

OOPC Practical Programs

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

RUSHIK RATHOD

20DCS103 DEPSTAR CSE : 3 - Batch : A

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.

PROGRAM CODE:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "\t\t *****" << endl;
    cout << "*   Programming Assignment 1   *" << endl;
    cout << "*   Computer Programming I       *" << endl;
    cout << "*   Author : Rushik Rathod       *" << endl ;
    cout << "*   Due Date : Friday, March 19  *\n";
    cout << "\t\t*****\t\t" << endl;
    cout << "\n\n Made by : RUSHIK RATHOD\n      20DCS103\n";
    return 0;
}
```

OUTPUT:

```
*****
*   Programming Assignment 1   *
*   Computer Programming I    *
*   Author : Rushik Rathod    *
*   Due Date : Friday, March 19 *
*****

Made by : RUSHIK RATHOD
         20DCS103

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

CONCLUSION:

I got to know about the basics of C++ programming including syntax, how to print the sentences with different symbols. Additionally, learnt the difference between endl manipulator and \n.

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 a=1,i,n;
    for(i=1;i<=4;i++)
    {
        for(n=1;n<=4;n++)
        {
            a=i*n;
            cout << setw(5) <<a;

        }
        cout << endl;
    }
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

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

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)    execution time : 0.016 s
Press any key to continue.
_
```

CONCLUSION:

The endl manipulator prints the output in a new line character and flush the stream and setw manipulator set the field width in the output to the number which must be ≥ 0 .

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 a, b, sum, *p1, *p2;
    cout<<"Enter two floating numbers to get addition : " << endl;
    cin >> a >> b;
    p1 = &a;
    p2 = &b;
    sum = *p1 + *p2;
    cout << "\nSum of two numbers...\n" << setprecision(2) << fixed << "By using
fixed : " << sum << endl;
    cout << scientific << "By using scientific : " << sum << endl;
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103" << endl;
    return 0;
}
```

OUTPUT:

```
Enter two floating numbers to get addition :  
12.25  
13.25  
  
Sum of two numbers...  
By using fixed : 25.50  
By using scientific : 2.55e+01  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 3.848 s  
Press any key to continue.  
_
```

CONCLUSION:

The setprecision(n) manipulators will display the n number precision after the '.' .

Further, fixed manipulator will display the output in fixed point notations and by using scientific, output will be written in scientific notations.

4. Write a C++ program to find out sum of array element using Recursion.
Question: Show stepwise solution of winding and unwinding phase of recursion.

PROGRAM CODE:

```
#include <iostream>
using namespace std;
int sum(int arr[],int n)
{
    if(n==0)
    {
        return 0;
    }
    return arr[0]+sum(arr+1,n-1);
}
int main()
{
    int i,n;
    cout << "Enter the size of array :" << endl;
    cin >> n;
    int arr[n];
    cout << "Enter " << n << " values :" << endl;
    for(i=0; i<n; i++)
    {
        cin >> arr[i];
    }

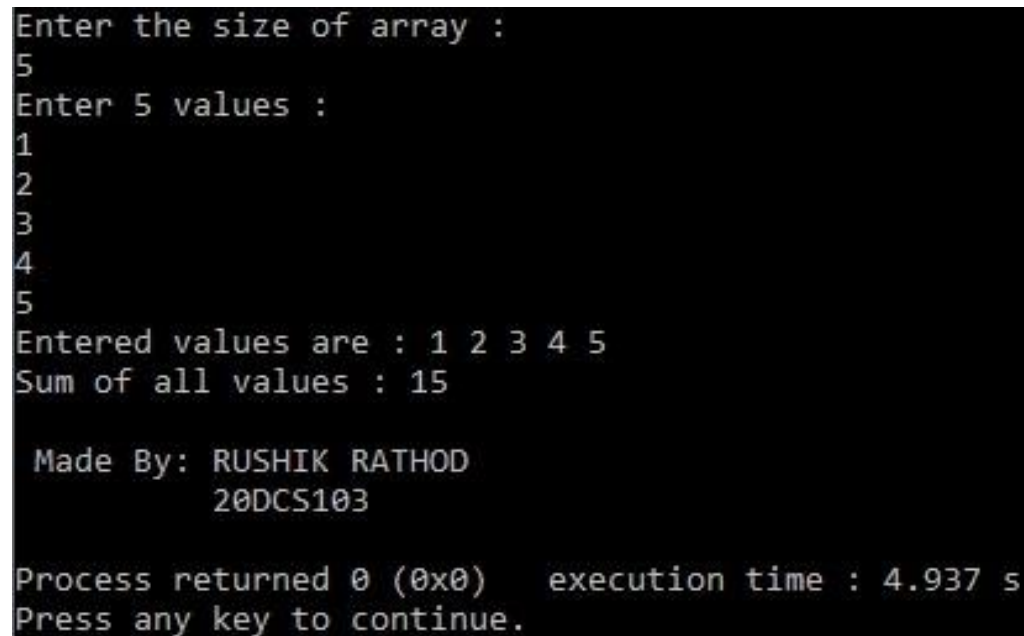
    cout << "Entered values are : ";
```



```
for(i=0; i<n; i++)
{
    cout << arr[i] << " ";
}

cout << "\nSum of all values : " << sum(arr,n);
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:



```
Enter the size of array :
5
Enter 5 values :
1
2
3
4
5
Entered values are : 1 2 3 4 5
Sum of all values : 15

Made By: RUSHIK RATHOD
      20DCS103

Process returned 0 (0x0)   execution time : 4.937 s
Press any key to continue.
```

CONCLUSION:

I learnt to get input from user for size of array and its elements, further I used the recursion to get the summation of the array elements.

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?

PROGRAM CODE:

```
#include<iostream>
using namespace std;
int main()
{
    int no1=10, no2=12;
    int & x=no1;
    int & r;
    int & c = NULL;
    int & d[2] = {no1,no2};
    cout<<"x = "<< x+20;
    cout<<"no1="<< no1+10;
    return 0;
}
```

OUTPUT:

Source code has errors ! So, no output to be shown.

CONCLUSION:

I. Can we declare an array of references?

No, we cannot declare an array of references because if we apply any operation on a reference, it will directly affect the original value. That means we cannot get pointer to the reference variable, instead of using 'array of reference' we should use 'array of pointer'

II. Can we assign NULL value to reference variable?

No, we cannot assign a NULL value to reference variable.

III. Is Reference variable a pointer variable?

No, reference variable is not a pointer variable.

IV. Can we declare a reference variable without initializing it?

No, we cannot declare a reference variable without initializing it. We must initialize reference variable at the time of declaration.

V. Does Reference Variable change the original value of variable?

Yes, reference variable do change the original value of variable.

6. Find output of the following code: Explain how scope Resolution operator is used to access global version of a variable.

PROGRAM CODE:

```
#include <iostream>
#include <conio.h>
using namespace std;
int m=30;
int main()
{
    int m=20;
    {
        int m=10;
        cout<<"we are in inner block"<<endl;
        cout<<"value of m="<<m<<"\n";
        cout<<"value of ::m="<<::m<<"\n";
    }
    cout<<"we are in outer block"<<endl;
    cout<<"value of m="<<m<<"\n";
    cout<<"value of ::m="<<::m<<"\n";
    getch();
    return 0;
}
```

OUTPUT:

```
we are in inner block
value of m=10
value of ::m=30
we are in outer block
value of m=20
value of ::m=30

Process returned 0 (0x0)   execution time : 6.936 s
Press any key to continue.
_
```

CONCLUSION:

When the value of m is declared in an inner block m, then it will access the value which is declared in that block only, but when we use scope resolution operator, m will access the value which is declared globally.

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 i,n;
    cout << "Enter the number of array elements : " << endl;
    cin >> n ;
    int *arr = new int(n);
    cout << "Enter the elements : " << endl;
    for(i=0; i<n; i++)
    {
        cin >> *(arr + i);
    }
    cout << "Entered array elements are : " << endl;
    for(i=0; i<n; i++)
    {
        cout << *(arr + i) << endl;
    }
    cout << "Now adding 2 in them, we get : " << endl;
    for(i=0; i<n; i++)
    {
```

```
        cout << *(arr + i) + 2 << endl;
    }

    int *ptr[i];
    for(i=0; i<n; i++)
    {
        ptr[i] = &arr[i];
    }

    delete [] arr;
    cout << "ARRAY DELETED !" << endl;

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Enter the number of array elements :
3
Enter the elements :
1
2
3
Entered array elements are :
1
2
3
Now adding 2 in them, we get :
3
4
5
ARRAY DELETED !

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 2.823 s
Press any key to continue.
```

CONCLUSION:

I learnt to allocate memory to an array with the help of dynamic memory management operator “new” and delete it by using “delete” operator.

Also, learnt to add values to the elements of an array in the block of for loop by changing the allocated addresses.

8. Find Error in the following code of a program and give explanation why these errors exist. NOTE: USE reference of other material as there is a mistake in 6th edition of Bala Guruswamy.

PROGRAM CODE:

//1. CONSTANT POINTER

```
#include <iostream>
using namespace std;
int main()
{
    int var1 = 35, var2 = 20;
    int *const ptr = &var1;
    ptr = &var2;    // constant pointer cannot point to any other variable, if a constant
                   // pointer is pointing to some variable
    cout << "var1 = " << *ptr;
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

//2. POINTER TO CONSTANT

```
#include <iostream>
using namespace std;
int main()
{
    int var1 = 43;
    const int* ptr = &var1;
    *ptr = 1;    //the value of the variable that the pointer points cannot be changed.
    var1=34;
    cout << "var1 = " << *ptr;
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

//3. CONSTANT POINTER TO A CONSTANT

```
#include <iostream>
using namespace std;
int main()
{
    int var1 = 0, var2 = 0;
    const int* const ptr = &var1;
    *ptr = 1;           // it can not change the value placed at this address
    ptr = &var2;        // it can not change the address of the variable to which it is
                        pointing
    cout << "Var1 = " << *ptr;
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

Source code has errors ! So, no output to be shown.

CONCLUSION:

From this practical, I learnt the various types of logic by using constant pointer, pointer to constant and constant pointer to a pointer. Also, I get to know about the different outputs on the console screen by using them.

9. Find the output of following program. Explain the use of bool data type.

PROGRAM CODE:

```
#include<iostream>
using namespace std;
int main()
{
    bool a = 321, b;

    cout << "Bool a Contains : " << a << endl;
    // a = 1 by converting the data type integers to bool type

    int c = true; int d = false;
    // in bool type, true is 1 and false is 0

    cout << "c = " << c << endl << "d = " << d;    // c = 1, d = 0
    c = a + a;                                     // c = 1 + 1 = 2
    // but will be printed c = 1 on console, as the value of c is non zero
    cout << "\nInteger c contain : " << c;

    b = c + a;                                     // b = 2 + 1 = 3
    // but will be printed b = 1 on console, as the value of b is non zero
    cout << "\nBool b contain : " << b;

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Bool a Contains : 1
c = 1
d = 0
Integer c contain : 2
Bool b contain : 1

  Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

CONCLUSION:

When we use the bool data type in the program then it will return 1 for true as well as non-zero numbers and 0 for false and zero.

