Clickseo.com

C Programming

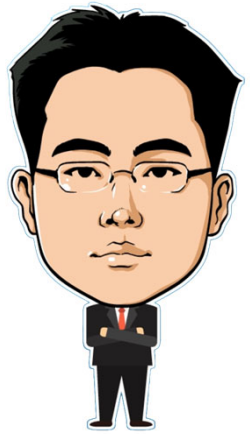
C 표준 라이브러리

(C Standard Library)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

표준 라이브러리 (1/2)

- **표준 라이브러리 함수 (standard library functions)**
  - C 언어와 C++ 언어에는 프로그래머의 편의성을 도모하기 위해 프로그램 언어 개발자들에 의해 작성되어 포함된 함수
  - **C 표준 라이브러리** (C standard library)
    - 'ISO C 라이브러리' 라고도 한다.
    - 'C POSIX 라이브러리' 와 거의 동시에 개발
  - **glibc** (The GNU C Library)
  - 표준 라이브러리 함수를 불러 들이기 위해 전처리 구문 영역에 **#include** 문을 사용
    - 대표적인 표준 라이브러리 함수인 scanf와 printf함수를 사용하기 위해 헤더파일 <stdio.h> 를 이용한다.

표준 라이브러리 (2/2)

- **C 표준 라이브러리**
  - 헤더 파일
    - 하나 이상의 함수 원형 선언과 자료형의 정의 그리고 다양한 매크로들을 포함
    - 현재 총 29개의 헤더 파일을 제공
  - 1995년, 규범 별첨 1(NA1)에서 3개의 헤더 파일 추가
    - iso646.h, wchar.h 그리고 wctype.h
  - 1999년, C99 버전에서 새롭게 6개의 헤더 파일 추가
    - complex.h, fenv.h, inttypes.h, stdbool.h, stdint.h 그리고 tgmath.h
  - 2011년, C11 버전에서 5개의 헤더 파일이 더 추가
    - stdalign.h, stdatomic.h, stdnoreturn.h, threads.h 그리고 uchar.h

C 표준 라이브러리

표준 입출력 : <stdio.h>

표준 입출력 (1/14)

● 입출력 함수

❍ 스트림(stream) : 디스크나 그 외 장치를 오가는 데이터의 모임

- 표준 입출력 라이브러리는 **text stream** 또는 **binary stream**을 처리
  - **text stream** : '₩n'으로 끝나는 행들의 모임
  - **binary stream** : 아무런 처리를 거치지 않은 데이터들의 모임

- **stream**은 '**open**'을 통해서 파일이나 '**device**'로 바꾸어 질 수 있고, '**close**'에 의해 종료된다.
  - 파일을 **open**하면 그 파일의 포인터가 반환 된다.

표준 입출력 (2/14)

- 데이터 유형 (Data Type)

```
#include <stdio.h>

typedef unsigned int size_t;

struct _iobuf
{
        char *_ptr;
        int   _cnt;
        char *_base;
        int   _flag;
        int   _file;
        int   _charbuf;
        int   _bufsiz;
        char *_tmpfname;
};
typedef struct _iobuf FILE;

typedef __int64 fpos_t;
```

표준 입출력 (3/14)

- **매크로 (Macro) : 변수 및 상수**

```
#include <stdio.h>

extern  FILE  __iob[FOPEN_MAX]
#define  stdin        (&__iob[0])          // 표준 입력 스트림
#define  stdout       (&__iob[1])          // 표준 출력 스트림
#define  stderr       (&__iob[2])          // 프로그램의 오류 메시지나 다른 예외적인
                                              내용을 출력하기 위한 출력 스트림
#define  EOF          (-1)
#define  NULL         ((void  *) 0)

#define  BUFSIZ       ...          // 256KB, 512KB 또는 4096KB

#define  FOPEN_MAX  ...            // 동시에 개방할  수 있는 스트림 개수(최소 8 이상)
#define  FILENAME_MAX  ...         // 개방할 수 있는 파일 이름의 최대 길이

#define  SEEK_SET  0
#define  SEEK_CUR 1
#define  SEEK_END 2
```

표준 입출력 (4/14)

- **파일 접근(File access) 함수**

```
#include <stdio.h>

FILE *fopen(const char *filename, const char *mode);

// stdin, stdout, stderr 중 하나를 다른 파일과 다시 연결할 때 주로 사용
FILE *freopen(const char *filename, const char *mode, FILE *stream);
```

반환(성공) : 개방된 파일의 스트림의 시작 주소를 반환
(실패) : **NULL** 값을 반환

```
#include <stdio.h>

FILE *fclose(FILE *stream);
int fflush(FILE *stream);
```

반환(성공) : **0** 값을 반환
(실패) : **EOF** 값을 반환

표준 입출력 (5/14)

- **직접적인 입출력(Direct I/O) 함수**

```
#include <stdio.h>

size_t fread(void *buffer, size_t size, size_t count, FILE *stream);
size_t fwrite(const void *buffer, size_t size, size_t count, FILE *stream);
```

- *buffer* : 읽기 또는 기록 할 메모리 시작 주소
- *size* : 각 객체(블록)의 크기
- *buffer* : 기록 할 객체(블록)의 수
- *size* : 개방된 스트림

반환(성공) : 성공적으로 읽기 도는 기록한 객체(블록) 수*(count)*를 반환
(실패) : 오류 발생

표준 입출력 (6/14)

● 파일 위치(File positioning) 함수 (1/2)

```
#include <stdio.h>

long  ftell(FILE  *stream);
```

반환(성공) : 지정된 파일 스트림에서 파일 포인터의 위치 값을 반환
(실패) : **EOF** 값을 반환

```
#include <stdio.h>

// 개방된 파일 스트림에서 파일 포인터의 위치를 직접 설정
int  fseek(FILE  *stream,  long  offset,  int  whence);
```

반환(성공) : **0** 값을 반환
(실패) : **0** 이외의 값을 반환

```
#include <stdio.h>

// (void)fseek(stream, 0L, SEEK_SET)과 동일
void  rewind(FILE  *stream);
```

표준 입출력 (7/14)

파일 위치 함수 (2/2)

```
#include <stdio.h>

int fgetpos(FILE *stream, fpos_t *pos);
int fsetpos(FILE *stream, const fpos_t *pos);
```

