Digit Grouping Quiz

Sunny Kwak
sunnykwak@daum.net

요약 (abstract)

- 간단한 프로그램 구현 퀴즈
  - 10 진수 숫자를 입력 받아 1000 자리 구분자를 표시한 후, 출력하는 프로그램을 작성한다.
  - 실무에서 활용 가능한 실용적인 문제를 풀어 보도록 한다.
- 컴퓨터의 문제 해결 방식 이해
  - 사람의 컴퓨터의 문제 해결 방식의 차이를 이해한다.
  - 컴퓨터의 문제 해결 방식에 적합한 논리를 구성한다.
- 다양한 문제 풀이 방식
  - 해답을 얻기 위한 다양한 절차 혹은 구현 방식을 경험한다.
- 성능 관점에서 바라보기
  - 최적(optimum)의 코드를 어떻게 작성하는가를 경험한다.
- 학습 목표
  - 중요한 것은 '정답'이 아니라, 정답에 접근하는 태도, 길을 찾는 방식.
  - 프로그래밍을 잘하는 사고는 '암기'로 배울 수 없고,
    '뇌 회로'를 논리적인 문제 풀이에 적합하도록 '훈련' 시키는 과정을 필요로 한다.

목차

- 선행 지식과 문제 이해
  - 선행 지식
  - 요구사항
  - 인간과 컴퓨터의 문제 해결 방식
  - 표현에 따른 처리 방식 차이
- 문제 풀이
  - 문자 타입으로 해결
  - 숫자 타입으로 해결
- 성능 관점에서 다시보기

문제를 풀기 위한 준비 운동

선행 지식과 문제 이해

선행 지식

- **숫자 그룹(Digit Grouping)**
  - 숫자의 크기를 쉽게 파악할 수 있도록, 구분자(seperator)를 표시하는 것.
- **1000 자리 구분자(thousands separator)**
  - 3자리 마다 구분자를 표시하는 방식
  - 예를 들어, 5자리 숫자 "15368"을 표기할 때, "15,368"로 표현한다.
  - 영어권에서는 3자리 마다 구분자를 넣는 것이 자연스러움.
    (1 thousand, 10 billion, 32 million ...)
- **10000 자리 구분자**
  - 동양권에서는 4자리 마다 구분자를 넣는 것이 자연스러움.
    (10만, 100억, 1234조)
  - 그러나, global standard 가 3자리!

인용 : http://en.wikipedia.org/wiki/Decimal_mark

요구사항 (Requirements)

- **프로그램의 기능에 대한 요구사항은 다음과 같다.**

- **10 진수 숫자 입력**
  - 10진수 숫자를 입력 받는다.
  - 예를 들면, '99' 혹은 '123984' 등 정해지지 않은 자릿수의 값이다.

- **100 자리 구분자 표기 후 출력**
  - 구분자 표기라 함은 3자리 마다 쉼표(comma)를 추가하는 것.
  - 앞서 입력한 10진수를 예로 들자면, '99' 와 '123,984' 형태로 출력되어야 한다.

인간과 컴퓨터의 문제 해결 방식

- **인간의 문제 해결 방식**
  - 사람은 '숫자'를 숫자 뿐만 아니라 문자로도 이해한다. 그렇게 학습하고 또한 사고할 수 있도록 훈련되어 있다.
  - '1397'를 '1,397'를 변환할 경우, 숫자이며 동시에 문자인 것을 이해하고 손쉽게 3개의 숫자마다 중간에 구분자 표시(comma)를 넣을 수 있다.

- **컴퓨터의 문제 해결 방식**
  - 컴퓨터는 데이터를 숫자 혹은 문자로만 이해한다. 동시에 2가지 관점으로 바라보는 것이 불가능하다.
  - '1397'이라는 값이 입력되었을 때, 숫자로 이해(처리)할 것인지 혹은 문자로 받아들일지에 따라 변환 절차(알고리즘)가 달라지게 된다.

- **컴퓨터 관점에서 바라보기**
  - 컴퓨터와 인간의 문제 해결 방식 차이를 받아들여야(혹은 이해해야), 컴퓨터로 처리할 수 있는 알고리즘을 작성할 수 있다.

표현에 따른 처리 방식 차이

- **컴퓨터의 관점에서 문제해결 실마리 찾기**
  - 컴퓨터는 데이터를 '문자' 혹은 '숫자'로 표현한다.
    (데이터 표현의 차이를 다른 말로 데이터 타입의 차이라고 한다.)
  - 2가지 표현 간에 변환은 가능하지만, 하나의 데이터를 동시에 다른 형식(타입)으로 처리하지는 못한다.
  - 따라서, '문자' 혹은 '숫자'로 처리할지를 먼저 결정해야 한다.

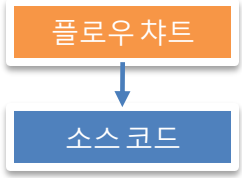
- **'문자' 타입으로 처리하는 해법**
  - 입력 값을 문자열로 변환한 후에 3자리씩 낮은 자리부터 분리한 다음, 구분자를 추가하면서 새로운 문자열에 추가한다.

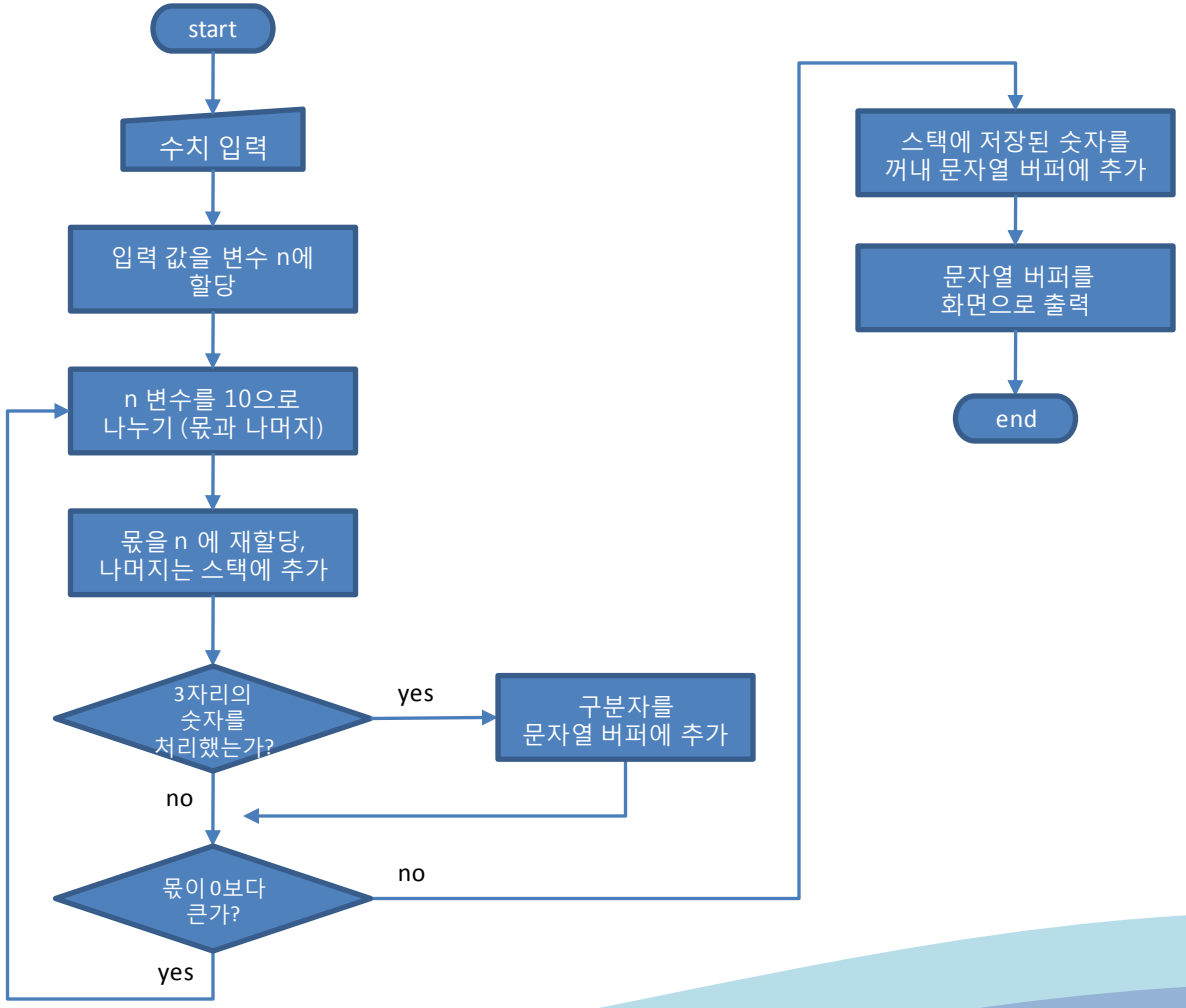
- **'숫자' 타입으로 처리하는 해법**
  - 입력 값이 10으로 나누고, 몫이 0이 될 때까지 나머지를 계속 스택(stack)에 저장한다. 스택에 저장된 나머지를 꺼내면서 구분자와 함께 출력한다.

숫자 타입과 문자 타입으로 처리하기

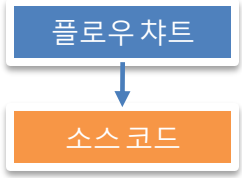
문제 풀이

숫자 타입으로 처리


플로우 챠트
소스 코드


start
수치 입력
입력 값을 변수 n에 할당
n 변수를 10으로 나누기 (몫과 나머지)
몫을 n 에 재할당, 나머지는 스택에 추가
3자리의 숫자를 처리했는가?
yes
구분자를 문자열 버퍼에 추가
no
몫이 0보다 큰가?
no
yes
스택에 저장된 숫자를 꺼내 문자열 버퍼에 추가
문자열 버퍼를 화면으로 출력
end

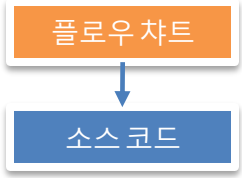
숫자 타입으로 처리

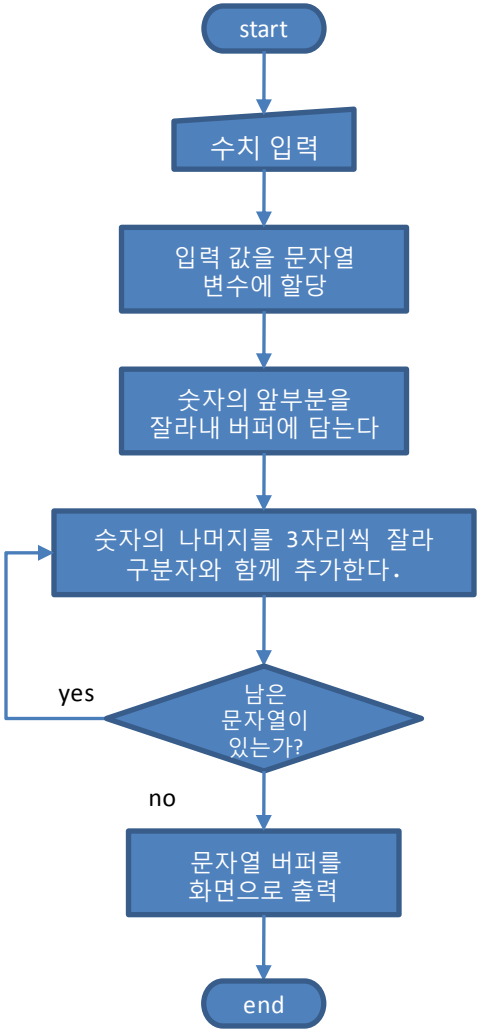



```
import java.util.Scanner;
import java.util.Stack;
import java.util.EmptyStackException;

/**
 * 10진수를 입력받아, 천(1000)자리 구분자를 추가하는 프로그램.
 */
public class DigitGrouping {

    public static void main(String args[]) {

        // 10진수 값 입력
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number : ");
        int inputNumber = sc.nextInt();

        // 구분자를 추가한 후, 출력.
        System.out.println("Output : " + convert(inputNumber));
    }
```

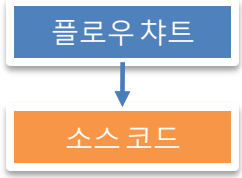
```
    private static String convert(int number) {
        final int DECIMAL_NUMBER = 10;
        final char THOUSANDS_SEPARATOR = ',';

        Stack<Character> digitStack = new Stack<Character>();
        StringBuilder sb = new StringBuilder();

        int quotient, spare, count = 0;
        do {
            // 몫과 나머지 계산
            quotient = number / DECIMAL_NUMBER;
            spare = number % DECIMAL_NUMBER;

            // 몫을 n 에 재할당, 나머지는 스택에 추가
            number = quotient;
            digitStack.push(Character.forDigit(spare, 10));

            // 3개의 숫자를 추가했다면, 구분자를 버퍼에 추가
            if(++count == 3 && quotient > 0) {
                digitStack.push(THOUSANDS_SEPARATOR);
                count = 0;
            }

        } while(quotient > 0);

        // 스택에 저장된 숫자를 꺼내 문자열 버퍼에 추가
        while(!digitStack.empty()) {
            sb.append(digitStack.pop());
        }

        return sb.toString();
    }

}
```

'문자' 타입으로 처리


플로우 챠트
소스 코드


start
수치 입력
입력 값을 문자열
변수에 할당
숫자의 앞부분을
잘라내 버퍼에 담는다
숫자의 나머지를 3자리씩 잘라
구분자와 함께 추가한다.
yes
남은
문자열이
있는가?
no
문자열 버퍼를
화면으로 출력
end

‘문자’ 타입으로 처리




```
import java.util.Scanner;
import java.util.Stack;
import java.util.EmptyStackException;

/**
 * 10진수를 입력받아, 천(1000)자리 구분자를 추가하는 프로그램.
 */
public class DigitGrouping2 {

    public static void main(String args[]) {

        // 10진수 값 입력
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number : ");
        int inputNumber = sc.nextInt();

        // 구분자를 추가한 후, 출력.
        System.out.println("Output : " + convert(inputNumber));
    }
```

```
    private static String convert(int number) {
        final char THOUSANDS_SEPARATOR = ',';
        StringBuilder sb = new StringBuilder();

        // 입력 숫자를 문자열 타입으로 변환
        String numberAsStr = String.valueOf(number);

        // 맨 앞부분에서 잘라낼 길이를 계산하고, 출력 버퍼에 담는다.
        int separationIndex = numberAsStr.length() % 3;
        sb.append(numberAsStr.substring(0, separationIndex));

        // 숫자의 나머지를 3자리씩 잘라서 구분자와 함께 추가한다.
        while(separationIndex < numberAsStr.length()) {
            if(separationIndex > 0) {
                sb.append(THOUSANDS_SEPARATOR);
            }
            sb.append(numberAsStr.substring(separationIndex,
                    separationIndex+3));
            separationIndex += 3;
        }

        return sb.toString();
    }

}
```

최적(optimum)의 코드를 어떻게 작성하는가?

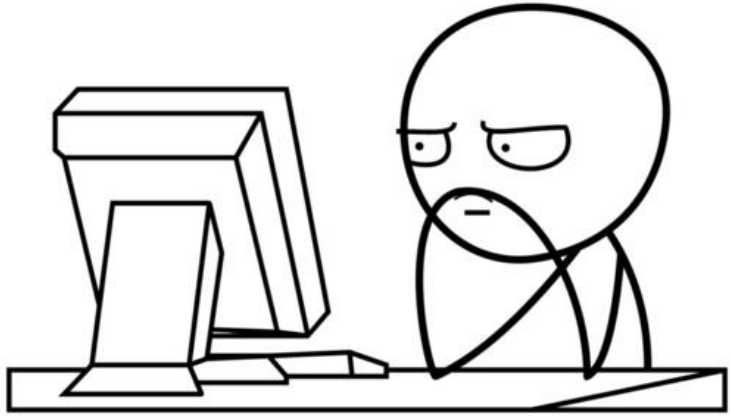
성능 관점에서 바라보기

문제 해결이 전부인가?

- **메모리와 CPU 처리 속도**
  - 2가지 방법 중에서 메모리 사용량과 처리 속도가 빠른 것은 어떤 방식일까?
  - 작은 기능을 수행하는 함수라서, '고민거리'가 아니라고 생각할수도 있다.

- **그러나, 현장에서는...**
  - 만일 1,000 만명에 달하는 고객에게 카드 명세서를 보내는 작업을 해야 한다면 이런 함수가 '천만 * 수십번 혹은 수백번' 실행된다. 따라서, 많은 시간이 소모되죠.
  - 매달 이런 처리를 해야 하는 금융 기관에서는 이런 일괄 작업들이 많은데, 명세서를 늦게 발송하게 된다면 큰 문제가 된다. 별 거 아닌 코드를 섬세하게 들여다 볼 줄 아는 자세도 필요하다. 그래야 정말 큰 문제도 풀 수 있다.

- **계획, 구현, 그리고 검증!**
  - 실행 시간을 비교해 보아야 한다. 예상과 달리 두 가지 처리 방식의 수행 시간이 별 차이가 없을 수도 있다. 그러나 중요한 것은 '계획'하고 '구현'하고 나아가, '검증'하는 자세이다.



To be continued !!