10. Define three functions named divide(). First function takes numerator and denominator as an input argument and checks if 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 numbers as argument and finds out the average of the numbers. Use the concept of Function Overloading / static binding.

PROGRAM CODE:

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

void divide(int, int);
void divide(int);
void divide(float, float, float);
int main()
{
    int x,y;
    cout << "Enter numerator and denominator respectively : " << endl;
    cin >> x >> y ;
    divide(x, y);

    cout << "Enter a number to check prime number : ";
    cin >> x;
    divide(x);

    float a,b,c;
    cout << "Enter three float numbers to get an average : " << endl;
    cin >> a >> b >> c ;
    divide(a, b, c);

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
}
void divide(int x, int y)
```

```
{
    if (y == 0 )
        cout << "Denominator can not be 0 !" << endl;
    else
        cout << "Answer = " << x/y << endl << endl;
}

void divide(int x)
{
    int n, i, m, temp = 0;
    m = x/2;
    for(i = 2; i <= m; i++)
    {
        if(x % i == 0)
        {
            cout << x << " is not Prime number." << endl << endl;
            temp = 1;
            break;
        }
    }
    if (temp == 0)
        cout << x << " is Prime number." << endl << endl;
}

void divide(float a, float b, float c)
{
    cout << "Average = " << (a+b+c)/3 << endl;
}
```

OUTPUT:

```
Enter numerator and denominator respectively :  
10  
2  
Answer = 5  
  
Enter a number to check prime number : 7  
7 is Prime number.  
  
Enter three float numbers to get an average :  
11.5  
12.50  
55.67  
Average = 26.5567  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 21.227 s  
Press any key to continue.
```

CONCLUSION:

This practical includes concept of function overloading, in which when we call the function with same name, it will call the function of the same number of arguments which we are passing.

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 &, int &);
int main()
{
    int x,y;
    cout << "Enter first number : ";
    cin >> x;
    cout << "Enter second number : ";
    cin >> y;
    tonLarge(x,y);
    cout << "\nAfter setting 100 to a larger number, we get...\n" << endl;

    cout << "First number = " << x << endl;
    cout << "Second number = " << y << endl;

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
Int& tonLarge(int &x,int &y)
{
    if(x>y)
    {
        x = 100;
        return(x);
    }
    else
    {
        y = 100;
```



```
    return(y);  
}  
}
```

OUTPUT:

```
Enter first number : 18  
Enter second number : 22  
  
After setting 100 to a larger number, we get...  
  
First number = 18  
Second number = 100  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 7.055 s  
Press any key to continue.  
_
```

CONCLUSION:

When we pass the reference of a variable to a particular function, it will directly affect the original value. By using this method, one can save the memory and time additionally it makes the program easy to read.

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 temp=1;
    if(p>=0)
    {
        while(p-->0)
        {
            temp = temp*n;
        }
        return temp;
    }
    else
    {
        return 1/power(n,abs(p));
    }
}
int main()
{
    double n;
    int p, x;
    cout << "Enter double value : " << endl;
    cin >> n;
    cout << "\nDo you want to enter the power ?";
    cout << "\nPress 1 for YES and 2 for NO...\n";
```

```
cin >> x;
switch(x)
{
    case 1 :
        cout << "\nEnter the power : ";
        cin >> p;
        cout << "\nResult = " << power(n,p) << endl;
        break;
    case 2 :
        cout << "\nResult = " << power(n) << endl;
        break;
    default :
        cout << "\nInvalid entry !" << endl;
        break;
}
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
}
```

OUTPUT:

```
Enter double value :  
25  
  
Do you want to enter the power ?  
Press 1 for YES and 2 for NO...  
1  
  
Enter the power : 3  
  
Result = 15625  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 6.022 s  
Press any key to continue.  
_
```

```
Enter double value :  
25  
  
Do you want to enter the power ?  
Press 1 for YES and 2 for NO...  
2  
  
Result = 625  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 3.598 s  
Press any key to continue.
```

CONCLUSION:

I learnt one of the most useful concept: the default argument. It helps the program to get the default data when the actual argument is omitted in that respective code of the program.

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 using structure:

```
#include <iostream>
using namespace std;
struct Rectangle
{
    int width;
    int height;

    void get_values()
    {
        cout << "Enter the width for a rectangle : ";
        cin >> width;
        cout << "Enter the height for a rectangle : ";
        cin >> height;
    }
    int area()
    {
        return (width*height);
    }
};

int main()
{
    Rectangle rect1;
    rect1.get_values();
    cout<< "\nArea of the rectangle is : " << rect1.area();
```

```
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";  
return 0;  
}
```

OUTPUT:

```
Enter the width for a rectangle : 13  
Enter the height for a rectangle : 5  
  
Area of the rectangle is : 65  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 3.190 s  
Press any key to continue.  
_
```

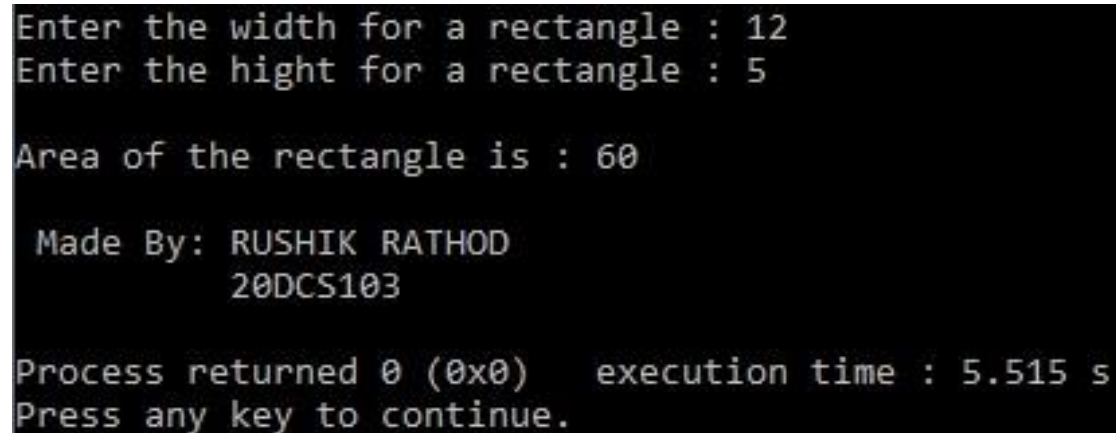
PROGRAM CODE using class:

```
#include <iostream>  
#include <conio.h>  
using namespace std;  
class rectangle  
{  
    int w,h;  
  
public :  
    get_values()  
    {  
        cout << "Enter the width for a rectangle : ";  
        cin >> w;  
        cout << "Enter the hight for a rectangle : ";  
        cin >> h;  
    }  
    area()
```

```
{
    cout << "\nArea of the rectangle is : " << w*h;
}
};

int main()
{
    rectangle r1;
    r1.get_values();
    r1.area();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
}
```

OUTPUT:



```
Enter the width for a rectangle : 12
Enter the hight for a rectangle : 5

Area of the rectangle is : 60

Made By: RUSHIK RATHOD
      20DCS103

Process returned 0 (0x0)   execution time : 5.515 s
Press any key to continue.
_
```

CONCLUSION:

1. The difference between a structure and a class is that structure members have public access by default and class members have private access by default.
2. We can derive another class from base class whereas in the structure, we cannot perform such type of inheritance.

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 : $\text{batting average} = \text{runs} / (\text{innings} - \text{notout})$

PROGRAM CODE:

```
#include <iostream>
#include <conio.h>
using namespace std;
class Batsman
{
private :
    char batsman_name[100];
    int bcode;
    int innings;
    int not_out;
    int runs;
    float batting_average;

// Member function declaration
public :

    inline float calcavg();
    void getdata();
    void putdata();

};

float Batsman::calcavg()
```



```
{
    return(runs/(innings-not_out));
}

void Batsman::getdata()
{
    cout << "Enter the name of batsman : ";
    gets(batsman_name);
    cout << "\nEnter the batsman code : ";
    cin >> bcode;
    cout << "\nEnter the number of innings : ";
    cin >> innings;
    cout << "\nEnter the number of not out : ";
    cin >> not_out;
    cout << "\nEnter the runs : ";
    cin >> runs;
}

void Batsman::putdata()
{
    cout << "\n\n*****Entered information*****" << endl;
    cout << "\nName of batsman : " << batsman_name << "\nBatsman code : " <<
bcode << "\nInnings : " << innings ;
    cout << "\nNumber of not out : " << not_out << "\nRuns : " << runs;
    cout << "\n\nBatting average of batsman : " << calcavg() << endl;
}

int main()
{
    Batsman b1;
    b1.getdata();
    b1.putdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n        20DCS103 \n";
}
```

OUTPUT:

```
Enter the name of batsman : Rushik Rathod

Enter the batsman code : 103

Enter the number of innings : 50

Enter the number of not out : 48

Enter the runs : 1000

*****Entered information*****

Name of batsman : Rushik Rathod
Batsman code : 103
Innings : 50
Number of not out : 48
Runs : 1000

Batting average of batsman : 500

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 14.736 s
Press any key to continue.
```

CONCLUSION:

When we define member functions outside the class we need to use scope resolution operator and to make a function inline we need to use inline keyword in the beginning of the declaration.

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
{
private:
    int rupee;
    int paisa;
public:
    void enter()
    {
        cout << "\nEnter the rupee : ";
        cin >> rupee;
        cout << "Enter the paisa : ";
        cin >> paisa;
        if(paisa>100)
        {
            rupee = rupee + paisa/100;
            paisa = paisa%100;
        }
    }

    void show()
    {
```

```
    cout << "\nThe total amount entered : " << rupee << "." ;
    if(paisa<10) { cout<<"0"; }
    cout << paisa << endl;
}

currency sum(currency c)
{
    currency temp;
    temp.paisa = paisa + c.paisa;
    temp.rupee = rupee + c.rupee;
    if(paisa + c.paisa > 100)
    {
        int t = (paisa + c.paisa)/100;
        temp.paisa = (paisa+c.paisa)%100;
        temp.rupee+=t;
    }
    return temp;
}

void add(currency c1,currency c2)
{
    paisa = c1.paisa + c2.paisa;
    rupee = c1.rupee + c2.rupee;
    if(c1.paisa + c2.paisa>100)
    {
        int t = (c1.paisa + c2.paisa)/100;
        paisa = (c1.paisa + c2.paisa)%100;
        rupee = rupee + t;
    }
}

};

int main()
{
    currency c1,c2,c3;
```

```
c1.enter();
c1.show();
c2.enter();
c2.show();
c3=c1.sum(c2);
cout << endl << "After c3=c1.sum(c2) ";
c3.show();
cout << endl << "After c3.add(c1,c2) ";
c3.add(c1,c2);
c3.show();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:

```
Enter the rupee : 15
Enter the paisa : 103

The total amount entered : 16.03

Enter the rupee : 20
Enter the paisa : 50

The total amount entered : 20.50

After c3=c1.sum(c2)
The total amount entered : 36.53

After c3.add(c1,c2)
The total amount entered : 36.53

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 6.211 s
Press any key to continue.
```

CONCLUSION:

From this program I learnt to pass the objects as function arguments and returning object. Additionally, when we define more than one function with the same name but with the different arguments it is called the concept of function overloading.

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"

PROGRAM CODE:

