**객체지향프로그래밍 LAB #11**

**<기초문제>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. 아래의 프로그램을 작성하시오. (/\*구현\*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

|  |
| --- |
| #include <iostream>  using namespace std;  class Point {  private:  int x;  int y;  static int numCreatedObjects;  public:  Point() : x(0), y(0) {  numCreatedObjects++;  }  // int \_x 와 int \_y를 입력으로 받는 생성자  /\*구현\*/  ~Point() {  cout << "Destructed..." << endl;  }  void setXY(int \_x, int \_y) {  //this-> 사용한 초기화  /\*구현\*/  }  int getX() const { return x; }  int getY() const { return y; }  // \*this + pt2 ->  Point operator+(Point& pt2) {  /\*구현\*/  }  //operator overloading(연산자 오버로딩)  Point& operator=(Point& pt) {  /\*구현\*/  }  static int getNumCreatedObject() { return numCreatedObjects; }  friend void print(const Point& pt);  friend ostream& operator<<(ostream& cout, Point& pt);  friend class SpyPoint;  };  //static 맴버 변수 초기화 (numCreatedObjects)  /\*구현\*/  //객체 call by reference시: const로 함수 입력시 const method만 함수에서 사용가능  // const: 객체 내부의 member data가 상수(변하지 않는다)  void print(/\*구현\*/) {  cout << pt.x << ", " << pt.y << endl;  }  //Point operator+(Point& pt1, Point& pt2){  // Point result(pt1.getX() + pt2.get(X), pt1.getY() + pt2.getY());  // return result;  //}  ostream& operator<<(ostream& cout, Point& pt) {  /\*구현\*/  }  class SpyPoint {  public:  //다음과 같이 출력 되도록 hack\_all\_info함수 구현  //Hacked by SpyPoint  //x: 40  //y: 60  //numCreatedObj.: 10  /\*구현\*/  };  int main() {  Point pt1(1, 2);  cout << "pt1 : ";  print(pt1);  cout << endl;  // 포인터  Point\* pPt1 = &pt1;  // pPt1의 값을 통해 getX, getY를 호출하여 출력  cout << "pt1 : ";  /\*구현\*/  // pPt1를 통해 호출 getX, getY를 호출하여 출력  cout << "pt1 : ";  /\*구현\*/  cout << endl;  //동적으로 Point\* pPt2할당하여 10,20넣은 뒤 ->사용하여 출력(pt1 출력 참고)  /\*구현\*/  cout << "pt2 : ";  /\*구현\*/  cout << endl;    //pPt1, pPt2의 메모리 해제  /\*구현\*/  cout << "pt1 NumCreatedObject : ";  cout << /\*구현\*/ << endl;  // 연산자 오버로딩  //pt4 = pt2, pt3값 더하기  /\*구현\*/  cout << "pt2 : ";  cout << pt2 << endl;  cout << "pt3 : ";  cout << pt3 << endl;  cout << "pt4 : ";  cout << pt4 << endl;  cout << "pt1 NumCreatedObject : ";  cout << /\*구현\*/ << endl << endl;  // object array  Point\* ptAry = /\*구현\*/;  cout << "pt2 NumCreatedObject : ";  cout << /\*구현\*/ << endl;  cout << endl;  // ptAry 메모리 해제  /\*구현\*/  cout << endl;  // friend class  SpyPoint spy;  cout << "pt1 info" << endl;  /\*구현\*/  cout << "pt4 info" << endl;  /\*구현\*/  return 0;  } |
|  |
| #include <iostream>  using namespace std;  class Point {  private:  int x;  int y;  static int numCreatedObjects;  public:  Point() : x(0), y(0) {  numCreatedObjects++;  }  // int \_x 와 int \_y를 입력으로 받는 생성자  Point(int \_x, int \_y) : x(\_x), y(\_y) {  numCreatedObjects++;  }  ~Point() {  cout << "Destructed..." << endl;  }  void setXY(int \_x, int \_y) {  //this-> 사용한 초기화  this->x = \_x;  this->y = \_y;  }  int getX() const { return x; }  int getY() const { return y; }  // \*this + pt2 ->  Point operator+(Point& pt2) {  Point result;  result.x = this->x + pt2.x;  result.y = this->y + pt2.y;  return result;  }  //operator overloading(연산자 오버로딩)  Point& operator=(Point& pt) {  this->x = pt.x;  this->y = pt.y;  return \*this;  }  static int getNumCreatedObject() { return numCreatedObjects; }  friend void print(const Point& pt);  friend ostream& operator<<(ostream& cout, Point& pt);  friend class SpyPoint;  };  //static 맴버 변수 초기화 (numCreatedObjects)  int Point::numCreatedObjects = 0;  //객체 call by reference시: const로 함수 입력시 const method만 함수에서 사용가능  // const: 객체 내부의 member data가 상수(변하지 않는다)  void print(const Point& pt) {  cout << pt.x << ", " << pt.y << endl;  }  //Point operator+(Point& pt1, Point& pt2){  // Point result(pt1.getX() + pt2.get(X), pt1.getY() + pt2.getY());  // return result;  //}  ostream& operator<<(ostream& cout, Point& pt) {  cout << pt.x << ", " << pt.y ;  return cout;  }  class SpyPoint {  public:  //다음과 같이 출력 되도록 hack\_all\_info함수 구현  //Hacked by SpyPoint  //x: 40  //y: 60  //numCreatedObj.: 10  void hack\_all\_info(Point& pt) {  cout << "Hacked by SpyPoint" << endl;  cout << "x: " << pt.x << endl;  cout << "y: " << pt.y << endl;  cout << "numCreatedObj.: " << pt.numCreatedObjects << endl<< endl;  }  };  int main() {  Point pt1(1, 2);  cout << "pt1 : ";  print(pt1);  cout << endl;  // 포인터  Point\* pPt1 = &pt1;  // pPt1의 값을 통해 getX, getY를 호출하여 출력  cout << "pt1 : ";  cout << (\*pPt1).getX() << ", " << (\*pPt1).getY() << endl;  // pPt1를 통해 호출 getX, getY를 호출하여 출력  cout << "pt1 : ";  cout << pPt1->getX() << ", " << pPt1->getY() << endl;  cout << endl;  //동적으로 Point\* pPt2할당하여 10,20넣은 뒤 ->사용하여 출력(pt1 출력 참고)  Point\* pPt2 = new Point(10, 20);  cout << "pt2 : ";  cout << pPt2->getX() << ", " << pPt2->getY() << endl;  cout << endl;  //pPt1, pPt2의 메모리 해제  delete pPt2;  cout << "pt1 NumCreatedObject : ";  cout << pt1.getNumCreatedObject() << endl;  // 연산자 오버로딩  //pt4 = pt2, pt3값 더하기  Point pt2(10, 20), pt3(30, 40);  Point pt4 = pt2 + pt3;  cout << "pt2 : ";  cout << pt2 << endl;  cout << "pt3 : ";  cout << pt3 << endl;  cout << "pt4 : ";  cout << pt4 << endl;  cout << "pt1 NumCreatedObject : ";  cout << pt1.getNumCreatedObject() << endl << endl;  // object array  Point\* ptAry = new Point[5];  cout << "pt2 NumCreatedObject : ";  cout << pt2.getNumCreatedObject() << endl;  cout << endl;  // ptAry 메모리 해제  delete[] ptAry;  cout << endl;  // friend class  SpyPoint spy;  cout << "pt1 info" << endl;  spy.hack\_all\_info(pt1);  cout << "pt4 info" << endl;  spy.hack\_all\_info(pt4);  return 0;  } |
|  |

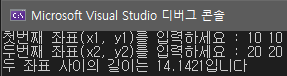
**<응용문제>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. 좌표값 두 개를 입력받고 두 좌표 사이의 거리를 출력하시오.

* 좌표값은 초기화 시, 값을 따로 주지 않으면 x = 0, y = 0으로 초기화함.
* “좌표 - 좌표” 연산자를 Operator overloading을 활용하여 선언함.
  + 를 클래스 내에서 구현.
* “좌표 \* 좌표” 연산자를 Operator overloading을 활용하여 선언함.
  + 를 클래스 내에서 구현.
* 위 두 연산자를 활용하여 두 좌표 사이의 거리를 구함.
* 제곱근 사용을 위해 #include<cmath> 선언.

|  |
| --- |
| int main() {  int x1 = 0, y1 = 0, x2 = 0, y2 = 0;  Point\* pP1, \* pP2, \* pP3;  cout << "첫번째 좌표(x1, y1)를 입력하세요 : ";  cin >> x1 >> y1;  cout << "두번째 좌표(x2, y2)를 입력하세요 : ";  cin >> x2 >> y2;  pP1 = new Point(x1, y1);  pP2 = new Point(x2, y2);  pP3 = new Point(); //x,y가 0으로 초기화  /\* 아래의 방식으로도 x, y값 설정이 가능해야 함 \*/  //pP1->setPoint(x1, y1);  //pP1->setPoint(x2, y2);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  \*pP3 = (\*pP1 - \*pP2) \* (\*pP1 - \*pP2);  /\* pP3을 활용하여 거리값을 구함 \*/  cout << "두 좌표 사이의 길이는 " << /\* 결과 값 \*/ << "입니다." << endl;  return 0;  } |

1-출력화면:



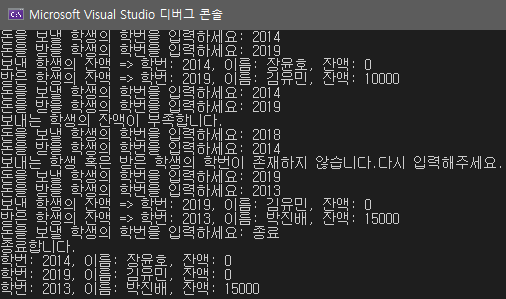
|  |
| --- |
| #include <iostream>  using namespace std;  #include <math.h>  class Point {  private:  double x;  double y;  public:  Point();  Point(int x, int y);  void setPoint(int x, int y);  int getX(void) const;  int getY(void) const;  Point operator - (const Point& point);  Point operator \* (const Point& point);  Point& operator = (const Point& point);  };  Point::Point() { x = y = 0; }  Point::Point(int x, int y) {  this->x = x;  this->y = y;  }  int Point::getX(void) const { return this->x; }  int Point::getY(void) const { return this->y; }  void Point::setPoint(int x, int y) {  this->x = x;  this->y = y;  }  Point Point::operator -(const Point& point) {  Point result(this->x - point.getX(), this->y - point.getY());  return result;  }  Point Point::operator \*(const Point& point) {  Point result(this->x \* point.getX(), this->y \* point.getY());  return result;  }  Point& Point::operator = (const Point& point) {  this->x = point.getX();  this -> y = point.getY();  return \*this;  }  int main() {  int x1 = 0, y1 = 0, x2 = 0, y2 = 0;  Point\* pP1, \* pP2, \* pP3;  cout << "첫번째 좌표(x1, y1)를 입력하세요 : ";  cin >> x1 >> y1;  cout << "두번째 좌표(x2, y2)를 입력하세요 : ";  cin >> x2 >> y2;  pP1 = new Point(x1, y1);  pP2 = new Point(x2, y2);  pP3 = new Point(); //x,y가 0으로 초기화  /\* 아래의 방식으로도 x, y값 설정이 가능해야 함 \*/  //pp1->setpoint(x1, y1);  //pp1->setpoint(x2, y2);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  \*pP3 = (\*pP1 - \*pP2) \* (\*pP1 - \*pP2);  /\* pp3을 활용하여 거리값을 구함 \*/  cout << "두 좌표 사이의 길이는 " << sqrt((pP3 -> getX()) + (pP3->getY())) << "입니다." << endl;  return 0;  } |
|  |

2. 학생들끼리 송금하는 프로그램을 작성하시오.

* 한번 송금할 때, 돈을 보내는 학생의 전 재산(balance)를 송금함.
* Account(계좌) class는 멤버변수로 string name(이름), string id(학번), int balance를 갖고 있도록 함.
* 사용자로부터 돈을 보낼 학생의 학번과 돈을 받을 학생의 학번을 입력 받음. 이때, 다음의 경우에 대해서는 다시 입력 받도록 함.
  + 돈을 보낼 학생과 돈을 받을 학생의 학번이 동일한 경우
  + 보낼 학생 혹은 받을 학생의 학번이 존재하지 않는 경우
  + 보낼 학생의 잔액이 0인 경우
* setBalance함수를 사용하지 않고, Operator overloading(+, -)를 이용하여 송금 후 보낸 학생의 잔액과 받은 학생의 잔액을 계산함.
* 송금이 완료된 후, Operator overloading(<<)을 이용하여 돈을 보낸 학생과 받은 학생의 계좌를 출력함.
* 사용자로부터 돈을 보낼 학생의 학번을 “종료”라고 입력받았을 경우 Operator overloading(<<)을 이용하여 모든 학생의 계좌를 출력하고, 프로그램을 종료함.

|  |
| --- |
| int main() {  Account acnt[3] = {  Account("장윤호", "2014", 10000),  Account("김유민", "2019", 0),  Account("박진배", "2013", 5000),  };  /\* 구현 \*/  } |

2-출력화면:



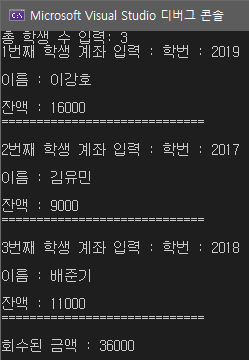
|  |
| --- |
| #include <iostream>  using namespace std;  class Account {  private:  string name;  string id;  int balance ;  public:  Account(string \_name, string \_id, int \_balance) : name(\_name), id(\_id), balance(\_balance) {}  ~Account() {}  string getName() const { return name; }  string getID() const { return id; }  int getBalance() const {  return balance;  }  Account& operator+(Account acnt) {  this->balance += acnt.getBalance();  return \*this;  }  Account& operator-(Account acnt) {  this->balance -= acnt.getBalance();  return \*this;  }  Account& operator=(Account acnt) {  this->name = acnt.getName();  this->id = acnt.getID();  this->balance = acnt.getBalance();  return \*this;  }  //friend void print(const Account& acnt);  friend void send(Account& acnt1, Account& acnt2);  friend ostream& operator<<(ostream& cout, Account& acnt);  };  void send(Account& acnt1, Account& acnt2) {  acnt2 + acnt1;  acnt1 - acnt1;  }  ostream& operator<<(ostream& cout, Account& acnt) {  cout << "학번: " << acnt.getID() << ", 이름: " << acnt.getName()<< ", 잔액: " << acnt.getBalance();  return cout;  }  int main() {  Account acnt[3] = {  Account("장윤호", "2014", 10000),  Account("김유민", "2019", 0),  Account("박진배", "2013", 5000),  };  string id1, id2;  while (1) {  std::cout << "돈을 보낼 학생의 학번을 입력하세요: ";  cin >> id1;  if (id1 == "종료") break;  std::cout << "돈을 받을 학생의 학번을 입력하세요: ";  cin >> id2;  if (id2 == "종료") break;  if (id1 == id2) {  std::cout << "보내는 학생과 받는 학생의 학번이 동일합니다" << endl;  continue;  }  int ex = 0;  for (int i = 0; i < 3; i++) {  for (int j = 0; j < 3; j++) {  if (acnt[i].getID() == id1 && acnt[j].getID() == id2) {  ex++;  if (acnt[i].getBalance() == 0) {  std::cout << "보내는 학생의 잔액이 부족합니다." << endl;  break;  }  send(acnt[i], acnt[j]);  std::cout << "보낸 학생의 잔액 => " << acnt[i] << endl;  std::cout << "받은 학생의 잔액 => " << acnt[j] << endl;  }  }  }  if (!ex) cout << "보내는 학생 혹은 받은 학생의 학번이 존재하지 않습니다. 다시 입력해주세요." << endl;  }  std::cout << "종료합니다."<<endl;  for (int i = 0; i < 3; i++) {  cout << acnt[i]<<endl;  }  } |
|  |

3. 학생 계좌 정보를 입력받고 학생 계좌 정보들을 모두 삭제하여 삭제된 계좌들의 잔액 총합을 출력하는 프로그램을 작성하시오.

* 총 학생 수를 사용자로부터 입력받음.
* 학생 계좌 정보 Account class는 학번, 이름, 잔액을 멤버 변수로 가지고 있음.
* Account 클래스 배열을 동적으로 생성함.
* 각 학생의 학번, 이름, 잔액을 입력받음. 이 때, 중복된 학번을 입력받으면 프로그램을 종료함.
* 모두 입력 받은 후, Account 클래스 배열을 delete하여 모든 학생 계좌의 잔액 총합을 출력함.

Hint) Account class 내 멤버 변수에서 static 변수를 사용, Account 클래스의 소멸자를 적절히 이용.

3-출력화면:



|  |
| --- |
| #include <iostream>  using namespace std;  class Account {  string name;  string id;  int balance ;  static int sum;  public:  Account() : name("홍길동"), id("0000"), balance(0) {}  Account(string \_name, string \_id, int \_balance) : name(\_name), id(\_id), balance(\_balance) { sum += balance; }  ~Account() {}  void set(string \_id, string \_name, int \_balance) {  this->id = \_id;  this->name = \_name;  this->balance = \_balance;  sum += balance;  }  string getName() const { return name; }  string getID() const { return id; }  int getBalance() const {  return balance;  }  static int getSum() {  return sum;  }  };  int Account::sum = 0;  int main() {  int size = 0;  cout << "총 학생 수 입력: ";  cin >> size;  Account\* acnts = new Account[size];  /\*  모두 입력 받은 후, Account 클래스 배열을 delete하여 모든 학생 계좌의 잔액 총합을 출력함.  Hint) Account class 내 멤버 변수에서 static 변수를 사용, Account 클래스의 소멸자를 적절히 이용.\*/  for (int i = 0; i < size; i++) {  string name, id;  int balance;  cout << i+1 << "번째 학생 계좌 입력 : 학번 : ";  cin >> id;    for (int j = 0; j < size; j++) {  if (id == acnts[j].getID()) exit(NULL);  }  cout << endl<<"이름 : ";  cin >> name;  cout << endl << "잔액 : ";  cin >> balance;  acnts[i].set(id, name, balance);  cout << "================================"<<endl<<endl;  }  cout <<"회수된 금액 : " << Account::getSum();  delete[] acnts;  return 0;  } |
|  |
| #include <iostream>  using namespace std;  class Complex {  double re, im;  public:  Complex(double r, double i)  {  re = r;  im = i;  }  Complex() { re = 0; im = 0; }  // + 연산자s  Complex operator+(Complex& other) {  Complex result(this->re + other.re, this->im + other.im);  return result;  }  Complex operator+(double num) {  Complex result(this->re + num, this->im);  return result;  }  // - 연산자  Complex operator-(Complex& other) {  Complex result(this->re - other.re, this->im - other.im);  return result;  }  Complex operator-(double num) {  Complex result(this->re - num, this->im);  return result;  }  void print() {  cout << re << " + i" << im << endl;  }  };  int main()  {  Complex a(.3, 8.4), b(4.5, 5.2), c(2.0, 7.7);  Complex sum, dif;  sum = a + b + 3.0; // 중요 : "3.0"과 정확히 동일하게 기입하여야 함  cout << "a + b + 3.0 = ";  sum.print();  dif = a - b - 2.0; // 중요 : "2.0"과 정확히 동일하게 기입하여야 함  cout << "a - b - 2.0 = ";  dif.print();  sum = a + b + 5 - c - 8; // 중요 : "5 - 8"과 정확히 동일하게 기입하여야 함  cout << "a + b + 5 - c - 8 = ";  sum.print();  return 0;  } |
| #include <iostream>  #include<string>  #include<algorithm>  using namespace std;  template<class T>  class Point {  public:  T x;  T y;  Point(T \_x, T \_y);  void display();  bool operator==(const Point& pt);  };  template<typename T>  Point<T>:: Point(T \_x, T \_y) : x(\_x), y(\_y) {}  template<typename T>  void Point<T>:: display() {  cout << "x: " << x << "\ty: " << y << endl;  }  template<typename T>  bool Point<T>:: operator ==(const Point& pt) {  if (this->x == pt.x && this->y == pt.y) return true;  else return false;  }  void get\_command(string& command) {  char input;  do {  cout <<"1: display all the points\n2: Count the value you want \n0: Exit the program/n" << "Enter your command: ";  cin >> input;  command = input;  } while (!isdigit(input)||  (int)input < 0 && (int)input > 3);  }  template <typename T>  void count\_val(Point<T>\* iter\_begin, Point<T>\* iter\_end) {  int cnt = 0;  cout << "Enter x, y of the point you want to count" << endl;  int x, y;  cout << "x: ";  cin >> x;  cout << "y: ";  cin >> y;  Point<int> seek(x, y);  for (auto cursor = iter\_begin; cursor != iter\_end; cursor++)  if (\*cursor == seek)  cnt++;  cout << "number of the values: " << cnt << endl;  }  int main() {  Point<int> myPoints[] = { {1, 2}, {-1, 5}, {3, 7}, {2, 4}, {1, 10}, {1, 2} };  string command;  Point<int> pt(1, 2);  while (1) {  get\_command(command);  if (command == "1") for\_each(begin(myPoints), end(myPoints), [](Point<int> x) {x.display(); });  else if (command == "2") count\_val<int>(begin(myPoints), end(myPoints));  else if (command == "0") break;  }  cout << "Exit program.." << endl;  return 0;  } |
| #include <iostream>  #include <string>  #include <iomanip>  using namespace std;  class Student //학생 클래스  {  private: //맴버 변수  string name; //학생 이름  string id; //학생 아이디  string code[100]; //들은 수업(course)의 code (최대 100개)  int numCourseTaken=0; //들은 수업의 총 개수  public:  Student() {}  Student(string \_name, string \_id) : name(\_name), id(\_id) {}  string getName() {  return name;  }  string getId() {  return id;  }  string getCode(int index) {  return code[index];  }  void setName(string \_name) {  name = \_name;  }  void setId(string \_id) {  id = \_id;  }  void addCourse(string \_code) {  code[numCourseTaken] = \_code;  numCourseTaken++;  }  friend class KHUIS;  friend ostream& operator<<(ostream& cout, Student st);  };  ostream& operator<< (ostream& cout, Student st) {  cout << "이름: " << st.getName() << "\t학번: " << st.getId() << endl;  return cout;  }  class Course //수업 클래스  {  private: //맴버 변수  string name; //수업의 이름  string code; //수업의 코드  public:  string getName() { return name; }  string getCode() { return code; }  void setName(string \_name) {  name = \_name;  }  void setCode(string \_code) {  code = \_code;  }  friend ostream& operator<<(ostream& cout, Course c);  friend class KHUIS;  };  ostream& operator<< (ostream& cout, Course c) {  cout << "과목명: " << c.name << "\t학수번호: " << c.code << endl;  return cout;  }  class KHUIS //종정시 클래스: 학생정보와 학생이 듣고있는 수업 정보를 출력, Student class와 Course class에서 friend class로 정의됨  {  public:  //학생의 정보를 입력으로 받아 학생정보 출력  //학생의 들은 수업(code)에 대한 수업 이름을 Course class에서 찾아서 출력  //numCourse(int)는 \_course 배열의 길이를 의미  void printInfo(Student& \_stu, Course\* \_course, int numCourse) {  cout << \_stu;  cout << "수강중인 과목: ";  for (int i = 0; i < \_stu.numCourseTaken; i++) {  for (int j = 0; j < numCourse; j++) {  if (\_course[j].code == \_stu.code[i]) cout << \_course[j].name << "\t";  }  }  cout << endl << endl;  }  };  int main() {  int numTotalCourse = 3; //총 과목의 개수  Course\* pCourse = new Course[numTotalCourse];  pCourse[0].setName("객체지향프로그래밍");  pCourse[0].setCode("CSE100");  pCourse[1].setName("영상처리");  pCourse[1].setCode("CSE200");  pCourse[2].setName("머신러닝");  pCourse[2].setCode("CSE300");  cout << pCourse[0].getCode() << '\t' << pCourse->getName() << endl; // 그림1. (1)  cout << setw(50) << setfill('-') << "" << endl;  cout << pCourse[0] << endl; // 그림1. (2)  cout << pCourse[1] << endl;  cout << pCourse[2] << endl;  cout << setw(50) << setfill('-') << "" << endl;  Student stu1("홍길동", "20181004"), stu2;  stu1.addCourse("CSE100");  stu1.addCourse("CSE200");  stu2.setName("김영희");  stu2.setId("20182000");  stu2.addCourse("CSE100");  stu2.addCourse("CSE300");  cout << stu1.getId() << '\t' << stu1.getName() << '\t' << stu1.getCode(0) << endl; // 그림1. (3)  cout << setw(50) << setfill('-') << "" << endl;  cout << stu1 << endl; // 그림1. (4)  cout << stu2 << endl;  cout << setw(50) << setfill('-') << "" << endl;  KHUIS khuis;  khuis.printInfo(stu1, pCourse, numTotalCourse); // 그림1. (5)  khuis.printInfo(stu2, pCourse, numTotalCourse);  delete[] pCourse;//pCourse 할당 해제 할 것  return 0;  } |