March 3, 2006 at 14:17

1. Introduction. 이 프로그램은 존 벤틀리의 유명한 저서 *Programming Pearls*에 나오는 비트맵을 이용한 정렬을 C 언어를 이용해서 구현한 것이다. 문제는 다음과 같다.

Input: 최대 n개의 양의 정수를 포함하는 파일로, 각 숫자는 $n(=10^7)$보다 작다. 파일 내의 어떤 숫자도 두 번 이상 나올 수 없으며, 숫자 이외에 관련된 데이터는 없다.
Output: 입력된 정수를 오름차순으로 정렬한 리스트
Constraints: 메모리를 많아야 대략 1MB정도를 사용할 수 있고, 디스크 공간은 충분하다. 실행 시간은 최대 몇 분 정도가 될 수 있고, 10초 정도 안에 작업을 끝낼 수 있으면 충분하다.

2. 이 프로그램의 구조는 다음과 같다.

```
#include <stdio.h>
  ⟨ 전역변수들 3 ⟩
  ⟨ 함수들 4 ⟩
  int main(void)
  {
    int i;
    for (i = 0; i < N; i++)      /* 모든 비트들을 0으로 초기화 한다 */
      clr(i);
    while (scanf("%d", &i) ≠ EOF)      /* 숫자에 해당하는 비트를 1로 마크한다. */
      set(i);
    for (i = 0; i < N; i++)      /* 결과 출력 */
      if (test(i)) printf("%d\n", i);
    return 0;
  }
```

3. Bitwise Operations. 대부분의 시스템에서 **int** 형 정수는 4바이트(32비트)로 표현 되므로, 10^7개의 비트를 표현하기 위해서는 $\lceil 10^7/32 \rceil$개의 정수가 필요한데, 이 숫자는 C의 정수 계산으로 $10^7/32+1$와 같다. 따라서 이러한 용도로 **int** 형 배열 a를 정의 하면, $a[0]$에는 0번째 부터 31번째 까지의 32개의 비트를 저장할 수 있고, 마찬가지로 32번째 부터 63까지의 비트들은 $a[1]$에 저장된다.

이해를 쉽게하기 위해서 구체적인 예를 들어보자. 100번째 비트는 배열 a의 몇 번째 원소의 몇 번째 비트일까? 먼저, 배열 a의 몇 번째 원소인지를 알기는 쉽다. 즉 100을 32로 나눈 몫이 바로 그 값이다. 100/32의 값은 3이므로, $a[3]$이 100번째 비트가 들어있는 원소이다.$(100/32 = 100 \gg 5)$ 그렇다면, $a[3]$의 몇 번째 비트가 100번째 비트일까? 이 값은 100을 32로 나눈 나머지일 것이다. 왜냐하면 $a[3]$의 첫번째 비트는 전체적으로 보면 96번째 비트가 될 것이고, 100번째 비트는 $a[3]$ 첫번째 비트에서 4만큼 떨어져 있기 때문이다.

```
#define BITSPERWORD 32
#define SHIFT 5
#define MASK #1F
#define N 10000000
⟨ 전역변수들 3 ⟩ ≡
  int a[1 + N/BITSPERWORD];
```

This code is used in section 2.

4. 위의 설명을 기반으로 i번째 비트를 1로 설정하는 함수는 다음과 같다. $i \gg$ SHIFT가 바로 i를 32로 나눈 몫을 나타내므로 해당 배열의 원소 위치를 나타내고, $1 \ll (i \,\&\, \texttt{MASK})$는 해당 원소의 몇번째를 위치하는 가를 계산하는 식이다. $i \,\&\, \texttt{MASK}$가 바로 i를 32로 나눈 나머지가 되는 것이다. 따라서 $1 \ll (i \,\&\, \texttt{MASK})$는 1을 위에서 설명한 대로 기준 위치에서 떨어진 만큼 1을 시프트하는 것이므로, 해당 원소의 해당 비트를 나타내는 것이다.

```
⟨ 함수들 4 ⟩ ≡
  void set(int i)
  {
    a[i ≫ SHIFT] |= (1 ≪ (i & MASK));
  }
```

See also section 5.

This code is used in section 2.

5. 위와 같은 원리로 i번째 비트를 0으로 클리어하고(clr), 그 비트의 값을 알아내는(test) 함수는 다음과 같다.

```
⟨ 함수들 4 ⟩ +≡
  void clr(int i)
  {
    a[i ≫ SHIFT] &= ∼(1 ≪ (i & MASK));
  }
  int test(int i)
  {
    return a[i ≫ SHIFT] & (1 ≪ (i & MASK));
  }
```

6. Index.

⟨ 전역변수들 3 ⟩ Used in section 2.
⟨ 함수들 4, 5 ⟩ Used in section 2.

BITMAP