반환(성공) : **0** 값을 반환
(실패) : **0** 이외의 값을 반환

`ftell` 함수와 `fseek` 함수 사용 시 단점 보완

- **fgetpos** 함수와 **fsetpos** 함수는 `ftell` 함수와 `fseek` 함수 사용 시 파일의 크기가 너무 커서 그 위치를 자료형 `long int`로 표현할 수 없는 경우를 위해 표준 C 언어에서 추가된 파일 위치 지정 함수

표준 입출력 (8/14)

- **형식화된 입출력(Formatted I/O) 함수**

```
#include <stdio.h>

int  scanf(const char  *format,  ...);
int  fscanf(FILE  *stream,  const char  *format,  ...);
int  sscanf(char  *buffer,  const  char  *format,  ...);
```

반환(성공) : 입력에 성공한 필드 개수를 반환
(실패) : **EOF** 값을 반환

```
#include <stdio.h>

int  printf(const char  *format,  ...);
int  fprintf(FILE *stream,  const char  *format, ...);
int  sprintf(char  *buffer , const char  *format,  ...);
int  snprintf(char  *buffer , int buf_siz, const char  *format,  ...);          // C99
```

반환(성공) : 출력한 바이트 수를 반환
(실패) : **EOF** 값을 반환

표준 입출력 (9/14)

- **비형식화 된 입출력(Unformatted I/O) 함수 (1/3)**

```
#include <stdio.h>

int  fgetc(FILE  *stream);
int  getc(FILE  *stream);

int  fputc(int  ch,  FILE  *stream);
int  putc(int  ch,  FILE  *stream);

int  ungetc(int  ch,  FILE  *stream);
```

반환(성공) : 스트림으로부터 읽거나 기록한 문자를 반환
(실패) : **EOF** 값을 반환

표준 입출력 (10/14)

- **비형식화 된 입출력 함수 (2/3)**

```
#include <stdio.h>

int  getchar();                 // int  getc(stdin)
int  putchar(int  ch);          // int  putc(ch,  stdout)
```

반환(성공) : 표준 입출력에서 읽거나 가록한 문자를 반환
(실패) : **EOF** 값을 반환

표준 입출력 (11/14)

● 비형식화 된 입출력 함수 (3/3)

```
#include <stdio.h>

char  *gets (char  *str);
char  *gets_s (char  *str,  rsize_t  n);                    // C11
char  *fgets (char  *str,  int  count,  FILE  *stream);
```

반환(성공) : *str*의 주소를 반환
(실패) : **NULL** 을 반환

```
#include <stdio.h>

int  puts (const  char  *str);
int  fputs (const  char  *str,  FILE  *stream);
```

반환(성공) : 음수가 아닌 정수를 반환
(실패) : **EOF** 를 반환

표준 입출력 (12/14)

- **오류 처리(Error handling) 함수**
  - 라이브러리에 있는 많은 함수들은 에러가 발생했을 때 에러가 발생했음을 알려준다.
    - <errno.h>에는 errno라는 변수가 선언되어 있어서 어떤 에러가 발생했는지를 알 수 있게 되어 있다.

```
#include <stdio.h>

// 해당 입력 스트림으로부터 파일의 끝에 도달했는지를 검사
// 파일의 끝에 도달한 경우 0이 아닌 값을, 도달하지 않은 경우에는 0을 반환
int  feof(FILE  *stream);

// 스트림의 오류 상태를 반환
// 입출력 도중 오류가 발생했다면 0이 아닌 값을, 그렇지 않다면 0을 반환
int  ferror(FILE  *stream);

// 주어진 스트림의 오류 발생 여부나 파일 끝에 도달했는지에 대한 정보를 초기화
void  clearerr(FILE  *stream);
```

표준 입출력 (13/14)

- **오류 처리 함수 (cont'd)**

```
#include <stdio.h>

// str과 error에 해당하는 에러 메시지를 출력한다.
// 출력 형식 : fprintf(stderr, "%s : %s₩n", str, "error message")
void  perror(const  char  *str);
```

표준 입출력 (14/14)

- **파일에 대한 작업(Operations on files) 함수**

```
#include <stdio.h>

// 지정된 파일(filename)을 삭제 한다.
int remove(const char *filename);

// 기존 파일 이름(old_filename)을 새로운 파일 이름(new_filename)으로 변경한다.
int rename(const char *old_filename, const char *new_filename);
```

호출 성공 : **0** 을 반환
에러 발생 : **0** 이 아닌 값을 반환

C 표준 라이브러리

문자 처리 : <ctype.h>

문자 분류 (1/4)

- **매크로(Macro) : 변수 및 상수**

```
#include <ctype.h>

#define  _UPPER          0x1      // upper case letter
#define  _LOWER          0x2      // lower case letter
#define  _DIGIT          0x4      // digit[0-9]
#define  _SPACE          0x8      // tab, carriage return, newline,
                                  // vertical tab or form feed
#define  _PUNCT          0x10     // punctuation character
#define  _CONTROL        0x20     // control character
#define  _BLANK          0x40     // space char
#define  _HEX            0x80     // hexadecimal digit

#define  _LEADBYTE       0x8000   // multibyte leadbyte
#define  _ALPHA    (0x0100|_UPPER|_LOWER)   // alphabetic character
```

문자 분류 (2/4)

● 문자 분류 함수 (1/2)

❍ 주어진 분류에 속하고 있는지 그렇지 않은지를 알려준다.

- 접두어 "is" 로 시작

판단 하려는 문자인 경우 : 0 이외의 값을 반환
판단 하려는 문자가 아닌 경우 : 0 값을 반환

```
#include <ctype.h>

int  islower( int  ch );          // 영문 소문자 여부 판단
int  isupper( int  ch );          // 영문 대문자 여부 판단

int  isdigit( int  ch );          // 10진수 숫자 문자 여부 판단
int  isxdigit( int  ch );         // 16진수 숫자 문자 여부 판단

// 영문 대소문자 여부 판단
int  isalpha( int  ch );          // isupper(ch) 또는 islower(ch)가 만족되는 경우

// 영문 대소문자와 숫자 여부 판단
int  isalnum( int  ch );          // isalpha(ch) 또는 isdigit(ch)가 만족되는 경우
```

문자 분류 (3/4)

문자 분류 함수 (2/2)

- 주어진 분류에 속하고 있는지 그렇지 않은지를 알려준다.

