|  |  |
| --- | --- |
| 1 | Access Private members outside class using friend function |
|  | #include "iostream"  using namespace std;  class xxx  {  private:  int a,b,c;  public:  friend int sum(xxx);  };  int sum(xxx p)  {  p.a=10;  p.b=20;  p.c=p.a+p.b;  return p.c;  }  main()  {  xxx p;  cout<<sum(p);  } |
| 2 | C++ program to demonstrate example of input and display elements |
|  | #include "iostream"  using namespace std;  class Demo  {  private:  int X;  int Y;  public:  Demo (int, int);  void Input();  void Display();  };  Demo:: Demo(int a, int b)  {  X = a;  Y = b;  }  void Demo:: Input()  {  cout << "Enter Value of X: "; cin >> X;  cout << "Enter Value of Y: "; cin >> Y;  }  void Demo:: Display()  {  cout << endl << "X: " << X;  cout << endl << "Y: " << Y << endl;  }  int main()  {  Demo d(10,20) ;  cout << endl <<"Method 1: " << endl;  cout << "Value after initialization: " ;  d.Display();  d.Input();  cout << "Value after User Input : ";  d.Display();  Demo d1 = Demo(10,20);  cout << endl << "Method 2: " << endl;  cout << "Value after initialization: ";  d1.Display();  return 0;  } |
| 3 | Copy Constructor program in C++. |
|  | /\*C++ program to demonstrate example of Copy Constructor.\*/  #include <iostream>  using namespace std;    //Class declaration.  class Demo  {      //Private block  to declare data member( X,Y ) of integer type.      private:          int X;          int Y;        //Public block of member function to access data members.      public:          //Declaration of parameterized constructor to initialize data members.              Demo (int a, int b);          //Declaration of copy constructor to initialize data members.              Demo (const Demo &d);          //To display output on screen.          void    Display();    };//End of class    //Definition of parameterized constructor.  Demo:: Demo(int a, int b)  {      X = a;      Y = b;  }    //Definition of copy constructor.  Demo:: Demo(const Demo &d)  {      X = d.X;      Y = d.Y;  }      //Definition of Display () member function.  void Demo:: Display()  {      cout << endl << "X: " << X;      cout << endl << "Y: " << Y << endl;  }    int main()  {      Demo d1(10,20) ; //Ctor automatically call when object is created.        //Display value of data member.      cout << endl <<"D1 Object: " << endl;      cout << "Value after initialization : " ;      d1.Display();        //Intialize object with other object using copy constructor      Demo d2 = Demo(d1);//also write like this :Demo d2(d1);        //Display value of data member.      cout << endl << "D2 Object: " << endl;      cout << "Value after initialization : ";      d2.Display();        return 0;  } |
| 4 | /\*C++ program to demonstrate example of constructor using this pointer.\*/ |
|  | #include <iostream>  using namespace std;    class Demo  {      private: //Private Data member section          int X,Y;      public://Public Member function section            //Default or no argument constructor.          Demo()          {                  X = 0;                  Y = 0;                    cout << endl << "Constructor Called";          }          //Perameterized constructor.          Demo(int X, int Y)          {                  this->X = X;                  this->Y = Y;                    cout << endl << "Constructor Called";          }            //Destructor called when object is destroyed          ~Demo()          {                  cout << endl << "Destructor Called" << endl;          }          //To print output on console          void putValues()          {                  cout << endl << "Value of X : " << X;                  cout << endl << "Value of Y : " << Y << endl;          }  };    //main function : entry point of program  int main()  {      Demo d1= Demo(10,20);        cout << endl <<"D1 Value Are : ";      d1.putValues();        Demo d2= Demo(30,40);        cout << endl <<"D2 Value Are : ";      d2.putValues();        //cout << endl ;        return 0;  } |
| 5 | /\*C++ program to create class to read and add two times.\*/ |
|  | #include <iostream>  using namespace std;    class Time  {  private:      int hours;      int minutes;      int seconds;    public:      void getTime(void);      void putTime(void);      void addTime(Time T1,Time T2);  };    void Time::getTime(void)  {      cout << "Enter time:" << endl;      cout << "Hours? ";    cin>>hours;      cout << "Minutes? ";  cin>>minutes;      cout << "Seconds? ";  cin>>seconds;  }    void Time::putTime(void)  {      cout << endl;      cout << "Time after add: ";      cout << hours << ":" << minutes << ":" << seconds << endl;  }    void Time::addTime(Time T1,Time T2)  {        this->seconds=T1.seconds+T2.seconds;      this->minutes=T1.minutes+T2.minutes + this->seconds/60;;      this->hours= T1.hours+T2.hours + (this->minutes/60);      this->minutes %=60;      this->seconds %=60;  }      int main()  {      Time T1,T2,T3;      T1.getTime();      T2.getTime();      //add two times      T3.addTime(T1,T2);      T3.putTime();        return 0;  } |