|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++\_10\_개념확인 | 학번 : | 20185309 | 이름 : | 황명원 |

* 기본 개념 확인

1. 다음 질문에 O, X로 답하시오
2. 기본 클래스로 생성된 객체는 파생 클래스 멤버에 접근할 수 있다( o )
3. 접근 지정자 protected로 선언된 멤버는 전역 함수와 외부클래스에서 접근할 수 있다( x )
4. 상속 접근 지정자가 생략되면 public이다 ( o )
5. C++는 다중 상속을 허용한다( o )
6. 파생클래스 생성자에서 명시적으로 기본 클래스 생성자를 선택하지 않으면 묵시적으로 기본 생성자를 호출한다. ( o )
7. 상속은 클래스를 선언할 때 extends 키워드를 사용한다 ( x )
8. 파생 클래스의 포인터가 기본클래스의 포인터에 치환되는 것을 다운캐스팅 이라고 한다 ( x )
9. 업 캐스팅은 반드시 강제 타입 변환이 필요하다 ( x )
10. 상속을 통하여 파생클래스에서는 기본클래스이 모든 멤버를 사용할 수 있다 ( x )
11. private 상속은 기본클래스의 protected, public 멤버를 private으로 계승한다 ( o )
12. 가상 상속을 선언할 때 사용하는 키워드는 무엇인가?

virtual

1. 다음 클래스 Rocket과 클래스 Computer를 동시에 상속받아 클래스 Statellite를 선언하시오

class Rocket{ …. }

class Computer{ … }

class Statellite : public Rocket,public Computer{ … };

1. 다음 코드에 대하여 객체 b가 생성될 때 화면에 출력되는 내용은 무엇인가?

|  |
| --- |
| class A {  public:  A() { cout << "생성자 A" << endl; }  A(int x) { cout << "생성자 A" << x<< endl; }  };  class B: public A {  public:  B() { cout << "생성자 B" << endl;}  B(int x) { cout << "생성자 B" << x << endl; }  B(int x, int y) : A(x+y+2) {cout << "생성자 B" << x + y + 2 << endl; }  }; |

