|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cpp\_14주 | 학번 : | 20175260 | 이름 : | 최영빈 |

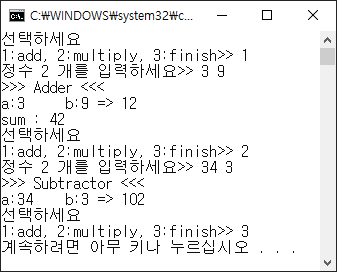
* **개념 적용 응용 프로그래밍 – 가상 함수와 추상클래스**

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();

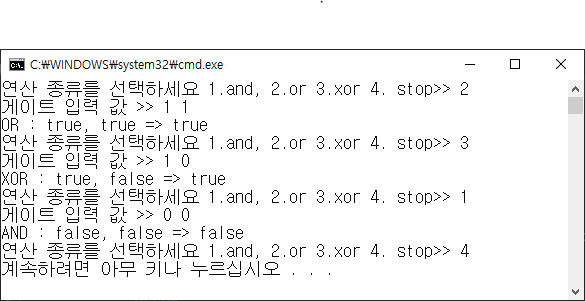
}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  #include <typeinfo>  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(); // Adder 구현 시 추가  };  int Adder::sum(){  int sum =0;  for(int i =a; i <= b; i++)  sum+= i;  return sum;  }  class Mul : public Calculator{  public:  Mul(int a,int b):Calculator(a,b){}  int calc(){return a\*b;}  void write(){  cout << ">>> Mul <<<" << endl;  Calculator::write();  }  };  class Manage{  public:  static void run();  };  void Manage::run(){  Calculator \*c = nullptr;  int a, b;  int count;    while (true)  {  cout << "선택하세요" << endl;  cout << "1:add, 2:multiply, 3:finish>>";  cin >> count;  if(count == 3)  break;  cout<< "정수 2 개를 입력하세요 >>";  cin >> a>>b;  switch (count)  {  case 1:  c = new Adder(a,b);  c->write();  cout << c->calc()<<endl<<"sum : ";  if(typeid(\*c) == typeid(Adder)){  Adder \*ac = dynamic\_cast<Adder \*>(c);  cout <<ac->sum()<<endl;  }  break;  case 2:  c = new Mul(a,b);  c->write();  cout <<c->calc()<<endl;  break;  }  }  delete c;  }  int main()  {  Manage::run();  } |
| [실행 결과]  이미지 |

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; // 순수 가상 함수  };  class ANDGate : public AbstractGate{  public:  bool operation(bool x, bool y){  return x&&y;  }  };  class ORGate : public AbstractGate  {  public:  bool operation(bool x, bool y){  return x||y;  }  };  class XORGate : public AbstractGate  {  public:  bool operation(bool x, bool y){  return x^y;  }  };  class Manage{  public:  static void go(){  AbstractGate \*ag = nullptr;  int a = 0;  bool x, y;  while (true)  {  cout << "연산 종류를 선택하세요 1.and, 2.or, 3.xor, 4. stop >>";  cin >> a;  if(a == 4){  break;  }  cout << "게이트 입력 값 >> ";  cin >> x >> y;  switch (a)  {  case 1:  ag = new ANDGate();  cout << "AND : " << boolalpha << x << " " << y << " => " << ag->operation(x, y) << endl;  break;  case 2:  ag = new ORGate();  cout << "OR : " << boolalpha << x << " " << y << " => " << ag->operation(x, y) << endl;  break;  case 3:  ag = new XORGate();  cout << "XOR : " << boolalpha << x << " " << y << " => " << ag->operation(x, y) << endl;  break;  }  delete ag;  }  }  };  int main()  {  Manage::go();  } |
| [실행 결과]  이미지 |

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; }

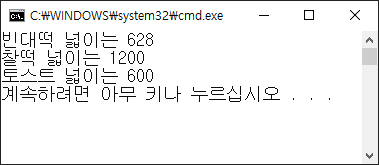
virtual double getArea() {

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++)

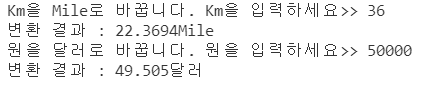
cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;

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 b,int w, int h):Shape(b,w,h){}  double getArea(){return 3.14\*width\*height;}  };  class Rect:public Shape{  public:  Rect(string b, int w, int h) : Shape(b, w, h) {}  double getArea(){return width\*height;}  };  class Triangular:public Shape{  public:  Triangular(string b, int w, int h) : Shape(b, 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]  이미지 |

1. 단위를 변환하는 Converter 클래스와 main()을 참고하여 프로그램을 완성하시오.



class Converter {

protected:

  double ratio;

  virtual double convert(double src)=0; // src를 다른 단위로 변환한다.

  virtual string getSourceString()=0; // 소스 단위 명칭

  virtual string getDestString()=0; // dest 단위 명칭

public:

  Converter(double ratio) { this->ratio = ratio; }

  void run() {

    double src;

    cout << getSourceString() << "을 " << getDestString() << "로 바꿉니다. ";

    cout << getSourceString() << "을 입력하세요>> ";

    cin >> src;

    cout << "변환 결과 : " << convert(src) << getDestString() << endl;

