컴프 #4 수업에서 다루어지는 것들: 문자 • 문자열 처리함수

■ 문자 단위 처리

□ 문자 입출력 함수

문자 입력 함수	설명	문자 출력 함수	설명
scanf("%c", &ch);	stdio.h	printf("%c", ch);	stdio.h
ch = getchar();	stdio.h, 줄단위	putchar(ch);	stdio.h
ch = fgetc(stdin);	conio.h, 줄단위	fputc(ch, stdout);	conio.h
ch = getch();	conio.h, 문자단위, 에코하지 않음	putch(ch)	conio.h
ch = getche();	conio.h, 문자단위		

✧ **사용예:**

어느 감시값('q' 혹은 파일의 끝(EOF))까지 문자 단위로 읽기 위해
아래와 같이 버전1, 버전2, 버전3의코드가 가능하다.
여기서는 **getchar, putchar**의 사용하였으며
fgetc, fputc 등등의 다른 문자 입출력 함수도 사용 가능하다.

```
// 버전1
while (1)
{
   ch = getchar();
   if (ch == 감시값)
     break;
   putchar(ch);
}
```

```
// 버전2
ch = getchar();
while (ch != 감시값)
{
   putchar(ch);
   ch = getchar();
}
```

```
// 버전3
while ((ch = getchar()) != 감시값)
   putchar(ch);
```

□ 문자 처리 라이브러리 함수(ctype.h에 포함)

- 문자 검사 라이브러리 함수(참이면 0이 아닌값, 거짓이면 0이 반환)
  - isalpha(c) : c가 영문자인가?(a-z, A-Z)
  - isupper(c) : c가 대문자인가?(A-Z)
  - islower(c) : c가 소문자인가?(a-z)
  - isdigit(c) : c가 숫자인가?(0-9)

- 문자 변환 라이브러리 함수(바꾼 문자를 반환)
  - toupper(c) : c의 대문자를 반환
  - tolower(c) : c의 소문자를 반환

✧ **사용 예**

문자형 변수 c가 대문자이면 이의 소문자를 출력하는 아래의 프로그램을

```
if (c >= 'A' && c <= 'Z')
      printf("%c", c + 32);
```

라이브러리 함수를 사용하여 다음과 같이 바꿀 수있다.

```
if (isupper(c))
     printf("%c", tolower(c)); // 이때 c가 소문자로 바뀌나?? 답: _____
```

■ 문자열 단위 처리

□ (문자열 입출력 함수)

문자열 입출력 함수는 문자 입출력 함수에 비해 조금 **복잡**하다.
특히, 문자열 입력 함수를 주의 깊게 살펴보라.

문자열 입력 함수:

```
① scanf("%s", str);        // white character(공백, 탭, 엔터)가 나올때까지 읽어서 str에 넣는다
② gets(str);               // 엔터가 나올때까지 읽는다. sizeof(str)을 이상이면 ERROR!
③ fgets(str, size, stdin); //  '₩0' 을 포함해서 size의 크기만큼
                           //혹은 엔터가 나올때까지 읽어 str에 넣는다. (이때, 엔터까지 str에 넣음!!)
                           // sizeof(str) 이상이면 (size - 1)까지 읽고 그다음에 '₩0' 을 넣는다.
```

→ scanf는 공백이 나올때까지 읽지만
→ gets, fgets는 엔터가 나올 때까지 읽기 때문에 한줄을 하나의 문자열로 읽을 수있다!

	gets	fgets
사용예	char s[5]; gets(s);	char s[5]; fgets(s, sizeof(s), stdin);
abc 입력시	s <- abc₩0	s <- abc₩n₩0 (₩n이 들어감에 유의)
abcdef	s <- abcde (위험: 문자열 아님)	s <- abcd₩0

gets 문자열은 사용에 주의를 해야한다.

