학번 : _____ 성명 : _____.

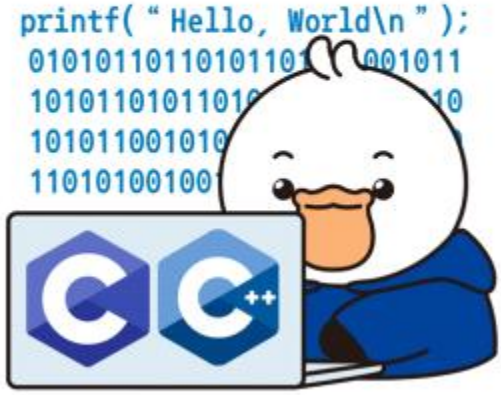
- 본 강의자료는 과학기술정보통신부 및 정보통신기획평가원에서 지원하는『소프트웨어중심대학』사업의 결과물입니다.
- 본 강의자료는 내용은 전재할 수 없으며, 인용할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원의 '소프트웨어중심대학'의 결과물이라는 출처를 밝혀야 합니다.

SW 대구가톨릭대학교 소프트웨어중심대학사업단

대구가톨릭대학교 컴퓨터소프트웨어학부 School of Computer Software, Daegu Catholic University

Part 12. 함수의 활용

목차



12.1 stdio.h

- C 표준 라이브러리
  - ✓ stdio.h, stdlib.h, conio.h, ctype.h, math.h 등
  - ✓ 각 헤더파일에는 수많은 라이브러리 함수 존재

12.1 stdio.h

이름	표준	상세
<assert.h>		assert 매크로를 포함하며, 프로그램의 디버깅 버전에서 논리 오류와 버그의 다른 타입을 탐지하는 것을 지원
<complex.h>	C99	복소수를 조작하는데 사용되는 함수들의 집합.
<ctype.h>		그들의 타입에 따라 문자들을 분류하거나 대소문자를 전환하는데 사용되는 함수들의 집합을 정의한다.
<errno.h>		라이브러리 함수들에 의해 리포트되는 오류 코드들을 테스팅할 때 사용된다.
<fenv.h>	C99	부동소수점 환경을 제어하는데 사용되는 함수들의 집합을 정의한다.
<float.h>		부동소수점 라이브러리의 구현된 속성을 명시하는 매크로 상수를 정의한다.
<inttypes.h>	C99	정확한 정수형을 정의한다.
<iso646.h>	NA1	여러 표준 토큰들을 표현하기 위한 대체 방식들을 구현하는 여러 매크로들을 정의한다.
<limits.h>		정수형 타입의 구현된 속성을 명시하는 매크로 상수를 정의한다.
<locale.h>		지역화 함수 정의
<math.h>		일반적인 수학 함수 정의
<setjmp.h>		setjmp 와 longjmp 매크로를 선언한다.
<signal.h>		시그널 핸들링 함수를 정의한다.
<stdalign.h>	C11	객체들의 얼라인먼트를 질의하고 명시하기 위한.

<c언어 표준 라이브러리 헤더 파일>

12.1 stdio.h

이름	표준	상세
<stdarg.h>		함수에 전달되는 인자들에 접근하기 위한.
<stdatomic.h>	C11	스레드 사이에서 공유되는 데이터의 원자적 동작을 위한.
<stdbool.h>	C99	불린 데이터 형 정의
<stddef.h>		여러 유용한 타입과 매크로 정의.
<stdint.h>	C99	정확한 정수형을 정의.
<stdio.h>		핵심 입력과 출력 함수들을 정의한다.
<stdlib.h>		숫자 변환 함수들, 슈도 랜덤 숫자 생성 함수들, 메모리 할당, 프로세스 제어 함수들을 정의한다.
<stdnoreturn.h>	C11	반환하지 않는 함수들을 명시하기 위한
<string.h>		문자열 처리 함수들을 정의한다.
<tgmath.h>	C99	포괄형 수학 함수들을 정의한다.
<threads.h>	C11	다중 스레드들과 뮤텍스 그리고 제어 변수들을 관리하는 함수들을 정의한다.
<time.h>		데이터와 시간 처리 함수들을 정의한다.
<uchar.h>	C11	유니코드 문자들과 이것들을 조작하는 함수들
<wchar.h>	NA1	wide 문자열 처리 함수들을 정의한다.
<wctype.h>	NA1	그들의 형에 따라 wide 문자들을 분류하거나 대소문자를 전환하는데 사용되는 함수들의 집합.

<c언어 표준 라이브러리 헤더 파일>

12.1 stdio.h

- stdio.h
  - ✓ [ ](표준입출력 라이브러리)의 약자
  - ✓ C언어 표준 입출력을 위한 매크로와 상수를 정의하고 다양한 입출력 함수를 포함된 헤더 파일
  - ✓ 제공되는 함수들은[ ]이용
    - ➢ 시스템의 특정한 형태 파일이나 키보드, 프린터, 터미널 등과 같은 입출력 처리와 관련

12.1 stdio.h

- stdio.h
  - ✓ stdio.h 헤더 파일을 포함하는 C 프로그램
    - ➢ 표준 입력 스트림(stdin), 표준 출력 스트림(stdout), 표준 에러 스트림(stderr) 사용 가능
  - ✓ 대표적인 함수 : printf(), scanf()
  - ✓ [ ] 함수
    - ➢ stdio.h 헤더파일에서 제공하는 함수 중 대표적인 콘솔 입출력 함수
    - ➢ 표준 출력 스트림을 이용하여 출력하는 기능 수행

12.1 stdio.h

	바이트 문자	확장 문자	설명
파일 접근	fopen()		파일 열기(윈도우에선 비-유니코드 파일 이름, 유닉스에서는 UTF-8 파일 이름)
	freopen()		존재하는 스트림으로 다른 파일 열기
	fflush()		대응되는 파일로 출력 스트림 동기화하기
	fclose()		파일 닫기
	setbuf()		파일 스트림에 버퍼 장착시키기
	setvbuf()		파일 스트림 크기에 맞게 버퍼 장착
	fwide()		전각 문자와 반각 문자간 파일 스트림 교환
직접 입출력	fread()		파일 읽기
	fwrite()		파일 쓰기
언포맷 입출력	fgetc() getc()	fgetwc() getwc()	파일 스트림으로 부터 바이트/wchar_t 읽기
	fgets()	fgetws()	파일 스트림으로부터 바이트/wchar_t 라인 읽기
	fputc() putc()	fputwc() putwc()	파일 스트림에 바이트/wchar_t 쓰기
	fputs()	fputws()	파일 스트림에 바이트/wchar_t 문자열 입력
	getchar()	getwchar()	표준 입력으로부터 바이트/wchar_t 입력
	putchar()	putwchar()	
	puts()		표준 출력으로 바이트 문자열 입력
	ungetc()	ungetwc()	파일 스트림에 바이트/wchar_t 제자리에 돌려놓기

