

# REPORT

## 보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,  
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과 목 : 프로그래밍 기초와 실습  
과 제 명 : 실습8: Strings & Pointers  
담당교수 : 민 형 복 교수  
학 과 : 자연과학계열  
학 년 : 1학년  
학 번 : 2014310407  
이 름 : 이준혁  
제 출 일 : 2014년 11월 30일

## 1. Introduction

이번 실습에서는 string에 대해 공부한다. 그러나 저번 실습과 마찬가지로, string 역시 일종의 pointer이면서 array이므로, pointer와 array에 대해서도 공부한다고 볼 수 있겠다.

사실 string은 문자형 pointer type으로 정의된다. 문자열 하나에 담긴 문자들을 array처럼 생각한다. 이 때 가장 첫 번째 문자의 주소를 가진 pointer가 string이 되는 것이다. 이렇게 string의 '처음'을 알게 된다. 그렇게 되면 string이 어디서 끝나는지 알아야 하는데, 끝을 '\0'을 이용해 표현한다. string은 array나 pointer가 '문자'일 때의 특별한 경우라고 볼 수도 있다. 그렇기 때문에 기본적으로 string을 처리하는 방식은 pointer나 array와 같으나, 약간 다른 점이 있다.

이번 실습에서 구현해 볼 것은 swap함수와 strtok함수이다. swap함수는 이전에 정수를 swap하는 것을 배웠지만, string을 swap하는 것은 처음이다. 위에서 기술했듯이 string은 일반적인 pointer와 약간 차이점이 있음을 유의해서 swap해야 한다. 또, string을 separator 단위로 token으로 끊어 주는 strtok함수 역시 구현해 봐야 한다.

이번 실습은 저번에 배웠던 array, pointer의 문자형 버전인 string을 다뤄보면서 string을 어떻게 볼 것인지를 알고, 어떤 차이점이 있는지 알아 보는 데에 초점을

맞춰서 수행하는 것에 큰 의미가 있다.

## **2. Problem Statement**

**(A)Describe what the problem is**

### **문제 1) swap 2 strings**

실습 6에서 두 정수를 맞추는 함수를 작성한 것과 비슷하게, 두 string을 맞추는 함수를 작성해야 한다.

그리고 그냥 string을 담고 있는 문자형 pointer를 입력으로 주었을 때 어떤 문제가 나타나는지를 우선 설명한다. 그 다음에 string을 맞추는 함수를 사용할 때 왜 string을 배열로 선언하면 안 되는지 역시 설명해야 한다.

### **문제 2) strtok function**

<string.h>에 정의된 strtok함수와 같은 기능을 하는 함수를 구현해야 한다. 이 함수가 수행하는 역할은 다음과 같다.

separator라는 string안에 있는 문자들을 기준으로, 어떤 string을 갈라주는 역할을

한다. 이 때 갈라진 string의 일부들을 token이라고 한다. 즉, string을 token으로 갈라준다.

예를 들어, "I/am%a;student"라는 string이 있고, "/"%;"라는 separator가 있을 때, 위 string은 "I" "am" "a" "student"라는 네 개의 token으로 갈라진다.

즉, 이 함수는 분리될 string하나와 분리할 separator라는 string하나를 입력으로 받는다.

다만 주의할 점은 어떤 NULL이 아닌 string을 입력한다면, 처음부터 separator를 찾는다. separator를 발견하면 그 separator를 0으로 초기화하고, 처음부터 0까지의 string을 반환하게 된다.

만약 위 과정을 거친 후 다음 function call때 string 자리에 NULL이라는 값을 주면 이전에 탐색했던 string의 0 이후부터 다시 separator를 찾는다. 역시 separator를 발견하면 그 separator를 0으로 초기화하고, 이전에 찾았던 곳 이후부터 0까지의 string을 반환하게 된다. 만약 더 이상 separator가 없어 반환할 token이 없다면 NULL을 반환한다.

이 문제에서는 처음에 string하나를 주고, 그 다음에는 while문을 이용해 string 자리에 계속 NULL을 주어 이 함수가 NULL을 반환할 때까지 function call을 한다.

