|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cpp\_13주 | 학번 : | 20175260 | 이름 : | 최영빈 |

* **개념 확인 – 가상함수와 추상 클래스**

1. 기본 클래스의 가상 함수와 동일한 타입의 함수를 파생 클래스에서 작성하는 것을 무엇이라고 하는가?

==== 풀이 ====

오버라이딩

1. 오버라이딩, 연산자 중복등은 c++언어의 어떤 객체 지향 특성인가?

==== 풀이 ====

실행 시간 다형성

1. 다음 중 의미가 다른 하나는?
2. dynamic binding
3. late binding
4. compile-time binging <—————————
5. run-time binding
6. 순수 가상 함수에 대해 잘못 말한 것은? 3
7. 순수 가상 함수를 가진 클래스는 무조건 추상 클래스이다
8. 순수 가상 함수는 실행이 목적이 아니라 파생 클래스가 구현해야 할 함수를 알려주기 위한 것이다
9. 외부 함수도 순수 가상 함수로 선언 가능하다 <—————————
10. 순수 가상 함수가 호출되면 동적 바인딩이 일어난다
11. 다음과 같은 코드가 있을 때, 클래스 A와 B가 다형성을 활용하게 해서 적절한 print 함수를 호출하게 하려면, 무엇을 추가해야 할지 설명하시오.

class A {...};

class B: public A {...};

int main(){

A\* ptr;

ptr = new A ();

ptr -> print ();

ptr = new B ();

ptr -> print ();

return 0;

}

==== 풀이 ====

virtual

1. 다음 코드에 대하여 질문에 답하시오

|  |
| --- |
| class Base{    public:    void func(){ f(); }    void f(){ cout<<"Base::f() called"<<endl;}  };  class Derived:public Base{    public:    void f() {cout<<"Derived::f() called"<<endl;}  };  int main(){    Derived der;    der.f();  //(1)    Base base;    Base \*p=&base;    p->f();   //(2)    p=&der;    p->f();  //(3)    p->func(); //(4)  } |

1. 주석으로 표시된 코드가 실행될 때 출력되는 내용은?

==== 풀이 ====

1 - Derived::f() called, 2 -Base::f() called, 3 - Base::f() called, 4 - Base::f() called

1. 다형성 적용을 위해 추가될 내용은 무엇인가?

==== 풀이 ====

virtual

1. 위2번 내용을 추가한 후 주석으로 표시된 코드가 실행될 때 출력되는 내용을 제시하시오

==== 풀이 ====

1 - Derived::f() called, 2 -Base::f() called, 3 - Derived::f() called, 4 - Derived::f() called

1. 다형성이 적용된 코드에서 주석 3번 문장 실행 시 Base클래스의 f()를 호출할 수 있도록 하려면 어떻게 해야 하는가?

==== 풀이 ====

p -> Base::f();

1. 다음 코드에 대하여 질문에 답하시오

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  class Person{    int id;    public:    Person(int id) : id(id){}    ~Person(){ cout<<"id= "<<id<<endl;}  };  class Student:public Person{    char \*name;    public:    Student(int id, const char\* name):Person(id){      int len=strlen(name);      this->name=new char[len];      strcpy(this->name, name);    }    ~Student(){       cout<<"name = "<< name <<endl;       delete[] name;    }  };  int main(){    Person \*p= new Student{12, "c++"};    delete p;  } |

1. 실행 결과는 무엇인가? 실행 결과의 문제점은 무엇인가?

==== 풀이 ====

id = 12, 정상적인 소멸이 안됨

1. Person 클래스나 Student 클래스를 수정하여 문제점을 해결하고 결과를 제시하시오.

==== 실행 결과 ====

name = c++

id = 12

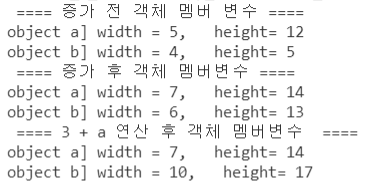
==== 문제점 해결 방법 ====

소멸자에 가상함수 키워드를 추가한다.