<stdio.h 의 멤버 함수>

12.1 stdio.h

	바이트 문자	확장 문자	설명
포맷 입출력	scanf() fscanf() sscanf()	wscanf() fwscanf() swscanf()	파일 스트림이나 버퍼의 표준 입력으로부터 형식화된 바이트/wchar_t 입력 읽기
	vscanf() vfscanf() vsscanf()	vwscanf() vfwscanf() vswscanf()	가변 인자 목록을 쓰는 파일 스트림이나 버퍼의 표준 입력으로부터 형식화된 바이트/wchar_t 읽기
	printf() fprintf() sprintf() snprintf()	wprintf() fwprintf() swprintf()	파일 스트림이나 버퍼의 표준 출력으로 형식화된 바이트/wchar_t 출력을 출력하기
	vprintf() vfprintf() vsprintf() vsnprintf()	vwprintf() vfwprintf() vswprintf()	가변 인자 목록을 쓰는 파일 스트림이나 버퍼의 표준 출력으로 형식화된 바이트/wchar_t 출력을 출력
	perror()		표준 에러의 현재 에러 설명 쓰기

<stdio.h 의 멤버 함수>

12.1 stdio.h

	바이트 문자	확장 문자	설명
파일 위치 조정	ftell() ftello()		현재 파일 포인터 되돌려주기
	fseek() fseeko()		파일의 특정 위치로 파일 포인터 이동
	fgetpos()		파일 포인터 얻기
	fsetpos()		파일의 특정 위치로 파일 포인터 이동
	rewind()		파일 포인터를 파일의 첫 시작 부분으로 이동
에러 조작	clearerr()		에러 삭제
	feof()		파일의 끝 체크
	ferror()		파일 에러 체크
파일 조작 명령	remove()		파일 삭제
	rename()		파일 이름 수정
	tmpfile()		임시 파일로 포인터 되돌리기
	tmpnam()		특수 파일 이름 되돌려주기

<stdio.h 의 멤버 함수>

12.1 stdio.h

- stdio.h의 멤버 함수
  - ✓ s로 시작하는 함수
    - ➢ [ ]과 관련된 함수
  - ✓ f로 시작하는 함수
    - ➢ 대부분[ ]과 관련된 함수

12.1 stdio.h

- stdio.h의 멤버 함수
  - ✓ [ ] 함수
    - ➢ 함수의 끝에 f글자 있는 함수
    - ➢ scanf(), printf() 등
  - ✓ [ ] 함수
    - ➢ fgetc(), getc(), fputc(), putc(), getchar(), putchar() 등

12.1 stdio.h

- 언포맷 입출력 함수
  - ✓ [ ]
    - ➢ 표준 입력(stdin)에서 문자 하나를 읽어오는 함수
  - ✓ [ ]
    - ➢ 표준 출력(stdout)에 문자를 출력하는 함수
    - ➢ 표준 출력에서 현재 위치 표시자 (커서: cursor)가 가리키는 곳에 문자를 작성한 후, 위치 표시자를 다음 위치로 이동

12.1 stdio.h

- 언포맷 입출력 함수
  - ✓ [ ]
    - ➢ gets() 함수에 대응하는 함수로, 표준 출력에 문자열 출력
    - ➢ 문자열 포인터가 가리키는 주소부터 널 문자에 도착할 때까지 표준 출력에 문자를 복사하여 출력하고, 마지막 널 문자를 대신해 '\n'인 줄바꿈 문자 출력

```
#include <stdio.h>
int getchar(void);
int putchar(int character);
int puts(const char * str);
```

<getchar(), putchar(), puts() 함수의 원형>

12.1 stdio.h

소스 12-1

```
1  #include <stdio.h>
2  int main(void) {
3        char ch;
4        printf("Enter text [(.)exit] : ");
5        do {
6              ch = getchar();
7              putchar(ch);
8        } while (ch != '.');
9        return 0;
10 }
```

입력 예	abcdef .
출력 예	Enter text [(.)exit] : abcdef abcdef . .

12.1 stdio.h

소스 12-2

```
1  #include <stdio.h>
2  int main(void) {
3        int i;
4
5        char str[] ="Hello, World!";
6
7        for (i =0; i <5; i ++) {
8              puts(str);
9        }
10       return 0;
11 }
```

