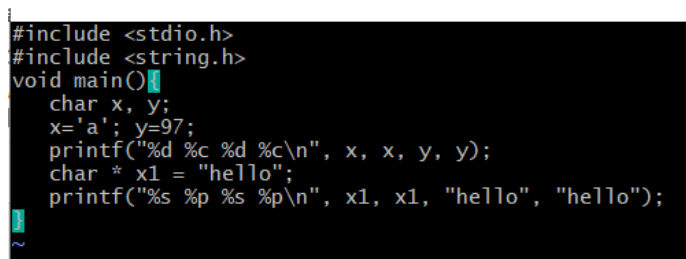


12201922 이규민


## 5. Homework

1) [A char constant is an ascii number. A string constant is an address where it is stored in the string area.] Explain the result for following code.

```
#include <stdio.h>
#include <string.h> // you need this header file for string functions
void main(){
    char x, y;
    x='a'; y=97;
    printf("%d %c %d %c\n", x, x, y, y);
    char * x1 = "hello";
    printf("%s %p %s %p\n", x1, x1, "hello", "hello"); // use %p for address
}
```



```
#include <stdio.h>
#include <string.h>
void main()
{
    char x, y;
    x='a'; y=97;
    printf("%d %c %d %c\n", x, x, y, y);
    char * x1 = "hello";
    printf("%s %p %s %p\n", x1, x1, "hello", "hello");
}
```



```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./char1
97 a 97 a
hello 0x10040300d hello 0x10040300d
```

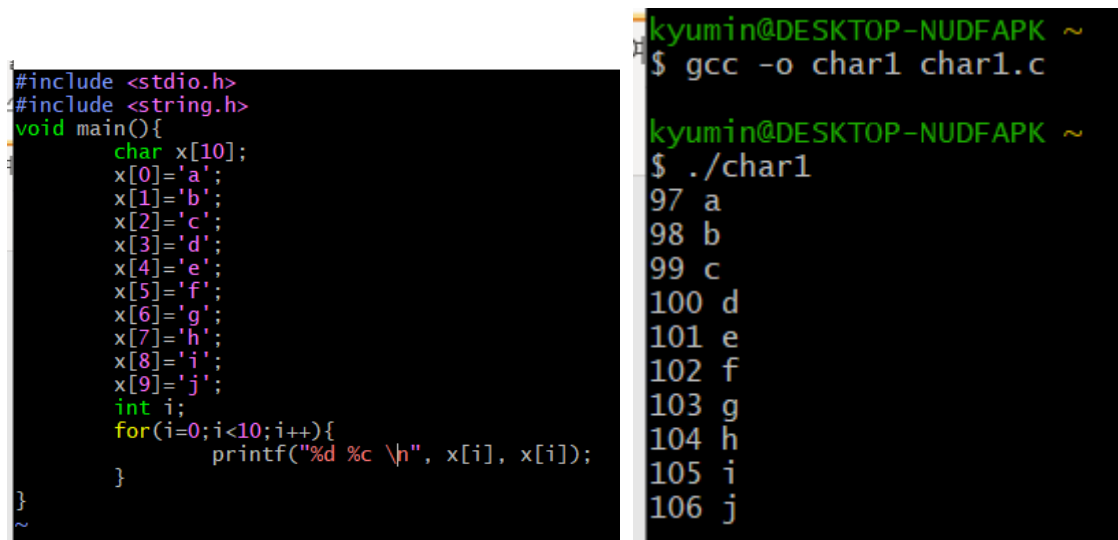
캐릭터 상수는 아스키코드를 저장하기 때문에 따옴표를 사용하거나 아스키코드를 사용해 문자를 저장할 수 있다. 'a'의 아스키 코드는 97이므로 위 코드의 x와 y는 같은 내용이 저장된 것을 볼 수 있다.

스트링 상수는 스트링이 저장되어있는 주소를 저장한다. X1의 경우 쌍따옴표를 사용하여 스트링의 주소를 저장했다. 위 결과 사진을 보면 x1의 내용과 주소는 스트링 "hello"의 내용과 주소와 동일한 것을 볼 수 있다.

Printf 에서 %d는 아스키 코드를 그대로 출력하고, %c는 문자로 변환하여 출력한다. %s는 스트링을 출력하고, %p는 스트링이 저장된 주소를 출력한다.

2) [A char constant is an ascii number] Try following code and explain the result.

```
char x[10];    // x is a character array with 10 rooms
x[0]='a'; x[1]='b'; x[2]='c'; x[3]='d'; x[4]='e';
x[5]='f'; x[6]='g'; x[7]='h'; x[8]='i'; x[9]='j';
int i;
for(i=0;i<10;i++){
    printf("%d %c \n", x[i], x[i]); // print each character with its ascii number
}
```



The image shows two side-by-side screenshots of a terminal window. The left screenshot displays the C code from the previous block, with syntax highlighting. The right screenshot shows the execution of the program, with the output of the printf statements.

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./char1
97 a
98 b
99 c
100 d
101 e
102 f
103 g
104 h
105 i
106 j
```

X 배열안에 따옴표를 이용해서 문자의 아스키코드를 하나씩 저장해주었다. X를 아스키코드와 문자로 출력해보면 그 결과는 위 사진과 같다.

3) Try below. Compare the result with that of Problem 2).

```
char x[10];    // x is a character array with 10 rooms
int i;
for(i=0;i<10;i++){
    x[i]=i+ 97;
}
for(i=0;i<10;i++){
    printf("%d %c \n", x[i], x[i]); // print each character with its ascii number
}
```

```
#include <stdio.h>
#include <string.h>
void main(){
    char x[10];    // x is a character array with 10 rooms
    int i;
    for(i=0;i<10;i++){
        x[i]=i+97;
    }
    for(i=0;i<10;i++){
        printf("%d %c \n", x[i], x[i]); // print each character with its
    }
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./char1
97 a
98 b
99 c
100 d
101 e
102 f
103 g
104 h
105 i
106 j
```

2번 문제에서는 따옴표를 이용해 문자로 아스키코드를 저장했지만, 3번 문제에서는 아스키 코드(숫자)로 저장을 했다. 사실상 동일하게 아스키코드로 저장했다고도 볼 수 있다.

4) Declare a character array with 128 rooms. Store 0 to 127 in this array and print the corresponding character for each ascii code in the array. Find ASCII table in the Internet and confirm the results.

```
char x[128];
for(i=0;i<128;i++){
    x[i]=i;
}
for(i=0;i<128;i++){
    printf("%d%c%dWn", x[i], x[i], x[i]);
}
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ gcc -o char1 char1.c  
kyumin@DESKTOP-NUDFAPK ~  
$ ./char1  
00  
11  
22  
33  
44  
55  
66  
77  
8  
9 9  
10  
10  
11  
11  
12  
12  
12
```

83S83  
84T84  
85U85  
86V86  
87W87  
88X88  
89Y89  
90Z90  
91\91  
92\92  
93\93  
94A94  
95\_95  
96\_96  
97a97  
98b98  
99c99  
100d100  
101e101  
102f102  
103g103  
104h104  
105i105  
106j106

	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400																																																																																																																																																																																											
0	0a00	NUL	32	0x20	SP	64	0a40																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				

0번부터 127번까지의 아스키코드를 출력해보았고, 아스키코드(10진수)와 문자를 출력해보았다. 영문자, 특수문자, 공백문자 등 아스키코드표와 동일하게 출력되는 것을 볼 수 있다.

5) [strlen] Read a string and display its length.

Enter a string

hello

The length is 5

```
#include <stdio.h>
#include <string.h>
void main(){
    char x[128];
    scanf("%s",x);
    printf("%s\n",x);
    printf("The length is %d",strlen(x));
}
```

```
kyumin@DESKTOP-NUDFAPK ~  
$ gcc -o char1 char1.c  
  
kyumin@DESKTOP-NUDFAPK ~  
$ ./char1  
hello  
hello  
The length is 5
```

scanf를 통해 입력 받은 내용을 배열 x가 가리키는 공간에 저장할 수 있도록 만들었다. Strlen 함수를 이용해서 x의 길이를 출력할 수 있다.

6) [A string is a char array ending with 0] Read a string and display each character in different lines.

Enter a string

hello

h

e

l

l

o

```
#include <stdio.h>
#include <string.h>
void main(){
    char x[128];
    scanf("%s",x);
    int i;
    for(i=0;i<strlen(x);i++){
        printf("%c \n",x[i]);
    }
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
hello
h
e
l
l
o
```

scanf를 통해 입력 받은 내용을 배열 x가 가리키는 공간에 저장해줬고, for문을 스트링의 길이만큼 돌려서 한 글자씩 출력해줬다.

6-1) [A string is a char array ending with 0] Try below and explain the result. Use g++ to compile.

```
char x[10];
strcpy(x, "hello");
strcpy(x, "hi");
for(int i=0;i<10;i++){
    printf("%d ", x[i]);
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o char1 char1.c
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
104 105 0 108 111 0 0 0 0 0
```

문자를 아스키코드로 출력해주는 코드다. 처음에는 “hello”라는 문자를 저장해줬기 때문에 배열 x에는 104 101 108 108 111 0으로 저장되어있었으나 “hi”라는 문자를 저장해줬기 때문에 104 105 0 108 111 0으로 저장되었다. 메시지 입력 끝은 0이기 때문에 104 105 뒤에도 0으로 바뀐 것을 알 수 있다.

7) [strlen, strcmp] Write a program that keeps reading a string, displaying its length, and checking whether it is "hello". If the input string is "hello", the program stops.

```
Enter a string
hi
You entered hi. length=2
No it is not hello
Enter a string
hello
You entered hello. length=5
Yes it is hello. Bye.
```

```
#include <stdio.h>
#include <string.h>
int main(){
    char x[10];
    for(;;){
        printf("Enter a string\n");
        scanf("%s",x);
        printf("You entered %s. length=%d\n",x,strlen(x));
        if(strcmp(x,"hello")!=0) break;
        else printf("No it is not hello\n");
    }
    printf("Yes it is hello. Bye\n");
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o char1 char1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Enter a string
hi
You entered hi. length=2
No it is not hello
Enter a string
how
You entered how. length=3
No it is not hello
Enter a string
hello
You entered hello. length=5
Yes it is hello. Bye
```

입력 받은 문자가 hello 라면 멈추고, 아니라면 계속 물어보는 코드를 만들어야한다. 무한으로 도는 for 문 안에서 scanf를 통해 입력을 받고, strlen을 통해 스트링의 길이를 출력하도록 만들었다. for문 안에서 입력 받은 스트링이 "hello"인지 확인하고, 맞다면 for 문을 나가도록 만들었다. "hello"인지 확인을 위해 strcmp 함수를 사용했다. 이후 위 사진처럼 정상적인 결과가 나오는 것을 볼 수 있다.

8) [strcpy] Read a string and copy it to three other string variables and change the first letter of them to 'a', 'b', and 'c', respectively, and display them.

Enter a string

hello

After copying and changing the first letter

aello bello cello

```
#include <stdio.h>
#include <string.h>
int main(){
    char x[10];
    char y[10];
    char z[10];
    printf("Enter a string\n");
    scanf("%s",x);
    strcpy(y,x);
    strcpy(z,y);
    x[0]=97;
    y[0]=98;
    z[0]=99;
    printf("%s %s %s \n",x,y,z);
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o char1 char1.c

kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Enter a string
hello
aello bello cello
```

배열 x, y, z를 만든다. Scanf를 통해 입력받은 스트링을 x의 주소에 저장한다. 그리고 strcpy로 x를 y와 z에 복사한다. 그러면 x, y, z는 모두 “hello”라는 내용을 가진다. X[0]=97로 string의 첫 글자를 ‘a’의 아스키코드로 저장해줬다. Y와 z에도 동일하게 진행해줬고, 정상적으로 결과가 나온 것을 볼 수 있다.

9) [string constant] A string constant such as "hello" is an address. Explain the result of following code.

```
char *x, *y, *z;
x="hello"; y="hi"; z="bye";
printf("%s %s %s\n", x, y, z);
printf("%p %p %p\n", x, y, z);
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
hello hi bye
0x100403000 0x100403006 0x100403009
```

위 코드에서는 x,y,z 포인터를 선언하였다. x="hello"는 “hello”라는 스트링의 주소 값을 x에 저장하라는 의미다. 그렇게 스트링의 주소를 x,y,z에 각각 저장해줬고, printf의 %s와 %p를 통해 스트링과 스트링의 주소를 출력해줬다.

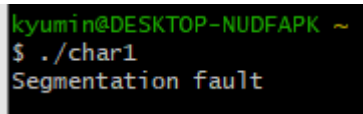
10) [string constant is an address] Try below and explain why we have an error.

```
char x[20];
strcpy(x, "hello"); // this is ok
x="hello"; // this is an error. "hello" is an address and we can't store address in
           // x which is not a pointer variable
```

배열 x에 hello 라는 스트링을 저장하는 strcpy는 사용할 수 있다. 그러나 “hello”는 스트링의 주소이므로 x=“hello”를 사용할 수 없다.

11) [You need memory space for strcpy] Try below and explain why we have an error. How can you fix it?

```
char *y;
y="hello1"; // this is ok
strcpy(y, "hello2"); // error because y has no space for "hello2"
```

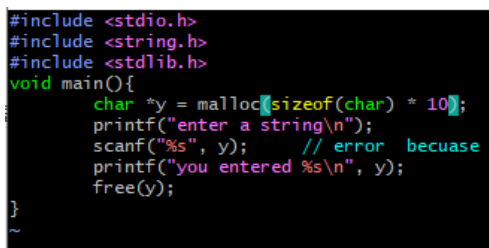


```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Segmentation fault
```

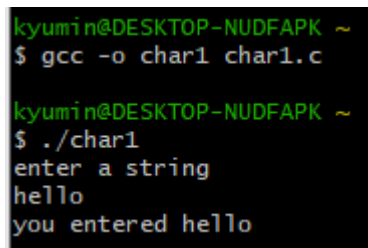
포인터는 주소를 가리키는 것이므로 y 안에 hello2 라는 내용을 넣을 수 없다. 그러므로 strcpy 사용 시 에러가 발생한다.  
Y=“hello2”라고 수정하면 문제없다.

12) [You need memory space for scanf] Try below and explain why you have an error. Fix it.

```
char *y;
printf("enter a string\n");
scanf("%s", y); // error because y has no space for a string
printf("you entered %s\n", y);
```



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main(){
    char *y = malloc(sizeof(char) * 10);
    printf("enter a string\n");
    scanf("%s", y); // error because
    printf("you entered %s\n", y);
    free(y);
}
```



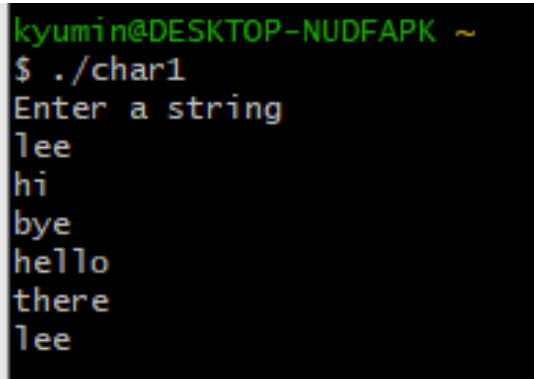
```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
enter a string
hello
you entered hello
```

y는 포인터이므로 주소를 입력 받아야하지만, scanf를 통해 스트링을 입력받으면 에러가 발생한다. 입력 값을 포인터에 저장하기 위해서 메모리 공간을 따로 마련해야한다. Malloc 함수로 메모리를 할당한 후에는 정상적으로 작동하는 것을 볼 수 있다.



13) [char pointer array] Define a character pointer array and store/display strings as below.

```
char * x[10];
x[0]="hi"; x[1]="bye"; x[2]="hello"; // store addresses of string constants
x[3]=new char[50];
strcpy(x[3], "there"); // copy a string
x[4]=new char[50];
printf("Enter a string\n");
scanf("%s", x[4]); // read a strign from the user
for(int i=0;i<5;i+ +)
    printf("%s \n", x[i]);
```



```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Enter a string
lee
hi
bye
hello
there
lee
```

X를 포인터 배열로 선언한 후 x[0]="hi"로 스트링 주소를 저장할 수 있다. 포인터는 스트링을 저장할 수 없기 때문에 strcpy와 scanf 사용을 할 수 없다. 하지만 new 함수로 메모리 공간을 마련한다면 사용할 수 있다. 위 사진처럼 정상적으로 작동하는 것을 볼 수 있다.

14) [char pointer array, strcmp, new] Write a program that keeps reading strings and store them in a character pointer array. It stops when the user enters "end" and displays all strings entered so far. Use "new" to allocate memory and use g++ to compile.

```
Enter a string
hi
Enter a string
aaa
Enter a string
bbb
Enter a string
end
Strings entered so far are
hi aaa bbb
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char * x[10];
    int i=0;
    for(;;i++){
        printf("Enter a string\n");
        x[i]=new char[50];
        scanf("%s", x[i]);
        if(strcmp(x[i],"end")==0){
            break;
        }
    }
    for(int j=0;j<i;j++)
        printf("%s ", x[j]);
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Enter a string
hi
Enter a string
aaa
Enter a string
ccc
Enter a string
bvb
Enter a string
end
hi aaa ccc bvb
```

무한 for문 안에서 new char를 통해 매번 포인터 배열 x에 메모리를 할당해줬다. 그리고 scanf를 통해 매번 입력을 받고, strcmp를 통해 입력받은 메시지가 "end"인지 확인을 해준다. 만약 "end"라면 무한 for문을 멈추고, x배열에 저장되어있던 스트링을 모두 출력하게만 들었다. 위 사진 처럼 정상 작동하는 것을 볼 수 있다.

15) [gets, fgets] Read the same sentence with gets() and fgets() and explain the difference. (Ignore warning for gets. It is a security warning because gets can cause security problem.)

```
char x[100];
printf("enter a sentence\n");
gets(x);
int slen=strlen(x);
printf("sentence length after gets:%d\n", slen);
for(i=0;i<slen;i++){
    printf("%x ", x[i]);
}

printf("\nenter the same sentence\n");
fgets(x, 99, stdin); // read max 99 char's.
slen=strlen(x);
printf("sentence length after fgets:%d\n", slen);
for(i=0;i<slen;i++){
    printf("%x ", x[i]);
}
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main(){
    char x[100];
    printf("enter a sentence\n");
    gets(x);
    int slen=strlen(x);
    printf("sentence length after gets:%d\n", slen);
    for(i=0;i<slen;i++){
        printf("%x ", x[i]);
    }
    printf("\nenter the same sentence\n");
    fgets(x, 99, stdin); // read max 99 char's.
    slen=strlen(x);
    printf("sentence length after fgets:%d\n", slen);
    for(i=0;i<slen;i++){
        printf("%x ", x[i]);
    }
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
enter a sentence
hello
sentence length after gets:5
68 65 6c 6c 6f
enter the same sentence
hello
sentence length after fgets:6
68 65 6c 6c 6f a
kyumin@DESKTOP-NUDFAPK ~
```

스트링 입력을 받으면 gets는 마지막 \n를 \0으로 바꿔주기 때문에 버퍼에는 \n이 없다. 하지만 Fgets는 \n과 \0을 모두 저장해준다.

위 코드와 같이 스트링의 길이를 확인해보면 fgets는 길이 값이 1만큼 큰 것을 볼 수 있다. Printf로 문자를 16진수 형태로 출력해보면 fgets로 입력 받은 스트링의 경우 a인 것을 볼 수 있다. 아스키코드표를 비교해보면 a는 LF(개행문자)임을 알 수 있다.

16) [strtok] Use strtok to extract words from a sentence and store them in an array. Display the number of words as below. Note that you need to copy the sentence to another string variable before doing strtok because strtok will destroy the original sentence.

algorithm:

read a line

tokenize

display tokens

Enter a sentence

aa bcd e e ff aa bcd bcd hijk lmn al bcd

You entered aa bcd e e ff aa bcd bcd hijk lmn al bcd

There were 12 words:

aa

bcd

e

e

ff

aa

bcd

bcd

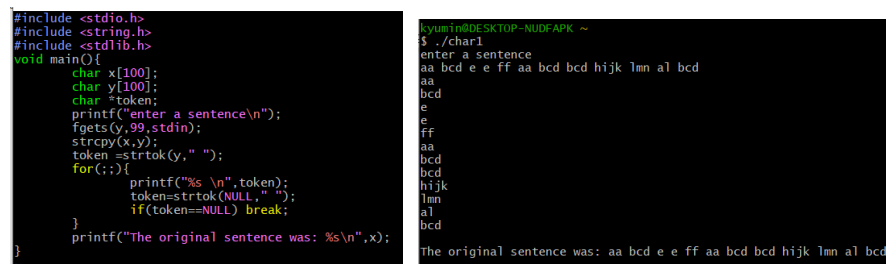
hijk

lmn

al

bcd

The original sentence was: aa bcd e e ff aa bcd bcd hijk lmn al bcd



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main() {
    char x[100];
    char y[100];
    char *token;
    printf("enter a sentence\n");
    fgets(y, 99, stdin);
    strcpy(x, y);
    token = strtok(y, " ");
    for(;;) {
        printf("%s\n", token);
        token = strtok(NULL, " ");
        if(token == NULL) break;
    }
    printf("The original sentence was: %s\n", x);
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
enter a sentence
aa bcd e e ff aa bcd bcd hijk lmn al bcd
aa
bcd
e
e
ff
aa
bcd
bcd
hijk
lmn
al
bcd
The original sentence was: aa bcd e e ff aa bcd bcd hijk lmn al bcd
```

strcpy 로 y 를 x 에 복사해줬다. 그리고 strtok 를 이용하여 y 의 스트링에서 공백을 찾아 단어를 나눴고, for 문으로 그 단어들을 출력해줬다. 그리고 마지막에는 배열 x 의 내용을 출력해줬다.

여기서 처음에 y 를 x 에 복사해준 이유는 strtok 때문이다. Strtok 사용 시 공백인 글자는 W0(문자 종료)로 바뀌기 때문에 단어를 나누는 명령이 끝나고 y 의 스트링을 출력해보면 첫 단어 aa 만 출력되기 때문이다.

17) [char pointer array, new, strcmp] Write a program that keeps reading a name and stores it in a character pointer array until the user enters bye. The program should display all names after it sees "bye".

```
Enter a name
kim han kook
Enter a name
park dong il
Enter a name
hong gil dong
bye
There were 3 names.
The names were
kim han kook
park dong il
hong gil dong
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(){
    char * x[100];
    int i = 0, j;
    for (; i++) {
        x[i] = new char[30];
        printf("Enter a name\n");
        fgets(x[i], 30, stdin);
        x[i][strlen(x[i]) - 1] = '\0';
        if (strcmp(x[i], "bye") == 0) break;
    }

    printf("There were %d names.\n\nThe names were\n", i);

    for (j = 0; j < i; j++) {
        printf("%s\n", x[j]);
    }
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Enter a name
kim han kook
Enter a name
park dong il
Enter a name
hong gil dong
Enter a name
bye
There were 3 names.
The names were
kim han kook
park dong il
hong gil dong
```

여러 개의 이름을 저장할 수 있도록 포인터 배열을 사용하였다. 무한 for 문 안에서 입력을 받아야하는데, 포인터에는 문자열을 저장할 수 없으므로 new를 사용하여 메모리를 할당하였다. 이름 입력을 받기 위해 명령어를 사용해야하는데 scanf의 경우 띄어쓰기가 입력되는 경우 첫번째 단어만 입력을 받기 때문에 좋은 선택지는 아니다. 그래서 fgets를 사용하였으나 줄바꿈이 같이 들어오기 때문에 마지막 Wn을 W0으로 바꿔주었다. 그리고 strcmp를 사용하여 bye라는 메시지가 입력되었다면 무한 for문을 중단하고 나가도록 만들었다. 이후 입력되었던 모든 이름들이 출력된다.

18) [There is a hidden 0 at the end of a string] Try below and explain why it behaves strange. How can you fix it?

```
int x3;
char x2[12];
char x1[12];
x1[0]=33;
x3=44;
strcpy(x2,"abcdefghijkl");
printf("%p %p %p %d %d %s", x1, x2, &x3, x1[0], x3, x2);
```

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c
char1.c: In function 'main':
char1.c:10:9: warning: '__builtin_memcpy' writing 13 bytes
12 overflows the destination [-Wstringop-overflow=]
   10 |         strcpy(x2,"abcdefghijkl");
      |         ~~~~~~
char1.c:6:14: note: destination object 'x2' of size 12
    6 |         char x2[12];
      |         ~
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
0x7ffffcc24 0x7ffffcc30 0x7ffffcc3c 33 0 abcdefghijkl
```

```
kyumin@DESKTOP-NUDFAPK ~
$ gcc -o char1 char1.c
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
0x7ffffcc23 0x7ffffcc2f 0x7ffffcc3c 33 44 abcdefghijkl
```

printf에서 5번째 즉 x3의 값이 44가 아닌 0이 출력된다는 문제가 있다. X2가 받을 수 있는 문자 수는 12개인데, strcpy를 통해 12개의 문자를 입력 받았고 문자 종료를 의미하는 0이 붙기 때문에 오버플로우가 발생한다. 그 0이 x3 주소의 값에 들어가기 때문에 이 문제가 발생한다.

고치는 방법은 여러가지지만 가장 쉬운 방법은 char x2[12]를 char x2[13]으로 바꾸는 방법이다. 그러면 오버플로우가 발생하지 않아 x3의 값은 그대로 유지된다.

19) [You need memory space for strcpy] What is wrong with the following program? How can you fix it?

```
int main(){
    char * strarr[10]={NULL};
    strarr[0]="hello";
    strcpy(strarr[1],"bye");
    printf("%s %s\n", strarr[0], strarr[1]);
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o char1 char1.c
char1.c: In function 'int main()':
char1.c:6:19: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
     6 |         strarr[0]="hello";
       |         ~~~~~
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
Segmentation fault
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(){
    char * strarr[10]={NULL};
    strarr[1]=new char[8];
    strarr[0]="hello";
    strcpy(strarr[1],"bye");
    printf("%s %s\n", strarr[0], strarr[1]);
    return 0;
}
```

```
kyumin@DESKTOP-NUDFAPK ~
$ g++ -o char1 char1.c
char1.c: In function 'int main()':
char1.c:7:19: warning: ISO C++ forbids co
-Wwrite-strings]
     7 |         strarr[0]="hello";
       |         ~~~~~
kyumin@DESKTOP-NUDFAPK ~
$ ./char1
hello bye
```

Char \* strarr[10]은 배열을 사용하긴했지만 여러 개의 포인터를 만든 것일 뿐이다. 포인터는 내용이 아닌 주소를 저장하는데, strcpy 는 내용을 저장하는 함수이므로 오류가 발생한다. 이 문제를 해결하기 위해서는 new 를 사용하여 포인터에 메모리를 할당하는 방법과 strcpy 가 아닌 strarr[1]="bye"를 사용하는 방법이 있다. 후자는 쉬운 방법이니 new 를 사용하여 해결해보았다. 정상적으로 출력되는 것을 볼 수 있다.

20) [char pointer array, strtok, strcmp] Write a program that reads a long sentence and displays the frequency of each word. It also prints the word that has the maximum frequency. Your main function should look like below. Implement each function.

```
int main(){
    // step 1. read a sentence into buf
    char buf[100];
    get_sentence(buf);

    // step 2. extract words and store in tokens array. return num of tokens
    int ntok;
    char *tokens[50];
    ntok = tokenize(buf, tokens);

    // step 3. show tokens
    display_tokens(tokens, ntok);

    // step 4. compute unique tokens into unique_tokens array
    char *unique_tokens[50];
    int nuniqtok;
    nuniqtok=compute_unique_tokens(tokens, ntok, unique_tokens);
    printf("unique tokens are ");
    display_tokens(unique_tokens, nuniqtok);

    // step 5. compute freq of each unique token into freq array
    int freq[50];
    compute_freq(tokens, ntok, unique_tokens, nuniqtok, freq);

    // step 6. show frequencies of each token
    show_freq(freq, unique_tokens, nuniqtok);

    // step 7. show max freq word
    show_max_freq_word(unique_tokens, nuniqtok, freq);
}
```

Enter a sentence

aa bcd e e ff aa bcd bcd hijk lmn al bcd



You entered aa bcd e e ff aa bcd bcd hijk lmn al bcd

There were 12 words: aa bcd e e ff aa bcd bcd hijk lmn al bcd

Frequencies: aa 2 bcd 4 e 2 ff 1 hijk 1 lmm 1 al 1

The word with the max freq: bcd

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(){
    // step 1. read a sentence into buf
    char buf[100];
    printf("Enter a sentence\n");
    fgets(buf,100,stdin);
    buf[strlen(buf)-1]=0;

    // step 2. extract words and store in tokens array. return num of tokens
    int ntok;
    int i,j;
    char *tokens[50];
    tokens[0]=strtok(buf, " ");
    for(i=0;;i++){
        tokens[i+1]=strtok(NULL, " ");
        if(tokens[i+1]==NULL) break;
    }
    ntok=i+1;

