rollNo, name;

public:
    void getdata()
    {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter Roll NO: ";
        cin >> rollNo;
    }
    void printdata()
    {
        cout << "Name: " << name << endl;
        cout << "ROLL No: " << rollNo << endl;
    }
};
class Ward : public virtual Hospital
{
    string wardNo;

public:
    void gdata()
    {
        cout << "Enter ward No: ";
        cin >> wardNo;
    }
    void pdata()
    {
        cout << "Ward no is: " << wardNo << endl;
    }
};

class Room : public virtual Hospital
{
    string bedNumber, natureOfillness;

public:
    void gtdata()
    {
        cout << "Enter Bed No: ";
        cin >> bedNumber;
        cout << "Nature of illness: ";
        cin >> natureOfillness;
    }
    void ptdata()
    {
        cout << "Bed no is: " << bedNumber << endl;
        cout << "Nature of illness is: " << natureOfillness << endl;
    }
};
class Patient : public Ward, public Room
{
};
int main()
{
    int i;
    Patient p[5];

    for (i = 0; i < 5; i++)
    {
        p[i].getdata();
        p[i].gdata();
        p[i].gtdata();

        cout << endl;

        p[i].printdata();
        p[i].pdata();
        p[i].ptdata();

        cout << endl;
    }
    return 0;
```

}

**OUTPUT:**



```
Enter name: Yatharth
Enter Roll NO: 19
Enter ward No: 1
Enter Bed No: 23
Nature of illness: covid

Name: Yatharth
ROLL No: 19
Ward no is: 1
Bed no is: 23
Nature of illness is: covid

Enter name: Riya
Enter Roll NO: 20
Enter ward No: 2
Enter Bed No: 18
Nature of illness: covid

Name: Riya
ROLL No: 20
Ward no is: 2
Bed no is: 18
Nature of illness is: covid

Enter name: Dev
Enter Roll NO: 21
Enter ward No: 3
Enter Bed No: 15
Nature of illness: covid

Name: Dev
ROLL No: 21
Ward no is: 3
Bed no is: 15
Nature of illness is: covid
```



```
Enter name: Arpan
Enter Roll NO: 22
Enter ward No: 4
Enter Bed No: 8
Nature of illness: covid

Name: Arpan
ROLL No: 22
Ward no is: 4
Bed no is: 8
Nature of illness is: covid

Enter name: Harmini
Enter Roll NO: 23
Enter ward No: 5
Enter Bed No: 5
Nature of illness: covid

Name: Harmini
ROLL No: 23
Ward no is: 5
Bed no is: 5
```

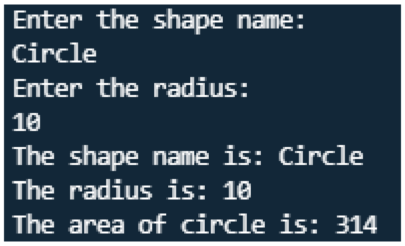
**CONCLUSION:** In this practical we learnt how to use the concept of virtual base class and hybrid inheritance.

| No. | Aim of the Practical |
|---|---|
| 34. | **Create a class shape having data member shape_name and member function to get and print shape_name. Derive a Class Circle which is inherited publicly from class shape and having data members radius of a circle and member function to get and print radius of a circle. Derive a Class Area which is inherited publicly from Class Circle and having data members area_of_circle and member function display () which displays area of a circle. Use object of class Area in main () function and get and display all the information. Use the concepts of Multilevel Inheritance.** |

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;

class shape
{
protected:
    string shape_name;

public:
    void getdata()
    {
        cin >> shape_name;
    }
};

class Circle : public shape
{
protected:
    float radius;

public:
    void getdata1()
    {
        cin >> radius;
    }
};

class Area : public Circle
{
protected:
    float area_of_circle;

public:
    float area()
    {
        area_of_circle = 3.14 * radius * radius;
        return area_of_circle;
    }
    void putdata2()
    {
        cout << "The shape name is: " << shape_name << endl;
        cout << "The radius is: " << radius << endl;
        cout << "The area of circle is: " << area() << endl;
    }
};
int main()
{
    Area a1;
    cout << "Enter the shape name: " << endl;
    a1.getdata();
    cout << "Enter the radius: " << endl;

    a1.getdata1();
    a1.putdata2();
}
```