1. 다음 중 컴파일 시에 바인딩되지 않는 것은?
2. 중복된 함수 중에서 구분하여 호출
3. 중복된 연산자 중에서 구분하여 호출
4. 범위 지정 연산자(::)를 이용한 함수 호출
5. 순수 가상 함수 호출 <—————

* **개념 적용 응용 프로그래밍 – 상속**

1. 사각형을 추상화 한 Rect 클래스에 대하여 다음 연산이 가능하도록 프렌드 함수로 작성하고 프로그램을 완성 하시오.



class Rect {

  int width, height;

public:

  Rect(int width = 0, int height=0);

  void show(string name);

};

int main() {

  Rect a(5, 12), b(4, 5);

  cout<<" ==== 증가 전 객체 멤버 변수 ==== "<<endl;

  a.show("a");

  b.show("b");

  cout<<" ==== 증가 후 객체 멤버변수 ==== "<<endl;

  ++a; // width와 height를 1 증가

  b = a++; // width와 height를 1 증가

  a.show("a");

  b.show("b");

  cout<<" ==== 3 + a 연산 후 객체 멤버변수  ==== "<<endl;

  b = 3 + a; // b의 width와 height 값을 a에 3을 더한 것으로 변경

  a.show("a");

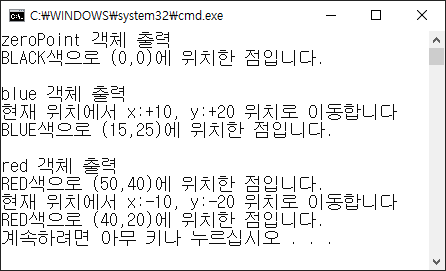
  b.show("b");

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Rect  {  int width, height;  public:  Rect(int width = 0, int height = 0);  void show(string name);  friend Rect& operator++(Rect& a);  friend Rect operator++(Rect& a, int b);  friend Rect operator+(int a, const Rect& b);  };  Rect::Rect(int w, int h){  width = w;  height = h;  }  void Rect::show(string name){  cout <<"object "<<name<<"] width = "<< width<<", height = "<<height<<endl;  }  Rect& operator++(Rect& a){  a.width++;  a.height++;  return a;  }  Rect operator++(Rect& a, int b){  Rect tmp = a;  a.height++;  a.width++;  return tmp;  }  Rect operator+(int a, const Rect& b){  Rect tmp = b;  tmp.width = b.width+a;  tmp.height = b.height + a;  return tmp;  }  int main()  {  Rect a(5, 12), b(4, 5);  cout << " ==== 증가 전 객체 멤버 변수 ==== " << endl;  a.show("a");  b.show("b");  cout << " ==== 증가 후 객체 멤버변수 ==== " << endl;  ++a; // width와 height를 1 증가  b = a++; // width와 height를 1 증가  a.show("a");  b.show("b");  cout << " ==== 3 + a 연산 후 객체 멤버변수  ==== " << endl;  b = 3 + a; // b의 width와 height 값을 a에 3을 더한 것으로 변경  a.show("a");  b.show("b");  } |
| [실행 결과]  이미지 |

1. Point 클래스를 상속받는 ColorPoint 클래스를 작성하고 제시된 결과처럼 실행될 수 있도록 main()을 완성하시오.

class Point {

 int x, y;

public:

Point(int x, int y) { this->x = x; this->y = y; }

int getX() { return x; }

int getY() { return y; }

protected:

void move\_up(int x, int y) { this->x += x; this->y += y; }

void move\_down(int x, int y) { this->x -= x; this->y -= y; }

};

int main() {

ColorPoint zeroPoint; // BLACK에 (0, 0) 위치의 점

ColorPoint blue(5, 5);

cout << endl<<"blue 객체 출력" << endl;

cout << "현재 위치에서 x:+10, y:+20 위치로 이동합니다" << endl;

blue.setPoint(10, 20, '+');

