|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_07 | 학번 : | 20135151 | 이름 : | 이갑성 |

* **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; //기본형 private으로 설정되어있기에 외부함수 equal()으로 접근할 수 없다.  --> 프렌드로 설정해서 접근  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 | 1번 |
| 2 | 4번 |
| 3 | class SampleManger;  class Sample {  friend SampleManager;  }; |
| 4 | class Sample {  friend bool SampleManager::compare(Sample& a, Sample& b);  }; |
| 5 | bool isValid(Student s) 함수는 외부함수이다. Student클래스의 멤버 id가 private로 설정되있어있기에 외부함수가 접근할 수가 없다. 따라서 friend함수로 Student클래스에 선언해주어 접근할 수 있게 해준다.  #include <iostream>  using namespace std;  bool isValid(Student s);  class Student;  bool isValid(Student s);  class Student {  int id;  public:  Student(int id) {  this->id = id;  }  friend bool isValid(Student s);  };  bool isValid(Student s) {  if (s.id > 0)  return true;  else  return false;  } |
| 6 | 외부함수 show()가 Student, Professor 클래스의 멤버변수에 접근할 수 없기에, 각 클래스에 friend함수를 이용해서 접근 가능하게 해준다.  #include <iostream>  #include <string>  using namespace std;  class Student;  class Professor;  void show(Student s, Professor p);  class Student {  int id;  public:  Student(int id) {  this->id = id;  }  friend void show(Student s, Professor p);  };  class Professor {  string name;  public:  Professor(string name) {  this->name = name;  }  friend void show(Student s, Professor p);  };  void show(Student s, Professor p) {  cout << s.id << p.name;  } |
| 7 | Person클래스에서 Food클래스에 있는 멤버변수 price에 접근하고 싶지만 private라서 접근할 수가 없다. 따라서 Food클래스에 Person클래스를 friend함수 구현해줘서 Person클래스가 접근할 수 있게 해준다.  #include <iostream>  #include <string>  using namespace std;  class Person;  class Food {  int price;  string name;  public:  Food(string name, int price);  void buy();  friend class Person;  };  class Person {  int id;  public:  void shopping(Food food) {  if (food.price < 1000)  food.buy();  }  int get() { return id; }  }; |
| 8 | 4번 |
| 9 | #include <iostream>  #include <string>  using namespace std;  class Sample;  bool isZero(Sample& a);  class Sample {  int x;  public:  Sample(int x) {  this->x = x;  }  friend bool isZero(Sample& a);  };  bool isZero(Sample& a) {  if (a.x == 0)  return true;  else  return false;  }  int main() {  Sample a(5), b(6);  bool ret = isZero(b);    } |
| 10 | 필요없다. 왜냐하면 int x 가 private이 아니기 때문이다. |
| 11 | 연산자 중복 |
| 12 | 노력 + 성실 = 좋은결과 |
| 13 | 4번 |
| 14 | 4번 |
| 15 | 3번 |
| 16 | 3번 |
| 17 | 연산자 = 을 사용하려면 피연산자가 숫자이어야한다. 따라서 현재 a와 b는 객체 임으로 원칙상 = 연산자를 사용할 수 없다. 그러나 연산자 중복을 통해 객체가 가능하게 할 수 있기에 작성해주어야 한다. |

* 프로그램 과제

|  |
| --- |
| 1. 교재 367p : 문제1~4번 |
| **[프로그램 소스 1]**  (1)  #include <iostream>  #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;  }  Book& operator+=(int n);  Book& operator-=(int n) {  price = price - n;    return \*this;  }  };  Book& Book::operator+=(int n)  {  price = price + n;    return \*this;  }  int main() {  Book a("청춘", 2000, 300), b("미래", 30000, 500);  a += 500;  b -= 500;  a.show();  b.show();  }  (2)  #include <iostream>  #include <string>  using namespace std;  class Book;  Book& operator+=(Book& op, int n);  Book operator-=(Book& op, int n);  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& op, int n);  friend Book operator-=(Book& op, int n);  };  Book& operator+=(Book& op, int n)  {  op.price += n;  return op;  }  Book operator-=(Book& op, int n)  {  Book tmp;  tmp.price = op.price - n;  return tmp;  }  int main() {  Book a("청춘", 2000, 300), b("미래", 30000, 500);  a += 500; // a = a + 500  b -= 500;  a.show();  b.show();  } |
