**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
    - Watch mode에서, event\_clock이 s\_timer에 의해 1초마다 실행되게 된다. 실행되었을 경우에, event\_clock은 현재 시간을 가져오고, 시간과 날짜를 라벨에 출력한다. 이를 통해 1초에 한번씩 시간과 날짜를 출력함으로써 시계의 기능을 수행한다.
  + Stopwatch mode
    - Stopwatch mode에서는 event\_stopwatch가 m\_timer에 의해 10 밀리초마다 실행되게 된다.
    - 실행되었을 경우, event\_stopwatch는 우선 스탑워치가 실행중인지 그 여부를 우선 체크한다. 만약 실행 중인 상태가 아니라면 즉시 종료하고, 아니라면 이후 과정을 수행한다.
    - 스톱워치는 QTime 오브젝트 stopwatch\_time으로 시간을 잰다. 만약, 현재까지 잰 stopwatch\_time의 밀리초가 60 \* 60000 밀리초, 즉 1시간이 되었다면 실행을 중지하고 초기화시킨다. 현재까지 잰 시간은 msecsTo 메서드를 통해 알 수 있다.
    - 아니라면, stopwatch\_time을 10 밀리초 증가시키고, 라벨에 표시하여 스톱워치를 실행시킨다.
    - 시작 버튼을 눌렀을 경우, 시작 여부를 표시해주고, 현재 문자열에서 fromString 메서드를 이용해 시간을 추출하여 stopwatch\_time을 그 시간으로 초기화한다.
    - 중지 버튼을 눌렀을 경우, 시작 여부를 false로 마킹해준다.
    - 리셋 버튼을 눌렀을 경우, stopwatch\_time을 0밀리초로 초기화하고, 시작 여부를 false로 마킹하고, 표시되는 라벨 또한 초기화시킨다.
  + Timer mode
    - Timer mode에서는 event\_timer가 s\_timer에 의해 1초에 한번씩 실행되게 된다.
    - 실행되었을 경우, event\_timer는 우선 타이머가 실행중인지 그 여부를 체크하여 실행중인 상태가 아니라면 즉시 event\_timer를 즉시 종료한다.
    - 타이머는 QTime 오브젝트, timer\_time으로 시간을 측정한다.
    - 만약 timer\_time의 누적 시간이 0초라면, timer\_done을 실행시키고 종료한다. 타이머가 종료되지 않은 상태라면, event\_timer에서 timer\_time을 1초 감소시키고, 그 시간으로 라벨을 업데이트 하여 타이머의 역할을 수행한다.
    - timer\_done에서는 효과음을 재생시키고, 중지 버튼을 눌렀을 때 실행되는 함수와 리셋 버튼을 눌렀을 때 실행하는 함수를 호출하여 타이머를 중지 및 리셋한다.
    - 각 항목의 증가/감소 버튼을 눌렀을 경우, 함수 timer\_label\_update가 각 최소/최댓값에 맞는 문자열을 반환하고, 이 문자열로 라벨을 업데이트 한다.

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

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