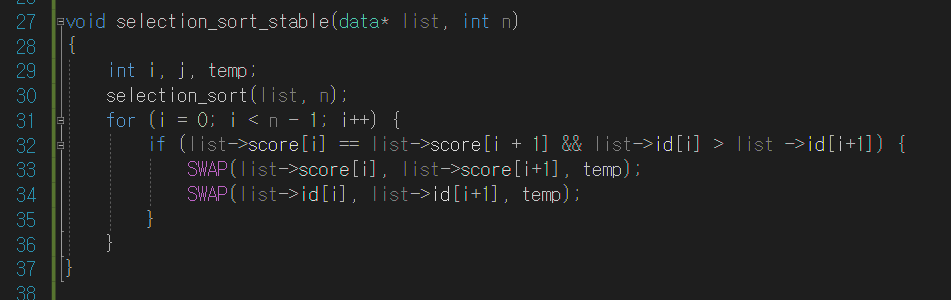
Homework #1

**Idea**

**Code Explanation**

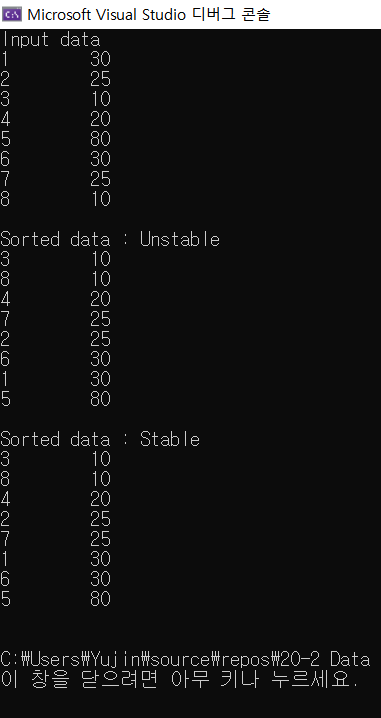
selection\_sort\_stable 함수만 새롭게 구현하였고, 나머지 코드는 selection\_sort\_skeleton.cpp 파일과 동일합니다.

**void selection\_sort\_stable(data\* list, int n)**



먼저 void selection\_sort(data\* list, int n) 함수를 실행한다. 그 후 결과값이 저장된 list 배열 반복문을 돌면서 data에 저장된 score 값이 같으면 id 값을 비교하여 같은 score 값들 중에서 id 값이 작을수록 list의 앞에 위치하도록 순서를 바꿔준다.

**실행결과**

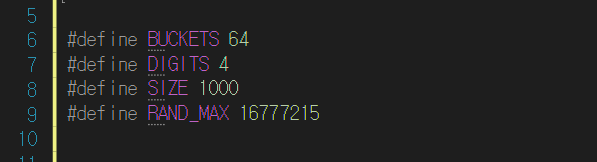


Homework #2

**Idea**

변수를 랜덤하게 생성하여 배열에 저장할 때, 각 자리수에 알맞은 값을 계산하여 함께 저장한다.

**Variable**

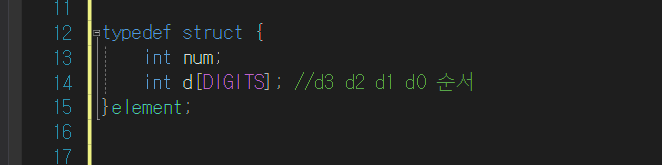


#define BUCKETS 64 -> 한 자리당 가능한 숫자의 개수 (0~2^6)

#define DIGITS 4 -> 자리수 개수

#define SIZE 1000 -> 정렬할 숫자 개수

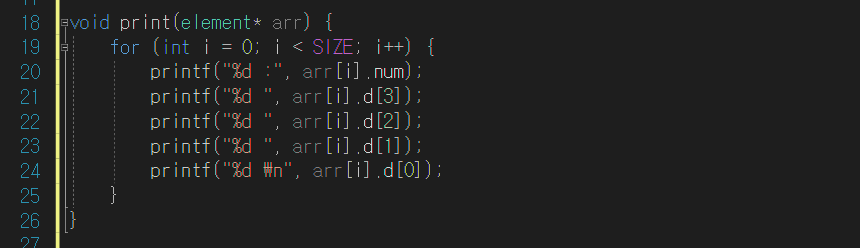
#define RAND\_MAX 16777215 -> 랜덤으로 생성할 수 있는 최대 숫자인 2^24-1의 값이다.



Num 변수에는 랜덤으로 생성한 정수값이, d[] 배열에는 각 자리수(digit)에 저장된 값이 저장된다.

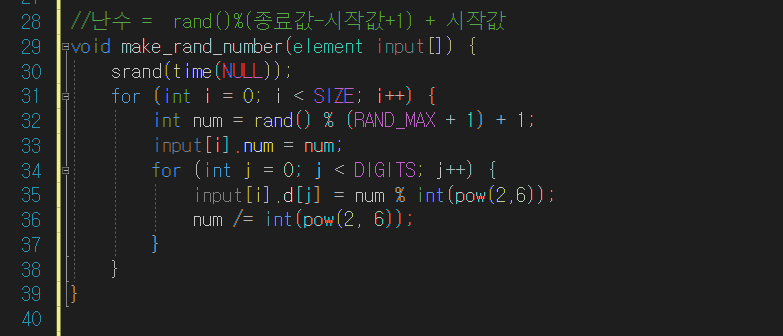
**Code Explanation**

1. **void print(element\* arr)**



element 배열을 출력하기 위한 print 함수이다. 제일 첫번째 수는 랜덤으로 생성된 값을 십진수로 출력한 값이고 그 이후에는 각 digit에 6bit씩 저장한 값이 차례로 출력된다.

1. **void make\_rand\_number(element input[])**

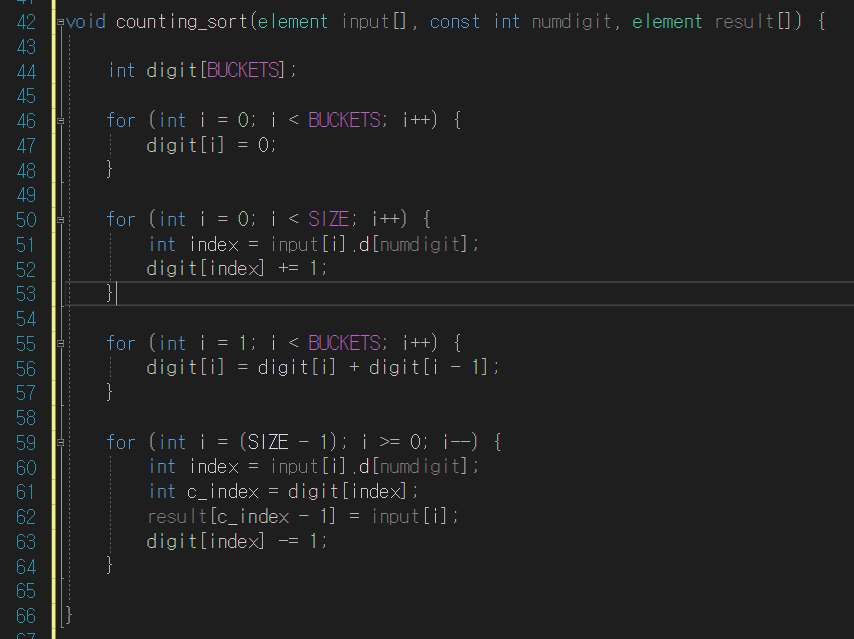


0~2^24-1 까지의 랜덤한 숫자 1000개를 생성해낸다.

하지만 여기서 문제의 조건인 data range: 0 ~2^24-1 를 만족하기 위하여 RAND\_MAX를 2^64-1 값으로 설정해놓았지만, c언어의 랜덤함수인 rand() 함수가 0~32767 사이의 값만 반환하기 때문에 큰 수는 생성하지 못한다. Rand() 함수끼리의 곱셈으로 그 이상의 자릿수를 나타내는 random 값을 생성하는 것도 가능하나 코드가 복잡해져 일단은 이렇게만 구현하였다.

이때 숫자 생성과 더불어 element구조체에 각 값들을 설정하고, 그 객체의 값을 element 배열에 넣는 작업도 같이 하기위해, input[] array를 인자로 받아 이 input[]에 random값을 저장한다. 이때 element 구조체의 요소인 d[] 배열에도 자리수마다 각각 저장하기 위해 for문을 한번 더 사용해서, 각 자리수마다 저장하는 연산을 한번 더 했다.

1. **void counting\_sort(element input[], const int numdigit, element result[])**



counting sort 알고리즘을 실행하는 코드이다.

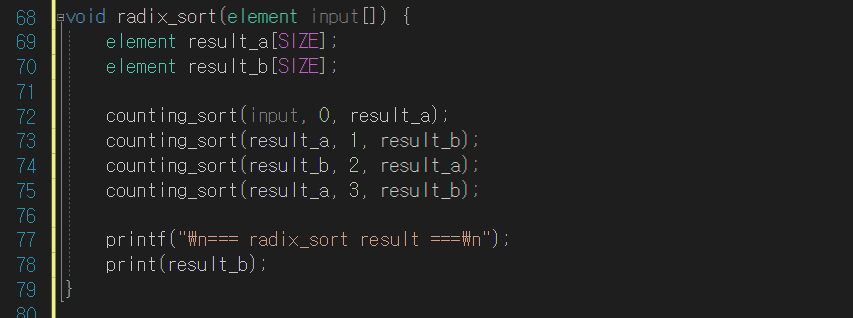
1) (0~2^6의 값을 가질 수 있는) 자리수 배열 생성 및 0으로 초기화

2) 자리수의 빈도수를 digit[] 배열에 저장한다.

3) 빈도수를 누적하여 나타내는 반복문 연산

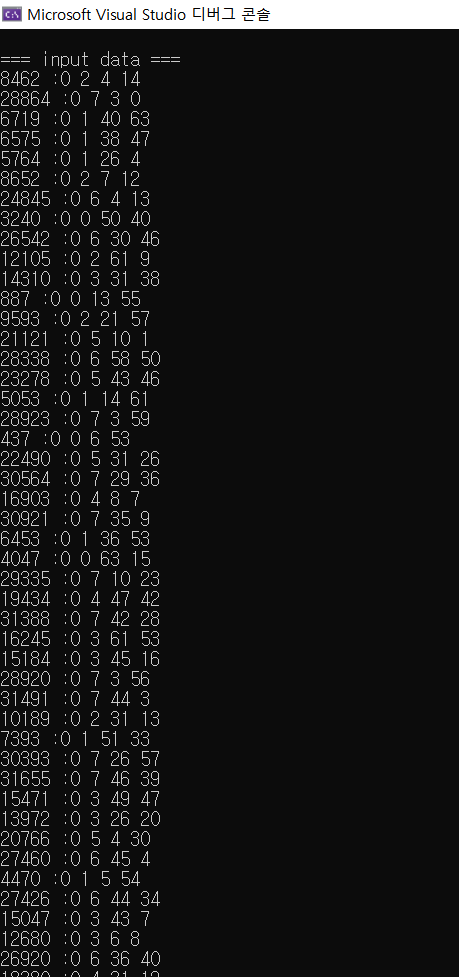
4) input의 가장 마지막 index에서 0까지 접근한다. 해당 input의 numdigit자리의 숫자가 가진 누적 빈도수를 digit에서 확인하고, 해당 digit[] 배열에 들어있는 값을 result[] 배열의 index로 설정하고, 누적 digit의 값을 1 감소한다.

1. **void radix\_sort(element input[])**

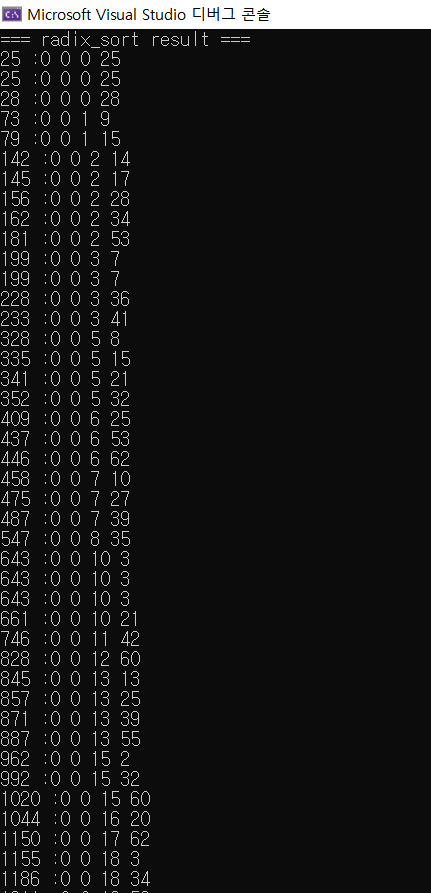


numdigit즉, digit의 자리수가 0에서부터 3까지 LSD에서 MSD로 증가하도록 count sort를 실행해준다.

**실행결과**



Input data는 random 하게 들어오지만,



radix\_sort(element input[]) 함수를 실행하고 결과를 출력해보면, 오름차순으로 정렬된 것을 확인할 수 있습니다.