|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++ \_12 | 학번 : | 20185309 | 이름 : | 황명원 |

* **프로그램 과제**

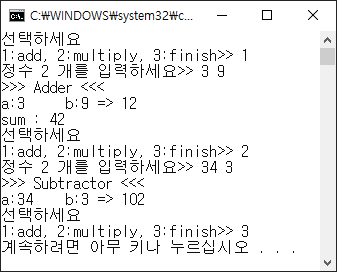
1. 다음과 같은 클래스를 상속받는 Adder, Mul 클래스를 구현하고 다형성을 사용하여 제시된 결과처럼 실행되는 프로그램을 완성 하시오. 단, 1을 선택하면 입력된 두 정수 a~b까지의 합을 계산하여 반환하는 sum() 함수를 호출하여 결과를 출력한다.

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();

}

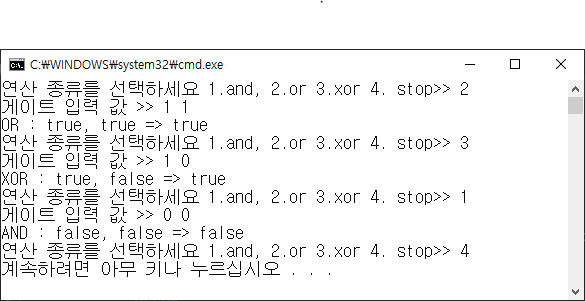
}

|  |
| --- |
| [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 sum(){  int total=0;  for(int i=a ; i<=b ; i++)  total+=i;  return total;  }//Adder 구현 시 추가  int calc(){  return a+b;  }  void write(){  cout<<">>> Adder <<<"<<endl;  Calculator::write();  }  }; 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 Calculator{ public:  void static run(){  Calculator \*ca= nullptr;  int num;  int a,b;  while(true) {  cout << "선택하세요" << endl;  cout << "1: add, 2:multiply, 3:finish>>";  cin>>num;  if(num==3)  break;  cout<<"정수 2 개를 입력하세요>>";  cin>>a>>b;  if(num==1)  ca = new Adder(a, b);  else if(num==2)  ca=new Mul(a,b);  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 |

1. 두 개의 입력 신호를 받아 and, or, xor 연산을 수행한 결과를 출력하는 기본 게이트를 구현하는 프로그램을 완성하시오. 단, 각각의 게이트는 ANDGate, ORGate, XORGate 클래스로 작성하며 AbstractGate를 상속받는다. – 다형성을 적용할 것

* 논리형 상수 값 출력 시 아래 문장을 참고하세요

cout<<boolalpha<<true;

class AbstractGate {

public:

virtual bool operation(bool x, bool y) = 0; // 순수 가상 함수

};

int main() {

Manage::go();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class AbstractGate { public:  virtual bool operation(bool x, bool y) = 0; // 순수 가상 함수 }; class ANDGate : public AbstractGate{ public:  bool operation(bool x,bool y){  if(x==true and y== true)  return true;  else  return false;  } }; class ORGate : public AbstractGate{ public:  bool operation(bool x,bool y){  if(x== false and y== false)  return false;  else  return true;  } }; class XORGate : public AbstractGate{ public:  bool operation(bool x,bool y){  if (x != y) return true;  else return false;  } }; class Manage : public AbstractGate{ public:  void static go(){  AbstractGate \*ag= nullptr;  int choice;  bool x,y;  while(true){  cout<<"연산 종류를 선택하세요 1.and, 2.or 3.xor 4.stop>>";  cin>>choice;  if(choice==4)  break;  cout<<"게이트 입력 값>>";  cin>>x>>y;  switch (choice) {  case 1:  ag = new ANDGate();  cout << "AND : " << boolalpha << x << ", " << boolalpha << y << " => ";  cout << ag->operation(x, y);  cout<<endl;  break;  case 2:  ag = new ORGate();  cout << "OR : " << boolalpha << x << ", " << boolalpha << y << " => ";  cout << ag->operation(x, y);  cout<<endl;  break;  case 3:  ag = new XORGate();  cout << "XOR : " << boolalpha << x << ", " << boolalpha << y << " => ";  cout << ag->operation(x, y);  cout<<endl;  break;  }  delete ag;    }  } }; int main() {  Manage::go(); } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 주어진 Shape 클래스를 추상클래스로 만들고, Shape 추상 클래스를 상속받는 파생 클래스를 작성하여 제시된 결과처럼 실행될 수 있도록 프로그램을 완성하시오.

class Shape {

protected:

string name; // 도형의 이름

int width, height; // 도형이 내접하는 사각형

public:

