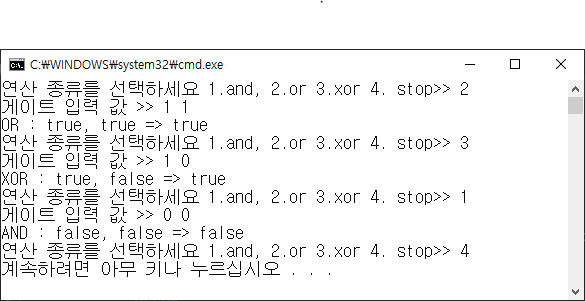
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020\_2\_C++ \_13 | 학번 : | 20175105 | 이름 : | 곽영주 |

* **프로그램 과제**

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>  #include <string>  using namespace std;  class AbstractGate {  public:  virtual bool operation(bool x, bool y) = 0;  };  class ANDGate : public AbstractGate {  public:  virtual bool operation(bool x, bool y) {  if (x == true && y == true)  return true;  else  return false;  }  };  class ORGate : public AbstractGate {  public:  virtual bool operation(bool x, bool y) {  if (x == true || y == true)  return true;  else  return false;  }  };  class XORGate : public AbstractGate {  public:  virtual bool operation(bool x, bool y) {  if ((x == true && y == true) || (x == false && y == false))  return false;  else  return true;  }  };  class Manage {  public:  static void go();  };  void Manage::go() {  AbstractGate\* gate;  int num;  bool i, j;  string gateName;  while (true) {  cout << "연산 종류를 선택하세요 (1.and, 2.or, 3.xor, 4.stop)>> ";  cin >> num;  switch (num) {  case 1:  gate = new ANDGate();  gateName = "AND";  break;  case 2:  gate = new ORGate();  gateName = "OR";  break;  case 3:  gate = new XORGate();  gateName = "XOR";  break;  default:  return;  }  cout << "게이트 입력 값 >> ";  cin >> i >> j;  cout << boolalpha << gateName + " : " << i << ", " << j << " => " << gate->operation(i, j) << endl;  delete gate;  }  }  int main() {  Manage::go();  return 0;  } |
| [실행 결과] |

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; }

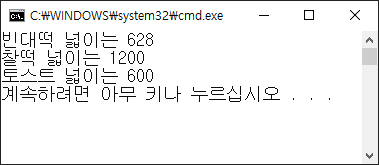
virtual double getArea() {

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++)

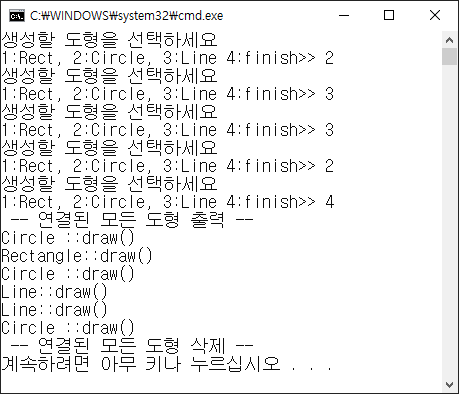
cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;

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

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  #define PI 3.14  using namespace std;  class Shape {  protected:  string name;  int width, height;  public:  Shape(string n = "", int w = 0, int h = 0) { name = n, width = w, height = h; }  virtual double getArea() = 0;  string getName() { return name; }  };  class Oval : public Shape {  public:  Oval(string n = "", int w = 0, int h = 0) : Shape(n, w, h) {}  virtual double getArea() {  return PI \* width \* height;  }  };  class Rect : public Shape {  public:  Rect(string n = "", int w = 0, int h = 0) : Shape(n, w, h) {}  virtual double getArea() {  return (double)width \* height;  }  };  class Triangular : public Shape {  public:  Triangular(string n = "", int w = 0, int h = 0) : Shape(n, w, h) {}  virtual 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];  return 0;  } |
| [실행 결과] |

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 drow() = 0;  public:  Graphic() { next = NULL; }  virtual ~Graphic() { delete next; }  void paint() { drow(); }  Graphic\* add(Graphic\* p);  Graphic\* getNext() { return next; }  };  Graphic\* Graphic::add(Graphic\* p) {  next = p;  return p;  }  class Circle : public Graphic {  protected:  virtual void drow() {  cout << "Circle::drow()" << endl;  }  };  class Line : public Graphic {  protected:  virtual void drow() {  cout << "Line::drow()" << endl;  }  };  class Rect : public Graphic {  protected:  virtual void drow() {  cout << "Rect::drow()" << endl;  }  };  int main() {  Graphic\* pStart = NULL;  Graphic\* pLast;  pStart = new Circle();  pLast = pStart;  int num;  while (true) {  cout << "생성할 도형을 선택하세요." << endl;  cout << "1:Rect, 2:Circle, 3:Line, 4:finish >> ";  cin >> num;  switch (num) {  case 1:  pLast = pLast->add(new Rect());  break;  case 2:  pLast = pLast->add(new Circle());  break;  case 3:  pLast = pLast->add(new Line());  break;  default:  num = 4;  break;  }  if (num == 4)  break;  }  Graphic\* p = pStart;  cout << "-- 연결된 모든 도형 출력 --" << endl;  while (p != NULL) {  p->paint();  p = p->getNext();  }    p = pStart;  cout << "-- 연결된 모든 도형 삭제 --" << endl;  while (p != NULL) {  Graphic\* q = p->getNext();  delete p;  p = q;  }  return 0;  } |
| [실행 결과] |