ColorPoint red(50, 40, "RED");

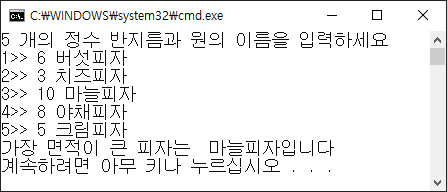
cout << "현재 위치에서 x:-10, y:-20 위치로 이동합니다" << endl;

red.setPoint(10, 20, '-');

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Point  {  int x, y;  public:  Point(){  x =0;  y =0;  }  Point(int x, int y)  {  this->x = x;  this->y = y;  }  int getX() { return x; }  int getY() { return y; }  protected:  void move\_up(int x, int y)  {  this->x += x;  this->y += y;  }  void move\_down(int x, int y)  {  this->x -= x;  this->y -= y;  }  };  class ColorPoint:public Point{  string c;  public :  ColorPoint();  ColorPoint(int a, int b, string c = "Blue");  void setPoint(int x,int y, char z);  void show();  };  ColorPoint ::ColorPoint():Point(0,0){  c = "Black";  }  ColorPoint :: ColorPoint(int a, int b, string c):Point(a,b){  this->c = c;  }  void ColorPoint::setPoint(int a , int b, char z){  if(z == '+')  move\_up(a,b);  else if(z == '-')  move\_down(a,b);  }  void ColorPoint::show(){  cout << c << "색으로 (" << getX() << "," << getY() << ") 에 위치한 점입니다." << endl;  }  int main()  {  cout << "zeroPoint 객체 출력 "<<endl;  ColorPoint zeroPoint; // BLACK에 (0, 0) 위치의 점  zeroPoint.show();  ColorPoint blue(5, 5);  cout << endl  << "blue 객체 출력" << endl;  cout << "현재 위치에서 x:+10, y:+20 위치로 이동합니다" << endl;  blue.setPoint(10, 20, '+');  blue.show();  ColorPoint red(50, 40, "RED");  red.show();  cout << "현재 위치에서 x:-10, y:-20 위치로 이동합니다" << endl;  red.setPoint(10, 20, '-');  red.show();  } |
| [실행 결과]  이미지 |

1. 원을 추상화 한 Circle 클래스에 대하여 다음과 같이 배열을 선언하여 제시된 결과처럼 실행될 수 있도록 Circle을 상속받은 NamedCircle 클래스와 main()함수 등 필요한 함수를 작성하시오

class Circle {

int radius;

public:

Circle(int radius = 0) { this->radius = radius; }

int getRadius() { return radius; }

void setRadius(int radius) { this->radius = radius; }

double getArea() { return 3.14\*radius\*radius; };

};

NamedCircle c[5];

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Circle  {  int radius;  public:  Circle(int radius = 0) { this->radius = radius; }  int getRadius() { return radius; }  void setRadius(int radius) { this->radius = radius; }  double getArea() { return 3.14 \* radius \* radius; };  };  class NamedCircle:public Circle{  string p;  public:  NamedCircle() = default;  void setName(string a);  string getName(){return p;}  };  void NamedCircle::setName(string a){  p = a;  }  string max();  NamedCircle c[5];  string max(){  int max = 0;  int count;  for(int i = 0; i < 5; i++){  if(max < c[i].getArea()){  count = i;  max = c[i].getArea();  }  }  return c[count].getName();  }  int main()  {  cout << "5 개의 정수 반지름과 원의 이름을 입력하세요 "<<endl;  int radius;  string pizza;  for(int i =0; i < 5; i++){  cout <<i+1<<">> ";  cin >> radius;  c[i].setRadius(radius);  cin >> pizza;  c[i].setName(pizza);  }  cout << "가장 면적이 큰 피자는 "<<max()<<"입니다."<<endl;  } |
| [실행 결과]  이미지 |