**OUTPUT:**



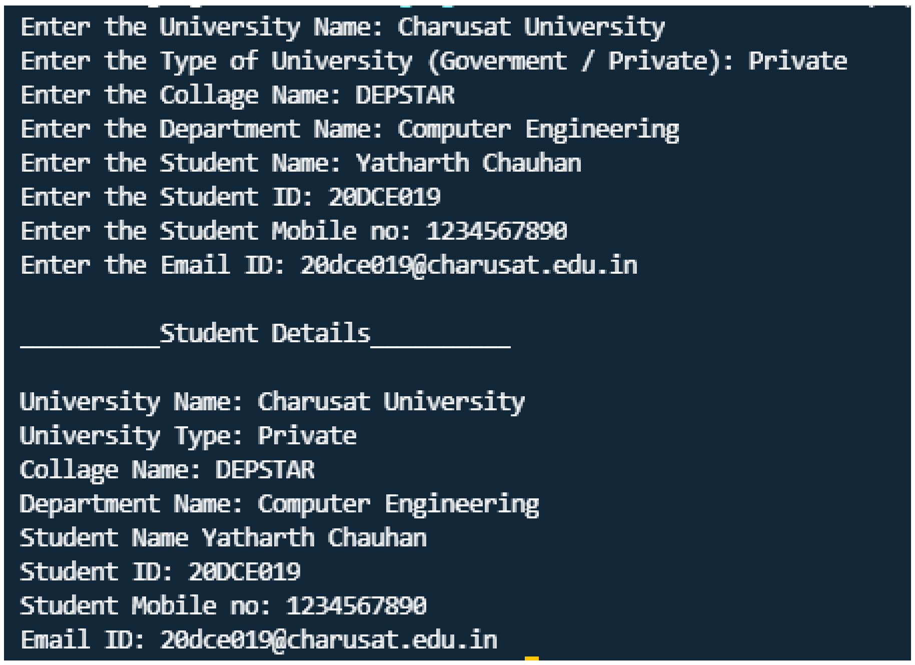
```
Enter the shape name:
Circle
Enter the radius:
10
The shape name is: Circle
The radius is: 10
The area of circle is: 314
```

**CONCLUSION:** In this practical we learnt how to use the concept of multilevel inheritance.

| No. | Aim of the Practical |
|---|---|
| 35. | **Create one application a group of 3 person which implement all type of inheritance.** |

**PROGRAM CODE :**

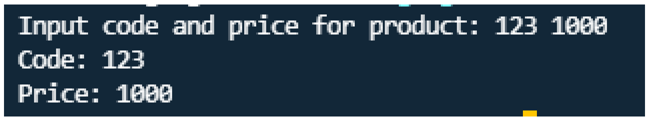
```
#include <iostream>
#include <iomanip>
using namespace std;

class University
{
public:
    char university_name[30], type[10];
    void getdata()
    {
        cout << "Enter the University Name: ";
        cin.getline(university_name, 30);
        cout << "Enter the Type of University (Goverment / Private): ";
        cin.getline(type, 30);
    }
    void putdata()
    {
        cout << "University Name: " << university_name << endl;
        cout << "University Type: " << type << endl;
    }
};

class Collage : public University
{
public:
    char collage_name[30], department_name[25];

    void gtdata()
    {
        cout << "Enter the Collage Name: ";
        cin.getline(collage_name, 30);
        cout << "Enter the Department Name: ";
        cin.getline(department_name, 25);
    }
    void ptdata()
    {
        cout << "Collage Name: " << collage_name << endl;
        cout << "Department Name: " << department_name << endl;
    }
};

class Student : public Collage
{
public:
    string id, mobile_no, email;
    char name[20];

    void gdata()
    {
        cout << "Enter the Student Name: ";
        cin.getline(name, 20);
        cout << "Enter the Student ID: ";
        cin >> id;
        cout << "Enter the Student Mobile no: ";
        cin >> mobile_no;
        cout << "Enter the Email ID: ";
        cin >> email;
        cout << endl;
    }
    void pdata()
    {
        cout << "Student Name " << name << endl;
        cout << "Student ID: " << id << endl;
        cout << "Student Mobile no: " << mobile_no << endl;
        cout << "Email ID: " << email << endl;
    }
};
int main()
{
    Student s;
    s.getdata();
    s.gtdata();
    s.gdata();

    cout << "___________Student Details___________" << endl;

    cout << endl;

    s.putdata();
    s.ptdata();
    s.pdata();
```

