Chapter 5
Pointers & Arrays
포인터 + 배열

Part 3
5.7~5.11

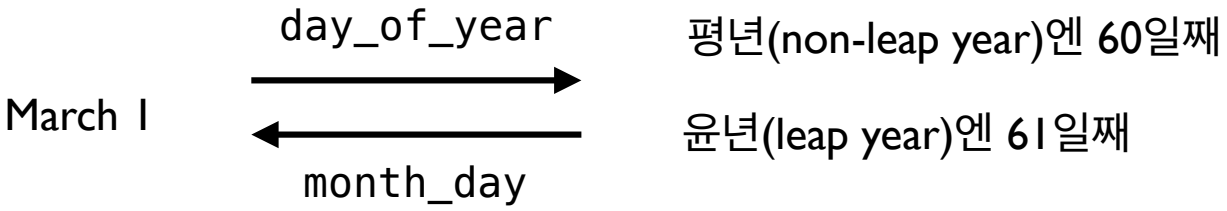
CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과
도경구

1. Pointers & Addresses
2. Pointers & Function Arguments
3. Pointers & Arrays
4. Address Arithmetic
5. Character Pointers & Functions
6. Pointer Arrays; Pointers to Pointers
7. Multi-dimensional Arrays
8. Initialization of Pointer Arrays
9. Pointers vs. Multidimensional Arrays
10. Command-line Arguments
11. Pointers to Functions
12. Complicated Declarations

Multidimensional Arrays

날짜 변환




```
static char daytab[2][13] = {
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
```

row column

```
f(int daytab[2][13]) { . . . }
f(int daytab[][13]) { . . . }
f(int (*daytab)[13]) { . . . }
```

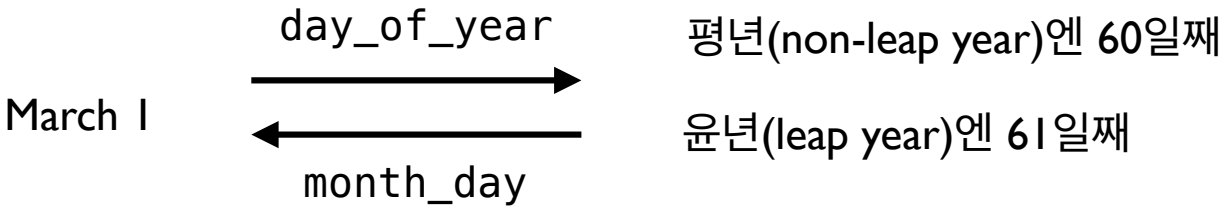
a pointer to an array of 13 integers

```
int *daytab[13]
```

an array of 13 pointers to integers

Multidimensional Arrays

날짜 변환

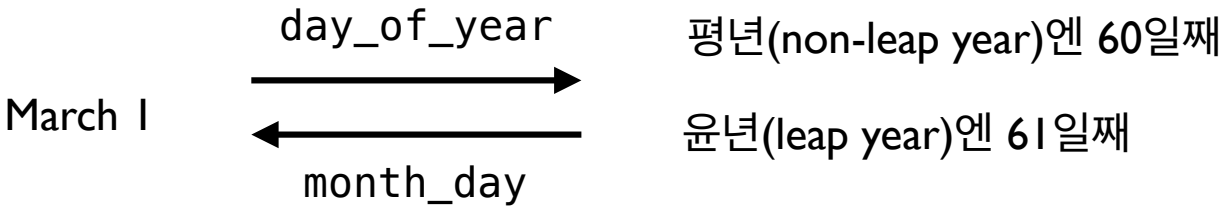



```
static char daytab[2][13] = {
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
```

```
/* day_of_year: set day of year from manth & day */
int day_of_year(int year, int month, int day) {
    int i, leap;

    leap = (year%4 == 0 && year%100 != 0) || year%400 == 0;
    for (i = 1; i < month; i++)
        day += daytab[leap][i];
    return day;
}
```

Multidimensional Arrays

날짜 변환




```
static char daytab[2][13] = {
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
```