1. 다음과 같은 클래스에 대하여 **상속을 적용하여** 제시된 결과처럼 실행될 수 있도록 프로그램을 작성하시오. 단, 잉크젯 프린터와 레이저 프린터 객체를 하나만 동적 생성하여 사용한다.

class InkJetPrinter {

int availableInk; //잉크 잔량

string model; //모델

string manufacturer; //제조사

int printedCount; //총 인쇄 매수

int availableCount; //인쇄 종이 잔량

public:

void printInkJet(int pages = 1); //pages 수만큼 용지 사용, 잉크잔량은 pages 수만큼 감소

void showStateInkJet(); //현재 상태 출력

};

class LaserPrinter {

int availableToner; //토너 잔량

string model; //모델

string manufacturer; //제조사

int printedCount; //인쇄매수

int availableCount; //인쇄 종이 잔량

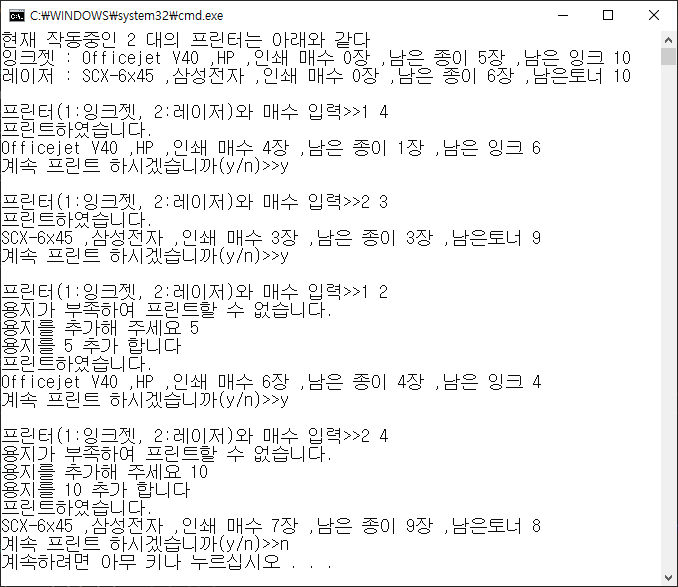
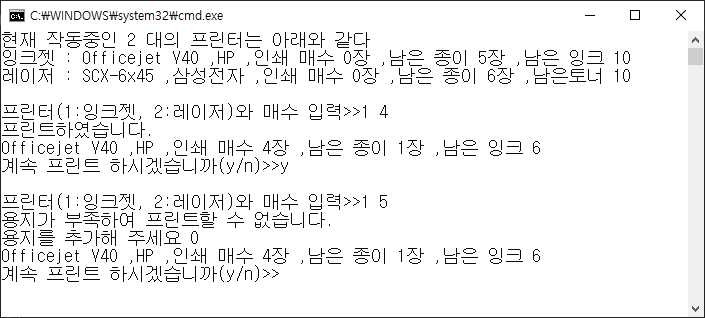
public:

void printLaser(int pages = 1); //pages 수만큼 용지 사용, 토너 잔량은 -1 감소

void showStateLaser(); //현재 상태 출력

};

**- 용지가 부족하면 입력한 값만큼 추가한 후 프린트 하고, 0을 입력하면 현재 프린터 상태만 출력한다**

int main() {

PrinterManager man;