|  |  |
|---|---|
|  | return 0;<br>}<br><br>**OUTPUT:**<br><br><br>```<br>Enter the University Name: Charusat University<br>Enter the Type of University (Goverment / Private): Private<br>Enter the Collage Name: DEPSTAR<br>Enter the Department Name: Computer Engineering<br>Enter the Student Name: Yatharth Chauhan<br>Enter the Student ID: 20DCE019<br>Enter the Student Mobile no: 1234567890<br>Enter the Email ID: 20dce019@charusat.edu.in<br><br>__________Student Details__________<br><br>University Name: Charusat University<br>University Type: Private<br>Collage Name: DEPSTAR<br>Department Name: Computer Engineering<br>Student Name Yatharth Chauhan<br>Student ID: 20DCE019<br>Student Mobile no: 1234567890<br>Email ID: 20dce019@charusat.edu.in<br>```<br><br>**CONCLUSION:** In this practical we learnt how to use inheritance concept to make application. |

| No. | Aim of the Practical |
|---|---|
| 36. | **What is the output of the following code:**<br><br>**(a) Pointer to Objects**<br><br>**PROGRAM CODE :**<br><br>#include <iostream><br>using namespace std;<br>class product |

```
{
    int code;
    float price;

public:
    void getdata(int a, float b)
    {
        code = a;
        price = b;
    }
    void show()
    {
        cout << "Code: " << code << endl;
        cout << "Price: " << price << endl;
    }
};
int main()
{
    product *p = new product;
    product *d = p;
    int x, i;
    float y;
    cout << "Input code and price for product: ";
    cin >> x >> y;
    p->getdata(x, y);
    d->show();
}
```

**OUTPUT:**



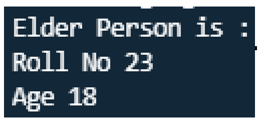
```
Input code and price for product: 123 1000
Code: 123
Price: 1000
```

**(b) this pointer**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class student
{
    int roll_no;
    float age;

public:
    student(int r, float a)
    {
        roll_no = r;
        age = a;
    }
    student &greater(student &x)
    {
        if (x.age >= age)
            return x;
        else
            return *this;
    }
    void display()
    {
        cout << "Roll No " << roll_no << endl;
        cout << "Age " << age << endl;
    }
};
int main()
{
    student s1(23, 18), s2(30, 20), s3(45, 16);
    student s = s1.greater(s3);
    cout << "Elder Person is :" << endl;
    s.display();
}
```

**OUTPUT:**



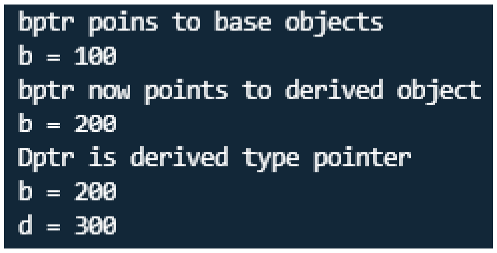
```
Elder Person is :
Roll No 23
Age 18
```

**(c) Pointers to Derived Objects**

**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class BC
{
public:
    int b;
    void show()
    {
        cout << "b = " << b << endl;
    }
};
class DC : public BC
{
public:
    int d;
    void show()
    {
        cout << "b = " << b << endl;
        cout << "d = " << d << endl;
    }
};
int main()
{
   BC *bptr;
   BC base;
   bptr = &base;
   bptr->b = 100;
   cout << "bptr poins to base objects" << endl;
   bptr->show();
   DC derived;
   bptr = &derived;
   bptr->b = 200;
   /*bptr->b = 300;*/ // wont work
   cout << "bptr now points to derived object" << endl;
   bptr->show();
   DC *dptr;
   dptr = &derived;
   dptr->d = 300;
   cout << "Dptr is derived type pointer" << endl;
   dptr->show();
   return 0;
}
```

**OUTPUT:**



```
bptr poins to base objects
b = 100
bptr now points to derived object
b = 200
Dptr is derived type pointer
b = 200
d = 300
```

**CONCLUSION:** In this practical we learn about pointer to object, this pointer and pointer to derived objects.

| No. | Aim of the Practical |
|---|---|
| 37. | **Create a class Media that stores the title (a string) and price (float). Class Mediahas two argument constructor which initializes data members of class Media. Also declare a virtual function display () in Class Media. From the class Media derive two classes: Class book, which contains data member page count (int): and Class tape, which contains data member playing time in minutes (float). Both Class book and Class tape should have a constructor which initializes base class constructor as well as its own data members and display ( ) function which displays book details and tape details respectively. Write a main ( ) to test book and tape classes by creating instances of them, asking the user to fill data and displaying them. Use the concept of Virtual function and Constructor in Derived Class.** |

