|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2018\_2\_C++ \_09 | 학번 : |  | 이름 : |  |

1. 프렌드 함수1 - 외부 함수를 프렌드로 선언

|  |
| --- |
| //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; } |

* 프렌드 함수2 - 다른 클래스의 멤버 함수를 프렌드로 선언

|  |
| --- |
| class Rect;  **class RectManager** { // RectManager 클래스 선언  public:  **bool equals(Rect r, Rect s);** };  **class Rect** { // Rect 클래스 선언  int width, height;  public:  Rect(int width, int height) { this->width = width; this->height = height; }  **//RectManager 클래스의 equals() 멤버를 프렌드로 선언**  **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;  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; } |

* 프렌드 함수3 - 다른 클래스 전체를 프렌드로 선언

|  |
| --- |
| 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;  } |

* 연산자 중복 정의 - 멤버 함수로 구현

***연산자 중복을 정의 한 함수를 주석으로 처리한 후 결과를 확인해 볼 것!!!!!!!***

|  |
| --- |
| 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  //참조 매개변수를 사용하면 원본 객체를 수정할 수 있기 때문에 주의해야 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  } |
| **[실행 결과] - 연산자 중복을 정의한 함수를 주석 처리한 후 결과** |

* 연산자 중복 정의 - 프렌드 함수 사용

|  |
| --- |
| 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**  };  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  }  int main() {  Power a(3,5), b;  a.show();  b.show();  **//operator+(2, a) 함수 호출**  **b = 2 + a; // 파워 객체 더하기 연산**  a.show();  b.show();  } |

* 내용 점검

1. 교재 7장 연습문제

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

* 프로그램 과제

1. 교재 316p : 문제6번

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class ArrayUtility2 {  public:  static int\* concat(int s1[], int s2[], int size); // s1과 s2를 연결한 새로운 배열을 동적 생성하고 포인터 리턴  static int\* remove(int s1[], int s2[], int size, int&retSize); // s1에서 s2에 있는 숫자를 모두 삭제한 새로운 배열을 동적 생성하여 리턴  };  int\* ArrayUtility2::concat(int s1[], int s2[], int size) {  int\* p = new int [size\*2];  if(!p)  return NULL;  for(int i=0; i<size; i++) p[i] = s1[i];  for(int i=0; i<size; i++) p[i+size] = s2[i];  return p;  }  int\* ArrayUtility2::remove(int s1[], int s2[], int size, int&retSize) {  int\* p = new int [size\*2];  if(!p) {  retSize = 0;  return NULL;  }  int k=0;  for(int i=0; i<size; i++) {  int j;  for(j=0; j<size; j++) {  if(s1[i] == s2[j])  break;  }  if(j==size) { // not found  p[k] = s1[i];  k++;  }  }  retSize = k;  if(k==0)  return NULL;  int\* q = new int [retSize]; // 결정된 retSize로 동적 배열 다시 할당  if(!q) {  retSize = 0;  return NULL;  }  for(int i=0; i<retSize; i++) q[i] = p[i]; // p[] 배열을 q[]배열에 복사  delete [] p; // 동적 배열 반환  return q;  }  int main() {  int x[5], y[5];  cout << "정수를 5 개 입력하라. 배열 x에 삽입한다>>";  for(int i=0; i<5; i++) cin >> x[i];    cout << "정수를 5 개 입력하라. 배열 y에 삽입한다>>";  for(int i=0; i<5; i++) cin >> y[i];  int\* p = ArrayUtility2::concat(x, y, 5);  cout << "합친 정수 배열을 출력한다" << endl;  for(int i=0; i<10; i++) cout << p[i] << ' ';  cout << endl;  int retSize;  int\* q = ArrayUtility2::remove(x, y, 5, retSize);  cout << "배열 x[]에서 y[]를 뺀 결과를 출력한다. 개수는 " << retSize << endl;  for(int i=0; i<retSize; i++) cout << q[i] << ' ';  cout << endl;  delete [] p;  delete [] q;  } |
| **[실행 결과]** |

