1. Chọn câu đúng:
   1. Các lớp cơ sở thì cụ thể hơn các lớp dẫn xuất
   2. C++ cung cấp 3 kiểu kế thừa: private, protected và public
   3. Một lớp dẫn xuất không thể đóng vai trò là một lớp cơ sở cho một lớp khác
   4. Hàm friend của lớp cơ sở không thể truy cập các thành viên protected
2. Nếu lớp Alpha thừa kế từ lớp Beta, lớp Alpha được gọi là \_\_\_\_\_\_, lớp Beta được gọi là \_\_\_\_\_\_\_\_\_
   1. lớp cơ sở lớp dẫn xuất
   2. lớp dẫn xuất lớp cơ sở
   3. lớp trừu tượng lớp cơ sở
   4. lớp dẫn xuất lớp trừu tượng
3. Xem xét đoạn mã sau:

class F{

protected:

int a,b;

private:

float c, d;

}

class C: public F{

}

C sử dụng được các biến thành viên nào của F

* 1. a, b, c, d
  2. c, d
  3. a, b
  4. không sử dụng được biến thành viên nào

1. Trong chương trình sau:

#include<iostream.h>

class A {public:

A() {cout << "Constructing A ";}

~A() {cout << "Destructing A ";}

};

class B: public A {public:

B() {cout << "Constructing B ";}

~B() {cout << "Destructing B ";}

};

int main() {

B obj;

return 0;

}

Đoạn chương trình in ra màn hình cái gì?

* 1. Constructing A Destructing A Constructing B Destructing B
  2. Constructing A Constructing B Destructing A Destructing B
  3. Constructing A Constructing B Destructing B Destructing A
  4. Constructing B Constructing A Destructing A Destructing B

1. Trong kế thừa, mối quan hệ giữa lớp dẫn xuất và lớp cơ sở
   1. Là mối quan hệ “is-a”
   2. Là mối quan hệ “has-a”
   3. Là mối quan hệ kết hợp các thuộc tính
   4. Không có mối quan hệ nào
2. Khi thực thi đoạn chương trình sau kết quả sẽ là

class Base{

public:

int xVal;

Base(int x=5){

xVal =x;

cout<<"xVal = "<<xVal<<endl;

}

};

class Derive: public Base

{

public: Derive(){xVal = 10;}

void Print(){

cout<<"xVal = "<<xVal<<endl;

}

};

void main(){

Derive d;

d.Print();

}

* 1. Màn hình xuất hiện: xVal = 5 xVal = 10
  2. Màn hình xuất hiện: xVal = 50
  3. Màn hình xuất hiện: xVal = 10
  4. Chương trình bị lỗi

1. Khi thực thi đoạn chương trình sau kết quả sẽ là

class Base{

public:

~Base(){ cout<<"Base class"<<endl; }

};

class Derive:Base

{

public: ~Derive(){ cout<<"Derive class"<<endl; }

};

void main(){

Base b;

Derive d;

}

* 1. Derive class Base class Base class
  2. Derive class Base class
  3. Base class Derive class
  4. Base class Derive class Base class

1. Cho đoạn mã

class A

{

public:

int a, b;

void nhap( )

{

cout << “a = ”; cin>>a;

cout << “b = ” ; cin>>b;

tinhtoan( );

}

void tinhtoan( )

{

cout << a +b;

}

};

class B: public A

{

void tinhtoan( )

{

cout << a\*b;

}

};

Nếu tạo ra đối tượng objB thuộc lớp B, khi gọi hàm nhap( ) và nhập a=13 và b=2 và gọi phuong thuc tinhtoan() thì kết quả nhận được là

* 1. 15
  2. 26
  3. Báo lỗi biên dịch
  4. Báo lỗi thực thi

1. Điều gì xảy ra nếu lớp cơ sở và lớp dẫn xuất cùng chứa một hàm có cùng khai báo
   1. Trình biên dịch sẽ báo lỗi
   2. Chỉ hàm của lớp cơ sở sẽ được gọi
   3. Chỉ hàm của lớp dẫn xuất sẽ được gọi
   4. Đối tượng của lớp cơ sở sẽ gọi hàm được định nghĩa ở lớp cơ sở, đối tượng của lớp dẫn xuất sẽ gọi hàm được định nghĩa ở lớp dẫn xuất
2. Khi khai báo kế thừa một lớp từ khóa nào sau đây là không hợp lệ
   1. public
   2. private
   3. protected
   4. friend
3. Các thành phần của lớp cơ sở có quyền truy cập là private thì được truy cập bởi:
   1. Các phương thức của lớp cơ sở và các lớp dẫn xuất
   2. Các phương thức của lớp cơ sở và các hàm bạn
   3. Các phương thức của lớp cơ sở, các lớp dẫn xuất và các hàm bạn bất kỳ

Ở bất kì đâu

1. Class C kế thừa class B, class B kế thừa class A, đều là kế thừa public. Vậy các phương thức của class C có thể truy cập
   1. Chỉ thành phần dữ liệu public và protected trong C và B
   2. Chỉ thành phần dữ liệu public và protected trong C
   3. thành phần dữ liệu private trong A và B
   4. thành phần dữ liệu protected trong A và B