|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_07 | 학번 : | 20145165 | 이름 : | 정균모 |

* **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 맴버 함수를 주석처리하면 operator 가 없기 때문에 a + b를 할 수 없어서 오류가 생긴다.** |

|  |
| --- |
| 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 | Friend SampleManager; |
| 4 | Friend bool SampleManager::compare(Sample &a, Sample &b); |
| 5 | isValid 함수에서 매개변수 s에서 id에 접근 권한이 없다. 해결 방안은 Student에서 friend를 선언하는 것  friend bool isValid(Student s); |
| 6 | show에서 각 클래스에 private 맴버 변수를 접근할 권한이 없기 때문에 friend를 선언해주면 된다.  friend void show(Student s, Professor p); |
| 7 | Person 클래스에서 Food에 price를 접근할 권한이 없다. 우선 Food에 friend를 선언 후 Person를 선언부와 구현부로 나눠서 만든다.  class Person {  int id;  public:  void shopping(Food food);  };  class Food {  int price;  string name;  public:  Food(string name, int price);  void buy();  friend void Person::shopping(Food food);  };  void Person::shopping(Food food) {  if (food.price < 1000) {  food.buy();  }  } |
| 8 | 4 |
| 9 | 옳지 않다. friend bool isZero(Sample &a) 이렇게 friend를 선언하면 맴버가 아닌 전역 함수가 된다. 그래서 main에서 선언할 때 a.isZero(b); 가 아닌 전역함수를 사용하는 것 처럼 bool ret=isZero(b); 로 선언 되어야 한다. |
| 10 | 프렌드를 선언하지 않아도 public 이기 때문에 a.x가 접근이 가능하다. |
| 11 | 서로 다른 객체가 동일한 메시지에 대하여 서로 다른 방법으로 응답하는 것을 보면 다형성을 보여주고 있다. |
| 12 | 공대생 + 공대생 = 프로젝트 완성 |
| 13 | 4 |
| 14 | 4 |
| 15 | 2 |
| 16 | 3 |
| 17 | 필요 없다. A= b 를 했을 때 a의 radius는 30의로 치환이 된다. |

* 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; }  void operator +=(int n) { price += n; }  void operator -=(int n) { price -= n; }  };  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500;  b -= 500;  a.show();  b.show();  }  (2)  #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; }  friend Book operator +=(Book &b, int n);  friend Book operator -=(Book &b, int n);  };  Book operator +=(Book &b, int n) {  b.price += n;  return b;  }  Book operator -=(Book &b, int n) {  b.price -= n;  return b;  }  int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  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 n) {  if( price == n ) return true;  else return false;  }  bool operator == (string n) {  if( title == n ) return true;  else return false;  }  bool operator == (Book n) {  if( title == n.title && price == n.price && pages == n.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;  if( a == "명품 C++" ) cout << "명품 C++ 입니다." << endl;  if( a == b ) cout << "두 책이 같은 책입니다." << endl;  }  (2)  #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; }  friend bool operator == (Book b, int n);  friend bool operator == (Book b, string n);  friend bool operator == (Book b1, Book b2);  };  bool operator == (Book b, int n) {  if( b.price == n ) return true;  else return false;  }  bool operator == (Book b, string n) {  if( b.title == n ) return true;  else return false;  }  bool operator == (Book b1, Book b2) {  if( b1.title == b2.title && b1.price == b2.price && b1.pages == b2.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;  if( a == "명품 C++" ) cout << "명품 C++ 입니다." << endl;  if( a == b ) cout << "두 책이 같은 책입니다." << endl;  } |
| **[프로그램 소스 3]**  #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 !() {  if( price == 0 ) return true;  else 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 {  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 s, Book b);  };  bool operator < (string s, Book b) {  if( s < b.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;  } |
| **[실행결과]** |

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

|  |
| --- |
| 1. 교재 370p 문제10번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class Statistics {  int \*p;  int count;  public:  Statistics() { p = new int[10]; count = 0; }  bool operator ! () {  if (count == 0) return true;  else return false;  }  Statistics& operator << (int n) {  p[count] = n;  count++;  return \*this;  };  void operator ~ () {  for (int i = 0; i < count; i++)  cout << p[i] << ' ';  cout << endl;  }  void operator >>(int& avg) {  int sum = 0;  for (int i = 0; i < count; i++)  sum += p[i];  avg = sum / count;  }  };  int main() {  Statistics stat;  if (!stat) 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];  stat << 100 << 200;  ~stat;  int avg;  stat >> avg;  cout << "avg=" << avg << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 370p 문제12번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class SortedArray {  int size;  int \*p;  void sort();  public:  SortedArray();  SortedArray(SortedArray& src);  SortedArray(int p[], int size);  ~SortedArray();  SortedArray operator + (SortedArray& op2);  SortedArray& operator = (const SortedArray& op2);  void show();  };  void SortedArray::sort() {  for (int i = 0; i < size - 1; ++i) {  for (int j = 1; j < size - i; ++j) {  if (p[j - 1] > p[j]) {  int temp = p[j - 1];  p[j - 1] = p[j];  p[j] = temp;  }  }  }  }  SortedArray::SortedArray() {  p = NULL; size = 0;  }  SortedArray::~SortedArray() {  if (p)  delete[] p;  }  SortedArray::SortedArray(int p[], int size) {  this->size = size;  this->p = new int[size];  for (int i = 0; i < size; ++i) {  this->p[i] = p[i];  }  }  SortedArray::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];  }  SortedArray SortedArray::operator+(SortedArray& op2) {  SortedArray tmp;  tmp.p = new int[this->size + op2.size];  tmp.size = this->size + op2.size;  for (int i = 0; i < this->size; ++i)  tmp.p[i] = this->p[i];  for (int i = 0; i < op2.size; ++i)  tmp.p[i + this->size] = op2.p[i];  return tmp;  }  SortedArray& SortedArray::operator=(const SortedArray& op2) {  delete[] this->p;  this->p = new int[op2.size];  this->size = op2.size;  for (int i = 0; i < op2.size; ++i)  this->p[i] = op2.p[i];  return \*this;  }  void SortedArray::show() {  this->sort();  cout << "배열 출력 : ";  for (int i = 0; i < this->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;  a.show();  b.show();  c.show();  system("pause");  } |
| **[실행결과]** |