실습문제-2
끝자리 숫자 계산하기
(오류수정의 예)

2023. 03

국민대학교

소프트웨어융합대학

실습문제 2: 끝자리 숫자 계산하기

끝자리 숫자 계산하기

여러 개의 자연수가 주어졌을 때, 주어진 모든 자연수를 곱한 수의 마지막 자리수(1의 자리수)를 계산하는 프로그램을 작성하시오.

예를 들어, 세 개의 자연수 9, 314, 27을 곱한 수의 마지막 자리수는 2 이다.

[입력과 출력의 예]

입력	출력
4 3 9 314 27 1 1 5 123456 234567 385025 218569 321237 16 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	2 1 0 1

오류 프로그램의 예 (1)

```
#include <iostream>
using namespace std;

int main(void)
{
    int numTestCases;

    /* read the number of test cases */
    cin >> numTestCases;


    for(int i=0; i<numTestCases; i++)
    {
        int numData, data;
        int last1Digit = 1;


        cin >> numData;

        for (int j=0; j<numData; j++)
        {
            cin >> data;
            last1Digit *= data;
        }

        cout << last1Digit % 10 << endl;
    }

    return 0;
}
```

```
4
3 9 314 27
1 1
5 123456 234567 385025 218569 321237
16 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

counter example
(반례)

정수연산 계산 도중에 overflow가 발생하여 잘못된 결과값이 구해질 수 있음

오류 프로그램의 예 (2)

```
#include <iostream>
using namespace std;

int main(void)
{
    int numTestCases;

    /* read the number of test cases */
    cin >> numTestCases;


    for(int i=0; i<numTestCases; i++)
    {
        int numData, data;
        int last1Digit = 1;


        cin >> numData;

        for (int j=0; j<numData; j++)
        {
            cin >> data;
            last1Digit *= data;
            last1Digit %= 10;
        }

        cout << last1Digit << endl;
    }

    return 0;
}
```

```
4
3 9 314 27
1 1
5 123456 234567 385025 218569 321237
16 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

<주의할 사항>

1. 이 프로그램을 구현하여 입력의 예에서 제시된 데이터의 출력을 구하시오.
2. 출력값은 '입출력의 예'제시된 출력값과 같은가요?
3. 그렇다면, 이 프로그램은 문제를 제대로 해결하는 프로그램이라고 말할 수 있나요? (Yes/No)
4. 그렇지 않다면, 프로그램의 오류를 발생시키는 counter example (반례)의 예를 하나만 제시하시오.
5. 오류를 수정하여 문제를 제대로 해결하는 프로그램을 작성하시오.

문제해결 프로그래밍 하기

1. 문제의 이해
   - 문제를 잘 읽어보고, 문제에서 주어진 예를 통하여 문제의 이해도를 높인다.
   - 문제를 해결하는 자신만의 해결방법(알고리즘)을 고안해 본다.

2. 프로그래밍
   - 단계 1에서 만든 해결방법을 프로그래밍 한다.

문제해결 프로그래밍 하기

3. 입출력 데이터의 예
   - 프로그램이 입출력의 예에서 제시된 입력에 대하여 출력값을 계산하는지를 검사한다.
   - 정확한 출력값을 계산하지 않는 경우에는
     - ✓ 자신이 고안한 알고리즘을 입력데이터에 대해서 손으로 시뮬레이션 해보고, 틀린 답을 계산한다면, 단계 1로 돌아가서 문제를 좀 더 정확하게 이해하고, 새로운 알고리즘을 고안해 본다.
     - ✓ 알고리즘이 맞다면, 단계2에서 자신이 만든 해결방법을 정확하게 프로그래밍하였는 지를 확인한다.
   - 정확하게 구현하였는데도 잘못된 출력값을 계산하는 경우에는 고안한 알고리즘이 틀린 경우이므로 단계 1로 되돌아 간다.

문제해결 방법 프로그래밍 하기

4. Counter Example 조사:
   - 자신이 고안한 알고리즘을 프로그램으로 구현하여 입력의 예에 대해서 정확한 출력값을 계산한다고 하더라도, 아직 문제를 해결하는 프로그래밍을 완성한 것은 아니다.
   - 단지 주어진 소수의 테스트 데이터에 대하여 검증을 하였을 뿐이다.
   - 충분한 개수의 테스트 데이터를 사용하여 검증하지 않았으므로, 자신의 프로그램에서 잘못된 출력을 내는 입력 데이터(즉, counter example)를 찾아본다.

5. 해결 프로그래밍 완성:
   - 충분한 개수의 입력 데이터에 대해서도 정확한 출력을 계산한다면 해결 프로그램이 완성되었다고 판단할 수 있다. (counter example을 찾을 수 없을 때)

문제해결 방법 프로그래밍 하기

(참고)

- 채점 서버는 충분한 양의 입력데이터를 가지고 있으며, 여러분의 프로그램에 존재하는 오류를 확인할 수 있는 counter example이 많다고 생각하면 된다.

- 자신의 프로그램을 '뽀개는' counter example을 찾는 능력이 프로그래밍 능력 중에서도 매우 중요한 능력 중의 하나이다.