기초프로그래밍

제16장 문자열 표준 함수와 기타 표준 함수

Sangsoo Lim
CSAI
Dongguk University

차례

- 문자열 처리 함수
- 기타 표준 함수

문자열 처리 함수

- **배울 내용**
  - ① `gets()` 함수와 `puts()` 함수
  - ② `strlen()` 함수
    - 문자열의 길이를 알려준다
  - ③ `strcpy()` 함수와 `strncpy()` 함수
    - 문자열을 복사한다
  - ④ `strcat()` 함수와 `strncat()` 함수
    - 문자열을 결합한다
  - ⑤ `strcmp()` 함수와 `strncmp()` 함수
    - 문자열을 비교한다

문자열 처리 함수

- gets()함수와 puts()함수
  - 헤더파일 : **stdio.h**
  - **gets()함수: 문자열을 입력받는 함수**
  - **puts()함수: 문자열을 출력하는 함수**

함수의 원형	예제	설명
#include<stdio.h> char* gets (char* s)	char array[10]; gets(array);	전달된 메모리 주소에 문자열 저장 **성공:** 입력된 문자열 반환 **실패:** NULL 포인터 반환
#include<stdio.h> int puts (const char* s)	char array[10] = "Good luck"; puts(array);	전달된 메모리 주소의 문자열 출력 **성공:** 0값 또는 0이 아닌값 반환 **실패:** EOF 반환

  - **EOF(End Of File)**
    - 파일의 끝을 의미
    - stdio.h 에 -1로 매크로 상수 정의
    - [ctr+z]를 입력하면 EOF로 인식

문자열 처리 함수

- **`char *fgets(char *str, int n, FILE *stream);`**
  - **str: 문자열 저장할 배열의 포인터**
  - N: 읽어들일 최대 문자수 (최대 n-1개 문자 & null 종결자 \0을 포함)
    - Stream: 입력 스트림(예: stdin은 표준입력)
  - 버퍼 오버플로우를 방지함.
  - 개행문자(\n) 포함.
  - 종료문자 추가: 읽은 문자열의 끝에 자동으로 null 종결자 (\0)을 추가

- **`puts()`**
  - **문자열 출력**
  - 출력하면서 자동으로 [Enter]키(개행 문자 \n)를 삽입
  - 에러가 발생하면 EOF(-1) 반환

문자열 처리 함수

```c
/* 16-1.c */
#include <stdio.h>
#include <string.h>  // strcspn을 사용하기 위한 헤더

int main(void)
{
    char array1[100];                  // 넉넉한 크기의 배열로 변경
    const char array2[] = "Good luck"; // 상수 문자열은 const 사용

    puts("문자열을 입력하세요:");

    // 안전한 입력 (최대 99자 + null terminator)
    if (fgets(array1, sizeof(array1), stdin) != NULL) {
        // fgets로 입력된 개행 문자 제거
        array1[strcspn(array1, "\n")] = '\0';

        puts(array1);      // 입력 문자열 출력
        puts(array2);      // 배열에 저장된 문자열 출력
        puts("Good luck"); // 리터럴 문자열 출력
    } else {
        puts("입력 오류가 발생했습니다.");
    }

    return 0;
}
```

문자열 처리 함수

```c
/* 16-2.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[100]; // 넉넉한 크기의 버퍼로 변경
    char array2[100];

    // 첫 번째 문자열 입력 (fgets 사용)
    printf("Enter a string: ");
    if (fgets(array1, sizeof(array1), stdin) != NULL) {
        array1[strcspn(array1, "\n")] = '\0'; // 개행 문자 제거
        printf("You entered: %s\n", array1);
    } else {
        printf("Input error for array1.\n");
    }

    // 두 번째 문자열 입력 (fgets로 통일)
    printf("Enter another string: ");
    if (fgets(array2, sizeof(array2), stdin) != NULL) {
        array2[strcspn(array2, "\n")] = '\0'; // 개행 문자 제거
        printf("You entered: %s\n", array2);
    } else {
        printf("Input error for array2.\n");
    }

    return 0;
}
```

문자열 처리 함수

- **strlen()함수**
  - **헤더파일: string.h**
  - **문자열의 길이를 알려주는 함수**
  - **주의 사항**
    - 문자열의 끝을 알리는 종료문자(\0)는 길이에 포함되지 않는다.

