|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_07 | 학번 : | 20145128 | 이름 : | 박인근 |

* **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**    ------------------------- 연산자 멤버 함수를 주석처리 후 결과--------------------------------------    연산자 멤버 함수 주석처리 할 경우 Power c = a + b; 에서 오류가 납니다. |

|  |
| --- |
| 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 | 1 |
| 2 | 4 |
| 3 | Sample 함수 내에 [ friend SampleManager; ] 을 넣어줍니다. |
| 4 | Sample 함수 내에 [ friend bool SampleManager::compare(Sample &a, Sample &b); ] 를 넣으면 됩니다. |
| 5 | isValid 함수에서 student 클래스의 private 멤버인 id 를 사용하기 있기 때문에 오류가 납니다.  해결 방법으로는 private 에 접근하기 위해 friend를 사용해줍니다.  public:  Student(int id) { this->id = id; }  friend bool isValid(Student s); |
| 6 | show 함수에서 Student와 Professor의 private 멤버인 id, name 을 사용하였기 때문에 오류가 납니다.  코드를 수정하면 먼저 전방 참조 문제를 해결하기 위해 코드의 맨 앞에 전방 선언문  class Student;  class Professor;  을 삽입해주고  각 class 의 public 부분에 friend를 사용하여  friend void show(Student s, Professor p); 를 삽입해줍니다. |
| 7 | food.price 에서 오류가 나며, 이유는 Food 클래스의 private 멤버인 price 에 접근하였기 때문입니다.  코드를 수정하면 먼저 전방 참조 문제를 해결하기 위해 코드의 맨 앞에 전방 선언문  class Food;  를 삽입해주고 food 클래스의 public 부분에  friend void Person::shopping(Food food); 를 삽입해주고 구현은 분리해줍니다. |
| 8 | 4 |
| 9 | Sample 클래스에서 isZero를 friend 함수로 구현했기 때문에 isZero는 Sample의 멤버 함수가 아닌 외부 함수이며 friend 입니다. 따라서 main 에서 bool ret = a.isZero(b); 에서 오류가 납니다.  올바르게 하려면 int x; 를 private 멤버로 변환하고, bool ret = isZero(b); 로 바꿔줘야 합니다. |
| 10 | Friend 선언이 의미가 없습니다. isZero 함수에서 Sample 클래스의 멤버 변수인 x 에 접근하는데 아무런 문제가 없기 때문입니다. Friend 선언이 필요하려면 int x; 를 private 멤버로 바꿔줘야 합니다. |
| 11 | int a = 4 << 2; 에서 << 는 시프트 연산자이고  cout << ‘a’; 에서는 출력하는 연산자로 연산자 중복입니다.  여기서 볼 수 있듯이 << 연산자는 피연산자에 따라 서로 다른 의미로 실행되며 다형성을 보여주고 있습니다. |
| 12 | 부모 + 자식 = 가족 |
| 13 | 4 |
| 14 | 4 |
| 15 | 3 |
| 16 | 3 |
| 17 | 포인터를 가진 멤버가 없으므로 치환 연산(=) 에 문제가 없습니다. 따라서 치환연산자를 작성할 필요도 없습니다. |

