|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2018\_2\_C++ \_06 | 학번 : | 20147351 | 이름 : | 정지현 |

1. 복사 생성자

|  |
| --- |
| #include <iostream>  using namespace std;  class Circle {  private:  int radius;  public:  Circle(Circle& c); // 복사 생성자 선언  Circle() { radius = 1; }  Circle(int radius) { this->radius = radius; }  double getArea() { return 3.14\*radius\*radius; }  };  Circle::Circle(Circle& c) { // 복사 생성자 구현  this->radius = c.radius;  cout << "복사 생성자 실행 radius = " << radius << endl;  }  int main() {  Circle src(30); // src 객체의 보통 생성자 호출  Circle dest(src); // dest 객체의 복사 생성자 호출  cout << "원본의 면적 = " << src.getArea() << endl;  cout << "사본의 면적 = " << dest.getArea() << endl;  } |
| **[실행결과] - 복사 생성자** |
| **[실행결과] - 디폴트 복사 생성자** |

* 얕은 복사로 문제가 발생하는 예제

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <cstring>  using namespace std;  class Person { // Person 클래스 선언  char\* name;  int id;  public:  Person(int id, const char\* name); // 생성자  ~Person(); // 소멸자  void changeName(const char \*name);  void show() { cout << id << ',' << name << endl; }  };  Person::Person(int id, const char\* name) { // 생성자  this->id = id;  int len = strlen(name); // name의 문자 개수  this->name = new char[len + 1]; // name 문자열 공간 핟당  strcpy(this->name, name); // name에 문자열 복사  }  Person::~Person() {// 소멸자  if (name) // 만일 name에 동적 할당된 배열이 있으면  delete[] name; // 동적 할당 메모리 소멸  }  void Person::changeName(const char\* name) { // 이름 변경  if (strlen(name) > strlen(this->name))  return;  strcpy(this->name, name);  }  int main() {  Person father(1, "Kitae"); // (1) father 객체 생성  Person daughter(father); // (2) daughter 객체 복사 생성. 복사생성자호출  cout << "daughter 객체 생성 직후 ----" << endl;  father.show(); // (3) father 객체 출력  daughter.show(); // (3) daughter 객체 출력  daughter.changeName("Grace"); // (4) daughter의 이름을 "Grace"로 변경  cout << "daughter 이름을 Grace로 변경한 후 ----" << endl;  father.show(); // (5) father 객체 출력  daughter.show(); // (5) daughter 객체 출력  return 0; // (6), (7) daughter, father 객체 소멸  } |

* 얕은 복사의 문제를 해결한 깊은 복사

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <cstring>  using namespace std;  class Person { // Person 클래스 선언  char\* name; int id;  public:  Person(int id, const char\* name); // 생성자  Person(Person& person); // 복사 생성자  ~Person(); // 소멸자  void changeName(const char \*name);  void show() { cout << id << ',' << name << endl; }  };  Person::Person(int id, const char\* name) { // 생성자  this->id = id;  int len = strlen(name); // name의 문자 개수  this->name = new char[len + 1]; // name 문자열 공간 핟당  strcpy(this->name, name); // name에 문자열 복사  }  **Person::Person(Person& person) { // 복사 생성자**  **this->id = person.id; // id 값 복사**  **int len = strlen(person.name);// name의 문자 개수**  **this->name = new char[len + 1]; // name을 위한 공간 핟당**  **strcpy(this->name, person.name); // name의 문자열 복사**  **cout << "복사 생성자 실행. 원본 객체의 이름 " << this->name << endl;**  **}**  Person::~Person() {// 소멸자 코드 생략  }  void Person::changeName(const char\* name) { // 이름 변경  if (strlen(name) > strlen(this->name))  return; // 현재 name에 할당된 메모리보다 긴 이름으로 바꿀 수 없다.  strcpy(this->name, name);  }  int main() {  Person father(1, "Kitae"); // (1) father 객체 생성  Person daughter(father); // (2) daughter 객체 복사 생성. 복사생성자호출  cout << "daughter 객체 생성 직후 ----" << endl;  father.show(); // (3) father 객체 출력  daughter.show(); // (3) daughter 객체 출력  daughter.changeName("Grace"); // (4) daughter의 이름을 "Grace"로 변경  cout << "daughter 이름을 Grace로 변경한 후 ----" << endl;  father.show(); // (5) father 객체 출력  daughter.show(); // (5) daughter 객체 출력  return 0; // (6), (7) daughter, father 객체 소멸  } |

* 내용 점검

1. 교재 5장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 13 | MyClass(); // 디폴트 생성자 , MyClass(MyClass&);//복사 생성자 |
| 14 | 2. //값이 아니라 주소를 넘겨주는 것이다. 복사 생성자는 멤버 변수 값을 1:1로 복사하는 것. |
| 15 | 1. MyClass::~MyClass(){delete element;} 2. MyClass::MyClass(MyClass& b){   this->size = b.size;  element = b.element;  }   1. MyClass::MyClass(MyClass& b){   this->size = b.size;  element = new int[size];  for(int i = 0; i < size; i++)  element[i] = b.element[i];  } |
| 16 | 1. |
| 17 | Student::Student(Student& S){  name = S.name;  id = S.id;  grade = S.grade;  } |
| 18 | Student::Student(Student& S){  pName = S.pName;  pId = S.pId  grade = S.grade;  } |
| 19 | a=b; 디폴트 복사 생성자는 1:1로 멤버 변수의 값을 복사하는 것인데, a의 객체가 소멸되면 b의 객체는 정상이지만 객체 a가 소멸될 때 소멸자에 의해 동일한 값을 반환하게 되므 로 실행 오류가 발생하여 비정상적으로 종료된다. |

* 프로그램 과제

1. 교재 274p : 문제11번

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Book {  char \*title; // 제목 문자열  int price; // 가격  public:  Book(const char\* title, int price);  Book(Book& b); // (3) 정답. 깊은 복사 복사 생성자 작성  ~Book();  void set(const char\* title, int price);  void show() { cout << title << ' ' << price << "원" << endl; }  };  Book::Book(const char\*title, int price) { // (1) 정답  int len = strlen(title);  this->title = new char[len + 1];  strcpy(this->title, title);  this->price = price;  }  Book::~Book() { // (1) 정답  if (title)  delete[] title;  }  void Book::set(const char\* title, int price) { // (1) 정답  if (this->title) // 현재 객체에 title에 메모리가 할당되어 책 이름이 적혀있는 상황  delete[] this->title; // 현재 title에 할당된 메모리 반환  int len = strlen(title);  this->title = new char[len + 1]; // 새로 메모리 할당  strcpy(this->title, title);  this->price = price;  }  /\*  Book::Book(Book& b) { // (2) 정답. 디폴트 복사 생성자  title = b.title;  price = b.price;  }  \*/  Book::Book(Book& b) { // (3) 정답. 깊은 복사 복사 생성자 작성  int len = strlen(b.title);  title = new char[len + 1];  strcpy(title, b.title);  price = b.price;  }  int main() {  Book cpp("명품C++", 10000);  Book java = cpp;  java.set("명품자바", 12000);  cpp.show();  java.show();  char ch = getchar();  }  #include <iostream>  #include <string>  using namespace std;  class Book {  string title; // 제목 문자열  int price; // 가격  public:  Book(string title, int price);  ~Book();  void set(string title, int price);  void show() { cout << title << ' ' << price << "원" << endl; }  };  Book::Book(string title, int price) { // (4) 정답  this->title = title;  this->price = price;  }  Book::~Book() { // (4) 정답    }  void Book::set(string title, int price) { // (4) 정답  this->title = title;  this->price = price;  }  int main() {  Book cpp("명품C++", 10000);  Book java = cpp;  java.set("명품자바", 12000);  cpp.show();  java.show();  char ch = getchar();  } |
| **[실행 결과]**  **오류**    11-4 |

1. 교재 275p : 문제12번

|  |
| --- |
| **[프로그램 소스]**  #define \_CRT\_SECURE\_NO\_WARNINGS // 비주얼 스튜디오에서 stpcpy() 때문에 발생하는 컴파일 오류를 막기 위해  #include<iostream>  #include<string>  #include<cstring>  using namespace std;  class Dept {  int size; //scores 배열 크기  int \*scores;//동적 할당 받을 정수 배열의 정수  public:  Dept(int size) {// 생성자  this->size = size;  scores = new int[size];  }  Dept(Dept &dept); //복사 생성자  ~Dept(); // 소멸자  int getSize() {  return size;  }  void read(); // size만큼 키보드에서 정수를 읽어 scores 배열에 저장  bool isOver60(int index); // index의 학생의 성적이 60보다 크면 return  };  //12-1  void Dept::read() {  for (int i = 0; i < size; i++) { // size개수  cout << "10개 점수 입력 >> " ;  cin >> scores[i]; // 성적을 score배열에 입력  }  }  bool Dept::isOver60(int index) {  if (scores[index] > 60)  return true; // 성적이 60 이상이면 true  else  return false; // 성적이 60 미만이면 false  }  Dept::Dept(Dept &dept) { // 복사 생성자  this->size = dept.size; // size 복사  scores = new int[size]; // score를 위한 동적 할당  for (int i = 0; i < size; i++) { // size 만큼 배열 원소 복사  this->scores[i] = dept.scores[i];  }  }  int countPass(Dept dept) { // dept 학과 60점 이상으로 통과하는 학생의 수 리턴  int count = 0;  for (int i = 0; i < dept.getSize(); i++) {// size 만큼 for문을 돌리고 60점 이상일때 true -> count++  if (dept.isOver60(i))  count++;  }  return count;  }  int main() {  Dept com(10); // 총 10명이 있는 학과 com size = 10  com.read(); // 총 10명의 학생들의 성적을 키보드로부터 읽어 scores 배열에 저장    char ch = getchar();  int n = countPass(com); // com 학과에 60점 이상으로 통과한 학생 수를 return  cout << "60점 이상은 " << n << "명";    }  12-1  #define \_CRT\_SECURE\_NO\_WARNINGS // 비주얼 스튜디오에서 stpcpy() 때문에 발생하는 컴파일 오류를 막기 위해  #include<iostream>  #include<string>  #include<cstring>  using namespace std;  class Dept {  int size; //scores 배열 크기  int \*scores;//동적 할당 받을 정수 배열의 정수  public:  Dept(int size) {// 생성자  this->size = size;  scores = new int[size];  }    Dept(Dept &dept); //복사 생성자    ~Dept() {  delete[] scores;  }; // 소멸자    int getSize() {  return size;  }  void read(); // size만큼 키보드에서 정수를 읽어 scores 배열에 저장    bool isOver60(int index); // index의 학생의 성적이 60보다 크면 return  };  //12-1  void Dept::read() {  cout << "10개 점수 입력 >>";  for (int i = 0; i < size; i++) { // size개수  cin >> scores[i]; // 성적을 score배열에 입력  }  }  bool Dept::isOver60(int index) {  if (scores[index] > 60)  return true; // 성적이 60 이상이면 true  else  return false; // 성적이 60 미만이면 false  }  Dept::Dept(Dept &dept) { // 복사 생성자  this->size = dept.size; // size 복사  scores = new int[size]; // score를 위한 동적 할당  for (int i = 0; i < size; i++) { // size 만큼 배열 원소 복사  this->scores[i] = dept.scores[i];  }  }  int countPass(Dept dept) { // dept 학과 60점 이상으로 통과하는 학생의 수 리턴  int count = 0;  for (int i = 0; i < dept.getSize(); i++) {// size 만큼 for문을 돌리고 60점 이상일때 true -> count++  if (dept.isOver60(i))  count++;  }  return count;  }  int main() {  Dept com(10); // 총 10명이 있는 학과 com size = 10  com.read(); // 총 10명의 학생들의 성적을 키보드로부터 읽어 scores 배열에 저장  int n = countPass(com); // com 학과에 60점 이상으로 통과한 학생 수를 return  cout << "60점 이상은 " << n << "명";    char ch;  ch = getchar();  ch = getchar();  //ch = getchar();  //ch = getchar();  }  12-2  /\*소멸자가 실행 될 때, 디폴트 복사 생성자의 객체가 소멸하여 메모리를 반환하고, c  om객체가 소멸될 때 동적메모리가 중복되어 해제하려고 하기 때문에 오류가 발생한다.  즉! 깊은 복사 생성자를 작성하여 깊은 복사를 수행 시켜야만 한다.\*/  12-3  #define \_CRT\_SECURE\_NO\_WARNINGS // 비주얼 스튜디오에서 stpcpy() 때문에 발생하는 컴파일 오류를 막기 위해  #include<iostream>  #include<string>  #include<cstring>  using namespace std;  class Dept {  int size; //scores 배열 크기  int \*scores;//동적 할당 받을 정수 배열의 정수  public:  Dept(int size) {// 생성자  this->size = size;  scores = new int[size];  }  //Dept(Dept &dept); //복사 생성자  ~Dept() {  delete[] scores;  }; // 소멸자  int getSize() {  return size;  }    void read(); // size만큼 키보드에서 정수를 읽어 scores 배열에 저장    bool isOver60(int index);// index의 학생의 성적이 60보다 크면 return    int countPass();  };  //12-1  void Dept::read() {  cout << "10개 점수 입력 >>";  for (int i = 0; i < size; i++) { // size개수  cin >> scores[i]; // 성적을 score배열에 입력  }  }  bool Dept::isOver60(int index) {  if (scores[index] > 60)  return true; // 성적이 60 이상이면 true  else  return false; // 성적이 60 미만이면 false  }  /\*Dept::Dept(Dept &dept) { // 복사 생성자  this->size = dept.size; // size 복사  scores = new int[size]; // score를 위한 동적 할당  for (int i = 0; i < size; i++) { // size 만큼 배열 원소 복사  this->scores[i] = dept.scores[i];  }  }\*/  int Dept::countPass() { // dept 학과 60점 이상으로 통과하는 학생의 수 리턴  int count = 0;  for (int i = 0; i < getSize(); i++) {// size 만큼 for문을 돌리고 60점 이상일때 true -> count++  if (isOver60(i))  count++;  }  return count;  }  int main() {  Dept com(10); // 총 10명이 있는 학과 com  com.read(); // 총 10명의 학생들의 성적을 키보드로부터 읽어 scores 배열에 저장  int n = com.countPass(); // com 학과에 60점 이상으로 통과한 학생 수를 return  cout << "60점 이상은 " << n << "명";  char ch;  ch = getchar();  ch = getchar();  } |
| **[실행 결과]**  **오류**    12-1    12-2  /\*소멸자가 실행 될 때, 디폴트 복사 생성자의 객체가 소멸하여 메모리를 반환하고, c  om객체가 소멸될 때 동적메모리가 중복되어 해제하려고 하기 때문에 오류가 발생한다.  즉! 깊은 복사 생성자를 작성하여 깊은 복사를 수행 시켜야만 한다.\*/  12-3 |