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;

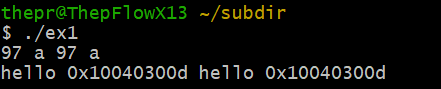
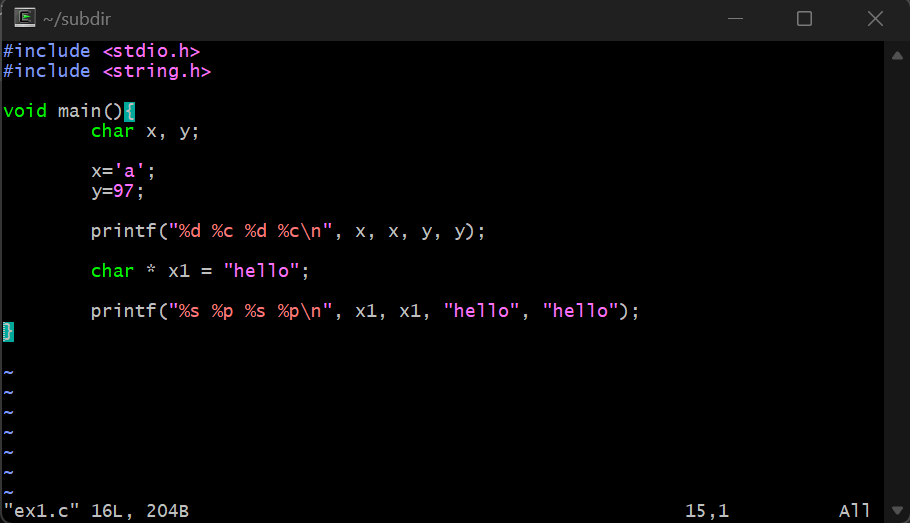
x='a'; y=97;

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

}



1. 첫 번째 printf

컴파일러는 character ‘a’를 ASCII number로 교체해서, x = 97;이 된다.

97은 a 의 ascii number이다.

따라서 x, y를 decimal 형식 지정자로 출력했을 때 97이 나오고, char 형식 지정자로 출력했을 때 a가 나온다.

1. 두 번째 printf

문자열 리터럴 “hello”는 컴파일 시 프로그램의 string area에 저장되고, “hello”는 그 주소로 대체된다.

char 포인터 x1은 “hello”의 주소를 저장하게 된다.

따라서 string 형식 지정자로 출력 시 hello가 출력되고, pointer 형식 지정자로 출력 시 “hello”의 주소가 출력된다.

string 형식 지정자는 인자의 주소로 이동해 0이 (null문자) 나올 때까지 출력한다.

“hello”는 실제로 0으로 끝나기 때문에 6byte의 공간을 차지한다.

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’;

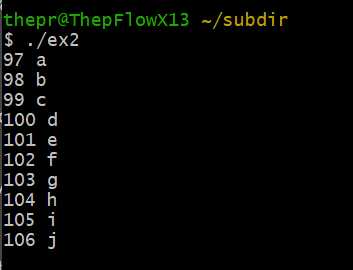
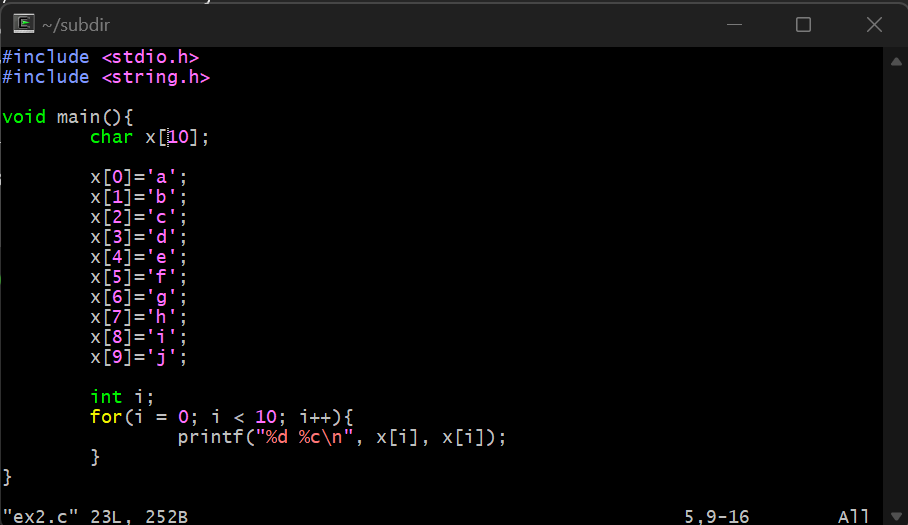
x[5]=’f’; x[6]=’g’; x[7]=’h’; x[8]=’i’; x[9]=’j’;

int i;

for(i=0;i<10;i++){

printf(“%d %c \n“, x[i], x[i]); // print each character with its ascii number

}



char constant 는 컴파일러가 해당하는 ASCII number로 교체한다. x[0]=’a’; 는 x[0]=97;과 동일하다.

형식 지정자를 Decimal로 출력할 경우 Ascii number가 그대로 나오며 char로 출력할 경우 Ascii 문자로 나온다.

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++){

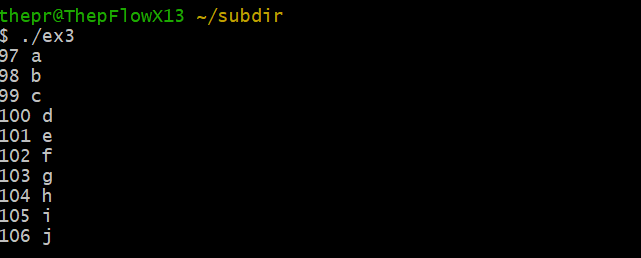
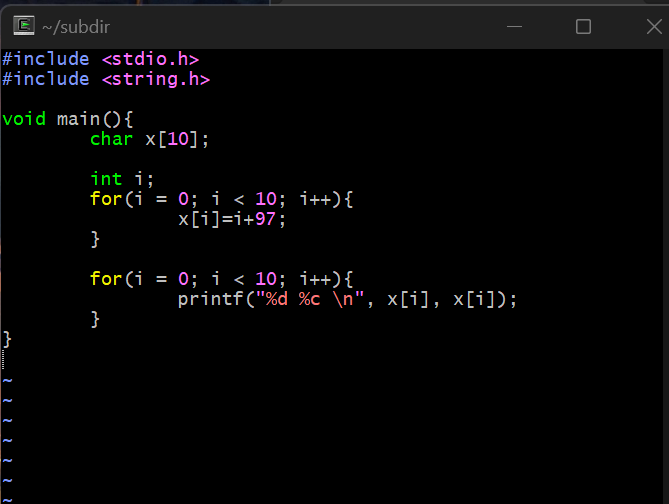
x[i]=i+97;

}

for(i=0;i<10;i++){

printf(“%d %c \n“, x[i], x[i]); // print each character with its ascii number

}



2번 문제의 결과와 동일하다. char constant 는 컴파일러가 해당하는 ASCII number로 교체한다. x[0]=’a’; 는 x[0]=97;과 동일하다.

형식 지정자를 Decimal로 출력할 경우 Ascii number가 그대로 나오며 char로 출력할 경우 Ascii 문자로 나온다.

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++){

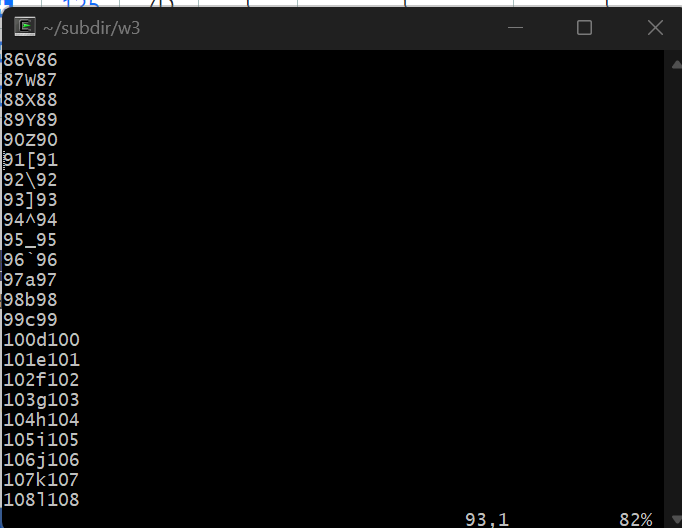
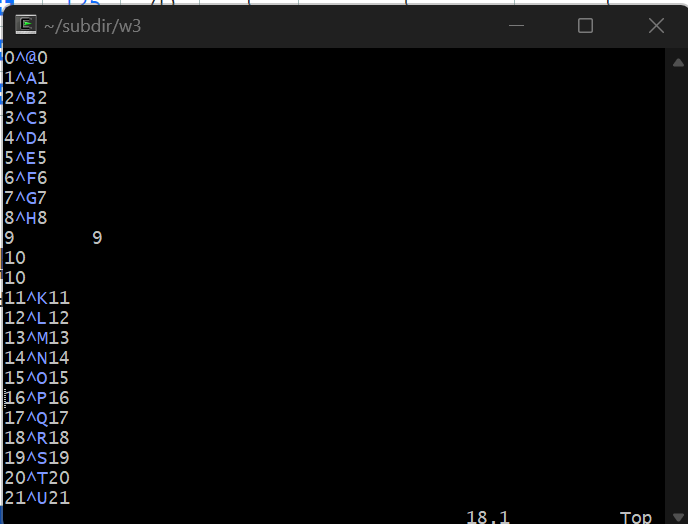
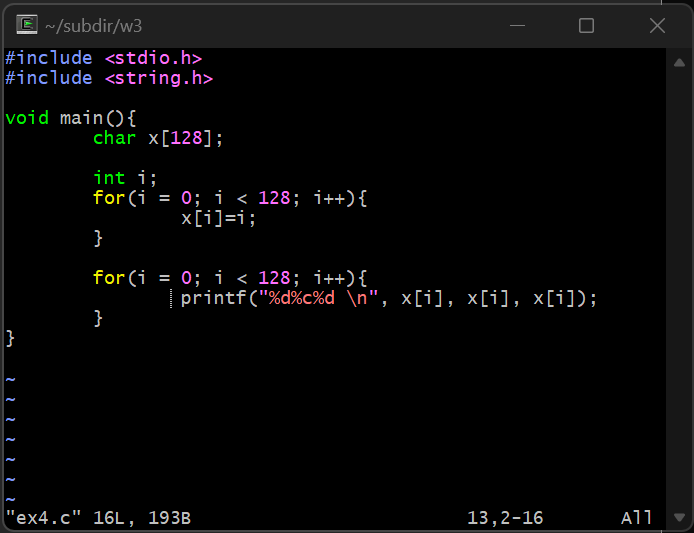
x[i]=i;

}

for(i=0;i<128;i++){

printf(“%d%c%d\n”, x[i], x[i], x[i]);

}



Windows-1252의 ASCII Table을 참고했다.

0~31은 제어 문자 (vi 편집기에선 표시된다.)

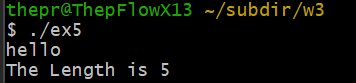
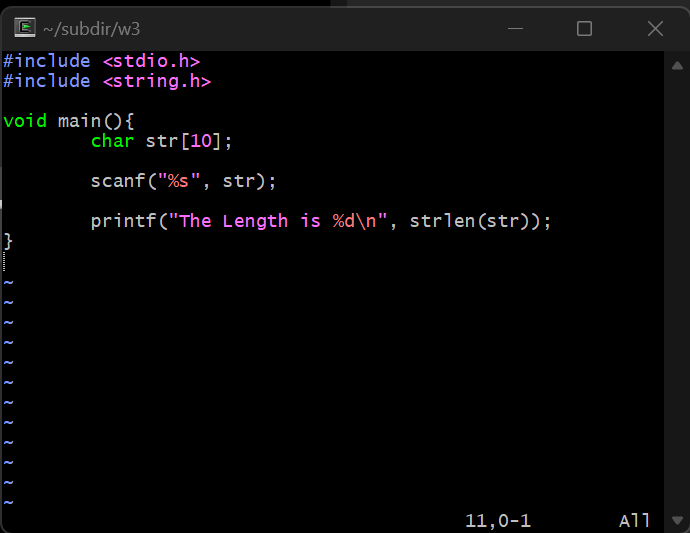
32~127은 인쇄 가능 문자 이다.

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

Enter a string

hello

The length is 5



strlen 함수는 인자로 받은 문자열의 길이를 출력한다. nul문자(0)이 나오기 전까지의 character 수를 반환하는 것 같다

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

Enter a string

hello

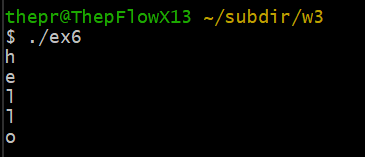
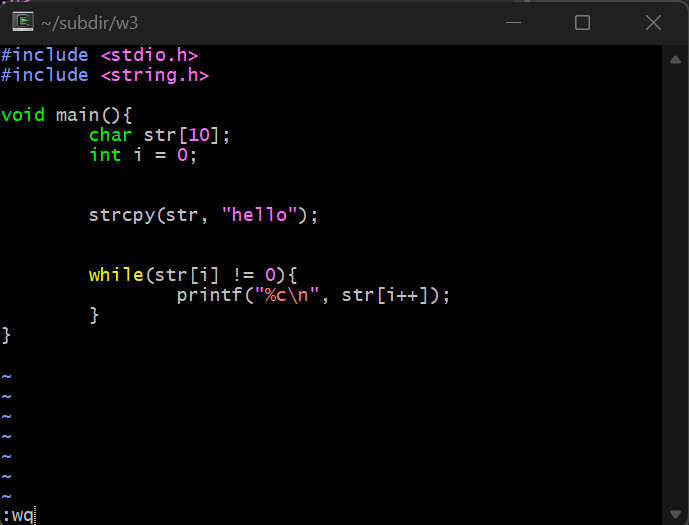
h

e

l

l

o



Constant string “hello”를 char 배열에 복사한 뒤, 0이 나오기 전까지만 순서대로 출력하는 프로그램을 작성했다.

hello 까지 출력하고 종료된 되었으므로, char 배열의 hello 뒤에 0이 있는 것이다.

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

char x[10];

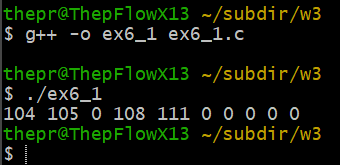
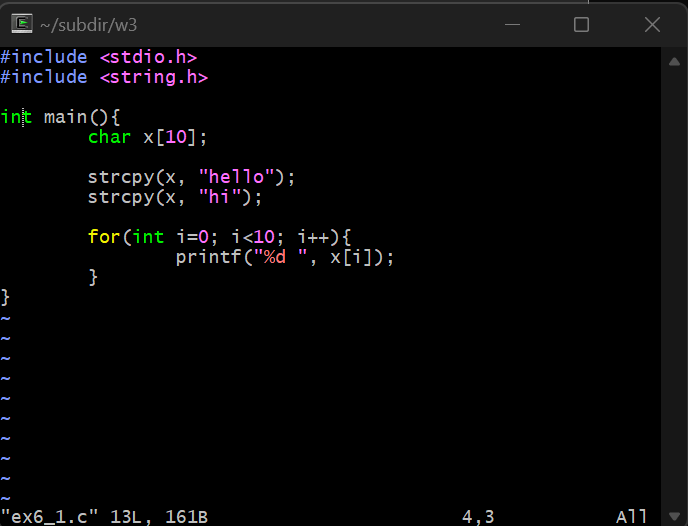
strcpy(x, "hello");

strcpy(x, "hi");

for(int i=0;i<10;i++){

printf("%d ", x[i]);

}



1. char array에 constant string “hello” 를 복사하면, 104 101 108 108 111 0 0 0 0 가 된다.
2. 동일한 char array에 constant string “hi”를 복사하면 맨 앞 3칸에 104 105 0 이 덮어씌워져 위 결과 사진과 같아진다.

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

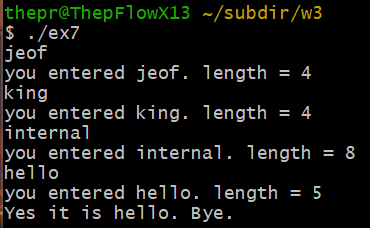
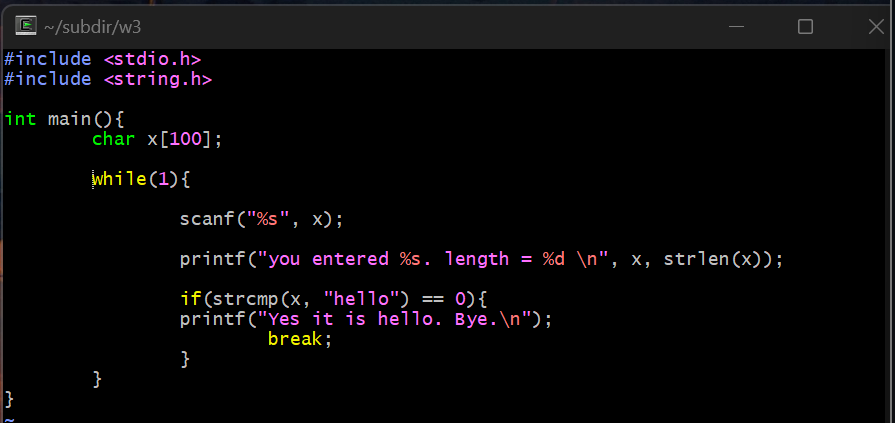
Enter a string

hello

You entered hello. length=5

Yes it is hello. Bye.

가나다



strcmp는 두 문자열을 비교해 null문자가 나오기 전까지 같을 경우 0을 리턴한다.

그 외의 경우는 0이 아닌 다른 수를 리턴한다.

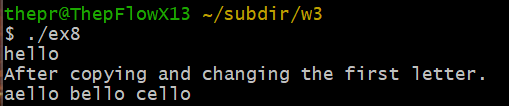
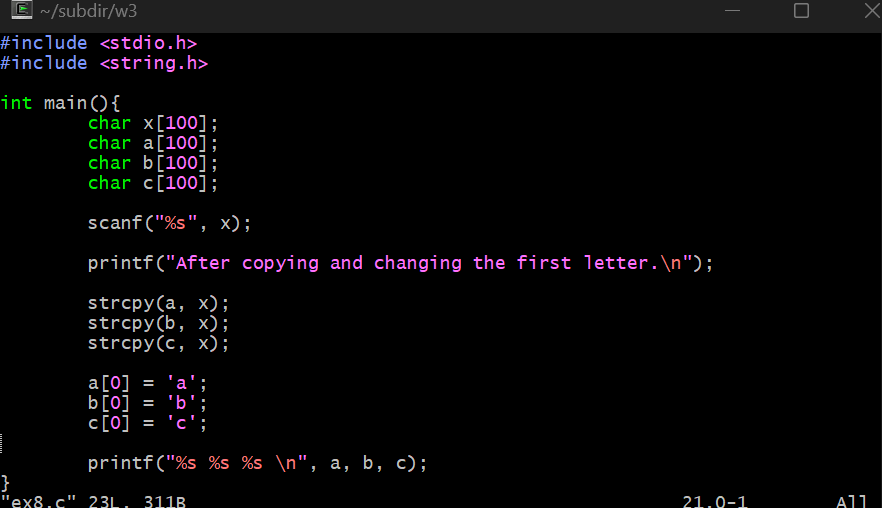
8) [strcpy] Read a string and copy it to three other string variables and change the first letter of them to 'a', 'b', and 'c', respectively, and display them.

Enter a string

hello

After copying and changing the first letter

aello bello cello



stycpy를 사용해 문자열을 다른 배열에 복사해 주었다.

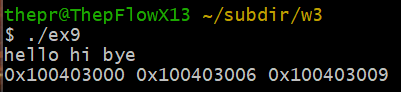
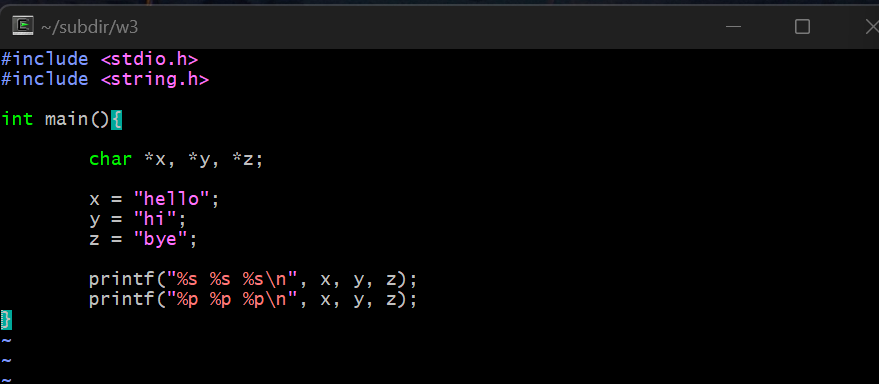
9) [string constant] A string constant such as "hello" is an address. Explain the result of following code.

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);



Constant String (문자열 리터럴)은 컴파일 시 프로그램의 string area에 저장되고, 문자열의 자리는 그 주소로 대체된다.

예를 들어 char 포인터 x은 “hello”의 주소를 저장하게 된다.

따라서 string 형식 지정자로 출력 시 hello가 출력되고, pointer 형식 지정자로 출력 시 “hello”의 주소가 출력된다.

hi, bye에 대해서도 마찬가지 이며,

주소를 확인했을 때 인접해 있는 것을 확인할 수 있다. (String area)

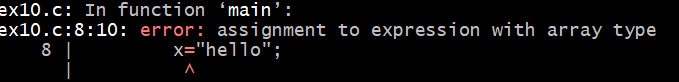
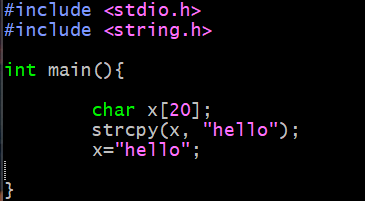
10) [string constant is an address] Try below and explain why we have an error.

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



“hello”는 문자열 리터럴로, string area를 가리키고 있는 주소값이다.

