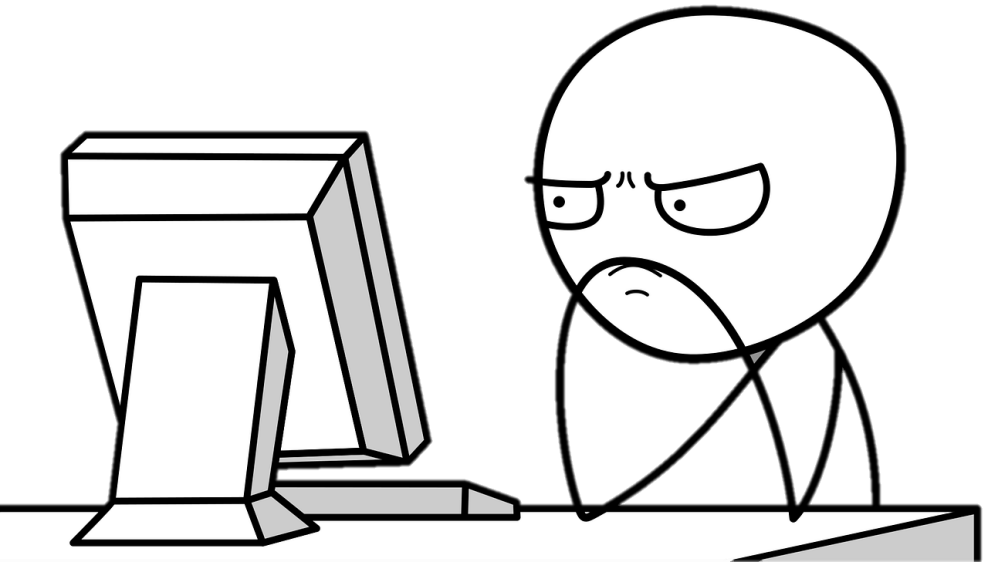

컴퓨터 프로그래밍
Computer Programming

# 11
# 문자와 문자열

# 목차

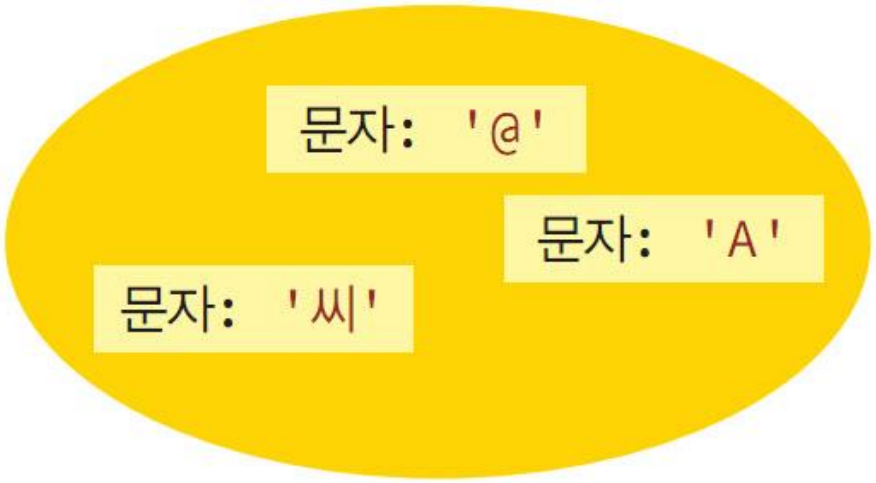
# 1. 문자와 문자열



# 문자

- 영어의 알파벳이나 한글의 한 글자를 작은 따옴표로 둘러싸서 'A'와 같이 표기
- C 언어에서 저장공간 크기 1바이트인 자료형 char로 지원
- 작은 따옴표에 의해 표기된 문자를 문자 상수

