|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++\_11\_개념확인 | 학번 : | 20185309 | 이름 : | 황명원 |

* 기본 개념 확인

1. 제시된 프로그램에 대한 결과를 제시하시오

class Base {

public:

void f() { cout << "Base::f() called" << endl; }

};

class Derived : public Base {

public:

void f() { cout << "Derived::f() called" << endl; }

};

int main(){

Derived d;

Derived \*pDer = &d;

pDer->f();

Base \*pBase = pDer;

pBase->f();

}

ㅡ실행결과

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 호출하는 함수의 결정을 실행 시간에 하도록 컴파일러에게 지시하는 키워드는 무엇인가?

virtual 키워드

1. 다음 중 순수 가상 함수는 무엇인가?

답:4번

1. void paint()=0;
2. virtual void paint(){ }
3. virtual void paint();
4. virtual void paint()=0; ->정답
5. 다음 중 추상클래스는 무엇인가?

답:2번

1. class A{ void draw()=0; };
2. class A{ virtual void draw()=0;}; ->정답
3. abstract class A { void draw(); };
4. class A{virtual void draw() { } };

1. 다음 중 오류가 발생하는 경우를 모두 고르시오.

class Shape {

public:

virtual void draw()=0;

};

class Circle : public Shape {

public:

double getArea() { return 3.14; }

};

답:1번,2번,3번,4번

1. Shape sh; ->오류
2. Shape \*sh; ->오류
3. Circle c; ->오류
4. Circle \*c; ->오류
5. 밑줄 친 부분을 채워 넣으시오

class Shape {

public:

virtual void draw(){ cout << "--Shape--"; }

};

class Circle : public Shape {

public:

virtual void draw() {

Shape::draw();1) // 기본 클래스의 draw() 호출

cout << "Circle" << endl;

}

};

int main(){

Shape \* pShape = new Circle();

pShape->Shape::draw(); 2) //기본 클래스 draw() 호출

}

1. 오버라이딩에 대하여 설명하시오

파생클래스에서 기본클래스의 가상함수와 동일한 이름의 함수선언

ㅡ기본 클래스의 가상 함수의 존재감 상실시킴  
ㅡ파생 클래스에서 오버라이딩 한 함수가 호출되도록 동적 바인딩

ㅡ함수 재정의라고도 부름 - 다형성의한종류

오버라이딩의 목적:

ㅡ 파생 클래스에서 구현할 함수 인터페이스 제공(파생 클래스의 다형성)

1. 추상 클래스에 대하여 설명하시오

최소한 하나의 순수 가상함수를 가지고있는 클래스

특징: 1.온전한 클래스가 아니므로 객체생성 불가능

2..추상클래스의 포인터 선언은 가능

목적:1.추상클래스의 인스턴스를 생성하는 목적이 아님

2.상속에서 기본클래스의 역할을 하기위함

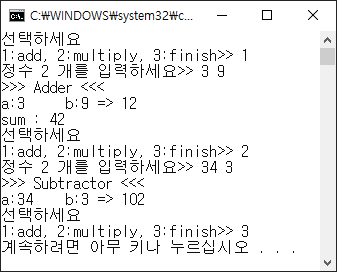
- 순수 가상 함수를 통해 파생 클래스에서 구현할 함수의 형태(원형)을 보여주는 인터페이스 역할

- 추상 클래스의 모든 멤버 함수를 순수 가상 함수로 선언할 필요 없음

1. 다음과 같은 클래스를 상속받는 Adder, Mul 클래스를 구현하고 다형성을 사용하여 제시된 결과처럼 실행되는 프로그램을 완성 하시오. 단, 1을 선택하면 입력된 두 정수 a~b까지의 합을 계산하여 반환하는 sum() 함수를 호출하여 결과를 출력한다

class Calculator {

protected:

int a, b;

public:

Calculator(int a, int b) : a(a), b(b) {}

virtual int calc() = 0;

virtual void write() {

cout << "a:" << a << “\tb:＂ << b <<“ =>＂ ;

}

};

class Adder : public Calculator {

public:

int sum(); //Adder 구현 시 추가

};

int main() {

Manage::run();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  using namespace std;  class Calculator { protected:  int a, b; public:  Calculator(int a, int b) : a(a), b(b) {}   virtual int calc() = 0;   virtual void write() {  cout << "a:" << a << "\tb:" << b << "=>";  } };  class Adder : public Calculator { public:  Adder(int a,int b):Calculator(a,b){}  int calc(){  return a+b;  }  void write(){  cout<<">>> Adder <<<"<<endl;  Calculator::write();  }  int sum(){  int total=0;  for(int i=a ;i<=b ;i++){  total+=i;  }  return total;  } }; class Mul:public Calculator{ public:  Mul(int a,int b):Calculator(a,b){}  int calc(){  return a\*b;  }  void write(){  cout<<">>> Subtractor <<<"<<endl;  Calculator::write();  } };  class Manage{ public:  static void run(){  Calculator \*ca= nullptr;  int choice,a,b;  while(true){  cout<<"선택하세요\n1:add, 2:multiply, 3:finish>>";  cin>>choice;  if(choice==3)  break;  cout<<"정수 2 개를 입력하세요>> ";  cin>>a>>b;  switch (choice) {  case 1:  ca=new Adder(a,b);  break;  case 2:  ca=new Mul(a,b);  break;  }  ca->write();  cout<<ca->calc()<<endl;  if(typeid(\*ca)== typeid(Adder)){  Adder \*ad=dynamic\_cast<Adder \*>(ca);  cout<<"sum : "<<ad->sum()<<endl;  }  delete ca;  }  } }; int main() {  Manage::run(); } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |