클래스+연산자오버로딩 실습

- 뒤의 4개의 과제를 구현하여 4개의 폴더에서 각각 구현하여 제출
  - \Amount : main.cpp Amount.cpp Amount.h
  - \Fraction : Fraction.cpp Fraction.h (지난 피드백 반영하여 수정할 것)
  - \Matrix : Matrix.cpp Matrix.h (지난 피드백 반영하여 수정할 것)
  - \DDay : 적절히 파일 분할하여 *.cpp를 컴파일하면 동작하도록 구성
    - 주된 클래스는 DDay.cpp DDay.h로 구성할 것
    - Dday 과제는 다다음주까지 연장제출 가능. 단 다음 주 제출에 비해 최대70% 점수 부여
  - 필요하다면 common.h와 common.cpp (또는 utils.h utils.cpp)등의 구현을 권장
- 다음 내용을 주요 채점 요소로 함
  - 코딩 스타일 지속적으로 채점
  - Class 설계 적절성을 매우 중요한 기준으로 점검
    - 함수에서의 적절한 const 정의 여부도 매우 중요
  - DDay의 경우 main의 적절한 구조도 고민할 것

실습 1

- dollar와 cent에 대한 class Money 에 대해 아래 연산자들을 구현
  - binary operator +, -, ==, !=, >=, <=, >, > 와 적절한 생성자
  - 음수 금액에 대해서는 처리하지 않음

- 아래와 같이 동작하는 stuff 구현

```
Input dollars and cents : 30 120
Input amount is $31.20
Input dollars and cents : 8 90
Input amount is $8.90
$31.20 + $8.90 = 40.10
$31.20 - $8.90 = 22.30
$31.20 == $8.90 is false.
$31.20 >= $8.90 is true.
$31.20 <= $8.90 is false.
$31.20 > $8.90 is true.
$31.20 < $8.90 is false.
Input dollars and cents : 0 0
Stop with 0 0
=== END ===
```

```
Money m1, m2, END_AMOUNT(0, 0);

while (m1.inputNreturn() != END_AMOUNT)
{
      m2.inputNreturn(); // $0.00 could be input

      cout << m1.toString() << " + " << m2.toString()
           << " = " << (m1 + m2).toString();

      cout << m1.toString() << " - " << m2.toString()
           << " = " << (m1 - m2).toString();

      cout << m1.toString() << " == " << m2.toString() << " is ";
      if (m1 == m2) cout << "true" << endl;
      else cout << "false." << endl;
      ...
}
cout << "Stop with 0 0" << endl << " === END === " << endl;
```

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

실습 2, 3

- 분수와 행렬 프로그램을 아래 main으로 동작하게 변경하시오.
  - 아래 main에 포함되어 있는 함수만 포함한 별도 main으로 테스트 예정

```
#include <iostream>
#include "Fraction.h"
using namespace std;

int main()
{
    Fraction f1(2,3), f2(-2, 5), f3 ;
    f3 = f1+f2 ;
    cout << f1 << " + " << f2 ;
    cout << " = " << f3 << endl ;
    return 0 ;
}
```

```
2/3 + -2/5 = 4/15
```

```
#include <iostream>
#include "Matrix.h"
using namespace std;

int main()
{
    Matrix m1, m2 ;
    cout << "Input : " ;
    cin >> m1 ; cout >> m1 ;
    cout << "Input : " ;
    cin >> m2 ; cout >> m2 ;

    cout << "Add " << endl ;
    cout<< m1 + m2 << endl ;

    cout << "Multiply " << endl ;
    cout<< m1 * m2 << endl ;
    return 0 ;
}
```

```
Input: 1 2 3 4 5 6 7 8 9
¦  1   2   3 ¦
¦  4   5   6 ¦
¦  7   8   9 ¦

Input: 1 -1 0 0 -1 1 -1 1 0
¦  1  -1   0 ¦
¦  0  -1   1 ¦
¦ -1   1   0 ¦

Add
¦  2   1   3 ¦
¦  4   4   7 ¦
¦  6   9   9 ¦

Multiply
¦ -2   0   2 ¦
¦ -2  -3   5 ¦
¦ -2  -6   8 ¦
```

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

실습 4 – D-day 계산

- class Day: 년/월/일 정보를 저장(default 생성자는 시스템 오늘 날짜)
- operator overloading을 구현
  - 단항 전위 연산자 ++와 --, 산술연산자 +와 -, 입출력 <<와 >>
    - Day d1, d2 ; d2 = d1 + 100과 같은 형태
      - +와 -의 첫 번째 인자는 Day instance, 두 번째 인자는 정수. Day 형instance.
      - + 100을 ++를 100번 수행하는 등의 비효율적인 방식으로 구현하면 부분 감점 큼

- 실행 예제와 같이 돌아가게 구성.
  - 날짜를 직접 쓰거나, 전날이나 다음날로 이동하면 현재 날짜가 변경
    - 입력되는 글자수는 최대 8개로 제한 (char [9]나 string으로 선언)
  - D-day를 설정한 후 날짜가 변경되면, D-day를 변경된 날짜에 맞추어 변경할 것
    - 아래 예에서 20130531의 D-300이 20120804였다가, 날짜가 20120229로 바뀌면 D-300이 20120229 기준의 D-300을 계산하여 20110505을 출력
    - D-day는 https://search.naver.com/search.naver?query=디데이계산기 기준으로 계산
      - "+정수"로 표현되는 이후 d-day는 네이버페이지의 "일째 되는 날"(당일을 포함하여 계산함)
        "-정수"로 표현되는 이전 d-day는 네이버페이지의 "D-"의 결과와 동일하게 나와야 함
  - [Date]로 시작하는 줄은 cout << currentDate ; 로 출력
    - 이 문제에서는 currentDate가 dday를 포함하는 클래스 설계여야 함
    - 여러 클래스를 선언할 수도 있음. 보다 나은 설계에 대해서 고민할 것

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.

[Date] 2022/09/29 [D-day] NONE
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : +
[Date] 2022/09/30 [D-day] NONE
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : +
[Date] 2022/10/01 [D-day] NONE
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : -
[Date] 2022/09/30 [D-day] NONE
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : -1004
[Date] 2022/09/30 [D-day] 2019/12/31
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : +
[Date] 2022/10/01 [D-day] 2020/01/01
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : 20300101
[Date] 2030/01/01 [D-day] 2027/04/03
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : +100
[Date] 2030/01/01 [D-day] 2030/04/10
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : 20300229
*** Error!! Set day as 1
[Date] 2030/02/01 [D-day] 2030/05/11
Move date(yyyymmdd, (tomorrow)+, (yesterday)-, set D-day(+/-int), Quit(Q/q)) : q
=== END ===

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.