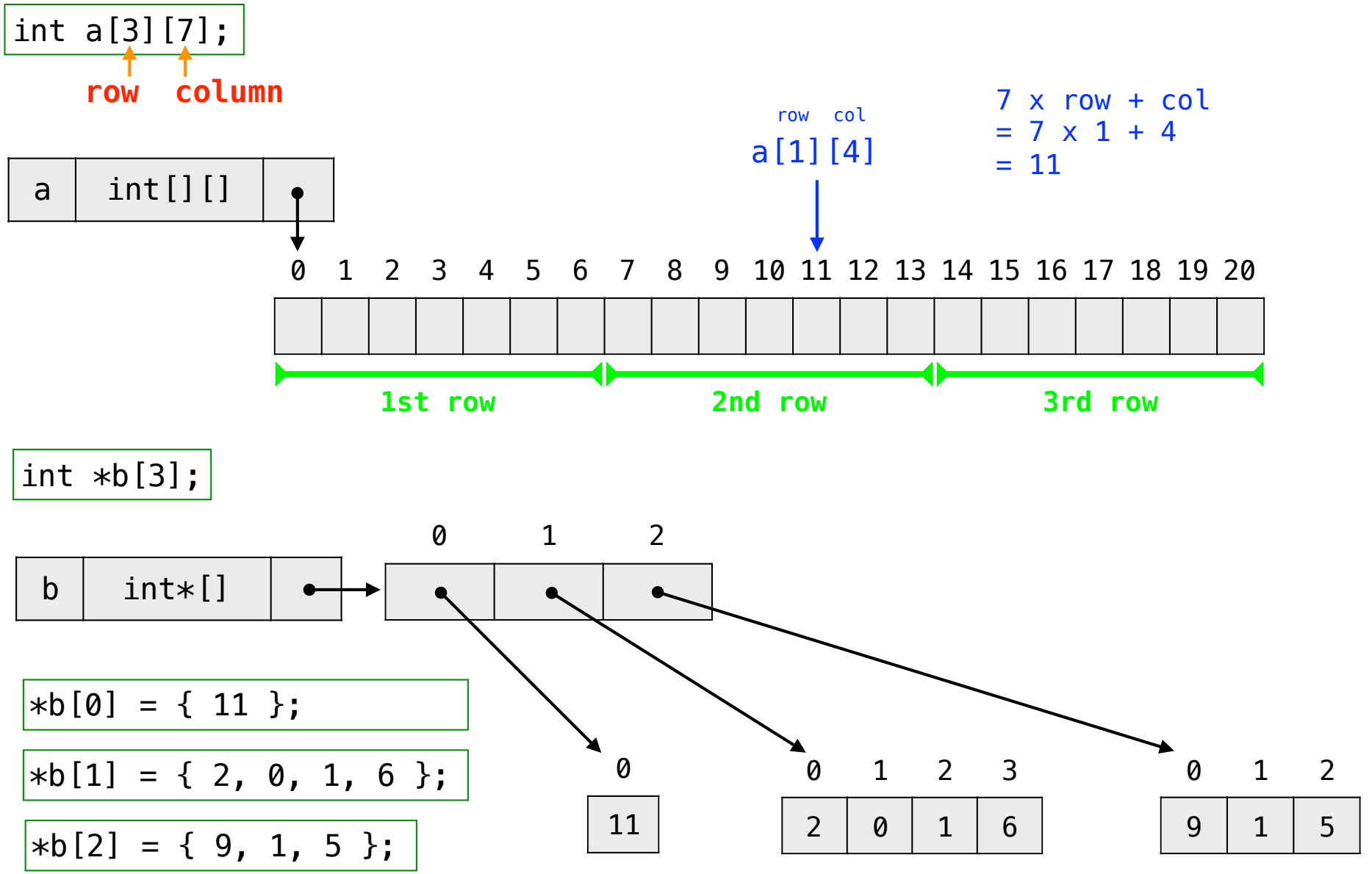
```
/* month_day: set month, day from day of year */
void month_day(int year, int yearday, int *pmonth, int *pday) {
    int i, leap;

    leap = (year%4 == 0 && year%100 != 0) || year%400 == 0;
    for (i = 1; yearday > daytab[leap][i]; i++)
        yearday -= daytab[leap][i];
    *pmonth = i;
    *pday = yearday;
}
```

month_day(2016, 100, &m, &d) => ?

