문자/문자열 입출력&처리

Main Category	Log
Category	Algorithm
Tags	
Status	In progress
Created Time	@September 18, 2023 10:19 PM
Updated Time	@September 21, 2023 12:02 AM

문자/문자열 입출력&처리

ch12_문자와문자열.pdf.pdf

https://discreet-bay-25b.notion.site/1d39536334884a1eb321f92b74b9b46b?pvs=4

```
dst[6];
src[6]="Hello";
py(dst, src);       // src를 dst로 복사한다.
```

ch12_문자와문자열.pdf.pdf

중요하거나 활용도가 높은 내용만 정리

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/e1e88c85-ac46-4094-b2e9-25151ff88996/%EC%B6%9C%EB%A0%A5.pdf

- 문자열 선언할 때 초기화를 하는 경우, 별도의 크기를 지정해주지 않아도 문자열의 길이만큼 크기를 자동으로 생성
- 문자열을 한 문자씩 문자열 끝까지 읽는 반복분을 만들 때, **종료조건**을 < **src[i] != '\0';** > 으로 사용하면 문자열의 크기를 몰라도 **끝까지 읽는 반복문을 편리하게 만들 수 있음.**

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/19301113-575f-4c63-9902-3b7cac04079b/%EC%B6%9C%EB%A0%A5.pdf

- 자바와 다르게 C는 기본 라이브러리에 sizeof 함수가 없다. 아래 방식으로 sizeof 함수를 간편하게 대체할 수 있다.

```
int size = 0;
while(str[size] != 0){
    size++;
}

printf("문자열\"%s\"의 길이는 %d입니다.\n", str, size);
```

- NULL과 '\0'과 0은 모두 같다. str[i] != 0 으로 써도 무방하다.

#include <ctype.h> → 문자(Char) 처리 라이브러리 알아보기

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/e8b8ec03-be13-40b5-8393-81cadfaffc19/%EC%B6%9C%EB%A0%A5.pdf

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/b2310311-9bca-4007-b327-42a77d2277d8/%EC%B6%9C%EB%A0%A5.pdf

예제 : 단어 세기

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/7e0f6191-19c9-44d4-a1c1-1c0c6da2bfa5/%EC%B6%9C%EB%A0%A5.pdf

```
int count_word (char * s)
{
  int i, wc = 0, waiting = 1;
  for(i = 0; s[i] != NULL; i++) {    // s의 각 글자 조사
    if( isalpha(s[i]) )    // s의 글자가 알파벳이면
    {
      if( waiting )    // 단어의 처음을 기다리고 있으면
      {
        wc++;    // 카운터를 증가
        waiting = 0;    // 단어를 처리하는 중
      }
    }
    else    // 알파벳이 아니면
      waiting = 1;    // 단어를 기다린다.
  }
}
```

이 예제 코드에서 waiting 변수를 조건문에 사용한 방식이 인상 깊어서 작성했다.

위 방식은 flag 변수의 다른 형태로, 다른 문제의 기능 구현 과정에 충분히 사용할 만한 구조로 보인다.

#include <string.h> → 문자열(string) 처리 라이브러리 알아보기

https://prod-files-secure.s3.us-west-2.amazonaws.com/58976c5c-3153-42f0-ac9c-7e9350d612f9/1000c232-6e9d-4fea-bb9e-38fed487841e/%EC%B6%9C%EB%A0%A5.pdf

String copy (문자열 복사)

Syntax: strcpy()

예

```
char dst[6];
char src[6]="Hello";
strcpy(dst, src);       // src를 dst로 복사한다.
```

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

int main(void)
{
  char dst[6];
  char src[6]="Hello";
  strcpy(dst, src);
  printf("%s",dst);

  return 0;
}
```

String concatenation (문자열 연결)

Syntax: strcat()

예

```
char dst[12]= "Hello";
char src[6] = "World";
strcat(dst, src);       // dst가 "HelloWorld"가 된다.
```

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

int main(void)
{
  char dst[13] = "Hello ";
  char src[6] = "World";
  strcat(dst, src);

  printf("%s",dst);
  return 0;
}
```

String compare (문자열 비교)

Syntax: strcmp()