**<u>PROGRAM CODE :</u>**

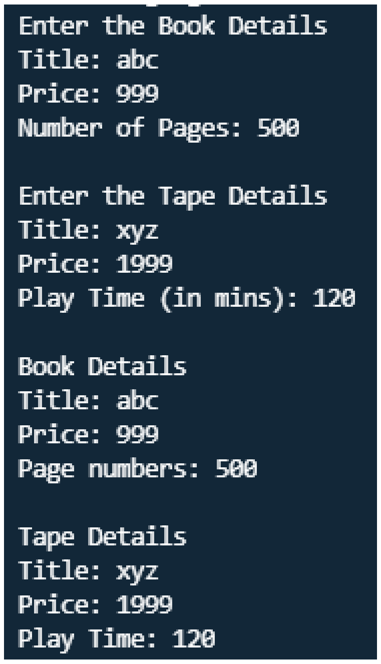
```
#include <iostream>
using namespace std;
class Media
{
public:
    string title;
    int price;
    Media()
    {
    }
    Media(string s, int p)
    {
        title = s;
        price = p;
    }
    virtual void display()
    {
    }
```

```
};
class Book : public Media
{
public:
   int pagecount;
   Book(string s, int p, int a) : Media(s, p)
   {
      pagecount = a;
   }
   void display()
   {
      cout << "Title: " << title << endl;
      cout << "Price: " << price << endl;
      cout << "Page numbers: " << pagecount << endl;
      cout << endl;
   }
};
class Tape : public Media
{
public:
   float time;
   Tape(string s, int p, int b) : Media(s, p)
   {
      time = b;
   }
   void display()
   {
      cout << "Title: " << title << endl;
      cout << "Price: " << price << endl;
      cout << "Play Time: " << time << endl;
   }
};
int main()
{
```

```
string title;
int price;
int pagecount;
int time;

cout << "Enter the Book Details" << endl;
cout << "Title: ";
cin >> title;
cout << "Price: ";
cin >> price;
cout << "Number of Pages: ";
cin >> pagecount;

cout << endl;

Book book1(title, price, pagecount);

cout << "Enter the Tape Details" << endl;
cout << "Title: ";
cin >> title;
cout << "Price: ";
cin >> price;
cout << "Play Time (in mins): ";
cin >> time;

cout << endl;

Tape tape1(title, price, time);

Media *m1, *m2;
m1 = &book1;
m2 = &tape1;

cout << "Book Details";
```

```
    cout << endl;
    m1->display();
    cout << "Tape Details";
    cout << endl;
    m2->display();
}
```

**OUTPUT:**



```
Enter the Book Details
Title: abc
Price: 999
Number of Pages: 500

Enter the Tape Details
Title: xyz
Price: 1999
Play Time (in mins): 120

Book Details
Title: abc
Price: 999
Page numbers: 500

Tape Details
Title: xyz
Price: 1999
Play Time: 120
```

**CONCLUSION:** In this practical we learn the concept of the virtual function and use it for the derived class. Also learn the initialization of constructor using derived class.

| **No.** | **Aim of the Practical** |
|---|---|
| **38.** | **Create an Abstract class vehicle having average as data and pure virtual function getdata() and putdata(). Derive class car and truck from class vehicle having data members: fuel type (petrol, diesel, CNG) and no of wheels respectively. Write a main ( ) that enters the data of two cars and a truck and** |

