소스 파일 구현 설명

202304148 조하린

▶ 다음은 학과를 나타내는 Dept 클래스와 이를 활용하는 main()을 보여준다.

```
class Dept {
   int size; // scores 배열의 크기
   int* scores; // 동적 할당 받을 정수 배열의 주소
```

```
public:
   Dept(int size) { // 생성자
      this->size = size;
      scores = new int[size];
   }
   Dept(const Dept& dept); // 복사 생성자
   ~Dept(); // 소멸자
   int getSize() { return size; }
   void read(); // size 만큼 키보드에서 정수를 읽어 scores 배열에 저장
   bool isOver60(int index); // index의 학생의 성적이 60보다 크면 true 리턴
};
```

```
int countPass(Dept dept) { // dept 학과에 60점 이상으로 통과하는 학생의 수 리턴
   int count = 0;
   for (int i = 0; i < dept.getSize(); i++) {
      if (dept.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 = countPass(com); // com 학과에 60점 이상으로 통과한 학생의 수를 리턴
   cout << "60점 이상은 " << n << "명";
}
```

(1) main() 실행 결과가 다음과 같이 되도록 Dept 클래스에 멤버들을 모두 구현하고, 전체 프로그램을 완성하라.

```
10개 점수 입력>> 10 20 30 40 50 60 70 80 90 100
60점 이상은 4명
```

(3) Dept 클래스에 복사 생성자를 제거하라. 복사 생성자가 없는 상황에서도 실행 오류가 발생하지 않게 하려면 어느 부분을 수정하면 될까? 극치 일부분의 수정으로 해결된다. 코드를 수정해보라.

문제 정의

총 10명이 있는 학과 com이 있다. 이 학생들의 성적을 키보드로부터 읽어 scores 배열에 저장하고 com 학과에 60점 이상으로 통과한 학생의 수를 리턴하는 문제이다. 추가로 복사 생성자가 없는 상황에서도 오류가 발생하지 않게 코드를 수정한다.

1. 클래스 : 학생들의 성적 정보를 Dept 클래스로 작성하여 배열로 관리한다. 배열 scores와 변수 size를 멤버로 가지고 getSize(), read(), isOver60(), countPass() 멤버 함수를 구현한다.
   (생성자 Dept(), 복사 생성자 Dept(const), 소멸자 ~Dept() 필요에 따라 사용)

2. 입력 : 키보드로부터 학생 10명의 점수를 입력받는다. read() 함수를 통해 학생 10명의 점수를 한꺼번에 모두 받고, 각각의 점수는 띄어쓰기로 구분한다. 입력받은 점수는 scores 배열에 저장된다.

3. 반복문과 조건문
   1) read() 함수에서 반복문을 통해 배열에 크기만큼 반복하여 학생들의 점수를 입력받아 차례대로 scores 배열에 저장한다.
   2) countPass() 함수에서 반복문을 통해 배열에 크기만큼 반복하여 60점 이상인 학생의 수를 계산한다.

4. 객체 전달 방식
   1) 값으로 전달 - 깊은 복사
      복사 생성자와 : Dept 객체가 얕은 복사가 되는 것을 방지하기 위해 정의한다. 원본 객체의 scores 배열을 새로운 배열로 깊은 복사를 하여 충돌을 방지한다.
   2) 참조로 전달 - 얕은 복사
      얕은 복사로 인해 두 객체가 동일한 메모리 주소를 가리키게 되어 메모리 누수나 잘못된 접근이 발생하는 것 방지한다.

파일에 대한 설명

Dept 프로젝트 - 복사 생성자가 정의된 버전

(1) dept.h
    복사 생성자가 정의된 Dept 클래스의 헤더 파일
    멤버 변수: int size, int* scores
    멤버 함수: 생성자 Dept(), 복사 생성자 Dept(), 소멸자 ~Dept(),
        getSize(), read(), isOver60(), countPass()
(2) Dept.cpp
    복사 생성자가 정의된 Dept 클래스의 멤버 함수 구현
    멤버 함수: 생성자 Dept(), 복사 생성자 Dept(), 소멸자 ~Dept(),
        getSize(), read(), isOver60(), countPass()
(3) main.cpp
    프로그램 시작
    복사 생성자가 정의된 Dept 클래스의 객체 com 생성
    함수 호출: read(), countPass()
    결과 출력

NoCconstDept 프로젝트 - 복사 생성자가 정의되지 않은 버전

(4) noCconstDept.h
    복사 생성자가 정의되지 않은 Dept 클래스의 헤더 파일
    멤버 변수: int size, int* scores
    멤버 함수: Dept(), ~Dept(), getSize(), read(), isOver60(), countPass()
(5) NoCconstDept.cpp
    복사 생성자가 정의되지 않은 Dept 클래스의 멤버 함수 구현
    멤버 함수: Dept(), ~Dept(), getSize(), read(), isOver60(), countPass()
(6) mainCC.cpp
    프로그램 시작
    복사 생성자가 정의되지 않은 Dept 클래스의 객체 com 생성
    함수 호출: read(), countPass()
    결과 출력

문제 해결 방법 (아이디어)

(1) 복사 생성자가 정의된 Dept 프로젝트

1. 클래스 - 학생들의 성적 정보 관리 Dept 클래스
   - 멤버 변수
     - int size: scores 배열의 크기
     - int* scores: 동적 할당된 배열의 주소
   - 멤버 함수
     - 학생 수 반환하는 getSize() 함수
     - 학생들의 점수 입력받고 저장하는 read() 함수
     - 점수가 60점 이상인지 판단하는 isOver60() 함수
     - 점수가 60점 이상인 학생 수 계산하는 countPass() 함수

