Reference 01. 수학 관련 함수

▌abs( ) 함수 : 절대값을 구함

헤더파일	#include <math.h>
함수원형	int abs(int x);

사용예제	int ix = -4, iy;
	iy = abs(ix);
	printf_s("The absolute value of %d is %d₩n", ix, iy);

실행결과	The absolute value of -4 is 4
함수설명	주어진 정수 x의 절대값을 구함

▌log( ), log10( ) 함수 : 로그값을 구함

헤더파일	#include <math.h>
함수원형	double log(double x); double log10(double x);

사용예제	double x = 9000.0;
	double y;
	y = log(x);
	printf("log(%.2f) = %f₩n", x, y);
	y = log10(x);
	printf("log10(%.2f) = %f₩n", x, y);

실행결과	log(9000.00) = 9.104980 log10(9000.00) = 3.954243
함수설명	자연로그($\log_e x$)와 상용로그($\log_{10} x$) 값을 구함

▌sqrt( ) 함수 : 제곱근을 구함

헤더파일	#include <math.h>
함수원형	double sqrt(double x);

사용예제	double question = 45.35, answer;
	answer = sqrt(question);
	printf("The square root of %.2f is %.2f₩n", question, answer);

실행결과	The square root of 45.35 is 6.73
함수설명	제곱근($\sqrt{x}$)을 구함

▌pow( ) 함수 : 제곱값(x^y)을 구함

헤더파일	#include <math.h>
함수원형	double pow(double x, double y);

사용예제	double x = 2.0, y = 3.0, z;
	z = pow(x, y);
	printf("%.1f to the power of %.1f is %.1f₩n", x, y, z);

실행결과	2.0 to the power of 3.0 is 8.0
함수설명	x의 y승을 구함

▌acos( ), asin( ), atan( ) 함수 : 아크 코사인, 아크 사인, 아크 탄젠트를 구함

헤더파일	#include <math.h>

함수원형	double acos(double x);
	double asin(double y);
	double atan(double z);

함수설명	아크 코사인과 아크 사인, 아크 탄젠트의 값을 구함

▌fabs( ) 함수 : 실수의 절대값을 구함

헤더파일	#include <math.h>
함수원형	double fabs(double x);
함수설명	주어진 실수의 값에 대한 절대값을 구함

▌ceil( ), floor( ) 함수 : 소수점 이하의 값 - 올림 ceil( ) 함수와 버림 floor( ) 함수

헤더파일	#include <math.h>
함수원형	double ceil(double x); double floor(double x);

사용예제	double y;
	y = floor(2.8);
	printf("The floor of 2.8 is %f₩n", y);
	y = floor(-2.8);
	printf("The floor of -2.8 is %f₩n", y);
	y = ceil(2.8);
	printf("The ceil of 2.8 is %f₩n", y);
	y = ceil(-2.8);
	printf("The ceil of -2.8 is %f₩n", y);

실행결과	The floor of 2.8 is 2.000000 The floor of -2.8 is -3.000000 The ceil of 2.8 is 3.000000 The ceil of -2.8 is -2.000000
함수설명	ceil 함수는 x보다 작지 않은 최소의 정수를 구하고, floor 함수는 x보다 크지 않은 최대의 정수를 구함

▌sin( ), cos( ), tan( ) 함수 : sine, cosine, tanget 값을 구함

헤더파일	#include <math.h>

함수원형	double sin(double x);
	double cos(double x);
	double tan(double x);

사용예제	double pi = 3.1415926535;
	double x, y;
	x = pi / 2;
	y = sin(x);
	printf("sin(%f) = %f₩n", x, y);
	y = cos(x);
	printf("cos(%f) = %f₩n", x, y);

실행결과	sin(1.570796) = 1.000000 cos(1.570796) = 0.000000
함수설명	사인, 코사인, 탄젠트의 값을 구함

▌rand( ), srand( ) 함수 : 난수 발생

헤더파일	#include <math.h> #include <stdlib.h>
함수원형	int rand(void); void srand(unsigned int seed);

사용예제	#include <stdio.h>
	#include <time.h>
	#include <stdlib.h>
	int main(void)
	{
	  int i;
	
	  // Seed the random-number generator with current time
	  srand((unsigned)time(NULL)); // system time을 seed로
	  // Display 10 numbers.
	  for(i = 0; i < 10;i++)
	  printf(" %6d₩n", rand()); // 0 to RAND_MAX(32767) // **1번**
	  printf("₩n");
	