```
#include <iostream>
#include <cmath>
using namespace std;
class Dist
{
    private:
        int feet;
        float inches;
    public:
        Dist()
        {
            feet=0;
            inches=0;
        }
        void add_data();
        void display_data();
        void scale(Dist &,float);
};

float remainder(float num)
{
    float sum;
    int i;
    for(i=1;i>0;++i)
    {
        sum=12*i;
```

```
        if(sum>num)
        {
            break;
        }
        if(sum==num)
        {
            return 0;
        }
    }
    return (num-(12*(i-1)));
}

void Dist::add_data()
{
    cout << "***ENTER THE DETAILS***" << endl;
    cout << "Enter the distance in feet : ";
    cin >> feet;
    cout << "Enter the distance in inches : ";
    cin >> inches;
    if(inches>=12)
    {
        feet+=(inches/12);
        inches=remainder(inches);
    }
}

void Dist::display_data()
{
    cout << "The distance = " << feet << "'-' " << inches << "'''" << endl;
}

void Dist::scale(Dist &d,float Scale_factor)
{
    float temp1 = feet*Scale_factor;
    d.feet = feet*Scale_factor;
```



```
float temp2 = floor(temp1);
d.inches = inches*Scale_factor+12*(temp1-temp2);
if(d.inches >= 12)
{
    d.feet+=(d.inches/12);
    d.inches=remainder(d.inches);
}
}

int main()
{
    Dist d1,d2;
    float scale;
    d1.add_data();
    d1.display_data();
    cout << "Enter the scale factor : ";
    cin >> scale;
    d1.scale(d2,scale);
    d2.display_data();
    cout << "\n\n Made By: RUSHIK RATHOD \n    20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
***ENTER THE DETAILS***
Enter the distance in feet : 50
Enter the distance in inches : 26
The distance = 52'-2''
Enter the scale factor : 0.5
The distance = 26'-1''

Made By: RUSHIK RATHOD
20DCS103

Process returned 0 (0x0)   execution time : 7.739 s
Press any key to continue.
```

CONCLUSION:

When we define member functions by passing reference, it will directly affect the original value of a variable. Therefore, the function works with the original value and hence it becomes easy for programmer to read and execute.

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 putdata () 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.

PROGRAM CODE:

```
#include <iostream>
#include <string.h>
using namespace std;
class Gate
{
    private:
        char Registration_number[20];
        char Name_of_student[20];
        char Examination_center[10];
        static int ECV_Cnt;
        static int ECS_Cnt;
        static int ECA_Cnt;
    public:
        void getdata();
        void putdata();
        static void getcount();
};
```

```
int Gate::ECV_Cnt=0;
int Gate::ECS_Cnt=0;
int Gate::ECA_Cnt=0;

void Gate::getdata()
{
    cout << "\nName : ";
    cin >> Name_of_student;
    cout << "Registration number : ";
    cin >> Registration_number;
    cout << "Examination center : ";
    cin >> Examination_center;
    if(!strcmp(Examination_center,"v")) { ECV_Cnt++;}
    if(!strcmp(Examination_center,"s")) { ECS_Cnt++;}
    if(!strcmp(Examination_center,"a")){ ECA_Cnt++;}
}

void Gate ::putdata()
{
    cout << "\nName : " << Name_of_student << endl;
    cout << "Registration no : " << Registration_number << endl;
    cout << "Exam center : ";
    if(!strcmp(Examination_center,"v")) { cout << "Vadodara" << endl; }
    if(!strcmp(Examination_center,"s")) { cout << "Surat" << endl;}
    if(!strcmp(Examination_center,"a")) { cout << "Ahemdabad" << endl;}
}

void Gate::getcount()
{
    cout << "\n***City wise number of students***\n" << endl;
    cout << "Vadodara : " << ECV_Cnt << endl;
    cout << "Surat : " << ECS_Cnt << endl;
    cout << "Ahemdabad : " << ECA_Cnt << endl;
}
```

```
int main()
{
    Gate s[5];
    int i;
    cout << "-----Enter the information-----"<<endl;
    cout << "(Note : Enter v for Vadodara, s for Surat, a for Ahmedabad.)" << endl;
    for(i=0;i<5;++i)
    {
        s[i].getdata();
    }

    int j;
    cout << "\n\n***Entered information***" << endl;
    for(j=0;j<5;++j)
    {
        s[j].putdata();
    }

    Gate::getcount();

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
-----Enter the information-----  
(Note : Enter v for Vadodara, s for Surat, a for Ahmedabad.)  
  
Name : Rushik  
Registration number : 103  
Examination center : v  
  
Name : Vivek  
Registration number : 111  
Examination center : s  
  
Name : Om  
Registration number : 555  
Examination center : a  
  
Name : Raj  
Registration number : 135  
Examination center : v  
  
Name : Deep  
Registration number : 80  
Examination center : s  
  
***Entered information***  
  
Name : Rushik  
Registration no : 103  
Exam center : Vadodara
```

```
***Entered information***  
  
Name : Rushik  
Registration no : 103  
Exam center : Vadodara  
  
Name : Vivek  
Registration no : 111  
Exam center : Surat  
  
Name : Om  
Registration no : 555  
Exam center : Ahemdabad  
  
Name : Raj  
Registration no : 135  
Exam center : Vadodara  
  
Name : Deep  
Registration no : 80  
Exam center : Surat  
  
***City wise number of students***  
  
Vadodara : 2  
Surat : 2  
Ahemdabad : 1
```

```
Name : Om
Registration no : 555
Exam center : Ahemdabad

Name : Raj
Registration no : 135
Exam center : Vadodara

Name : Deep
Registration no : 80
Exam center : Surat

***City wise number of students***

Vadodara : 2
Surat : 2
Ahemdabad : 1

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 60.366 s
Press any key to continue.
```

CONCLUSION:

I learnt the concept of static data member, static member function and array of objects and successfully utilize them in the above program.

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.

PROGRAM CODE:

```
#include <iostream>
using namespace std;
class Celsius;
class Fahrenheit
{
    private:
        float temp;
    public:
        Fahrenheit()
        {
            temp=0;
        }
        void add_data();
        void display();
        friend bool Compare_two_temprature(Fahrenheit f,Celsius c);
};

class Celsius
{
    private:
        float temperature;
    public:
        Celsius()
        {
```

```
        temperature=0;
    }
    void add_data();
    void display();
    friend bool Compare_two_temprature(Fahrenheit f,Celsius c);
};

void Fahrenheit:: add_data()
{
    cout << "Enter temperature in Fahrenheit : ";
    cin >> temp;
}
void Fahrenheit::display()
{
    cout << "Temperature = " << temp << " F" << endl;
}
void Celsius:: add_data()
{
    cout << "Enter temperature in Celsius : ";
    cin >> temperature;
}
void Celsius::display()
{
    cout << "Temperature = " << temperature << " C" << endl;
}
bool Compare_two_temprature(Fahrenheit f,Celsius c)
{
    float t = ((c.temperature*9)/5)+32;
    return f.temp > t ? true : false;
}
int main()
{
    Fahrenheit f;
    Celsius c;
    f.add_data();
```

```
c.add_data();

cout << "\n***Entered data***" << endl;
f.display();
c.display ();
if(Compare_two_temprature(f,c))
{
    cout << "\nClass Fahrenheit has more temperature !" << endl;
}
else
{
    cout << "\nClass Celsius has more temperature !" << endl;
}
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:

```
Enter temperature in Fahrenheit : 50
Enter temperature in Celsius : 35

***Entered data***
Temperature = 50 F
Temperature = 35 C

Class Celsius has more temperature !

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 3.344 s
Press any key to continue.
```

CONCLUSION:

When we define member functions outside the class, we need to use scope resolution operator whereas in the case of friend function we need not to use scope resolution operator when we define that particular function. Additionally, friend function can access the all data members of the class.

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

PROGRAM CODE:

```
#include<iostream>
using namespace std;
class Date
{
    private:
        int dd;
        int mm;
        int yyyy;
    public:
        void add_data();
        void display_data();
        friend void swapdates(Date& x, Date& y);
};

void Date::add_data()
{
    cout << "\nDate : ";
    cin >> dd;
    cout << "Month : ";
    cin >> mm;
    cout << "Year : ";
    cin >> yyyy;
}

void Date::display_data()
```

```
{
    cout << "The date(dd-mm-yyyy) : " << dd << "-" << mm << "-" << yyyy <<
endl;
}

void swapdates(Date& x, Date& y)
{
    int temp;
    temp = x.dd; x.dd = y.dd; y.dd = temp;
    temp = x.mm; x.mm = y.mm; y.mm = temp;
    temp = x.yyyy; x.yyyy = y.yyyy; y.yyyy = temp;
}

int main()
{
    Date d1,d2;
    cout << "Enter the required information..." << endl;
    d1.add_data();
    d2.add_data();

    cout << "\n*****Before swaping*****" << endl;
    d1.display_data();
    d2.display_data();

    cout << "\n*****After swaping*****" << endl;
    swapdates(d1,d2);
    d1.display_data();
    d2.display_data();

    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Enter the required information...

Date : 8
Month : 1
Year : 2002

Date : 16
Month : 4
Year : 2021

*****Before swaping*****
The date(dd-mm-yyyy) : 8-1-2002
The date(dd-mm-yyyy) : 16-4-2021

*****After swaping*****
The date(dd-mm-yyyy) : 16-4-2021
The date(dd-mm-yyyy) : 8-1-2002

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 12.150 s
Press any key to continue.
```

CONCLUSION:

I learnt to use quite useful concept of friend function which takes objects by reference.