2. 함수
   - Dept(int size): 배열 초기화하는 생성자
   - Dept(const Dept& dept): 깊은 복사하기 위한 복사 생성자
   - ~Dept(): 동적 메모리 해제하기 위한 소멸자
   - getSize(): 학생 수 return
   - read(): 저장된 선수 name 변수 반환
   - isOver60(int index): 점수가 60점 이상인지 판단하는 함수
   - countPass(Dept dept): 60점 이상인 학생 수 계산, Dept 객체 값으로 전달

3. 배열
   - scores = new int[size] 동적할당
   - delete[] scores 소멸자 통해 동적 메모리 해제

4. 키보드 입력
   - 키보드로부터 학생 10명의 점수를 입력받고 scores 배열에 저장

5. this 포인터
   - this->size, this->scores 현재 객체의 변수와 매개변수

6. 복사 생성자
   - Dept(const Dept& dept)
   - 깊은 복사를 통해 동적 배열 scores 할당

(3) 복사 생성자가 정의되지 않은 NoCconstDept 프로젝트

1. 클래스 - 학생들의 성적 정보 : Dept 클래스
   - 멤버 변수
     - int size: scores 배열의 크기
     - int* scores: 동적 할당된 배열의 주소
   - 멤버 함수
     - 학생 수 반환하는 getSize() 함수
     - 학생들의 점수 입력받고 저장하는 read() 함수
     - 점수가 60점 이상인지 판단하는 isOver60() 함수
     - 점수가 60점 이상인 학생 수 계산하는 countPass() 함수

2. 함수
   - Dept(int size): 배열 초기화하는 생성자
   - ~Dept(): 동적 메모리 해제하기 위한 소멸자
   - getSize(): 학생 수 return
   - read(): 저장된 선수 name 변수 반환
   - isOver60(int index): 점수가 60점 이상인지 판단하는 함수
   - countPass(Dept& dept): 60점 이상인 학생 수 계산, Dept 객체 참조로 전달

3. 배열
   - scores = new int[size] 동적할당
   - delete[] scores 소멸자 통해 동적 메모리 해제

4. 키보드 입력
   - 키보드로부터 학생 10명의 점수를 입력받고 scores 배열에 저장

5. this 포인터
   - 현재 객체를 가리키는 포인터
   - 함수의 매개변수가 자기 자신에 할당되는 경우 사용
   - this->size는 현재 객체의 size 변수, size는 함수의 매개변수
   - this->scores는 현재 객체의 scores 변수, scores는 함수의 매개변수

6. 참조
   - countPass(Dept& dept) - Dept 객체 참조로 전달

문제 해결 키 (아이디어)

객체 전달 방식: 값으로 전달 vs 참조로 전달

(1) 값으로 전달 - 깊은 복사
    복사 생성자 Dept(const Dept& dept)
    깊은 복사를 통해 동적 배열 scores 할당

(3) 참조로 전달 - 얕은 복사
    countPass(Dept& dept): Dept 객체 참조로 전달

프로그램 순서

1. 프로그램 시작
2. 학생 10명의 점수 입력
3. read() 함수 호출하여 점수 scores 배열에 저장
4. countPass() 함수 호출하여 60점 이상인 학생 수 계산
5. 60점 이상인 학생 수 출력
6. 프로그램 종료

아이디어 평가

1. 객체 전달 방식: 값으로 전달 vs 참조로 전달
   - Dept 프로젝트 - 값으로 전달
     복사 생성자를 통해 깊은 복사를 하게 된다. 그로 인해 원본 객체와 복사된 객체가 각각 독립적인 배열을 가지게 되므로 메모리 충돌을 방지하여 안전하게 메모리를 관리할 수 있다.
     각각 독립적인 배열을 가지게 되면 메모리 누수가 발생할 수 있는데 소멸자의 delete[]를 통해 할당된 메모리를 해제하여 메모리 누수를 방지할 수 있다.
   - NoCconstDept 프로젝트 - 참조로 전달
     복사 생성자를 정의하지 않으면 기본 복사 생성자가 호출되어 얕은 복사를 하게 된다. 그로 인해 두 객체가 동일한 메모리 주소를 가리키게 되어 메모리 누수나 잘못된 접근이 발생할 수 있다. 이 문제를 해결하기 위해 객체를 참조로 (Dept&) 전달하였다.

2. this 포인터
   this를 사용하지 않다면 클래스의 멤버 변수와 메서드의 매개 변수 이름이 동일해서 두 개의 변수를 만들어야 하는데, this를 사용함으로써 코드의 가독성을 높이고 변수들의 혼동을 줄일 수 있었다.

3. 파일 분리
   파일 분리를 통해 각 파일의 역할에 맞게 나뉘어 코드의 가독성을 향상할 수 있었다. 또한 코드를 재사용할 수 있고, 각 클래스를 독립적으로 관리할 수 있게 되었으며 디버깅에 용이해졌다. (Dept 프로젝트로 설명)
   dept.h : 학생들의 성적 정보 관리 Dept 클래스 정의
   멤버 변수: int size, int* scores
   멤버 함수: 생성자 Dept(), 복사 생성자 Dept(), 소멸자 ~Dept(),
   getSize(), read(), isOver60(), countPass()
   Dept.cpp : 복사 생성자가 정의된 Dept 클래스의 멤버 함수 구현
   main.cpp
   복사 생성자가 정의된 Dept 클래스의 객체 com 생성
   함수 호출: read(), countPass()