

# Mini Excel

Due Date: ~11/24(Tue), 23:59

이번 과제는 지금까지 OOP 수업시간에 배운 내용을 동원하여 간단한 스프레드시트 프로그램인 Mini Excel을 작성하는 것을 목표로 한다. 과제는 크게 두 단계로 나뉘어지며, 첫째 단계에서 **Mini Excel 엔진**을 구현한 후, 둘째 단계에서 이를 바탕으로 **Qt GUI 프레임워크를 적용**하는 것을 최종 목표로 한다.

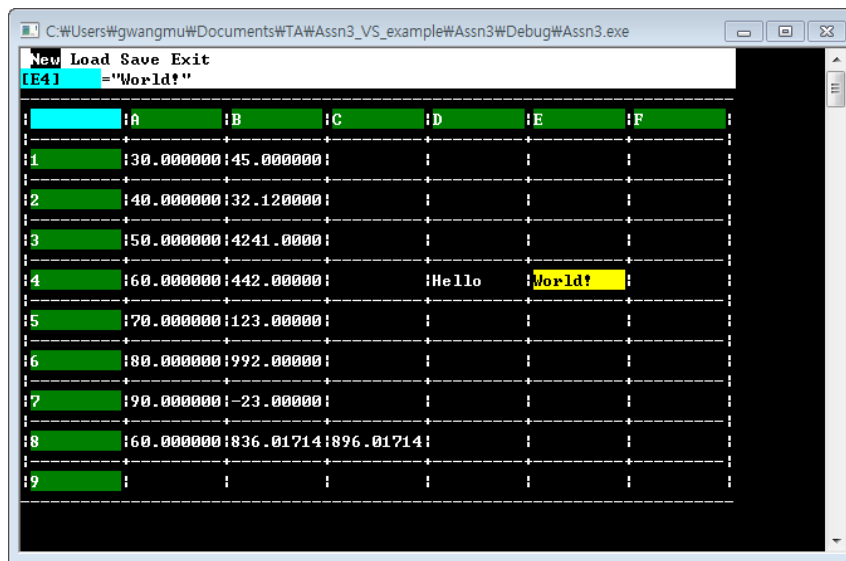


그림 1. Skeleton 파일을 이용한 Mini Excel 작성 모습의 예

## 1. 배경 설명

### 1.1. TSV 포맷

100 <Tab> 200 <Tab> 300
100 <Tab> 200 <Tab> 300

그림 2. TSV 포맷의 예제.

TSV 포맷은 텍스트 편집기로 작성이 가능한 직관적인 스프레드시트 포맷으로, 탭 문자로 열을 구분하고, 개행 문자로 다음 행으로 넘어감을 표시한다. 그림 2는 TSV 포맷으로 작성된 간단한 스프레드시트 파일으로, 이 그림에서 같은 숫자(100, 200, 300)끼리 같은 열에 위치해 있으며, 한 행에 100/200/300이 차례대로 열을 차지하도록 작성되어 있다. TSV 포맷은 행/열에 대한 별다른 제약사항이 없으며, 행의 길이가 달라도 적합한 TSV 파일로 취급된다. 그림 3은 행 길이가 다른 TSV 파일의 예제를 보여준다.

```
This <Tab> is <Tab> perfectly <Tab> legal.
100 <Tab> 200 <Tab> 300
```

그림 3. 행 길이가 다른 TSV 포맷의 예제.

TSV 확장자의 파일은 Microsoft Excel 2013에서 바로 읽을 순 없으나, 수동으로 불러온 후 탭으로 구분시키는 과정을 거쳐 간접적으로 데이터를 확인할 수 있다.

## 1.2. 확장 TSV 포맷

확장 TSV 포맷은 Assignment 3을 위해 특수하게 확장된 TSV 포맷이다. 확장 TSV 포맷은 기존의 TSV 포맷이 지원하는 숫자 및 문자 데이터에 덧붙여 **i) 다른 셀을 참조(Reference)하는 값, ii) 총합, 평균과 같은 함수**를 추가로 지원하도록 설계되어 있어, 텍스트 편집기를 통해 TSV를 작성한 후 Excel과 같은 스프레드시트 프로그램에서 불러와 계산 값을 확인할 수 있다.

```
30 <Tab> 40 <Tab> 50 <Tab> =SUM(A1:C1)
-20 <Tab> 100 <Tab> 300 <Tab> =AVERAGE(A2:C2)
```

그림 4. 행 길이가 다른 TSV 포맷의 예제.

TSV에서 확장한 문법의 대부분은 일반적인 스프레드시트 프로그램에서 사용되는 문법과 거의 유사하다. 가령 확장 TSV 포맷으로 작성된 예제(그림 3)에서 D1(넷째 열 첫째 행) 셀은 A1부터 C1까지의 합을 계산하며, D2 셀은 A2부터 C2까지의 평균을 계산한다. 일반적으로 적합한 확장 TSV 포맷으로 작성된 파일은 Microsoft Excel 2013에서 탭으로 분리를 가한 후 읽었을 때 결과를 확인할 수 있다.<sup>1</sup>

## 1.3. 확장 TSV 포맷 문법

아래는 확장 TSV 포맷에서 셀 데이터를 구성하는 값(요소)들을 정리한 것이다.

구분	예제	조건	계산(Evaluation) 결과
숫자	0, -12, 34.123	일반적인 정수 혹은 실수. 맨 처음에 '-' 하나를 포함할 수 있고, 소수점을 최대 하나까지 포함할 수 있다.	해당 숫자
문자열	"String Example"	처음과 끝이 "(쌍따옴표)"로 감싸져 있다.	해당 문자열
레퍼런스	A2, D8, B11	명시된 셀 번호에 대한 참조.	참조하는 셀의 계산 결과
범위	A2:A15	레퍼런스 범위	(계산 불가)
함수	ADD(10,A8)	인수를 받아 특정 기능을 하는 함수. 함수 이름 다음에 인수 목록이 괄호로 감싸여 있으며, 인수들은 쉼표로 구분된다.	해당 함수의 계산 결과

<sup>1</sup> 단 후에 설명할 ADD, MUL 함수를 사용한 경우엔 Microsoft Excel 2013에 해당 함수가 없어 올바른 결과가 출력되지 않을 수 있다.

		인수로 들어올 수 있는 값의 종류는 함수마다 상이하다.	
식	=10, ="asdf"	계산(Evaluation)이 필요한 값들에 대해 계산을 수행할 것을 명령. 등호('=')로 시작.	등호 다음 값의 계산 결과.

아래는 적합한 확장 TSV 포맷에서 요구하는 추가 조건을 정리한 것이다:

- 숫자/를 제외한 나머지 값 종류들은 반드시 식을 동원하여 사용되어야 한다.

ex) 셀의 데이터로 10을 저장 → 숫자 값이므로 적법

셀의 데이터로 "adfg"를 저장 → 문자열 값이므로 부적격. ="adfg"와 같이 식을 써야 함.

#### 1.4. Tiny UI Framework Ver.2

Tiny UI Framework Ver.2는 Assignment 2의 Tiny UI Framework의 요구 스펙을 기준으로 확장된 터미널 환경 GUI 프레임워크이다. *TextBox*, *ArrayView*와 같은 추가 *UIObject*를 수록하고 있는 것 외에도, *UIManager* 클래스가 *UIObject*에 이벤트가 발생했을 때 사용자가 미리 지정한 이벤트 처리 함수(즉, Callback 함수)를 부르는 방식으로 작동하도록 설계되었다는 점이 주요 차이점이다.<sup>2</sup> 가령 사용자가 *TextBox*의 글자가 바뀔 때마다 불릴 함수로 *onTextChanged* 라는 함수를 등록시켜놓았다면, 해당 *TextBox*의 글자가 바뀔 때마다 *onTextChanged* 함수가 호출되는 식이다.

이번 Assignment에서는 Tiny UI Framework 자체를 구현하는 대신, 완성본 형태로 제공되는 프레임워크를 Mini Excel을 구현하는데 사용할 것이다.

## 2. Tiny UI Framework Skeleton

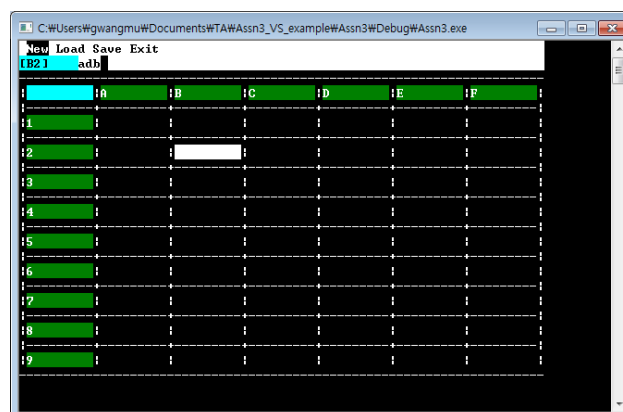


그림 5. 본 Assignment에서 제공하는 Skeleton의 스크린 샷

<sup>2</sup> 이를 일반적으로 **Event-driven** 방식이라고 한다.

이번 Assignment에서는 Mini Excel 엔진부 구현을 위해 Tiny UI Framework Ver.2 를 이용해 작성된 Skeleton 파일을 제공한다. 그림 5 는 아무런 수정을 가하지 않은 Skeleton 코드를 바로 컴파일하여 실행한 스크린 샷이다. Skeleton 에 이미 UI 관련 부분의 세팅이 완료된 상태이므로, Skeleton 의 외형은 엔진 완성 전후에 특별하게 차이 나지 않는다.

## 2.1. Skeleton 코드 런타임 동작

그림 5 는 Skeleton 코드를 실행한 결과이며, 프로그램 종료 전까지 이와 거의 비슷한 외형을 갖는다. 화면은 상단의 **MenuStrip** 과 그 아래의 **ColorLabel**, **셀 편집 TextBox**, 그리고 중앙에 TSV 파일을 표시할 **ArrayView** 로 구성되어 있다. 화면의 UIObject 는 ColorLabel 을 제외하고 모두 포커스를 가질 수 있는데, 포커스는 Tab 키를 눌러 이동할 수 있으며, 키보드 입력은 포커스를 가진 UIObject 에게로 전달된다. 포커스를 가진 UIObject 는 각자 방식으로 포커스가 있음을 표시하는데, MenuStrip 의 경우 맨 첫 칸이 파란색으로 점등되며, TextBox 의 경우 커서가 나타난다. ArrayView 의 경우 현재 선택된 셀이 노란색으로 바뀌며, 방향키를 이용해 상하좌우로 이동하거나 보이는 영역을 이동시킬 수 있다.

## 2.2. Event-driven 실행 방식

Tiny UI Framework Ver.2 로 작성된 Skeleton 코드는 Event 가 발생할 경우 미리 유저(프로그래머)가 등록한 함수를 호출시키는 Event-driven 방식으로 동작한다. 가령 TextBox 에 포커스된 상황에서 셀에 값을 등록하기 위해 Entry 키를 누른 경우, 미리 이러한 Event 에 등록된 함수가 자동으로 호출되는 방식으로 작동한다. 참고로 이렇게 Event 에 의해 자동으로 호출되는 함수를 Callback(콜백) 함수라고 명명한다.

Skeleton 코드에서는 Mini Excel 을 구현하는 데 필요한 콜백 함수가 미리 등록된 상태이지만, 이들에 대한 구현은 없는 상태로 남아있다. 아래는 미리 등록된 콜백 함수이 호출되는 조건과, 이들의 후보 동작을 정리한 것이다.

onFocusChangedAtArrayView ()	TSV 를 표시하는 ArrayView 의 포커스 셀이 변경되었을 때 호출되는 콜백 함수. 변경된 포커스에 맞게 ColorLabel 의 셀 번호를 변경해주어야 하며, 셀 편집 TextBox 의 내용을 알맞게 변경해야 한다.
onEntryKeyPressedAtTextBox ()	셀 편집을 위한 TextBox 에서 Enter 키가 눌렸을 때 호출되는 콜백 함수. ArrayView 에서 포커스 된 셀의 값을 변경하고, 해당 셀의 값 변경으로 인해 변경되는 다른 셀의 값들도 갱신해준다.
onSelectedAtMenuStrip ()	MenuStrip 에서 선택(Enter) 키를 눌렀을 때 호출되는 콜백 함수. 포커스가 New 인 경우 작업 내용이 모두 초기화되며, Load 인 경우 실행 파일이 있는 디렉토리의 'input.tsv'를 불러온다. Save 인 경우 'output.tsv'에 작업 내용을 확장 TSV 포맷으로 저장한다.

Qt 를 통한 GUI 프로그램 작성이 목표인 만큼 콜백 함수 및 Skeleton 코드의 동작은 위에서 명시된 것과 차이점이 있어도 무관하지만, 만약 Qt 가 아닌 Skeleton 상태의 코드를 제출할 경우, 위에 명시된 사항대로 Skeleton 코드가 동작하여야 한다.

### 3. 문제 설명

#### 3.1. Mini Excel 엔진 구현 (배점 비율 80%)

엔진 구현을 위해 작성해야 할 코드를 크게 네 개로 구분지을 수 있다.

- 유틸리티 (CUtil)

<namespace> CUtil
+ isNumber () + convStringToDouble () + convDoubleToString () + convColumnIndexToString () + convColumnStringToIndex () + trimString ()

그림 6. CUtil 네임스페이스의 구성

유틸리티는 향후 Mini Excel 구현을 위해 필요한 일반적인 함수들을 편의상 모아놓은 코드 부분으로, 모두 CUtil 네임스페이스에 포함된다. 유틸리티 함수들은 임의의 코드 부분에서 상시 사용될 수 있기 때문에, 클래스로 선언하지 않고 네임스페이스로 묶어 정리만 해두도록 한다. 아래는 각 함수들의 용도를 정리한 것이다:

함수 이름	용도
isNumber	특정 string이 숫자인지 판별하는 함수.
convStringToDouble	string을 double로 변환하는 함수.
convDoubleToString	double을 string으로 변환하는 함수
convColumnStringToIndex	열 문자(A, B, C..)를 숫자(1, 2, 3..) 으로 바꿔주는 함수. 열 문자는 최대 Z까지만 존재한다고 가정.
convColumnIndexToString	위 함수의 역함수
trimString	특정 string 전후의 공백(스페이스, 탭, 개행..)을 삭제.

참고로 이번 Assignment에서는 편의상 숫자 값을 모두 double형 변수로 처리한다고 하자.

- 셀 관리 (CSheet, CCell)

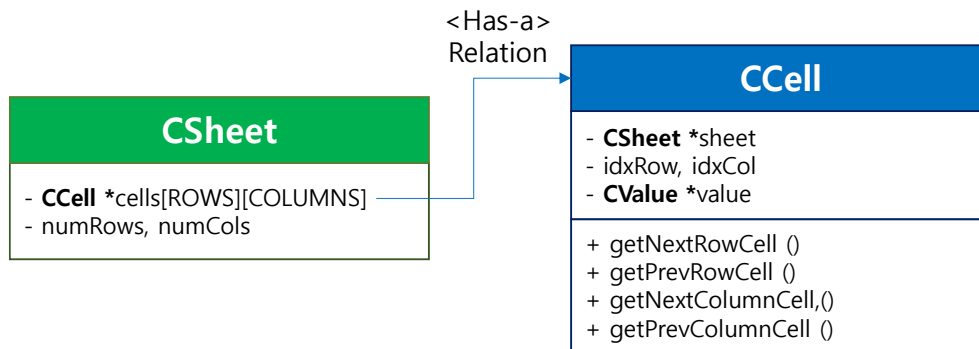


그림 7. 셀 관리 부분의 클래스 구조

셀 관리 부분은 전체 TSV 파일의 판과 셀 자체에 대한 관리를 담당하는 부분이다. 이 부분에 속하는 클래스는 CSheet, CCell 두 개로, 각각 TSV 파일의 판(Excel에서의 Sheet와 유사)과 셀 하나를 대표한다. 그림에서 클래스의 Getter 및 Setter 함수는 생략되었다. 아래는 주요 멤버의 역할을 정리한 것이다:

멤버 이름	용도
CCell::sheet	CCell이 속한 CSheet를 가리킴
CCell::value	해당 셀이 소유한 값. CValue는 다음 부분 설명에서 언급.
CCell::getNextRowCell ()	해당 셀 다음 행에 있는 셀을 반환.

CSheet 크기에는 10 x 10 이상의 상한이 존재할 수 있다. 즉 그림 7과 같이 cells 필드를 배열로 선언할 경우, 고정 크기로 ROWS와 COLUMNS를 각각 10 이상으로 줄 수 있다.

- 값 부분 (CValue, CConstant, CFunction...)

값 부분은 Mini Excel 엔진 부분 가운데 가장 클래스 구성 규모가 큰 부분으로, 셀 내에 저장되는 값을 체계적으로 관리하고 계산하기 위해 설계된 부분이다. 구성의 대부분은 1.3 장에서 설명된 확장 TSV 포맷의 값 종류 정의를 거의 그대로 차용하고 있다.

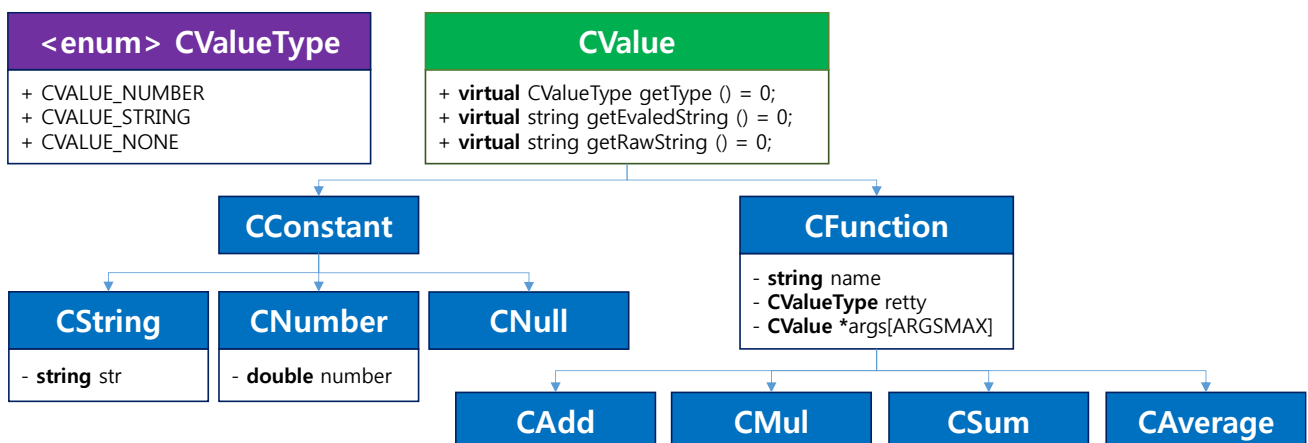


그림 8. 값 관련 부분의 클래스 구성도 (1)

그림 8은 값 부분의 최상위 클래스인 CValue 클래스를 상속하는 CConstant와 CFunction, 그리고 이를 상속하는 하부 클래스의 구조를 나타낸다. CValue는 Mini Excel 엔진이 다른 값을 대표하는 Base 클래스로서, 그 자체는 멤버 변수를 갖지 않고 추상 가상함수만을 갖는 순수 추상함수이다. CValue를 상속받는 클래스들은 이들에 대한 구현 제공의 의무를 갖게 된다. 아래는 이들 추상 가상함수의 역할을 정리한 것이다:

함수 이름	용도
getType	값의 타입을 리턴. CValueType에서 정의된 바와 같이, 숫자인 경우 CVALUE_NUMBER, 문자열이면 CVALUE_STRING, 계산할 수 없는 타입이면 CVALUE_NONE을 리턴한다.
getEvaldString	해당 값을 계산(Evaluate)한 결과를 string으로 리턴.
getRawString	해당 값에 해당하는 원본 표현을 string을 리턴.

CConstant 클래스는 문자열(CString), 숫자(CNumber), 공백 셀(CNull)들의 상위 클래스로, 값들이 다른 값에 의존적이지 않고 스스로 정의된 상수들의 상위 클래스로 역할을 한다. CConstant는 특별한 멤버가 필요 없는 공백 클래스(Empty Class)이며, 단지 상수 역할을 할 클래스들에 대한 Base 클래스 역할만을 한다.

