**2020 Spring OOP Assignment Report**

과제 번호 : 5

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**
   * 이 프로그램은 C++와 QT Framework를 이용해 간단한 시계 어플리케이션을 구현한 것이다.
   * 빌드된 파일을 실행하고, 버튼을 눌러서 프로그램을 이용할 수 있다.
   * SmartClass.h에 SmartPtr, SmartArray, SmartMatrix 등의 클래스가 선언되어 있다.
2. **프로그램의 구조 및 알고리즘**

* Class[[1]](#footnote-1)

|  |  |
| --- | --- |
| **SmartPtr** | |
| shared\_ptr의 구현 클래스 | |
| private | |
| 멤버 변수 | |
| CountedObjectContainer \*m\_ref\_object | SmartPtr이 레퍼런스하는 컨테이너 |
| public | |
| 멤버 함수 | |
| SmartPtr(ObjectType\* object) | 오브젝트를 받아서 선언하는 생성자 |
| SmartPtr(const SmartPtr& pointer) | SmartPtr를 받아서 선언하는 생성자 |
| ~SmartPtr() | SmartPtr 소멸자 |
| SmartPtr& operator=(ObjectType \*object) | 오브젝트 복사 연산자 오버로딩 |
| SmartPtr& operator=(const SmartPtr &ref\_pointer) | SmartPtr 복사 연산자 오버로딩 |

|  |  |
| --- | --- |
| **SmartMatrix** | |
| SmartPtr을 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) | SmartMatrix를 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
  + Watch mode
  + Stopwatch mode
  + Timer mode

1. **토론 및 개선**
   * + Qt를 이용하여 c++에서 라이브러리를 이용하는 방법에 대해 알아보고, event-driven 방식으로 프로그래밍 할 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.
2. **참고 문헌**

1. 중점적인 부분만 기록되어 있다. [↑](#footnote-ref-1)