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

판단 하려는 문자인 경우 : 0 이외의 값을 반환
판단 하려는 문자가 아닌 경우 : 0 값을 반환

```
int  isblank( int  ch );        // ' ', '₩t'
int  isspace( int  ch );        // ' ', '₩f', '₩r', '₩n', '₩t', '₩v'

int  iscntrl( int  ch );        // 제어 문자 여부 판단
int  isgraph( int  ch );        // 그래픽 문자 여부 판단

int  isprint( int  ch );        // 주어진 문자를 인쇄할 수 있는지 여부 판단
int  ispunct( int  ch );        // 영문자, 숫자 공백이 아닌 특수 문자 여부 판단
```

문자 분류 (4/4)

ASCII values (hex)		characters	iscntrl iswcntrl	isprint iswprint	isspace iswspace	isblank iswblank	isgraph iswgraph	ispunct iswpunct	isalnum iswalnum	isalpha iswalpha	isupper iswupper	islower iswlower	isdigit iswdigit	isxdigit iswxdigit
0 - 8	0x00-0x08	control codes (NUL, etc.)	≠0	0	0	0	0	0	0	0	0	0	0	0
9	0x09	tab (\t)	≠0	0	≠0	≠0	0	0	0	0	0	0	0	0
10 - 13	0x0A-0x0D	whitespaces (\n,\v,\f,\r)	≠0	0	≠0	0	0	0	0	0	0	0	0	0
14 - 31	0x0E-0x1F	control codes	≠0	0	0	0	0	0	0	0	0	0	0	0
32	0x20	space	0	≠0	≠0	≠0	0	0	0	0	0	0	0	0
33 - 47	0x21-0x2F	!"#$%&'()*+,-./	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
48 - 57	0x30-0x39	0123456789	0	≠0	0	0	≠0	0	≠0	0	0	0	≠0	≠0
58 - 64	0x3a-0x40	:;<=>?@	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
65 - 70	0x41-0x46	ABCDEF	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	≠0
71 - 90	0x47-0x5A	GHIJKLMNOPQRSTUVWXYZ	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	0
91 - 96	0x5B-0x60	[\]^_`	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
97 -102	0x61-0x66	abcdef	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	≠0
103-122	0x67-0x7A	ghijklmnopqrstuvwxyz	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	0
123-126	0x7B-0x7E	{|}~	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
127	0x7F	backspace character (DEL)	≠0	0	0	0	0	0	0	0	0	0	0	0

[출처 : http://en.wikipedia.org/wiki/C_character_classification]

문자 변환 (1/2)

● 문자 변환 함수

❍ 영문 대문자를 소문자로 또는 영문 소문자를 대문자로 변환

- 접두어 "to" 로 시작
- 변환된 문자 값인 정수를 반환

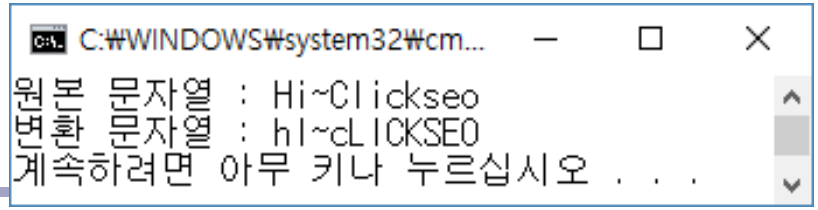
```
#include <ctype.h>

int  tolower(int  ch );          // 문자 ch 를 소문자로 변환
int  toupper(int  ch );          // 문자 ch 를 대문자로 변환
```

대소문자 변환 성공 : **변환 된 문자** ch 값(ASCII code) **반환**
대소문자 변환 실패 : 기존 문자 ch 값(ASCII code) 반환

문자 변환 (2/2)

예제 6-1 : 문자 분류 및 변환 함수

```
#include <stdio.h>
#include <ctype.h>      // islower, isupper, tolower, toupper

int main(void)
{
        char    str[] = "Hi~Clickseo";
        char    *pStr = str;

        fputs("원본 문자열 : ", stdout);
        puts(str);

        printf("변환 문자열 : ");
        for (; *pStr != '\0'; pStr++)
        {
                if ( islower(*pStr) )       putchar( toupper(*pStr) );
                else if ( isupper(*pStr) )  putchar( tolower(*pStr) );
                else                        putchar( *pStr );
        }
        putchar('\n');

        return 0;
}
```



C 표준 라이브러리

문자열 처리 : <string.h>

문자열 처리 (1/8)

- 데이터 유형(Data Type)

```
#include <string.h>

typedef  unsigned  int  size_t;
typedef  unsigned  short  wchar_t;
```

- 매크로(Macro) : 변수 및 상수

```
#include <string.h>

#define  NULL        ((void  *) 0)
```

문자열 처리 (2/8)

- **문자열 조작(String manipulation) 함수**
  - 널 문자를 포함한 원본 문자열을 목적지 문자열로 복사

```
#include <string.h>

char *strcpy (char *dest, const char *src);
char *strncpy (char *dest, const char *src, size_t size);
```

반환 값 : 대상이 되는 메모리 공간의 시작 주소*(dest)*

  - 한 문자열을 다른 문자열의 끝에 연결

```
#include <string.h>

char *strcat (char *dest, const char *src);
char *strncat (char *dest, const char *src, size_t size);
```

반환 값 : 대상이 되는 메모리 공간의 시작 주소*(dest)*

문자열 처리 (3/8)

● 문자열 검사(String examination) 함수 (1/4)

❍ 문자열 길이를 반환

```
#include <string.h>

size_t      strlen (const char *str);
```

반환 값 : 대상 문자열의 길이(byte) 반환

❍ 두 개의 문자열을 비교

```
#include <string.h>

int  strcmp (const  char  *str1,  const  char  *str2);
int  strncmp (const  char  *str1,  const  char  *str2,  size_t  size);
```

- 반환 값 : `(str1 < str2)` 음수 값을 반환
  `(str1 > str2)` 양수 값을 반환
  `(str1 == str2)` 0 값을 반환

문자열 처리 (4/8)

문자열 검사 함수 (2/4)

```
#include <string.h>

// 문자가 문자열에 존재하는지 검색
char *strchr (const char *str, int ch);
char *strrchr (const char *str, int ch);

// 문자열에서 부분 문자열이 존재하는지 검색
char *strstr (const char *dest, const char *src);
```

