|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_06 | 학번 : | 20135151 | 이름 : | 이갑성 |

* **강의 내용**

|  |
| --- |
| 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]** |

|  |
| --- |
| 1. 깊은 복사 |
| #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  } |
| **[실행결과]** |

|  |
| --- |
| 1. 디폴트 매개변수 |
| #include <iostream>  using namespace std;  class MyVector{  int \*p;  int size;  public:  MyVector(int n=100) {  p = new int [n];  size = n;  }  ~MyVector() { delete [] p; }  };  int main() {  MyVector \*v1, \*v2;  v1 = new MyVector(); // 디폴트로 정수 배열 100 동적 할당  v2 = new MyVector(1024); // 정수 배열 1024 동적 할당  delete v1;  delete v2;  } |
| **[실행결과]** |

|  |
| --- |
| 1. static 멤버 사용하기 |
| class Person {  public:  double money; // 개인 소유의 돈  void addMoney(int money) { this->money += money; }  static int sharedMoney; // 공금  static void addShared(int n) {  sharedMoney += n;  }  };  // static 변수 생성. 전역 공간에 생성  int Person::sharedMoney=10; // 10으로 초기화  int main() {  // static 멤버 접근, 공금=60, han 객체가 생기기전부터 static 멤버접근  **Person::addShared(50);**  cout << **Person::sharedMoney** << endl;  Person han;  han.money = 100;  han.sharedMoney = 200; // static 멤버 접근, 공금=200  **Person::sharedMoney = 300;** // static 멤버 접근, 공금=300  **Person::addShared(100);** // static 멤버 접근, 공금=400  cout << han.money << ' ' << **Person::sharedMoney** << endl;  } |
| **[실행결과]** |

* 내용 점검 – 6장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |

* 프로그램 과제

|  |
| --- |
| 1. 교재 274p 문제11번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 275p 문제12번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 313p 문제2번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 313p 문제3번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 316p 문제6번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 316p 문제7번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 318p 문제9번 |
| **[프로그램 소스]** |
| **[실행결과]** |