Initialization of Pointer Arrays

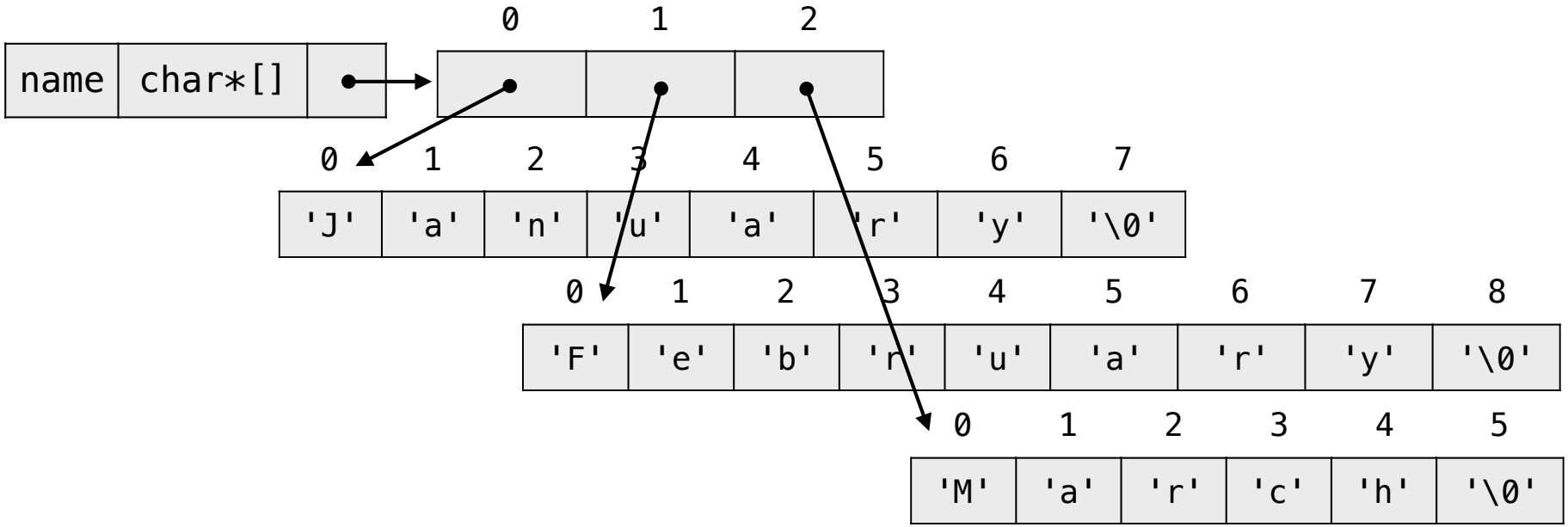
```
/* month_name: return name of n-th month */
char *month_name(int n) {
    static char *name[] = {
        "Illegal month",
        "January", "February", "March",
        "April", "May", "June",
        "July", "August", "September",
        "October", "November", "December"
    };

    return (n < 1 || n > 12) ? name[0] : name[n];
}
```

Pointers vs. Multi-dimensional Arrays

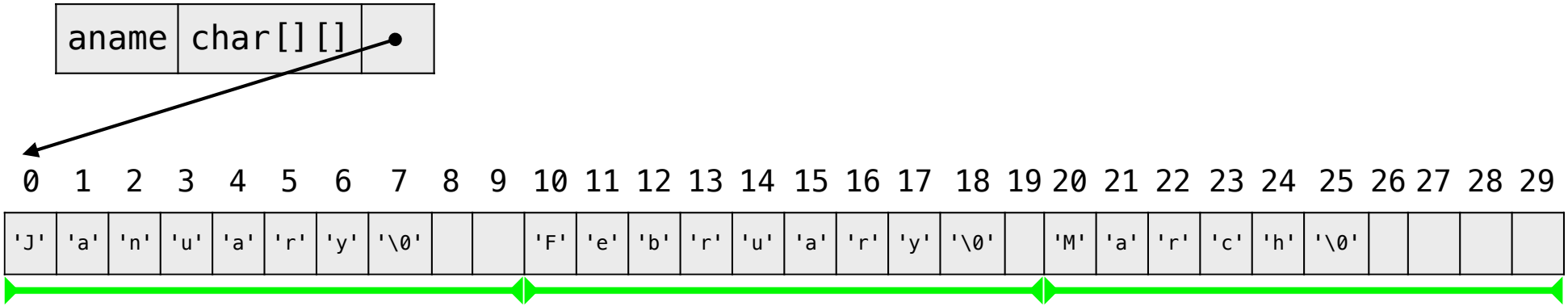


Pointers vs. Multi-dimensional Arrays

```
char *name[] = { "January", "February", "March"};
```




```
char aname[][10] = { "January", "February", "March"};
```