출력 예

```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```


12.1 stdio.h

- stdio.h의 멤버 상수
  - ✓ 다양한 상수들이 정의되어 제공
    - ➢ [ ]을 나타내는 EOF
    - ➢ 파일이름의 최대 길이를 나타내는 FILENAME_MAX
    - ➢ 임시 파일의 (최대) 개수를 나타내는 TMP_MAX

12.1 stdio.h

이름	해설
EOF	end-of-file 가리키는 용도로 사용되는 int 형의 음의 정수
BUFSIZ	setbuf() 함수에 의해 버퍼 크기를 나타내는 정수
FILENAME_MAX	충분히 열 수 있는 저장 가능한 파일 이름의 char 형의 배열 크기
FOPEN_MAX	동시에 열 수 있는 파일의 개수; 최소 8
_IOFBF	"input/output fully buffered"의 약어. 이 정수값은 setvbuf() 함수에 넘겨서 버퍼화된 블록의 스트림을 열기 위해 입출력에 요구한다.
_IOLBF	"input/output fully buffered"의 약어. 이 정수값은 setvbuf() 함수에 넘겨서 버퍼화된 블록의 스트림을 열기 위해 입출력에 요구한다.
_IONBF	"input/output fully buffered"의 약어. 이 정수값은 setvbuf() 함수에 넘겨서 버퍼화된 블록의 스트림을 열기 위해 입출력에 요구한다.
L_tmpnam	char 배열이 tmpnam() 함수를 발생시킬 정도의 충분한 크기
NULL	널 포인터의 약어인 매크로 상수. 이 상수는 메모리의 어떤 유효한 위치의 개체도 가리키지 않는 포인터 값이다.
SEEK_CUR	현재 파일 위치에 대해 위치 변경을 요청하는 fseek()에 전달되는 정수
SEEK_END	파일의 끝에 대해 위치 조정을 요청하기 위한 fseek() 함수에 전달되는 정수.
SEEK_SET	파일의 시작 위치를 기준으로 한 위치 지정을 요청하기 위한 fseek() 함수에 전달되는 정수.
TMP_MAX	tmpnam() 기능에 의해 만들어지는 특수 파일이름의 최대길이. (최소 25자)

< stdio.h의 멤버 상수 >

12.1 stdio.h

- stdio.h의 멤버 형식
  - ✓ FILE, fpos_t, size_t의 데이터 타입 제공

이름	해설
FILE	입출력 작동에 필요한 파일/텍스트 스트림에 대한 정보를 포함하는 구조이다.
fpos_t	유일하게 파일의 바이트 위치를 식별할 수 있는 비 배열 형식
size_t	sizeof 연산자의 결과값을 나타내는 형식의 양의 정수형

< stdio.h의 멤버 형식 >

12.2 stdlib.h

- [ ]
  - ✓ 문자열 변환과 의사 난수 생성, 동적 메모리 관리 등의 함수들을 포함
- stdlib.h의 멤버 함수
  - ✓ 프로그램을 작성할 때 다양한 용도로 널리 사용됨
  - ✓ 다양한 기능의 함수 제공
    - ➢ ex) 문자열 변환, 의사 난수 생성, 동적 메모리 관리, 프로세스 제어, 검색과 정렬, 정수 산술 등

12.2 stdlib.h

함수	설명
문자열 변환	
int atoi (const char * str);	str을 int로 변환한다.
long int atol (const char * str);	str을 long으로 변환한다.
double atof (const char * str);	str을 double으로 변환한다.
long int strtol (const char * str, char ** endptr, int base);	str을 base진법으로 long으로 변환한 뒤 endptr!=NULL이면 숫자가 끝난 후 첫 문자의 위치를 endptr에 반환한다.
unsigned long int strtoul (const char * str, char ** endptr, int base);	str을 base진법으로 unsigned long으로 변환한 뒤 endptr!=NULL이면 숫자가 끝난 후 첫 문자의 위치를 endptr에 반환한다.
double strtod (const char * str, char ** endptr);	str을 base진법으로 double으로 변환한 뒤 endptr!=NULL이면 숫자가 끝난 후 첫 문자의 위치를 endptr에 반환한다.
의사 난수 생성	
int rand (void);	0부터 RAND_MAX 사이의 의사 난수를 반환한다.
void srand (unsigned int seed);	의사 난수 발생기를 seed로 초기화한다. 보통 seed는 time(NULL)로 설정된다.

< stdlib.h의 멤버 함수 >

12.2 stdlib.h

함수	설명
동적 메모리 관리	
`void * malloc ( size_t size );`	size 바이트의 메모리를 힙에서 할당하여 반환한다.
`void * calloc ( size_t num, size_t size );`	(num * size) 바이트의 메모리를 힙에서 할당하여 반환한다.
`void * realloc ( void * ptr, size_t size );`	ptr이 가리키는 메모리를 size 바이트만큼 힙에서 재할당하여 반환한다.
`void free ( void * ptr );`	ptr이 가리키는 메모리를 해제한다.(할당했으면 반드시 해제해야 한다.)
프로세스 제어	
`void abort ( void );`	현재 프로세스를 비정상적으로 종료한다.
`int atexit ( void ( * function ) (void) );`	프로세스가 정상적으로 종료되었을 때 실행할 함수를 지정한다.
`void exit ( int status );`	현재 프로세스를 정상적으로 종료한다. status는 부모 프로세스에게 전달된다.
`char * getenv ( const char * name );`	name에 해당하는 환경 변수를 반환한다.
`int system ( const char * command );`	command의 시스템 명령을 실행한다.

< stdlib.h의 멤버 함수 >

12.2 stdlib.h

함수	설명
검색/정렬	
`void * bsearch ( const void * key, const void * base, size_t num, size_t size, int ( * comparator ) ( const void *, const void * ) );`	이진 검색을 한다.
`void qsort ( void * base, size_t num, size_t size, int ( * comparator ) ( const void *, const void * ) );`	퀵 정렬 알고리즘을 사용하여 정렬한다.
정수 산술	
`int abs ( int n );` `long int labs ( long int n );`	n의 절댓값을 반환한다.
`div_t div ( int numerator, int denominator );` `ldiv_t ldiv ( long numerator, long denominator );`	numerator를 denominator로 나누어 div_t 또는 ldiv_t 구조체에 quot 멤버에는 몫을, rem 멤버에는 나머지를 채운 뒤 반환한다.

< stdlib.h의 멤버 함수 >

12.2 stdlib.h

- stdlib.h의 멤버 함수
  - ✓ 문자열 변환 함수
    - ➢ 문자열로 입력받은 데이터를 [ ] 하는 기능
    - ➢ 문자열을 int형이나 double형 등으로 변환 가능
    - ➢ ex) [ ]
      - ascii to int 의 약자
      - 문자열을 int형으로 변환하는 함수

12.2 stdlib.h

소스 12-3

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void) {
4       printf("int형 수: %d\n", atoi("12345"));
5       printf("long형 수: %ld\n", atol("123456789"));
6       printf("double형 수: %f\n", atof("12345.6789"));
7       return 0;
8 }
```

atoi() : [ ]형 변환
atol() : [ ]형 변환
atof() : [ ]형 변환

출력 예

```
int형 수: 12345
long형 수: 123456789
double형 수: 12345.678900
```

12.2 stdlib.h

- stdio.h의 멤버 함수
  - ✓ 의사 난수 생성함수 : rand() 함수
    - ➢ 0에서 RAND_MAX 상수 사이의 랜덤한 난수 반환
    - ➢ RAND_MAX : 32767로 정의된 함수
      - 컴파일 환경에 따라 다른 범위 가질 수 있음
    - ➢ 프로그램 실행 시 매번 동일한 난수 값 나옴
      - 프로그램 생성 시 값이 정해지기 때문

12.2 stdlib.h

소스 12-4

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void) {
4     int i;
5     for (i=0; i<10; i++) {
6         printf("Random value : %d\n", rand());
7     }
8     return 0;
9 }
```

출력 예

```
Random value : 41
Random value : 18467
Random value : 6334
Random value : 26500
Random value : 19169
Random value : 15724
Random value : 11478
Random value : 29358
```

12.2 stdlib.h

- stdio.h의 멤버 함수
  - ✓ 의사 난수 생성함수 : [ ] 함수
    - ➢ rand() 함수로 난수를 발생할 때 사용하는 씨앗(씨드 : seed) 값을 초기화하여 매 프로그램 실행마다 난수 발생을 다르게 할 수 있도록 하는 역할
    - ➢ srand() 함수가 rand() 함수의 난수 발생 패턴을 초기화할 수 있는 이유?
      - rand() 함수가 난수를 발생할 때 사용하는 [ ]을 srand() 함수가 변경해주기 때문

12.2 stdlib.h

소스 12-5

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(void) {
4        printf("rand() with default\n");
5        printf("rand : %d\n", rand());
6        printf("rand : %d\n", rand());
7        printf("rand() with same seed\n");
8        srand(32323);
9        printf("rand : %d\n", rand());
10       srand(32323);
11       printf("rand : %d\n", rand());
12       srand(32323);
13       printf("rand : %d\n", rand());
14       printf("rand() with 4 different seed\n");
15       for (int i =0; i < 4; ++i) {
16             srand(i);
17             printf("rand : %d\n", rand());
18       }
19       return 0;
20 }
```

출력 예

```
rand() with default
rand : 1804289383
rand : 846930886
rand() with same seed
rand : 829295540
rand : 829295540
rand : 829295540
rand() with 4 different seed
rand : 1804289383
rand : 1804289383
rand : 1505335290
rand : 1205554746
```

12.2 stdlib.h

소스 12-6

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  int main(void) {
5    int i, num;
6    srand(time(0));
7    for (i=0; i<10;++i) {
8      num = rand();
9      printf("Random value : %d\t1~10 : %d\n", num, num%10+1);
10   }
11   return 0;
12 }
```

✓ time()함수 : 1970년 1월 1일을 시작으로 하는 초 단위의 현재 시간 반환

출력 예

```
Random value : 415507132        1~10 : 3
Random value : 1373473373       1~10 : 4
Random value : 2018578750       1~10 : 1
Random value : 390311475        1~10 : 6
Random value : 501434264        1~10 : 5
Random value : 1260846406       1~10 : 7
Random value : 1907674561       1~10 : 2
Random value : 1294776604       1~10 : 5
Random value : 606008473        1~10 : 4
Random value : 1630367994       1~10 : 5
```


12.2 stdlib.h

- stdio.h의 멤버 함수
  - ✓ 정수 산술 함수
    - ➢ [ ] 함수
      - int형 절댓값을 계산하는 함수
    - ➢ [ ] 함수
      - long형 절댓값을 계산하는 함수

12.2 stdlib.h

소스 12-7

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(void) {
4        int n1 = -12;
5        int n2 = 15;
6        printf("abs(%d) : %d\n", n1, abs(n1));
7        printf("abs(%d) : %d\n", n2, abs(n2));
8        printf("\n");
9        return 0;
10 }
```

출력 예

```
abs(-12) : 12
abs(15) : 15
```


12.3 ctype.h

- [ ]

✓ C 언어의 문자들을 처리하는 다양한 함수들 제공

- 문자가 주어진 조건에 맞는지 검사하거나 문자를 변환
- 아스키 값을 판별하여 해당 문자가 숫자를 표현하는 문자인지 알파벳을 표현하는 문자인지, 특수 문자인지 등을 판별
- 알파벳 대문자를 소문자로 변환하거나 소문자를 대문자로 변환

12.3 ctype.h

- ctype.h의 멤버 함수
  - ✓ is로 시작하는 함수
    - ➢ 주로 뒤에 나오는 단어의 [ ]인지를 검사하는 함수
      - ex) isalpha()
  - ✓ to로 시작하는 함수
    - ➢ 뒤에 나오는 형태로 [ ]하는 함수
      - ex) tolower(), toupper()

12.3 ctype.h

함수	설명
문자검사	
int isalnum (int c);	c가 알파벳 또는 숫자이면 0이 아닌 값을 반환한다.
int isalpha (int c);	c가 알파벳이면 0이 아닌 값을 반환한다.
int iscntrl (int c);	c가 제어 문자이면 0이 아닌 값을 반환한다.
int isdigit (int c);	c가 숫자이면 0이 아닌 값을 반환한다.
int isgraph (int c);	c가 그래픽 문자이면 0이 아닌 값을 반환한다.
int islower (int c);	c가 소문자이면 0이 아닌 값을 반환한다.
int isprint (int c);	c가 출력할 수 있는 문자이면 0이 아닌 값을 반환한다.
int ispunct (int c);	c가 구두점 문자이면 0이 아닌 값을 반환한다.
int isspace (int c);	c가 공백 문자이면 0이 아닌 값을 반환한다.
int isupper (int c);	c가 대문자이면 0이 아닌 값을 반환한다.
int isxdigit (int c);	c가 16진 숫자이면 0이 아닌 값을 반환한다.
문자변환	
int tolower (int c);	c를 소문자로 변환한다.
int toupper (int c);	c를 대문자로 변환한다.
int __toascii (int c);	c를 아스키 코드로 변환한다.