호출 성공 : 검색된 문자 또는 문자열 위치의 메모리 주소를 반환
에러 발생 : **NULL** 값을 반환

- **strchr** : 문자열의 처음 부터 처음으로 일치하는 문자를 검색
- **strrchr** : 문자열의 끝에서 부터 일치하는 문자를 검색

문자열 처리 (5/8)

● 문자열 검사 함수 (3/4)

```
#include <string.h>

size_t *strspn(const char *dest, const char *src);
size_t *strcspn(const char *dest, const char *src);
```

호출 성공 : 찾아낸 세그먼트의 길이
에러 발생 : **0** 값을 반환(세그먼트가 존재하지 않을 경우)

- **strspn** : 원본 문자열에서 대상 문자열의 순서대로 존재하지 않더라도,
  대상 문자열에 존재하는 문자들 중에서 원본 문자열에
  연속적으로 몇 개나 존재하는 검색하는 함수

- **strcspn** : 대상 문자열을 구성하는 문자가
  원본 문자열에 연속적으로 존재하지 않는 길이를 구하는 함수

문자열 처리 (6/8)

문자열 검사 함수 (4/4)

```
#include <string.h>

const char *strpbrk(const char *dest, const char *str);
```

호출 성공 : 문자열 str 에 있는 문자 중 첫 번째로 발견된 위치(주소)
에러 발생 : **NULL** 값을 반환(한 문자도 존재하지 않을 경우)

- **strpbrk** : 특정 문자열에서 여러 개의 문자 중 하나라도 존재하는지 검색

```
#include <string.h>

char *strtok(char *str, const char *token);
```

호출 성공 : 찾아낸 **token** 위치(주소)
에러 발생 : **NULL** 값을 반환(**token** 이 더 이상 없을 경우)

- **strtok** : 원하는 구분자를 이용하여 문자열을 쪼갤 수 있도록 하는 함수

문자열 처리 (7/8)

● 메모리 복사(Memory manipulation) 함수 (1/2)

```
#include <string.h>

void *memset(void *dest, int ch, size_t count);
void *memcpy(void *dest, const void *src, size_t count);
void *memmove(void *dest, const void *src, size_t count);
```

호출 성공 : 대상이 되는 메모리 공간의 시작 주소*(dest)*
에러 발생 : **NULL** 값을 반환

- **memset** : 주어진 메모리 공간에 특정 데이터를 저장
- **memcpy** : 주어진 메모리 공간을 원하는 만큼 복사
- **memmove** : **memcpy** 함수와 그 기능이 비슷하다.

문자열 처리 (8/8)

메모리 복사 함수 (2/2)

```
#include <string.h>

int memcmp(const void *dest, const void *src, size_t count);
```

- **memcmp** : 주어진 메모리 공간을 원하는 만큼 비교
- 반환 값 : `(des < src)` 음수 값을 반환
  `(des > src)` 양수 값을 반환
  `(des == src)` 0 값을 반환

```
#include <string.h>

int memchr(const void *dest, int ch, size_t count);
```

호출 성공 : 검색 할 문자(*ch*)를 발견한 가장 첫 번째 메모리 주소
에러 발생 : **NULL** 값을 반환

- **memchr** : 특정 메모리 공간에서 사용자가 지정한 문자를 검색

C 표준 라이브러리

표준 라이브러리 : <stdlib.h>

표준 라이브러리 (1/15)

- 데이터 유형 (Data Type)

```
#include <stdlib.h>

typedef  unsigned  int  size_t;
typedef  unsigned  short  wchar_t;

// 몫과 나머지를 계산하는 div, ldiv 그리고 lldiv 함수에서 사용하는 데이터 구조
typedef struct  _div_t
{
        int   quot;          // 몫 (The quotient)
        int   rem;           // 나머지 (The remainder)
}div_t;

sturct  _ldiv_t
{
        long   quot;         // 몫 (The quotient)
        long   rem;          // 나머지 (The remainder)
}ldiv_t;

sturct  _lldiv_t
{
        long  long  quot;    // 몫 (The quotient)
        long  long  rem;     // 나머지 (The remainder)
}lldiv_t;
```

표준 라이브러리 (2/15)

- **매크로 (Macro) : 변수 및 상수 그리고 매크로 함수**

```
#include <stdlib.h>

#define  NULL        ((void  *) 0)

// exit 함수를 위해 정의된  값
#define  EXIT_SUCCESS    0
#define  EXIT_FAILURE    1

// rand 함수 사용 시 임의의 난수 발생 최대값
#define  RAND_MAX  0x7fff

#define  __max(a,  b)  (((a)  >  (b))  ?  (a)  :  (b))
#define  __min(a,  b)  (((a)  <  (b))  ?  (a)  :  (b))
```

표준 라이브러리 (3/15)

- **숫자를 문자열로 변환하는 함수**
  - 정수형 숫자를 2진수, 8진수, 10진수 또는 16진수의 문자열로 변환
    - itoa 함수 : integer to ascii
    - ltoa 함수 : long to ascii

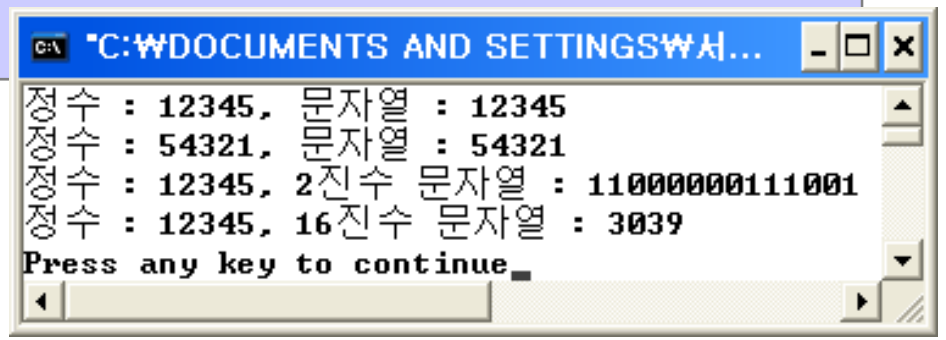
```
#include <stdlib.h>

// int 형 숫자를 문자열 str 로 변환
char  *itoa (int  value,  char  *str,  int  radix);

// long 형 숫자를 문자열 str 로 변환
char  *ltoa (long  value,  char  *str,  int  radix);
```