■

문자열 출력 함수:

```
printf("%s", str);     // str이 가르키는 곳부터  '₩0' 이 나올때까지
puts(str);             // str이 가르키는 곳부터  '₩0' 이 나올때까지, 자동 줄바꿈!
fputs(str, stdout);    // str이 가르키는 곳부터  '₩0이 나올때까지
```

□ 문자열 처리 함수

기능	사용형식	비고
문자열 길이	**strlen**(str) // string length - str 의 길이를 반환한다	string.h
문자열 복사	**strcpy**(str1, str2) // string copy - str2 를 str1 에 복사한다.(그리고 그 복사된 str1 을 반환)	
	strncpy(str1, str2, n) - str2 의 문자 n 개를 str1 에 복사. 이때 '₩0'은 붙이지 않음.	
문자열 접합	**strcat**(str1, str2) // string concatenation (접합) - str2 를 str1 에 붙인다. (그리고 그 str1 을 반환한다)	
	strncat(str1, str2, n) - str2 의 문자 n 개를 str1 뒤에 붙인다. 이때 '₩0'을 끝에 붙임.	
문자열 비교	**strcmp**(str1, str2) // string compare - str1 이 str2 보다 작으면 -1, 같으면 0, 크면 1 "abc" < "bbb" 이며 "abc" > "ABC"로 본다.(아스키 값 비교)	

문자열 토큰 분리	**strtok**(char *str, const char * delimiter) // string token
	- str 에서 토큰을 찾아서 이를 반환한다. 토큰이 더 없으면 NULL 반환.
	delimiter 는 토큰 분리자
	- str 에서 다음 토큰을 읽으려면 str 대신 NULL 으로 호출
	char name[20] = "Suehee Sue Pak";
	char * firstName, *middleName, *lastName;
	firstName = strtok(name, " ");
	middleName = strtok(NULL, " ");
	lastName = strtok(NULL, "₩0");	

문자열 수치 변환	int **atoi**(str); // 문자열을 int 형으로 변환
	- atoi("111")은 정수형 111 을 반환함
	double **atof**(const char *str); // 문자열을 double 형으로 변환
		stdlib.h

LAB 9 문자 및 문자열 처리 함수

✪ 강의 요약(summary)를 참고하여 다음의 실습숙제를 프로그래밍 하라.

■ **9_0(문자형 입출력 연습)**

□ **LAB9_0_1(버전 1 을 사용하여** 여러 문자열 입력 함수를 연습)아래의 주어진 프로그램에 대해서 예 1) 예 2) 예 3) 예 4) 예 5)를 차례로 주석처리 취소시키면서 실행시켜보자. 즉 5 가지 종류의 문자열 입력 함수를 연습하는 문제이다. q 를 입력하면 프로그램은 끝난다.

입력: abcde(리턴) abcdeq(리턴) 등등

결과: 예 1), 예 2), 예 3)은 실행결과가 같고 예 4) 그리고 예 5)는 각각 다른 결과를 보여준다. 어떻게 다른지 잘 생각해보라.

- 줄단위/문자단위
- 에코함/에코하지 않음

```
// 버전 1 의 연습
#include <stdio.h>
#include <conio.h>
int main(void)
{
   char ch;
   while(1)
   {
      // scanf("%c", &ch); // 예 1)
      ch = getchar();  // 예 2)
      // ch = fgetc(stdin); // 예 3)
      // ch = getch(); // 예 4)
      // ch = getche(); // 예 5)

      if( ch == 'q' ) break;
      putch(ch); // printf("%c", ch), putchar(ch), fputc(ch, stdout)나 모두 같은 결과
   }
}
```

■ **LAB9_0_2(버전 2, 3 의 연습)** 아래의 주어진 프로그램은 문자들을 입력받고 이를 그대로 출력(echo)하는 단순한 프로그램이다. 이 프로그램은 파일의 끝 ^Z(control 과 z 를 동시에 누름)까지 실행된다. ^z 는 키보드 입력시 파일의 끝을 나타낸다.

```
#include <stdio.h>
#include <ctype.h> //문자처리 함수 사용을 위해
void main()
{
   char c;
   printf("Enter characters(^Z for exit):\n");
   c = getchar();
   while (c != EOF)
   {
      putchar(c);
      c = getchar();
   }
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
Enter characters(^Z for exit):
abcd
abcd
DongDuk2010
DongDuk2010
^Z
계속하려면 아무 키나 누르십시오 . . .
```

□ **LAB9_0_2a**(isupper, islower, toupper, tolower 함수의 연습)
위에 주어진 프로그램을 약간 수정하여
오른 쪽과 같은 실행결과를 내는 프로그램을 작성하라.
즉, 파일의 끝까지, 문자를 입력 받아서 소문자는 대문자로
대문자는 소문자로 바꾸어 출력하는 프로그램이다.
문자 입출력을 위해 getchar, putchar 를 사용한다.

```
C:₩windows₩system32₩cmd.exe
Enter characters(^Z for exit):
abcd
ABCD
DongDuk
dONGdUK
DongDuk2010
dONGdUK2010
^Z
계속하려면 아무 키나 누르십시오 . . .
```

□ **LAB9_0_2b** 입력, 출력 함수로 fgetc, fputc 를 사용하여
위의 문제를 다시 풀되
반복의 형태를 버전 3 과 같이 변경하라.

■ **LAB9_1 (문자입출력와 버퍼비우기)**

□ **아래 프로그램을 작성한 후 지시에 따라 작업을 수행해 보자.**

가) 위의 프로그램을 **그대로** 실행시켜 결과를 살펴보는 문제이다.
첫번째 입력 문자열로 AAAqXXX,
두번째 입력 문자열로 BBBq를 입력한다.

아래의 입력 반복문은 q가 나올때까지 입력된 문자들을 출력하고 뒤에 나오는 문자들을 무시하려는 의도를 갖는다. 그러나 의도와는 달리 뒤의 XXX가 다음 반복문에서 출력되는 것을 볼 수있다.(왼쪽 실행예)

첫번째 입력인 AAAqXXX이 있은 후에 AAA는 출력하고
q를 만나 첫번째 while문을 끝내게 된 후,
입력 버퍼에 XXX가 남아있게 됨으로서 생기는 문제이다.

```
C:₩windows₩system32₩cmd.exe
Enter 문자열<q이전까지 문자열로 인정함>
AAAqXXX
AAA
--------------------------------------------------------------------------------
Enter 두번째 문자열<q이전까지 문자열로 인정함>
XXX
BBBq
BBB
계속하려면 아무 키나 누르십시오 . . .
```

<의도하지 않은 결과>

```
C:₩windows₩system32₩cmd.exe
Enter 문자열<q이전까지 문자열로 인정함>
AAAqXXX
AAA
--------------------------------------------------------------------------------
Enter 두번째 문자열<q이전까지 문자열로 인정함>
BBBq
BBB
계속하려면 아무 키나 누르십시오 . . .
```

<의도한 결과: AAA, BBB가 출력됨>

나) 오른쪽 실행예와 같이 '제대로' 실행되게 하려면 위의 코드를 어떻게 변경해야하는가? 즉, 첫번째 반복문이 실행한 후 입력 버퍼에 남아있는 문자들을 어떻게 비울 것인가?
(힌트: 컴프 1 시간에 배운데로 버퍼를 비우는 문장을 삽입한다.
fflush 대신
while (getchar() != '₩n'); 를 사용하라.

```
#include <stdio.h>
void main()
{
   char c;

   printf("Enter 문자열(q이전까지 문자열로 인정함)\n");

   c = getchar();
   while (!(c == 'q'))
   {
      putchar(c);

      c = getchar();
   }

   printf("\n--------------------------------------------------------------------------------\n");
   printf("Enter 두번째 문자열(q이전까지 문자열로 인정함)\n");
   c = getchar();
   while (!(c == 'q'))
   {
      putchar(c);

      c = getchar();
   }
   putchar('\n');
}
```

■ 문자열 단위 처리

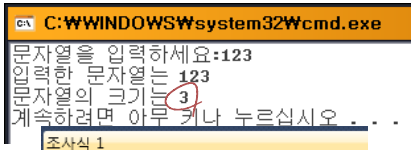
■ **LAB9_2** fgets 와 gets 의 차이점을 살펴보자. 아래와 같이 str1, str2 의 크기를 5 로 선언하면 최대 4 개의 문자를 넣고 '₩0'을 넣음으로써 정상적인 문자열이 될 수있다.

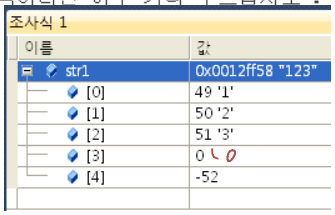
```c
#include <stdio.h> // LAB9_2a
#include <string.h>
int main(void)
{
   char str1[5];

   printf("문자열을 입력하세요:");
   gets(str1);
   printf("입력한 문자열는 %s\n", str1);
   printf("문자열의 크기는 %d\n", strlen(str1));
}
```

```c
#include <stdio.h> // LAB9_2b
#include <string.h>
int main(void)
{
   char str2[5];

   printf("문자열을 입력하세요:");
   fgets(str2, sizeof(str2), stdin);
   printf("입력한 문자열는 %s\n", str2);
   printf("문자열의 크기는 %d\n", strlen(str2));
}
```

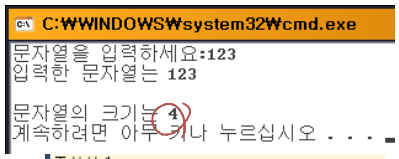
☐ 문자가 4 개 이하로 입력되는 경우의 실행예이다.
왼쪽의 gets 함수실행에서는 '1' ,' 2' ,' 3' 그리고 '₩0' 으로 문자열이 만들어지며
오른쪽의 fgets 함수 실행에서는 '1' ,' 2' ,' 3' 그리고 '₩n' 그리고 '₩0' 으로 문자열이
만들어진다. 아래(문자열 123 입력)의 실행결과와 메모리 상태를 이해한 후, 문자열 1234 를 입력한 후 str1 혹은 str2 에 어떤 값이 저장되는지 살펴보라.

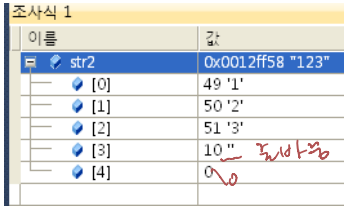
gets



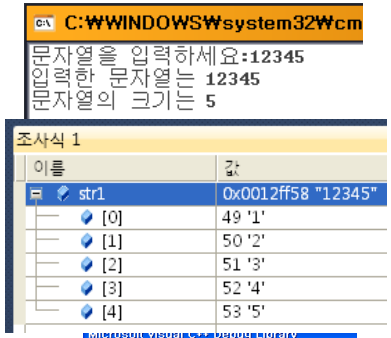


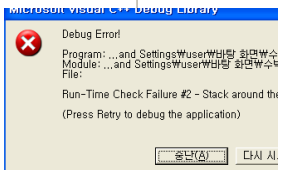
fgets

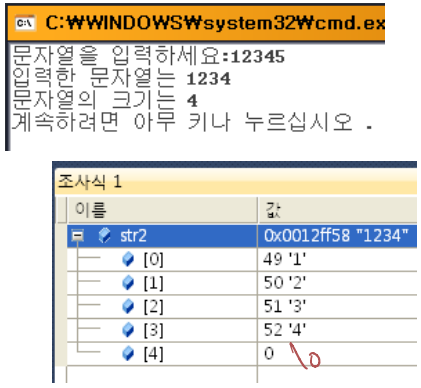




☐ 문자가 5 개 이상 입력된 경우의 실행예이다.
왼쪽의 gets 함수에서는 Error 가 나며(5 개가 꽉차서 '₩0' 을 붙이지 못함!!, 즉 문자열이 아님!!)
오른쪽의 fgets 는 (입력 문자의 개수가 정해져있으므로) 성공적으로 수행된다.
(단, 이때 입력버퍼에 나머지 문자가 (여기서는 5 와 ₩n)이 들어있으므로 다음 입력 처리를 위해서는 fflush(stdin)가 필요할 수도 있다)







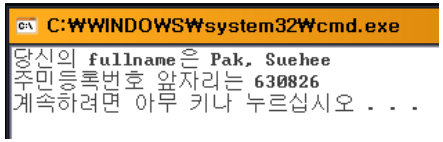
■ **LAB9_3(문자열 처리 함수)** 우리는 문자열이나 포인터 실습/숙제에서 stringLength, stringCompare 등을 직접 정의하여 사용하여왔다. string.h 헤더파일에서 제공하는 편리한 문자열 함수들이 있으므로 그것을 사용하면 편리하다.

함수들을 사용해보자. 아래의 실행결과를 예측하여 출력문 옆에 써보고 실제의 실행결과와 비교해보자.

각 문자열 처리함수의 사용을 숙지하라.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(void)
{
  char str1[15] = "ABCDED";
  char str2[15] = "abcde";
  char str3[15] = "1234";
  int n;

  char s[] = "Life is short, but art is long";
  char delimiters[] = " ,\n\0"; // 네 개의 분리 문자들(공백,콤마,줄바꿈,널문자) 지정한다
  char *token;

  printf("가)str1 의 길이 = %d\n", strlen(str1));
  printf("   str2 의 길이 = %d\n", strlen(str2));

  strcpy(str1, str2);
  printf("나)str1 = %s\t\tstr2 = %s\n", str1, str2);

  strncpy(str1, str3, 2); // strncpy 는 자동으로 '\0'을 넣지 않음에 주의하자!
  printf("다)str1 = %s\t\tstr3 = %s\n", str1, str3);

  strcat(str1, str2);
  printf("라)str1 = %s\tstr2 = %s\n", str1, str2);

  strncat(str1, str3, 2); // strncat 는 '\0'을 자동으로 넣는다!
  printf("마)str1 = %s\tstr3 = %s\n", str1, str3);

  printf("바)%d %d %d\n", strcmp("aaa", "abc"), strcmp("aaa", "aaa"), strcmp("ddd", "ccc", str3));

  strcpy(str1, "111");
  n = atoi(str1) * 7;
  printf("사)str1 를 7 배 한 값은 %d\n", n);

  printf("아)\n"); // 중요!!
  n = 1;
  token = strtok(s, delimiters);
  while (token != NULL)
  {
     printf("%d 번째 토큰은 %s\n", n++, token);
     token = strtok(NULL, delimiters);
  }
}
```

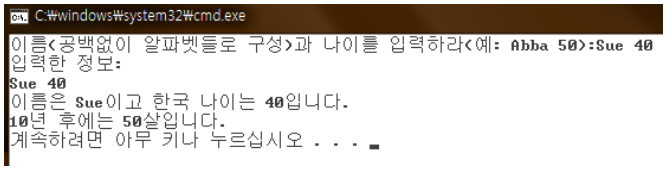
■ **LAB9_4(문자열 처리 함수, strcpy, strcat, strncpy 등등)** 아래의 실행결과가 나오도록 프로그램을 완성하라.




```
#include <stdio.h>
#include <string.h>
int main(void)
{
   char fullName[50];
   char lastName[50] = "Pak";
   char firstName[50] = "Suehee";

   char id[15] = "630826-2020222"; // 14 개의 문자 + '\0'
   char idFront[7];

   // fullName 에 성과 ", " 그리고 이름을 넣는다.



   printf("당신의 fullname 은 %s\n", fullName);

   // idFront 에 주민등록 번호의 앞 6 자리를 넣는다.


   printf("주민등록번호 앞자리는 %s\n", idFront);
}
```

■ **LAB9_5(문자열 처리)(strtok, atoi 함수 사용 연습)**

아래와 같이 이름과 나이를 하나의 문자열로 읽어서 이중 이름만을 출력하는 프로그램을 작성하라.
(이름은 공백 없는 문자들로, 그리고 공백문자후에 나이가 정확히 입력된다고 가정하자)
힌트: 이때는 이름의 길이를 알 수 없으므로 공백문자가 나올때까지 문자들을 읽으면서 name 에 저장하여야한다.

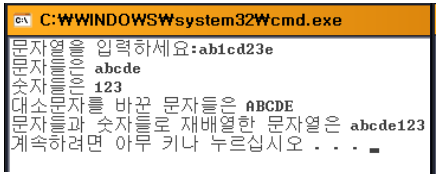


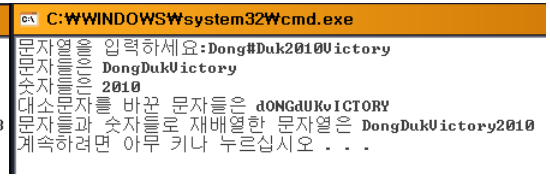

```
#include <stdio.h>
#include <string.h>
int main(void)
{
   char str[50];
   int i;
   char *name, *age;

   printf("이름(공백없이 알파벳들로 구성)과 나이를 입력하라(예: Abba 50):");
   gets(str);

   puts("입력한 정보: ");
   puts(str); // 바로 출력해본다.


   printf("이름은 %s 이고 한국 나이는 %s 입니다.\n", name, age);
   printf("10 년 후에는 %d 살입니다.\n",              )
}
```

HW 9

■ **HW9_1 적당한 길이의 문자열을 입력받는다(fgets 사용). 그런후**
1. **그 안에 존재하는 문자들만을 한 단어(alphabetString1)로 출력하고,**
2. **그 안에 존재하는 숫자들로 한 단어(digitString)를 출력하고**
3. **그 단어 안의 소문자는 대문자로 대문자는 소문자로 바꾼 단어(alphabetString2)를 출력하고,**
4. **문자열단어와 숫자단어를 하나의 단어(convertedString)로 출력하는 프로그램을 작성하라.**

힌트 및 요구사항
- 위의 1,2를 먼저 성공한 후 3, 4를 차례로 진행하라.
- 문자열 입력시 fgets를 사용하라.
- 문자 처리 함수(isalpha, isdigit, isupper 등등), 문자열 처리 함수(strcpy, strcat, strlen등등)를 사용하라.
- 아래 주어진 printf문 외에 더 이상의 printf문은 사용하지말라.



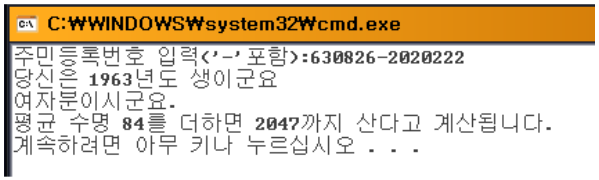


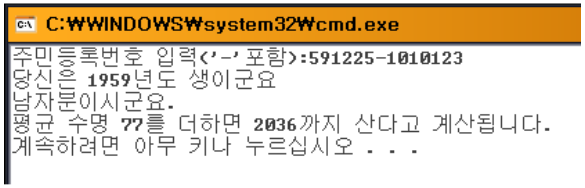

```
#include <stdio.h>
#include <string.h>

int main(void)
{
   char string[50]; // 입력되는 문자열을 저장
   char alphaString1[50]; // 입력문자열에서 알파벳만 저장
   char alphaString2[50]; // 대소문자를 바꿈
   char digitString[50];
   char convertedString[50]; // 문자들과 숫자들로 재배열한 문자열


   printf("문자열을 입력하세요:");









   printf("문자들은 %s\n", alphaString1);
   printf("숫자들은 %s\n", digitString);









   printf("대소문자를 바꾼 문자들은 %s\n", alphaString2);
   printf("문자들과 숫자들로 재배열한 문자열은 %s\n", convertedString);

}
```

■ **HW9_2**

주민등록번호를 입력받아서('-'를 포함해서 정확히 정확히 입력된다고 가정)
평균수명을 더한 년도를 출력하는 프로그램을 작성하라.
남자, 여자의 평균수명은 각각 77 세, 84 세이다.
주민등록번호 뒷부분의 첫자리가 1 인경우는 남성, 2 인경우는 여성이다.
1900 년대 출생한 사람들의 주민등록번호만 다른다고 가정한다.

힌트
- 한꺼번에 문제를 풀려고 하지말고, 문제를 단계로 나누어서 풀도록 하라.
  1 단계: 당신은 ****년도 생이군요를 출력
  2 단계: 여자분이시군요를 출력
  3 단계: 평균 수명 84 를 더하면…를 출력
- 반드시 디버거를 사용해서 문제를 풀도록 하라. 이 문제의 경우 자신이 생각하는 데로 문자열에 값이 들어가나를 살펴보는 것이 중요하므로 디버거가 특히 더 유용하다.





오목

- 오목 프로그램을 작성하자. 즉, 상/하/좌/우/대각선 방향으로 다섯 개의 돌이 연속인지 여부를 확인하여 이겼는지 여부를 체크

Lab(오목 준비: 말 놓기)

- Player X와 Player O가 차례로 3X3칸으로 된 게임판의 말(O 혹은 X)을 놓은 것을 구현한다.

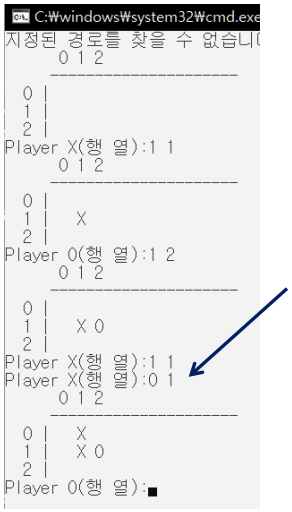
기능

1. 두 명의 플레이어가 번갈아 O와 X의 말을 놓는다.
2. 이미 고른 칸을 선택하면 다시 입력 프롬트를 보여준다.(화살표)
3. 지금은 게임판에 말이 다 놓여지면 게임 끝나는 것으로 하자
   - 추후, 실제 오목게임에서는 오목이 생성되면 게임이 끝남

프로그램 설계

- 함수화
  - 각 선택이 표시된(X와 O로) 게임판을 출력하는 부분을 함수화 하라

  void display(int b[][10])



X 안하는 애일것! (+) X랑 O을 번갈아가면서 하게!

```
#include <stdio.h>
#define BOARD_SIZE 3  // 추후에 10으로 바꾸어서 진행한다

void display(char b[][BOARD_SIZE])
{
        char ch;
        int i, j;
        printf("    ");
        for (i = 0; i < BOARD_SIZE; i++)
                printf("%2d", i);
        printf("₩n    ----------₩n");
        for (i = 0; i < BOARD_SIZE; i++) {
                printf("%3d |", i);
                for (j = 0; j < BOARD_SIZE; j++)
                        printf(" %c", b[i][j]);
                printf("₩n");
        }
}
```

```
void main()
{
   char board[BOARD_SIZE][BOARD_SIZE];
   char turn = 'X';
   int r, c;
   int i, j;
   int count;

   for(i =0; i < BOARD_SIZE ; i++ )
   for(j = 0 ; j < BOARD_SIZE ; j++ )
   board[i][j] = ' ';

   count = 1;
   display(board);
   do
   {
        printf("Player %c(행 열):", turn);
        scanf("%d %d", &r, &c);

        board[r][c] = turn;
        display(board);

        count++;
   } while(count <= BOARD_SIZE * BOARD_SIZE);
}
```

HW(수평오목)

- 수평오목 프로그램을 구현하자.(게임판은 10X10으로 하자)

기능

1. 수평으로 돌이 5개가 완성되면 승리로 본다.
   - 단, 6개 이상 연속은 이긴 것이 아님
   - 이미 고른 칸을 선택하면 다시 입력 프롬트를 보여준다(실행예 화살표 참조)
   - 승리자가 없으면 Nobody wins!라는 출력을 보여준다.

프로그램 설계

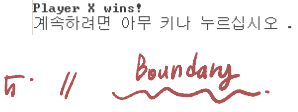
이 문제를 어떻게 나눌 것인가?를 생각하고 각 부분을 함수화하여 단계적으로 문제를 풀어본다.

- 게임판 출력

  void display(char b[][BOARD_SIZE]);
- 승리여부체크: r열, c행에 말을 놓았을때 오목성공이면 1, 아니면 0

  int winCheck(char b[][BOARD_SIZE], int r, int c);

2. for(or while)/if절이 사용될 것인데 if 수행이 최소화 하는 방향으로 작성한다.
3. 충분한 test case를 실행시켜 보면서 프로그램을 테스트한다
   - 예: 가로 완성, 6목 경우...



HW(오목)

- 이제 완전한 오목 프로그램을 구현하자.

  - 충분한 test case를 실행시켜 보면서 프로그램을 테스트한다
    - 예: 수평오목에 수직/대각선/역대각선 오목도 완성