<ctype.h 의 멤버 함수>

12.3 ctype.h

소스 12-8

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 int main(void) {
5     printf("%d\n", isalnum('!'));
6     printf("%d\n", isalnum('2'));
7     printf("%d\n", isalnum('a'));
8     return 0;
9 }
```

출력 예	0
	4
	2

12.3 ctype.h

소스 12-9

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 int main(void) {
5       char ch;
6       scanf("%c", &ch);
7       if (isupper(ch)) printf("%c", tolower(ch));
8       else printf("%c", toupper(ch));
9 }
```

입력 예 1	A
출력 예 1	a
입력 예 2	D
출력 예 2	d

12.3 ctype.h

소스 12-10

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  int main(void) {
5          char str[50];
6          int trans_ch;
7          int count =0;
8          scanf("%s", str);
9          while (str[count] != '\0') {
10                 trans_ch = __toascii(str[count]);
11                 printf("%d", trans_ch);
12                 count++;
13         }
14 }
```

입력 예	abcdef
출력 예	979899100101102

12.4 math.h

- [ ]
  - ✓ 여러 가지 수학관련 함수들 제공
  - ✓ 부동소수점을 다루는 함수와 삼각 함수 등을 제공
- math.h의 멤버 함수
  - ✓ 삼각함수를 비롯하여 역 삼각 함수, 쌍곡선 함수, 지수 함수, 대수 함수, 거듭제곱 함수, 제곱근 함수, 천정 함수, 바닥 함수, 절댓값 함수, 나머지 함수 등

12.4 math.h

함수	설명
삼각 함수	
`double sin ( double x );`	사인 x를 구한다.
`double cos ( double x );`	코사인 x를 구한다.
`double tan ( double x );`	탄젠트 x를 구한다.
역 삼각 함수	
`double asin ( double x );`	아크 사인 x를 구한다.
`double acos ( double x );`	아크 코사인 x를 구한다.
`double atan ( double x );`	아크 탄젠트 x를 구한다.
`double atan2 ( double y, double x );`	아크 탄젠트 y/x를 구한다.
쌍곡선 함수	
`double sinh ( double x );`	하이퍼볼릭 사인 x를 구한다.
`double cosh ( double x );`	하이퍼볼릭 코사인 x를 구한다.
`double tanh ( double x );`	하이퍼볼릭 탄젠트 x를 구한다.

<math.h의 멤버 함수>

12.4 math.h

함수	설명
지수, 대수 함수	
`double exp ( double x );`	e^x를 구한다.
`double frexp ( double x, int * exp );`	지수를 exp가 가리키는 변수에 저장하고 가수를 반환한다.
`double ldexp ( double x, int exp );`	$x * 2^{exp}$를 반환한다.
`double log ( double x );`	loge x를 구한다.
`double log10 ( double x );`	log10 x를 구한다.
`double modf ( double x, double * intpart );`	정수부를 intpart가 가리키는 변수에 저장하고 소수부를 반환한다.
거듭제곱, 거듭제곱근, 올림, 내림, 절댓값, 나머지 함수	
`double pow ( double x, double y );`	x^y를 구한다.
`double sqrt ( double x );`	$\sqrt{x}$를 구한다.
`double ceil ( double x );`	x보다 작지 않은 가장 작은 정수를 구한다.
`double floor ( double x );`	x보다 크지 않은 가장 큰 정수를 구한다.
`double fabs (double x);`	x의 절댓값을 구한다.
`double fmod ( double x, double y );`	x를 y로 나눈 나머지를 구한다.

<math.h의 멤버 함수>

12.4 math.h

- math.h의 멤버 함수
  - ✓ 정수형 변환 함수
    - ➢ [ ] 함수
      - 실수형 데이터를 정수형으로 변환하며 인자값 x를 초과하는 정수 중에서 가장 작은 정수를 출력하는 천정 함수
    - ➢ [ ] 함수
      - x보다 작은 정수 중에서 가장 큰 정수를 출력하는 바닥 함수

함수	f = 1234.567	f = -1234.567
ceil(f)	1235	-1234
floor(f)	1234	-1235

<floor()함수와 ceil()함수>

12.4 math.h

소스 12-11

```
1  #include <stdio.h>
2  #include <math.h>
3  int main(void) {
4    double du1 =1234.567, du2 =-1234.567;
5    printf("ceil(%.3f) = %.3f\n", du1, ceil(du1));
6    printf("floor(%.3f) = %.3f\n", du1, floor(du1));
7    printf("ceil(%.3f) = %.3f\n", du2, ceil(du2));
8    printf("floor(%.3f) = %.3f\n", du2, floor(du2));
9    return 0;
10 }
```

출력 예

```
ceil(1234.567) = 1235.000
floor(1234.567) = 1234.000
ceil(-1234.567) = -1234.000
floor(-1234.567) = -1235.000
```

12.4 math.h

소스 12-12

```
1  #include <stdio.h>
2  #include <math.h>
3  int main(void) {
4      double du1 = 4.2;
5      double du2 = 4.7;
6      printf("%0.1lf\n", ceil(du1));
7      printf("%0.1lf\n", floor(du1));
8      printf("%0.1lf\n", floor(du1 +0.5));
9      printf("%0.1lf\n", floor(du2 +0.5));
10     return 0;
11 }
```

출력 예

```
5.0
4.0
4.0
5.0
```


12.4 math.h

소스 12-13

```
1  #include <stdio.h>
2  #include <math.h>
3  int main(void) {
4        double du =12.3;
5        int i;
6        int i1 =4;
7        int *pt =&i;
8        printf("%f\n", exp(du));
9        printf("%f\n", frexp(du,pt));
10       printf("%f\n", ldexp(du,i1));
11       printf("%f\n", log(du));
12       printf("%f\n", log10(du));
13       return 0;
14 }
```

✓ 지수, 로그 함수

출력 예

```
exp(1.2) : 219695.988672
frexp(1.2, pt) : 0.768750
ldexp(1.2, 4) : 196.800000
log(1.2) : 2.509599
log10(1.2) : 1.089905
```

12.4 math.h

- math.h의 멤버 함수

각도를 라디안으로 변환	라디안을 각도로 변환
radian = defree * PI/180;	degree = radian*180/PI;

<각도와 라디안의 변환 공식>

  - ✓ 삼각함수
    - ➢ math.h 라이브러리의 삼각함수의 데이터형 : [ ]
    - ➢ x와 y 모두 [ ] 형으로 라디안 값 사용
    - ➢ 삼각함수의 계산 결과는 [ ] 형으로 반환
    - ➢ 각도(degree)와 라디안(호도, radian) 구분해서 사용할 것
      - 각도 : 원 한바퀴의 각을 360으로 나눈 값을 기본 단위로 사용
      - 라디안 : 반지름이 r이고 호의 길이도 r인 반지름과 같은 길이의 호가 이루는 각을 기본 단위로 사용

12.4 math.h

소스 12-14

```
1  #include <stdio.h>
2  #include <math.h>
3  #define PI 3.141592
4  int main(void) {
5     int i;
6     double radian, s, c, t;
7     printf("input degree : ");
8     scanf("%d", &i);
9     radian = (double)i * PI/180;
10    s = sin(radian);
11    c = cos(radian);
12    printf("sin(%2d) =%9.6f, cos(%2d) =%9.6f\n", i, s, i, c);
13    return 0;
14 }
```

입력 예	12
출력 예	input degree : 12 sin(12) = 0.207912, cos(12) = 0.978148

12.4 math.h

소스 12-15

