단어 번역기 (메모리 버전)

```
#include <stdio.h>
#include <string.h>

#define MAX_WORD  23

char g_szDic[MAX_WORD][2][32] =
{
        {"i",           "나"          },
        {"you",         "너"          },
        {"me",          "나를"        },
        {"he",          "그"          },
        {"she",         "그녀"        },
        {"man",         "남자"        },
        {"woman",       "여자"        },
        {"boy",         "소년"        },
        {"girl",        "소녀"        },
        {"school",      "학교"        },
        {"hate",        "증오하다"    },
        {"this",        "이것"        },
        {"that",        "저것"        },
        {"a",           "하나의"      },
        {"an",          "하나의"      },
        {"help"         "도와주다"    },
        {"hello",       "안녕"        },
        {"world",       "세계"        },
        {"love",        "사랑"        },
        {"game",        "게임"        },
        {"go",          "가다"        },
        {"crazy",       "미친"        },
        {"are",         "는"          }
};


//-------------------------------------------------------------------
// 번역 함수 선언..
//-------------------------------------------------------------------
void Trans(char *szIN, char *szOUT, int *ipWordCount);

//-------------------------------------------------------------------
// g_szDic 배열에서 원하는 문자열을 검색한다.
//-------------------------------------------------------------------
char *FindWord(char *szEng);

//-------------------------------------------------------------------
// 메인 함수.
//-------------------------------------------------------------------
void main(void)
{
        char szINText[128] = "";          // 영어 입력 문자열.
        char szOUTText[256] = "";         // 번역된 문자열 출력용.
        int iWordCount;

        printf("63자 이하로 영어를 입력해주세요. Wn:");

        //-------------------------------------------------------------------
        // gets 함수를 사용해서 문자열을 입력받는다. 엔터코드가 들어올때까지 키보드 입력을 받는다.
        // szINText 가 128 길이로 생성되어있으므로 128자 이상이 들어보면
        // 오버플로우 에러가 발생할 수 있다. gets 함수는 이를 체크하지 못한다.
        //-------------------------------------------------------------------
        gets(szINText);

        //-------------------------------------------------------------------
        // g_szDic 배열엔 모두 소문자로만 사용했으므로 입력받은 문자들을 전부 소문자로 바꿔주자.
        //-------------------------------------------------------------------
        strlwr(szINText);

        //-------------------------------------------------------------------
        // 번역 함수 호출.
        // szINText 로 입력받은 문자열을 넘겨주면, 두번째 인자로 들어간 szOUTText 포인터에
        // 번역된 문자열을 넣어서 돌려준다.
        //-------------------------------------------------------------------
        Trans(szINText, szOUTText, &iWordCount);
        printf("Wn - 총 %d 단어가 번역 되었습니다. Wn", iWordCount);
        printf("# %s Wn", szOUTText);
}



//-------------------------------------------------------------------
// 번역 함수 부분.  szIN 으로 영어 문장이 들어오며, 이를 한글로 번역한 문장을 szOUT 에 넣어준다.
// 세번째 인자 ipWordCount 는 int 포인터로 검색된 단어의 개수를 알려준다.
//-------------------------------------------------------------------
void Trans(char *szIN, char *szOUT, int *ipWordCount)
{
        char *chpEngPtr, *chpKorPtr;
        int iCount, iLen = strlen(szIN);

        //-------------------------------------------------------------------
        // 번역된 단어 개수를 카운트 할것이다. 미리 0 으로 초기화.
        // 포인터로 입력 받았으므로 * 로 접근하여 사용하는것로 값이 전달된다.
        //-------------------------------------------------------------------
        *ipWordCount = 0;

        //-------------------------------------------------------------------
        // chEngPtr에 입력받은 문자열의 첫번째 위치를 넣어두자. 이를 사용해서 단어검색을 할것이다.
        //-------------------------------------------------------------------
        chpEngPtr = szIN;
```

```
//-------------------------------------------------------------
// 단어를 구분하기 위해서 스페이스(띄어쓰기)를 검색해야 한다.
// 스페이스는 아스키코드 0x20 이므로 0x20 을 검색하면 된다.
// iLen (문자열 길이) 에 + 1 로 루프를 도는 이유는 이 루프 안에서 마지막 단어까지 한번에
// 검색하기 위함이다. 마지막 글자의 경우 0x20 검사로 찾을 수 없으므로 NULL 값을 검사해서 찾는다.
//-------------------------------------------------------------
for ( iCount = 0; iCount < iLen + 1; ++iCount )
{
        //-------------------------------------------------------
        // 0x20 스페이스의 경우와, 문자열의 마지막 NULL 인 경우를 찾는다.
        //-------------------------------------------------------
        if ( 0x20 == szIN[iCount] || '₩0' == szIN[iCount] )
        {
                //-----------------------------------------------
                // 스페이스가 입력된 부분에 NULL 을 입력하자.
                // NULL 을 입력하는 이유는 strcmp 함수를 이용해서 단어단위로 검색을 할 예정이므로
                // 단어 뒤에 NULL 이 입력되어야만 strcmp 함수에서 이를 문장으로 인식한다.
                //-----------------------------------------------
                szIN[iCount] = '₩0';
                //-----------------------------------------------
                // szIN[iCount] 위치에서 스페이스를 찾았으므로
                // chpEngPtr[0] ~ chpEngPtr[iCount] 사이가 하나의 단어라는 이야기다.
                // 위에서 iCount 위치에 NULL 을 넣었으므로 chpEngPtr 은 단어 하나가 있는
                // 문자열을 가르키는 포인터가 되었다.
                // 이를 FindWord 함수에 넣고 g_szDic 배열에서 검색된 한글 문자열 포인터를 얻자.
                //-----------------------------------------------
                chpKorPtr = FindWord(chpEngPtr);
                //-----------------------------------------------
                // 단어별로 칸을 띄워주기 위해 번역문자열이 입력되는 szOUT 에 단어를 입력하기 전에
                // " " 로 스페이스를 입력해준다.
                //-----------------------------------------------
                strcat(szOUT, " ");
                if ( NULL == chpKorPtr )
                {
                        //---------------------------------------
                        // strcat 함수는 첫번째 인자 문자열에 두번째 인자의 문자열을 붙여준다.
                        // chpKorPtr 이 NULL 이면 원하는 단어를 찾지 못했다. 그냥 영어 붙이자.
                        //---------------------------------------
                        strcat(szOUT, chpEngPtr);
                }
                else
                {
                        //---------------------------------------
                        // NULL 이 아니라면 단어가 있으므로 한글단어로 붙여주자.
                        //---------------------------------------
                        strcat(szOUT, chpKorPtr);
                        (*ipWordCount)++;
                }
                //-----------------------------------------------
                // 방금 찾은 단어 뒤의 단어부터 다시 검색하므로 chpEngPtr 을 iCount + 1
                // 위치로 옮겨주자.
                // 그래야 위에서 입력한 NULL 다음부터 검색이 가능해진다.
                //-----------------------------------------------
                chpEngPtr = &szIN[iCount + 1];
        }
}
}
```

```
//-------------------------------------------------------------
// g_szDic 배열에서 원하는 문자열을 검색한다.
//
//-------------------------------------------------------------
char *FindWord(char *szEng)
{
        //-------------------------------------------------------
        // g_szDic 배열에서 입력된 문자열(szEng) 을 검색하여
        // 이에 해당하는 한글 문자열 포인터를 리턴한다.
        //-------------------------------------------------------

        //-------------------------------------------------------
        // 영문 - g_szDic[N][0]
        // 한글 - g_szDic[N][1]
        //-------------------------------------------------------
        int iCount;
        for ( iCount = 0; MAX_WORD > iCount; ++iCount )
        {
                //---------------------------------------------
                // strcmp 함수는 두개의 문자열 (char 포인터) 을 입력 받아
                // 두개의 문자열이 같다면 0 을 돌려주고, 다르다면 0 이외의 값을 돌려준다.
                //---------------------------------------------
                if ( 0 == strcmp(szEng, g_szDic[iCount][0]) )
                {
                        //-------------------------------------
                        // 입력 받은 szEng 와 같은 영어단어를 찾았다면 그에 해당하는
                        // 한글 문자열 [N][1] 의 포인터를 리턴하자.
                        //-------------------------------------
                        return g_szDic[iCount][1];
                }
        }

        //-------------------------------------------------------
        // for 문 검색에서 찾지 못했다면 없다.. 그러므로 NULL 리턴.
        //-------------------------------------------------------
        return NULL;
}
```