CFunction 클래스는 함수 값에 대한 일반적인 멤버를 정의한다. name, retty, args 멤버는 각각 함수의 이름과 리턴 결과의 타입, 인수들을 저장하는 필드이다. CFunction을 상속받는 함수 클래스들은 기능에 따라 받는 인수나 CValue의 종류가 다를 수 있으며, 이들은 생성자를 통해 지정될 수 있다. 각 함수 클래스들의 리턴 타입과 인수 명세는 아래와 같다:

함수 클래스	기능	리턴 타입	인수
CAdd	더하기	CVALUE_NUMBER	(CValue, CValue)
CMul	곱하기	CVALUE_NUMBER	(CValue, CValue)
CSum	범위 총합	CVALUE_NUMBER	(CRefRange)
CAverage	범위 평균	CVALUE_NUMBER	(CRefRange)

이번 Assignment에서는 함수의 종류를 늘리는 방식으로 추가 구현을 해보는 것도 좋은 확장이 될 것이다. 가령 CVALUE\_STRING 타입의 리턴 값을 갖는 함수가 없으니, 이를 고려하여 CONCATENATE(문자열 이어붙이기) 함수를 추가해볼 수도 있겠다.

그림 9(다음 장)는 기타 값 클래스의 구조를 나타낸 것이다. CExpression, CReference, CRefRange는 각각 1.3장에서 소개된 값 종류 가운데 식, 레퍼런스, 범위를 나타낸다. CError는 셀에 입력된 데이터(문자열)을 특정 값 클래스로 만들어낼 수 없을 때 생성되는 클래스로, 사용자가 요청한 데이터 문자열을 rawstr 필드에 저장하고 있다. CError는 원천적으로 계산될 수 없으므로, getEvaldString 함수에서 Exception을 throw한다.

참고로 CError 클래스 외의 다른 클래스 역시 getEvaedString 함수에서 Exception을 throw할 수 있는데, 가령 CAverage와 같은 함수 클래스에서 계산할 수 없는 범위가 들어온다든지 하는 경우가 여기에 해당한다. 만약 getEvaedString이 예외를 throw하는 경우에는, "ERROR!"와 같이 적절한 에러 문자열을 셀에 출력해야 한다.

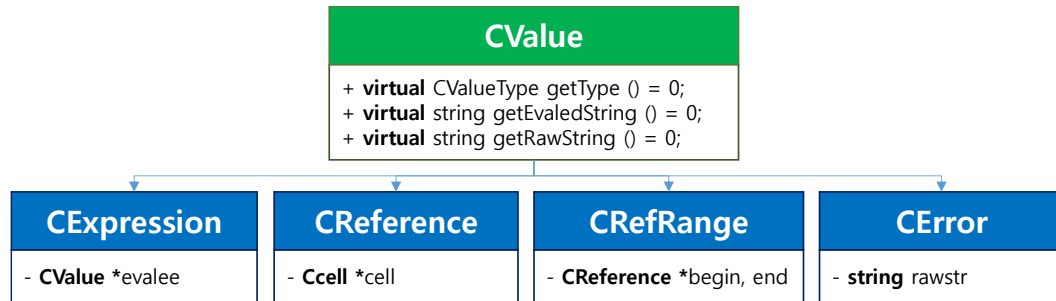


그림 9. 값 관련 부분의 클래스 구성도 (2)

#### - 값 생성 부분 (CValueFactory)

값 생성 부분은 사용자에게서 입력받은 문자열을 이용해 CValue 클래스를 생성하는 부분이다. 문장 분석(Lexical Analysis) 알고리즘을 요하므로 Mini Excel 엔진 내에서 이상 작동을 일으킬 가능성이 가장 높은 부분이지만, 확장 TSV의 간단한 문법과 Exception 핸들링(try-catch-throw)을 이용하면 비교적 어렵지 않게 구현이 가능하다.