| **[프로그램 소스 2]**  **(1)**  #include <iostream>  #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 cmpPrice)  {  if (price == cmpPrice)  return true;  else  return false;  }  bool operator==(string cmpTitle)  {  if (title == cmpTitle)  return true;  else  return false;  }  bool operator==(Book cmpBook)  {  if (price == cmpBook.price) {  if (title == cmpBook.title) {  if (pages == cmpBook.pages) {  return true;  }  }  }  else  return false;  }  };  int main() {  Book a("명품 C++", 30000, 500), b("고품 C++", 30000, 500);  if (a == 30000) //가격 price비교  cout << "정가 30000원" << endl;  if (a == "명품 C++") //책 title비교  cout << "명품 C++입니다." << endl;  if (a == b) //title, price, pages 모두 비교  cout << "두 책이 같은 책입니다." << endl;    }  (2)  #include <iostream>  #include <string>  using namespace std;  class Book;  bool operator==(Book op, int cmpPrice);  bool operator==(Book op, string cmpTitle);  bool operator==(Book op, Book cmpBook);  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 op,int cmpPrice);  friend bool operator==(Book op, string cmpTItle);  friend bool operator==(Book op, Book cmpBook);    };  bool operator==(Book op, int cmpPrice)  {  if (op.price == cmpPrice)  return true;  else  return false;  }  bool operator==(Book op, string cmpTitle)  {  if (op.title == cmpTitle)  return true;  else  return false;  }  bool operator==(Book op, Book cmpBook)  {  if (op.price == cmpBook.price) {  if (op.title == cmpBook.title) {  if (op.pages == cmpBook.pages) {  return true;  }  }  }  else  return false;  }  int main() {  Book a("명품 C++", 30000, 500), b("고품 C++", 30000, 500);  if (a == 30000) //가격 price비교  cout << "정가 30000원" << endl;  if (a == "명품 C++") //책 title비교  cout << "명품 C++입니다." << endl;  if (a == b) //title, price, pages 모두 비교  cout << "두 책이 같은 책입니다." << endl;    } |
| **[프로그램 소스 3]**  #include <iostream>  #include <string>  using namespace std;  class Book;  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!();  };  bool Book::operator!()  {  if (price == 0)  return true;  else  return false;  }  int main() {  Book book("벼룩시장", 0, 50);  if (!book)  cout << "공짜다" << endl;    } |
| **[프로그램 소스 4]**  #include <iostream>  #include <string>  using namespace std;  class Book;  bool operator<(string str, Book op2);  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 str, Book op2);    };  bool operator<(string str, Book op2)  {  if (str < op2.getTitle())  return true;  else  return false;  }    int main() {  Book a("청춘", 20000, 300);  string b;  cout << "책이름을 입력하세요 >> ";  getline(cin, b);  if (b < a)  cout << a.getTitle() << "이 " << b << "보다 뒤에 있구나! " << endl;  } |
| **[실행결과]**  1.    2.    3.    4. |

|  |
| --- |
| 1. 교재 369p : 문제7번 |
| **[프로그램 소스]**  (1)  #include <iostream>  using namespace std;  class Matrix {  int arr[4];  public:  Matrix()  {  for (int i = 0; i < 4; i++)  {  arr[i] = 0;  }  }  Matrix(int a, int b, int c, int d)  {  arr[0] = a;  arr[1] = b;  arr[2] = c;  arr[3] = d;  }  void operator>>(int \*arr);  void operator<<(int \*arr);  void show();  };  void Matrix::operator>>(int \*arr)  {  for (int i = 0; i < 4; i++)  {  arr[i] = this->arr[i];  }  }  void Matrix::operator<<(int \*arr)  {  for (int i = 0; i < 4; i++)  {  this->arr[i] = \*(arr + i);  }  }  void Matrix::show()  {  cout << "Maxtrix = { ";  for (int i = 0; i < 4; i++)  {  cout << arr[i] << " ";  }  cout << " }" << endl;  }  int main() {  Matrix a(4, 3, 2, 1), b;  int x[4], y[4] = { 1, 2, 3, 4 };  a >> x;  b << y;  for (int i = 0; i < 4; i++) {  cout << x[i] << ' ';  }  cout << endl;  b.show();  }  (2)  #include <iostream>  using namespace std;  class Matrix;  void operator>>(Matrix obj, int \*x);  void operator<<(Matrix& op, int \*arr);  class Matrix {  int arr[4];  public:  Matrix() {  for (int i = 0; i < 4; i++)  arr[i] = 0;  }  Matrix(int a, int b, int c, int d)  {  