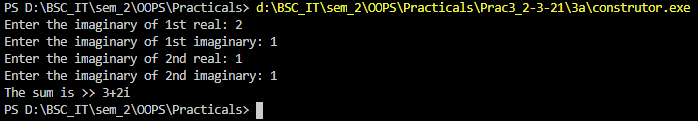
Practical 3: Constructors and method overloading.

1. Design a class Complex for adding the two complex numbers and also show the use of constructor.

Code:

|  |
| --- |
| /\*a. Design a class complex for      adding the two numbers and      also show the use of constructor\*/  #include<iostream>  class complex  {  private:      int real;      int imaginary;  public:      complex(){ real=0, imaginary=0;}      complex(int , int);      friend void disp\_sum(complex, complex);      ~complex();  };  int main()  {      int x, y;      std::cout << "Enter the imaginary of 1st real: ";      std::cin >> x;      std::cout << "Enter the imaginary of 1st imaginary: ";      std::cin >> y;      complex var1(x, y);      std::cout << "Enter the imaginary of 2nd real: ";      std::cin >> x;      std::cout << "Enter the imaginary of 2nd imaginary: ";      std::cin >> y;      complex var2(x, y);      disp\_sum(var1, var2);      return 0;  }  void disp\_sum(complex a, complex b)  {      std::cout << "The sum is >> "      << a.real + b.real<< "+"      <<a.imaginary+b.imaginary<<"i"<<std::endl;  }  complex::complex(int x, int y)  {      real = x;      imaginary = y;  }  complex::~complex()  {  } |

Output:

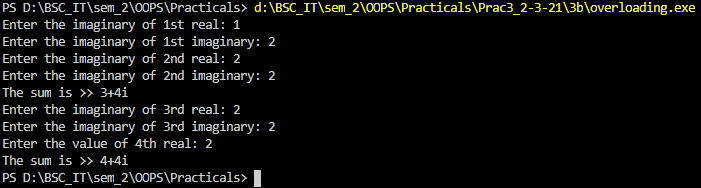


1. Design a class complex for adding the two numbers and also show the overloading of constructor

Code:

|  |
| --- |
| /\*b. Design a class complex for      adding the two numbers and      also show the overloading of constructor\*/  #include<iostream>  class complex  {  private:      int real;      int imaginary;  public:      complex(){ real=0, imaginary=0;}      complex(int x){ real=x, imaginary=x;}      complex(int x, int y){          real = x;          imaginary = y;      }      friend void disp\_sum(complex, complex);      ~complex(){}  };  int main()  {      int x, y;      std::cout << "Enter the imaginary of 1st real: ";      std::cin >> x;      std::cout << "Enter the imaginary of 1st imaginary: ";      std::cin >> y;      complex var1(x, y);      std::cout << "Enter the imaginary of 2nd real: ";      std::cin >> x;      std::cout << "Enter the imaginary of 2nd imaginary: ";      std::cin >> y;      complex var2(x, y);      disp\_sum(var1, var2);      std::cout << "Enter the imaginary of 3rd real: ";      std::cin >> x;      std::cout << "Enter the imaginary of 3rd imaginary: ";      std::cin >> y;      complex var3(x, y);      std::cout << "Enter the value of 4th real: ";      std::cin >> x;      complex var4(x);      disp\_sum(var3, var4);  }  void disp\_sum(complex a, complex b)  {      std::cout << "The sum is >> "      << a.real + b.real<< "+"      <<a.imaginary+b.imaginary<<"i"<<std::endl;  } |

Output:

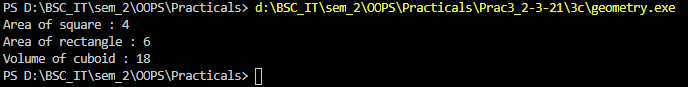


1. Design a class Geometry containing the methods area() and volume() and also overload the area()function.

Code:

|  |
| --- |
| /\*      Design a class Geometry containing      the methods areaO() and volumeO() and      also overload the areaO() function  \*/  #include<iostream>  using std::endl;  using std::string;  using std::cout;  using std::cin;  using std::getline;  class geometry  {  private:      float length, breadth, height, areaO, volumeO;  public:      geometry()      {          length=0, breadth=0, height=0, areaO = 0, volumeO = 0;      }      geometry(float x, float y, float z);      void area();      void area(string shape);      void volume();      float disp\_area();      float disp\_vol();      ~geometry();  };  int main()  {      geometry square(2, 0, 0);      geometry rectangle(2, 3, 0);      geometry cuboid(2, 3, 3);      square.area();      rectangle.area("rect");      cuboid.volume();      cout << "Area of square : " << square.disp\_area() << endl;      cout << "Area of rectangle : " << rectangle.disp\_area() << endl;      cout << "Volume of cuboid : " << cuboid.disp\_vol();  }  // constructor for cube  geometry::geometry(float x, float y, float z)  {      length = x, breadth = y, height = z, areaO = 0, volumeO = 0;  }  void geometry::area()  {      areaO = length\*length;  }  void geometry::area(string shape)  {      if(shape == "rect"){          areaO = length \* breadth;      }  }  void geometry::volume()  {      volumeO = length\*breadth\*height;  }  float geometry::disp\_area()  {      return areaO;  }  float geometry::disp\_vol()  {      return volumeO;  }  geometry::~geometry()  {  } |

Output:

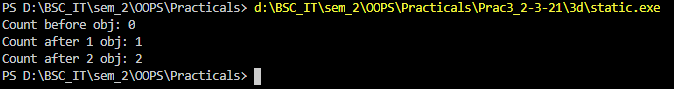


1. Design a class StaticDemo to show the implementation of static variable and staticfunction

Code:

|  |
| --- |
| /\*      d. Design a class StaticDemo to show the      implementation of static variable and static function  \*/  #include<iostream>  using std::cout;  class StaticDemo  {  private:      float l, b, h;      static int countObj;  public:      StaticDemo(){ l=2,b=4,h=5.3; countObj++;};      ~StaticDemo();      static int getCount() {          return countObj;      }  };  int StaticDemo::countObj = 0;  int main()  {      cout << "Count before obj: " << StaticDemo::getCount()<<std::endl;      StaticDemo a;      cout << "Count after 1 obj: " << StaticDemo::getCount()<<std::endl;      StaticDemo b;      cout << "Count after 2 obj: " << StaticDemo::getCount()<<std::endl;  }  StaticDemo::~StaticDemo()  {  } |

Output:



Write-up:

