2020 Spring OOP Assignment Report

과제 번호 : 4
학번 : 20190084
이름 : 권민재
Povis ID : mzg00

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. 프로그램 개요

- shared_ptr을 직접 구현해보고, 이를 이용하여 행렬 클래스를 구현한 것이다.
- SmartClass.h 헤더파일을 인클루드하여 다른 cpp 파일에서 이를 이용할 수 있다.
- SmartClass.h에 SmartPtr, SmartArray, SmartMatrix 등의 클래스가 선언되어 있다.

2. 프로그램의 구조 및 알고리즘

- Class[1]

SmartPtr	
shared_ptr 의 구현 클래스	
private	
멤버 변수	
CountedObjectContainer *m_ref_object	SmartPtr 이 레퍼런스하는 컨테이너
public	
멤버 함수	
SmartPtr(ObjectType* object)	오브젝트를 받아서 선언하는 생성자
SmartPtr(const SmartPtr& pointer)	SmartPtr 를 받아서 선언하는 생성자
~SmartPtr()	SmartPtr 소멸자
SmartPtr& operator=(ObjectType *object)	오브젝트 복사 연산자 오버로딩
SmartPtr& operator=(const SmartPtr &ref_pointer)	SmartPtr 복사 연산자 오버로딩

[1] 본인이 수정한 부분들만 표기되어 있다.

SmartMatrix	
SmartPtr 을 이용한 행렬 구현	
private	
멤버 변수	
int m_rows, m_cols	행렬의 행, 열 개수
SmartArray<T> m_values	행렬의 데이터
public	
멤버 함수	
SmartMatrix(int rows, int cols)	행, 열 개수를 받아서 선언하는 생성자
SmartMatrix(const SmartMatrix<T>& mtx)	SmartMatrix 를 받아서 복사 선언하는 생성자
SmartMatrix(int rows, int cols, const T* values)	행, 열 개수, 데이터를 받아서 선언하는 생성자
void AddRow(const T* values)	행을 추가하는 메서드
void AddCol(const T* values)	열을 추가하는 메서드
const SmartMatrix<T> Inverse()	이 행렬의 역행렬을 반환하는 메서드
const SmartMatrix<T> operator+(const SmartMatrix<T>& a, const SmartMatrix<T>& b)	행렬 덧셈 오버로딩
const SmartMatrix<T> operator-(const SmartMatrix<T>& a, const SmartMatrix<T>& b)	행렬 뺄셈 오버로딩
const SmartMatrix<T> operator*(const SmartMatrix<T>& a, T s)	행렬 스칼라 곱 오버로딩
inline const SmartMatrix<T> operator*(const SmartMatrix<T>& a, const SmartMatrix<T>& b)	행렬 곱 오버로딩

□ 알고리즘

■ SmartPtr

◆ SmartPtr 생성자

- 데이터를 받아서 생성되는 경우에는, 해당 데이터를 이용해서 새로운 CountedObjectContainer를 생성한다.
- 기존의 SmartPtr 오브젝트를 복사하여 생성하는 경우에는, 기존 오브젝트의 m_ref_object를 복사하고, 해당 컨테이너의 참조 카운트를 증가시킨다.

◆ SmartPtr 소멸자

- SmartPtr의 컨테이너에서 참조하는 컨테이너의 참조 카운트를 감소시킨

다. decrease_ref_count에서 할당 해제를 관리하므로, 따로 할당 해제를 할 필요는 없다.

- ◆ 대입연산자
  - 오브젝트를 받아서 대입하는 경우에는 기존의 컨테이너의 참조 카운트를 감소시킨 후 새로운 CountedObjectContainer를 제작하여 대입한다.
  - SmartPtr의 오브젝트를 대입하는 경우에는 파라미터로 들어온 SmartPtr의 컨테이너의 참조 카운트를 증가시킨다. 그 후, 기존 컨테이너의 참조 카운트를 감소시키고 파라미터의 컨테이너를 대입한다.

■ SmartMatrix

- ◆ SmartMatrix 생성자
  - 행과 열의 개수만 들어온 상황이라면, 행렬의 행, 열 개수를 설정하고, 새로운 빈 행렬을 할당한다.
  - 다른 SmartMatirix가 파라미터로 전달된 상황이라면, 파라미터의 행, 열 개수를 복사하고, 행렬 값을 딥 카피한다.
  - 행, 열의 개수와 배열이 파라미터로 전달된 상황이라면, 행, 열 개수를 설정하고, 해당 배열로 행렬 값을 설정한다.
- ◆ Row 추가
  - row 개수가 1 큰 임시 배열을 설정하고, 기존 값을 복사한다. 그 후, 파라미터로 들어온 배열을 메모리 카피 한 후, 임시 배열을 행렬의 값으로 설정하여 row를 추가한다.
- ◆ Column 추가
  - column 개수가 1 큰 임시 배열을 설정하고, 기존 값을 복사한다. 그 과정에서, 적절한 위치에 파라미터로 들어온 배열의 값을 삽입한다. 그 후, 임시 배열을 행렬의 값으로 설정하여 column을 추가한다.
- ◆ 역행렬 계산
  - 우선 행렬의 행, 열 개수가 2인지 확인하고, determinant가 0이 아닌지 체크한다. 조건에 부합한다면, 2x2 행렬의 역행렬을 구하는 공식을 이용하여 새로운 행렬을 생성하여 반환한다. 또한, 기존 행렬의 값도 수정한다.
- ◆ 더하기 연산자

- 파라미터로 들어온 SmartMatrix의 행, 열 개수와 현재 행렬의 행, 열 개수가 같은지 체크한다. 같다면, 각 entry들을 더한 결과를 대입한다.

◆ 빼기 연산자

- 파라미터로 들어온 SmartMatrix의 행, 열 개수와 현재 행렬의 행, 열 개수가 같은지 체크한다. 같다면, 각 entry들을 뺀 결과를 대입한다.

◆ 곱하기 연산자

- 파라미터로 행렬이 담고 있는 데이터형이 들어온다면, 일종의 "스칼라 곱"을 수행한다. 행렬의 모든 엔트리에 파라미터로 들어온 변수를 곱한다.
- 파라미터로 다른 SmartMatrix가 들어온다면, 새로운 행렬을 생성하고, 새로운 행렬의 entry에 앞 행렬의 row와 뒤 행렬의 col을 내적 한 값을 대입하고, 새로운 행렬을 반환한다.

3. 토론 및 개선

□ C++의 shared_ptr를 이용하면 참조 카운팅을 통해 일종의 가비지 콜렉팅을 할 수 있다는 사실을 알았다. 하지만, 디버깅을 하는 과정에서 참조 카운팅을 이용한 가비지 콜렉팅이 프로그램의 실행 속도를 늦출 수 있음을 알았다.

□ 템플릿 클래스를 이용하면 여러 자료형에 대해 쓸 수 있기에, 코드의 재사용성을 높일 수 있음을 알 수 있었다.

□ 대입 연산자를 오버로딩 할 때, 딥 카피를 할 필요가 없음에도 딥 카피를 수행하여 성능을 하락시키고 컨테이너의 존재 의미를 무력하게 만든 것이 아쉬웠다.

4. 참고 문헌

□