REPORT #6



|  |
| --- |
| **제 출 일** : 2017년 11월 28일 |
| **과 목 명** : 컴퓨터프로그래밍및실습 |
| **담당교수 : 김정국 교수님** |
| **학 과** : 컴퓨터전자시스템 |
| **학 번** : 201703091 |
| **성 명 : 전기범** |

**HANKUK UNIVERSITY OF FOREIGN STUDIES**



**1 . 가상 함수의 예제 1**

**1.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Figure // Figure 클래스  {  protected:  int x, y; // x, y 좌표  public:  virtual void draw(); // 가상 함수 사용 한 draw  void setOrigin(int x, int y); // x, y 값 설정  };  void Figure::draw()  {  cout << "Figure Draw" << endl;  }  void Figure::setOrigin(int x, int y)  {  this->x = x;  this->y = y;  }  class Rectangle :public Figure // Rectangle 클래스, Figure 클래스 상속 받음  {  private:  int width, height; // 높이, 폭  public:  void setWidth(int w); // 폭 설정  void setHeight(int h); // 높이 설정  void draw();  };  void Rectangle::setWidth(int w)  {  width = w;  }  void Rectangle::setHeight(int h)  {  height = h;  }  void Rectangle::draw()  {  cout << "Rectangle Draw" << endl;  }  class Circle :public Figure // Circle 클래스, Figure 클래스 상속 받음  {  private:  int radius; // 반지름  public:  void setRadius(int r); // 반지름 설정  void draw();  };  void Circle::setRadius(int r)  {  radius = r;  }  void Circle::draw()  {  cout << "Circle draw" << endl;  }  int main()  {  Figure \*ps = new Rectangle(); // 생성  ps->draw(); // Rectangle::draw(); 호출  delete ps; // 반납  Figure \*ps2 = new Circle(); // 생성  ps2->draw(); // Circle::draw(); 호출  delete ps2; // 반납  } |

**1.2 결과 화면**

|  |
| --- |
|  |

**2 . 가상 함수를 사용하지 않았을 때의 예제**

**2.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Figure // Figure 클래스  {  protected:  int x, y; // x, y 좌표  public:  void draw(); // 가상 함수 사용 안한 draw  void setOrigin(int x, int y); // x, y 값 설정  };  void Figure::draw()  {  cout << "Figure Draw" << endl;  }  void Figure::setOrigin(int x, int y)  {  this->x = x;  this->y = y;  }  class Rectangle :public Figure // Rectangle 클래스, Figure 클래스 상속 받음  {  private:  int width, height; // 높이, 폭  public:  void setWidth(int w); // 폭 설정  void setHeight(int h); // 높이 설정  void draw();  };  void Rectangle::setWidth(int w)  {  width = w;  }  void Rectangle::setHeight(int h)  {  height = h;  }  void Rectangle::draw()  {  cout << "Rectangle Draw" << endl;  }  class Circle :public Figure // Circle 클래스, Figure 클래스 상속 받음  {  private:  int radius; // 반지름  public:  void setRadius(int r); // 반지름 설정  void draw();  };  void Circle::setRadius(int r)  {  radius = r;  }  void Circle::draw()  {  cout << "Circle draw" << endl;  }  int main()  {  Figure \*ps = new Rectangle(); // 생성  ps->draw(); // Figure::draw(); 호출  delete ps; // 반납  Figure \*ps2 = new Circle(); // 생성  ps2->draw(); // Figure::draw(); 호출  delete ps2; // 반납  } |

**2.2 결과 화면**

|  |
| --- |
|  |

**3 . 참조자와 가상 함수의 예제**

**3.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Animal // 동물 클래스  {  public:  virtual void speak(); // 가상 함수  };  void Animal::speak()  {  cout << "Animal speak()" << endl;  }  class Dog :public Animal // 개 클래스, 동물 클래스 상속  {  public:  void speak();  };  void Dog::speak()  {  cout << "멍멍!" << endl;  }  class Cat :public Animal // 고양이 클래스, 동물 클래스 상속  {  public:  void speak();  };  void Cat::speak()  {  cout << "야옹~" << endl;  }  int main()  {  Dog d; // 개 클래스 d생성  Animal &a1 = d; // 동물 클래스 a1 = d  a1.speak(); // a1.speak() 호출 = d.speak()호출  Cat c; // 고양이 클래스 c생성  Animal &a2 = c; // 동물 클래스 a2 = c  a2.speak(); // a2.speak() 호출 = c.speak()호출  } |