호출 성공 : 변환된 문자열의 시작 주소(str)
범위 오류 : **NULL** 값을 반환

표준 라이브러리 (4/15)

프로그램 예제 : 숫자를 문자열로 변환

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

int main(void)
{
        int             num1 = 12345;
        long int        num2 = 54321;
        char            str[1024];

        itoa(num1, str, 10);
        printf("정수 : %d, 문자열 : %s \n", num1, str);

        ltoa(num2, str, 10);
        printf("정수 : %ld, 문자열 : %s \n", num2, str);

        itoa(num1, str, 2);
        printf("정수 : %d, 2진수 문자열 : %s \n", num1, str);

        itoa(num1, str, 16);
        printf("정수 : %d, 16진수 문자열 : %s \n", num1, str);

        return 0;
}
```



표준 라이브러리 (5/15)

● 문자열을 숫자로 변환하는 함수 (1/4)

❍ 문자열을 숫자로(실수형 또는 정수형) 변환

```
#include <stdlib.h>

// 문자열 str 을 int, long  또는 long long 형 정수로 변환
int  atoi (const  char  *str);
long  atol (const  char  *str);
long  long  atoll (const  char  *str);                    // C99
```

호출 성공 : 변환된 정수형 값을 반환
범위 오류 : **0** 값을 반환(더 이상 변환을 수행할 수 없을 경우)

```
#include <stdlib.h>

// 문자열 str 을 double 형 실수로 변환
double  atof (const  char  *str);
```

호출 성공 : 변환된 실수형 값을 반환
범위 오류 : **0.0** 값을 반환(더 이상 변환을 수행할 수 없을 경우)

표준 라이브러리 (6/15)

● 문자열을 숫자로 변환하는 함수 (2/4)

❍ 문자열을 정수(long 또는 long long) 값으로 변환

```
#include <stdlib.h>

// 문자열 str 을 long 또는 long long 형 정수로 변환
long strtol (const char *str, char **str_end, int radix);
long long strtoll (const char *str, char **str_end, int radix);        // C99
```

호출 성공 : 변환된 정수형 값을 반환
범위 오류 : **LONG_MAX, LONG_MIN, LLONG_MAX**
또는 **LLONG_MIN** 반환
0 값을 반환(더 이상 변환을 수행할 수 없을 경우)

- 문자열의 구조
  [공백 또는 탭 문자] [부호 (+ 또는 -)] [숫자 ('0' ~' 9')
- *str_end* 는 변환이 중지 될 경우, 중지된 문자를 가리키는 용도(에러 검색 시 사용)
- *radix*는 해석할 진법을 지정

표준 라이브러리 (7/15)

● 문자열을 숫자로 변환하는 함수 (3/4)

❍ 문자열을 부호 없는 정수(long) 값으로 변환

```
#include <stdlib.h>

// 문자열 str 을 long 또는 long long 형 정수로 변환
unsigned long  strtoul (const  char  *str,  char  **str_end,  int  radix);
unsigned long long  strtoull (const  char  *str,  char  **str_end,  int  radix);        // C99
```

호출 성공 : 변환된 정수형 값을 반환
범위 오류 : **ULONG_MAX, ULLONG_MAX** 반환
0 값을 반환(더 이상 변환을 수행할 수 없을 경우)

- 문자열의 구조
  [공백 또는 탭 문자] [부호 (+ 또는 –)] [숫자 ('0' ~' 9')
- *str_end* 는 변환이 중지 될 경우, 중지된 문자를 가리키는 용도(에러 검색 시 사용)
- *radix*는 해석할 진법을 지정

표준 라이브러리 (8/15)

● 문자열을 숫자로 변환하는 함수 (4/4)

❍ 문자열을 부동 소수점(float, double, long double)으로 변환

```
#include <stdlib.h>

// 문자열 str 을 double 형 실수로 변환
float  strtof (const  char  *str,  char  **str_end);          // C99
double  strtod (const  char  *str,  char  **str_end);
long doube  strtold (const  char  *str,  char  **str_end);    // C99
```

호출 성공 : 변환된 부동 소수점 값을 반환
범위 오류 : **HUGE_VAL, HUGE_VALF** 또는 **HUGE_VALL** 반환
0 값을 반환(더 이상 변환을 수행할 수 없을 경우)

- 문자열의 구조

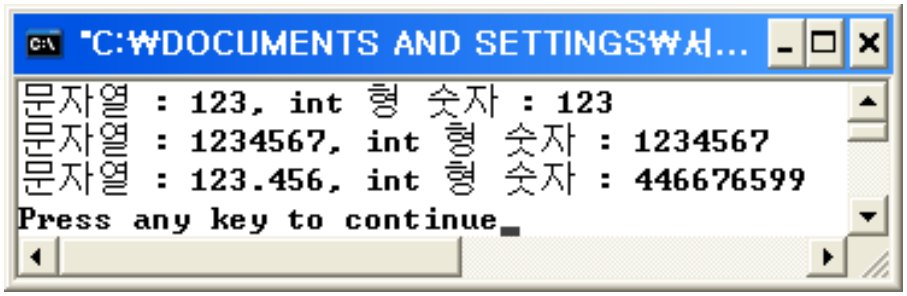
  [공백 또는 탭 문자] [부호 (+ 또는 –)] [숫자 ('0' ~' 9')] [소수점 (.)] [e]

- *str_end* 는 변환이 중지 될 경우, 중지된 문자를 가리키는 용도(에러 검색 시 사용)

표준 라이브러리 (9/15)

프로그램 예제 : 문자열을 숫자로 변환

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

int main(void)
{
        char *str1 = "123";
        char *str2 = "1234567";
        char *str3 = "123.456";

        int             num1;
        long int        num2;
        double          num3;

        num1 = atoi(str1);
        num2 = atol(str2);
        num3 = atof(str3);

        printf("문자열 : %s, int 형 숫자 : %d \n", str1, num1);
        printf("문자열 : %s, int 형 숫자 : %d \n", str2, num2);
        printf("문자열 : %s, int 형 숫자 : %d \n", str3, num3);

        return 0;
}
```



표준 라이브러리 (10/15)