	  // Usually, you will want to generate a number in a specific range,
	  // such as 0 to 100, like this:
	  {

```
        int RANGE_MIN = 0;
        int RANGE_MAX = 100;
        for (i = 0;      i < 10; i++ )
         {
            int rand100 = (((double) rand() /
                           (double) RAND_MAX) * RANGE_MAX + RANGE_MIN);
            printf( "   %6d₩n", rand100);    // 2번
         }
      }
}
```

실행결과

실행 결과(1번)

```
24052
20577
 2235
29883
26046
22303
19311
 5143
 3208
 8804
```

실행 결과(2번)

```
49
90
91
16
21
16
91
68
30
31
```

함수설명

rand 함수는 0에서 RAND_MAX(32767)까지의 난수를 발생하고, srand 함수는 난수의 seed 번호(수)를 지정

Reference 02. 문자열 조작 함수

strcpy( ) 함수 : 문자열을 복사

헤더파일	#include <string.h>
함수원형	char *strcpy(char *str1, const char *str2);
함수설명	문자열 str2를 문자열 str1에 복사함

strcat( ) 함수 : 문자열을 연결

헤더파일	#include <string.h>
함수원형	char *strcat(char *str1, const char *str2);

사용예제	char string[80];
	strcpy(string, "Hello world from ");
	strcat(string, "strcpy ");
	strcat(string, "and ");
	strcat(string, "strcat!");
	printf("String = %s₩n", string);

실행결과	String = Hello world from strcpy and strcat!
함수설명	문자열 str1의 끝에 문자열 str2를 연결함

▌strlen( ) 함수 : 문자열의 길이 구함

헤더파일	#include <string.h>
함수원형	size_t strlen(const char *str);
함수설명	문자열 str의 길이를 byte 수로 구함

▌strcmp( ) 함수 : 문자열을 비교하여 해당 값을 반환

헤더파일	#include <string.h>
함수원형	int strcmp(const char *str1, const char *str2);

사용예제	#include <string.h>
	#include <stdio.h>
	#include <stdlib.h>
	char string1[] = "The quick brown dog jumps over the lazy fox";
	char string2[] = "The QUICK brown dog jumps over the lazy fox";
	int main(void)
	{
	  char tmp[15];
	  int result;
	  printf("Compare strings:₩n %s₩n %s₩n₩n", string1, string2);
	  result = strcmp(string1, string2);
	  if(result > 0)
	  strcpy(tmp, "greater than");
	  else if(result < 0)
	  strcpy(tmp, "less than");
	  else
	  strcpy(tmp, "equal to");
	  printf(" strcmp: String 1 is %s string 2₩n", tmp);
	}

실행결과	Compare strings:
	  The quick brown dog jumps over the lazy fox
	  The QUICK brown dog jumps over the lazy fox
	  strcmp: String 1 is greater than string 2

함수설명	두 개의 문자열 str1과 str2를 비교함 (str1이 크면 양수, str2가 크면, 같으면 0을 반환)

▌strchr( ) 함수 : 문자열에서 지정한 문자의 마지막 위치 반환

헤더파일	#include <string.h>
함수원형	char *strchr(const char *string, int c);

사용예제	#include <string.h>
	#include <stdio.h>
	int ch = 'r';
	char string[] = "The quick brown dog jumps over the lazy fox";
	char fmt1[] = " 1 2 3 4 5";
	char fmt2[] = "12345678901234567890123456789012345678901234567890";
	int main(void)
	{
	char *pdest;
	int result;
	
	printf("String to be searched:₩n %s₩n", string);
	printf(" %s₩n %s₩n₩n", fmt1, fmt2);
	printf("Search char: %c₩n", ch);
	
	// Search forward.
	pdest = strchr(string, ch);
	result = (int)(pdest - string + 1);
	if (pdest != NULL)
	printf("Result: first %c found at position %d₩n",
	ch, result);
	else
	printf("Result: %c not found₩n");
	// Search backward.
	pdest = strrchr(string, ch);
	result = (int)(pdest - string + 1);
	if (pdest != NULL)
	printf("Result: last %c found at position %d₩n", ch, result);
	else
	printf("Result:₩t%c not found₩n", ch);
	}

실행결과	String to be searched: The quick brown dog jumps over the lazy fox 1 2 3 4 5 12345678901234567890123456789012345678901234567890 Search char: r Result: first r found at position 12 Result: last r found at position 30
함수설명	문자열 string에서 포함된 문자 c의 마지막 위치를 반환 (발견하지 못하면 NULL을 반환)

▌strstr( ) 함수 : 포함되어 있을 경우 시작 포인터를 반환

헤더파일	#include <string.h>
함수원형	char *strstr(const char *str1, const char *str2);

예 제	#include <string.h>
	#include <stdio.h>
	char str[] = "lazy";
	char string[] = "The quick brown dog jumps over the lazy fox";
	char fmt1[] = " 1 2 3 4 5";
	char fmt2[] = "12345678901234567890123456789012345678901234567890";
	int main(void)
	{
	char *pdest;
	int result;
	printf("String to be searched:₩n %s₩n", string);
	printf(" %s₩n %s₩n₩n", fmt1, fmt2);
	pdest = strstr(string, str);
	result = (int)(pdest - string + 1);
	if (pdest != NULL)
	printf("%s found at position %d₩n", str, result);
	else
	printf("%s not found₩n", str);
	}

실행결과	Output String to be searched: The quick brown dog jumps over the lazy fox 1 2 3 4 5 12345678901234567890123456789012345678901234567890 lazy found at position 36
함수설명	문자열 str2가 문자열 str1에 포함되어 있다면 그 시작 포인터를 반환하고, 아니면 NULL을 반환

▌strtok( ) 함수 : 문자열에서 토큰을 추출하여 반환

헤더파일	#include <string.h>
함수원형	char *strtok(char *strToken, const char *strDelimit);

사용예제	#include <string.h>
	#include <stdio.h>
	char string[] = "A string₩tof ,,tokens₩nand some more tokens";
	char seps[] = " ,₩t₩n"; // 분리자 : blank, comma, tab, new line 설정
	char *token;
	int main(void)
	{
	  printf("Tokens:₩n");
	  // Establish string and get the first token
	  token = strtok(string, seps);
	  while(token != NULL)
	  {
	  // While there are tokens in "string"
	  printf(" %s₩n", token);
	  // Get next token:
	  token = strtok(NULL, seps);
	  }
	}

실행결과	Tokens: A string of tokens and some more tokens
함수설명	문자열(strToken)에서 토큰을 추출하여 반환하고, 토큰은 분리자(strDelimit)에 의하여 구분이 되며, 더 이상 토큰이 없으면 NULL값을 반환

Reference 03. 문자 분류 함수

▌ isalnum( ) 함수 : 영문자이거나 숫자인지를 검사

헤더파일	#include <ctype.h>
함수원형	int isalnum(int c);
함수설명	영문자(A~Z, a~z)이거나 숫자(0~9)인지 검사하고, 영문자 또는 숫자이면 0 이외의 값(nonzero)을 반환

▌ isalpha( ) 함수 : 영문자인지를 검사

헤더파일	#include <ctype.h>
함수원형	int isalpha(int c);
함수설명	영문자(A~Z, a~z)인지를 검사하고, 영문자이면 0 이외의 값(nonzero)을 반환 (그렇지 않은 경우에는 0을 반환)

▌ isdigit( ) 함수 : 숫자인지를 검사

헤더파일	#include <ctype.h>
형 식	int isdigit(int c)
함수설명	숫자(0~9)인지 검사하고, 숫자이면 0 이외의 값(nonzero)을 반환 (그렇지 않으면 0을 반환)

▌ iscntrl( ) 함수 : 제어 문자인지를 검사

헤더파일	#include <ctype.h>

함수원형	int iscntrl(int c);
함수설명	제어 문자(0x00~0x1F or 0x7F)인지를 검사. 제어 문자(control character)이면 0 이외의 값(nonzero)을 반환하고, 아니면 0을 반환

▌islower( ) 함수 : 소문자인지를 검사

헤더파일	#include <ctype.h>
함수원형	int islower(int c);
함수설명	소문자(a~z)인지 검사하여 소문자이면 0 이외의 값(nonzero)을 반환하고, 그렇지 않으면 0을 반환

▌isupper( ) 함수 : 대문자인지를 검사

헤더파일	#include <ctype.h>
함수원형	int isupper(int c);
함수설명	대문자(A~Z)인지 검사하여 소문자이면 0 이외의 값(nonzero)을 반환하고, 그렇지 않으면 0을 반환

