OBJECT ORIENTED PROGRAMING(LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG)

Các tính chất :

Tính kế thừa

Tính đa hình

Tính đóng gói

Tính trừu tượng

Ví dụ tính kế thừa

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  class conVat  {  private:      /\* data \*/  public:      conVat(/\* args \*/)      {          cout<<"day la mot loai vat!"<<endl;      }  };  class conGa:public conVat  {  private:      /\* data \*/  public:      conGa(/\* args \*/):conVat()      {          cout<<"day con ga!"<<endl;      }  };  int main(int argc, const char\*\* argv) {   conGa ctrong;      return 0;  } |

lớp con conGa kế thừa các thuộc tính và phương thức của lớp cha có thế sử dụng lại phương thức contructor của lớp cha, kết quả khi thực hiện đoạn code trên conVat và conGa sẽ được thực hiện.

Ví dụ tính đa hình

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  class phepToan  {  private:      /\* data \*/  public:      uint8\_t phepCong(uint8\_t x, uint8\_t y)      {          return x+y;      }      uint16\_t phepCong(uint16\_t x, uint16\_t y)      {          return x+y;      }      uint32\_t phepCong(uint32\_t x, uint32\_t y)      {          return x+y;      }  }; |

ở đây có nhiều phương thưc phepCong nhưng kiểu trả về và kiểu tham số đầu vào hoàn toàn khác nhau, ví dụ này thể hiện tính đa hình trong lập trình hướng đối tượng

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  class phepToan  {  private:      uint8\_t x,y;  public:      phepToan(uint8\_t x=0,uint8\_t y=0)      {          this->x=x;          this->y=y;      }      uint8\_t phepCong()      {          return x+y;      }      uint8\_t phepNhan()      {          return x\*y;      }  }; |

Trong ví dụ trên tính đóng gói thể hiện ở chỗ bên ngoài có thể thao tác được các phương thức phạm vi public tuy nhiên không thể thay đổi trực tiếp các thuộc tính từ bên ngoài, nhờ đó tăng sự bảo mật .

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  class phepToan  {  private:      int x,y;      int phepNhan()      {          return x\*y;      }      void datGiaTri(int x,int y)      {          this->x=x;          this->y=y;      }  public:      phepToan(int x=0,int y=0)      {          this->x=x;          this->y=y;      }      void kiemTraKetQua(int x,int y)      {          datGiaTri(x,y);          if(phepNhan()>0)          {              cout<<"ket qua duong";          }          else          {              cout<<"ket qua am";          }      }  }; |

Từ ví dụ trên để kiểm tra một kết quả giữa 2 số là âm hay dương ta sử dụng phương thức kiemTraKetQua chúng ta không cần quan tâm trong phương thức đã hoạt động như thế nào không cần quan tâm để có được kết quả thì phải chọn số tính toán….