**3.2 결과 화면**

|  |
| --- |
|  |

**4 . 가상 소멸자 예제**

**4.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Animal // 동물 클래스  {  public:  Animal(); // 생성자  virtual ~Animal(); // 가상 소멸자  virtual void speak(); // 가상 함수  };  Animal::Animal()  {  cout << "Animal 생성자" << endl;  }  Animal::~Animal()  {  cout << "Animal 소멸자" << endl;  }  void Animal::speak()  {  cout << "Animal speak()" << endl;  }  class Dog :public Animal // 개 클래스, 동물 클래스 상속  {  public:  Dog(); // 생성자  ~Dog(); // 소멸자  void speak();  };  Dog::Dog()  {  cout << "Dog 생성자" << endl;  }  Dog::~Dog()  {  cout << "Dog 소멸자" << endl;  }  void Dog::speak()  {  cout << "멍멍!" << endl;  }  int main()  {  Animal \*a1 = new Dog(); // Dog 생성  a1->speak(); // Dog::speak() 호출  delete a1;  } |

**4.2 결과 화면**

|  |
| --- |
|  |

**5 . 순수 가상 함수 예제 1**

**5.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Figure // Figure 클래스  {  protected:  int x, y; // x, y 좌표  public:  void setOrigin(int x, int y); // x, y 값 설정  virtual void draw() = 0; // 순수 가상 함수 사용 한 draw  };  void Figure::setOrigin(int x, int y)  {  this->x = x;  this->y = y;  }  class Circle :public Figure // Circle 클래스, Figure 클래스 상속 받음  {  private:  int radius; // 반지름  public:  void setRadius(int r); // 반지름 설정  void draw();  };  void Circle::setRadius(int r)  {  radius = r;  }  void Circle::draw()  {  cout << "Circle draw" << endl;  }  int main()  {  Figure \*ps = new Circle(); // 생성  ps->draw(); // 실제 객체의 draw() 호출 = Circle::draw() 호출  delete ps; // 반납  } |

**5.2 결과 화면**

|  |
| --- |
|  |

**6 . 프렌드 함수 예제 1**

**6.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Company // 회사 클래스  {  private:  int sales, profit;  friend void sub(Company& c); // 프렌드 선언  public:  Company();  };  Company::Company()  {  sales = 0;  profit = 0;  }  void sub(Company& c)  {  cout << "profit = " << c.profit << endl; // sub함수는 Company의 private부분에 접근 가능  }  int main()  {  Company c1;  sub(c1); // profit = 0  } |

**6.2 결과 화면**

|  |
| --- |
|  |

**7 . 프렌드 함수 예제 2**

**7.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Date  {  friend bool equals(Date d1, Date d2); // 프렌드 함수 선언  private:  int year, month, day; // 년,월, 일  public:  Date(int y, int m, int d);  };  Date::Date(int y, int m, int d)  {  year = y;  month = m;  day = d;  }  bool equals(Date d1, Date d2)  {  return d1.year == d2.year&&d1.month == d2.month&&d1.day == d2.day; // 두 날짜가 같으면 1 아니면 0 반환  }  int main()  {  Date d1(1975, 5, 22), d2(2002, 8, 12);  if (equals(d1, d2) == 1)  cout << "두 날짜는 같다!" << endl;  else  cout << "두 날짜는 다르다!" << endl;  } |

**7.2 결과 화면**

|  |
| --- |
|  |

**8 . 프렌드 함수 예제 3**

**8.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Complex // 복소수 클래스  {  private:  int re, im; // 실수, 허수  public:  friend Complex add(Complex a1, Complex a2); // 프렌드 함수 선언, 두 객체 선언  Complex();  Complex(double r);  Complex(double r, double i);  void output();  };  Complex::Complex()  {  re = 0;  im = 0;  }  Complex::Complex(double r)  {  re = r;  im = 0;  }  Complex::Complex(double r, double i)  {  re = r;  im = i;  }  void Complex::output()  {  cout << re << " + " << im << "i" << endl;  }  Complex add(Complex a1, Complex a2) // 프렌드 함수 정의  {  return Complex(a1.re + a2.re, a1.im + a2.im); // 실수부, 허수부 따로 계산 후 값 반환  }  int main()  {  Complex c1(4, 6), c2(5, 8);  Complex c3 = add(c1, c2); // 계산  c3.output(); // 출력  } |