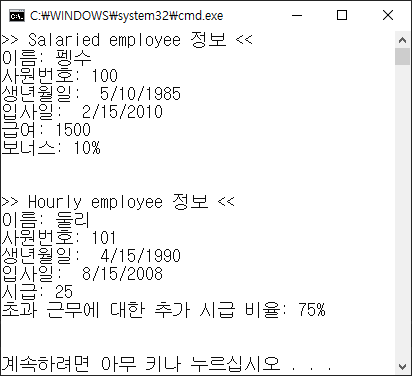
man.operater();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <cstring>  using namespace std;  class base{  protected:  string model; //모델  string manufacturer; //제조사  int printedCount; //총 인쇄 매수  int availableCount; //인쇄 종이 잔량  public :  //base() = default;  base(string m,string ma,int pc, int ac);  bool show(int pages);  };  base::base(string m,string ma,int pc, int ac){  model = m;  manufacturer = ma;  printedCount = pc;  availableCount = ac;  }  bool base::show(int pages){  int a;  bool flag =true;  if (availableCount > pages)  {  printedCount += pages;  cout << "프린트하였습니다." << endl;  availableCount -= pages;  }  else  {  cout << "용지가 부족하여 프린트할 수 없습니다. " << endl  << "용지를 추가해주세요 ";  cin >> a;  if (a == 0)  {  flag = false;  }  else  {  availableCount += a;  printedCount += pages;  cout << "용지를 " << a << "추가 합니다" << endl  << "프린트하였습니다" << endl;  availableCount -= pages;  }  }  return flag;  }  class InkJetPrinter:public base  {  int availableInk; //잉크 잔량  public:  InkJetPrinter();  void printInkJet(int pages = 1); // pages 수만큼 용지 사용, 잉크잔량은 pages 수만큼 감소  void showStateInkJet(); //현재 상태 출력  };  InkJetPrinter::InkJetPrinter() : base("Officejet V40", "HP", 0, 5)  {  availableInk = 10;  cout << "잉크젯 : ";  showStateInkJet();  }  void InkJetPrinter::printInkJet(int pages){  if(show(pages))  availableInk -= pages;  showStateInkJet();  }  void InkJetPrinter::showStateInkJet(){  cout << model << " ," << manufacturer << " ,인쇄 매수 " << printedCount << "장"  << "남은 종이 " << availableCount << "장 ,남은 잉크 " << availableInk << endl;  }  class LaserPrinter:public base  {  int availableToner; //토너 잔량  public:  LaserPrinter();  void printLaser(int pages = 1); // pages 수만큼 용지 사용, 토너 잔량은 -1 감소  void showStateLaser(); //현재 상태 출력  };  LaserPrinter::LaserPrinter() : base("SCX-6x45", "삼성전자", 0, 6){  availableToner = 10;  cout << "레이저 : " << model << " ," << manufacturer << " ,인쇄 매수 " << printedCount << "장, "  << "남은 종이 " << availableCount << "장 ,남은 토너 " << availableToner<< endl;  }  void LaserPrinter::printLaser(int pages)  {  if(show(pages)){  availableToner--;  }  showStateLaser();  }  void LaserPrinter::showStateLaser()  {  cout << model << " ," << manufacturer << " ,인쇄 매수 " << printedCount << "장, "  << "남은 종이 " << availableCount << "장 ,남은 토너 " << availableToner << endl;  }  //- 용지가 부족하면 입력한 값만큼 추가한 후 프린트 하고, 0을 입력하면 현재 프린터 상태만 출력한다  class PrinterManager{  InkJetPrinter \*i;  LaserPrinter \*l;  int pages;  int num;  public:  PrinterManager();  void operater();  ~PrinterManager();  };  PrinterManager::PrinterManager(){  cout << "현재 작동중인 2 대의 프린터는 아래와 같다." << endl;  i = new InkJetPrinter();  l = new LaserPrinter();  }  void PrinterManager::operater(){  char a;  bool flag = true;  while(flag){  cout << "프린터(1: 잉크젯, 2: 레이저)와 매수 입력>>";  cin>>num>>pages;  switch (num)  {  case 1:  i->printInkJet(pages);  break;    case 2:  l->printLaser(pages);  break;  }  while(true){  cout <<"계속 프린트 하시겠습니까(y/n)>>";  cin >> a;  if(a == 'y'){  flag = true;  cout << endl;  break;  }  else if(a == 'n'){  flag = false;  break;  }  else{  cout << "다시 입력하세요. "<<endl;  continue;  }  }  }  }  PrinterManager::~PrinterManager(){  delete i;  delete l;  }  int main(){  PrinterManager man;  man.operater();  } |
| [실행 결과]  이미지 |