* 프로그램 과제

|  |
| --- |
| 1. 교재 367p : 문제1~4번 |
| **[프로그램 소스 1]**  (1)  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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; }  void operator +=(int a) { price += a; }  void operator -=(int a) { price -= a; }  };  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500; //책 a 의 가격 500원 증가  b -= 500; //책 b 의 가격 500원 감소  a.show();  b.show();  system("pause");    }  (2)  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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 &a, int b);  friend Book operator -=(Book &a, int b);  };  Book operator +=(Book &a, int b) {  a.price += b;  return a;  }  Book operator -=(Book &a, int b) {  a.price -= b;  return a;  }  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500; //책 a 의 가격 500원 증가  b -= 500; //책 b 의 가격 500원 감소  a.show();  b.show();  system("pause");    } |
| **[프로그램 소스 2]**  (1)  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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 a) {  if (price == a) return true;  else return false;  }  bool operator ==(string a) {  if (title == a) return true;  else return false;  }  bool operator ==(Book a) {  if (title == a.title && price == a.price && pages == a.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  system("pause");    }  (2)  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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 ==(Book a, int b);  friend bool operator ==(Book a, string b);  friend bool operator ==(Book a, Book b);  };  bool operator ==(Book a, int b) {  if (a.price == b) return true;  else return false;  }  bool operator ==(Book a, string b) {  if (a.title == b) return true;  else return false;  }  bool operator ==(Book a, Book b) {  if (a.title == b.title && a.price == b.price && a.pages == b.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  system("pause");    } |
| **[프로그램 소스 3]**  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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 !() {  if (price == 0) return true;  else return false;  }  };  int main() {  Book book("벼룩시장", 0, 50); //가격은 0  if (!book) cout << "공짜다" << endl;  system("pause");    } |
| **[프로그램 소스 4]**  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Book {  string title;  int price, 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 a, Book b);  };  bool operator <(string a, Book b) {  if (a < b.title) return true;  else return false;  }  int main() {  Book a("청춘", 20000, 300);  string b;  cout << "책 이름을 입력하세요>>";  getline(cin, b); // 키보드로부터 문자열로 책 이름을 입력받음  if (b < a)  cout << a.getTitle() << "이 " << b << "보다 뒤에 있구나!" << endl;  system("pause");    } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 369p : 문제7번 |
| **[프로그램 소스]**  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Matrix {  int q, w, e, r;  public:  Matrix(int a=0, int b=0, int c=0, int d=0) {  this->q = a, this->w = b, this->e = c, this->r = d;  }  void show() {  cout << "Matrix = { " << q << " " << this->w << " " << this->e << " " << this->r << " }" << endl;  }  void operator >>(int b[]) {  b[0] = q;  b[1] = w;  b[2] = e;  b[3] = r;  }  void operator <<(int b[]) {  q = b[0];  w = b[1];  e = b[2];  r = b[3];  }  };  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 원소의 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();  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제10번 |
| **[프로그램 소스]**  #include <iostream>  #include <cstring>  #include <string>  using namespace std;  class Statistics {  int \*p; // int 배열 동적 할당  int cnt;  public:  Statistics() {  p = new int[7];  cnt = 0;  }  Statistics& operator <<(int a) {  p[cnt] = a;  cnt++;  return \*this;  }  void operator >>(int& a) {  int sum=0;  for (int i = 0; i < cnt; i++) {  sum += p[i];  }  a = sum / cnt;  }  bool operator !() {  if (cnt == 0) return true;  else return false;  }  void operator ~() {  for (int i = 0; i < cnt; i++) {  cout << p[i] << ' ';  }  cout << endl;  }  };  int main() {  Statistics stat;  if (!stat) cout << "현재 통계 데이터가 없습니다." << endl;  int x[5];  cout << "5 개의 정수를 입력하라>>";  for (int i = 0; i < 5; i++) cin >> x[i]; // x[i]에 정수 입력  for (int i = 0; i < 5; i++) stat << x[i]; // x[i] 값을 통계 객체에 삽입한다.  stat << 100 << 200; // 100, 200을 통계 객체에 삽입한다.  ~stat; // 통계 데이타를 모두 출력한다.  int avg;  stat >> avg; // 통계 객체로부터 평균을 받는다.  cout << "avg=" << avg << endl;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제12번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class SortedArray {  int size; // 현재 배열의 크기  int \*p; // 정수 배열의 포인터  void sort() {  for (int i = 0; i < this->size - 1; i++) {  int temp = i;  for (int j = i + 1; j < this->size; j++) {  if (p[temp] >= p[j]) temp = j;  }  int tmp = p[i];  p[i] = p[temp];  p[temp] = tmp;  } // 선택 정렬  } //정수 배열을 오름차순으로 정렬  public:  SortedArray() {  p = NULL;  size = 0;  }// p는 NULL로 size는 0으로 초기화    SortedArray(SortedArray& src) {  this->size = src.size;  this->p = new int[src.size];  for (int i = 0; i < src.size; i++) {  this->p[i] = src.p[i];  }  } //복사 생성자    SortedArray(int p[], int size) {  this->size = size;  this->p = new int[size]; // 크기 전달 받은 만큼 할당  for (int i = 0; i < size; i++) {  this->p[i] = p[i]; // 정수 배열 전달 받는것  }  } // 생성자, 정수 배열과 크기를 전달받음    ~SortedArray() {  if(p) // p가 할당되어있는지 확인  delete[] p; // 배열 할당 해제  } //소멸자    SortedArray operator + (SortedArray& op2) {  SortedArray temp; // 배열을 추가하기 위한 임시 객체 배열  temp.size = this->size + op2.size; // 기존 크기와 추가된 객체 배열의 크기를 더함  temp.p = new int[temp.size]; // 기존의 크기와 추가 배열의 크기를 더한 만큼 할당  // 배열을 추가하게 되면 추가한 만큼의 공간이 필요하기 때문  for (int i = 0; i < this->size; i++) {  temp.p[i] = this->p[i];  } // 기존 배열 삽입  for (int i = 0; i < op2.size; i++) {  temp.p[i + this->size] = op2.p[i];  } // 기존 배열 뒤에 op2 배열 추가  return temp;  } // 현재 배열에 op2 배열 추가    SortedArray& operator = (const SortedArray& op2) {  delete[] p; // 현재 배열에 op2 배열을 복사하기 위해 할당 해제  this->p = new int[op2.size]; // op2 크기만큼 배열 할당  this->size = op2.size;  for (int i = 0; i < op2.size; i++) {  this->p[i] = op2.p[i]; // 배열 복사  }  return \*this;  } // 현재 배열에 op2 배열 복사    void show() {  this->sort(); // 배열을 선택 정렬  cout << "배열 출력 : ";  for (int i = 0; i < this->size; i++) {  cout << p[i] << ' ';  }  cout << endl;  } //배열의 원소 출력  };  int main() {  int n[] = { 2, 20, 6 };  int m[] = { 10, 7, 8, 30};  SortedArray a(n, 3), b(m, 4), c;  c = a + b; // +, = 연산자 작성 필요  // + 연산자가 SortedArray 객체를 리턴하므로 복사 생성자 필요  a.show();  b.show();  c.show();  system("pause");  } |
| **[실행결과]** |