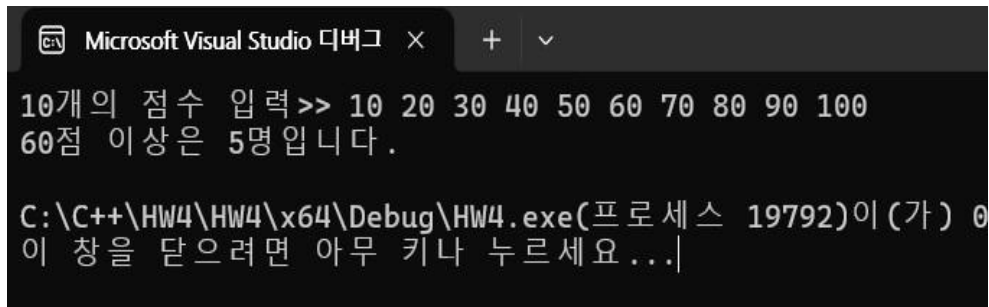


객체지향프로그래밍 과제#3

소프트웨어전공 202284012 김주원 / 소프트웨어전공 202304012 김도연

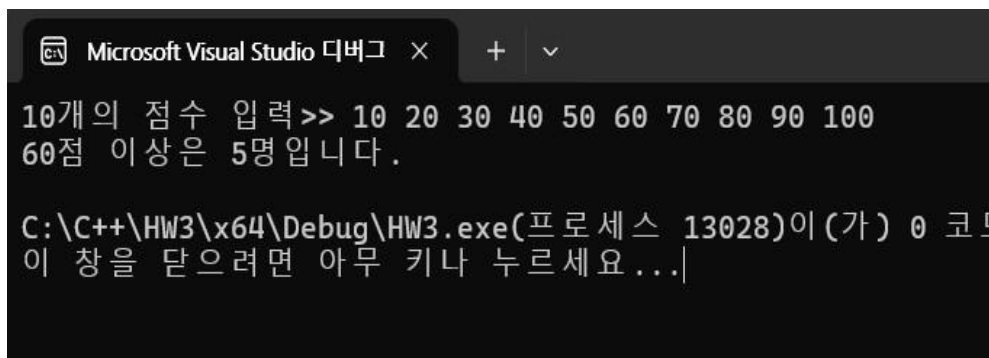
1. 소스 수행 결과 화면

(1)번



```
Microsoft Visual Studio 디버그 × + v
10개의 점수 입력>> 10 20 30 40 50 60 70 80 90 100
60점 이상은 5명입니다.
C:\C++\HW4\HW4\x64\Debug\HW4.exe(프로세스 19792)이(가) 0
이 창을 닫으려면 아무 키나 누르세요 ...|
```

(3)번



```
Microsoft Visual Studio 디버그 × + v
10개의 점수 입력>> 10 20 30 40 50 60 70 80 90 100
60점 이상은 5명입니다.
C:\C++\HW3\x64\Debug\HW3.exe(프로세스 13028)이(가) 0 코
이 창을 닫으려면 아무 키나 누르세요 ...|
```

2. 문제 정의

(1)번

Dept 클래스를 구현하여 학생 성적을 동적 배열로 관리하는 프로그램을 작성하는 문제이다. 이 클래스는 생성자, 소멸자, 성적 입력 및 60점 이상인 학생 수를 반환하는 함수를 포함하고 60점 이상의 학생 수를 계산하여 결과를 출력하는 기능을 구현한다.

(3)번

(1)번 코드에서 복사 생성자를 제거하여 실행 오류가 발생하지 않도록 수정하는 문제이다. 동적 메모리(int* scores)를 사용하는 Dept 클래스에서 복사 생성자 없이 객체가 복사되지 않도록 해야하는데 이를 위해 객체를 함수에 참조로 전달하여 복사 생성자 호출을 방지하고 소멸자에서 할당된 메모리를 적절히 해제하여 메모리 관리가 이루어지도록 한다.

3. 문제 해결 방법 - (1)번 문제

<Dept 클래스 선언부 - Dept.h>

1. 기본 로직

Dept 클래스는 학생들의 성적을 관리하고 60점 이상인 학생의 수를 계산한다. 이 클래스는 동적 배열을 사용하여 성적 데이터를 저장하고 다양한 메서드를 통해 성적 입력과 결과 계산을 수행한다. 생성자, 복사 생성자, 소멸자, 학생 수 반환 함수, 60점 이상인 학생 수 반환 함수, 학생 성적 입력 함수의 선언이 포함된다.

2. 헤더 가드

#ifndef DEPT_H, #define DEPT_H, #endif를 사용하여 구성된 헤더 가드는 헤더 파일이 여러 번 포함되더라도 컴파일 에러가 발생하지 않도록 방지함

3. 함수 구현

멤버 변수

- int size: 학생들의 성적을 저장할 배열의 크기, 즉 학생 수를 저장
- int* scores: 학생들의 성적을 저장할 동적 배열의 주소를 가리킴
- int passCount: 점수가 60점 이상인 학생의 수를 저장

멤버 함수

- int getSize() const: 학생 수를 반환하는 함수
- int getPassCount() const: 점수가 60점 이상인 학생 수를 반환하는 함수
- void read(): 학생들의 성적을 입력받아 scores 배열에 저장하고 60점 이상인 학생 수를 계산

생성자

- Dept(int size): 객체를 생성할 때 학생 수를 받아 size와 scores 배열을 초기화

복사 생성자

- Dept(const Dept& dept): 다른 Dept 객체를 복사할 때 호출되며 얇은 복사가 아닌 깊은 복사를 수행

소멸자

- ~Dept(): 객체가 소멸할 때 호출되어 동적 할당된 메모리를 해제

4. 헤더 파일 보호

헤더 파일이 여러 번 포함되는 것을 방지하고 코드의 안정성과 가독성을 높이기 위해 #endif 사용

<Dept 클래스 구현부 – Dept.cpp>

1. 기본 로직

Dept 클래스의 모든 멤버 함수의 내부 동작을 정의한다. Dept.h 파일에서 선언된 Dept 클래스의 동적 메모리 관리, 깊은 복사, 성적 입력과 처리, 결과 반환 등의 구체적인 로직을 구현하여 Dept 클래스가 올바르게 동작하도록 만든다.

2. 헤더 파일을 포함하는 구문

#include " Dept.h"를 통해 Dept.h에서 선언된 Dept 클래스의 정의를 포함하여 해당 클래스의 멤버 변수와 함수들에 접근할 수 있게 함

#include <iostream>으로 cout, cin과 같은 표준 입출력 객체를 사용하기 위한 라이브러리 포함

using namespace std;를 통해 std 네임스페이스에 있는 식별자를 쉽게 사용하도록 설정

3. 함수 구현

생성자 (Dept::Dept(int size))

- size 변수는 학생 수를 저장하며 이 값에 따라 scores 배열을 동적으로 할당
- passCount는 60점 이상인 학생 수를 세는 역할을 하므로 생성자에서 이를 0으로 초기화

복사 생성자 Dept::Dept(const Dept& dept)

- 다른 Dept 객체로부터 값을 복사할 때 호출되며 깊은 복사를 수행
- new 키워드 사용하여 동적으로 할당된 scores 배열을 새로운 메모리 공간에 할당한 후 각 배열 요소를 복사
- 이를 통해 두 객체가 독립적으로 메모리를 관리하도록 보장하여 얕은 복사에서 발생할 수 있는 메모리 문제를 예방함

소멸자 Dept::~Dept()

- 객체가 소멸할 때 호출되어 동적으로 할당된 메모리(scores 배열)를 delete[] 연산자를 사용하여 해제
- 이를 통해 메모리 누수를 방지할 수 있음

학생 수 반환 함수 Dept::getSize()

- 객체에 저장된 학생 수(size)를 반환하는 간단한 함수로 객체의 상태를 변경하지 않기 때문에 const로 선언됨

60점 이상인 학생 수 반환 함수 Dept::getPassCount()

- 이는 60점 이상인 학생의 수를 의미하는 passCount 값을 반환

성적 입력 및 계산 함수 Dept::read()

- 학생들의 성적을 입력받아 scores 배열에 저장하고 입력된 성적이 60점 이상일 경우 passCount를 증가시킴
- 반복문을 사용하여 학생 수(size)만큼 성적을 입력받고 if 문을 통해 각 성적이 60점 이상인지 검사

<main 함수 – main.cpp>

1. 기본 로직

Dept 클래스를 사용하여 10명의 학생 성적을 입력받고 60점 이상인 학생 수를 계산한 후 그 결과를 출력한다. 프로그램 종료 시 동적으로 할당된 메모리는 소멸자를 통해 자동으로 해제된다.

2. 헤더 파일을 포함하는 구문

#include " Dept.h"를 통해 Dept.h에서 선언된 Dept 클래스의 정의를 포함하여 해당 클래스의 멤버 변수와 함수들에 접근할 수 있게 함

#include <iostream>으로 cout, cin과 같은 표준 입출력 객체를 사용하기 위한 라이브러리 포함

using namespace std;를 통해 std 네임스페이스에 있는 식별자를 쉽게 사용하도록 설정

3. 함수 구현

Dept 객체 생성

- Dept com(10);은 10명의 학생 성적을 저장할 com 객체를 생성
- 이때 size는 10으로 설정되고 동적으로 10개의 정수를 저장할 배열이 생성됨

성적 입력

- com.read();를 호출하여 사용자가 학생들의 성적을 입력할 수 있도록 함
- 입력받은 성적 중 60점 이상인 경우 passCount를 증가시킴

결과 출력

- cout << "60점 이상은 " << com.getPassCount() << "명입니다."; 구문을 통해 60점 이상인 학생 수를 출력

프로그램 종료

- 프로그램 종료 시 com 객체가 소멸하며 소멸자가 호출되어 동적으로 할당된 메모리(scores 배열)가 안전하게 해제됨

4. 문제 해결 방법 - (3)번 문제

<Dept 클래스 선언부 - Dept.h>

1. 기본 로직

Dept 클래스는 학생 성적을 관리하기 위해 설계되었으며 동적 메모리를 사용하여 성적 배열(scores)을 포함한다. 이 클래스는 학생 수, 60점 이상인 학생 수, 성적 입력 및 확인 기능을 제공한다. **복사 생성자를 정의하지 않고 객체를 참조로 전달하여** 구현함으로써 객체 복사가 필요할 때마다 복사 생성자가 호출되는 것을 방지한다. 이를 통해 복사 생성자로 인한 메모리 관리 문제를 예방할 수 있다.

2. 헤더 가드

#ifndef DEPT_H, #define DEPT_H, #endif를 사용하여 구성된 헤더 가드는 헤더 파일이 여러 번 포함되더라도 컴파일 에러가 발생하지 않도록 방지함

3. 함수 구현

멤버 변수

- int size: scores 배열의 크기 (학생 수)
- int* scores: 동적 할당된 정수 배열의 포인터
- int passCount: 60점 이상의 학생 수

멤버 함수

- 생성자와 소멸자는 메모리 할당과 해제 담당
- getSize(): 학생 수 반환
- getPassCount(): 60점 이상인 학생 수 반환
- read(): 성적 입력
- isOver60(int index): 특정 학생의 성적 확인

생성자 Dept(int size)

- 입력받은 size를 사용하여 **scores** 배열을 동적으로 할당
- 초기값으로 **passCount**를 0으로 설정

소멸자 ~Dept()

- scores 배열을 해제하여 메모리 누수를 방지

getSize() 메소드

- size를 반환하여 학생 수를 제공

getPassCount() 메소드

- passCount를 반환하여 60점 이상의 학생 수를 제공

read() 메소드

- 사용자로부터 학생 성적을 입력받아 scores 배열에 저장
- 입력이 완료된 후 passCount를 계산하고 이때 입력값의 유효성을 검사하여 잘못된 입력이 발생하지 않도록 처리함

isOver60(int index) 메소드

- 주어진 인덱스의 성적이 60점 이상인지 확인하여 결과를 반환

예외 처리

- read() 메소드에서 입력값의 유효성을 검사하여 잘못된 입력이 발생하지 않도록 처리함

const 키워드

코드의 안전성을 높이기 위해 메소드를 const로 선언하여 사용자가 해당 메소드 호출해도 객체의 멤버 변수 변경되지 않도록 함

4. 헤더 파일 보호

헤더 파일이 여러 번 포함되는 것을 방지하고 코드의 안정성과 가독성을 높이기 위해 #endif 사용

<Dept 클래스 구현부 - Dept.cpp>

1. 기본 로직

Dept 클래스의 메서드 구현을 포함하고 있으며 학생 성적을 관리하고 처리하는 기능을 제공한다. 클래스의 생성자, 소멸자, 성적 입력, 학생 수 및 60점 이상인 학생 수를 확인하는 기능을 포함한다.