1. B b;

텍스트이(가) 표시된 사진

자동 생성된 설명

1. B b(10);

텍스트이(가) 표시된 사진

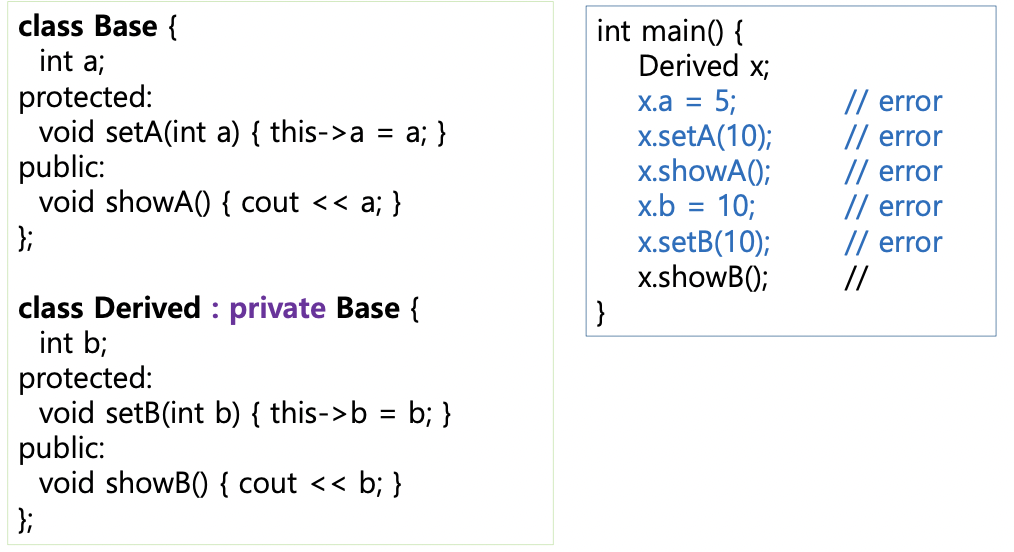
자동 생성된 설명

1. B b(10, 20);

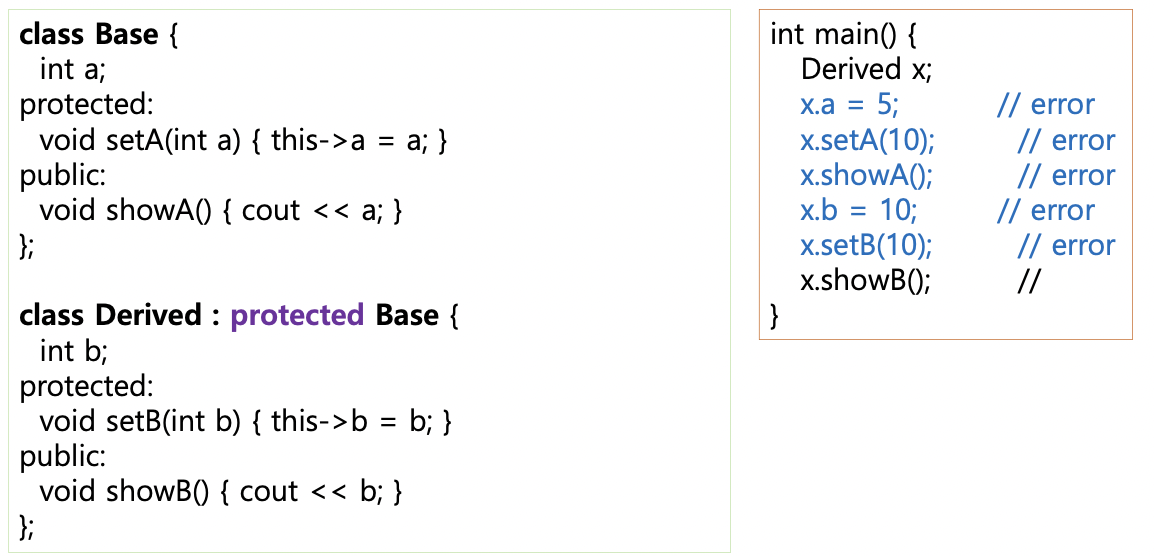
텍스트이(가) 표시된 사진

자동 생성된 설명

1. C++ 상속의 종류에 대하여 예를 제시하고 설명하시오.
2. private 상속 : 기본 클래스의 protected, public 멤버를 private으로 계승

ex)

1. protected 상속: 기본 클래스의 protected,public 멤버를 protected로 계승

ex)

1. public 상속: 기본 클래스의 protected, public 멤버를 public으로 계승

ex)

class Base{

int a;

protected:

void setA(int a) {this->a=a;}

public:

void showA(){cout<<a;}

};

class Derived : private Base {

int b;

protected:

void setB(int b) { this->b=b;}

public:

void showB(){cout<<b;}

};

int main(){

Derived x;

x.a=5;//error

x.setA(10); //error

x.showA();

x.b=10; //error

x.setB(10); //error

x.showB();

}

1. 다중 상속의 문제점과 해결방법을 예를 제시하고 설명하시오.

//예를 위한 코드

class BaseIO{

public:

int mode;

};

class In : public BaseIO{

public:

int readPos;

};

class Out : public BaseIO{

public:

int writePos;

};

class InOut : public In,public Out{

…

};

int main(){

inOut ioObj;

ioObj.readPos=10;

ioObj.writePos=10;

ioObj.safe=true;

ioObj.mode=5;

}

문제점:

-Base의 멤버가 이중으로 객체에 삽입되는 문제점.

-동일한 x를 접근하는 프로그램이 서로 다른 x에 접근하는 결과를 낳게 되어 잘못된 실행 오류가

발생된다

해결방법:

기본 클래스 앞에 virtual 을 선언하여 기본 클래스의 멤버가 중복하여 생성되는것을 방지한다.