**(B) Describe how you solve the problem.**

## 문제 1) swap 2 strings

swap 함수의 내부를 어떻게 만들 지는 해본 적이 있어서 어렵지 않아 보였다. 다만 이 함수의 입/출력을 어떻게 할 지가 문제였다. swap함수는 함수 내부에서 값을 맞바꾸고 return하는 값은 없으니 출력은 금방 해결됐으나, 입력을 어떻게 할까 생각해 보았다.

정수를 swap할 때처럼 그 변수의 주소를 넘겨줘야 했다. 그런데 string은 문자형 pointer type이므로, 포인터의 주소를 넘겨 주는 식으로 만들어야겠다고 생각했다. 즉, 문자형 pointer pointer type인 두 string이 저장되어 있는 'pointer 변수의 주소'를 입력으로 받기로 했다. 이렇게 하면 별 문제가 없을 것 같았다.

## 문제 2) strtok function

이 함수가 어떤 구조를 가지도록 만들어야 할 지에 대해 구상해 보았다. 그러기 위해서는 입/출력을 생각해야 했다. 입력은 당연히 string과 token이고, 출력은 token화 된 string의 첫 문자의 주소를 출력으로 하면 string을 출력할 수 있을 것이라 보았다.

그리고 민형복 교수님의 practice8.pdf에도 나와있는 내용이지만 static으로 선언된 string을 담은 문자형 pointer 변수로 논리를 전개해 나가는게 좋을 것

같았다. 입력 받은 string이 NULL인지 여부를 판단해서, NULL이면 입력 받은 string을 static변수에 저장, 아니면 원래 있던 static변수의 값을 사용하는 식으로 하면 될 것 같았다.

그 후에는 입력 받은 string의 문자들을 하나씩 검사하는 작업이 필요하다고 보았다. 처음 문자부터 'W0'까지 각 문자와 token의 어느 것이라도 일치하는지 여부를 확인하기로 했다. 그렇기에 반복문을 두 번 사용할 것이라 보았다. 그리고 만약 일치하는 것이 있으면 static변수로 선언된 string의 그 문자를 'W0'으로 초기화하고 string의 시작점의 문자의 주소를 반환하기로 했다.

만약 string으로 NULL을 입력 받는다면 이전에 0으로 초기화 한 곳 다음부터 시작해야 했다. 이를 위해 셈을 하는 static변수가 하나 더 필요할 것이라 생각했다. 그 다음에는 위의 과정과 같은 방법을 거치면 될 것 같았다.

그런데 이렇게 구현하면 두 가지 문제가 있을 것 같았다. 첫 번째는 연속해서 separator들이 나오는 경우는 이상하게 출력될 것 같았다. 그래서 처음에 separator가 오거나 separator가 연속해서 온다면 적당한 방법으로 그 separator를 0으로 초기화 시킨 후에 다음 문자부터 시작하기로 했다. 이 과정에서 변수 하나가 더 필요할 것 같았다.

두 번째는 마지막 token을 출력할 수 없다는 것이었다. 만약 separator가 ;인 상태에서 "a;b"를 입력하면 a는 출력되지만 b는 출력되지 않을 것이었다. 이

경우는 '다음 function call에 NULL을 출력하도록'하면 될 것 같았다. 즉, 더 이상 separator가 없으면 일단 남은 token을 출력하고, 그 다음 번 function call때 NULL을 return하기로 했다. 그런데 이 방법은 두 번의 function call에 걸쳐 있는 방법이므로, 역시 static변수를 사용해야 될 것으로 보였다.

### 3. Implementation

main함수 내부에 주어진 내용은 민형복 교수님이 작성하신 p8.c와 같다. 이 부분은 변경할 수 없으므로 그대로 사용했다. 따라서 implementation에서는 함수들을 어떻게 작성 했는지를 기술할 것이다.

#### 3. 1. 문제 1 – swap 2 strings

민형복 교수님의 practice8.pdf의 4페이지에 탑재된 코드가 어떤 문제가 있는지, swap function을 어떻게 구현했는지, practice8.pdf의 6페이지에 탑재된 코드를 왜 쓸 수 없는 지와 어떻게 하면 swap할 수 있는지 총 세 가지에 대해 순서대로 기술하겠다.

##### 3. 1. 1. string을 입력으로 주면 안 되는 이유에 대한 설명

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

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

```

void swap(char *s1, char *s2)
{
    char temp[128];
    strcpy(temp, s1);
    strcpy(s1, s2);
    strcpy(s2, temp);
}

```

```

int main(void) {
    char *str1 = "I am a boy";
    char *str2 = "You are a girl";
    swap(str1, str2);
    printf("%s\n", str1);
    printf("%s\n", str2);
}

```

이 경우에는 str1, str2를 문자형 pointer가 아닌 string의 값이라고 생각해서 접근해야 한다. 문자형 pointer라고 생각하면 주소를 넘겨준다고 생각할 수도 있겠지만, 그런 게 아니라 그냥 'string의 값'만 넘겨 준 것이다. C언어의 함수는 call – by – value를 따르므로, 함수 내부에서는 값이 변경될 수 있겠지만, main에서는 결국 변한 것이 없다. 즉, 위와 같이 프로그램을 짜게 된다면 두



string이 서로 맞교환 되지 않을 것이다.

### 3. 1. 2. swap function 구현

문제 해결 방법에서 구상했듯이, 이 함수의 입력은 string의 주소였다. 그렇기 때문에 function call할 때 입력은 &str1처럼 받도록 main에 작성되어 있었다.

그렇기 때문에 void swap(char \*\*str1, char \*\*str2)와 같이 function definition의 윗부분을 작성했다. 그 다음에는 swap을 매개하기 위한 문자형 pointer변수 temp를 선언했다.

str1, str2는 모두 string의 주소이므로, main에 있는 두 변수의 string의 값을 맞교환 하기 위해서 정수 swap 함수를 짤 때처럼 dereference operator를 이용했다. 다음은 그 내용이다.

```
temp = *str1;
```

```
*str1 = *str2;
```

```
*str2 = temp;
```

### 3. 1. 3. array표현을 쓸 수 없는 이유와 swap 하기 위한 방안

main 함수에는 다음과 같이 pointer를 이용해 string을 선언했다.

```
char *str1 = "I am a boy";
```

```
char *str2 = "You are a girl";
```

그리고 위에서 구현한 다음 함수를 이용해 string의 값을 맞바꿨다.

```
swap(&str1, &str2);
```

그런데 아래와 같이 array를 이용해 string의 값을 선언하면, 위와 같은 함수를 사용할 수 없게 된다.

```
char str1[100] = "I am a boy";
```

```
char str2[100] = "You are a girl";
```

그 이유는 바로 str1, str2가 모두 상수여서 '주소'가 없기 때문이다. 따라서 array표현을 사용할 때는 &str1, &str2처럼 사용할 수 없다

이 경우에는 그냥 str1, str2값을 주면 사용할 수 있다. 그리고 strcpy함수를 이용해 두 string의 값을 맞바꾸면 된다. 3.1.1.과 달리 strcpy함수를 이용해도 되는 이유는 이것이 array로 선언되었기 때문이다. array로 선언되어서, strcpy를 다음과 같이 사용하면 두 array에 있는 값들이 서로 맞바뀐다. 즉, pointer와 달리 함수 내부를

```
char temp[128];
```

```
strcpy(temp, s1);
```

```
strcpy(s1, s2);
```

```
strcpy(s2, temp);
```

처럼 작성해도 문제 없다는 것이다.

### 3. 2. 문제 2 – strtok function

우선 이 함수의 입력은 다음과 같이 string 하나와 separator로 이루어진 string을 받기로 했다.

```
char *user_strtok(char *str, char *separator)
```

#### 3. 2. 1. 변수 선언

가장 먼저 현재 몇 번째 문자를 검사하고 있다는 것을 알기 위한 변수를 선언해야 했다. 그러기 위해 셈을 의미하는 count1이라는 정수형 static변수를 먼저 선언했다. 또한, string을 어디서부터 출력해야 할 지를 알려주기 위한 저장한다라는 의미의 save라는 정수형 변수가 필요했다.

또한, 문제 해결 방법에서 구상한 것처럼 다음 function call에 null을 return하기 위해 변수가 하나 필요했다. null을 return한다는 의미로 return\_null이라 명명했고, 0으로 초기화 했다. if(return\_null)꼴을 이용해 만약 0이 아니면 다음 번에 null을 return하도록 하기 위해서 말이다. 이 변수 역시 static으로 선언했다.

그리고 프로그램의 주축이 되는 `modified_str`이라는 변경된 string이라는 의미의 static 문자형 pointer type의 string을 선언했다. 이 string은 NULL이 아닌 string 값이 함수의 입력으로 올 시 그 string으로 값을 초기화시키고, NULL인 string이 오면 초기화 없이 이전에 사용한 string을 계속 사용한다. 이런 이유 때문에 static으로 선언했다.

마지막으로 separator들을 세기 위한 변수인 정수형 변수 `count2`를 선언했다.

### 3. 2. 2. 입력 받은 string이 NULL이 아닐 때

우선 입력 받은 string이 NULL이 아닌 어떤 string이면, `modified_str`을 입력받은 원래 string으로 초기화 했다. 그리고 token화 될 string의 '시작점'을 표시하는 변수인 `save`를 0으로 초기화 시켰다.

그 다음에 두 번의 for문을 이용했다. 첫 번째는 string의 모든 문자를 처음부터 끝까지 검사하는 것이고, 두 번째 for문은 한 문자가 separator 중 어느 하나와도 일치하는 지를 알아보는 것이었다. 즉, 다음과 같이 작성했다.

```
for (count1 = 0; modified_str[count1] != (char)0; ++count1)
    for (count2 = 0; separator[count2] != (char)0; ++count2)
```

이 때 `count1`은 어느 문자를 세고 있는지 알기 위해서 선언한 변수인데, 처음부터 시작하기 위해 0으로 선언했다. `count2` 역시 같은 이유로 0으로

선언했다.

두 번째 for문을 수행하는 중 만약 string중 한 문자와 separator의 한 문자가 일치 한다면, 우선 그 문자를 0으로 만들어야 했다. 그 후에 token의 시작점을 반환해야 했다. 그런데 만약 separator가 첫 문자에 왔다면, 굳이 string의 첫 문자를 반환해야 할 필요가 없게 되었다. 만약 반환해도 'w0'로만 이루어진 string이 되기 때문이다. 따라서 그런 경우에는 token화 될 string의 '시작점'을 다음으로 미루기 위해 save를 1증가 시켰다. 그리고 그 for문을 빠져나가 count1을 1증가 시켜야 하므로, break 문을 사용했다. (프로그램 코드를 참고해주세요.)

이 때 첫 문자가 separator인지 아닌 지 알기 위해 다음과 같이 if문을 사용해 string의 시작점과 검사하는 문자가 같은 지 여부를 판별했다.

```
if (save == count1) {  
    save++;  
    break;  
}
```

이렇게 하면 처음에 separator가 와도, 예를 들어 "::a"처럼 입력이 주어져도, 앞의 두 separator ::의 주소를 반환하지 않고 진행한다. 왜냐하면 첫 번째 ;에서는 save가 0, count1이 0이 되고, 두 번째 ;에서는 save가 1, count1이 1이 되어서 모두 break를 만나기 때문이다.

이 if문 다음에는 token의 첫 주소를 반환해야 했다. 그런데 string의 처음 주소가 아닌 save만큼의 다음 주소를 반환해야 하므로, 다음과 같이 반환했다.

```
return (modified_str + save);
```

여기까지는 모두 두 번째 for문 내부의 내용이다. 그런데 만약 string의 어느 문자도 separator가 아닐 수 있다. 그런 경우에는 return되지 않고 첫 번째 for문을 탈출할 것이다. 그렇게 되면 우선 이 string을 출력하고 다음 번에 null을 출력해야 하므로 우선 return\_null의 값을 하나 증가 시킨 후에 (modified\_str + save)을 반환했다.

### 3. 2. 3. 입력 받은 string이 NULL일 때

입력 받은 string이 NULL이라면 이전에 NULL이 아닌 string을 입력 받은 채 function call이 한번 이상 수행되었어야 한다. 그렇기 때문에, modified\_str은 어떤 값으로 초기화 되어 있을 것이고, 따라서 여기서 초기화 할 필요는 없다.

또 return\_null이 0인지 확인해야 한다. 만약 return\_null이 0이 아니면 이전 function call에서 '다음 번 function call에 NULL을 반환하고 싶다'라는 의미로 0이 아닌 값을 준 것이므로 NULL을 반환했다. 즉, 다음과 같다.

```
if (return_null)
```

```
return (char*)NULL;
```

이 다음에 저번 function call에서 0으로 초기화한 문자 다음부터 separator가 있는지 위에서 한 것과 같은 작업을 수행해야 했다. 어떤 문자를 검색 할 것인지에 대한 변수 count1의 값을 하나 증가시켜야 했다. 또, 그 값을 save에 assign함으로 인해 반환할 string의 시작점을 저번 function call에서 0으로 초기화한 문자 다음으로 하기로 했다. 즉, 다음과 같이 assign했다

```
save = ++count1;
```

이는 count1이 static변수라 이전의 값을 기억하고 있었기 때문에 가능했다.

이 다음에는 위에서 한 것과 같이 for문을 두 번 사용해서 문자를 검사했다. 그런데 여기서 첫 번째 for문의 초기화 과정에서 count1 = 0은 제외했다. count1을 0으로 assign 한 이유는 첫 번째 문자부터 검사하기 위해서인데 지금은 첫 문자부터 검사할 이유가 없기 때문이다.

나머지 부분은 위와 동일하게 작성했다.

```
if (save == count1) {  
    save++;  
    break;  
}
```

이 부분 역시 작성했는데, 이는 연속된 separator들을 무시하기 위함이었다. 만약 "aa;b"라는 string을 처음 function call때 입력 받았다고 해 보자. separator는 ";"이다. 그러면 처음 function call 때 첫 ;를 0을 만들고 return했을 것이다. 그 다음에 NULL을 입력으로 하고 function call을 하면, 두 번째 ;을 0으로 만들 것이다. 그 후에 저 과정에서 break를 만나기 때문에 통해 두 번째 ;의 주소를 반환하지 않게 된다. 이런 이유 때문에 연속된 separator를 무시할 수 있는 것이다.

이렇게 하면 user\_strtok라는 함수가 완성된다. 이 함수는 main에서 string하나와 separator 하나를 입력으로 받아서 call되고, 그 다음에 이 함수가 NULL을 반환할 때까지 NULL과 separator를 입력으로 받아서 call된다. 또, 각 function call 마다 출력(printf)되기 때문에 main에서 주어진 string의 모든 token들이 출력될 것이다.

## 4. Result

이번 실습에서는 다음 사항들을 확인하려 했다.

문제 1) 두 string의 swap이 정상적으로 행해지는지 여부

문제 2) strtok함수가 다음 사항들을 지키면서 token화 시키는지

(1) separator에 의해 잘 분리되는지

(2) 마지막 token이 잘 출력되는지

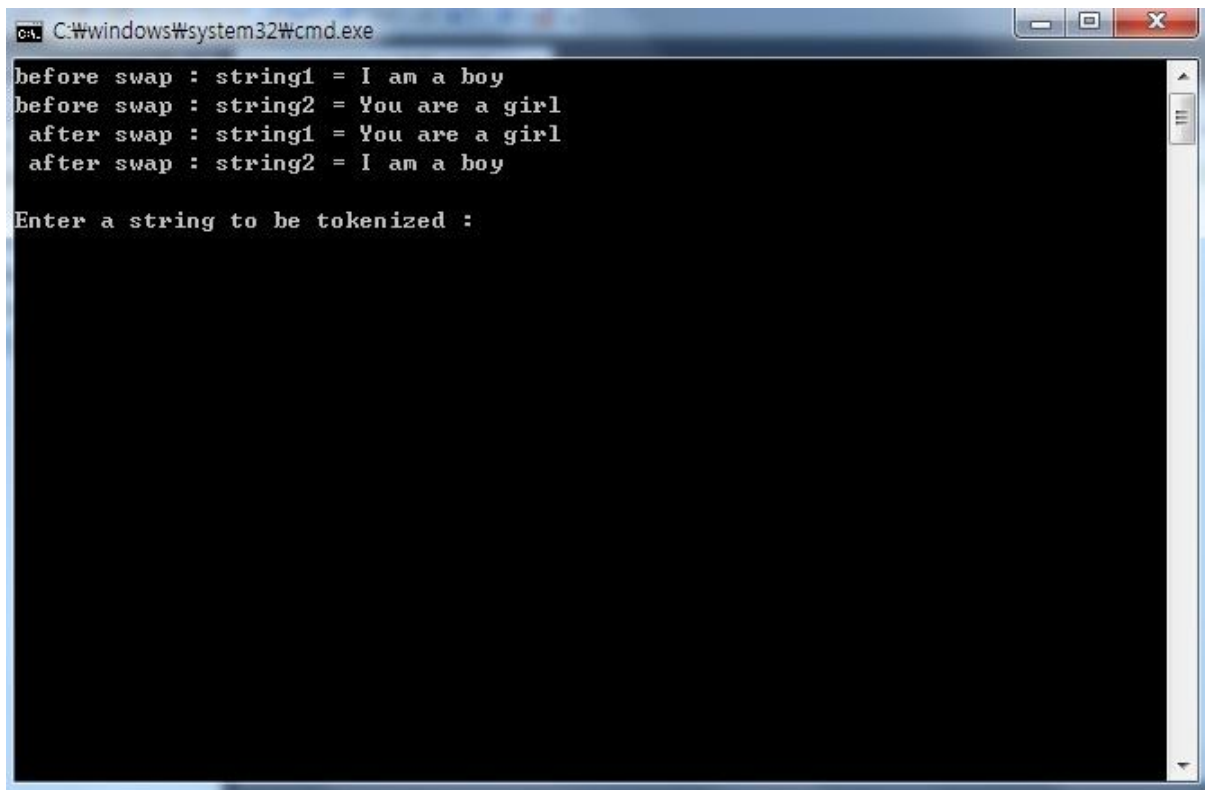


(3) 연속한 두 개 이상의 separator가 있을 때 출력하지 않는지

(4) 처음에 separator가 오면 출력하지 않는지

문제 1)

프로그램을 실행해 보았다.



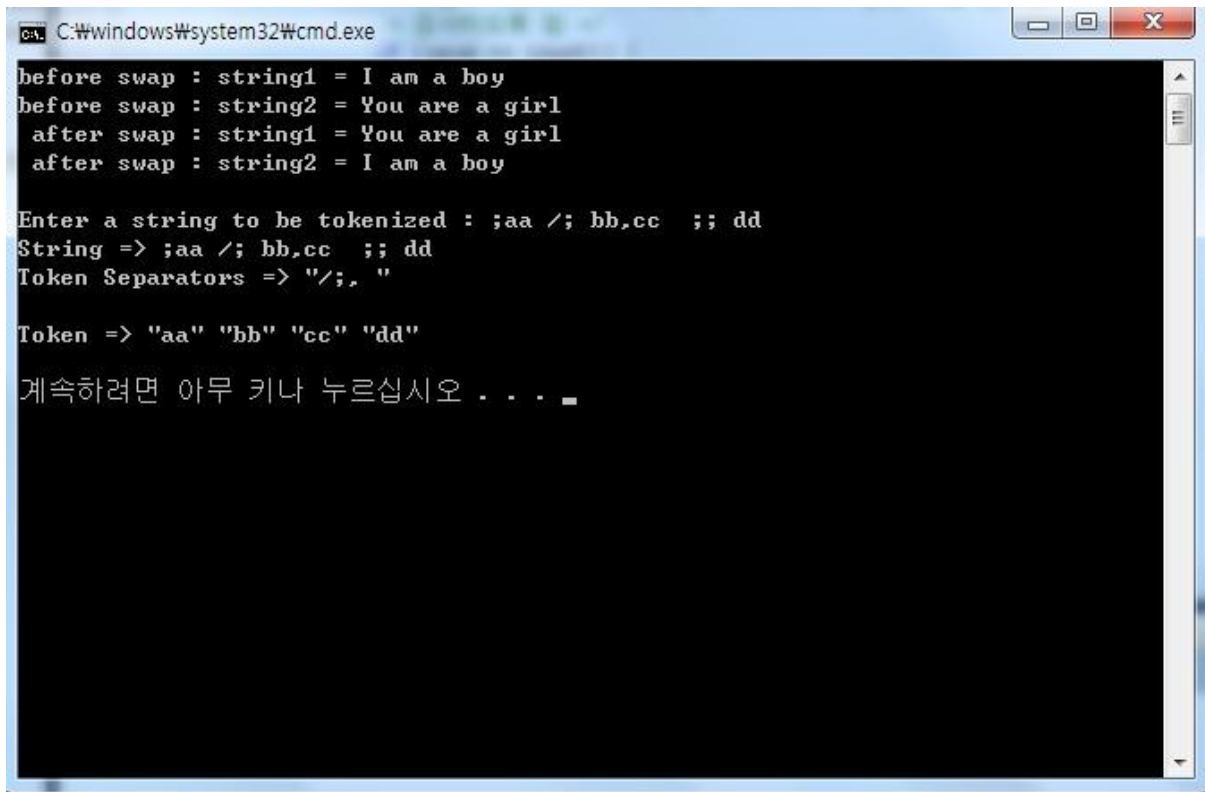
```
C:\windows\system32\cmd.exe
before swap : string1 = I am a boy
before swap : string2 = You are a girl
after swap : string1 = You are a girl
after swap : string2 = I am a boy

Enter a string to be tokenized :
```