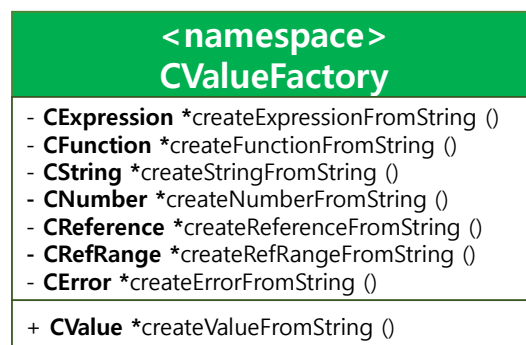


그림 10. CValueFactory 네임스페이스의 구조

그림 10은 값 생성을 담당하는 네임스페이스인 CValueFactory의 구조를 나타낸 것이다. 그림에서 +로 표시된 함수는 소스 바깥에서도 호출할 수 있는 전역(Global) 함수이며, -로 표시된 함수는 CValueFactory 소스 안에서만 부를 수 있는 정적(Static) 함수들이다. 유일한 전역 함수인 createValueFromString 함수는 Lexical Analysis의 진입점으로, 이 함수는 일단 전달된 문자열 데이터의 큰 특성을 파악하여 알맞은 함수를 호출해 값 클래스를 생성시킨다. 가령 첫 글자가 '='로 시작하면 createExpressionFromString을 호출하여 식 값을 생성시키는 식이다.

createValueFromString은 소스 외부 뿐만 아니라, 필요한 경우 CValueFactory 소스 내부에서도 호출할 수 있다. 가령 함수의 경우 인수들을 값으로 생성시킬 때 다시 이 함수를 이용한다면 편리할 것이다.



### ● Tiny UI Framework Skeleton 런타임 동작 (추가)

엔진 구현 후에 Skeleton 소스에 추가된 동작을 묘사하면 다음과 같다. 일단 `ArrayView`에서 셀을 포커스 하면 `TextBox`의 내용을 셀의 값으로 갱신시켜야 하는데, 이 때 `TextBox`는 셀의 내용을 편집하는데 이용되는 것이므로, 여기에 갱신되는 값은 `getRowString()` 함수가 리턴하는 원본 표현이어야 한다. 한편 `ArrayView`의 셀 내용에는 계산(Evaluation)된 셀의 값이 들어가야 하므로 `getEvalString()`의 값이 들어가야 한다.

### 3.2. Qt를 이용한 Mini Excel 포팅 (배점 비율 20%)

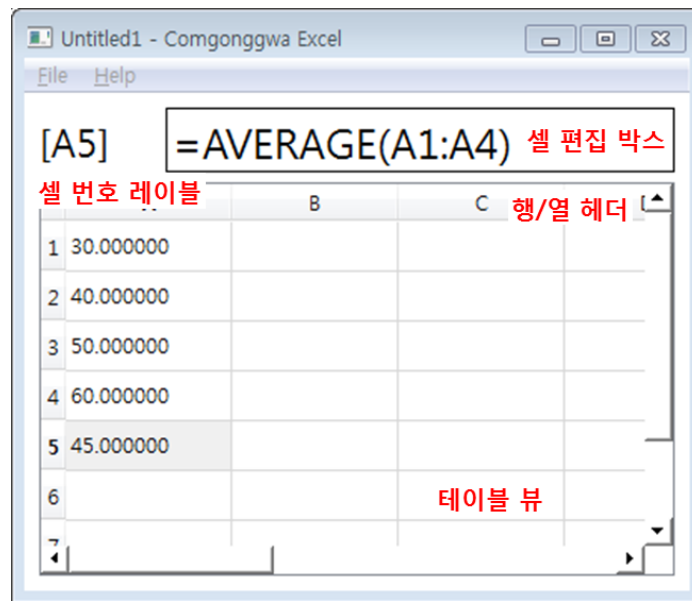


그림 11. Qt로 포팅된 Mini Excel의 예제

Tiny UI Framework Skeleton을 이용하여 구현한 Mini Excel 엔진에 Qt를 적용하여 GUI 프로그램을 작성한다. 그림 11은 Qt로 포팅된 Mini Excel 프로그램의 예제를 나타낸다.

Qt로 포팅된 결과 프로그램은 기본적으로 Skeleton에서 수행할 수 있었던 동작이 모두 가능해야 한다. 아래는 Qt 포팅 프로그램이 UI 측면에서 갖추어야 하는 기능을 정리한 것이다:

- ✓ New, Load, Save, Exit에 해당하는 기능.
  - New: Sheet를 초기화시키고, 기타 UI 구성요소(테이블 뷰, 셀 편집 박스, 셀 번호 레이블)들 역시 초기화.
  - Load: 실행 파일의 경로에 있는 'input.tsv' 파일을 읽어온 후 테이블 뷰에 반영.
  - Save: 실행 파일 경로에 'output.tsv'로 작업한 Sheet 내용을 저장.
  - Exit: 프로그램 종료
- ✓ 그림 11의 셀 번호 레이블, 셀 편집 박스, 테이블 뷰에 해당하는 UI 요소의 존재.
- ✓ 테이블 뷰에서 테이블 맨 좌측과 위에 각각 행 번호와 열 번호를 나타내는 헤더 표시.