Shape(string n = "", int w = 0, int h = 0) { name = n; width = w; height = h; }

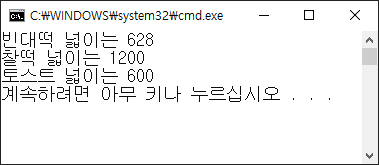
virtual double getArea() {

return 0;

}

string getName() { return name; } // 이름 리턴

};

int main() {

Shape \*p[3];

p[0] = new Oval("빈대떡", 10, 20);

p[1] = new Rect("찰떡", 30, 40);

p[2] = new Triangular("토스트", 30, 40);

for (int i = 0; i < 3; i++)

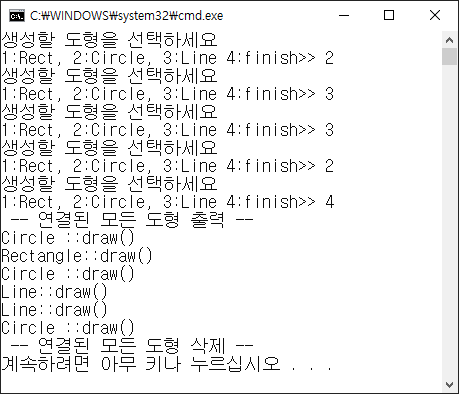
cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;

for (int i = 0; i < 3; i++) delete p[i];

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> #include <string> using namespace std;  class Shape { protected:  string name; // 도형의 이름  int width, height; // 도형이 내접하는 사각형 public:  Shape(string n = "", int w = 0, int h = 0) { name = n; width = w; height = h; }  virtual double getArea() {  return 0;  }  string getName() { return name; } // 이름 리턴 };  class Oval :public Shape{ public:  Oval(string n,int w,int h):Shape(n,w,h){}  double getArea(){  return width\*height\*3.14;  }  }; class Rect : public Shape{ public:  Rect(string n,int w,int h):Shape(n,w,h){}  double getArea(){  return width\*height;  } }; class Triangular : public Shape{ public:  Triangular(string n,int w,int h):Shape(n,w,h){}  double getArea(){  return width\*height/2;  }  };  int main() {  Shape \*p[3];  p[0] = new Oval("빈대떡", 10, 20);  p[1] = new Rect("찰떡", 30, 40);  p[2] = new Triangular("토스트", 30, 40);  for (int i = 0; i < 3; i++)  cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;   for (int i = 0; i < 3; i++) delete p[i]; } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 제시된 결과처럼 실행 될 수 있도록 프로그램을 완성 하시오.

class Graphic {

Graphic \* next;

protected:

virtual void draw();

public:

Graphic() { next = NULL; }

virtual ~ Graphic() { }

void paint();

Graphic \* add(Graphic \* p);

Graphic \* getNext() { return next; }

};

class Circle : public Graphic {

protected:

virtual void draw();

};

class Line : public Graphic {

protected:

virtual void draw();

};

class Rect : public Graphic {

protected:

virtual void draw();

};

int main() {

Graphic \*pStart = NULL;

Graphic \*pLast;

pStart = new Circle(); // 처음에 원 도형을 생성한다.

pLast = pStart;

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Graphic {  Graphic \* next; protected:  virtual void draw()=0; public:  Graphic() { next = NULL; }  virtual ~ Graphic() { }  void paint(){  draw();  }  Graphic \* add(Graphic \* p){  this->next=p;  return p;  }  Graphic \* getNext() { return next; } }; class Circle : public Graphic { protected:  virtual void draw(){  cout<<"Circle :: draw()"<<endl;  } }; class Line : public Graphic { protected:  virtual void draw(){  cout<<"Line :: draw()"<<endl;  } }; class Rect : public Graphic { protected:  virtual void draw(){  cout<<"Rect :: draw()"<<endl;  } };  int main() {  Graphic \*pStart = NULL;  Graphic \*pLast;    pStart = new Circle(); // 처음에 원 도형을 생성한다.  pLast = pStart;   int choice;   while(true){  cout<<"생성할 도형을 선택하세요"<<endl;  cout<<"1:Rect, 2:Circle, 3:Line 4:finish>>";  cin>>choice;  if(choice==1){  pLast=pLast->add(new Rect());  }  else if(choice==2){  pLast=pLast->add(new Circle());  }  else if(choice==3){  pLast=pLast->add(new Line());  }  else  break;  }  cout<<" -- 연결된 모든 도형 출력 --"<<endl;   Graphic \*p=pStart;  while(p!= nullptr){  p->paint();  p=p->getNext();  }   cout<<" -- 연결된 모든 도형 삭제 --"<<endl;  p=pStart;  while(p!= nullptr){  Graphic \*q=p->getNext();  delete p;  p=q;  } } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |