|  |  |
| --- | --- |
| **Ex. No.: 5** | **Date: 07**/04/2022 |
| 5. Write a program explaining the concept of friend function and friend class. | |
| **Aim:**  To implement the concept of friend function and friend class. | |
| a)// C++ program to demonstrate the working of friend function  #include <iostream>  using namespace std;  class Distance {  private:  int meter;    // friend function  friend int addFive(Distance);  public:  Distance() : meter(0) {}    };  // friend function definition  int addFive(Distance d) {  //accessing private members from the friend function  d.meter += 5;  return d.meter;  }  int main() {  Distance D;  cout << "Distance: " << addFive(D);  return 0;  }  b)USE OF FRIEND CLASS  // C++ program to demonstrate the working of friend class  #include <iostream>  using namespace std;  // forward declaration  class ClassB;  class ClassA {  private:  int numA;  // friend class declaration  friend class ClassB;  public:  // constructor to initialize numA to 12  ClassA() : numA(12) {}  };  class ClassB {  private:  int numB;  public:  // constructor to initialize numB to 1  ClassB() : numB(1) {}    // member function to add numA  // from ClassA and numB from ClassB  int add() {  ClassA objectA;  return objectA.numA + numB;  }  };  int main() {  ClassB objectB;  cout << "Sum: " << objectB.add();  return 0;  } | |
| **Output:**  **a)**    **b)** | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 6** | **Date: 07**/04/2022 |
| 6. Write a program to understand the array of objects. | |
| **Aim:**  To implement the concept of the array of objects. | |
| #include <iostream>  using namespace std;  int main() {  int numbers[5] = {7, 5, 6, 12, 35};  cout << "The numbers are: ";  // Printing array elements  // using range based for loop  for (const int &n : numbers) {  cout << n << " ";  }  cout << "\nThe numbers are: ";  // Printing array elements  // using traditional for loop  for (int i = 0; i < 5; ++i) {  cout << numbers[i] << " ";  }  return 0;  } | |
| **Output:** | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 7** | **Date: 07**/04/2022 |
| 7. Write a program explaining the use of “this” operator. | |
| **Aim:**  To implement the concept of “this” operator. | |
| #include <iostream>    using namespace std;  class Box {  public:  // Constructor definition  Box(double l = 2.0, double b = 2.0, double h = 2.0) {  cout <<"Constructor called." << endl;  length = l;  breadth = b;  height = h;  }  double Volume() {  return length \* breadth \* height;  }  int compare(Box box) {  return this->Volume() > box.Volume();  }    private:  double length; // Length of a box  double breadth; // Breadth of a box  double height; // Height of a box  };  int main(void) {  Box Box1(3.3, 1.2, 1.5); // Declare box1  Box Box2(8.5, 6.0, 2.0); // Declare box2  if(Box1.compare(Box2)) {  cout << "Box2 is smaller than Box1" <<endl;  } else {  cout << "Box2 is equal to or larger than Box1" <<endl;  }    return 0;  } | |
| **Output:** | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 8** | **Date: 07**/04/2022 |
| 8. Write a program explaining the concept of unary and binary operator overloading. | |
| **Aim:**  To implement the concept of unary and binary operator overloading. | |
| a)as member function  #include<iostream>  using namespace std;  class num  {  private:  int a,b,c;  public:  num(int j,int k,int m)  {  a=j;b=k;c=m;  }  void show(void);  void operator ++( );  };  void num::show()  {  cout<<"\n a="<<a<<"\n b="<<b<<"\n c= "<<c;  }  void num::operator ++( )  {  ++a;  ++b;  ++c;  }  int main()  {  num n(13,63,241);  n.show();  ++n;  n.show();  }  b)as non member function  #include <iostream>  using namespace std;  class Complex  {  private:  float real;  float imag;  public:  Complex(){}  Complex(float r, float i)  {  real = r;  imag = i;  }  void display()  {  cout<<real<<"+i"<<imag;  }  friend Complex operator +(Complex &, Complex &);  };  Complex operator +(Complex &c1, Complex &c2)  {  Complex temp;  temp.real = c1.real + c2.real;  temp.imag = c1.imag + c2.imag;  return temp;  }  int main()  {  Complex c1(3, 4);  Complex c2(4, 6);  Complex c3 = c1+c2;  c3.display();  return 0;  } | |
| **Output:** | |
| **Result:** The above experiment successfully completed. | |