두 string이 정상적으로 맞교환 되는 것을 확인할 수 있었다.

문제 2)

입력하는 곳에 ";aa /; bb,cc ;;dd"를 입력해 보았다.



```
C:\windows\system32\cmd.exe

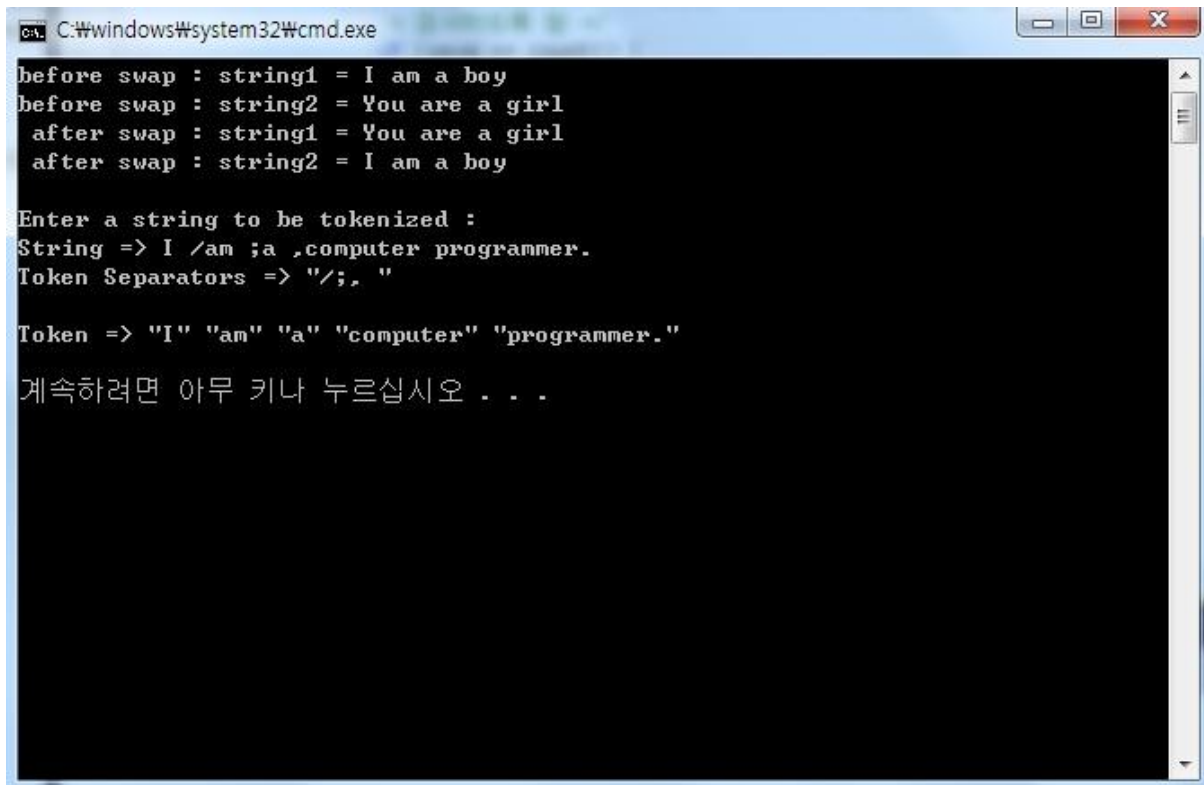
before swap : string1 = I am a boy
before swap : string2 = You are a girl
after swap : string1 = You are a girl
after swap : string2 = I am a boy

Enter a string to be tokenized : ;aa /; bb,cc ;; dd
String => ;aa /; bb,cc ;; dd
Token Separators => "/;, "
Token => "aa" "bb" "cc" "dd"

계속하려면 아무 키나 누르십시오 . . .
```