함수의 원형	예제	설명
`#include<string.h>` `size_t strlen (const char* s)`	`char array[10] = “Good luck”;` `strlen(array);`	전달된 메모리 주소 array 부터 종료문자를 만날때 까지 저장된 **문자열의 길이를 반환**

문자열 처리 함수

```c
/* 16-3.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[] = "Hello C";      // ASCII 문자로 구성된 문자열
    char array2[] = "안녕하세요";     // UTF-8 인코딩의 한글 문자열

    // strlen: 문자열의 실제 문자 수 반환 (널 문자 제외, 한글 한 글자는 2~3바이트)
    printf("영문 문자열의 문자 수: %zu\n", strlen(array1)); // 7 출력
    printf("한글 문자열의 바이트 수: %zu\n", strlen(array2)); // 15 출력 (한글 5글자 × 3바이트)

    // sizeof: 배열의 전체 크기 (널 문자 포함)
    printf("영문 문자열의 바이트 크기: %zu\n", sizeof(array1)); // 8 출력 (7 + 1)
    printf("한글 문자열의 바이트 크기: %zu\n", sizeof(array2)); // 16 출력 (15 + 1)

    return 0;
}
```

문자열 처리 함수

- **strcpy()함수와 strncpy()함수 - 문자열을 복사한다.**
  - 헤더파일: **string.h**
  - **strcpy()함수**
    - **두 번째 인자의 문자열**을 **첫 번째 인자의 메모리 주소**에 **복사**
  - **strncpy()함수:**
    - **두 번째 인자의 문자열**을 **첫 번째 인자의 메모리 주소**에 **복사**
    - **세 번째 인자**는 복사해야 할 크기

함수의 원형	예제	설명

`#include<string.h>`
`char* strcpy (char* dest, const char* src)`
	`char array1[10] = "Good luck";`
	`char array2[10];`
	`strcpy(array2, array1);`
		array1의 문자열을 array2에 복사,
		성공: 복사된 문자열의 시작 주소 반환

`#include<string.h>`
`char* strncpy (char* dest, const char* src, size_t n)`
	`char array1[10] = "Good luck";`
	`char array2[10];`
	`strncpy(array2, array1, 3);`
		array1의 문자열을 array2에 3byte 만큼 복사,
		성공: 복사된 문자열의 시작 주소 반환

문자열 처리 함수

```c
/* 16-4.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    const char array1[] = "Hello world"; // 원본 문자열 (자동 '\0' 포함)
    char array2[sizeof(array1)];         // strcpy 목적지
    char array3[sizeof(array1)];         // strncpy 목적지

    // strcpy: 널 문자 포함 복사 (주의: 복사 대상은 충분히 커야 함)
    strcpy(array2, array1);

    // strncpy: 명시적 널 문자 보장 필요
    strncpy(array3, array1, sizeof(array3) - 1);
    array3[sizeof(array3) - 1] = '\0'; // 수동으로 널 문자 추가

    // 결과 출력
    puts("array2 (strcpy 결과):");
    puts(array2);

    puts("array3 (strncpy 결과):");
    puts(array3);

    return 0;
}
```

문자열 처리 함수

```c
/* 16-5.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[6] = "Hello"; // 원본 문자열 (널 문자 포함 총 6바이트)
    char array2[4];           // 목적 배열 (3문자 + 널 문자)

    // array1에서 앞의 3글자만 복사
    strncpy(array2, array1, 3);
    array2[3] = '\0'; // 수동으로 널 문자 추가 (strncpy는 보장 안됨)

    // 결과 출력
    printf("array2 (앞 3글자 복사): %s\n", array2); // "Hel" 출력

    return 0;
}
```

문자열 처리 함수

- **strcat()함수와 strncat()함수 – 문자열을 결합한다.**
  - 헤더파일: **string.h**
  - **strcat()**
    - **두 번째 인자의 문자열**을 **첫 번째 인자의 메모리 주소**에 **결합**
  - **strncat()**
    - **두 번째 인자의 문자열**을 **첫 번째 인자의 메모리 주소**에 **결합**
    - **세 번째 인자**는 결합해야 할 크기

함수의 원형	예제	설명

`#include<string.h>`
`char* strcat (char* dest,`
`const char* src)`
	`char array1[10] = "Good";`
	`char array2[5] = "luck";`
	`strcat(array1, array2);`
		array1 에 array2 문자열을 결합
		성공: 결합된 문자열의 시작 주소 반환

`#include<string.h>`
`char* strncat (char* dest,`
`const char* src, size_t n)`
	`char array1[10] = "Good";`
	`char array2[5] = "luck";`
	`strncat(array1, array2, 3);`
		array1에 array2 문자열을 3byte 만큼 결합,
		성공: 결합된 문자열의 시작 주소 반환

문자열 처리 함수

```c
/* 16-6.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[100]; // 결과를 저장할 문자열 (충분한 공간 확보)
    char array2[50];  // 결합할 두 번째 문자열

    // 첫 번째 문자열 입력
    printf("첫 번째 문자열 입력: ");
    if (fgets(array1, sizeof(array1), stdin) != NULL) {
        array1[strcspn(array1, "\n")] = '\0'; // 개행 문자 제거
    }

    // 두 번째 문자열 입력
    printf("두 번째 문자열 입력: ");
    if (fgets(array2, sizeof(array2), stdin) != NULL) {
        array2[strcspn(array2, "\n")] = '\0'; // 개행 문자 제거
    }

    // 결합 가능 여부 확인 (널 문자 고려)
    size_t total_length = strlen(array1) + strlen(array2);
    if (total_length < sizeof(array1)) {
        strcat(array1, array2); // 문자열 결합
    } else {
        printf("!! 문자열 결합 시 크기 초과 발생! (최대 %zu바이트)\n", sizeof(array1) - 1);
        return 1; // 비정상 종료
    }

    // 결과 출력
    printf("결합된 문자열: %s\n", array1);

    return 0;
}
```

문자열 처리 함수

```c
/* 16-7.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[100]; // 결과 문자열 저장용 버퍼
    char array2[50];  // 추가할 문자열 버퍼

    // 첫 번째 문자열 입력
    printf("첫 번째 문자열 입력: ");
    if (fgets(array1, sizeof(array1), stdin) != NULL) {
        array1[strcspn(array1, "\n")] = '\0'; // 개행 문자 제거
    }

    // 두 번째 문자열 입력
    printf("두 번째 문자열 입력: ");
    if (fgets(array2, sizeof(array2), stdin) != NULL) {
        array2[strcspn(array2, "\n")] = '\0'; // 개행 문자 제거
    }

    // 최대 6바이트만 array2에서 복사 → 현재 길이 + 6 + '\0'이 array1 크기를 넘지 않도록 체크
    size_t curr_len = strlen(array1);
    size_t max_append = 6;

    if (curr_len + max_append < sizeof(array1)) {
        strncat(array1, array2, max_append); // 최대 6바이트만 덧붙이기 (널 문자 자동 추가)
    } else {
        printf("!! 결합 시 배열 크기를 초과합니다! (최대 %zu 바이트 허용)\n", sizeof(array1) - 1);
        return 1;
    }

    // 결과 출력
    printf("전체 문자열 출력: %s\n", array1);

    return 0;
}
```

문자열 처리 함수

- **strcmp()함수와 strncmp()함수 – 문자열을 비교한다.**
  - 헤더파일 : **string.h**
  - **strcmp()**
    - **첫 번째 인자의 문자열**과 **두 번째 인자의 문자열**을 **비교**
  - **strncmp()**
    - **첫 번째 인자의 문자열**과 **두 번째 인자의 문자열**을 **비교**
    - **세 번째 인자**는 비교해야 할 크기

함수의 원형	예제	설명

`#include<string.h>`
`int strcmp (const char* s1,`
`const char* s2)`
	`char array1[10] = "Good";`
	`char array2[10] = "luck";`
	`strcmp(array1, array2);`
		array1의 문자열과 array2의 문자열을 비교합니다.
		성공: array1 과 array2의 비교 결과를 반환

`#include<string.h>`
`int strncmp (const char* s1,`
`const char* s2, size_t n)`
	`char array1[10] = "Good";`
	`char array2[10] = "luck";`
	`strncmp(array1, array2, 3);`
		array1의 문자열과 array2의 문자열을 3개까지 비교합니다.
		성공: array1 과 array2의 비교 결과를 반환

문자열 처리 함수

- **strcmp()**
  - **첫 번째 인자의 문자열**과 **두 번째 인자의 문자열**을 **비교**
- **strncmp()**
  - **첫 번째 인자의 문자열**과 **두 번째 인자의 문자열**을 **비교**
  - **세 번째 인자**는 비교해야 할 크기

- **비교 결과**

반환값	설명
양수(0보다 큰 값)	array1의 문자열이 array2의 문자열보다 크다.
0	array1의 문자열이 array2의 문자열과 같다.
음수(0보다 작은 값)	array1의 문자열이 array2의 문자열보다 작다.

문자열 처리 함수

```c
/* 16-8.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[20] = "Good-morning";
    char array2[20] = "Good-afternoon";
    char array3[20] = "Good-evening";
    int result1, result2, result3;

    // 문자열 전체 비교: array1 vs array2
    result1 = strcmp(array1, array2);

    // 앞의 5글자만 비교: array1 vs array2 ("Good-")
    result2 = strncmp(array1, array2, 5);

    // 문자열 전체 비교: array2 vs array3
    result3 = strcmp(array2, array3);

    // 결과 출력
    printf("strcmp(array1, array2): %d → \"%s\" vs \"%s\"\n", result1, array1, array2);
    printf("strncmp(array1, array2, 5): %d → 앞 5글자 비교\n", result2);
    printf("strcmp(array2, array3): %d → \"%s\" vs \"%s\"\n", result3, array2, array3);

    return 0;
}
```

문자열 처리 함수

- **strchr()함수와 strstr()함수**
  - 헤더파일: string.h
  - strchr()함수: 문자의 위치를 찾는 함수
  - strstr()함수: 문자열의 위치를 찾는 함수

함수의 원형	예제	성공
#include<string.h> char* strchr (const char* s, int c)	char array1[10] = "Good"; strchr(array1, 'd');	array1에서 문자 'd'의 메모리 주소를 찾는다. **성공:** 찾은 **문자**의 메모리 주소를 반환
#include<string.h> char* strstr (const char* s1, const char* s2)	char array1[10] = "Good-morning"; char array2[10] = "morning"; strstr (array1, array2);	array1에서 array2에 저장된 문자열을 찾는다. **성공:** 찾은 **문자열**의 메모리 주소를 반환

문자열 처리 함수

```c
/* 16-9.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array1[50] = "Good-morning, Good-afternoon, Good-evening";
    char array2[10] = "morning";
    char *p1 = NULL;
    char *p2 = NULL;

    // 문자 'a'의 첫 번째 위치 찾기
    p1 = strchr(array1, 'a');

    // 문자열 "morning"의 시작 위치 찾기
    p2 = strstr(array1, array2);

    // 결과 출력 - strchr
    if (p1 != NULL) {
        printf("문자 'a'의 주소: %p\n", (void *)p1);
        printf("문자 'a' 이후 문자열: %s\n", p1);
    } else {
        printf("문자 'a'를 찾을 수 없습니다.\n");
    }

    printf("-------------------------------------------------\n");

    // 결과 출력 - strstr
    if (p2 != NULL) {
        printf("문자열 \"morning\"의 시작 주소: %p\n", (void *)p2);
        printf("문자열 \"morning\" 이후 문자열: %s\n", p2);
    } else {
        printf("문자열 \"morning\"을 찾을 수 없습니다.\n");
    }

    return 0;
}
```

문자열 처리 함수

- **sscanf()함수와 sprintf()함수**
  - 헤더파일: **string.h**
  - **sscanf()함수: 배열(메모리)로부터 문자열을 입력받는 함수**
  - **sprintf()함수: 배열(메모리)에 문자열을 출력하는 함수**

함수의 원형	설명
`#include<string.h>` `int sscanf (const char * s1, const char * s2, ...);`	메모리에서 데이터를 입력 받는다. **성공:** 데이터의 개수 반환
`#include<string.h>` `int sprintf (char * s1, const char * s2, ...);`	메모리에 데이터를 출력한다. **성공:** 문자열의 길이 반환

문자열 처리 함수

```c
/* 16-10.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    // 입력 문자열 (공백으로 구분된 세 개의 항목)
    char array[50] = "100 3.14 good-morning";

    // 파싱 결과를 저장할 변수들
    int num1;          // 정수 저장용
    double num2;       // 실수 저장용
    char str[50];      // 문자열 저장용

    // 문자열 파싱: array에서 세 항목 추출
    sscanf(array, "%d %lf %s", &num1, &num2, str);

    // 결과 출력
    puts("파싱 결과:");
    printf("정수: %d\n", num1);
    printf("실수: %.2lf\n", num2);
    printf("문자열: %s\n", str);

    return 0;
}
```

문자열 처리 함수

```c
/* 16-12.c */
#include <stdio.h>
#include <string.h>

int main(void)
{
    char array[100];                // 포맷팅된 문자열 저장용
    int num1 = 100;                 // 정수
    double num2 = 3.14;             // 실수
    char str[50] = "good-morning"; // 문자열

    // 1. 직접 화면에 출력
    printf("직접 출력 → %d, %.2lf, %s\n", num1, num2, str);

    // 2. 문자열로 포맷팅하여 배열에 저장 (보안 강화를 위해 snprintf 사용)
    snprintf(array, sizeof(array), "%d, %.2lf, %s\n", num1, num2, str);

    // 3. 배열 내용을 출력
    printf("배열에 저장된 문자열:\n%s", array);

    return 0;
}
```

기타 표준 함수

- **배울 내용**
  - ① 데이터 변환 표준 함수
  - ② 수학 관련 표준 함수

기타 표준 함수

- **데이터를 변환하는 함수**
  - 헤더파일: **stdlib.h**

함수의 원형	설명
double **atof** (const char* str);	문자열을 **double형** 데이터로 변환
int **atoi** (const char* str);	문자열을 **int형** 데이터로 변환
long **atol** (const char* str);	문자열을 **long형** 데이터로 변환

기타 표준 함수

```c
/* 16-13.c */
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    // 숫자를 나타내는 문자열 입력
    const char* str1 = "3.14";
    const char* str2 = "100";
    const char* str3 = "10000000";

    // 변환된 값을 저장할 변수
    double num1;
    int num2;
    long num3;

    // 문자열을 숫자로 변환
    num1 = atof(str1);  // 문자열 → 실수 (double)
    num2 = atoi(str2);  // 문자열 → 정수 (int)
    num3 = atol(str3);  // 문자열 → 긴 정수 (long)

    // 결과 출력
    printf("변환된 실수 값: %lf\n", num1);
    printf("변환된 정수 값: %d\n", num2);
    printf("변환된 긴 정수 값: %ld\n", num3);

    return 0;
}
```

기타 표준 함수

- **데이터를 변환하는 함수**
  - 헤더파일: **ctype.h**

함수의 원형	설명
int **toascii** (int num);	문자를 **ASCII 문자**로 변환
int **tolower** (int num);	문자를 **소문자**로 변환
int **toupper** (int num);	문자를 **대문자**로 변환

기타 표준 함수

```c
/* 16-14.c */
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char a1 = 'A';
    char a2 = 'a';

    // 문자 'A'의 아스키 코드 값을 출력
    printf("아스키 코드 값: %d\n", toascii(a1));  // 65 출력

    // 문자 'a'를 소문자로 변환 (이미 소문자라 그대로 출력됨)
    printf("소문자 변환 결과: %c\n", tolower(a2));  // a 출력

    // 문자 'a'를 대문자로 변환
    printf("대문자 변환 결과: %c\n", toupper(a2));  // A 출력

    return 0;
}
```

기타 표준 함수

- **수학 관련 함수**
  - 헤더파일: **math.h**

함수의 원형	설명
double **ceil** (double x);	x보다 큰 정수 반환
double **floor** (double x);	x보다 작은 정수 반환
double **fabs** (double x);	x의 절댓값 반환
double **pow** (double x, double y);	x^y
double **sqrt** (double x);	$\sqrt{x}$
double **exp** (double x);	e^x e는 자연 상수 (오일러의 수)
double **log** (double x);	$\log_e x$
double **log10** (double x);	$\log_{10} x$

기타 표준 함수

gcc test.c -o test -lm

```c
/* 16-15.c */
#include <stdio.h>
#include <math.h>

int main(void)
{
    double a1 = 3.14;
    double a2 = -3.14;

    // 올림(ceil), 내림(floor)
    printf("Ceil 함수 결과 (%.2lf): %.2lf\n", a1, ceil(a1));
    printf("Floor 함수 결과 (%.2lf): %.2lf\n", a1, floor(a1));
    printf("----------------------------\n");

    // 절댓값, 제곱, 제곱근
    printf("절댓값 (%.2lf): %.2lf\n", a2, fabs(a2));
    printf("2의 8제곱: %.2lf\n", pow(2, 8));
    printf("2의 제곱근: %.2lf\n", sqrt(2));
    printf("----------------------------\n");

    // 지수 함수, 자연로그, 상용로그
    printf("지수 함수 exp(1): %.2lf\n", exp(1));
    printf("자연로그 log(exp(1)): %.2lf\n", log(exp(1)));
    printf("상용로그 log10(10): %.2lf\n", log10(10));
    printf("----------------------------\n");

    return 0;
}
```

기타 표준 함수

- **math.h**에 있는 여러 삼각함수

함수의 원형	설명
double **sin** (double x);	삼각함수에서 x의 sin 값
double **cos** (double x);	삼각함수에서 x의 cos 값
double **tan** (double x);	삼각함수에서 x의 tan 값
double **sinh** (double x);	삼각함수에서 x의 sinh 값
double **cosh** (double x);	삼각함수에서 x의 cosh 값
double **tanh** (double x);	삼각함수에서 x의 tanh 값
double **asin** (double x);	삼각함수에서 x의 asin 값
double **acos** (double x);	삼각함수에서 x의 acos 값
double **atan** (double x);	삼각함수에서 x의 atan 값

기타 표준 함수

- **rand()함수와 srand()함수 -난수를 생성 시킨다.**
  - 헤더파일: **stdlib.h**
  - **rand()함수: 난수를 생성시키는 함수**
    - **한가지 패턴으로 난수를 생성 시킨다.**
  - **srand()함수: 난수의 패턴을 생성시키는 함수**
    - **여러 가지 패턴으로 난수를 생성 시킨다.**

함수의 원형	설명
`int rand (void);`	난수를 생성
`int srand (unsigned int seed);`	seed를 지정하여 난수를 생성 (seed: 난수 생성 패턴)

기타 표준 함수

```c
/* 16-16.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    int i = 0;

    // 현재 시간을 기반으로 난수 시드 설정
    srand((unsigned int)time(NULL));

    // 안내 메시지 출력
    printf("10개의 난수를 발생시킵니다:\n");

    // 난수 10개 생성 및 출력
    while (i < 10)
    {
        printf("%d\n", rand());
        i++;
    }

    return 0;
}
```

```
10개의 난수를 발생시킵니다.
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
1189641421
```

기타 표준 함수

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

int main(void)
{
    int i;
    int n = 5;       // 생성할 난수 개수
    time_t t;

    // 현재 시간을 기반으로 난수 시드 초기화
    srand((unsigned)time(&t));

    // 안내 메시지 출력
    printf("%d개의 랜덤 숫자를 생성합니다 (0부터 49까지):\n", n);

    // 난수 생성 및 출력
    for (i = 0; i < n; i++)
    {
        printf("%d\n", rand() % 50);  // 0 이상 49 이하의 난수 출력
    }

    return 0;
}
```

Summary

- 다양한 문자열 처리 함수들
- 기타 표준 함수
  - 데이터 변환 함수
  - 수학함수