    // step 3. show tokens
    printf("You entered ");
    for(i=0;i<ntok;i++) printf("%s ",tokens[i]);
    printf("\nThere were %d words: ",ntok);
    for(i=0;i<ntok;i++) printf("%s ",tokens[i]);

    // step 4. compute unique tokens into unique_tokens array
    char *unique_tokens[50];
    int nuniqtok,max=0;
    int freq[50];

    for(i=0;i<ntok;i++) {
        unique_tokens[i]=tokens[i]; //배열 복제
        freq[i]=1; //단어 수의 초기 값을 1로 고정
    }
    for(i=0;i<ntok;i++){
        //현재 위치가 공백이라면 다음 인덱스를 확인
        if(unique_tokens[i]==""){
            for(j=i+1;j<ntok;j++){
                if(unique_tokens[j]!=""){
                    unique_tokens[i]=unique_tokens[j];
                    unique_tokens[j]="";
                    break;
                }
            }
        }
    }
}
```

```

//현재 위치가 공백이라면 다음 인덱스를 확인
if(unique_tokens[i]==""){
    for(j=i+1;j<ntok;j++){
        if(unique_tokens[j]!=""){
            unique_tokens[i]=unique_tokens[j];
            unique_tokens[j]="";
            break;
        }
    }
    //여전히 공백이라면 마지막까지 공백이니 for i문을 멈춰야함.
    if(unique_tokens[i]=="") break;
}

//현재 위치와 동일한 내용이 있다면 내용을 공백으로 바꾸고 카운트
for(j=i+1;j<ntok;j++){
    char A[50],B[50];
    strcpy(A,unique_tokens[i]);
    strcpy(B,unique_tokens[j]);

    if(strcmp(A,B)==0){
        freq[i]++; //step 5
        unique_tokens[j]="";
    }
}

//for i문을 나온다면 i가 유닛토큰의 개수임
nuniqtok=i;

//step 6
//유닛토큰을 출력하고, j는 유닛이 최대일 때의 배열 번호임
printf("\nFrequencies: ");
for(i=0,j=0;i<nuniqtok;i++){
    printf("%s %d ",unique_tokens[i],freq[i]);
    if(max<freq[i]){
        max=freq[i];
        j=i;
    }
}

//step 7
printf("\nThe word with the max freq: %s",unique_tokens[j]);

return 0;
}
~

```

42,2-16

Bot

Step1. 띄어쓰기를 포함한 모든 문자를 받기 위해 fgets를 사용했고, 줄바꿈을 제거하기 위해 Wn을 W0으로 바꿨다.

Step2. 입력받은 문자열을 strtok를 사용하여 단어별로 나눴다. 그리고 단어의 개수를 세기 위해서 ntok=i+1이라고해줬다. 그리고 for문 안에서 tokens[i+1]=strtok(NULL," ");처럼 i+1을 사용해야 ntok=i+1 한 문장으로 단어를 셀 수 있다.

Step3. Printf를 사용하여 token 포인터들이 가리키는 내용을 출력해줬다.

Step4. Step5. 내가 사용한 방식은 이와 같다. 우선 토큰 배열을 유닛 토큰배열에 동일한 내용으로 복사해준다. 이후 첫번째 위치의 단어가 다른 위치에도 있는지 확인한다. 두번째 토큰부터 보는데 만약 동일한 단어가 있다면 그 토큰은 "" 공백으로 내용을 바꿔주고 그 단어의 개수인 freq[i]를 카운트해준다. 그렇게 마지막까지 동일한 작업을 해준다. 여기까지가 for문에서 i가 0일 때 작업이 완료된 것이다. 이후 두번째 토큰과 동일한 단어가 있는지 찾

고 공백으로 바꾸고 이 과정을 끝까지 반복해준다. 여기서 위 작업을 반복 하다 보면 동일한 단어들은 모두 제거가 되어 공백 밖에 남지 않으면 그 때의 I값이 유니 토큰의 개수이다. 이 작업을 통해 유니 토큰과, 그 단어의 개수를 알 수 있다.

Step6. Printf를 통해 유니 토큰을 출력한다. 출력함과 동시에 freq값이 최대일 때를 찾고, 그게 몇 번째 배열인지 확인하고 j 변수에 저장해준다.

Step7. 위 단계에서 찾은 j를 사용하면 가장 많이 입력된 단어를 출력할 수 있다.