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;

class TILES
{
    int l,w,area2;
public :
    void getdata_tiles();
    void putdata_tiles();
    void cal_area_tiles();
    void number_of_tiles(LAND&);
};

class LAND
{
    int length, width, area1;
```



```
public :
    void getdata_land();
    void putdata_land();
    void cal_area_land();
    friend void TILES::number_of_tiles(LAND&);
};

void TILES::number_of_tiles(LAND& temp)
{
    cout << "\nNumber of tiles required : "<< (temp.area1)/area2 << endl;
}

void LAND:: getdata_land()
{
    cout << "----- Enter the Information For Land -----" << endl;
    cout << "Enter the length : ";
    cin >> length;
    cout << "Enter the width : ";
    cin >> width;
}

void LAND:: putdata_land()
{
    cout << "\n**Entered Information**" << endl;
    cout << "Length = " << length << endl;
    cout << "Width = " << width << endl;
}

void LAND:: cal_area_land()
{

```

```
    area1 = (length*width);
    cout << "Area of land : " << area1 << endl;
}

void TILES:: getdata_tiles()
{
    cout << "\n----- Enter the Information For Tiles -----" << endl;
    cout << "Enter the length : ";
    cin >> l;
    cout << "Enter the width : ";
    cin >> w;
}

void TILES:: putdata_tiles()
{
    cout << "\n**Entered Information**" << endl;
    cout << "Length = " << l << endl;
    cout << "Width = " << w << endl;
}

void TILES:: cal_area_tiles()
{
    area2 = (l*w);
    cout << "Area of a single tile : " << area2 << endl;
}

int main()
{
    LAND c1;
```

```
c1.getdata_land();
```

```
c1.putdata_land();
```

```
c1.cal_area_land();
```

```
TILES t1;
```

```
t1.getdata_tiles();
```

```
t1.putdata_tiles();
```

```
t1.cal_area_tiles();
```

```
t1.number_of_tiles(c1);
```

```
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
```

```
}
```

OUTPUT:

```
----- Enter the Information For Land -----
Enter the length : 12
Enter the width : 12

**Entered Information**
Length = 12
Width = 12
Area of land : 144

----- Enter the Information For Tiles -----
Enter the length : 6
Enter the width : 6

**Entered Information**
Length = 6
Width = 6
Area of a single tile : 36

Number of tiles required : 4

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 2.706 s
Press any key to continue.
```

CONCLUSION:

I learnt the concept of member function of one class can be a friend function of another class from this program.

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 child's data and DisplayChildData() which takes child's object as argument and displays child's data. Use the concepts of Friend Class.

PROGRAM CODE:

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

class Child;

class Parent
{
public :
    void ReadChildData(Child& temp);
    void DisplayChildData(Child );
};

class Child
{
    char name[50];
    char gender[50];
public :
    friend class Parent;
};
```

```
void Parent::ReadChildData(Child& temp)    //takes object by reference
{
    cout << "Enter the name : ";
    cin >> temp.name;
    cout << "Enter the gender : ";
    cin >> temp.gender;
}

void Parent::DisplayChildData(Child temp) // takes object as argument
{
    cout << "\n***Entered Information***" << endl;
    cout << "Name : " << temp.name << endl;
    cout << "Gender : " << temp.gender << endl;
}

int main()
{
    Child c1,temp;
    Parent p1;
    p1.ReadChildData(c1);
    p1.DisplayChildData(c1);

    cout << "\n\n Made By: RUSHIK RATHOD \n    20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Enter the name : Rushik
Enter the gender : Male

***Entered Information***
Name : Rushik
Gender : Male

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 5.038 s
Press any key to continue.
_
```

CONCLUSION:

I learnt to create one class, a friend of another class which can access all the data of the friend class.

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:

1) Example of constant member functions

PROGRAM CODE:

// Example 1 Example of constant member functions

```
#include <iostream>
```

```
using namespace std;
```

```
class sample
```

```
{
```

```
    int m, n;
```

```
public:
```

```
    void getdata();
```

```
    void putdata() const;
```

```
};
```

```
void sample::getdata()
```

```
{
```

```
    cout<< "Enter m & n : " << endl;
```

```
    cin>>m>>n;
```

```
}
```

```
void sample::putdata() const //putdata() function is trying to change the data  
which is against the rule
```

```
{
```


//m=12; //By not letting this function to change the values, we can run this program

//n=34;

cout<< "m = " << m << endl << "n = " << n;

}

int main()

{

sample s1;

s1.getdata();

s1.putdata();

cout << "\n\n Made By: RUSHIK RATHOD \n 20DCS103 \n";

return 0;

}

OUTPUT:

```
Enter m & n :  
11  
12  
m = 11  
n = 12  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 6.136 s  
Press any key to continue.  
_
```

CONCLUSION:

Constant member function cannot change any data. So, changing the values violates the rules if the member function is defined/declared constant.

2-A) Pointer to data members

PROGRAM CODE:

// Example 2a Pointer to data members

```
#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;
```

```
    cout << "\n\n Made By: RUSHIK RATHOD \n        20DCS103 \n";
```

```
    return 0;
```

```
}
```

OUTPUT:

```
Roll no is 42
Roll no is 45

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
_
```

CONCLUSION:

In the above program there were some error, which is solved by the following rules of C++

The operator .* is used when the object itself is used with the member pointer.

The operator ->* is used to access a member when we use pointers to both the object and the member.

2-B) Pointer to member functions

PROGRAM CODE:

// Example 2b Pointer to member functions

```
#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)();
```

```
    cout << "\n\n Made By: RUSHIK RATHOD \n    20DCS103 \n";
```

```
    return 0;
```

```
}
```

OUTPUT:

```
Hi hello
Hi hello

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
_
```

CONCLUSION:

In the above program there were some error, which is solved by the following rules of C++

The operator .* is used when the object itself is used with the member pointer.

The operator ->* is used to access a member when we use pointers to both the object and the member.

3) Example of local classes

PROGRAM CODE:

// Example 3 Example of Local Classes

```
#include<iostream>
```

```
using namespace std;
```

```
void testlocalclass()
```

```
{
```

```
    class Test
```

```
    {
```

```
        int cnt;    // local classes can not have static data member
```

```
    public:
```

```
        void set()
```

```
        {
```

```
            cout<<"Enter Count: "; cin>>cnt;
```

```
        }
```

```
        void get() // member functions must be defined inside the class
```

```
        {
```

```
            cout<<"Count: = " <<cnt;
```

```
        }
```

```
};
```

```
Test t;
```

```
t.set();
```

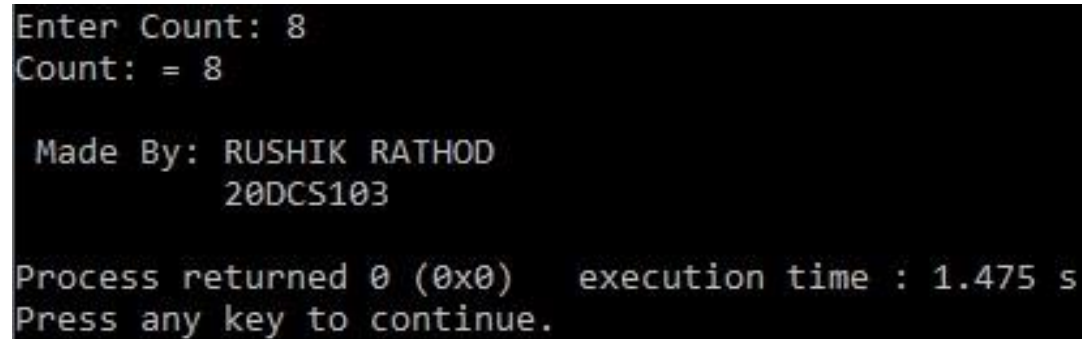
```
t.get();
```

```
}
```

```
int main()
```

```
{  
    testlocalclass();  
    cout << "\n\n Made By: RUSHIK RATHOD \n        20DCS103 \n";  
    return 0;  
}
```

OUTPUT:

A screenshot of a terminal window with a black background and white text. The text shows the program's execution: it prompts for a count, receives the value 8, and then prints the author's name and ID. It also shows the process returning 0 and the execution time.

```
Enter Count: 8  
Count: = 8  
  
Made By: RUSHIK RATHOD  
        20DCS103  
  
Process returned 0 (0x0)   execution time : 1.475 s  
Press any key to continue.
```

CONCLUSION:

In the concept of local classes, the member function must be define inside the class and it cannot have static data members.

23. Write a C++ program having class time with data members: hr, min and sec. Define following member functions.

- 1) getdata() to enter hour, minute and second values
- 2) putdata() to print the time in the format 11:59:59
- 3) default constructor
- 4) parameterized constructor
- 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 h,m,s;
public :
    void getdata()
    {
        cout << "\nEnter time in hh:mm:ss format - " << endl;
        cin >> h >> m >> s;
    }
    void putdata()
    {
        cout << "Answer = " << h << ":" << m << ":" << s;
```

```
    }  
    Time();           //Default Constructor  
    Time(int, int, int); //Parameterized Constructor  
    Time (Time &);     //Copy Constructor  
    ~Time()  
    {  
        cout << "\nDestructor" << endl;  
    }  
};  
Time::Time()  
{  
    cout << "From default constructor -> ";  
    h = 0;  
    m = 0;  
    s = 0;  
    cout << h << ":" << m << ":" << s << endl;  
}  
Time::Time(int x, int y, int z=52)  
{  
    h=x;  
    m=y;  
    s=z;  
    cout << endl << "From parameterized constructor -> " << h << ":" << m << ":" << s << endl;;  
}  
Time::Time(Time &t)  
{
```

```
    h=t.h;
    m=t.m;
    s=t.s;

    cout << endl << "\nFrom copy constructor -> " << h << ":" << m << ":" << s <<
endl;
}

int main()
{
    Time t1;          // Default constructor is called
    t1.putdata();

    int x,y,z;
    cout << endl << "Enter hour, minute and second respectively : " << endl;
    cin >> x >> y >> z;
    Time t2(x,y,z);    // Parameterized constructor
    t2.putdata();

    cout << "\n\nDefault value for second is 52 ";
    Time t3(x,y);
    t3.putdata();      // Default second 52

    Time t4(t2);        // Copy Constructor      //Time t4=t2
    t4.putdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
From default constructor -> 0:0:0
Answer = 0:0:0
Enter hour, minute and second respectively :
11
12
13

From parameterized constructor -> 11:12:13
Answer = 11:12:13

Default value for second is 52
From parameterized constructor -> 11:12:52
Answer = 11:12:52

From copy constructor -> 11:12:13
Answer = 11:12:13

  Made By: RUSHIK RATHOD
          20DCS103

Destructor
Destructor
Destructor
Destructor

Process returned 0 (0x0)   execution time : 14.646 s
Press any key to continue.
_
```

CONCLUSION:

Default constructor: when we don't pass any arguments to the constructor, it is called default constructor.

Parameterized constructor: when we pass arguments to the constructor when calling, then it is called parameterized constructor.

Copy constructor: when we pass arguments with the & operator, it is called copy constructor.

Constructor with default arguments: when we don't pass arguments but still want to get default values, then we use constructor with default arguments.

Destructor: when we want to destroy the constructor, we use ~ symbol to declare the destructor.

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. Use the concept of Overloading Unary Operators.

PROGRAM CODE:

```
#include <iostream>

using namespace std;

class Number
{
    int n;
public:
    Number() { n=0; }
    Number(int m) { n=m; }
    Number(const Number &temp) { n=temp.n; }
    void input()
    {
        cout << "Enter n2 : " << endl;
        cin >> n;
    }
    void output()
    {
        cout << " : " << n << endl;
    }
    Number operator++(int)
    {
```

```
    Number t1;
    t1.n=n++;
    return t1;
}
Number operator++()
{
    Number t2;
    t2.n=++n;
    return t2;
}
Number operator-()
{
    Number t3;
    t3.n=-n;
    return t3;
}
};
int main()
{
    Number n1,n2,n3;
    n2.input();

    n1=n2++;      // post n1=n2.operator++(int);
    cout << endl << "n1=n2++" << endl;
    cout << "n1";
    n1.output();
    cout << "n2";
```

```
n2.output();

n3=++n1;      // pre n3=n1.operator++();
cout << endl << "n3=++n1" << endl;
cout << "n1";
n1.output();
cout << "n3";
n3.output();

n3=-n3;      //n3=n3.operator-();
cout << endl << "n3=-n3" << endl;
cout << "n3";
n3.output();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```


OUTPUT:

```
Enter n2 :  
5  
  
n1=n2++  
n1 : 5  
n2 : 6  
  
n3=++n1  
n1 : 6  
n3 : 6  
  
n3=-n3  
n3 : -6  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 1.287 s  
Press any key to continue.  
_
```

CONCLUSION:

When we overload unary operator with the help of member functions, we need not to pass any arguments at the time of calling. We can define the member function outside the class by the syntax:

```
return_type operator op()  
{  
    // body part  
}
```

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. Use the concept of Overloading Unary Operators with friend function.

