|  |
| --- |
| **2019년 10월 17일 실습보고서** |
| **학번 : 201810754, 이름 : 조윤직** |
| **실습자료1 : [복사 생성자]-** |
| **소스코드** |
| ArrayData arr2 = ArrayData::getCopyInstance(arr);  //Arraydata.h #pragma once class ArrayData { private: double\* data; int capacity; int used; static int count; public: static int getCount(); ArrayData(const int& capacity); ArrayData(const ArrayData& arr); ArrayData(ArrayData&& arr);  ArrayData(); ~ArrayData(); void addElement(double num); bool full() const; int getCapacity() const; int getUsed() const ; void emptyArray(); void showData() const; friend double getArraySum(const ArrayData& arr); ArrayData getObject() { return \*this; } ​ static ArrayData getCopyInstance(const ArrayData&copy) ; }; //Arraydata.cpp #include "ArrayData.h" #include<iostream> using namespace std; int ArrayData::getCount() { return count; } ArrayData::ArrayData(const int& capacity):capacity(capacity), used(0) { cout << this->capacity << "배열 생성\n"; count++; this->data = new double[capacity]; } ​ ArrayData::ArrayData(const ArrayData& arr) : capacity(arr.capacity),used(arr.used) { cout << this->capacity << "배열 복사\n"; count++; this->data = new double[this->capacity]; for (int i = 0; i < used; i++) data[i] = arr.data[i]; } ​ ArrayData::ArrayData(ArrayData&& copy) :capacity(copy.capacity), used(copy.used), data(copy.data) { cout << this->capacity << "이동 생성자\n"; copy.data = nullptr; } ​ ArrayData::ArrayData() :ArrayData(5) { } ​ int ArrayData::count = 0; ​ ​ ArrayData::~ArrayData() {  cout << this->capacity << "배열 삭제\n"; count--; if (data != nullptr) { delete[] this->data; data = nullptr; }  } ​ void ArrayData::addElement(double num) { if (!full()) this->data[used++] = num; else cout << "빈 공간이 없음\n";  } ​ bool ArrayData::full() const { return capacity==used; } ​ int ArrayData::getCapacity() const { return this->capacity; } ​ int ArrayData::getUsed() const { return this->used; } ​ void ArrayData::emptyArray() { this->used = 0; } ​ void ArrayData::showData() const { double\* p = this->data; double\* endp = p  + this->used; cout << "배열 출력 :"; while (p < endp) { cout << \*(p++)<<" ";  } cout << endl; } ​ ArrayData ArrayData::getCopyInstance(const ArrayData& copy) { ArrayData arr(copy); return arr; } ​ //main.cpp #include <iostream> #include <string> #include "ArrayData.h" using namespace std; ​ void showArray(ArrayData arr) { //여기서도 복사 생성자 arr.showData(); } double getArraySum(const ArrayData& arr) { double total = 0; for (int i = 0; i < arr.getUsed(); i++) { total += arr.data[i]; } return total; } ​ int main() { //cout << "201810754 조윤직\n"; //ArrayData arr1(3); //arr1.addElement(1.1); //arr1.addElement(3.1); ​ //ArrayData arr2 = arr1.getObject(); //ArrayData arr3 = ArrayData(10); ​ /\*ArrayData arr[] = {  ArrayData(10),ArrayData(20),ArrayData(30)  };\*/ ArrayData arr(10);    ArrayData arr2 = ArrayData::getCopyInstance(arr); system("pause"); } |
| **실행결과** |
| 인스 턴스 멤버 변수에 포인터(배열등), 의 주소 값이 담겨 깊은 복사가 필요한 클래스 의 경우 깊은 복사를 위해 깊은 복사 생성자를 둔다. 하지만 임시로 만든 객체를 복사할때는 같은 주소값을 그냥 담아도 상관이 없기 때문에 r-value참조를 이용한 '이동 생성자'를 사용하여 임시 클래스 객체를 객체에 복사(이동)한다.  ArrayData arr2 = ArrayData::getCopyInstance(arr); |
| **실습자료2 : []** |
| **소스코드** |
|  |
| **실행결과** |
|  |
| **실습자료3 : []-** |
| **소스코드** |
|  |
| **실행결과** |
|  |