**8.2 결과 화면**

|  |
| --- |
|  |

**9 . 연산자 중복 예제 1**

**9.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Vector // 벡터 클래스  {  private:  double x, y;  public:  Vector(double xvalue = 0.0, double yvalue = 0.0) :x(xvalue), y(yvalue){} // 기본 초기화  void display() // 출력  {  cout << "(" << x << "," << y << ")" << endl;  }  Vector operator+(Vector& v2) // 멤버 함수로 연산자 중복, +연산자 함수 정의  {  Vector v;  v.x = this->x + v2.x;  v.y = this->y + v2.y;  return v;  }  };  int main()  {  Vector v1(1.0, 2.0), v2(3.0, 4.0);  Vector v3 = v1 + v2;  v3.display();  } |

**9.2 결과 화면**

|  |
| --- |
|  |

**10 . 연산자 중복 예제 2**

**10.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Vector // 벡터 클래스  {  friend Vector operator+(const Vector& v1, const Vector& v2); // 전역 함수로 연산자 함수 선언  private:  double x, y;  public:  Vector(double xvalue = 0.0, double yvalue = 0.0) :x(xvalue), y(yvalue){} // 기본 초기화  void display() // 출력  {  cout << "(" << x << "," << y << ")" << endl;  }  };  Vector operator+(const Vector& v1, const Vector& v2) // 전역 함수로 +연산자 함수 구현  {  Vector v(0.0, 0.0);  v.x = v1.x + v2.x;  v.y = v1.y + v2.y;  return v;  }  int main()  {  Vector v1(1.0, 2.0), v2(3.0, 4.0);  Vector v3 = v1 + v2;  v3.display();  } |

**10.2 결과 화면**

|  |
| --- |
|  |

**11 . \*연산자 중복 예제**

**11.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Vector // 벡터 클래스  {  friend Vector operator\*(Vector& v, double alpha); // 전역 함수로 연산자 함수 선언  friend Vector operator\*(double alpha, Vector& v);  private:  double x, y;  public:  Vector(double xvalue = 0.0, double yvalue = 0.0) :x(xvalue), y(yvalue){} // 기본 초기화  void display() // 출력  {  cout << "(" << x << "," << y << ")" << endl;  }  };  Vector operator\*(Vector& v, double alpha) // \*연산자 함수 정의  {  return Vector(alpha\*v.x, alpha\*v.y);  }  Vector operator\*(double alpha, Vector& v)  {  return Vector(alpha\*v.x, alpha\*v.y);  }  int main()  {  Vector v(1.0, 1.0);  Vector w = v\*2.0;  Vector z = 2.0\*v;  w.display();  z.display();  } |

**11.2 결과 화면**

|  |
| --- |
|  |

**12 . ==연산자 중복 예제**

**12.1. 소스 코드**

|  |
| --- |
| #include <iostream>  using namespace std;  class Vector // 벡터 클래스  {  private:  double x, y;  public:  Vector(double xvalue = 0.0, double yvalue = 0.0) :x(xvalue), y(yvalue){} // 기본 초기화  void display() // 출력  {  cout << "(" << x << "," << y << ")" << endl;  }  friend bool operator==(const Vector& v1, const Vector& v2);  friend bool operator!=(const Vector& v1, const Vector& v2);  };  bool operator==(const Vector& v1, const Vector& v2) // ==연산자 함수 정의, 전역 함수로 구현  {  return v1.x == v2.x&&v1.y == v2.y; // 같으면 1 반환  }  bool operator!=(const Vector& v1, const Vector& v2)  {  return !(v1 == v2); // 다르면 1 반환  }  int main()  {  Vector v1(1, 2), v2(1, 2);  Vector v3(5, 6), v4(7, 8);  if (v1 == v2)  cout << "true" << endl;  if (v3 != v4)  cout << "false" << endl;  } |

**12.2 결과 화면**

|  |
| --- |
|  |