1. 직원을 나타내는 Employee 클래스, 그리고 이를 기반으로 하는 월급을 받는 직원을 나타내는 Salaried 클래스, 시급을 받는 직원을 나타내는 Hourly 클래스를 작성하시오. 모든 Employee 클래스는 데이터 멤버로 이름(name), 직원 번호(employee number), 생일(birth date), 입사일(date hired)을 갖는다. 그리고 Salaried 클래스는 월급여(monthly salary)와 연간 보너스(annual bonus)(0~10%의 범위)를 추가로 갖고, Hourly 클래스는 시급(hourly wage)과 초과 근무에 대한 추가 시급 비율(overtime rate)(50~100%의 범위)을 추가로 갖는다.

int main(){

 cout << ">> Salaried employee 정보 <<" << endl;

Date birth1(5, 10, 1985); //생년월일

Date hired1(2, 15, 2010); //입사일

Salaried smith("펭수", 100, birth1, hired1, 1500.00, 10);

smith.print();

cout << endl << endl;

cout << ">> Hourly employee 정보 <<" << endl;

Date birth2(4, 15, 1990); //생년월일

Date hired2(8, 15, 2008); //입사일

Hourly howard("둘리", 101, birth2, hired2, 25.00, 75);

howard.print();

cout << endl << endl;

return 0;

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Date{  string name;  public:  Date() = default;  Date(int a, int b, int c);  string getName();  };  Date::Date(int a, int b, int c){  name = to\_string(a) + "/" + to\_string(b) + "/" + to\_string(c);  }  string Date::getName(){  return name;  }  class Employee{  string name;  int number;  Date birthdate;  Date datehired;  public:  Employee(string a, int b, Date c, Date d);  void print();  };  Employee::Employee(string a,int b, Date c,Date d){  name = a;  number = b;  birthdate = c;  datehired = d;  }  void Employee::print(){  cout<< "이름 : "<< name <<endl << "사원번호 : "<<number<<endl;  cout<<"사원번호 : "<<birthdate.getName()<<endl<<"입사일 : "<<datehired.getName()<<endl;  }  class Salaried : public Employee{  double salary;  int annualbonus;  public :  Salaried(string a,int b,Date c,Date d,double e,int f);  void print();  };  Salaried::Salaried(string a, int b, Date c, Date d, double e, int f):Employee(a,b,c,d){  salary = e;  annualbonus = f;  }  void Salaried::print()  {  Employee::print();  cout << "급여 :" << salary << endl;  cout << "보너스 : " << annualbonus << "%" << endl;  }  class Hourly : public Employee  {  double hourlywage;  int overtimerate;  public:  Hourly(string a, int b, Date c, Date d, double e, int f);  void print();  };  Hourly::Hourly(string a,int b,Date c, Date d, double e, int f):Employee(a,b,c,d){  hourlywage = e;  overtimerate = f;  }  void Hourly::print(){  Employee::print();  cout << "시급 " << static\_cast<int>(hourlywage)<<endl;  cout << "초과 근무에 대한 추가 시급 비율 : "<<overtimerate<<"%"<<endl;  }  int main()  {  cout << ">> Salaried employee 정보 <<" << endl;  Date birth1(5, 10, 1985); //생년월일  Date hired1(2, 15, 2010); //입사일  Salaried smith("펭수", 100, birth1, hired1, 1500.00, 10);  smith.print();  cout << endl<< endl;  cout << ">> Hourly employee 정보 <<" << endl;  Date birth2(4, 15, 1990); //생년월일  Date hired2(8, 15, 2008); //입사일  Hourly howard("둘리", 101, birth2, hired2, 25.00, 75);  howard.print();  cout << endl<< endl;  return 0;  } |
| [실행 결과]  이미지 |