- ✓ 테이블 뷰에서 셀을 클릭하였을 때 셀 번호 레이블 및 셀 편집 박스의 기능을 갱신하는 기능.
- ✓ 셀 편집 박스에서 엔터 키를 눌렀을 때 Sheet에 내용을 반영하고, 테이블 뷰를 갱신하는 기능.

#### 4. 알림 사항

- 본 Assignment와 관련한 Qt 사용 소개는 Intro Class 및 Intro Class 수업 자료를 참조.
- 이번 과제에서는 최종적으로 Qt Framework를 이용한 Mini Excel 프로그램의 소스 및 Qt 프로젝트 전체를 제출해야 한다. Qt 프로젝트는 Visual Studio 2010 Plug-in을 이용한 작성이 권장되지만, 보고서에 명시하는 조건으로 Qt Creator로의 작성 역시 가능하다.
- Qt로의 포팅을 성공하지 못한 경우 **Qt 포팅 점수 비율 20%의 점수 패널티를 적용하여 Tiny UI Framework Skeleton으로 작성한 프로그램 제출을 허용**한다.
- 최종 제출물이 Qt 프로그램인 점을 감안하여 Tiny UI Framework Skeleton을 이용하지 않고 곧바로 Qt를 이용하는 것도 가능하지만, Qt 사용이 능숙하지 않을 경우 먼저 Skeleton을 이용해 엔진 작업을 하는 것을 권장.
- 부득이하게 Tiny UI Framework Skeleton을 최종 제출하게 될 경우 Linux 서버에서 구동이 되어야 하며, Visual Studio의 Skeleton을 제출한 경우 소량의 감점이 적용된다.
- 전체 Assignment는 STL를 사용하지 않고도 작성할 수 있도록 설계되어 있으나, 보고서에 명시하는 조건 하에 STL 라이브러리를 사용할 수 있다. 배열을 사용할 경우 배열 원소가 동적으로 변하는 경우를 고려하지 않아도 좋으며, 적당한 크기의 정적 배열을 잡는 것으로 충분하다.
- 추가 점수를 위한 추가 구현은 적극 권장되나, 추가 구현의 결과는 문서에 명시된 기능을 논리적으로 '포함'해야 하며, 구현을 더 간단하게 하는 방향으로 작용하여서는 안 된다.
- Tab 문자가 열 구분에 이용되는 만큼, 셀의 값으로 Tab 문자는 입력되지 않는다고 가정한다.  
(Tab 문자가 입력될 때를 대비하여 적당한 예외 처리 방법을 갖춘 경우 추가 기능으로 인정.)
- 추가로 구현한 함수의 경우 해당 함수의 기능을 보고서에 명시해야 한다. 참고로 추가 함수가 반드시 Microsoft Excel에 존재하는 함수일 필요는 없다.
- 제시된 클래스 구성도에서 세부 멤버의 경우 추가/수정이 가능하다. 단 수정의 경우, 수정 후의 구조 및 기능이 제시된 것과 타당한 선에서 합치되어야 한다.
- 클래스의 상속 구조는 제시된 클래스 구조도를 따라야 한다.
- 구성원들의 모든 멤버 변수는 'private'으로 관리되어야 하며, 이들 멤버 변수에 대한 접근은 모두 'public' 멤버 변수를 통해 이루어져야 한다.
- 보고서에는 작성한 클래스의 멤버 변수 및 함수에 대한 설명을 포함하여야 한다.
- 보고서에 위의 요구 조건을 확인할 수 있는 시나리오와 결과를 캡처하여 첨부해야 한다.
- 보고서에 위 요구사항 중 만족한 것과 만족하지 못한 것을 정확하게 명시하여야 하며, 정확하게 명시하지 않을 경우 '프로그램 설계 및 구현', '보고서 구성 및 내용, 양식' 점수가 감점될 수 있다.
- 채점 기준은 AssnReadMe.pdf 파일을 참고한다.