Command-line Arguments

실행 명령과 함께 프로그램에 전달하는 인수

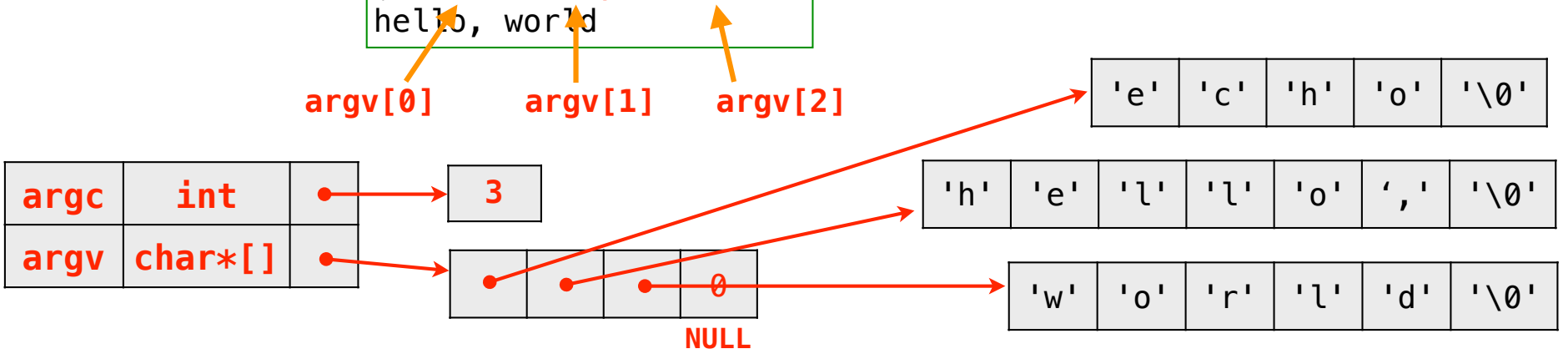
argc (argument count) 인수의 개수

argv (argument vector) 인수 문자열 배열 포인터

Array version

```
#include <stdio.h>

/* echo command-line arguments; 1st version */
int main (int argc, char *argv[]) {
    int i;

    for (i = 1; i < argc; i++)
        printf("%s%s", argv[i], (i < argc-1) ? " " : "");
    printf("\n");
    return 0;
}
```

```
$ gcc echo.c -o echo
$ echo hello, world
hello, world
```



Command-line Arguments

실행 명령과 함께 프로그램에 전달하는 인수

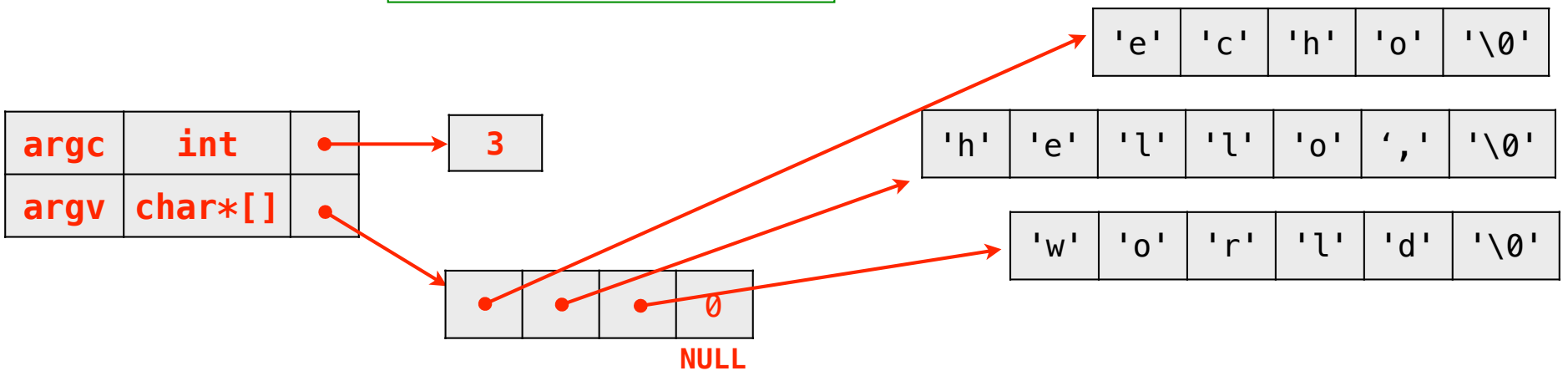
argc (argument count) 인수의 개수

argv (argument vector) 인수 문자열 배열 포인터

Pointer version

```
#include <stdio.h>

/* echo command-line arguments; 2nd version */
int main (int argc, char* argv[]) {
    while (--argc > 0)
        printf("%s%s", *++argv, (argc > 1) ? " " : "");
    printf("\n");
    return 0;
}
```

```
$ gcc echo.c -o echo
$ echo hello, world
hello, world
```