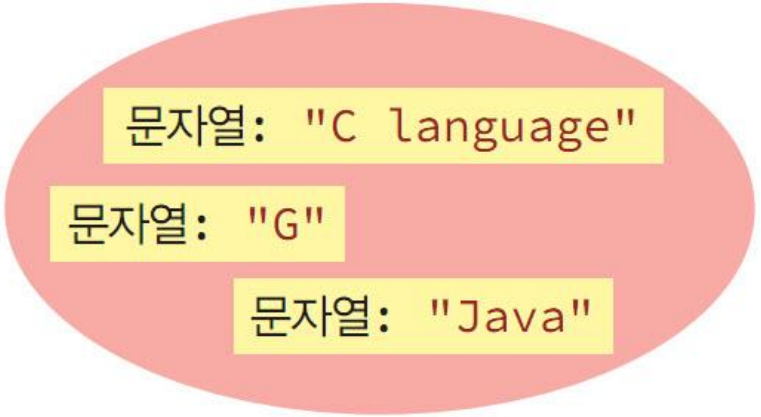
```
char ch ='A';
```



# 문자열string

- 문자의 모임인 일련의 문자
  - 일련의 문자 앞 뒤로 큰 따옴표로 둘러싸서 "java"로 표기
- 큰 따옴표에 의해 표기된 문자열을 문자열 상수
  - "A"처럼 문자 하나도 큰 따옴표로 둘러싸면 문자열 상수
  - 'ABC'처럼 작은 따옴표로 둘러싸도 문자가 될 수 없으며 오류가 발생

```
char c[] = "C language";
```



# char형 변수에 문자를 저장

- 문자열을 저장하기 위한 자료형을 따로 제공하지 않음
- 문자 배열을 선언하여 각각의 원소에 문자를 저장
- 문자열의 마지막을 의미하는 NULL 문자 '\0'가 마지막에 저장
- 문자열이 저장되는 배열크기
  - 반드시 저장될 문자 수보다 1이 커야 널(NULL) 문자를 문자열의 마지막으로 인식
  - 문자열의 마지막에 널(NULL) 문자가 없다면 출력과 같은 문자열 처리에 문제가 발생
- 배열 csharp의 크기를 3으로 선언한 후 배열 csharp에 문자열 "C#"을 저장
  - 마지막 원소인 csharp[2]에 '\0'을 저장

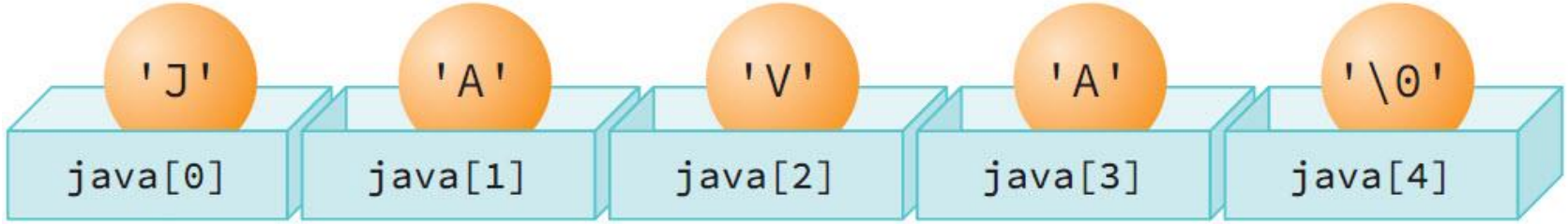
```
char ch ='A';

char csharp[3];
csharp[0] = 'C'; csharp[1] = '#'; csharp[2] = '\o';
```

# 배열 선언 시 초기화 방법

- 중괄호를 사용
- 문자 하나 하나를 쉼표로 구분하여 입력하고 마지막 문자로 널(NULL)인 '\0'을 삽입

```
//문자 하나 하나 저장 시 마지막에 '\0' 문자 저장
char java[] = {'J','A','V','A', '\0'};
```

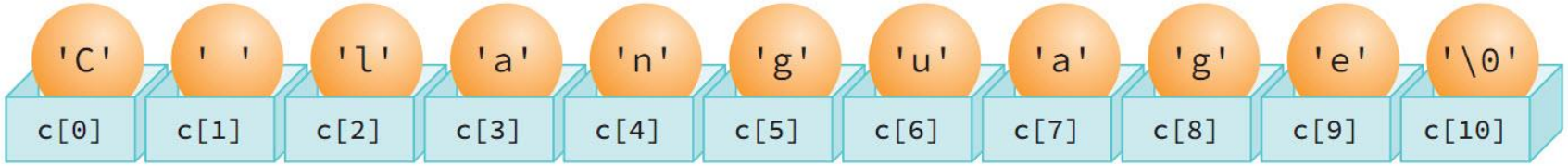
마지막에 '\0'을 빼면, 대입 시에는 문제가 없으나 출력 등에서 문제가 발생한다.



