객체지향프로그래밍 (202004100 장주훈, 202084023 이병길)

**9주차** - 프렌드와 연산자 중복

텍스트, 화이트보드, 친필이(가) 표시된 사진

자동 생성된 설명

Q1. 5가지중 올바른 것은?

|  |
| --- |
| 1. Power& ret\_This(){ return \* this; } 2. Power& ret\_This(){ return \* this; } 3. Power& ret\_This(){ return \* this; } 4. Power ret\_This(){ return \* this; } 5. Power ret\_This(){ return this; } |
| Answer: 호출 시 리턴 타입이 일치하는지를 보면 됌.  This는 Rect라는 클래스가 있었고, 객체 a가 있음  메모리공간에 a라는 객체생성  This는 객체 자신을 가리키는 포인터(객체의 주소)  int \*a, b;  a=&b;  a= b의 주소, \*a = b의 주소가 가리키고 있는 값   1. Power& ret\_This(){ return \* this; }   올바른 함수, Power&(객체의 참조 = 원본) 과 \*this 주소의 값이므로 리턴 타입 같음.   1. Power ret\_This(){ return \* this; }   올바름. Power(복사생성자(객체)) \*this는 주소가 가지고 있는 값. 따라서, 리턴 값이 리턴이므로 리턴 타입 같음. But, 원본 파워에 대한 복사본을 만들어내기 때문에 컴파일은 가능하지만 원하는 바는 아닐 수 있음.(주의!)   1. Power& ret\_This(){ return this; }   컴파일 오류 발생. Power&(객체의 참조 - 원본), this (주소 값)   1. Power\* ret\_This(){ return this; }   올바른 함수. Power은 power의 주소, this도 주소 따라서 올바름.   1. Power ret\_This(){ return this; }   컴파일 오류 발생. This는 주소값을 반환 따라서, power의 객체와는 다름. |

프렌드 함수: 클래스 멤버 함수가 아닌데 모든 멤버를 접근할 수 있는 권한 부여

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

프렌드 함수가 되는 3 가지

* 전역 함수: 클래스 외부에 선언된 전역 함수
* 다른 클래스의 멤버 함수: 다른 클래스의 특정 멤버 함수
* 다른 클래스 전체: 다른 클래스의 모든 멤버 함수

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

실습 예제 7-1 전역 함수

|  |
| --- |
| #include <iostream>  using namespace std;  class Rect;  bool equals(Rect r, Rect s); // equals() 함수 선언  class Rect { // Rect 클래스 선언  int width, height; //private  public:  Rect(int width, int height) { this->width = width; this->height = height; }  friend bool equals(Rect r, Rect s);  };  bool equals(Rect r, Rect s) { // 외부 함수  if (r.width == s.width && r.height == s.height) return true;  //equals() 함수는 private 속성을 가진 width, height에 접근 가능(why? 프렌즈 함수)  else return false;  }  int main() {  Rect a(3, 4), b(3, 4);  if (equals(a, b)) cout << "equal" << endl;  else cout << "not equal" << endl;  } |

실습예제 7-2 다른 클래스의 멤버 함수

|  |
| --- |
| #include <iostream>  using namespace std;  class Rect; // 참조오류를 막는 선언문  bool equals(Rect r, Rect s); // equals() 함수 선언  class RectManager {  public:  bool equals(Rect r, Rect s);  };  class Rect { // Rect 클래스 선언  int width, height; //private  public:  Rect(int width, int height) { this->width = width; this->height = height; }  friend bool RectManager::equals(Rect r, Rect s);  };  bool RectManager::equals(Rect r, Rect s) {  if (r.width == s.width && r.height == s.height) return true;  //RectManager 클래스의 equals() 멤버를 프렌드로 선언  else return false;  }  int main() {  Rect a(3, 4), b(3, 4);  RectManager man;  if (man.equals(a, b)) cout << "equal" << endl;  else cout << "not equal" << endl;  } |

실습 예제 7-3 다른 클래스 전체

|  |
| --- |
| #include <iostream>  using namespace std;  class Rect;  class RectManager {  public:  bool equals(Rect r, Rect s);  void copy(Rect& dest, Rect& src);  };  class Rect { // Rect 클래스 선언  int width, height; //private  public:  Rect(int width, int height) { this->width = width; this->height = height; }  friend RectManager; //RectManager 클래스를 프렌드 함수로 선언  };  bool RectManager::equals(Rect r, Rect s) {  if (r.width == s.width && r.height == s.height) return true;  else return false;  }  void RectManager::copy(Rect& dest, Rect& src) {  dest.height = src.height;  dest.width = src.width;  }  int main() {  Rect a(3, 4), b(5, 6);  RectManager man;    man.copy(b, a);  if (man.equals(a, b))  cout << "equal" << endl;  else  cout << "not equal" << endl;  } |

연산자 중복

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

특징  
텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

연산자 함수 구현 방법 2가지

1. 클래스의 멤버 함수로 구현

2. 외부 함수로 구현하고 클래스에 프렌드 함수로 선언

|  |
| --- |
| Color a(BLUE), b(RED), c; c = a + b; // a와 b를 더하기 위한 + 연산자 작성 필요  //객체 a + 객체 b는 오류 발생. 따라서, 선언이 필요함.  if(a == b) { // a와 b를 비교하기 위한 == 연산자 작성 필요 ... } |
|  |

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

실습 예제 7-4

|  |
| --- |
| #include <iostream>  using namespace std;  class Power {  int kick;  int punch;  public:  Power(int kick = 0, int punch = 0) {  this->kick = kick; this->punch = punch;  }  void show();  Power operator+ (Power op2); // + 연산자 함수 선언  };  void Power::show() {  cout << "kick=" << kick << ',' << "punch=" << punch << endl;  }  Power Power::operator+(Power op2) {  Power tmp; // 임시 객체 생성  tmp.kick = this->kick + op2.kick; // kick 더하기  tmp.punch = this->punch + op2.punch; // punch 더하기  return tmp; // 더한 결과 리턴  }  int main() {  Power a(3, 5), b(4, 6), c;  c = a + b; // 파워 객체 + 연산  a.show();  b.show();  c.show();  } |

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**시험! 코드를 보고 결과 값을 맞춰라.**

**Call by value는 복사본 생성 후 진행 call by reference는 원본 참조 후 진행 이므로**

**원본이 바뀌는 경우에 대해 자세히 알기!**

**텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명**

b = a++; a=0일 때, a=1, b=0 |||||| b = ++a; a=0일 때, a=1, b=1 이므로

++a는 power 원본을 참조 a++은 변경되기 전의 나의 값을 복사해서 넘겨야 되므로 복사본을 만들고, 변경 전 값을 return해줌

Q. 예제 7-12를 c=a.+(b)로 변경해보고

Q. c=+(a,b)로 변경해보기

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명