PROGRAM CODE:

```
#include <iostream>
using namespace std;
class Complex
{
    int real, img;
public :
    Complex() //Default constructor
    {
        real=0;
        img=0;
        cout << endl << "-----Default Constructor-----" << endl;
    }
    Complex(int x, int y) //Parameterized Constructor
    {
        real=x;
        img=y;
        cout << endl << "-----Parameterized Constructor-----" << endl;
    }
    Complex(Complex &c) //Copy Constructor
    {
        real=c.real;
```

```
        img=c.img;
        cout << endl << "-----Copy Constructor-----" << endl;
    }
    void putdata()
    {
        cout << endl << "real : " << real << endl << "img : " << img << endl;
    }
    void friend operator-(Complex r);
};
void operator-(Complex &r) //pass by reference
{
    r.real=-r.real;
    r.img=-r.img;
}
int main()
{
    Complex c;
    Complex c1(30,40);

    cout << endl << " : Before overloading : " << endl;
    c1.putdata();

    cout << endl << " : After overloading : " << endl;
    -c1;
    c1.putdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
```

```
}
```

OUTPUT:

```
-----Default Constructor-----  
-----Parameterized Constructor-----  
: Before overloading :  
real : 30  
img  : 40  
: After overloading :  
-----Copy Constructor-----  
real : 30  
img  : 40  
.  
Made By: RUSHIK RATHOD  
        20DCS103  
Process returned 0 (0x0)   execution time : 0.035 s  
Press any key to continue.  
_
```

CONCLUSION:

When we overload unary operator with the help of friend functions, we need to pass only one argument at the time of calling. We can define the friend function outside the class by the syntax:

```
return_type operator op(argument)
```

```
{
```

```
// body part
```

```
}
```

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 CLASS. Use the concept of overloading Binary Operators.

PROGRAM CODE:

```
#include<iostream>
#include<string.h>
#include<cstring>
using namespace std;
class String
{
    char str[20];
public:
    String() // Default constructor
    { }
    String(char str[])
    {
        strcpy(this->str,str);
    }
    void get()
    {
        cout << "\nEnter a string : ";
        cin >> str;
    }
    void display()
    {
        cout << "Entered string : " << str << endl;
```

```
    }  
    String operator +(String s2)  
    {  
        String s3;  
        strcat(str, " ");  
        strcat(str,s2.str);  
        strcpy(s3.str,str);  
        return s3;  
    }  
    String operator +=(String s2)  
    {  
        strcat(str, " ");  
        strcat(str,s2.str);  
        return str;  
    }  
    int operator==(String s2)  
    {  
        if(strcmp(str,"Rathod")==0)  
            return 1;  
        else  
            return 0;  
    }  
};  
int main()  
{  
    String s1,s3;  
    s1.get();
```

```
String s2("Rathod");
cout << "\nAfter ==(S1=S2) : " << endl;
if(s1==s2)
    cout << "Both strings are same." << endl;
else
    cout << "Both strings are different." << endl;

s3=s1+s2;
cout << "\nAfter +(S3=S1+S2) : " << endl;
s3.display();
s1+=s2;
cout << "\nAfter +=(S1+=S2) : " << endl;
s1.display();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:

```
Enter a string : Rushik

After ==(S1=S2) :
Both strings are different.

After +(S3=S1+S2) :
Entered string : Rushik Rathod

After +=(S1+=S2) :
Entered string : Rushik Rathod Rathod

Made By: RUSHIK RATHOD
20DCS103

Process returned 0 (0x0)   execution time : 2.956 s
Press any key to continue.
```

CONCLUSION:

When we overload binary operator with the help of member functions, we need to pass only one argument at the time of calling. We can define the member function outside the class by the syntax:

```
return_type operator op(argument)
{
// body part
}
```


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. Use the concept of Overloading Binary Operators with friend function.

PROGRAM CODE:

```
#include <iostream>
using namespace std;
class Measure
{
    float meter, cm;
public :
    void get()
    {
        cout << "Enter meter : ";
        cin >> meter;
        cout << "Enter centimeter : ";
        cin >> cm;
    }
    void put()
    {
        cout << "    Meter : " << meter << endl;
        cout << "Centimeters : " << cm << endl;
    }
    friend Measure operator+(Measure, int);
    friend Measure operator-(Measure, float);
    friend Measure operator+(float, Measure);
```

```
friend Measure operator-(float, Measure);  
};  
Measure operator+(Measure r, int a)  
{  
    Measure temp;  
    temp.meter = r.meter + a;  
    temp.cm = r.cm + a;  
    return(temp);  
}  
Measure operator-(Measure r, float a)  
{  
    Measure temp;  
    temp.meter = r.meter - a;  
    temp.cm = r.cm - a;  
    return(temp);  
}  
Measure operator+(float a, Measure r)  
{  
    Measure temp;  
    temp.meter = a + r.meter;  
    temp.cm = a + r.cm;  
    return(temp);  
}  
Measure operator-(float a, Measure r)  
{  
    Measure temp;  
    temp.meter = a - r.meter;
```

```
temp.cm = a - r.cm;
return(temp);
}

int main()
{
    Measure m1,m2,m3,m4;
    cout << "Enter values for m2 : " << endl;
    m2.get();
    cout << endl;
    m2.put();

    cout << endl << "Enter values for m4 : " << endl;
    m4.get();
    cout << endl;
    m4.put();

    cout << endl << "\n*****Values After Manipulating*****" << endl;
    m1 = m2 + 15;
    cout << endl << "\n-----: m1 = m2 + 15 :-----" << endl;
    cout << "    m2" << endl;
    m2.put();
    cout << "    m1" << endl;
    m1.put();

    m3 = m1 - 4.5;
    cout << endl << "\n-----: m3 = m1 - 4.5 :-----" << endl;
```

```
cout << "  m1" << endl;
m1.put();
cout << "  m3" << endl;
m3.put();

m1 = 5.0 + m2;
cout << endl << "\n-----: m1 = 5.0 + m2 :-----" << endl;
cout << "  m2" << endl;
m2.put();
cout << "  m1" << endl;
m1.put();

m3 = 2.0 - m4;
cout << endl << "\n-----: m3 = 2.0 - m4 :-----" << endl;
cout << "  m4" << endl;
m4.put();
cout << "  m3" << endl;
m3.put();

cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:

```
Enter values for m2 :
Enter meter : 3
Enter centimeter : 4

    Meter : 3
Centimeters : 4

Enter values for m4 :
Enter meter : 5
Enter centimeter : 6

    Meter : 5
Centimeters : 6

*****Values After Manipulating*****

-----: m1 = m2 + 15 :-----
    m2
        Meter : 3
Centimeters : 4
    m1
        Meter : 18
Centimeters : 19

-----: m3 = m1 - 4.5 :-----
    m1
        Meter : 18
Centimeters : 19
    m3
        Meter : 13.5
Centimeters : 14.5
```

```
-----: m1 = 5.0 + m2 :-----
    m2
    Meter : 3
Centimeters : 4
    m1
    Meter : 8
Centimeters : 9

-----: m3 = 2.0 - m4 :-----
    m4
    Meter : 5
Centimeters : 6
    m3
    Meter : -3
Centimeters : -4

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 4.289 s
Press any key to continue.
```

CONCLUSION:

When we overload binary operator with the help of friend functions, we need to pass two arguments at the time of calling. We can define the friend function outside by the syntax:

```
return_type operator op(argument1, argument2)
{
// body part
}
```

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 c;
public:
    Celsius() // default constructor
    {
        c=0; // default value
    }

    Celsius(float k) // parameterized constructor
    {
        c=k;
    }

    operator float()
    {
        return(c);
    }
}
```

```
void showdata()
{
    cout << c << endl;
}

};

int main()
{
    Celsius c1,c2;
    float temperature;
    c1=30.5;
    temperature=c2;
    cout << endl << " Output temperature for C1 : ";
    c1.showdata();
    cout << endl << " Output temperature for C2 : ";
    c2.showdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```


OUTPUT:

```
Output temperature for C1 : 30.5  
Output temperature for C2 : 0  
  
Made By: RUSHIK RATHOD  
20DCS103  
  
Process returned 0 (0x0)   execution time : 0.996 s  
Press any key to continue.  
_
```

CONCLUSION:

When we the concept of type conversion from basic type to class type, we need to use constructor and in the conversion of class type to basic type we need to use operator keyword.

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>

using namespace std;

class Fahrenheit;

class Celsius
{
public:
    float ct;

    Celsius()          //default constructor
    {
        ct=0;
    }

    Celsius(float temp) //parameterized constructor
    {
        ct = temp;
    }

    void getdata()
    {
        cout << "Enter temperature : ";
        cin >> ct;
    }
}
```

```
void putdata()
{
    cout << "Temperature : " << ct << endl;
}

Celsius(const Fahrenheit &f);

};

class Fahrenheit
{
public:
    float ft;
    Fahrenheit()          // default constructor
    {
        ft = 0;
    }
    Fahrenheit(float temp)
    {
        ft = temp;
    }
    Fahrenheit(const Celsius &c)
    {
        ft=c.ct;
    }
    void putdata()
    {
        cout << "Temperature : " << ft << endl;
    }
}
```

```
void getdata()
{
    cout << "Enter temperature : ";
    cin >> ft;
}
};
```

```
Celsius::Celsius(const Fahrenheit &f)
{
    ct=f.ft;
}
```

```
int main()
{
    Celsius c1, c2 = 5.0;
    c2.putdata();
    Fahrenheit f1, f2(13.2);
    f1 = c2;           // assign f1 5.0 i.e c2
    f1.putdata();
    c1 = f2;           //assign c1 13.2 i.e f2
    c1.putdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Temperature : 5
Temperature : 5
Temperature : 13.2

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

CONCLUSION:

We can do type conversion from one class type to another class type in the two different methods: 1) by using the concept of constructor and 2) with the help of operator keyword as show in the above program.