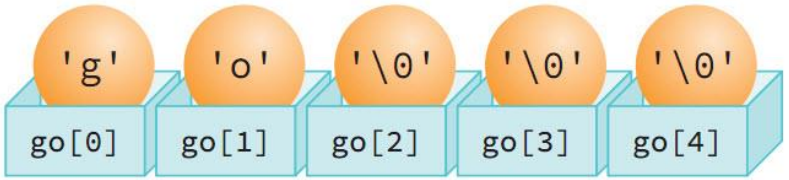
# 배열 선언 시 저장할 큰 따옴표를 사용해 문자열 상수를 바로 대입

- 배열 초기화 시 배열크기는 지정하지 않는 것이 더 편리
  - 지정한다면 마지막 문자인 '\0'을 고려해 실제 문자 수보다 1이 더 크게 배열크기를 지정
  - 지정한 배열크기가 (문자수+1)보다 크면 나머지 부분은 모두 '\0' 문자로 채워짐
- 만일 배열크기가 작으면
  - 문자열 상수가 아닌 단순한 문자 배열이 되므로 문자열 출력 등에서 문제가 발생

```
char c[] = "C language";        //크기를 생략하는 것이 간편
char c[11] = "C language";      //크기 지정 시 (문자수+1)
```




```
char go[5] = "go";              //크기가 (문자+1)보다 크면 나머지는 모두 '\0'로 채워짐
```



# Source Code #01: chararray.c

```c
// file: chararray.c
#include <stdio.h>

int main(void)
{
    //문자 선언과 출력
    char ch = 'A';
    printf("%c %d\n", ch, ch);

    //문자열 선언 방법1
    char java[] = { 'J', 'A', 'V', 'A', '\0' };
    printf("%s\n", java);
    //문자열 선언 방법2
    char c[] = "C language";    //크기를 생략하는 것이 간편
    printf("%s\n", c);
    //문자열 선언 방법3
    char csharp[5] = "C#";
    printf("%s\n", csharp);

    //문자 배열에서 문자 출력
    printf("%c%c\n", csharp[0], csharp[1]);

    return 0;
}
```

```
A 65
JAVA
C language
C#
C#
```

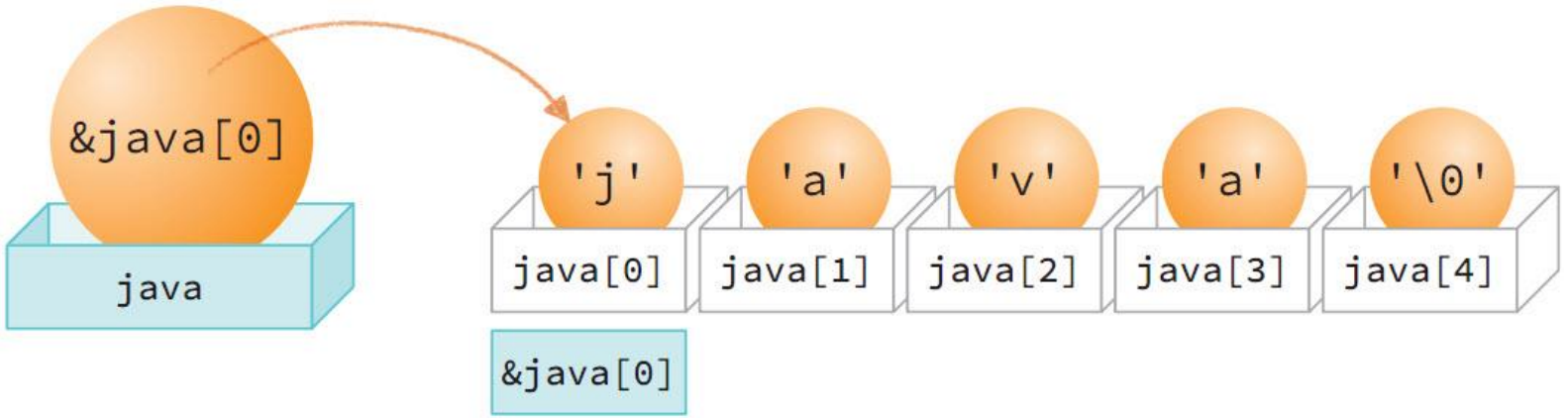
- 함수 printf()를 사용한 문자와 문자열 출력
- 함수 printf()
  - 형식제어문자 %c로 문자를 출력
  - 배열이름 또는 문자 포인터를 사용하여 형식제어문자 %s로 문자열을 출력
- 함수 puts(csharp)
  - 한줄에 문자열을 출력한 후 다음 줄에서 출력을 준비
- 함수 printf(c)
  - 배열이름을 인자로 사용해도 문자열 출력