```c
#include <stdio.h>
#include <math.h>
#define PI 3.141592
int main(void) {
    int i =0;
    int graph =0;
    double s =0;
    for (i =0; i <=360; i +=20) {
        s = sin((PI * i) /180.0);
        printf("sin(%3d) %+.4f", i, s);
        for (graph =-15; graph < s *10; graph++) {
            printf(" ");
        }
        printf("*\n");
    }
}
```

출력 예

```
sin(  0) +0.0000               *
sin( 20) +0.3420                   *
sin( 40) +0.6428                      *
sin( 60) +0.8660                         *
sin( 80) +0.9848                          *
sin(100) +0.9848                          *
sin(120) +0.8660                         *
sin(140) +0.6428                      *
sin(160) +0.3420                   *
sin(180) +0.0000               *
sin(200) -0.3420           *
sin(220) -0.6428       *
sin(240) -0.8660     *
sin(260) -0.9848   *
sin(280) -0.9848   *
sin(300) -0.8660     *
sin(320) -0.6428       *
sin(340) -0.3420           *
sin(360) -0.0000               *
```


12.5 time.h

- time.h
  - ✓ [ ] 처리하기 위한 함수들을 제공하는 헤더 파일
- time.h의 멤버 함수
  - ✓시간 조작 함수와 변환 함수로 분류

12.5 time.h

함수	설명
시간 조작	
clock_t clock (void);	프로그램이 시작될 때부터 지난 시간(단위ms)을 반환한다.
double difftime (time_t time2, time_t time1);	time2와 time1의 차이를 반환한다.
time_t time (time_t * timer);	timer가 NULL이 아니면 timer가 가리키는 변수에 현재 시간을 채운다.
변환	
char * asctime (const struct tm * timeptr);	timeptr이 가리키는 구조체를 문자열로 변환한다.
char * ctime (const time_t * timer);	timer가 가리키는 변수를 문자열로 변환한다.
time_t mktime (struct tm * timeptr);	timeptr이 가리키는 구조체를 time_t 형식으로 변환한다.
struct tm * gmtime (const time_t * timer);	timer가 가리키는 변수를 UTC 시간 기준으로 구조체로 변환해 그 주소를 반환한다.
struct tm * localtime (const time_t * timer);	timer가 가리키는 변수를 지역 시간 기준으로 구조체로 변환해 그 주소를 반환한다.
size_t strftime (char * ptr, size_t maxsize, const char * format, const struct tm * timeptr);	시간을 문자열로 서식화한다.

<time.h의 멤버 함수>

12.5 time.h

- time.h의 멤버 함수
  - ✓ 시간 조작 함수
    - ➢ [ ] 함수
      - 프로그램일 시작될 때부터 지난 시간을 반환
    - ➢ [ ] 함수
      - 특정 두 시각의 차이를 계산
    - ➢ [ ] 함수
      - 현재 시간을 반환

12.5 time.h

소스 12-16

```c
#include <stdio.h>
#include <time.h>
int main(void) {
    int i, j;
    int sum = 0;
    clock_t start, end;
    double result_time;
    start = clock(); //시간 측정 시작
    for (i = 0; i < 3000; i++) {
        for (j = 0; j < 40000; j++) {
            sum += i + j;
        }
    }
    end = clock(); //시간 측정 끝
    result_time = (double)(end - start);
    printf("%.2fms", result_time);
    return 0;
}
```

출력 예	447190.00ms

12.5 time.h

소스 12-17

```c
#include <stdio.h>
#include <time.h>
int main(void) {
	int i, j;
	int sum = 0;
	time_t start, end;
	double result_time, dif;
	time(&start); //시간 측정 시작
	for (i = 0; i < 3000; i++) {
		for (j = 0; j < 40000; j++)
			sum += i + j;
	}
	time(&end);
	dif = difftime(end, start);
	printf("%fms", dif);
	return 0;
}
```

✓ difftime() 함수

➢ 초(second) 기준으로 시간 값 반환

✓ clock() 함수

➢ 밀리초(milli second) 기준으로 시간 값 반환

출력 예	2.00s(실행환경에 따라 시간이 바뀜)

12.5 time.h

- time.h의 멤버 함수
  - ✓ 변환 함수
    - ➢ 시간 정보를 다른 데이터 타입으로 변환하는 기능을 가진 함수
    - ➢ [ ] 함수
      - timeptr이 가리키는 시간 관련 구조체 tm을 문자열로 변환
    - ➢ [ ] 함수
      - timer가 가리키는 시간 관련 변수를 문자열로 변환
    - ➢ [ ] 함수
      - timeptr이 가리키는 구조체 tm을 time_t형식의 문자열로 변환

12.5 time.h

```
struct tm {
int tm_sec;
int tm_min;
int tm_hour;
int tm_mday;
int tm_mon;
int tm_year;
int tm_wday;
int tm_yday;
int tm_isdst;
};
```

< 구조체 tm의 구조 >

멤버	의미
tm_sec	현재 시각이 몇 초인지
tm_min	현재 시각이 몇 분인지
tm_hour	현재 시각이 몇 시인지
tm_mday	오늘 날짜
tm_mon	이번 달
tm_year	올해
tm_wday	오늘의 요일
tm_yday	1월 1일부터 몇 일 지났는지
tm_isdst	서머 타임제를 실시 하는지

< 각 멤버 변수의 의미 >

12.5 time.h

소스 12-18

```
1  #include <stdio.h>
2  #include <time.h>
3  int main(void) {
4          time_t rawtime;
5          struct tm * timeinfo;
6          time(&rawtime);
7          timeinfo = localtime(&rawtime);
8          printf("The current date and time : %s", asctime(timeinfo));
9          return 0;
10 }
```

출력 예

```
The current date and time : Wed Oct 28 11:12:41 2020
```


12.5 time.h

소스 12-19

```
1  #include <stdio.h>
2  #include <time.h>
3  int main(void) {
4          time_t timer;
5          struct tm * timeinfo;
6          timer = time(NULL);
7          timeinfo = localtime(&timer);
8          printf("this month : %d \n", timeinfo ->tm_mon +1);
9          printf("today : %d \n", timeinfo ->tm_mday);
10 }
```

출력 예