30. Define a Base Class Vegetable having data member Color and member function getdata() which takes color as an input and putdata() which prints 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 the color : ";
        cin >> color;
    }
    void putdata()
    {
        cout << endl << "Color : " << color << endl;
    }
};
class Tomato : public Vegetable
{
    float weight;
```

```
float size;
public :
void input()
{
    cout << endl << "----- Enter The Information -----" << endl;
    getdata();
    cout << "Enter weight : ";
    cin >> weight;
    cout << "Enter size : ";
    cin >> size;
}
void output()
{
    putdata();
    cout << "Weight : " << weight << endl;
    cout << "Size  : " << size << endl;
}
};
int main()
{
    Tomato t1;
    t1.input();
    cout << endl << "----- Calling By The Object of Tomato -----" << endl;
    t1.output();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
----- Enter The Information -----  
Enter the color : Red  
Enter weight : 15.6  
Enter size : 10.8  
  
----- Calling By The Object of Tomato -----  
  
Color   : Red  
Weight  : 15.6  
Size    : 10.8  
  
Made By: RUSHIK RATHOD  
        20DCS103  
  
Process returned 0 (0x0)   execution time : 14.819 s  
Press any key to continue.  
_
```

CONCLUSION:

When we derive only one class from only one base class, then it is called single inheritance. In the single inheritance, derived class can access the properties and member functions of base class if it is declared publicly in the base class.

So, we need not to write the same functions again and again in the derived class.

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 medicine_type;
    string company_name;
    string manu_date;
public :
    void getdata()
    {
        cout << "Enter medicine type : ";
        cin >> medicine_type;
        cout << "Enter company name : ";
        cin >> company_name;
        cout << "Enter manufacturing date : ";
        cin >> manu_date;
    }
    void putdata()
    {
```

```
        cout << "Medicine type      : " << medicine_type << endl;
        cout << "Company name      : " << company_name << endl;
        cout << "Manufacturing date : " << manu_date << endl;
    }
};

class Tablet : public Medicine
{
    string tab_name;
    int tab_quantity;
    float tab_price;
public :
    void input()
    {
        cout << endl << "----- Enter The Information About Tablet -----" <<
endl;
        getdata(); // CALLING THE FUNCTION OF THE BASE CLASS
        cout << "Enter the name of tablet : ";
        cin >> tab_name;
        cout << "Enter the quantity per pack : ";
        cin >> tab_quantity;
        cout << "Enter the price of one tablet : ";
        cin >> tab_price;
    }
    void output()
    {
```

```
    cout << endl << "\n----- Entered Information About Tablet -----" <<
endl;
```

```
    putdata(); // CALLING THE FUNCTION OF THE BASE CLASS
```

```
    cout << "Name of tablet    : " << tab_name << endl;
```

```
    cout << "Quantity per pack  : " << tab_quantity << endl;
```

```
    cout << "Price of one tablet : " << tab_price << endl;
```

```
    }
```

```
};
```

```
class Syrup : public Medicine
```

```
{
```

```
    int syrup_quantity;
```

```
    int syrup_dasage;
```

```
public :
```

```
    void gdata()
```

```
    {
```

```
        cout << endl << "\n----- Enter The Information About Syrup -----" <<
endl;
```

```
        getdata();
```

```
        cout << "Enter the quantity per bottle : ";
```

```
        cin >> syrup_quantity;
```

```
        cout << "Enter the dosage unit : ";
```

```
        cin >> syrup_dasage;
```

```
    }
```

```
    void pdata()
```

```
    {
```

```
        cout << endl << "\n----- Entered Information About Syrup -----" << endl;
```

```
        putdata();
        cout << "Quantity per bottle : " << syrup_quantity << endl;
        cout << "Dosage unit      : " << syrup_dasage << endl;
    }
};

int main()
{
    Tablet t1;
    t1.input();
    t1.output();
    Syrup s1;
    s1.gdata();
    s1.pdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
----- Enter The Information About Tablet -----
Enter medicine type : abc
Enter company name : xyz
Enter manufacturing date : 25/05/2021
Enter the name of tablet : pqr
Enter the quantity per pack : 150
Enter the price of one tablet : 400

----- Entered Information About Tablet -----
Medicine type      : abc
Company name       : xyz
Manufacturing date : 25/05/2021
Name of tablet     : pqr
Quantity per pack  : 150
Price of one tablet : 400

----- Enter The Information About Syrup -----
Enter medicine type : abc2
Enter company name  : xyz2
Enter manufacturing date : 20/05/2021
Enter the quantity per bottle : 50
Enter the dosage unit : 5

----- Entered Information About Syrup -----
Medicine type      : abc2
Company name       : xyz2
Manufacturing date : 20/05/2021
Quantity per bottle : 50
Dosage unit        : 5

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 83.915 s
Press any key to continue.
```

CONCLUSION:

When we derive more than one class from only one base class, then it is called hierarchical inheritance. In this inheritance, derived class can access the properties and member functions of the base class if it is declared publicly in the base class.

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.

PROGRAM CODE:

```
#include <iostream>

using namespace std;

class Alpha
{
    int x;
public:
    Alpha(int a)
    {
        x=a;
    }
    void printx()
    {
        cout << "x : " << x << endl;
```

```
    }  
};  
  
class Bit  
{  
    float y;  
public:  
    Bit(float b)  
    {  
        y=b;  
    }  
    void printy()  
    {  
        cout << "y : " << y << endl;  
    }  
};  
  
class Gamma:public Alpha, public Bit  
{  
    int m,n;  
public :  
    Gamma(int p, float q, int r, int s):Alpha(p), Bit(q)  
    {  
        m=r;  
    }  
};
```

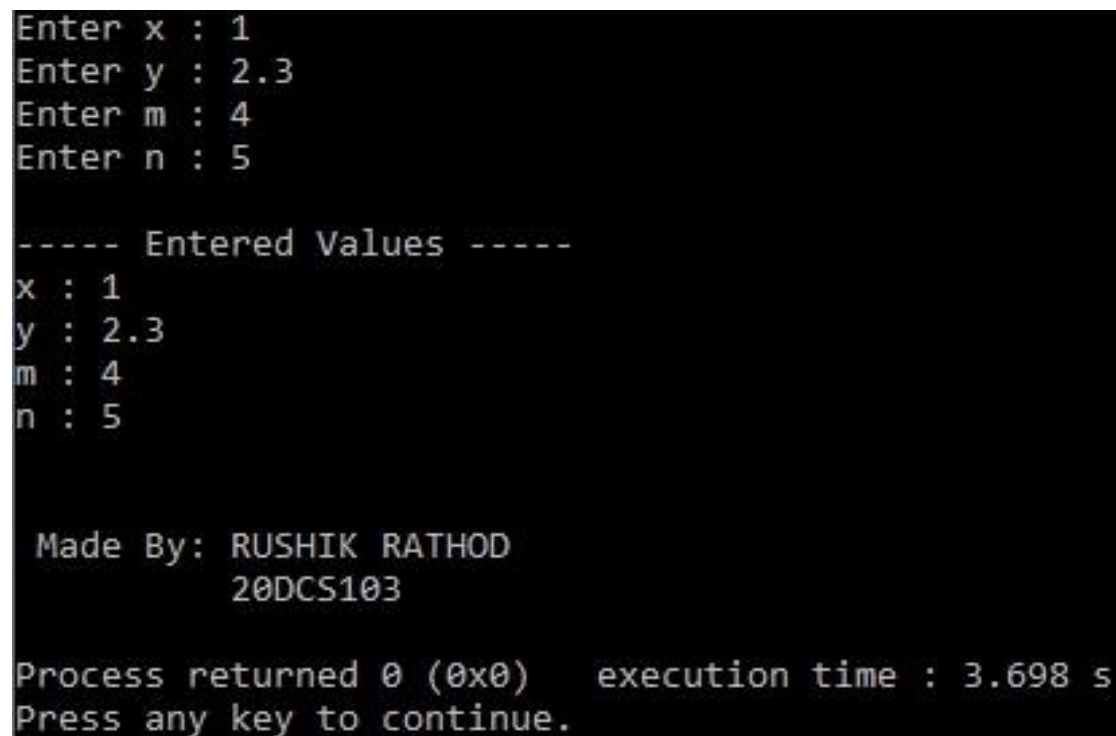
```
        n=s;
    }
    void printmn()
    {
        printx();
        printy();
        cout << "m : " << m << endl;
        cout << "n : " << n << endl;
    }
};
int main()
{
    int n1,n3,n4;
    float n2;
    cout << "Enter x : ";
    cin >> n1;
    cout << "Enter y : ";
    cin >> n2;
    cout << "Enter m : ";
    cin >> n3;
    cout << "Enter n : ";
    cin >> n4;

    Gamma g1(n1,n2,n3,n4);
```



```
cout << endl << "----- Entered Values -----" << endl;
g1.printmn();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:



```
Enter x : 1
Enter y : 2.3
Enter m : 4
Enter n : 5

----- Entered Values -----
x : 1
y : 2.3
m : 4
n : 5

Made By: RUSHIK RATHOD
      20DCS103

Process returned 0 (0x0)   execution time : 3.698 s
Press any key to continue.
```

CONCLUSION:

When we derive only one class from more than one base class, then it is called multiple inheritance. In this inheritance, derived class can access the properties and member functions of both of the base classes when it is declared publicly in the base classes. Further, we use the concept of constructor in the derived classe in the above program.

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
{
    int roll_no;
    char name[50];
public :
    void geth()
    {
        cout << "Roll number : ";
        cin >> roll_no;
        cout << "Name : ";
        cin >> name;
    }
    void puth()
    {
        cout << "Roll number    : " << roll_no << endl;
        cout << "Name          : " << name << endl;
    }
}
```

```
};

class Ward:public virtual Hospital
{
    int ward_no;
public :
    void getw()
    {
        cout << "Ward number : ";
        cin >> ward_no;
    }
    void putw()
    {
        cout << "Ward number      : " << ward_no << endl;
    }
};
```

```
class Room:public virtual Hospital
{
    int bed_no;
    char nature[50];
public :
    void getr()
    {
        cout << "Bed number : ";
        cin >> bed_no;
        cout << "Nature of illness : ";
```

```
        cin >> nature;
    }
    void putr()
    {
        cout << "Bed number      : " << bed_no << endl;
        cout << "Nature of illness : " << nature << endl;
    }
};
```

```
class Patient:public Ward, public Room
```

```
{
public :
    void getdata()
    {
        geth();
        getw();
        getr();
    }
    void display()
    {
        puth();
        putw();
        putr();
    }
};
```

```
int main()
{
    Patient p[5];
    int x=1;
    for(int i=0;i<=4;i++)
    {
        cout << endl << "\tPATIENT : " << x << endl;
        p[i].getdata();
        x++;
    }
    cout << endl << endl << "----- Entered Information -----" << endl;
    int y=1;
    for(int i=0;i<=4;i++)
    {
        cout << endl << "\tPATIENT : " << y << endl;
        p[i].display();
        y++;
    }
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
PATIENT : 1
Roll number : 1
Name : Doraemon
Ward number : 3
Bed number : 5
Nature of illness : Fever

PATIENT : 2
Roll number : 2
Name : Nobita
Ward number : 2
Bed number : 4
Nature of illness : Cough

PATIENT : 3
Roll number : 3
Name : Jaggu
Ward number : 6
Bed number : 8
Nature of illness : Weakness

PATIENT : 4
Roll number : 4
Name : Hathori
Ward number : 8
Bed number : 12
Nature of illness : Fever

PATIENT : 5
Roll number : 5
Name : Jiyaan
Ward number : 9
Bed number : 11
Nature of illness : Weakness

----- Entered Information -----
```

```
----- Entered Information -----  
  
      PATIENT : 1  
Roll number      : 1  
Name             : Doraemon  
Ward number      : 3  
Bed number       : 5  
Nature of illness : Fever  
  
      PATIENT : 2  
Roll number      : 2  
Name             : Nobita  
Ward number      : 2  
Bed number       : 4  
Nature of illness : Cough  
  
      PATIENT : 3  
Roll number      : 3  
Name             : Jaggu  
Ward number      : 6  
Bed number       : 8  
Nature of illness : Weakness  
  
      PATIENT : 4  
Roll number      : 4  
Name             : Hathori  
Ward number      : 8  
Bed number       : 12  
Nature of illness : Fever  
  
      PATIENT : 5  
Roll number      : 5  
Name             : Jiyaan  
Ward number      : 9  
Bed number       : 11  
Nature of illness : Weakness  
  
Made By: RUSHIK RATHOD  
        20DCS103  
  
Process returned 0 (0x0)  execution time : 103.473 s  
Press any key to continue.
```

CONCLUSION:

The process of applying two or more types of inheritance to design a program is called hybrid inheritance. When all three types of inheritance, namely multilevel, multiple and hierarchical inheritance are involved, then we need to use virtual keyword to make classes virtual to avoid ambiguity in the program.