Command-line Arguments

find.c

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

#define MAXLINES 1000

int readline(char *line, int max);

/* find: print lines that match pattern from 1st arg */
int main(int argc, char *argv[]) {
    char line[MAXLINE];
    int found = 0;

    if (argc != 2)
        printf("Usage: find pattern\n");
    else
        while (readline(line, MAXLINE) > 0)
            if (strstr(line, argv[1]) != NULL) {
                printf("%s", line);
                found++;
            }
    return found;
}
```

Command-line Arguments

find+.c

option 1	–x	print all lines except those that match the pattern
option 2	–n	precede each printed line by its line number

```
$ find find.c –o find
$ find –x –n pattern
$ find –xn pattern
```

```
int main(int argc, char *argv[]) {
    char line[MAXLINE];
    long lineno = 0;
    int c, except = 0, number = 0, found = 0;

    while (--argc > 0 && (*++argv)[0] == '-')
        while ((c = *++argv[0]))
            switch (c) {
                case 'x':
                    except = 1;
                    break;
                case 'n':
                    number = 1;
                    break;
                default:
                    printf("find+: illegal option %c\n", c);
                    argc = 0;
                    found = -1;
                    break;
            }
    . . .
```

****++argv**

Command-line Arguments

find+.c

option 1	–x	print all lines except those that match the pattern
option 2	–n	precede each printed line by its line number

```
$ find find.c –o find
$ find –x –n pattern
$ find –xn pattern
```

```
    . . .
    if (argc != 1)
        printf("Usage: find –x –n pattern\n");
    else
        while (readline(line, MAXLINE) > 0) {
            lineno++;
            if ((strstr(line, *argv) != NULL) != except) {
                if (number)
                    printf("%ld:", lineno);
                printf("%s", line);
                found++;
            }
        }
    return found;
}
```

Pointers to Functions

Quicksort string & numeral

strcmp numcmp

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

#define MAXLINES 5000  /* max #lines to be sorted */

char *lineptr[MAXLINES];  /* pointers to text lines */

int readlines(char *lineptr[], int nlines);
void writelines(char *lineptr[], int nlines);

void qsort(char *lineptr[], int left, int right,
           int (*comp)(char *, char *));
int numcmp(char *, char *);

/* sort input lines */
int main(int argc, char *argv[]) {
    int nlines;       /* number of input lines read */
    int numeric = 0;  /* 1 if numeric sort */

    if (argc > 1 && strcmp(argv[1], "-n") == 0)
        numeric = 1;
    if ((nlines = readlines(lineptr, MAXLINES)) >= 0) {
        qsort(lineptr, 0, nlines-1, numeric ? numcmp : strcmp);
        writelines(lineptr, nlines);
        return 0;
    }
    else {
        printf("input too big to sort\n");
        return 1;
    }
}
```

Pointers to Functions

Quicksort string & numeral

strcmp numcmp

```
/* numcmp: compare s1 and s2 numerically */
int numcmp(char *s1, char *s2) {
    double v1, v2;

    v1 = atof(s1);
    v2 = atof(s2);
    if (v1 < v2)
        return -1;
    else if (v1 > v2)
        return 1;
    else
        return 0;
}
```

```
/* qsort: sort v[left]...v[right] into increasing order */
void qsort(char *v[], int left, int right,
           int (*comp)(char *, char *)) {
    int i, pivot;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if ((*comp)(v[i], v[left]) < 0)
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1, comp);
    qsort(v, pivot+1, right, comp);
}
```

Pointers to Functions

`int *f();`	function returning pointer to int
`int (*pf)();`	pointer to function returning int

Dynamic Memory Allocation

<stdlib.h>

calloc() **malloc()**

contiguous memory
allocation allocation

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

int main(void) {
    int *a;             /* 배열로 쓸 목적 */
    int n;              /* 배열의 크기 */
    ..                  /* n값을 어디서 가져옴 */
    a = calloc(n, sizeof(int));   /* 배열 a로 쓸 메모리 공간을 할당받음 */
                                  /* 모두 0으로 초기화 */
    ..                  /* 배열 사용 */
```

```
    a = malloc(n * sizeof(int));   /* 지정 크기만큼 메모리 공간 할당받음 */
                                   /* 초기화 하지 않음 */
```

```
    free(ptr);   /* ptr이 가리키는 메모리 공간 해제 */
```