**display the details of them. Use the concept of Abstract Base class and Pure Virtual functions.**

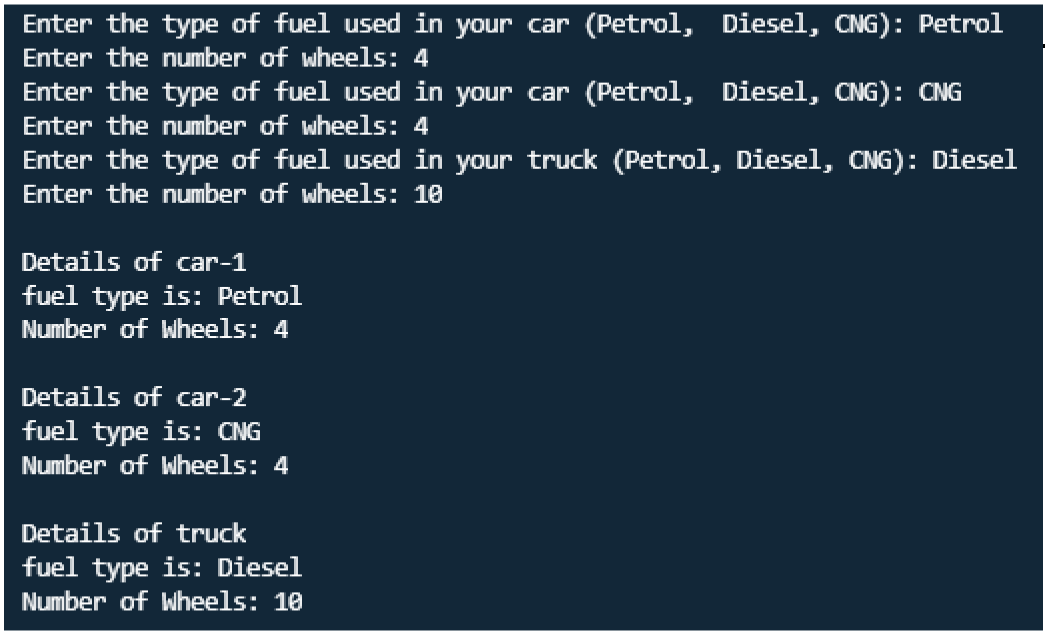
**PROGRAM CODE :**

```
#include <iostream>
using namespace std;
class vehicle
{
public:
    float average;
    virtual void getdata() = 0;
    virtual void putdata() = 0;
};
class car : public vehicle
{

public:
    string fuel_type;
    int nowheel;
    void getdata()
    {
        cout << "Enter the type of fuel used in your car (Petrol,  Diesel, CNG): ";
        cin >> fuel_type;
        cout << "Enter the number of wheels: ";
        cin >> nowheel;
    }
    void putdata()
    {
        cout << "fuel type is: " << fuel_type << endl;
        cout << "Number of Wheels: " << nowheel << endl;
        cout << endl;
    }
};
```

```
class truck : public vehicle
{

public:
    string fuel;
    int wheel;
    void getdata()
    {
        cout << "Enter the type of fuel used in your truck (Petrol, Diesel, CNG): ";
        cin >> fuel;
        cout << "Enter the number of wheels: ";
        cin >> wheel;

        cout << endl;
    }
    void putdata()
    {
        cout << "fuel type is: " << fuel << endl;
        cout << "Number of Wheels: " << wheel << endl;
        cout << endl;
    }
};
main()
{
    car c1, c2;
    truck t;
    c1.getdata();
    c2.getdata();
    t.getdata();
    cout << "Details of car-1" << endl;
    c1.putdata();
    cout << "Details of car-2" << endl;
    c2.putdata();
    cout << "Details of truck" << endl;
```

```
   t.putdata();
}
```

**OUTPUT:**



```
Enter the type of fuel used in your car (Petrol,  Diesel, CNG): Petrol
Enter the number of wheels: 4
Enter the type of fuel used in your car (Petrol,  Diesel, CNG): CNG
Enter the number of wheels: 4
Enter the type of fuel used in your truck (Petrol, Diesel, CNG): Diesel
Enter the number of wheels: 10

Details of car-1
fuel type is: Petrol
Number of Wheels: 4

Details of car-2
fuel type is: CNG
Number of Wheels: 4

Details of truck
fuel type is: Diesel
Number of Wheels: 10
```

**CONCLUSION:** In this practical learn the concept of Abstract Base class and Pure Virtual functions and how to implement it.

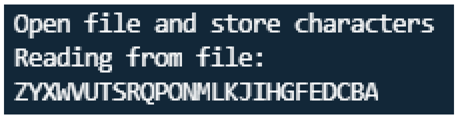
| No. | Aim of the Practical |
|---|---|
| 39. | **Write a program that creates a text file that contains ABC…Z. A program should print the file in reverse order on the screen. i.e. ZYX…BA. Use concept of Opening the file using constructor and open() function. Use all error handling functions like eof(), fail(), bad(), good() and functions for manipulation of file pointer like seekg() and tellg().**<br><br>**PROGRAM CODE :**<br><br>#include &lt;iostream&gt;<br>#include &lt;fstream&gt; |

```
#include <string.h>
using namespace std;
int main()
{
   fstream fp;
   char str[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   cout << "Open file and store characters" << endl;

   int len = strlen(str);

   fp.open("2CE1.txt", ios::out);

   for (int i = 0; i < len; i++)
   {
      fp.put(str[i]);
   }
   fp.close();

   fp.open("2CE1.txt", ios::in);

   cout << "Reading from file:" << endl;
   fp.seekg(-1, ios::end);

   char ch;

   while (fp.tellg() >= 0)
   {
      fp.get(ch);
      cout << ch;
      fp.seekg(-2, ios::cur);
   }
   fp.close();

   return 0;
```

| | |
|---|---|
| | } |

}

**OUTPUT:**



```
Open file and store characters
Reading from file:
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

**CONCLUSION:** In this practical we learn the concept of Opening the file using constructor and open() function. And also the use of all error handling functions like eof(), fail(), bad(), good() and functions for manipulation of file pointer like seekg() and tellg().