Ex)

class In : virtual public BaseIO{

public:

int readPos;

};

class Out : virtual public BaseIO{

public:

int writePos;

};

1. 정사각형을 나타내는 Square 클래스를 작성하시오. Square 클래스는 데이터 멤버로 한 변의 길이를 나타내는 side를 가지며 둘레를 구하는 getPeri(), 넓이를 구하는 getArea()라는 멤버 함수를 갖는다

Square 클래스를 상속해서, 정육면체를 나타내는 Cube 클래스를 작성하시오. Cube 클래스는 추가적인 데이터 멤버가 없다. 대신 겉넓이를 구하는 getArea()와 부피를 구하는 getVolume() 멤버 함수를 갖는다

Square 클래스와 Cube 클래스 모두 적절한 생성자와 소멸자를 갖는다.

파생 클래스 객체를 생성하고 테스트하시오

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Square{  int size; public:  Square(int s);  ~Square();  int getPeri() const;  int getArea() const;  int getSize() const;  void print() const; };  class Cube : public Square{ public:  Cube(int s);  ~Cube();  int getSurface() const;  int getVolume() const;  void print() const;  };  Square::Square(int s):size(s) {} Square::~Square() {}  int Square::getArea() const {  return size\*size; } int Square::getPeri() const {  return 2\*(size+size);; } int Square::getSize() const {  return size; } void Square::print() const {  cout<<"Size: "<<size<<endl;  cout<<"Perimeter: "<<getPeri()<<endl;  cout<<"Area: "<<getArea()<<endl; } Cube::Cube(int s): Square(s) {} Cube::~Cube() {}  int Cube::getSurface() const {  return getPeri()\*getSize()+2\*getArea(); } int Cube::getVolume() const {  return getArea()\*getSize(); } void Cube::print() const{  cout<<"Size: "<<getSize()<<endl;  cout<<"Volume: "<<getVolume()<<endl;  cout<<"Surface: "<<getSurface()<<endl; }  int main(){  Square s(3);  s.print();  Cube c(4);  c.print(); } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 데이터 멤버로 length와 width를 갖는 Rectangle 클래스를 작성하시오. 클래스의 생성자와 소멸자를 정의하고, 둘레와 넓이를 구하는 멤버 함수도 작성한다

Rectangle 클래스를 상속하고, 추가적인 데이터 멤버로 height을 갖는 상자를 나타내는 Cuboid 클래스를 작성하시오. Cuboid의 생성자와 소멸자를 정의하고, 겉넓이와 부피를 구하는 멤버 함수도 작성한다

파생 클래스 객체를 생성하고 테스트하시오

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Rectangle{  int length;  int width; public:  Rectangle(int l,int w);  ~Rectangle();  int getPerimeter()const;  int getArea()const;  void print()const; };  Rectangle::Rectangle(int l, int w):length(l),width(w) {} Rectangle::~Rectangle() {} int Rectangle::getPerimeter() const {  return 2\*(length+width); } int Rectangle::getArea() const {  return length\*width; } void Rectangle::print() const {  cout<<"length: "<<length<<endl;  cout<<"width: "<<width<<endl;  cout<<"Perimeter: "<<getPerimeter()<<endl;  cout<<"Area: "<<getArea()<<endl;  } class Cuboid : public Rectangle{  int height; public:  Cuboid(int l,int w,int h);  ~Cuboid();  int getSurface() const;  int getVolume() const;  void print() const; };  Cuboid::Cuboid(int l, int w, int h) : Rectangle(l,w),height(h) {} Cuboid::~Cuboid() {} int Cuboid::getSurface() const {  return getPerimeter()\*height+2\*getArea(); } int Cuboid::getVolume() const {  return getArea()\*height; } void Cuboid::print() const {  Rectangle::print();  cout<<"Height : "<<height<<endl; }  int main(){  cout<<"Rectangle 객체 초기화"<<endl;  Rectangle rectangle(5,4);  rectangle.print();   cout<<"Perimeter: "<<rectangle.getPerimeter()<<endl;  cout<<"Area: "<<rectangle.getArea()<<endl<<endl;   cout<<"Cuboid 객체 초기화"<<endl;  Cuboid cuboid(5,4,8);  cuboid.print();  cout<<"Surface : "<<cuboid.getSurface()<<endl;  cout<<"Volume : "<<cuboid.getVolume()<<endl; } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |