|  |  |
| --- | --- |
| **Ex. No. : 9** | **Date:** DD/MM/YYYY |
| **Experiment Heading** | |
| **Aim:** Write a program to explain the concept of Constructor overloading | |
| **Program:**  #include <iostream>  using namespace std;  class construct  {  public:  float area;    // Constructor with no parameters  construct()  {  area = 0;  }    // Constructor with two parameters  construct(int a, int b)  {  area = a \* b;  }    void disp()  {  cout<< area<< endl;  }  };  int main()  {  construct o;  construct o2( 10, 20);  o.disp();  o2.disp();  return 1;  } | |
| **Output:**  0  200 | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 13** | **Date:** DD/MM/YYYY |
| **Experiment Heading** | |
| **Aim:** Write a program to explain the concept of operator overloading. | |
| **Program:**  #include<iostream>  using namespace std;  class Complex {  private:  int real, imag;  public:  Complex(int r = 0, int i = 0) {real = r; imag = i;}    // This is automatically called when '+' is used with  // between two Complex objects  Complex operator + (Complex const &obj) {  Complex res;  res.real = real + obj.real;  res.imag = imag + obj.imag;  return res;  }  void print() { cout << real << " + i" << imag << '\n'; }  };  int main()  {  Complex c1(10, 5), c2(2, 4);  Complex c3 = c1 + c2;  c3.print();  } | |
| **Output:**  12 + i9 | |
| **Result:** The above experiment successfully completed. | |
| **Ex. No.: 14** | **Date:** DD/MM/YYYY |
| **Experiment Heading** | |
| **Aim:**  Write a program for the concept of virtual functions**.** | |
| **Program:**  #include<iostream>  using namespace std;  class base {  public:  virtual void print()  {  cout << "print base class\n";  }  void show()  {  cout << "show base class\n";  }  };  class derived : public base {  public:  void print()  {  cout << "print derived class\n";  }  void show()  {  cout << "show derived class\n";  }  };  int main()  {  base \*bptr;  derived d;  bptr = &d;  // Virtual function, binded at runtime  bptr->print();  // Non-virtual function, binded at compile time  bptr->show();    return 0;  } | |
| **Output:**  print derived class  show base class | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 15** | **Date:** DD/MM/YYYY |
| **Experiment Heading** | |
| **Aim:**  Write a program to understand the concept of inheritance. | |
| **Program:**  #include<iostream>  using namespace std;  class A  {  public:  int a=10;  void DisplayA()  {  cout<<"DisplayA : a="<<a<<endl;  }  };  class B: public A  {  public:  int b=20;  void DisplayB()  {  cout<<"DisplayB : b="<<b<<endl;  }  };  main()  {  B b1;  b1.DisplayA();  b1.DisplayB();  } | |
| **Output:**  DisplayA : a=10  DisplayB : b=20 | |
| **Result:** The above experiment successfully completed. | |