▌isspace( ) 함수 : 공백 문자인지를 검사

헤더파일	#include <ctype.h>
함수원형	int isspace(int c);
함수설명	공백 문자인지 검사하여 공백 문자이면 0 이외의 값(nonzero)을 반환하고, 그렇지 않으면 0을 반환. 공백문자는 공백, '\t', '\n' 등으로 아스키 코드값 (0x09 ~ 0x0D or 0x20)

Reference 04. 자료 변환 함수

▌atof( ) 함수 : 숫자로 된 문자열을 실수형으로 변환

헤더파일	#include <stdlib.h>
함수원형	double atof(const char *str);
사용예제	str = " 333640273517170716032O "; value = atof(str); printf("Function: atof(₩"%s₩") = %e₩n", str, value); str = "3.1412764583d210"; value = atof(str); printf("Function: atof(₩"%s₩") = %e₩n", str, value); str = " -2309.12E-15"; value = atof(str); printf("Function: atof(₩"%s₩") = %e₩n", str, value);
실행결과	Function: atof(" 333640273517170716032O ") = 3.336403e+021 Function: atof("3.1412764583d210") = 3.141276e+210 Function: atof(" -2309.12E-15") = -2.309120e-012
함수설명	숫자로 된 문자열을 실수형 자료(double)로 변환

▌atol( ) 함수 : 문자열을 long형 정수로 변환

헤더파일	#include <stdlib.h>
함수원형	double atol(const char *str);
함수설명	숫자로 된 문자열을 long형 정수로 변환

▌ **atoi( ) 함수 :** 숫자로 된 문자열을 정수형 자료(int)로 변환

헤더파일	#include <stdlib.h>
함수원형	int atoi(const char *str);

예 제	str = " -2309 ";
	value = atoi(str);
	printf("Function: atoi(₩"%s₩") = %d₩n", str, value);
	str = "31412764";
	value = atoi(str);
	printf("Function: atoi(₩"%s₩") = %d₩n", str, value);
	str = "3336402735171707160320";
	value = atoi(str);
	printf("Function: atoi(₩"%s₩") = %d₩n", str, value);
	if (errno == ERANGE)
	{
	  printf("Overflow condition occurred.₩n");
	}

실행결과	Function: atoi(" -2309 ") = -2309 Function: atoi("31412764") = 31412764 Function: atoi("3336402735171707160320") = 2147483647 Overflow condition occurred.
함수설명	숫자로 된 문자열을 정수형 자료(int)로 변환하고, 변환된 자료의 값이 자료형의 범위를 벗어나면 errno 값이 ERANGE로 지정. 즉, errno 값이 ERANGE와 같으면 오버플로우(Overflow) 발생

▌ **itoa( ) 함수 :** 지정된 진법을 사용하여 문자열로 변환

헤더파일	#include <stdlib.h>
함수원형	char *itoa(int value, char *str, int radix);
함수설명	value를 radix로 지정된 진법을 사용하여 문자열로 변환

▌ ltoa( ) 함수 : long형 정수를 지정된 진법에 의해 문자열로 변환

헤더파일	#include <stdlib.h>
함수원형	char *ltoa(long value, char *str, int radix);
함수설명	long형 정수인 value를 radix로 지정된 진법을 사용하여 문자열로 변환

▌ tolower( ), toupper( ) : 알파벳 대소문자의 변환

헤더파일	#include <stdlib.h> #include <ctype.h>
함수원형	int tolower(int c); int toupper(int c);

예 제	#include <string.h>
	#include <ctype.h>
	char msg[] = "Some of THESE letters are Capitals.";
	char *p;
	int main(void)
	{
	printf("%s₩n", msg);
	/* Reverse case of message. */
	for(p = msg; p < msg + strlen(msg); p++)
	{
	if(islower(*p))
	putchar(toupper(*p));
	else if(isupper(*p))
	putchar(tolower(*p));
	else
	putchar(*p);
	}
	}

실행결과	Some of THESE letters are Capitals. sOME OF these LETTERS ARE cAPITALS.
함수설명	tolower 함수는 알파벳을 소문자로 변환하고, toupper 함수는 알파벳을 대문자로 변환(tolower, toupper 함수를 사용하기 위해서는 <ctype.h> 헤더 파일을 포함해야 함)

Reference 05. 메모리 할당 관련 함수

▌calloc( ) : 메모리 영역 확보와 주소 반환

헤더파일	#include <stdlib.h> #include <malloce.h>
함수원형	void *calloc(size_t n, size_t size);

예 제	// This program uses calloc to allocate space for
	// 40 long integers. It initializes each element to zero.
	#include <stdio.h>
	#include <malloc.h>
	int main(void)
	{
	long *buffer;
	buffer = (long *)calloc(40, sizeof(long));
	if(buffer != NULL)
	printf("Allocated 40 long integers₩n");
	else
	printf("Can't allocate memory₩n");
	free(buffer);
	}

실행결과	Allocated 40 long integers
함수설명	크기가 size 바이트인 메모리 영역 n개를 확보하여 해당 주소를 반환하고, 이때 확보된 메모리를 0으로 초기화 함.

▌malloc( ) : 메모리 확보와 포인터 반환

헤더파일	#include <stdlib.h> #include <malloce.h>
함수원형	void *malloc(size_t size);

예 제	char *string;
	string = malloc(100);
	if(string == NULL)

	printf("Insufficient memory available₩n");
	else
	{
	printf("Memory space allocated₩n");
	free(string);
	printf("Memory freed₩n");
	}

실행결과	Memory space allocated for path name Memory freed
함수설명	지정된 size(바이트 수) 만큼 메모리 블록을 할당하고, 포인터를 반환 (할당 가능한 메모리가 부족하면 NULL을 반환)

▌free( ) : 할당된 메모리 블록을 회수

헤더파일	#include <stdlib.h> #include <malloce.h>
함수원형	void free(void *memblock);

사용예제	string = malloc(100); // 메모리 블록 할당
	free(string); // 메모리 회수

함수설명	memblock이 가리키고 있는 메모리 블록을 해제

Reference 06. 버퍼 조작 함수

▌ memchr( ) : 버퍼에 있는 문자열 중에서 문자를 검색한 곳의 포인터를 반환

헤더파일	#include <string.h>
함수원형	void *memchr(void *str, int c, size_t n);
함수설명	버퍼 str에 있는 n개의 문자열을 대상으로 문자 c를 검색하여 처음 발견된 곳의 포인터를 반환

▌ memcmp( ) : 두 개의 버퍼에서 문자를 대상으로 비교

헤더파일	#include <string.h>
함수원형	int memcmp(void *str1, void str2, size_t n);
함수설명	두 개의 버퍼 str1과 str2에서 n개의 문자를 대상으로 비교

▌ memcpy( ) : 지정된 개수만큼의 문자를 복사

헤더파일	#include <string.h>
함수원형	void memcpy(void *str1, void *str2, size_t n);
함수설명	n개의 문자를 str2에서 str1로 복사

▌ memmov( ) : 문자열 중에서 지정된 개수만큼의 문자를 복사(겹침 허용)

헤더파일	#include <string.h>

함수원형	void memmov(void *dest, void *src, size_t n);
함수설명	버퍼 src에 있는 문자열 중에서 n개의 문자를 dest에 복사 (src와 dest가 겹쳐도 복사 허용)

▌memset( ) : 주어진 문자로 버퍼에 지정된 개수만큼 채움

헤더파일	#include <string.h>
함수원형	void memset(void *buf, int c, size_t n);
함수설명	주어진 문자 c로 버퍼 buf에 n개를 채움

Reference 07. 표준 입출력 함수

▌getch( ) : 키보드로 문자를 입력 받음 (화면에는 나타나지 않음)

헤더파일	#include <conio.h>
함수원형	int getch(void);
함수설명	키보드로부터 문자를 입력받지만, Echo 기능이 없어서 화면에 입력한 문자가 보이지는 않음

▌getchar( ) : 키보드로 문자를 입력받음

헤더파일	#include <stdio.h>
함수원형	int getchar(void);
함수설명	키보드로부터 문자를 입력받음

▌getche( ) : 키보드로 문자를 입력받음 (Echo 기능)

헤더파일	#include <conio.h>
함수원형	int getche(void);
함수설명	키보드로부터 문자를 입력받음 (입력받은 문자가 화면에 나타남)

▌gets_s( ) : 키보드로 문자열을 입력받음

헤더파일	#include <stdio.h>
함수원형	int *gets_s(char *buf);
함수설명	키보드로부터 문자를 입력받음

▌printf( ) : 서식화된 문자열을 화면에 출력

헤더파일	#include <stdio.h>
함수원형	int printf(format [argument, ...]);
함수설명	서식화된 문자열을 화면에 출력

▌scanf_s( ) : 키보드로 값을 입력

헤더파일	#include <stdio.h>
함수원형	int scanf_s(char *format_str, args);
함수설명	서식화된 값을 키보드로부터 입력 받음

▌putchar( ) : 문자를 화면에 출력

헤더파일	#include <stdio.h>
함수원형	int putchar(int c);
함수설명	문자 c를 화면에 출력

▌puts( ) : 문자열을 화면에 출력

헤더파일	#include <stdio.h>
함수원형	int puts(char *str);
함수설명	문자열 str을 화면에 출력

Reference 08. 파일 입출력 함수

▌clearerr( ) : 오류 지시자를 지움

헤더파일	#include <stdio.h>
함수원형	void clearerr(FILE *fpc);
함수설명	스트림 fpc의 오류 지시자를 지움

▌fclose( ) : 열린 파일 닫기

헤더파일	#include <stdio.h>
함수원형	int fclose(FILE *fpc);
함수설명	fpc가 가리키는 열린 파일을 닫음. (정상적인 수행은 0, 오류 발생은 EOF를 반환)

▌feof( ) : 파일의 끝을 검사

헤더파일	#include <stdio.h>
함수원형	int feof(FILE *fpc);
함수설명	fpc가 가리키는 파일의 끝을 검사 (지정된 파일이 EOF이면, 0 이외의 값을 반환)

▌ferror( ) : 오류 발생 검사

헤더파일	#include <stdio.h>
함수원형	int ferror(FILE *fpc);
함수설명	fpc가 가리키는 파일의 입출력 동안 오류 발생을 검사 (오류가 있으면 0 이외의 값을 반환)

▌fflush( ) : 버퍼를 비움

헤더파일	#include <stdio.h>
함수원형	int fflush(FILE *fpc);
함수설명	버퍼의 내용을 fpc 파일에 쓰는 버퍼를 비움 (정상 수행은 0, 그 이외에는 EOF를 반환)

▌fgetc( ) : 파일에서 한 문자를 읽어옴

헤더파일	#include <stdio.h>
함수원형	int fgetc(FILE *fpc);
함수설명	fpc 파일로부터 한 문자를 읽어옴 (오류이거나 파일의 끝이면 EOF를 반환)

▌fget( ) : 지정된 파일에서 문자열을 읽어 와서 저장

헤더파일	#include <stdio.h>
함수원형	int fgets(char *str, int n, FILE *fpc);
함수설명	fpc가 가리키는 파일에서 n개의 문자열을 읽어 와서 str에 저장 (읽어 들이는 단위는 EOF, 개행문자는 n-1의 문자 길이, 실패는 NULL 반환)

▌fopen( ) : 파일 열기

헤더파일	#include <stdio.h>
함수원형	FILE *fopen(char *filename, char *mode)
함수설명	파일 열기에서 지정된 모드(mode)로 파일을 열음 (성공할 경우는 FILE형 구조체 포인터, 오류가 발생할 경우는 NULL을 반환)

▌fprintf( ) : 지정된 서식으로 파일 출력

헤더파일	#include <stdio.h>
함수원형	int fprintf(FILE *fpc, char *format_string, args);
함수설명	지정된 서식으로 파일을 출력

▌fput( ) : 문자를 파일에 출력

헤더파일	#include <stdio.h>
함수원형	int fput(int c, FILE *fpc);
함수설명	문자 c를 파일 fpc에 출력 (정상출력은 출력한 문자의 수, 오류는 EOF를 반환)

▌fputs( ) : 문자열을 파일에 출력

헤더파일	#include <stdio.h>
함수원형	int fputs(char *str, FILE *fpc);
함수설명	문자열 str을 fpc 파일에 출력 (오류 발생은 EOF를 반환)

▌fread( ) : 파일에서 지정한 개수만큼 데이터를 읽어옴

헤더파일	#include <stdio.h>
함수원형	size_t fread(char *buf, size_t m, size_t n);
함수설명	파일로부터 m*n개의 데이터를 읽어옴

▌ **fseek( ) :** 파일의 포인터 위치 이동

헤더파일	#include <stdio.h>
함수원형	int fseek(FILE *fpc, long offset, int whence)
함수설명	현재 파일 fpc의 포인터 위치를 whence에서 offset로 이동

▌ **ftell( ) :** 파일 포인터의 현재 위치 반환

헤더파일	#include <stdio.h>
함수원형	long ftell(FILE *fpc);
함수설명	파일 포인터의 현재 위치를 반환 (오류 발생은 -1을 반환)

▌ **fwrite( ) :** 버퍼에 있는 데이터를 지정 개수만큼 파일에 출력

헤더파일	#include <stdio.h>
함수원형	size_t fwrite(char *buf, size_t m, size_t n, FILE *fpc);
함수설명	버퍼에 있는 m*n개의 데이터를 fpc 파일에 출력

▌ **getc( ) :** 파일에서 한 개의 문자를 읽어옴

헤더파일	#include <stdio.h>
함수원형	int getc(FILE *fpc);
함수설명	fpc 파일에서 한 개의 문자를 읽어옴

▌ putc( ) : 문자 한 개를 파일에 출력

헤더파일	#include <stdio.h>
함수원형	int putc(int c, FILE *fpc);
함수설명	하나의 문자 c를 fpc 파일에 출력

▌ rewind( ) : 파일 포인터의 첫 번째 위치로 이동

헤더파일	#include <stdio.h>
함수원형	void rewind(FILE *fpc)
함수설명	fpc 파일 포인터의 위치를 첫 번째 위치로 이동

▌ setbuf( ) : 파일에 새로운 버퍼 지정

헤더파일	#include <stdio.h>
함수원형	void setbuf(FILE *fpc, char *buf)
함수설명	fpc 파일에 새로운 buf 버퍼를 지정

Reference 09. 디렉터리 관련 함수

▌chdir( ) : 현재의 디렉터리 위치의 경로 변경

헤더파일	#include <dir.h>
함수원형	int chdir(char *path);
함수설명	현재의 디렉터리의 위치를 주어진 path 경로로 변경

▌getcwd( ) : 현재 작업 중인 디렉터리명 반환

헤더파일	#include <dir.h>
함수원형	char *getcwd(char *path, int numchars);
함수설명	현재 작업 중인 디렉터리의 이름을 반환

▌mkdir( ) : 디렉터리 생성

헤더파일	#include <dir.h>
함수원형	int mkdir(char *path);
함수설명	주어진 경로 path를 사용하여 새로운 디렉터리를 생성

▌rmdir( ) : 디렉터리 삭제

헤더파일	#include <dir.h>
함수원형	int rmdir(char *path);
함수설명	주어진 경로 path에 있는 디렉터리를 삭제

Reference 10. 시간 관련 함수

▌ tm 구조체의 프레임

```
struct tmset
{
  int tmset_sec;
  int tmset_min;
  int tmset_hour;
  int tmset_mday;
  int tmset_mon;
  int tmset_year;
  int tmset_wday;
  int tmset_wday;
  int tmset_yday;
  int tmset_isdst;
};
```

▌ asctime( ) : 시간을 문자열로 반환

헤더파일	#include <time.h>
함수원형	char *asctime(struct tmset *time)
함수설명	구조체 tmset 형식의 시간을 문자열로 변환

▌ gtime( ) : SMT를 구조체 포인터로 변환

헤더파일	#include <time.h>
함수원형	struct tmset *gtime(time_history *timeptr)
함수설명	GMT(Greenwich Meam Time)을 tmset 구조체 포인터로 반환

▌ localtime( ) : local time을 구조체 포인터로 반환

헤더파일	#include <time.h>
함수원형	struct tmset *localtime(time_history *time)
함수설명	local time을 tmset 구조체 포인터로 반환

▌ ftime( ) : 1970년 이후 경과된 시간 반환

헤더파일	#include <time.h>
함수원형	time_history time(time_history *timeptr)
함수설명	1970. 1. 1. 00:00:00초부터 현재까지 경과된 시간을 초로 환산하여 반환

▌ clock( ) : 경과된 시간 반환

헤더파일	#include <time.h>
함수원형	clock_history time(void)
함수설명	지금부터 경과된 시간을 반환

▌ ctime( ) : 날짜와 시간을 반환

헤더파일	#include <time.h>
함수원형	char *ctime(any_time *timestmp)
함수설명	날짜와 시간을 문자열로 반환

▌ **difftime( ) :** 시간의 차이를 초로 환산

헤더파일	#include <time.h>
함수원형	double difftime(time_history time1, time_history time2)
함수설명	두 시간 time1과 time2 사이의 시간 차이를 초 단위로 환산하여 반환