예 `int result = strcmp("dog", "dog");` `// 0이 반환된다.`

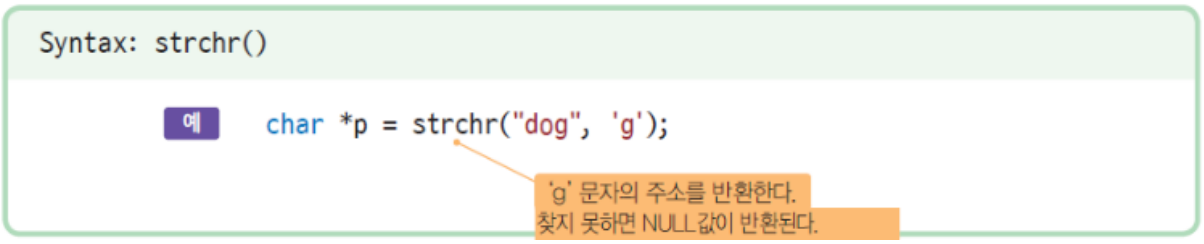
strcmp()는 문자열 s1과 s2를 비교하여 사전적인(lexicographic) 순서에서 s1이 앞에 있으면 음수가 반환되고, 같으면 0이, 뒤에 있으면 양수가 반환된다.

반환값	s1과 s2의 관계
< 0	s1이 s2보다 앞에 있다.
0	s1 == s2
> 0	s1이 s2보다 뒤에 있다.

35

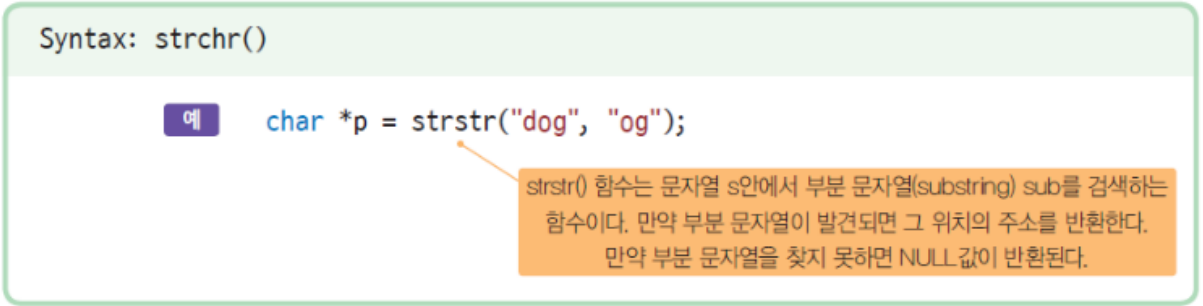
```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char s1[80];
  char s2[80];
  int result;

  printf("첫번째 단어를 입력하시오: ");
  scanf("%s", s1);
  printf("두번째 단어를 입력하시오: ");
  scanf("%s", s2);

  result = strcmp(s1, s2);

  if( result < 0)
    printf("%s가 %s보다 앞에 있습니다.\n",s1, s2);
  else if( result == 0)
    printf("%s가 %s와 같습니다.\n",s1, s2);
  else
    printf("%s가 %s보다 뒤에 있습니다.\n",s1, s2);

  return 0;
}
```

Search char in String(문자열에서 문자 검색)




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

int main(void)
{
  char s[] = "language";
  char c = 'g';
  char *p;
  int loc;

  p = strchr(s, c);
  loc = (int)(p - s); //ex) s의 주소 1001, p의 주소 1004이면, p-s는 3
  if(p != NULL)
    printf("첫번째 %c가 %d에서 발견되었음\n", c, loc);
  else
    printf("%c가 발견되지 않았음\n", c);

  return 0;
}
```

Search String in String (문자열에서 문자열 검색)




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

```c
int main(void)
{
  char s[] = "A joy that's shared is a joy made double";
  char sub[] = "joy";
  char *p;
  int loc;

  p = strstr(s, sub);
  loc = (int)(p - s);
  if ( p != NULL)
    printf("첫번째 %s가 %d에서 발견되었음\n", sub, loc);
  else
    printf("%s가 발견되지 않았음\n", sub);

  return 0;
}
```

strtok (String 토큰 분리)

Syntax:

예

```c
char s[] = "Hello World";
char delimit[] = " ";
char *p = strtok(s, delimit);
```

문자열을 스페이스문자를 사용하여 단어들로 분리한다.

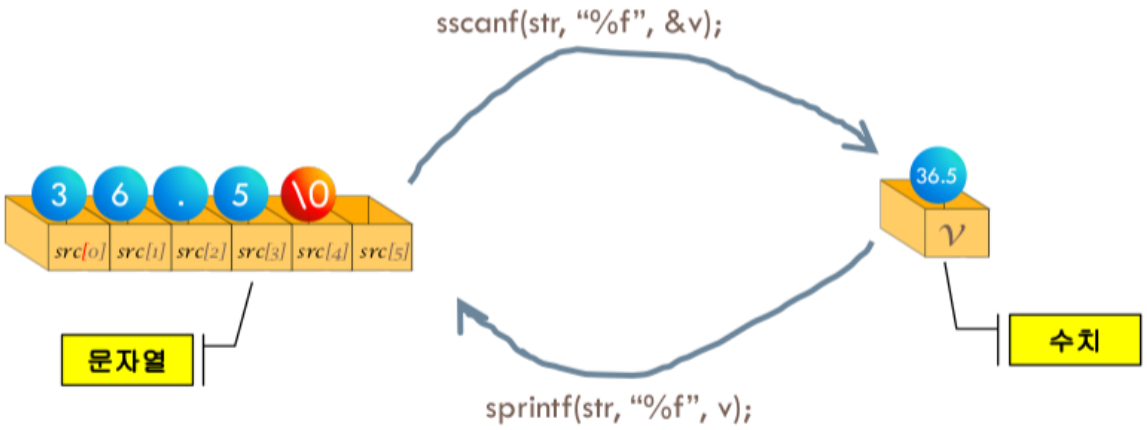
```c
#include <stdio.h>
#include <string.h>

char s[] = "Man is immortal, because he has a soul";
char seps[] = " ,\t\n"; //분리자(separators)
char *token;

int main(void)
{
  //문자열을 전달하고 다음 토큰을 얻는다.
  token = strtok(s, seps);
  while(token != NULL)
  {
    //문자열 s에 토큰이 있는 동안 반복한다.
    printf("토큰: %s\n", token);
    //다음 토큰을 얻는다.
    token = strtok(NULL, seps); //문자열 대신 NULL을 넘겨주면 그 위치부터 다시 탐색 시작
                  //(분리된 그 다음 위치를 기억하기 때문)
```

```
  }
  return 0;
}
```

strtok 함수의 특성

1. **구분자가 연속으로 나올 경우는 무시**
2. **분리된 그 다음 위치를 기억**(문자열 대신 ★★ **NULL을 넘겨주면 그 위치부터 다시 탐색 시작**★★)
3. **검색된 기준 문자는 '\0'로 변환** (NULL값 변환)
4. 주어진 **문자열의 끝에 도달하면 NULL값 반환**

String ↔ 수치 변환



sscanf 함수 : 문자열을 수치로 변환

sprintf 함수 : 수치를 문자열로 변환

```
#include <stdio.h>

int main(void)
{
  char s[] = "A100";
  char ch;
  int value;

  sscanf(s,"%c%d",&ch, &value); //문자열 s를 문자 ch와 정수 value로 각각 변환
  printf("%c, %d\n", ch, value);

  ch++; value++; //문자와 정수를 각각 1씩 증가시킴
```

```
  sprintf(s, "%c%d", ch, value); //문자와 정수를 문자열로 합침
  printf("%s\n",s);
  return 0;
}
```

String → 수치 변환하는 전용함수 (문자열 → 수치만 가능, 역은 불가능)

- stdlib.h에 원형 정의- 반드시 포함

함수	설명
int atoi(const char *str);	str을 int형으로 변환한다.
long atol(const char *str);	str을 long형으로 변환한다.
double atof(const char *str);	str을 double형으로 변환한다.

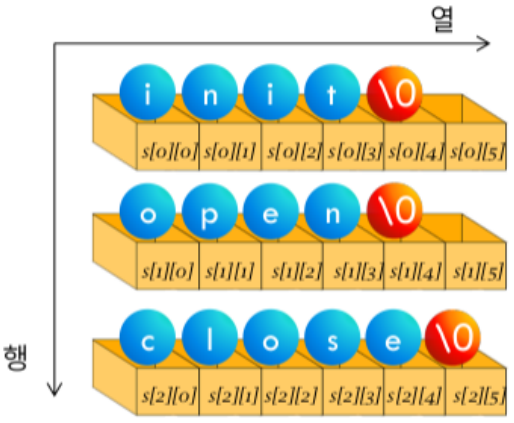
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  char s1[] = "100";
  char s2[] = "12.93";
  char buffer[100];
  int i;
  double d, result;

  i = atoi(s1); //문자열 to 수치(정수) 전용함수
  d = atof(s2); //문자열 to 수치(실수) 전용함수
  result = i + d;

  sprintf(buffer, "%f", result); //수치 to 문자열
  printf("연산 결과는 %s입니다.\n", buffer);
  return 0;
}
```

String 여러 개를 저장하는 방법

1. String의 배열
2. 문자 포인터 배열

(1) String의 배열

```
char s[3][6] = {
    "init",
    "open",
    "close"
};
```



(2) 문자 포인터 배열

```
char *s[3] = {
    "init",
    "open",
    "close"
};
```

- 장점: 서로 길이가 다른 경우 memory를 절약할 수 있음
- 단점: 초기화를 안 한 경우, 실제 string이 들어갈 공간을 확보해야 함.
  - char * s[3];

- 문자 포인터 배열의 **단점에 대한 부가 설명**

  초기화를 안한다면, 문자 포인터 배열 방식을 못 쓰고 **문자열 배열 방식만 사용 가능하다.**

문자 포인터 배열 입력 예제 (초기화를 한 경우)

```
#include <stdio.h>
int main( void )
{
  int i;
  char *menu[5] = {
    "init",
    "open",
    "close",
    "reading",
    "writing"
  };

  for(i = 0; i < 5; i++)
    printf("%d 번째 메뉴: %s \n", i, menu[i]);
}
```

- 주의할 점 : printf 함수에서 간접 참조 연산자 * 사용하면 안된다.
  - %s 자체가 주소값을 받으며, 배열 이름 자체로 문자열(문자 배열)의 주소이기 때문이다.

문자열 배열 입력 예제 (초기화를 안 한 경우)

```
#include <stdio.h>
int main( void )
{
  int i;
  char fruits[3][20]; //초기화를 안 했으므로, 문자열 배열 방식 사용

  for(i = 0; i < 3; i++) {
    printf("과일 이름을 입력하시오: ");
    scanf("%s", fruits[i]); //앞에 &을 붙이면 안됨!
                     //배열 이름 자체로 문자열(문자 배열)의 주소이므로 &를 붙이면 안됨.
  }

  for(i = 0; i < 3; i++)
    printf("%d번째 과일: %s\n", i, fruits[i]);

  return 0;
}
```

- **배열 이름 자체로 안쪽 배열의 주소를 의미한다는 것을 잊지 말자.**

예제 : 영한 사전 구현

예제: 영한 사전의 구현

- 3차원 string array을 이용하여 간단한 한영 사전을 구현하여 보자.

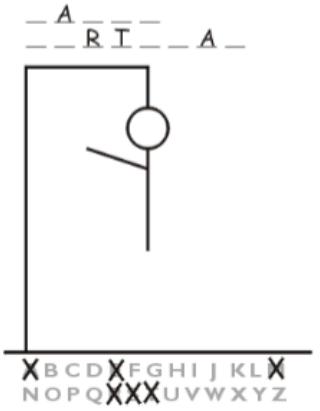

```
#include <stdio.h>
#include <string.h>
#define ENTRIES 5

int main( void )
{
  int i;
  char *dic[ENTRIES][2] = {
    {"book", "책"},
    {"boy", "소년"},
    {"computer", "컴퓨터"},
    {"language", "언어"},
    {"rain", "비"}
  };
  char word[30];
  printf("단어를 입력하시오:");
  scanf("%s",word);

  for(i=0; i<ENTRIES; i++)
  {
    if(strcmp(dic[i][0], word)== 0)
    {
      printf("%s: %s\n", word, dic[i][1]);
      return 0;
    }
  }
  printf("사전에서 발견되지 않았습니다.\n");
```

```
  return 0;
}
```

숙제 12-2 : 행맨 게임

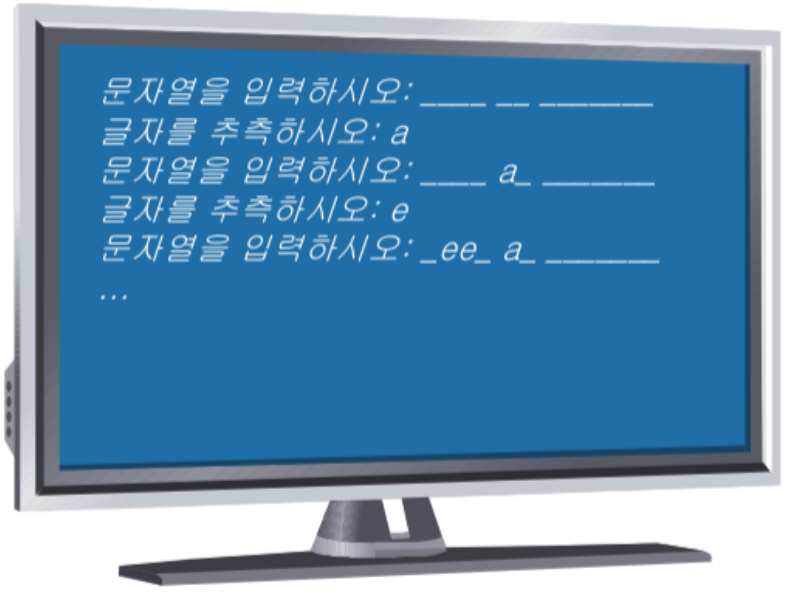
숙제 12-2: 행맨 게임

- 빈칸으로 구성된 문자열이 주어지고 사용자는 문자열에 들어갈 글자들을 하나씩 추측해서 맞추는 게임
- 사용자가 문자열에 들어 있는 글자를 정확하게 입력했으면 화면에 그 글자를 출력한다.
- main() 함수가 잘 작동할 수 있도록 check() 함수를 완성할 것



66

실행 결과




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

int check(char s[], char a[], char ch);
int main (void) {
  char solution[100] = "meet at midnight";
  char answer[100] = "____ __ ________"; // 위의 솔루션과 길이가 일치
  char ch;

  while(1) {
    printf("문자열을 입력하시오: %s\n", answer);
    printf("글자를 추측하시오: ");
    ch = getchar();
    if( check(solution, answer, ch) == 1 ){
      printf("성공: %s \n", answer);
      break;
    }
    while (ch = getchar() != '\n' && ch != EOF) {}; // 버퍼 제거
  }
  return 0;
}

int check(char s[], char a[], char ch){
  int loc=0;
  char *p = NULL;
  while(strchr(s,ch) != NULL){
    p = strchr(s, ch);
    loc = (int)(p-s);
```

```
    s[loc] = '_';
    a[loc] = ch;
  }
  if(strchr(a,'_') == NULL) return 1;
  return -1;
}
```

도전문제 (숙제 아님)

- "meet at midnight"에서 "____ __ _____"을 자동으로 생성할 수 있는가?
- 여러 개의 단어들이 들어 있는 2차원 배열을 생성하고, 그 배열로부터 랜덤하게 문제를 만들 수 있도록 프로그램을 업그레이드하라.
- 일정한 횟수만 시도할 수 있게 하라.

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

#include <stdlib.h> //rand함수의 무작위성 향상을 위한 라이브러리 1
#include <time.h>   //rand함수의 무작위성 향상을 위한 라이브러리 2

int check(char s[], char a[], char ch);
int main (void) {
  srand(time(NULL)); //현재 시간으로 난수 시드 초기화

  char *dictionary[100] = {
    "meet at midnight",
    "believe in yourself",
    "seize the day",
    "love yourself",
    "the die is cast",
    "be brave",
    "hang in there",
    "time is gold"
```

```
  };

  int size=0;
  while(dictionary[size] != NULL) size++; //사전 크기 확인
  int idx = rand()%size; //사전 크기에 맞는 랜덤한 인덱스 생성

  char solution[100]; strcpy(solution, dictionary[idx]); //사전 내 랜덤한 단어를 답으로 지정
  char answer[100];
  char ch;

  int limit = strlen(solution) * 1.5;

  for(int i=0; i < strlen(solution); i++){
    if(isspace(solution[i])) answer[i] = ' ';
    else answer[i] = '_';
  }

  while(limit > 0) {
    printf("남은 시도 가능 횟수: %d번\n",limit);
    printf("문자열을 입력하시오: %s\n", answer);
    printf("글자를 추측하시오: ");
    ch = getchar();
    if( check(solution, answer, ch) == 1 ){
      printf("성공: %s \n", answer);
      break;
    }
    while (ch = getchar() != '\n' && ch != EOF) {}; // 버퍼 제거
    limit--;
  }
  if (limit == 0) printf("\n실패! 정답은 \"%s\" 였습니다.\n",dictionary[idx]);
  return 0;
}

int check(char s[], char a[], char ch){
  int loc=0;
  char *p = NULL;
  while(strchr(s,ch) != NULL){
    p = strchr(s, ch);
    loc = (int)(p-s);
    s[loc] = '_';
    a[loc] = ch;
  }
  if(strchr(a,'_') == NULL) return 1;
  return -1;
}
```

깨달은 점

- 문자열을 잘 다룰 줄 아는 것은 강력한 무기이다.
- 언제든지 다시 찾아와 참고할만한 자료이다.