```
this month : 10
today : 28
```

12.5 time.h

소스 12-20

```
1 #include <stdio.h>
2 #include <time.h>
3 int main(void) {
4       time_t rawtime;
5       time(&rawtime);
6       printf("The current local time : %s", ctime(&rawtime));
7       return 0;
8 }
```

출력 예	The current local time : Wed Oct 28 11:22:51 2020

12.5 time.h

- 자료형
  - ✓ [ ]
    - ISO_C 라이브러리에서 시스템 시간을 저장하도록 정의된 자료형
    - 표준 함수인 time()의 리턴 값
    - <time.h> 헤더 파일에서 정의됨
    - 유닉스나 POSIX 호환 운영체제에서는 32bit나 64bit 부호 있는 정수형으로 정의
    - UTC 1970-1-1 0:00:00 이후 경과한 초

12.5 time.h

- 자료형
  - ✓ [ ]
    - ➢ 프로세스가 현재 실행까지 걸린 클럭(clock) 수
    - ➢ 1 클럭
      - time.h에 정의되어있는 CLOKS_PER_SEC(초당 클럭 수) 로 알 수 있음
      - ex) CLOKS_PER_SEC가 1000으로 정의되어 있다면, 1초당 1000 클럭
  - ✓ [ ]
    - ➢ sizeof 연산자의 결과값을 나타내는 unsigned long형 정수

12.6 기타 헤더 파일

- [ ]
  - ✓ 콘솔 입출력 함수를 제공하는 헤더 파일
  - ✓ 화면과 커서의 위치 제어 가능
  - ✓ 리눅스나 OS X에서는 사용 불가하지만, 윈도우 환경에서는 사용 가능
    - ➢ MS-DOS 시절부터 사용되었으나, 현재는 C언어 표준도 아닐 뿐더러 POSIX 함수도 아니기 때문

12.6 기타 헤더 파일

소스 12-21

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(void) {
    int i;
    for (i =1; i <=9; i++) {
        system("cls");
        printf("%d*%d=%d\n",4,i,4*i );
        printf("아무키나 누르시오.\n");
        _getch();
    }
}
```

✓ 기존의 getch()함수가 _getch()로 변경됨

출력 예

```
4*1=4
아무키나 누르시오.
4*2=8
아무키나 누르시오.
4*3=12
아무키나 누르시오.
```


12.6 기타 헤더 파일

- [ ]

✓ C의 표준 라이브러리 내의 헤더 파일

✓ errno라는 정적 메모리 위치에 저장된 오류 코드를 통해 오류 상태를 보고 및 검색하기 위한 매크로 정의

✓ 오류 번호는 에러를 감지했을 때, 특정 라이브러리 함수에 의해 errno에 저장됨

✓ 함수 실행 중 에러가 발생하지 않고 실행된다면 errno는 0의 값 가지고, 수행 중 에러가 발생하면 0이외의 값을 가짐

12.6 기타 헤더 파일

소스 12-22

```c
#include <stdio.h>
#include <errno.h>
int main(void) {
    FILE* fp;
    fp = fopen("./test.c", "r"); //exist file
    printf("error = %d\n", errno);
    fclose(fp);
    fp = fopen("./nofile.c", "r"); //not exist file
    printf("error = %d\n", errno);
    return 0;
}
```

출력 예

```
error = 0
error = 2
```


12.7 사용자 헤더 파일

- 사용자 헤더 파일
  - ✓ 사용자가 [ ] 하는 헤더 파일
  - ✓ C 표준 라이브러리 함수를 포함하는 시스템 헤더 파일과 같이 사용자가 만든 함수들을 라이브러리처럼 사용할 수 있도록 함

    ex) `printf()`함수를 대신하는 함수 `myprintf()`를 프로그래머가 직접 작성했다면, 헤더 파일을 만들고 사용할 수 있음

12.7 사용자 헤더 파일

- 사용자 헤더 파일
  - ✓ 여러 개의 .c 파일을 [ ]로 묶어 관리할 수 있음
    - ➢ 보통 사용자 헤더 파일은 프로그램 관련 코드의 길이가 길어져서 함수나 변수의 선언부와 정의부를 따로 분리해서 관리하도록 하는 경우에 사용하기도 함

헤더 파일	구현 함수	메인 함수

#ifndef _MYFUNC_H
#define _MYFUNC_H
int add(int a, int b);
#endif
	#include "MyFunc.h"
	int add(int a, int b)
	{
	int c;
	c = a +b;
	return c;
	}
		#include <stdio.h>
		#include "MyFunc.h"
		int main(void)
		{
		int c = add(10,20);
		printf("10 + 20 = %d\n", c);
		return 0;
		}

< 파일을 나누어 저장한 프로그램 예시 >

12.7 사용자 헤더 파일

- 사용자 헤더 파일
  - ✓주로 데이터 타입을 표시하는 코드를 작성하는 것이 일반적
  - ✓프로그램 실행 시, 변수 공간을 중복 할당하는 경우 프로그램에 문제 야기
    - ➢ 여러 파일에서 중복으로 #include 전처리기를 이용하여 헤더 파일을 포함하기 때문
  - ✓헤더 파일에는 주로 #define, enum, struct 헤더 부분, 함수 헤더 부분 구성
  - ✓소스 코드가 복잡해지는 경우 프로그래밍 시 변수 사용에 어려움 겪을 수 있으므로 잘 정의하여 헤더 파일을 사용하는 습관 필요

12.7 사용자 헤더 파일

소스 12-23

MyFunc.h

```
1 #ifndef _MYFUNC_H
2 #define _MYFUNC_H
3 int add(int i1, int i2);
4 int sub(int i1, int i2);
5 int multi(int i1, int i2);
6 double div(double d1, double d2);
7 #endif
```

MyFunc.c

```
1  #include "MyFunc.h"
2  int add(int i1, int i2) {
3          int i3;
4          i3 = i1 + i2;
5          return i3;
6  }
7  int sub(int i1, int i2) {
8          int i3;
9          i3 = i1 - i2;
10         return i3;
11 }
12 int multi(int i1, int i2) {
13         int i3;
14         i3 = i1 * i2;
15         return i3;
16 }
17 double div(double d1, double d2) {
18         double d3;
19         d3 = d1 / d2;
20         return d3;
21 }
```