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
{
    string shape_name;
public:
    void s_get()
    {
        cout << endl << "Enter the name of shape : ";
        cin >> shape_name;
    }
    void s_put()
    {
        cout << endl << "Shape : " << shape_name << endl;
    }
};

class Circle:public Shape
```

```
{
protected:
    float radius_circle;
public:
    void c_get()
    {
        s_get();
        cout << "Enter the radius of a circle : ";
        cin >> radius_circle;
    }
    void c_put()
    {
        s_put();
        cout << "Radius : " << radius_circle << endl;
    }
};

class Area:public Circle
{
    float area_of_circle;
public :
    void display()
    {
        area_of_circle = (3.14*(radius_circle*radius_circle));
        cout << endl << " ---> The area of a circle is : " << area_of_circle << endl;
    }
};
```

```
int main()
{
    Area a1;
    a1.c_get();
    cout << endl << "----- Entered Information -----" << endl;
    a1.c_put();
    a1.display();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Enter the name of shape : Circle
Enter the radius of a circle : 25

----- Entered Information -----

Shape : Circle
Radius : 25

---> The area of a circle is : 1962.5

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 6.294 s
Press any key to continue.
```

CONCLUSION:

The process of deriving a class from another derived class is called multilevel inheritance. When we want to use the already defined functions in the base classes we use the concept of multilevel inheritance.

35. Create one application in a group of 3 persons which implements all types of inheritance.

PROGRAM CODE:

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

class Social_Media
{
protected :
    string name;
public :
    void getname()
    {
        cout << "Enter your name : ";
        cin >> name;
    }
};

class Facebook:virtual public Social_Media
{
protected :
    int f_friends;
    int f_posts;
    string f_username;
public :
    void get_facebook()
```

```
{
    cout << "Enter your username : ";
    cin >> f_username;
    cout << "Enter number of friends : ";
    cin >> f_friends;
    cout << "Enter number of posts : ";
    cin >> f_posts;
}
};

class Instagram:virtual public Social_Media
{
protected :
    int i_followers;
    int i_posts;
    string i_username;
public :
    void get_insta()
    {
        cout << "Enter your username : ";
        cin >> i_username;
        cout << "Enter number of followers : ";
        cin >> i_followers;
        cout << "Enter number of posts : ";
        cin >> i_posts;
    }
};
```

```
class WhatsApp:public Social_Media
{
protected :
    int w_year;
    int w_contacts;
public :
    void get_whatsapp()
    {
        cout << "Using WhatsApp since : ";
        cin >> w_year;
        cout << "Enter number of contacts : ";
        cin >> w_contacts;
    }
    void put_whatsapp()
    {
        cout << endl << "\tWhatsApp Info" << endl;
        cout << "You are using WhatsApp since : " << w_year << endl;
        cout << "Number of contacts      : " << w_contacts << endl;
    }
};
```

```
class Information: public Facebook, public Instagram
{
public :
    void max_posts()
    {
```

```
    if(i_posts>f_posts)
        cout << endl << "---> Great! You have more posts on Instagram !" << endl;
    else if (i_posts<f_posts)
        cout << endl << "---> Great! You have more posts on Facebook !" << endl;
    else
        cout << endl << "---> Great! You have posted same number of posts on
both the Facebook and Instagram !" << endl;
}
void max_fan()
{
    if(i_followers>f_friends)
        cout << endl << "---> Congratulations! You have more fan following on
Instagram !" << endl;
    else if(i_followers<f_friends)
        cout << endl << "---> Congratulations! You have more fan following on
Facebook !" << endl;
    else
        cout << endl << "---> Congratulations! You are popular on both the
Facebook & Instagram !" << endl;
}
void getdata()
{
    getname();
    cout << endl << "\tFacebook" << endl;
    get_facebook();
    cout << endl << "\tInstagram" << endl;
    get_insta();
}
```



```
void putdata()
{
    cout << "\tFacebook Info" << endl;
    cout << "Your username      : " << f_username << endl;
    cout << "Number of friends   : " << f_friends << endl;
    cout << "Number of posts      : " << f_posts << endl;
    cout << endl << "\tInstagram Info" << endl;
    cout << "Your username      : " << i_username << endl;
    cout << "Number of followers : " << i_followers << endl;
    cout << "Number of posts      : " << i_posts << endl;
}

};

int main()
{
    Information a1;
    a1.getdata();
    cout << endl << "\tWhatsApp" << endl;
    WhatsApp w1;
    w1.get_whatsapp();
    cout << endl << "-----" << endl <<
endl;
    a1.putdata();
    w1.put_whatsapp();
    a1.max_posts();
    a1.max_fan();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
```

```
    return 0;
}
```

OUTPUT:

```
Enter your name : Rushik

    Facebook
Enter your username : Rushik_Rathod
Enter number of friends : 100
Enter number of posts : 20

    Instagram
Enter your username : rushik_rathod_
Enter number of followers : 400
Enter number of posts : 10

    WhatsApp
Using WhatsApp since : 2017
Enter number of contacts : 500

-----

    Facebook Info
Your username      : Rushik_Rathod
Number of friends  : 100
Number of posts    : 20

    Instagram Info
Your username      : rushik_rathod_
Number of followers : 400
Number of posts    : 10

    WhatsApp Info
You are using WhatsApp since : 2017
Number of contacts           : 500

---> Great! You have more posts on Facebook !

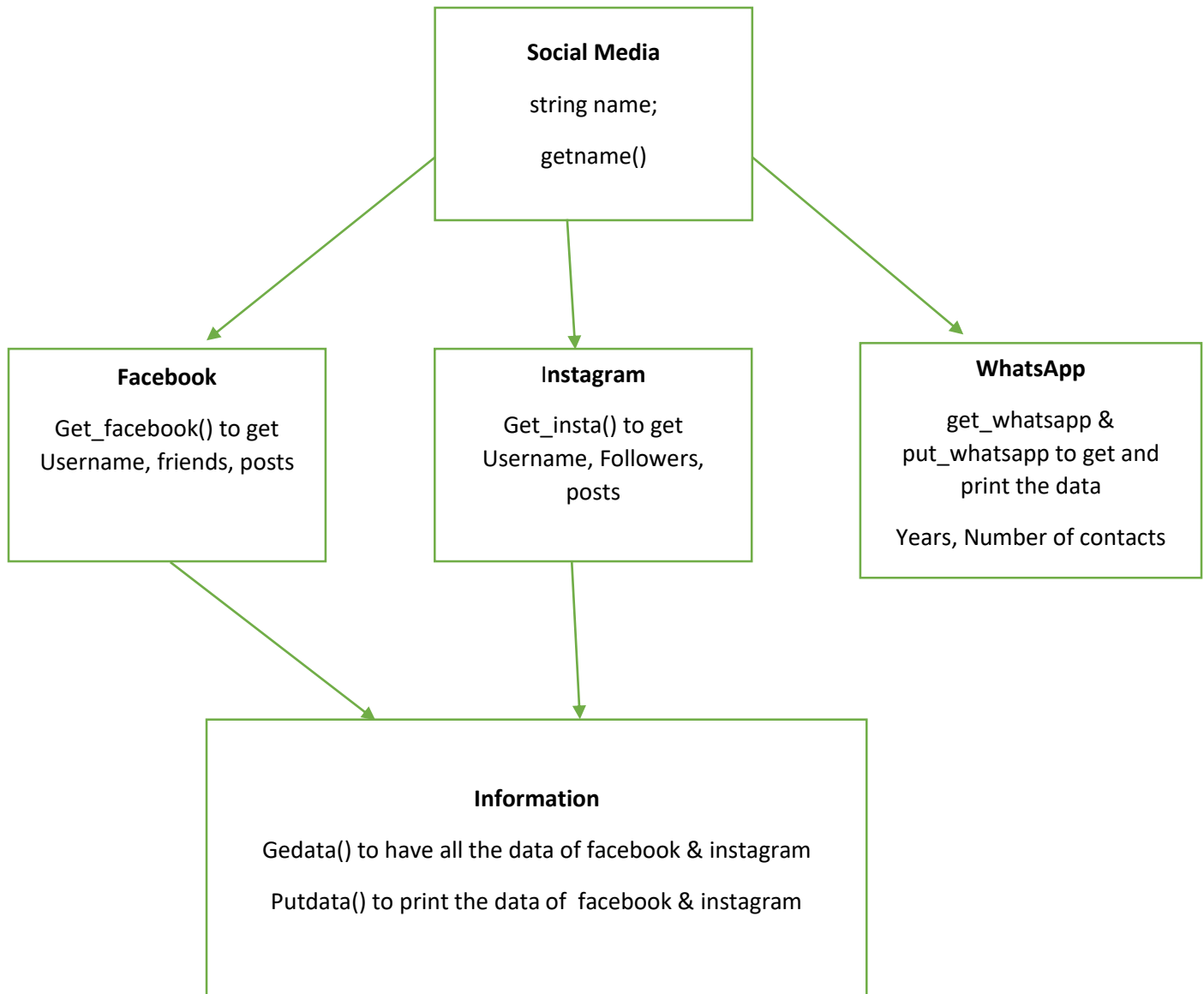
---> Congratulations! You have more fan following on Instagram !

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 30.131 s
Press any key to continue.
_
```

CONCLUSION:

In the above program I have used all types of inheritance which is demonstrated in this graph:



36. What is the output of the following code:

(A) Pointer to object :

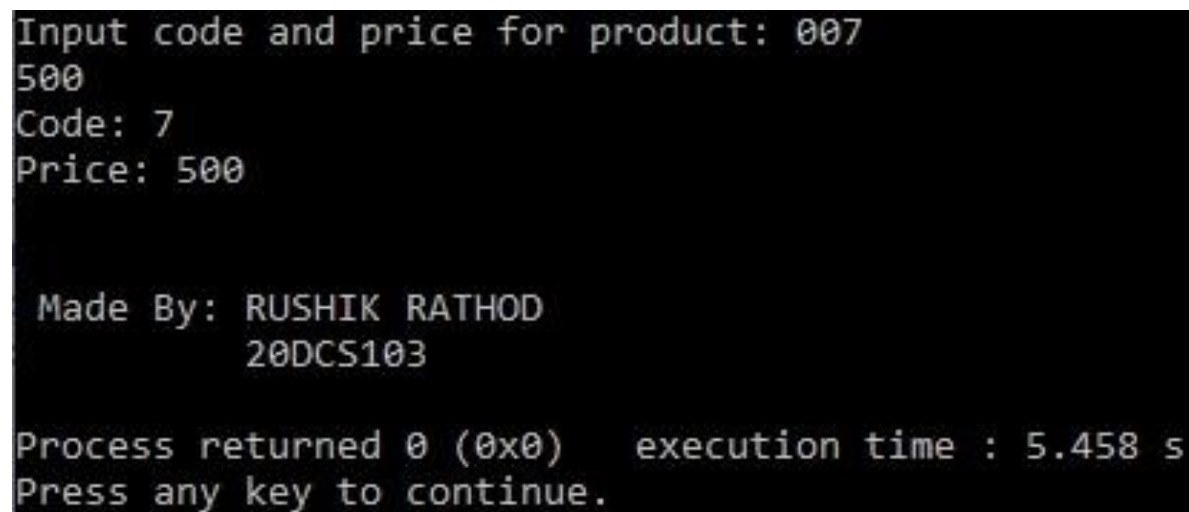
PROGRAM CODE:

```
#include<iostream>
using namespace std;
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();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
}
```

OUTPUT:

A screenshot of a terminal window showing the output of a C++ program. The text is as follows:
Input code and price for product: 007
500
Code: 7
Price: 500

Made By: RUSHIK RATHOD
20DCS103

Process returned 0 (0x0) execution time : 5.458 s
Press any key to continue.
_

CONCLUSION:

When we use the concept of pointer to object, we can call the member functions by the -> operator.

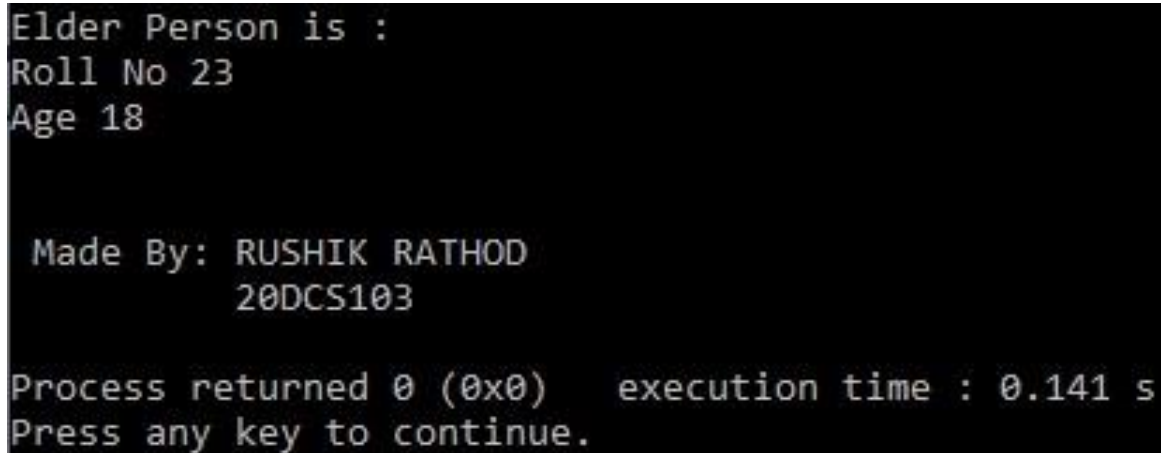
(B) this pointer :

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);    // s1 is calling function
    cout<<"Elder Person is :"<<endl;
    s.display();
    cout << "\n\n Made By: RUSHIK RATHOD \n    20DCS103 \n";
}
```

OUTPUT:

A screenshot of a terminal window showing the output of a C++ program. The output consists of three lines: "Elder Person is :", "Roll No 23", and "Age 18". There is a blank line, followed by "Made By: RUSHIK RATHOD" and "20DCS103" on the next line. Another blank line follows, then "Process returned 0 (0x0) execution time : 0.141 s" and "Press any key to continue." on the final line. A cursor is visible at the end of the last line.

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

Made By: RUSHIK RATHOD
20DCS103

Process returned 0 (0x0) execution time : 0.141 s
Press any key to continue.
```

CONCLUSION:

this pointer is used to invoke member function with the help of current object.

(C)Pointer 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;    // base class pointer
    BC base;     // base class object
```



```
bptr = &base;
bptr->b = 100;
cout<<"bptr points to base objects"<<endl;
bptr->show();    // it will call the method of which class it is made
DC derived;     // derived class pointer
bptr = &derived;
bptr->b = 200;
    // /*bptr->b = 300;*/ // won't 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();
cout << "\n\n Made By: RUSHIK RATHOD \n    20DCS103 \n";
return 0;
}
```

OUTPUT:

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

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

CONCLUSION:

In the concept of pointer to derived objects, when we call the member function with help of pointer, it will give first preference to the class of which it is made a pointer.

37. Create a class Media that stores the title (a string) and price (float). Class Media has 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.

PROGRAM CODE:

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

class Media
{
protected:
    string title;
    float price;
public:
    Media() // default constructor
    {}
    Media(string s, float p) // parameterized constructor
    {
        title=s;
        price=p;
```

```
    }  
    virtual void display() // virtual function  
    {  
        cout << "Title : " << title << endl;  
        cout << "Price : " << price << endl;  
    }  
};
```

```
class Book:public Media  
{  
    int page;  
public:  
    Book(string s, float p, int a):Media(s,p)  
    {  
        page=a;  
    }  
    void display()  
    {  
        cout << "Title : " << title << endl;  
        cout << "Price : " << price << endl;  
        cout << "Pages : " << page << endl;  
    }  
};
```

```
class Tape:public Media  
{  
    float time;
```

public:

```
Tape(string s, float p, int b):Media(s,p)
{
    time=b;
}
void display()
{
    cout << "Title : " << title << endl;
    cout << "Price : " << price << endl;
    cout << "Time : " << time << " minutes" << endl;
}
};
```

```
int main()
{
    string book_title,tape_title;
    float book_price, tape_price, time;
    int page;
    cout << endl << "Enter Book Details..." << endl;
    cout << "Enter title : ";
    cin >> book_title;
    cout << "Enter price : ";
    cin >> book_price;
    cout << "No. of pages : ";
    cin >> page;

    cout << endl << "Enter Tape Details..." << endl;
```

```
cout << "Enter title : ";
cin >> tape_title;
cout << "Enter price : ";
cin >> tape_price;
cout << "Enter playing time (minutes) : ";
cin >> time;

Book book1(book_title, book_price, page);
Tape tape1(tape_title, tape_price, time);

Media *m1, *m2;

m1=&book1;
m2=&tape1;
cout << endl << "----- Entered Details -----" << endl;
cout << endl << "\t: Book : " << endl;
m1->display();
cout << endl << "\t: Tape : " << endl;
m2->display();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
return 0;
}
```

OUTPUT:

```
Enter Book Details...
Enter title : abc
Enter price : 100
No. of pages : 250

Enter Tape Details...
Enter title : xyz
Enter price : 200
Enter playing time (minutes) : 56

----- Entered Details -----

      : Book :
Title : abc
Price : 100
Pages : 250

      : Tape :
Title : xyz
Price : 200
Time  : 56 minutes

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 23.077 s
Press any key to continue.
```

CONCLUSION:

When we call the member function with help of pointer of a particular class, it will give first preference to the class of which it is made a pointer. If that class has normal member function then it will invoke that else if it has virtual function then it will invoke the derived class function.

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
{
protected:
    float average;
public :
    virtual void getdata(){} //pure virtual function
    virtual void putdata(){} //pure virtual function
};

class Car:public Vehicle
{
    string car_fuel;
    int car_wheels;
public :
    void getdata()
    {
        cout << "Enter The Information for Car..." << endl;
```



```
        cout << endl << "Fuel type : ";
        cin >> car_fuel;
        cout << "No. of wheels : ";
        cin >> car_wheels;
    }
    void putdata()
    {
        cout << "Fuel type    : " << car_fuel << endl;
        cout << "No. of wheels : " << car_wheels << endl;
    }
};

class Truck:public Vehicle
{
    string truck_fuel;
    int truck_wheels;
public:
    void getdata()
    {
        cout << endl << "Enter The Information for Truck..." << endl;
        cout << endl << "Fuel type : ";
        cin >> truck_fuel;
        cout << "No. of wheels : ";
        cin >> truck_wheels;
    }
    void putdata()
    {
```

```
        cout << "Fuel type    : " << truck_fuel << endl;
        cout << "No. of wheels : " << truck_wheels << endl;
    }
};

int main()
{
    Vehicle *v1,*v2;
    Car c1;
    Truck t1;
    v1=&c1;
    v2=&t1;

    v1->getdata();
    v2->getdata();
    cout << endl << "-----" << endl;
    cout << "\n----- Entered Information of Car -----" << endl;
    v1->putdata();
    cout << "\n----- Entered Information of Truck -----" << endl;
    v2->putdata();
    cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
    return 0;
}
```

OUTPUT:

```
Enter The Information for Car...

Fuel type : Petrol
No. of wheels : 4

Enter The Information for Truck...

Fuel type : Diesel
No. of wheels : 8

-----

----- Entered Information of Car -----
Fuel type      : Petrol
No. of wheels  : 4

----- Entered Information of Truck -----
Fuel type      : Diesel
No. of wheels  : 8

Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 20.652 s
Press any key to continue.
```

CONCLUSION:

If a class contains at least one pure virtual function, then it is called an abstract class. We can define pure virtual function as follows: virtual void func()=0;

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().

PROGRAM CODE:

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

int main()
{
    ofstream fp("2cse3a.txt"); //constructor
    //fstream fp;
    //fp.open("xyz.txt",ios::out); //open function
    if(fp.good())
    {
        for(char i=65;i<91;i++)
        {
            fp.put(i);
        }
    }

    fp.close();
```

```
//fp.open("xyz.txt",ios::in);
ifstream ip("2cse3a.txt");
cout<<"Opening the file in reading mode...";

if(!ip.bad() && !ip.fail() && !ip.eof())
{
    cout << "\nReading from the file : ";
    ip.seekg(-1,ios::end);
    cout << "\nFP : " << ip.tellg() << endl;
    char ch;
    while(ip.tellg()>=0)
    {
        if(!ip.eof())
        {
            ip.get(ch);
            cout << ch; //z
            ip.seekg(-2,ios::cur);
        }
    }
}
ip.close();
cout << "\n\n Made By: RUSHIK RATHOD \n      20DCS103 \n";
}
```

OUTPUT:

```
Opening the file in reading mode...
Reading from the file :
FP : 25
ZYXWUTSRQPONMLKJIHGFEDCBA

  Made By: RUSHIK RATHOD
        20DCS103

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

CONCLUSION:

From this program, I learnt about handling the files and the proper uses of the functions manipulators for pointers like seekg(), tellg(). Additionally, I learnt the file error handling functions such as eof(), fail(), bad(), good().

Thank you...