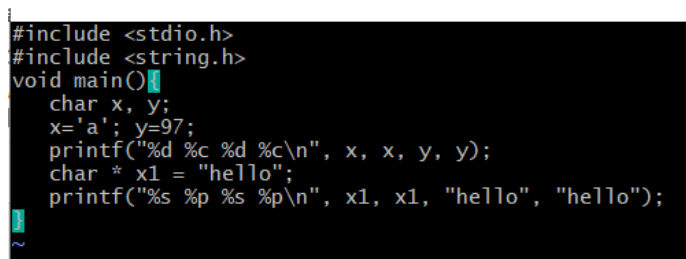


12201922 이규민


## 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
```

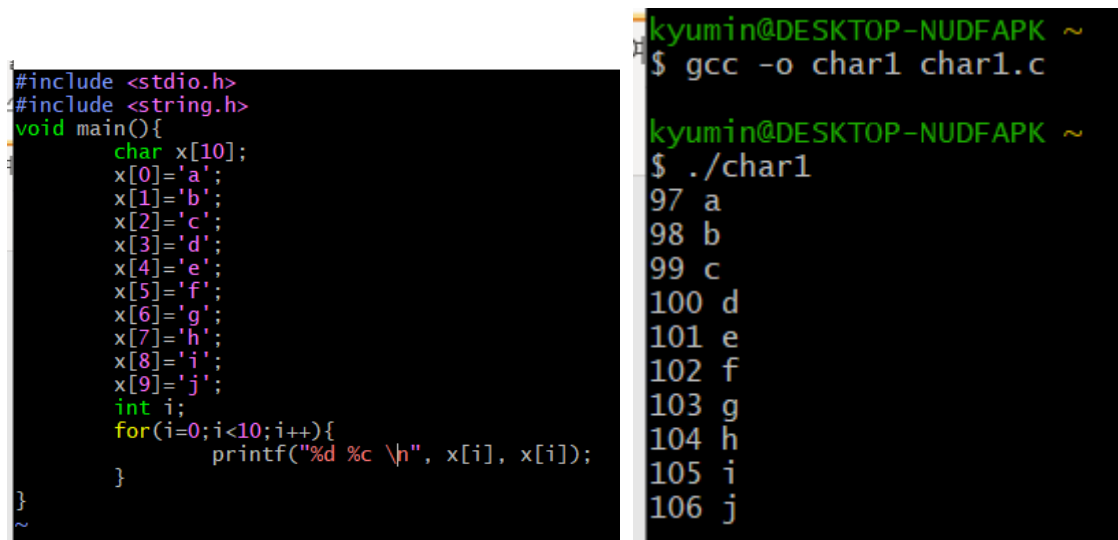
캐릭터 상수는 아스키코드를 저장하기 때문에 따옴표를 사용하거나 아스키코드를 사용해 문자를 저장할 수 있다. ‘a’의 아스키 코드는 97이므로 위 코드의 x와 y는 같은 내용이 저장된 것을 볼 수 있다.

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 terminal windows. The left window displays the C code from the previous block, with syntax highlighting. The right window shows the execution of the program. It starts with the user running 'gcc -o char1 char1.c' and then './char1'. The output is a list of ASCII values and characters from index 0 to 9: 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
        // ASCII number
    }
}
~

```

```

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]);
}

```

```

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

```

```

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
13

```

제어 문자			공백 문자			구두점			숫자			알파벳		
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	a			
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a			
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b			
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c			
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d			
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e			
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f			
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g			
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h			
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i			
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j			
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k			
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l			
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m			
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n			
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o			
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p			
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q			
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r			
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s			
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t			
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u			
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v			
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w			
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x			
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y			
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z			
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	[			
28	0x1C	FS	60	0x3C	<	92	0x5C	\	124	0x7C	\			
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	]			
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	^			
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL			

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

```

문자를 아스키코드로 출력해주는 코드다. 처음에는 “hello”라는 문자를 저장해줬기 때문에 배열 x에는 104 101 108 108 111 0으로 저장이 되어있었으나 “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.

```

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

```

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);
```

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
```

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"
```

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);
```

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 string from the user
for(int i=0;i<5;i++)
    printf("%s\n", x[i]);
```

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
```

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]);
}
```



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

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
```

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);
```

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]);
}
```

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