1. 교재 318p : 문제9번

|  |
| --- |
| **[프로그램 소스]**  #ifndef BOARD\_H  #define BOARD\_H  #include <string>  using namespace std;  class Board {  static int size;  static string text[100];  Board() {} // private으로 선언하여 외부에서 Board 객체를 생성할 수 없도록 함  public:  static void add(string text);  static void print();  };  #endif  #include <iostream>  #include <string>  using namespace std;  #include "Board.h"  // static 멤버 변수는 반드시 아래와 같이 전역 변수로 선언하여 공간을 생성해주어야 함  // 클래스의 선언은 어떤 멤버가 static 임을 선언하는 데 그치기 때문에  // static 멤버 변수들이 변수 공간을 얻기 위해서는 아래와 같이 전역 변수로 선언해야 함  int Board::size = 0; //static 멤버 변수를 전역 변수로 선언하여 변수 공간 할당  string Board::text[100]; //static 멤버 변수를 전역 변수로 선언하여 배열 공간 할당  void Board::add(string t) {  if (size > 99)  return; // 게시판이 꽉 차서 더 이상 추가 못함  text[size] = t;  size++;  }  void Board::print() {  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\* 게시판입니다. \*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  for (int i = 0; i < size; i++) {  cout << i << ": " << text[i] << endl;  }  cout << endl;  }  #include "Board.h"  int main() {  // Board myBoard; // 객체 생성은 컴파일 오류입니다.  Board::add("중간고사는 감독 없는 자율 시험입니다.");  Board::add("코딩 라운지 많이 이용해 주세요.");  Board::print();  Board::add("진소린 학생이 경진대회 입상하였습니다. 축하해주세요");  Board::print();  } |
| **[실행 결과]** |

1. 교재 366p : 문제1~4번

|  |
| --- |
| **[프로그램 소스] -1번**  //멤버 함수로 구현  #include <iostream>  #include <string>  using namespace std;  class Book {  string title;  int price;  int pages;  public:  Book(string title="", int price=0, int pages=0) {  this->title = title; this->price = price; this->pages = pages;  }  void show() {  cout << title << ' ' << price << "원 " <<pages << " 페이지" << endl;  }  string getTitle() { return title; }  Book& operator +=(int price);  Book& operator -=(int price);  };  Book& Book::operator +=(int price) {  this->price += price;  return \*this;  }  Book& Book::operator -=(int price) {  this->price -= price;  return \*this;  }  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500; // 책 a의 가격 500원 증가  b -= 500; // 책 b의 가격 500원 감소  a.show();  b.show();  }  //프렌드 함수로 구현  #include <iostream>  #include <string>  using namespace std;  class Book {  string title;  int price;  int pages;  public:  Book(string title="", int price=0, int pages=0) {  this->title = title; this->price = price; this->pages = pages;  }  void show() {  cout << title << ' ' << price << "원 " <<pages << " 페이지" << endl;  }  string getTitle() { return title; }  friend Book& operator +=(Book& book, int price);  friend Book& operator -=(Book& book, int price);  };  Book& operator +=(Book& book, int price) {  book.price += price;  return book;  }  Book& operator -=(Book& book, int price) {  book.price -= price;  return book;  }  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500; // 책 a의 가격 500원 증가  b -= 500; // 책 b의 가격 500원 감소  a.show();  b.show();  } |
| **[프로그램 소스] -2번**  **//**#include <iostream>  #include <string>  using namespace std;  class Book {  string title;  int price;  int pages;  public:  Book(string title="", int price=0, int pages=0) {  this->title = title; this->price = price; this->pages = pages;  }  void show() {  cout << title << ' ' << price << "원 " <<pages << " 페이지" << endl;  }  string getTitle() { return title; }  bool operator ==(int price);  bool operator ==(string title);  bool operator ==(Book book);  };  bool Book::operator ==(int price) {  if(this->price == price) return true;  else return false;  }  bool Book::operator ==(string title) {  if(this->title == title) return true;  else return false;  }  bool Book::operator ==(Book book) {  if(this->title == book.title &&  this->price == book.price &&  this->pages == book.pages) return true;  else return false;  }  int main() {  Book a("명품 C++", 30000, 500), b("고품 C++", 30000, 500);  if(a == 30000) cout << "정가 30000원" << endl; // price 비교  if(a == "명품 C++") cout << "명품 C++ 입니다." << endl; // 책 title 비교  if(a == b) cout << "두 책이 같은 책입니다." << endl; // title, price, pages 모두 비교  } |
| **[프로그램 소스] -3번**  #include <iostream>  #include <string>  using namespace std;  class Book {  string title;  int price;  int pages;  public:  Book(string title="", int price=0, int pages=0) {  this->title = title; this->price = price; this->pages = pages;  }  void show() {  cout << title << ' ' << price << "원 " <<pages << " 페이지" << endl;  }  string getTitle() { return title; }  bool operator !();  };  bool Book::operator !() {  if(price == 0) return true;  else return false;  }  int main() {  Book book("벼룩시장", 0, 50); // 가격은 0  if(!book) cout << "공짜다" << endl;  } |
| **[프로그램 소스] -4번**  #include <iostream>  #include <string>  using namespace std;  class Book {  string title;  int price;  int pages;  public:  Book(string title="", int price=0, int pages=0) {  this->title = title; this->price = price; this->pages = pages;  }  void show() {  cout << title << ' ' << price << "원 " <<pages << " 페이지" << endl;  }  string getTitle() { return title; }  friend bool operator <(string title , Book book);  };  bool operator < (string title , Book book) {  if(title < book.title) return true;  else return false;  }  int main() {  Book a("청춘", 20000, 300);  string b;  cout << "책 이름을 입력하세요>>";  getline(cin, b);  if(b < a)  cout << a.getTitle() << "이 " << b << "보다 뒤에 있구나!" << endl;  } |
| **[실행 결과]** |