- **메모리 관련 함수**
  - 동적 메모리 할당 및 해제 함수

```
#include <stdlib.h>


void  *malloc (size_t  size);
void  *calloc(size_t  num,  size_t  size);
void  *realloc (void  *ptr,  size_t  new_size);
```

호출 성공 : 동적으로 할당 받은 메모리 공간의 시작 주소를 반환
범위 오류 : **NULL** 값을 반환

```
#include <stdlib.h>


void  free (void  *ptr);
```

표준 라이브러리 (11/15)

- **프로세스 제어 관련 함수**

```
#include <stdlib.h>

void  exit (int  exit_code);        //프로그램을 정상적인 상태로 종료
void  abort ();                     // 프로그램을 그 상태에서 정지(비정상적인 종료)

// 프로세스가 exit 함수를 호출하여 종료할 때 수행되는 함수들을 등록한다.
int  atexit (void  (*func)(void));

// exit 함수와 같지만 clean-up-action을 수행하지 않고 프로그램 종료
void  _exit(int  exit_code);
void  _Exit(int  exit_code);                              // C99
```

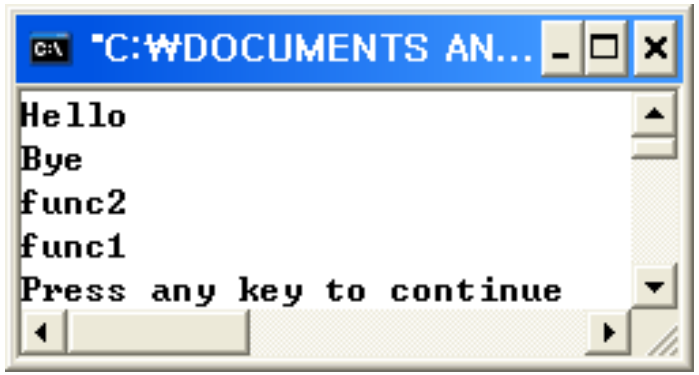
```
#include <stdlib.h>

int  system(const  char  *command);
```

호출 성공 : **0** 값을 반환
범위 오류 : **-1** 값을 반환

표준 라이브러리 (12/15)

프로그램 예제 : 프로그램 흐름 제어 관련 함수

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

void func1(void);
void func2(void);

int main(void)
{
        printf("Hello\n");
        atexit(func1);
        atexit(func2);
        printf("Bye\n");

        exit(0);          // _exit(0); 수정 시 func1과 func2는 실행되지 않고 종료한다.
}

void func1(void)
{
        printf("func1\n");
}

void func2(void)
{
        printf("func2\n");
}
```



C 표준 라이브러리 (13/15)

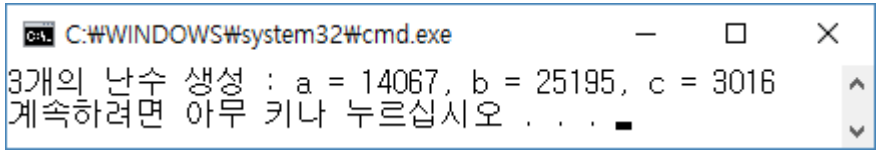
- **임의의 난수 생성 함수**

```
#include <stdlib.h>

// 0 ~ RAND_MAX 사이에서 임의의 난수를 생성하여 반환
int  rand (void);

// seed 있는 랜덤발생 함수, 초기의 seed 는 1이다.
void  srand ( unsigned  int  seed );
```

C 표준 라이브러리 (14/15)

예제 12-5 : 임의의 난수 생성

```c
#include <stdio.h>
#include <stdlib.h>   // rand, srand
#include <time.h>     // time

int main(void)
{
        int a, b, c;

        srand( (unsigned int) time(NULL) );

        a = rand();
        b = rand();
        c = rand();

        printf("3개의 난수 생성 : a = %d, b = %d, c = %d \n",  a, b, c );

        return 0;
}
```



C 표준 라이브러리 (15/15)

예제 12-6 : 임의의 난수 생성과 범위 설정

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

int main(void)
{
        srand( (unsigned int)time(NULL) );

        printf("##### 로또 복권 생성 ##### \n\n");
        for ( int i=1; i<=6; i++ )
        {
                int     temp = rand() % 45 + 1;         // 최소값과 최대값

                printf("%3d", temp );
        }

        printf("\n\n");

        return 0;
}
```

C 표준 라이브러리

수학 관련 함수 : <math.h>

수학 관련 함수 (1/14)

- **수학 계산 함수 : abs**

```
#include <math.h>

// 정수형의 절대값 을 계산하여 반환
int  abs( int  num );
long  labs( long  num );
long  long  llabs( long  long  num );                    // C99
```

```
#include <math.h>

// 부동 소수점의 절대값을 계산하여 반환
float  fabsf( float  arg );                              // C99
double  fabs( double  arg );
long  double  fabsl( long  double  arg );                // C99
```

수학 관련 함수 (2/14)

- **수학 계산 함수** : div

```
#include <math.h>

// 나눗셈 함수(몫과 나머지 계산)
div_t  div( int  x,  int  y );
ldiv_t  ldiv( long  x,  long  y );
lldiv_t  lldiv( long  long  x,  long  long  y );                    // C99
```

```
#include <math.h>

struct div_t            // div_t 구조체
{
        int         quot;
        int         rem;
};
```

수학 관련 함수 (3/14)

- **수학 계산 함수** : fmod, fmax, fmin

```
#include <math.h>

// 실수 x 를 실수 y 로 나눈 나머지를 계산하여 반환
float  fmodf( float  x,  float  y );                                  // C99
double  fmod( double  x,  double  y );
long  double  fmodl( long  double  x,  long  double  y );             // C99

// 실수 x 와 실수 y 에서 큰 값을 반환
float  fmaxf( float  x,  float  y );                                  // C99
double  fmax( double  x,  double  y );                                // C99
long  double  fmaxl( long  double  x,  long  double  y );             // C99

// 실수 x 와 실수 y 에서 작은 값을 반환
float  fminf( float  x,  float  y );                                  // C99
double  fmin( double  x,  double  y );                                // C99
long  double  fminl( long  double  x,  long  double  y );             // C99
```

수학 관련 함수 (4/14)

- **지수와 로그 함수 : log10**

```
#include <math.h>

// arg 의 자연 로그(기본 base는 10) 값을 반환
float  log10f( float  arg );                          // C99
double  log10( double  arg );
long  double  log10l( long  double  arg );            // C99
```

수학 관련 함수 (5/14)

- **지수와 로그 함수** : exp, log

```
#include <math.h>

// 기본 base 가 자연 로그(e)인 arg 제곱 값을 반환
float  expf( float  arg );                          // C99
double  exp( double  arg );
long  double  expl( long  double  arg );            // C99
```

```
#include <math.h>

// arg 의 자연 로그(기본 base 는 e) 값을 반환
float  logf( float  arg );                          // C99
double  log( double  arg );
long  double  logl( long  double  arg );            // C99
```

수학 관련 함수 (6/14)

- **지수와 로그 함수** : pow, sqrt

```
#include <math.h>

// base 의 exp 승 값을 계산하여 반환
float  powf( float  base,  float  exp );                          // C99
double  pow( double  base,  double  exp );
long  double  powl( long  double  base,  long  double  exp );     // C99
```

```
#include <math.h>

// arg 의 0 이 아닌 제곱근을 계산하여 반환
float  sqrtf( float  arg );                                       // C99
double  sqrt( double  arg );
long  double  sqrtl( long  double  arg );                         // C99
```

수학 관련 함수 (7/14)

- **삼각 함수** : sin, cos, tan

```
#include <math.h>

// arg 의 sine 값을 계산하여 반환
float  sinf( float  arg );                                  // C99
double  sin( double  arg );
long  double  sinl( long  double  arg );                    // C99

// arg 의 cosine 값을 계산하여 반환
float  cosf( float  arg );                                  // C99
double  cos( double  arg );
long  double  cosl( long  double  arg );                    // C99

// arg 의 tangent 값을 계산하여 반환
float  tanf( float  arg );                                  // C99
double  tan( double  arg );
long  double  tanl( long  double  arg );                    // C99
```

수학 관련 함수 (8/14)

- **삼각 함수** : asin, acos, atan

```
#include <math.h>

// arg 의 arc  sine 값을 계산하여 반환
float  asinf( float  arg );                                  // C99
double  asin( double  arg );
long  double  asinl( long  double  arg );                    // C99

// arg 의 arc  cosine 값을 계산하여 반환
float  acosf( float  arg );                                  // C99
double  acos( double  arg );
long  double  acosl( long  double  arg );                    // C99

// arg 의 arc  tangent 값을 계산하여 반환
float  atanf( float  arg );                                  // C99
double  atan( double  arg );
long  double  atanl( long  double  arg );                    // C99
```

수학 관련 함수 (9/14)

- **삼각 함수** : atan2

```
#include <math.h>

// y/x 의 arc  tangent 값을 계산하여 반환
float  atan2f( float  y,  float  x );                          // C99
double  atan2( double  y,  double  x );                        // C99
long  double  atan2l( long  double  y,  long  double  x );     // C99
```

수학 관련 함수 (10/14)

- **쌍곡선 함수** : sinh, cosh, tanh

```
#include <math.h>

// arg 의 쌍곡선(hyperbolic)  sine 값을 계산하여 반환
float  sinhf( float  arg );                                  // C99
double  sinh( double  arg );
long  double  sinhl( long  double  arg );                    // C99

// arg 의 쌍곡선(hyperbolic)  cosine 값을 계산하여 반환
float  coshf( float  arg );                                  // C99
double  cosh( double  arg );
long  double  coshl( long  double  arg );                    // C99

// arg 의 쌍곡선(hyperbolic)  tangent 값을 계산하여 반환
float  tanhf( float  arg );                                  // C99
double  tanh( double  arg );
long  double  tanhl( long  double  arg );                    // C99
```

수학 관련 함수 (11/14)

- **쌍곡선 함수** : asinh, acosh, atanh

```
#include <math.h>

// arg 의 쌍곡선(hyperbolic)  arc  sine 값을 계산하여 반환
float  asinhf( float  arg );                                  // C99
double  asinh( double  arg );
long  double  asinhl( long  double  arg );                    // C99

// arg 의 쌍곡선(hyperbolic)  arc  cosine 값을 계산하여 반환
float  acoshf( float  arg );                                  // C99
double  acosh( double  arg );
long  double  acoshl( long  double  arg );                    // C99

// arg 의 쌍곡선(hyperbolic)  arc  tangent 값을 계산하여 반환
float  atanhf( float  arg );                                  // C99
double  atanh( double  arg );
long  double  atanhl( long  double  arg );                    // C99
```

수학 관련 함수 (12/14)

- 가장 가까운 정수 또는 부동 소수점 연산 함수 (1/2)

```
#include <math.h>

// arg 를 초과하는 정수 중에서 가장 가까운 정수를 반환
float  ceilf( float  arg );                                  // C99
double  ceil( double  arg );
long  double  ceill( long  double  arg );                    // C99

// arg 보다 작은 정수 중에서 가장 가까운 정수를 반환
float  floorf( float  arg );                                 // C99
double  floor( double  arg );
long  double  floorl( long  double  arg );                   // C99

// arg 의 소수점 이하를 버리고 정수를 반환
float  truncf( float  arg );                                 // C99
double  trunc( double  arg );                                // C99
long  double  truncl( long  double  arg );                   // C99
```

수학 관련 함수 (13/14)

- 가장 가까운 정수 또는 부동 소수점 연산 함수 (2/2)

```
#include <math.h>

// arg 에서 가까운 정수(즉, 반올림 값)를 반환
float  roundf( float  arg );                          // C99
double  round( double  arg );                         // C99
long  double  roundl( long  double  arg );            // C99

long  lroundf( float  arg );                          // C99
long  lround( double  arg );                          // C99
long  lroundl( long  double  arg );                   // C99

long  long  llroundf( float  arg );                   // C99
long  long  llround( double  arg );                   // C99
long  long  llroundl( long  double  arg );            // C99
```

수학 관련 함수 (14/14)

- **부동 소수점 조작 함수 : modf, ldexp**

```
#include <math.h>

// 실수 x 를 정수 부분과 실수 부분으로 분리하여 한다.
// 정수부분은 iptr 에 저장되고, 실수부분은 반환
float  modff( float  x,  float  *iptr );                          // C99
double  modf( double  x,  double  *iptr );
long  double  modfl(long  double  x,  long  double  *iptr);       // C99
```

```
#include <math.h>

// arg * 2^exp 를 계산하여 반환
// arg * pow(2, exp) 의 결과와 동일하다.
float  ldexpf( float  arg,  int  exp );                           // C99
double  ldexp( double  arg,  int  exp );
long  double  ldexpl( long  double  arg,  int  xp );             // C99
```