- (1) string이 separator들에 의해 "aa", "bb", "cc", "dd"로 잘 쪼개졌다.
- (2) 마지막 token "dd" 역시 잘 출력되었다.
- (3) 연속한 두 개 이상의 separator들을 출력하지 않았다.
- (4) 처음에 오는 separator도 무시하고 출력되지 않았다.

추가적으로 처음에 enter키만 쳐서 얻은 default값에 대해서도 확인해 보았다.



```
C:\windows\system32\cmd.exe
before swap : string1 = I am a boy
before swap : string2 = You are a girl
after swap : string1 = You are a girl
after swap : string2 = I am a boy

Enter a string to be tokenized :
String => I /am ;a ,computer programmer.
Token Separators => "/; , "
Token => "I" "am" "a" "computer" "programmer."
```

계속하려면 아무 키나 누르십시오 . . .

문제에서 요구한 것처럼 "I" "am" "a" "computer" "programmer."로 출력되었다.

모든 부분이 예상한 대로 출력되었다.

## 5. Conclusion & Evaluation

string에 대한 실습이 끝났다. 솔직히 말해서, 이번 실습은 꽤 어려웠다. string에 대해서 완벽하게 이해를 하지 못한 채 실습을 시작해서, 많이 헤맸다. 물론 이번 실습을 통해 지금은 string에 대해 어느 정도 이해를 했지만, 아직 잘 느낌이 오지 않는다. 아마 string을 이용해 다시 프로그램을 짤다면 짤 수야 있겠지만 약간 헤맬 것 같다.

그럼에도 불구하고 이번 실습에서는 string에 대해 많은 것을 배울 수 있었다. 가령 string, array, pointer의 관계가 어떤 지에 대해 잘 알아볼 수 있었다. 또 실습 첫 번째 문제 swap을 풀면서, 문자형 pointer type을 그냥 하나의 string으로 생각한다는 점을 알게 되었는데 이렇게 생각해 보니 string에 대해 더 이해하기 쉬워졌다.

실습 두 번째 문제인 user\_strtok를 구현할 때는 정말 많이 함수를 고쳤다. 동작 원리도 쉽지 않았지만, 프로그램에서 생각해야 할 부분이 너무 많아서 수십 번은 수정한 것 같다. 그래도 완성하고 나니 함수의 동작 원리가 생각보다 어렵지 않게 이해가 되면서, string을 array처럼 이용하는 것이 잘 이해가 되었다. 뿌듯하다.

이번 실습에서는 string에 대한 것도 배웠지만, 프로그램 자체를 보는 능력도 어느 정도 높아진 것 같다. 지금까지 배운 것을 바탕으로 프로그램을 전체적인 측면에서 바라 볼 수 있어서, 프로그램의 흐름이 한 눈에 보이는 것 같다.

## 6. 참고 문헌

[1] 민형복, program template, p8.c.

[2] 프로그래밍 기초와 실습 사이트, <http://class.icc.skku.ac.kr/~min/C/>, 보고서 작성  
요령, 2014 년 11 월.

[3] Al Kelly, Ira Pohl, *C by Dissection: The Essentials of C Programming, Fourth Edition*, Pearson, p.338 ~ p.345.

[4] 민형복, practice8.pdf.