arr[0] = a;  arr[1] = b;  arr[2] = c;  arr[3] = d;  }  friend void operator>>(Matrix obj, int \*x);  friend void operator<<(Matrix& op, int \*arr);  void show();  };  void operator>>(Matrix obj, int \*x)  {  for (int i = 0; i < 4; i++)  {  x[i] = obj.arr[i];  }  }  void operator<<(Matrix& op, int \*arr) {  for (int i = 0; i < 4; i++)  {  op.arr[i] = arr[i];  }  }  void Matrix::show() {  cout << "Matrix = { ";  for (int i = 0; i < 4; i++) {  cout << this->arr[i] << " ";  }  cout << "}" << endl;  }  int main() {  Matrix a(4, 3, 2, 1);  Matrix b;  int x[4], y[4] = { 1, 2, 3, 4 };  a >> x;  b << y;  for (int i = 0; i < 4; i++)  {  cout << x[i] << ' ';  }  cout << endl;  b.show();  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제10번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class Statistics {  int \*p;  int index;  public:  Statistics() {  p = new int[7];  index = 0;  for (int i = 0; i < 7; i++)  p[i] = -1;  }  bool operator!() {  if (p[0] == -1)  return true;  else  return false;  }  Statistics& operator<<(int n);  void operator~();  void operator>>(int& avg);    ~Statistics() {  delete[] p;  }  };  Statistics& Statistics::operator<<(int n)  {  this->p[index++] = n;  return \*this;  }  void Statistics::operator~()  {  for (int i = 0; i < index; i++)  {  cout << p[i] << " ";  }  cout << endl;  }  void Statistics::operator>>(int& avg)  {  int total = 0, result;  for (int i = 0; i < index; i++)  {  total += p[i];  }  result = total / index;  avg = result;  }  int main() {  Statistics stat;  if (!stat) //!연산자 중복사용  cout << "현재 통계 데이타가 없습니다." << endl;  int x[5];  cout << "5 개의 정수를 입력하라 >>> ";  for (int i = 0; i < 5; i++)  cin >> x[i];  for (int i = 0; i < 5; i++)  stat << x[i]; //<< 연산자 중복사용: x[i]에 있는 값을 통계 객체에 삽입한다.  stat << 100 << 200;;  ~stat; //~ 연산자 중복사용: 통계 데이터를 모두 출력한다.  int avg;  stat >> avg; //>> 연산자 중복사용: 통계 객체로부터 평균을 받는다.  cout << "avg = " << avg << endl; //평균을 출력한다.  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제12번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class SortedArray {  int size; // 현재배열의 크기  int \*p; //정수 배열에 대한 포인터  void sort();  public:  SortedArray()  {  p = NULL;  size = 0;  }  SortedArray(const SortedArray& src)  {  this->size = src.size;  this->p = new int[size];  for (int i = 0; i < size; i++)  p[i] = src.p[i];  }  SortedArray(int inputP[], int size)  {  this->size = size;  p = new int[size];  for (int i = 0; i < size; i++)  {  p[i] = inputP[i];  }  }  ~SortedArray()  {  delete[] p;  }  SortedArray operator+(SortedArray& op2); // 현재 배열에 op2 배열 추가  SortedArray& operator=(const SortedArray& op2); //현재 배열에 op2 배열 복사  void show(); //배열 원소 출력  };  SortedArray SortedArray::operator+(SortedArray& op2) // 현재 배열에 op2 배열 추가  {  SortedArray tmp;  tmp.size = this->size + op2.size;  tmp.p = new int[tmp.size];  for (int i = 0; i < this->size; i++)  {  tmp.p[i] = p[i];  }  for (int i = 0; i < op2.size; i++)  {  tmp.p[this->size + i] = op2.p[i];  }  return tmp;    }  SortedArray& SortedArray::operator=(const SortedArray& op2) //현재 배열에 op2 배열 복사  {  if (!p) //현재C의 값에 p가 가르키는 동적메모리가 있으면 지워준다.  {  delete[] p;  }  p = new int[op2.size];  this->size = op2.size;  for (int i = 0; i < op2.size; i++)  {  p[i] = op2.p[i];  }  return \*this;  }  void SortedArray::sort()  {  for (int i = 0; i < size - 1; i++)  {  for (int j = i + 1; j < size; j++)  {  if (p[i] > p[j]) {  int tmp;  tmp = p[i];  p[i] = p[j];  p[j] = tmp;  }  }  }  }  void SortedArray::show()  {  sort(); //배열 정렬해주기  cout << "배열 출력: ";  for (int i = 0; i < size; i++)  {  cout << p[i] << " ";  }  cout << endl;  }  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  a.show();  b.show();  c.show();  } |
| **[실행결과]** |