C 표준 라이브러리

날짜 및 시간 관련 함수 : <time.h>

날짜 및 시간 관련 함수 (1/7)

- 데이터 유형 (Data Type)

```
#include <time.h>

struct tm
{
        int     tm_sec;     // 초 [0 - 59]
        int     tm_min;     // 분 [0 - 59]
        int     tm_hour;    // 시 [0 - 23]
        int     tm_mday;    // 일자 [1 - 31]
        int     tm_mon;     // 1월에서 12월 [0 – 11, 0 = January]
        int     tm_year;    // 1900년 이후부터 경과한 년도
        int     tm_wday;    // 일요일에서 토요일 [0 – 6, 0 = Sunday]
        int     tm_yday;    // 1월 1일부터 경과된 날짜 [0 - 365]
        int     tm_isdst;   // 서머 타임 설정 여부 플러그
};

typedef long time_t;        // 시간을 표현하는 산술 유형
typedef long clock_t;       // 프로세스 실행 시간 유형
```

날짜 및 시간 관련 함수 (2/7)

- **매크로 (Macro) : 변수 및 상수**

```
#include <time.h>


#define  CLOCKS_PER_SEC  1000

#define  CLK_TCK  CLOCKS_PER_SEC
```

날짜 및 시간 관련 함수 (3/7)

- **시간 조작 함수**

```
#include <time.h>

// 프로그램 시작 후의 경과한 clock tick(1초에 18.2만큼 증가)를 반환
clock_t  clock();
```

호출 성공 : **process tick count** 값을 반환
호출 실패 : **(clock_t) -1** 값을 반환

```
#include <time.h>

// 1970년 1월 1일 자정부터 경과된 현재 시간을 초 단위로 계산하여 반환
time_t  time(time_t  *time);
```

호출 성공 : 현재 시간 값을 반환
호출 실패 : **(time_t) -1** 값을 반환

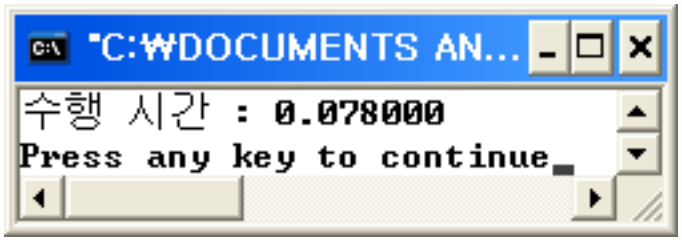
```
#include <time.h>

// time2 – time1 을 초로 계산하여 결과 값을 반환
double  difftime(time_t  time2,  time_t  time1);
```

호출 성공 : 두 시간의 차를 계산한 실수 값을 반환

날짜 및 시간 관련 함수 (4/7)

프로그램 예제 : 시간 관련 함수

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

int main(void)
{
        clock_t         start, end;
        int             i = 0;
        double          time;

        start = clock();
        while(i < 30000000)
                i++;
        end = clock();

        time = (double) (end - start) / CLK_TCK;

        printf("수행 시간 : %lf\n", time );

        return 0;
}
```



날짜 및 시간 관련 함수 (5/7)

● 형식 변환 함수 (1/2)

```
#include <time.h>

// 구조체 time_ptr 에 저장된 값을 지정된 형식의 문자열로 변환
char *asctime(const tm *time_ptr);

// time 값을 시간과 날짜 형식의 지정된 문자열로 변환
char *ctime(const time_t *time);
```

호출 성공 : 지정된 형식으로 변환된 문자열의 시작 주소를 반환
호출 실패 : **NULL** 값을 반환

❍ 문자열 형식

Www Mmm dd hh:mm:ss yyy

- **Www** : 요일 (Mon, Tue, Wed, Thu, Fri, Sat, Sun)
- **Mmm** : 월 (Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec)
- **dd** : 일 (the day of the month)
- **hh** : 분 (minutes)
- **ss** : 초 (seconds)
- **yyyy** : 년도 (years)

날짜 및 시간 관련 함수 (6/7)

- **형식 변환 함수 (2/2)**

```
#include <time.h>

// time 함수에 의해 구해진 시간을 날짜와 시간 관련 구조체 tm 에 저장(GMT 표준 시간)
tm  *gmtime(const  time_t  *time);

// time 함수에 의해 구해진 시간을 tm 구조체에 저장(지역 표준 시간)
tm  *localime(const  time_t  *time);
```

호출 성공 : 변환된 날짜와 시간 관련 구조체 **tm**의 시작 주소를 반환
호출 실패 : **NULL** 값을 반환

날짜 및 시간 관련 함수 (7/7)

프로그램 예제 : 날짜 관련 함수

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

int main(void)
{
        time_t          t;
        struct tm       *gmt, *local;

        // time 함수를 이용하여 현재 시간을 구한다.
        t = time(NULL);
        printf("현재 GMT : %d \n", t);

        // 현재 시간을 GMT와 지역 표준 시간으로 변환
        gmt = gmtime(&t);
        local = localtime(&t);

        // tm 구조체에 저장된 값을 지정된 문자열로 변환
        // 지정된 문자열 형식 : Www Mmm dd hh:mm:ss yyy
        printf("GMT  표준 시간 : %s \n", asctime(gmt) );
        printf("지역 표준 시간 : %s \n", asctime(local) );

        return 0;
}
```

참고문헌

[1] 서두옥, 이동호(감수), (열혈강의)"또 하나의 C : 프로그래밍은 셀프입니다", 프리렉, 2012.

[2] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.

[3] SAMUEL P. HARBISON Ⅲ, GUY L. STEELE, "C 프로그래밍 언어, C : A Reference Manual", 5/E, Pearson Education Korea, 2005.

[4] Behrouz A. Forouzan, Richard F. Gilberg, 김진 외 7인 공역, "구조적 프로그래밍 기법을 위한 C", 도서출판 인터비젼, 2004.

[5] Brian W. Kernighan, Dennis M. Ritchie, 김석환 외 2인 공역, "The C Programming Language", 2/E, 대영사, 2004.