|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_07 | 학번 : |  | 이름 : |  |

* **강의 내용**

|  |
| --- |
| 1. 프렌드 함수1 : 외부 함수를 프렌드로 선언 |
| #include <iostream>  using namespace std;  //Rect 클래스가 선언되기 전에 먼저 참조되는 컴파일 오류(forward reference)를 막기 위한 선언문  class Rect;  bool equals(Rect r, Rect s); // equals() 함수 선언  class Rect {  int width, height;  public:  Rect(int width, int height) { this->width = width; this->height = height; }  //equals() 함수를 프렌드로 선언  //equals() 함수는 private 속성을 가진 width, height에 접근할 수 있다  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;  else  return false;  }  int main() {  Rect a(3, 4), b(4, 5);  if (equals(a, b)) cout << "equal" << endl;  else cout << "not equal" << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 프렌드 함수2: 다른 클래스 전체를 프렌드로 선언 |
| #include <iostream>  using namespace std;  class Rect;  class RectManager { // RectManager 클래스 선언  public:  bool equals(Rect r, Rect s);  void copy(Rect& dest, Rect& src);  };  class Rect { // Rect 클래스 선언  int width, height;  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) { // r과 s가 같으면 true 리턴  if (r.width == s.width && r.height == s.height)  return true;  else  return false;  }  void RectManager::copy(Rect& dest, Rect& src) { // src를 dest에 복사  dest.width = src.width;  dest.height = src.height;  }  int main() {  Rect a(3, 4), b(5, 6);  RectManager man;  man.copy(b, a); // a를 b에 복사한다.  if (man.equals(a, b))  cout << "equal" << endl;  else  cout << "not equal" << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 연산자 중복 정의 – 멤버 함수로 구현 |
| #include <iostream>  using namespace std;  class Power {  int kick, punch;  public:  Power(int kick = 0, int punch = 0) {  this->kick = kick;  this->punch = punch;  }  void show();  Power operator+ (Power op2); // + 연산자 함수 선언, Power& op2 로 해도 됨  //또는 Power operator+ (const Power &op2);  //참조 매개변수를 사용시 원본 객체의 수정을 방지하기 위해 const로 함  };  void Power::show() {  cout << "kick=" << kick << ',' << "punch=" << punch << endl;  }  //+ 연산자 멤버 함수 구현  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(20, 20), b(12, 30);  Power c = a + b;  a.show();  b.show();  c.show();  return 0;  } |
| **[실행결과] – 연산자 멤버 함수를 주석처리 후 결과도 확인 할 것** |

|  |
| --- |
| 1. 연산자 중복 정의 : 프렌드 함수 사용 |
| #include <iostream>  using namespace std;  class Power {  int kick, punch;  public:  Power(int kick = 0, int punch = 0) { this->kick = kick; this->punch = punch; }  void show();  friend Power operator+(int op1, Power op2); // 프렌드 선언  friend Power operator+(Power op1, Power op2); // 프렌드 선언  };  void Power::show() {  cout << "kick=" << kick << ',' << "punch=" << punch << endl;  }  //+ 연산자 함수를 외부 함수로 구현  //private 속성인 kick, punch를 접근하도록 하기 위해, 연산자 함수를 friend로  //선언해야 함  Power operator+(int op1, Power op2) {  Power tmp; // 임시 객체 생성  tmp.kick = op1 + op2.kick; // kick 더하기  tmp.punch = op1 + op2.punch; // punch 더하기  return tmp; // 임시 객체 리턴  }  Power operator+(Power op1, Power op2) {  Power tmp; // 임시 객체 생성  tmp.kick = op1.kick + op2.kick; // kick 더하기  tmp.punch = op1.punch + op2.punch; // punch 더하기  return tmp; // 임시 객체 리턴  }  int main() {  Power a(3, 5), b;  a.show();  b.show();  //operator+(2, a) 함수 호출  b = 2 + a; // 파워 객체 더하기 연산  b.show();  //operator+(a,b) 함수 호출  b = a + b;  b.show();  } |
| **[실행결과]** |

|  |
| --- |
| 1. 연산자 중복 정의 : 객체 참조 리턴 |
| #include <iostream>  using namespace std;  class Power {  int kick, punch;  public:  Power(int kick = 0, int punch = 0) { this->kick = kick; this->punch = punch; }  void show();  Power& operator << (int n); // 연산 후 Power 객체의 참조 리턴  };  void Power::show() {  cout << "kick=" << kick << ',' << "punch=" << punch << endl;  }  //참조 리턴  Power& Power::operator <<(int n) {  kick <<= n;  punch <<= n;  return \*this; // 이 객체의 참조 리턴  }  int main() {  Power a(1, 2);  a << 3 << 5 << 6;  a.show();  } |
| **[실행결과]** |

* 내용 점검 – 7장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |

* 프로그램 과제

|  |
| --- |
| 1. 교재 367p : 문제1~4번 |
| **[프로그램 소스 1]** |
| **[프로그램 소스 2]** |
| **[프로그램 소스 3]** |
| **[프로그램 소스 4]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 369p : 문제7번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제10번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제12번 |
| **[프로그램 소스]** |
| **[실행결과]** |