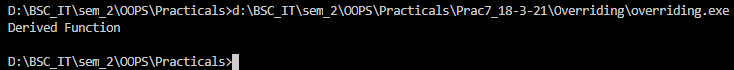
Practical 7: Virtual functions and abstract classes.

1. Implement the concept of method overriding.

Code:

|  |
| --- |
| // Implement the concept of method overriding.  #include<iostream>  using std::cout;  using std::endl;  class Base  {  public:      Base(){}      ~Base(){}      void print(){          cout << "Base Function"<<endl;      }  };  class Derived: public Base  {  public:      Derived(){}      ~Derived(){}      void print(){          cout << "Derived Function"<<endl;      }  };  int main(){      Derived D1;      D1.print();      return 0;  } |

Output:

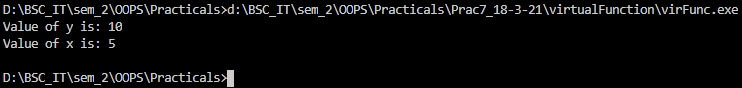


1. Show the use of virtual function.

Code:

|  |
| --- |
| // Show the use of virtualfunction  #include<iostream>  using std::cout;  using std::endl;  class A  {  private:      int x = 5;  public:      A(){}      virtual void display(){          cout << "Value of x is: " << x <<endl;      }      ~A(){}  };  class B: public A  {  private:      int y = 10;  public:      B(){}      void display(){          cout << "Value of y is: " << y <<endl;      }      ~B(){}  };  int main(){      A \*a;      A a1;      a = new B;      a->display();      a1.display();      return 0;  } |

Output:

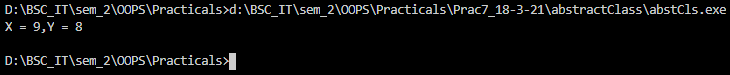


1. Show the implementation of abstract class.

Code:

|  |
| --- |
| // Show the implementation of abstract class.  #include<iostream>  using std::cout;  using std::endl;  //----------Base Class----------  class base  {  protected:      int x;  public:      base(int b){x = b;}      virtual void pure() = 0;      ~base(){}  };  //-------Derived Class-----------  class Dervied: public base  {  protected:      int y;  public:      Dervied(int b, int d): base(b) {y = d;}      void pure(){          cout << "X = "<< x <<",Y = "<< y<<endl;      }      ~Dervied(){}  };  //---------Main Program------------  int main()  {      base \*a;      a = new Dervied(3, 8);      a->pure();      return 0;  } |

Output:



Write-Up:

