과제 명세

1. 내용

- PDF 강의자료 Chapter11_객체의다양한활용 27p, 예제 11-11 참조
- (1) CStudent 클래스 설계 및 구현
- (2) 정적멤버의 사용
- (3) 주석으로 코드 설명 (자세히)
- 클래스 설계, 구현, 테스트 구분하여 작성

2. 구현 조건

- 클래스명은 CStudent로 지정
- 이름(m_strName), 핸드폰번호(m_strPhoneNumber), 이메일주소(m_strEmail)를 저정할 멤버변수 3개
- 생성자, get/set 멤버함수, const 멤버함수, 인라인 함수 구현
- 각 객체의 멤버변수값을 화면에 출력하는 멤버함수(printStudentInfo)구현
- 정적변수 s_nCountObject를 선언하되 등록한 학생이 있을 때마다 등록한 학생의 수를 세려고 객체가 하나씩 생성될 때마다 자동 호출되는 생성자에서 1씩 증가
- 함수 오버로딩 또는 기본 매개변수 사용
- 테스트의 경우 객체 3개이상 사용하여 모든 멤버함수 이용한 결과 제시

3. 제출

- 클래스 헤더파일 및 소스파일
- 테스트 파일
- header file(.h) 1개, source file(.cpp) 2개, 총 3개 파일 압축(학번_이름.zip)하여 업로드

4. 기한

- 2020.04.20.월까지

과제 풀이

그냥 학생 하나를 1개의 인스턴스로 보고,

학생 3명의 정보를 객체를 통해 다룰 수 있는지 묻는 문제였다.

```
int main()
{
```

```
    /* 생성자가 잘 호출되고 인스턴스가 정상적으로 생성되는가 */

    CStudent *student1 = new CStudent("name1", "010-1111-1111",
"student1@naver.com"); // 학생1 : 포인터로
    CStudent::PrintCurrentDataCount();
    cout << endl;
    CStudent student2("name1", "010-1111-1112", "student1@naver.com"); // 학생2
    CStudent::PrintCurrentDataCount();
    cout << endl;
    CStudent student3("name1", "010-1111-1113", "student1@naver.com"); // 학생3
    cout << endl;
    CStudent::PrintCurrentDataCount();
    cout << endl;


    /* 멤버 함수 및 멤버 변수가 잘 동작하는가 */

    student1->set_strPhoneNumber("NAN"); // set_strPhoneNumber Test
    student1->PrintStudentInfo();
    cout << endl;
    student2.set_strEmail("student2@gmail.com"); // set_strEmail Test
    student2.PrintStudentInfo();
    cout << endl;
    student3.set_strName("name3"); // set_strName Test
    student3.PrintStudentInfo();
    cout << endl;
    CStudent::PrintCurrentDataCount();


    /* 객체 제거가 잘 동작하는가 */

    delete student1; // 동적할당을 통해 생성한 객체 제거
    cout << endl;
    CStudent::PrintCurrentDataCount();
    cout << endl;
    student2.~CStudent(); // 일반 인스턴스 소멸자 호출을 통한 객체 제거
    student2.PrintStudentInfo();
    cout << endl;
    CStudent::PrintCurrentDataCount();
}
```

```
class CStudent
{

private:
    string m_strName;
    string m_strPhoneNumber;
    string m_strEmail;

    // 고유한 값이므로 const 상수
    const int m_kId;


public:
```

```
    // 외부에서 초기화를 해주기 위해 public 으로 선언
    static int s_nCountObject;
    CStudent(string Name, string PhoneNumber, string Email, int id =
s_nCountObject);
    ~CStudent() { s_nCountObject--; };

    // 간단한 함수에 한해 inline 으로 구현

    // getter, inline function
    string get_strName() const { return m_strName; };
    string get_strPhoneNumber() const { return m_strPhoneNumber; };
    string get_strEmail() const { return m_strEmail; };
    static int get_nCountObject() { return s_nCountObject; }

    // setter, inlnine function
    void set_strName(string strName) { m_strName = strName; };
    void set_strPhoneNumber(string strPhoneNumber) { m_strPhoneNumber =
strPhoneNumber; };
    void set_strEmail(string strEmail) { m_strEmail = strEmail; };

    // methods
    void PrintStudentInfo() const;
    static void PrintCurrentDataCount() { cout << "Current " << s_nCountObject
<< " data exist in the storage." << endl; };
};
```

```
int CStudent::s_nCountObject = 0;

CStudent::CStudent(string Name, string PhoneNumber, string Email, int id):
m_kId(id+1)
// parameter : Name, Phone Number, Email
// default parameter : id 를 현재 저장되어 있는 데이터 + 1 로 초기화하도록 유도
// const member initialization : 콜론 초기화
{
    m_strName = Name;
    m_strPhoneNumber = PhoneNumber;
    m_strEmail = Email;
    s_nCountObject++;
}


void CStudent::PrintStudentInfo() const
// 현재 상태를 바꿀 권한을 주어서는 안 되는 함수이므로 const 선언
{
    cout << "student info ------: " << endl;
    cout << "m_strName----------: " << m_strName << endl;
    cout << "m_strPhoneNumber---: " << m_strPhoneNumber << endl;
    cout << "m_strEmail---------: " << m_strEmail << endl;
    cout << "m_kid--------------: " << m_kId << endl;
}
```

- 포인터로 객체를 선언하는 것은 new 와 함께하고, 정리하는 것은 delete 로 함.
- 객체 소멸자를 호출하면 어느정도 정리해 줌.
- const 와 static 은 함께 사용할 수 없음.
- const 멤버변수를 초기화할 때에는 콜론 초기화를 이용해야 함.
- 기본 파라미터를 설정할 때에는, '선언' 에 설정하고 '정의' 에는 적어넣으면 안 됨.
- static 멤버 변수를 클래스 밖에서 초기화해야, 전역변수 영역에 메모리가 잡힘.

유의사항

이름 규칙

참조

- NHN/C++ 코딩 규칙
- 안용학 교수님이 수업에서 언급했던 규칙
- 과제 명세
- Google C++ Style Guide

파일명

- 지금 하고 있는 프로젝트의 컨벤션, 안용학교수님의 컨벤션에 따른다.
- 클래스를 설계하는 경우, 파일 앞에 대문자 C 를 붙인이고, 대쉬 (-) 를 붙이고, 대표 class 이름으로 사용하며, C 를 제외하고는 소문자와 언더바를 사용한다. ex : **C-rect_base**
- 테스트용 (실행 파일) 의 경우, Test- 를 가장 처음에 포함한다. ex : **Test-rect_base**
- 파일 이름에 대쉬(-) 를 두 개 이상 사용하지 않는다.

함수명

- 일반적인 함수는 대문자로 시작하며, 각 새로운 단어마다 대문자를 사용한다. 언더라인은 사용하지 않는다. ex : **MyExcitingFunction()**
- 접근자와 수정자(get, set)는 변수 이름과 일치시킨다. ex : **set_my_exciting_member_variable()**
- True/False 값을 return 하는 경우, 함수 이름은 is 혹은 has 로 시작한다. ex : **IsHungry()**
- private 함수 이름은 언더바(_) 로 시작한다. ex : **_DontTouchMe()**

타입명

- 타입명은 대문자로 시작하며, 각 새로운 단어마다 대문자를 갖으며 언더라인을 사용하지 않는다. ex : **MyRectangle**

변수 및 상수명

- 변수명은 소문자로 시작하며, 대문자와 소문자를 섞어서 사용한다.

- 클래스 멤버 변수는 'm_' 으로 시작하며, 간단한 자료형 [string : str, integer : n]을 그 뒤에 표기한 후 이름을 붙인다. ex : **m_strMyExcitingLocalVariable**
- static 멤버 변수의 경우 's_' 으로 시작한다. ex : **s_nMyExcitingStaticVariable**
- 이름은 가능한 설명적으로 짓는다. 공간 절약이 중요한 게 아니라, 코드를 즉시 보고 이해할 수 있어야 한다. ex : **numCompletedConnections**
- 모호한 약어나 의미를 알 수 없는 임의의 문자를 사용하지 않는다. ex : **nerr** (?)
- 구조체의 데이터 멤버는 일반적인 변수처럼 이름을 짓는다. 클래스처럼 언더라인으로 끝나지 않는다.
- 전역 변수는 특별한 요구사항이 없으며, 거의 사용을 하지 않는다. 만약 사용한다면, g_로 시작하거나 로컬 변수와 구별되는 표시를 한다.
- 상수는 k로 시작하며 대소문자를 섞어서 사용한다 : ex : **kDaysInAWeek**

기타

- 들여쓰기는 Tab 을 사용한다.
- 간단한 생성자 초기화는 콜론 초기화로 한다.
- 이항 연산자 (=, >, <, 등..) 앞과 뒤에 공백을 제공한다. ex : **a = b + c**
- 단항 연산자 앞과 뒤에 공백을 제공하나, (A++), [--BB], {--KK}와 같이 사용할 때는 공백이 없어도 좋다.
- 일부 연산자(“ , “, “ ; “)는 연산자 뒤에 공백을 제공해야 한다. ex : **for(i = 0; i < 3; i++)**
- brace({)는 분리된 라인에 작성한다.

```
class People
{
    // 내용
}

void main()
{
    // 내용
}

struct DataStructure
{

}
```

참고한 내용

https://gracefulprograming.tistory.com/11

https://dayday-kim.tistory.com/5

C++ 클래스의 static 멤버 변수의 초기화

기본 인수 재정의 문제 해결

객체생성-두가지-방법과-의문점new를-이용한-동적할당객체