# 문자열을 처리하는 다른 방법

- 문자열 상수를 문자 포인터에 저장하는 방식
- 문자 포인터 변수에 문자열 상수를 저장
- 문자열 출력도 함수 printf()에서 포인터 변수와 형식제어문자 %s
- 문자 포인터에 의한 선언으로는 문자 하나 하나의 수정은 불가능

```
int i = 0;
char *java = "java";
printf("%s ", java);
```

```
while (java[i] != '\0')
   printf("%c", java[i++]);
printf("\n");
```



# Source Code #02: charpointer.c

```c
// file: charpointer.c
#include <stdio.h>

int main(void)
{
    char *java = "java";
    printf("%s ", java);

    //문자 포인터가 가리키는 문자 이후를 하나 하나 출력
    int i = 0;
    while (java[i]) //while (java[i] != '\0')
        printf("%c", java[i++]);
    printf(" ");

    i = 0;
    while (*(java + i) != '\0') //java[i]는 *(java + i)와 같음
        printf("%c", *(java + i++));
    printf("\n");

    //수정 불가능, 실행오류 발생
    java[0] = 'J';

    return 0;
}
```

- 변수의 값과 주소값의 대입과 활용
- 문자열을 구성하는 하나 하나의 문자를 배열형식으로 직접 참조하여 출력
- 변수 java를 사용하여 문자를 수정하거나 수정될 수 있는 함수의 인자로 사용
  - 실행 오류가 발생
  - 변수 java가 가리키는 문자열은 상수이므로 수정은 불가능
- 출력할 문자열의 끝을 '₩0' 문자로 검사하면 편리
- 반복문을 이용하여 문자가 '₩0'이 아니면 문자를 출력

# Source Code #03: string.c

```c
// file: string.c
#include <stdio.h>

int main(void)
{
    char c[] = "C C++ Java";
    printf("%s\n", c);
    c[5] = '\0'; //널 문자에 의해 문자열 분리
    printf("%s\n%s\n", c, (c + 6));

    //문자 배열의 각 원소를 하나 하나 출력하는 방법
    c[5] = ' '; //널 문자를 빈 문자로 바꾸어 문자열 복원
    char *p = c;
    while (*p) //(*p != '\0')도 가능
        printf("%c", *p++);
    printf("\n");

    return 0;
}
```

```
C C++ Java
C C++
Java
C C++ Java
```