12.7 사용자 헤더 파일

main.c

```
1  #include <stdio.h>
2  #include "MyFunc.h"
3  int main(void) {
4          int add_i = add(10, 20);
5          int sub_i = sub(10, 20);
6          int multi_i = multi(10, 20);
7          double div_d = div(10, 20);
8          printf("10 + 20 = %d\n", add_i);
9          printf("10 - 20 = %d\n", sub_i);
10         printf("10 * 20 = %d\n", multi_i);
11         printf("10 / 20 = %.2f\n", div_d);
12         return 0;
13 }
```

출력 예

```
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.50
```

12.7 사용자 헤더 파일

소스 12-24

MyString.h

```
1  #pragma once
2  #ifndef _MYFUNC_H
3  #define _MYFUNC_H
4  void my_strcat(char * pch1, char * pch2);
5  #endif
```

MyString.c

```
1  #include "MyString.h"
2  void my_strcat(char * pch1, char * pch2) {
3          while (*pch1) {
4                  *pch1++;
5          }
6          while (*pch1 = *pch2) {
7                  *pch1++;
8                  *pch2++;
9          }
10 }
```


12.7 사용자 헤더 파일

main.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include "MyString.h"
4  int main(void) {
5          char arr1[11] = "Hello";
6          char arr2[] = "wo";
7          char arr3[11] = "hello";
8          char arr4[] = "WO";
9          strcat(arr1, arr2);
10         strcat(arr1, "LRD");
11         printf("%s\n", arr1);
12         my_strcat(arr3, arr4);
13         my_strcat(arr3, "lrd");
14         printf("%s\n", arr3);
15         return 0;
16 }
```

출력 예	HellowoLRD helloWOlrd

12.8 Q&A

Q 49. 여러 개의 파일로 이루어진 프로그램을 만들었다. 그런데, 어떤 것들을 .c 파일에 두어야 하고, 어떤 것들을 .h 파일에 두어야 하는지를 모르겠다.

A . 보통 헤더(.h) 파일에 넣는 것들은 다음과 같다.

- 매크로 정의(#define)
- structure, union, enum 선언
- typedef 선언
- 외부(external) 함수 선언
- 전역(global) 변수 선언

특히 여러 파일에서 공통적으로 쓰이는 선언이나 정의는 꼭 헤더 파일에 넣는 것이 중요합니다. 공통적으로 쓰이는 이름을 여러 소스 파일에 중복해서 선언하거나 정의하지 말기 바란다; 이런 부분들은 헤더 파일에 집어 넣고, #include를 써서 포함해야 한다. 단순히 타이핑하는 수고를 덜기 위해 이러한 규칙을 정한 것이 아니다. 만약 공통적인 부분을 나중에 고쳐야 한다면, 한 파일을 고쳐서, 이 변경 사항이 모든 소스 파일에 반영되도록 해야 하기 때문이다. 또, 정의나 선언이 하나의 .c 파일에서만 쓰인다면, 그 파일에 두어도 좋다.
마지막으로, 실제 코드(actual code)나 (예를 들어, 함수의 몸체), 전역 변수 정의를 (정의 또는 초기화하는 코드) 헤더 파일에 두면 안된다. 또, 여러 소스 파일에서부터 프로젝트를 설계했다면, 각각의 소스 파일을 따로 컴파일하고 (대개 컴파일러 옵션을 써서 컴파일만 하게 할 수 있습니다) 마지막으로 모든 오브젝트 파일을 링크해야 한다.

12.8 Q&A

Q 50. 헤더 파일에서 다른 파일을 #include하는 것은 괜찮나?

A . 스타일에 관한 질문은 상당한 논란의 여지가 있다. 많은 사람들이 중첩된(nested) #include 파일"을 쓰지 않는 것이 좋다고 말한다. 권위있는 Indian Hill Style Guide 에서도 이런 쓰임새를 피하라고 쓰여있다. 관련된 정의를 찾기가 훨씬 더 어렵기 때문이다. 또한 두번 #include하는 경우, 중복된 정의 에러가 (multiple-denition error) 발생할 가능성이 높다. 또 수동으로 Makele을 만들 경우, 상당히 복잡해질 가능성이 있다. 그러나 헤더 파일을 중첩하여 포함할 경우, 각각의 헤더 파일을 모듈화해서 (modular way), 헤더 파일에서 필요한 다른 헤더 파일을 #include함으로써 수고를 덜어줄 수 있다는 장점도 있다. grep과 같은 툴을 (또는 tags 파일) 사용하면 정의가 어떤 파일에 되어 있느냐에 상관없이 쉽게 찾을 수 있다. 다음과 같은 트릭을 쓰면, 헤더 파일이 여러 곳에서 포함되었느냐에 상관없이, 딱 한번만 포함되게(idempotent) 할 수 있습니다.

12.9 실습 및 과제

- 실습 및 과제 진행
  - DCU Code : http://code.cu.ac.kr
  - 12 주차 실습
    - 코딩 12-1, 코딩 12-2, 코딩 12-3, 코딩 12-4, 코딩 12-5
  - 12 주차 과제
    - 12장 프로그래밍 연습 1-7번

12.10 참고 문헌

- 위키백과 C 표준 라이브러리 참조(https://ko.wikipedia.org/wiki/C_표준_라이브러리)
- 위키피디아 stdio.h 참조(https://ko.wikipedia.org/wiki/Stdio.h)
- 위키백과 C 파일 입출력 참조(https://ko.wikipedia.org/wiki/C_파일_입출력)
- 위키백과 stdlib.h 참조(https://ko.wikipedia.org/wiki/Stdlib.h)
- 위키백과 C 문자 분류 참조(https://ko.wikipedia.org/wiki/C_문자_분류)
- 위키백과 C 수식 함수 참조(https://ko.wikipedia.org/wiki/C_수식_함수)
- 위키백과 C 날짜와 시간 함수 참조(https://ko.wikipedia.org/wiki/C_날짜와_시간_함수)
- 위키백과 Errno.h 참조(https://ko.wikipedia.org/wiki/Errno.h)
- 위키백과 헤더파일 참조(https://ko.wikipedia.org/wiki/헤더파일)
- C Programming FAQs 참조 (http://cinsk.github.io/ko/cfaqs/index.html)

E N D