2. 헤더 파일을 포함하는 구문

#include " Dept.h"를 통해 Dept.h에서 선언된 Dept 클래스의 정의를 포함하여 해당 클래스의 멤버 변수와 함수들에 접근할 수 있게 함

#include <iostream>으로 cout, cin과 같은 표준 입출력 객체를 사용하기 위한 라이브러리 포함

using namespace std;를 통해 std 네임스페이스에 있는 식별자를 쉽게 사용하도록 설정

3. 함수 구현

멤버 변수

- size: 동적 배열의 크기로 학생 수를 나타냄
- scores: 학생 성적을 저장하는 동적 할당 배열
- passCount: 60점 이상인 학생 수를 카운트

멤버 함수

- 생성자 Dept::Dept(int size)
- 소멸자 Dept::~~Dept()
- getSize() const
- getPassCount() const
- read()
- isOver60(int index) const

생성자 구현

- `size`를 매개변수로 받아 객체를 초기화
- `this->size`를 사용하여 현재 객체의 멤버 변수와 매개변수를 구분
- `new` 키워드 사용하여 `scores` 배열을 동적으로 할당
- `passCount`를 0으로 초기화하여 60점 이상인 학생의 수를 카운트할 준비

소멸자 구현

- `scores` 배열을 해제하여 메모리 누수를 방지
- `delete[]` 연산자를 사용하여 할당된 메모리를 안전하게 해제

getSize()와 getPassCount() 메소드 구현

- `getSize()` 메소드는 학생 수 반환
- `getPassCount()` 메소드는 60점 이상인 학생 수 반환

read() 메소드 구현

- `for` 루프 사용하여 `size` 만큼 반복하여 학생의 성적을 사용자에게 입력받음
- 입력된 성적을 `scores` 배열에 저장
- `if` 문 사용하여 각 성적이 60점 이상인 경우 `passCount`를 증가시킴

isOver60(int index) 메소드 구현

- `index`는 성적을 확인할 학생의 인덱스
- 특정 학생의 성적을 확인하고 60점 이상인지 확인하여 결과를 반환
- 인덱스가 유효한 범위 내에 있는지 검사하여 오류를 방지하고 참이면 `true` 거짓이면 `false`가 반환됨

const 키워드

- 메소드에 `const` 키워드가 붙임으로써 객체의 상태가 변경되지 않음을 보장함
- 객체의 불변성을 유지하여 코드를 더 안전하고 직관적으로 만들어 줌

<main 함수 – main.cpp>

1. 기본 로직

Dept 객체를 생성하고 10명의 학생의 성적을 입력받은 후 **countPass** 함수를 호출하여 60점 이상의 학생 수를 계산하고 그 결과를 출력한다.

2. 헤더 파일을 포함하는 구문

#include " Dept.h"를 통해 Dept.h에서 선언된 Dept 클래스의 정의를 포함하여 해당 클래스의 멤버 변수와 함수들에 접근할 수 있게 함

#include <iostream>으로 cout, cin과 같은 표준 입출력 객체를 사용하기 위한 라이브러리 포함

using namespace std;를 통해 std 네임스페이스에 있는 식별자를 쉽게 사용하도록 설정

3. 함수 구현

countPass 함수 정의

- 매개변수로 참조 전달 - Dept 객체를 const 참조로 전달하여 복사 생성자가 호출되는 것을 방지하고 객체의 상태를 변경하지 않도록 함
- **dept.getSize()**를 사용하여 총 학생 수를 가져오고 for 루프를 통해 각 학생의 성적을 확인
- **dept.isOver60(i)**를 호출하여 특정 학생의 성적이 60점 이상인지 확인하고 조건을 만족할 경우 **count**를 증가시킴
- **return count;**로 60점 이상의 학생 수를 계산하여 반환

main 함수 정의

- **Dept com(10)**으로 Dept 클래스의 객체 com을 생성하며 10명의 학생을 가정
- **com.read()** 메소드를 호출하여 사용자로부터 학생들의 성적을 입력받음 – 이 과정에서 Dept 클래스의 read() 메소드가 실행됨
- **countPass(com)**을 호출하여 60점 이상의 학생 수를 계산하고 결과를 n에 저장
- cout을 사용하여 60점 이상인 학생 수를 출력
- main 함수 종료 시 com 객체의 소멸자가 자동으로 호출되어 동적 할당된 scores 배열이 해제됨