배열 x는 배열의 시작을 가리키고 있는 상수 포인터로 생각할 수 있다.

따라서 x에 hello의 주소를 할당하는 것은 불가능하므로 컴파일 에러가 난다.

문자열 할당 또는 복사를 의도했다면 strcpy를 이용해야 한다.

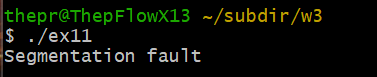
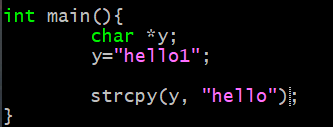
또는 x 배열에 문자열 하나씩 직접 할당해 주어야 한다.

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"



char 포인터 y는 임의의 주소를 가리키는 변수이다.

문자열 리터럴 “hello1”은 string area의 hello의 시작 주소 값이므로 y가 가리킬 수 있다.

strcpy는 첫 인자를 char 배열(문자열)이라고 생각하고, y의 주소에 할당 작업을 할 것이다.

하지만 y가 가리키고 있는 주소는 string area (text section)이므로 쓰기작업을 할 시 Segmentation error가 발생한다.

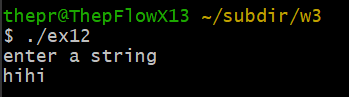
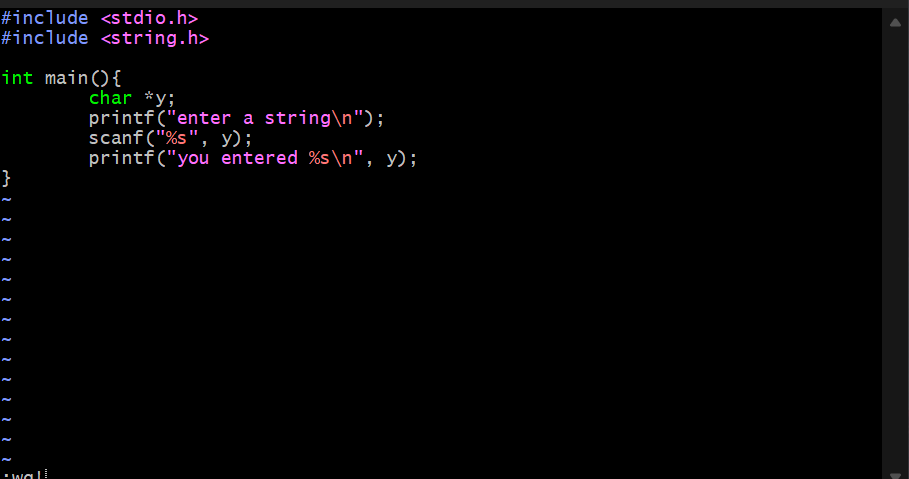
12) [You need memory space for scanf] Try below and explain why you have an error. Fix it.

char \*y;

printf("enter a string\n");

scanf("%s", y); // error becuase y has no space for a string

printf("you entered %s\n", y);



char pointer y는 임의의 주소를 가리키는 포인터 변수이다. 이 변수가 초기화 되어있지 않으므로 0일 것이다.

scanf에서는 y가 가리키는 주소에 접근하려 할 것이고, 0x0은 접근 불가능한 영역이기 때문에 Segmentation fault가 발생할 것이다.

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

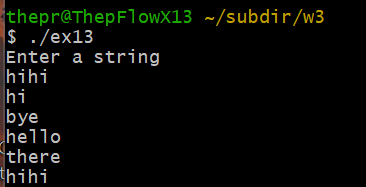
x[4]=new char[50];

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]);



Char pointer array의 원소는 char pointer이므로 1. 문자열 리터럴의 주소 또는 2. 동적 할당된 공간의 주소 를 가리킬 수 있다.

operator new로 동적 할당 후 strcpy 문자열이 잘 저장되어 있는 것을 알 수 있다.

operator new는 할당한 공간의 주소를 Return하기 때문이다.

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

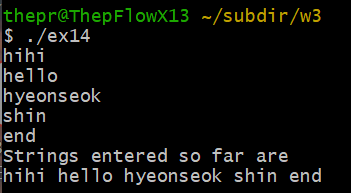
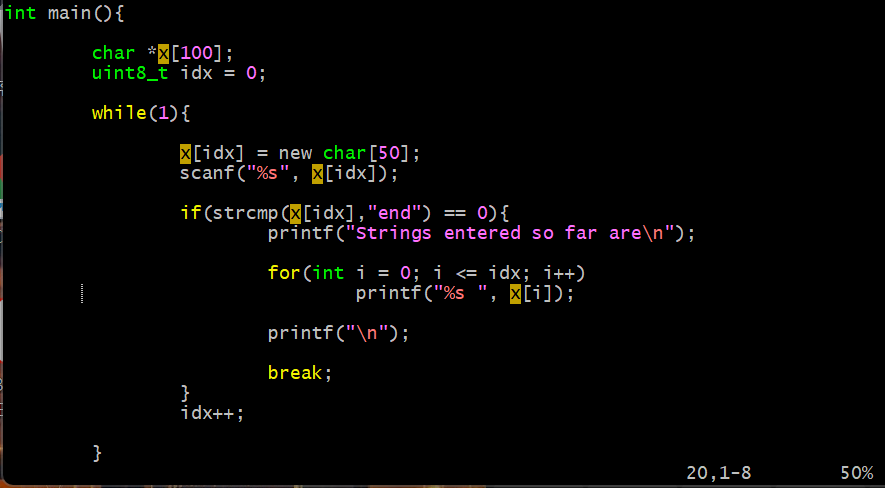
bbb

Enter a string

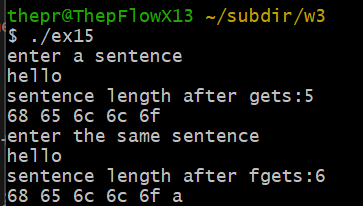
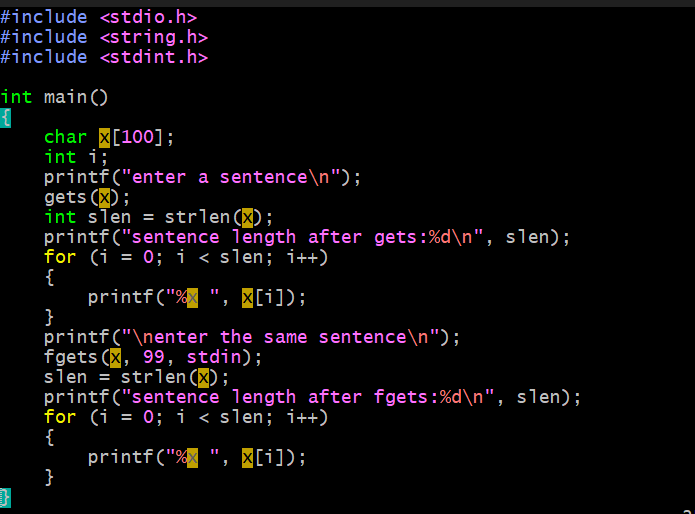
end

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.)



gets와 fgets의 차이

1. 둘 다 Strlen이 동작했다. = 받은 입력은 Null 문자 (0)로 끝난다.
2. fgets 사용 시 문자를 하나 더 받고, 개행 문자 a가 있다. = fgets는 개행 문자까지 입력받음.

위 결과를 종합해 보면 아래와 같은 결론을 얻는다.

gets는 개행문자를 입력 받지 않고 받은 문자열 뒤에 null문자를 붙인다.

fgets는 개행문자까지 입력받고 받은 문자열 뒤에 null문자를 붙인다.

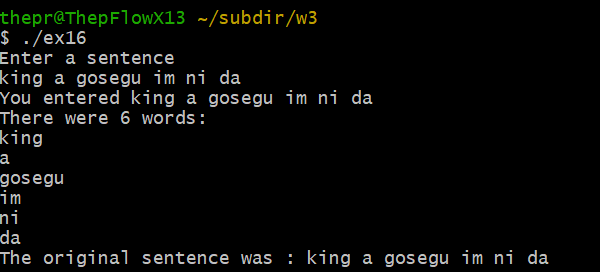
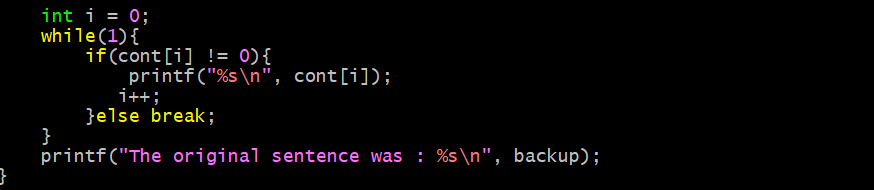
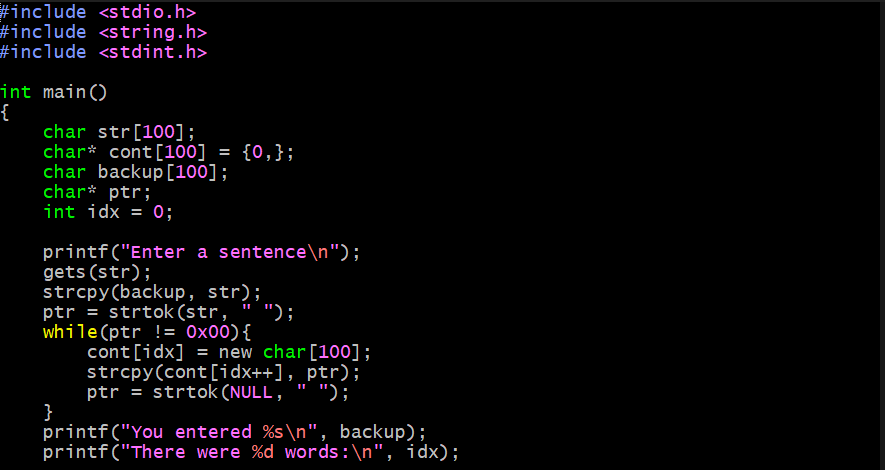
16) [strtok] Use strtok to extract words from a sentence and store them in an array. Display the number of words as below. Note that you need to copy the sentence to another string variable before doing strtok because strtok will destroy the original sentence.

algorithm:

read a line

tokenize

display tokens



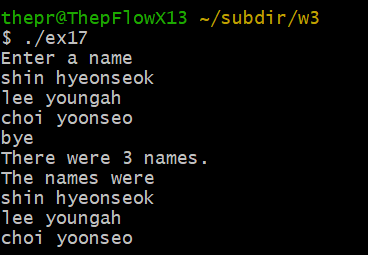
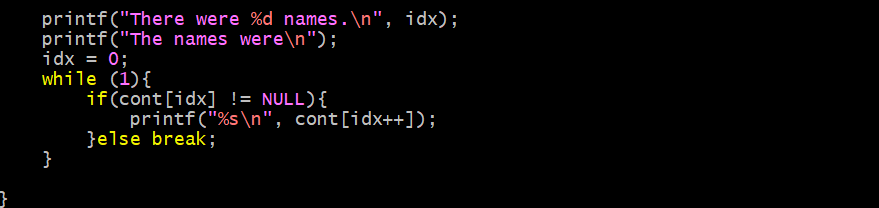
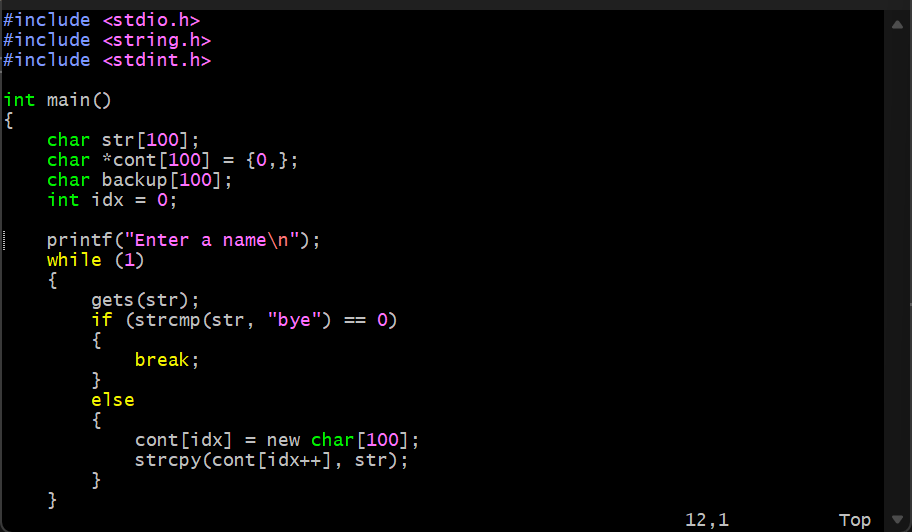
strtok의 동작 방식에 대해 알 수 있었다.

strtok함수는 파싱할 문자열, 구분자를 인자로 받고, 파싱할 문자열에서 구분자가 나올 때까지 탐색해 구분자를 0으로 바꾼 뒤 구분된 문자열의 포인터를 반환한다.

이후 strtok(NULL, 구분자)를 입력하면 전에 구분자를 찾았던 그 주소에서 탐색을 다시 시작한다.

위 동작을 반복해 char\* 배열에 저장해 주면 문자열을 분리할 수 있다.

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".



위의 문제에서 설명한 것과 동일한 함수들을 사용해 프로그램을 구현했다.

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];

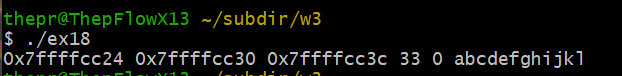
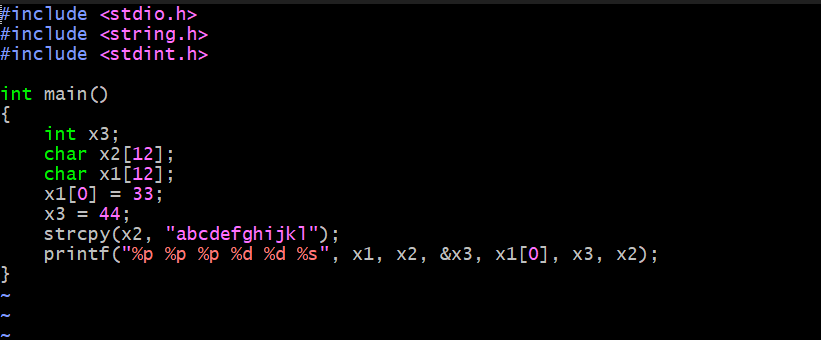
char x1[12];

x1[0]=33;

x3=44;

strcpy(x2,"abcdefghijkl");

printf("%p %p %p %d %d %s", x1, x2, &x3, x1[0], x3, x2);



문제점은 변수 x3 에 원래 44가 대입되어 있어야 하지만, 다른 주소에 strcpy 후 다시 print해보니 0이 들어있다는 것이다.

문자열 리터럴 abcdefghijkl은 null문자까지 포함하면 13 바이트지만, x2의 크기는 12이다.

즉 x2가 차지하고 있는 공간을 넘어 바로 옆에 (포인터 주소 참조)있던 x3의 자리에 널문자 0을 써버린 것이다.

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

int main(){

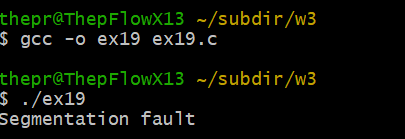
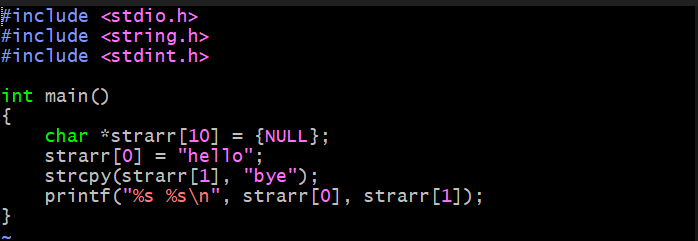
char \* strarr[10]={NULL};

strarr[0]=”hello”;

strcpy(strarr[1],”bye”);

printf("%s %s\n", strarr[0], strarr[1]);

}



char \* 배열을 생성하고 NULL로 초기화 한다.

배열의 첫 번째 원소는 문자열 리터럴로, string area에 옮겨지고 그 문자열의 첫 번째 주소로 치환된다.

두 번째 원소에 strcpy를 시도하는데, 이 경우 현재 strarr[1]은 NULL이기 때문에 Segmentation Fault가 발생한다.

주소 0은 시스템 영역이라 접근할 수 없기 때문이다.

제대로 작동하려면 strarr[1]은 할당된 공간의 주소를 가리키고 있어야 한다.

따라서 printf함수가 실행되기 전에 프로그램이 종료된다.

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

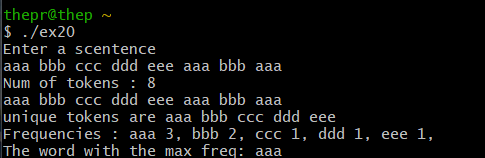
aa bcd e e ff aa bcd bcd hijk lmn al bcd

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

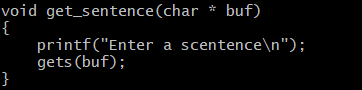
Frequncies: aa 2 bcd 4 e 2 ff 1 hijk 1 lmm 1 al 1

The word with the max freq: bcd



실행 결과는 위와 같다. 잘 동작하는 것을 알 수 있다.

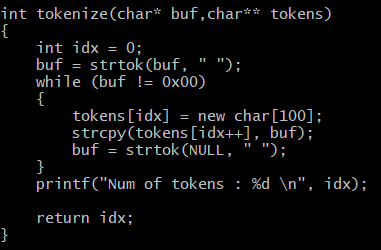
각 구현한 함수에 대해 설명하겠다.



1. get\_sentence

Enter a scentence를 출력하고 gets로 문자열을 입력받는다. Gets는 줄바꿈 문자를 제외하고 입력받으므로 사용하기에 가장 적절하다고 생각했다.

1. tokenize

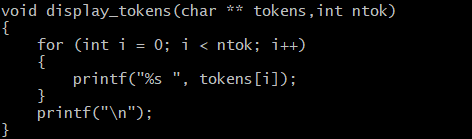


띄어쓰기 공백 문자를 기준으로 Tokenize한다. 과제 16번에서 구현한 것과 같은 코드이다.

각 토큰은 문자열 배열 (char\*\*) tokens에 동적 할당하고 strcpy로 복사한다.

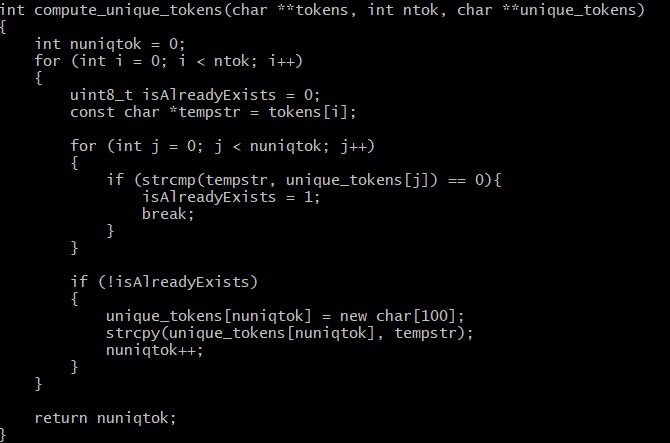
다음 토큰의 수를 return한다.

1. display\_tokens



문자열 배열 tokens과 길이를 입력받아 토큰들을 출력한다.

1. compute\_unique\_tokens

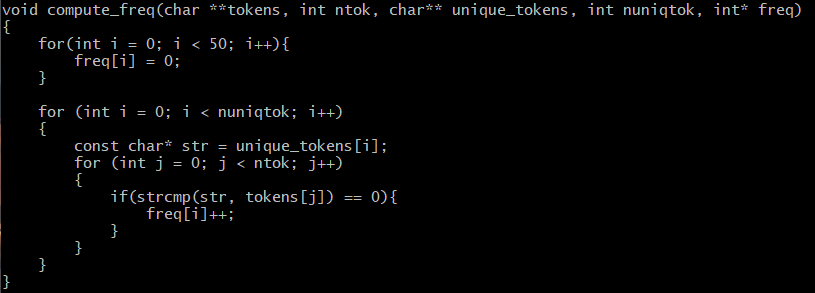


tokens 중 중복되지 않는 고유한 토큰들을 unique\_tokens 배열로 옮긴다.

tokens의 각 원소에 대해 unique\_tokens에 이미 존재하는 지 확인하고, 존재하지 않을 경우 복사한다.

다음 unique\_tokens의 길이를 return한다.

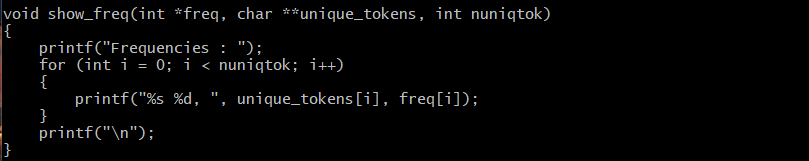
1. compute\_freq



tokens에서 각 원소의 빈도를 count해 int 배열 freq에 저장한다.

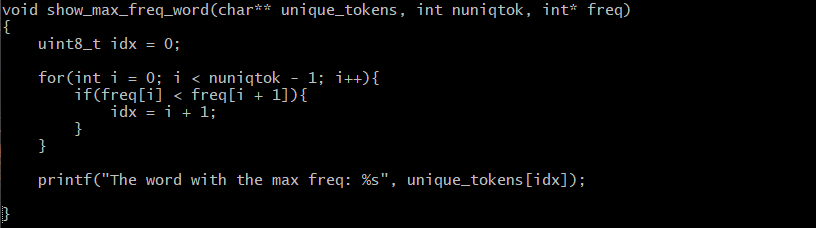
unique\_tokens의 각 원소를 tokens에서 찾아 횟수를 count한 뒤, freq 배열에 저장한다.

1. show\_freq



각 unique\_tokens에 대응하는 freq를 출력하는 함수이다.

1. show\_max\_freq\_word



Freq 중 가장 큰 값을 찾아 대응하는 token을 unique\_tokens에서 출력한다.