1. 교재 369p : 문제7번

|  |
| --- |
| **[프로그램 소스]**  **//멤버 함수로 구현**  #include <iostream>  using namespace std;  class Matrix {  int ar[4];  public:  Matrix() { for(int i=0; i<4; i++) ar[i] = 0; }  Matrix(int a1, int a2, int b1, int b2) {  ar[0] = a1;  ar[1] = a2;  ar[2] = b1;  ar[3] = b2;  }  void show() {  cout << "Matrix = { ";  for(int i=0; i<4; i++) cout << ar[i] << ' ';  cout << "}" << endl;  }  Matrix& operator >> (int []);  Matrix& operator << (int []);  };  Matrix& Matrix::operator >> (int m[]) {  for(int i=0; i<4; i++)  m[i] = ar[i];  return \*this;  }  Matrix& Matrix::operator << (int m[]) {  for(int i=0; i<4; i++)  ar[i] = m[i];  return \*this;  }  int main() {  Matrix a(4,3,2,1), b;  int x[4], y[4] = {1,2,3,4}; // 2차원 행렬의 4 개의 원소 값  a >> x; // a의 각 원소를 배열 x에 복사. x[]는 {4,3,2,1}  b << y; // 배열 y의 원소 값을 b의 각 원소에 설정  for(int i=0; i<4; i++) cout << x[i] << ' '; // x[] 출력  cout << endl;  b.show();  }  **//프렌드 함수로 구현**  #include <iostream>  using namespace std;  class Matrix {  int ar[4];  public:  Matrix() { for(int i=0; i<4; i++) ar[i] = 0; }  Matrix(int a1, int a2, int b1, int b2) {  ar[0] = a1;  ar[1] = a2;  ar[2] = b1;  ar[3] = b2;  }  void show() {  cout << "Matrix = { ";  for(int i=0; i<4; i++) cout << ar[i] << ' ';  cout << "}" << endl;  }  friend Matrix& operator >> (Matrix&, int []);  friend Matrix& operator << (Matrix&, int []);  };  Matrix& operator >> (Matrix& matrix, int m[]) {  for(int i=0; i<4; i++)  m[i] = matrix.ar[i];  return matrix;  }  Matrix& operator << (Matrix& matrix, int m[]) {  for(int i=0; i<4; i++)  matrix.ar[i] = m[i];  return matrix;  } |
| **[실행 결과]** |