  }

};

int main() {

  KmToMile toMile(1.609344); // 1mile은 1.609344 Km

  WonToDollar wd(1010); // 1 달러에 1010원

  Converter \*cp = &toMile;

  cp->run();

  cp = &wd;

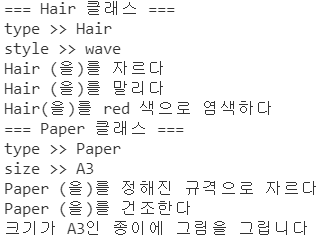
  cp->run();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Converter  {  protected:  double ratio;  virtual double convert(double src) = 0; // src를 다른 단위로 변환한다.  virtual string getSourceString() = 0; // 소스 단위 명칭  virtual string getDestString() = 0; // dest 단위 명칭  public:  Converter(double ratio) { this->ratio = ratio; }  void run()  {  double src;  cout << getSourceString() << "을 " << getDestString() << "로 바꿉니다. ";  cout << getSourceString() << "을 입력하세요>> ";  cin >> src;  cout << "변환 결과 : " << convert(src) << getDestString() << endl;  }  };  class KmToMile:public Converter{  protected:  double convert(double src){  return src / 1.609;  ;  }  string getSourceString(){  return "km";  }  string getDestString(){  return "Mile";  }  public:  KmToMile(double ratio):Converter(ratio){}  };  class WonToDollar:public Converter{  protected:  double convert(double src){  return src/1010;  }  string getSourceString(){  return "원";  }  string getDestString(){  return "달러";  }  public:  WonToDollar(double ratio) : Converter(ratio) {}  };  int main()  {  KmToMile toMile(1.609344); // 1mile은 1.609344 Km  WonToDollar wd(1010); // 1 달러에 1010원  Converter \*cp = &toMile;  cp->run();  cp = &wd;  cp->run();  } |
| [실행 결과]  이미지 |

1. 제시된 코드와 실행 결과를 참조하여 프로그램을 완성 하시오.

class Tool{

  string type;

public:

  Tool() = default;

  Tool(string type) : type(type) {}

  string getType(){

    return type;

  }

  virtual void write(){

    cout << "type >> " << type << endl;

  }

  virtual void cut() = 0; //자르다

  virtual void dry() = 0; //말리다

};

1) Tool 클래스를 상속 받는 Hair 클래스 구현

* 헤어 스타일을 저장하는 필드 style 추가 , private
* 생성자 - 매개변수로 필드 초기화
* void tint(String color) – 기본 클래스 type을 color로 염색하는 함수 추가
* 기본클래스 write() 재정의 - type과 style 출력, 단, type은 기본클래스 write() 사용

2) Tool 클래스를 상속 받는 Paper 클래스 구현

* 용지 크기를 저장하는 필드 size 추가
* 생성자 - 매개변수로 필드 초기화
* void draw() – 기본 클래스 type에 그림을 그리는  메소드
* 기본 클래스 write() 재정의 - type과 size 출력, 단, type은 기본클래스 write() 사용

3) 전역함수 show 구현

int main(){

  Hair h("Hair", "wave");

  Paper p("Paper", "A3");

  show(h);

  show(p);

    return 0;

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Tool  {  string type;  public:  Tool() = default;  Tool(string type) : type(type) {}  string getType()  {  return type;  }  virtual void write()  {  cout << "type >> " << type << endl;  }  virtual void cut() = 0;//자르다  virtual void dry() = 0; //말리다  };  void Tool::cut(){  return;  }  void Tool::dry(){  return;  }  class Hair:public Tool{  string style;  public:  Hair(string a, string b):Tool(a){  style = b;  }  void write(){  cout <<getType()<< " >> " << style <<endl;  }  void tint(string color){  cout <<getType()<< "을(를) "<<color <<" 색으로 염색하다 "<< endl;  }  void cut(){  cout << getType()<<" (을)를 자르다"<<endl;  }  void dry(){  cout <<getType()<< " (을)를 말리다" <<endl;  }  };  class Paper:public Tool{  string size;  public:  Paper(string a,string b):Tool(a){  size = b;  }  virtual void write()  {  cout << getType() << " >> " << size << endl;  }  void draw(){  cout << "크기가 "<<size<<"인 "<< getType() <<"에 그림을 그립니다"<<endl;  }  void cut()  {  cout <<getType() << " (을)를 정해진 구격으로 자르다" << endl;  }  void dry()  {  cout << getType() <<" (을)를 건조한다" << endl;  }  };  void show(Tool& a){  if(typeid(a) == typeid(Hair)){  Hair& b = dynamic\_cast<Hair& >(a);  cout << "=== " << b.getType() << "클래스 "<< "===" << endl;  b.Tool::write();  b.write();  b.cut();  b.dry();  b.tint("red");  }  else{  Paper& c = dynamic\_cast<Paper& >(a);  cout << "=== " << c.getType() << "클래스 "<< "===" << endl;  c.Tool::write();  c.write();  c.cut();  c.dry();  c.draw();  }  }  int main()  {  Hair h("Hair", "wave");  Paper p("Paper", "A3");  show(h);  show(p);  return 0;  } |
| [실행 결과]  이미지 |