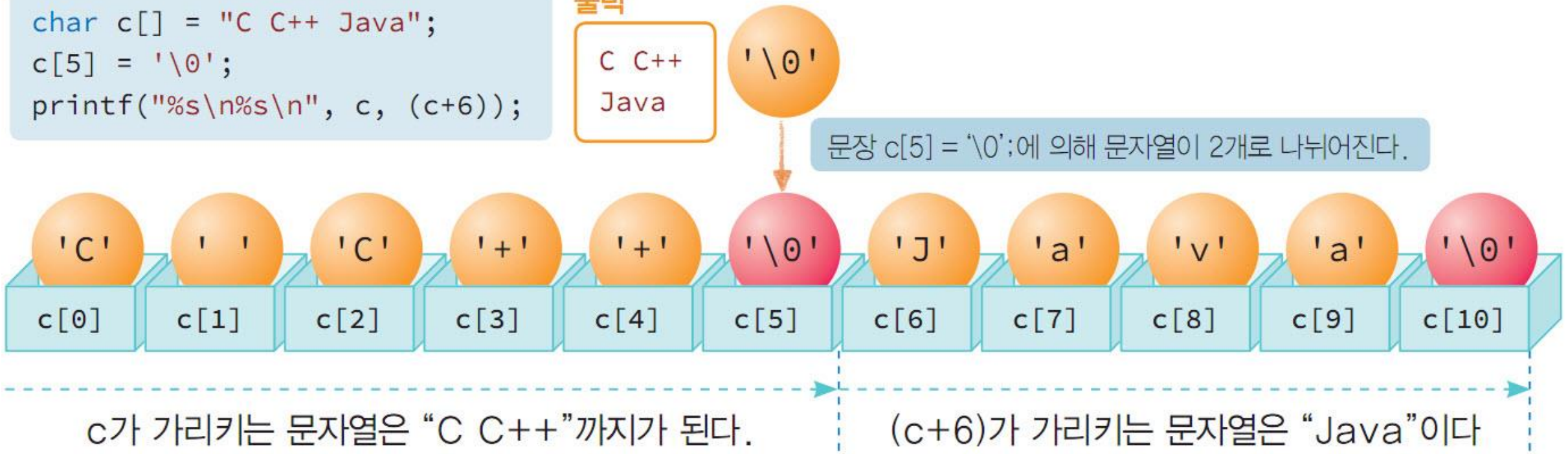
- 문자배열로 문자열을 처리
- 함수 printf()
  - %s는 문자 포인터가 가리키는 위치에서 NULL 문자까지를 하나의 문자열로 인식
- 배열 c[]
  - 처음에 문자열 "C C++ Java"가 저장되고 마지막에 NULL 문자가 저장
- 만일 배열 c에 문장 c[5] = '₩0';을 실행하고 c를 출력하면 무엇이 출력될까?
  - c[5]에 저장된 '₩0' 문자에 의해 c가 가리키는 문자열은 "C C++"까지
  - 즉 문자열은 시작문자부터 '₩0' 문자가 나올 때까지 하나의 문자열로 처리
- (c+6)로 문자열을 출력
  - "Java" 출력

# '\0' 문자에 의한 문자열 분리

```
char c[] = "C C++ Java";
c[5] = '\0';
printf("%s\n%s\n", c, (c+6));
```

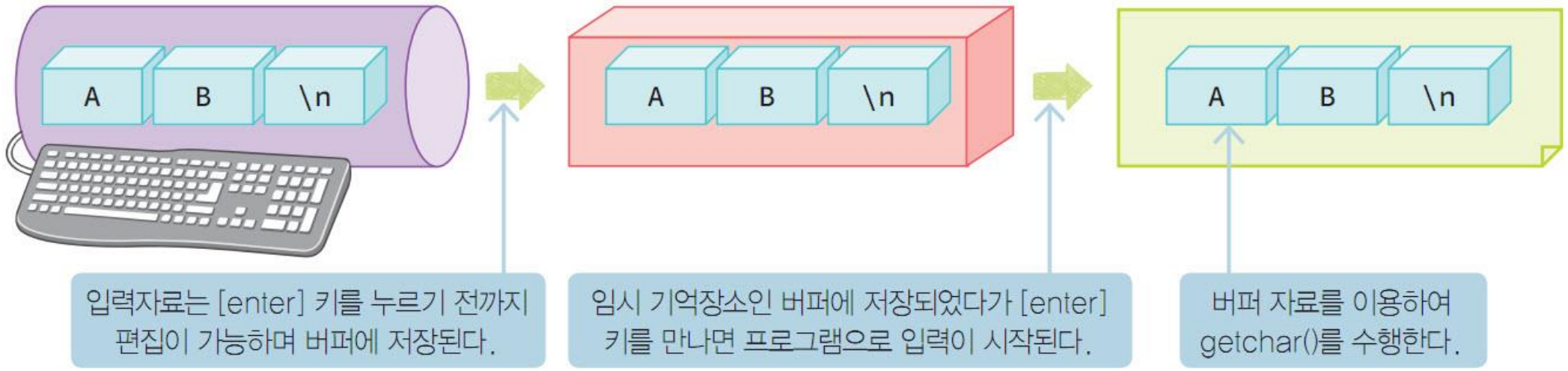
출력

```
C C++
Java
```



# 함수 getchar()

- 문자의 입력에 사용
- 라인 버퍼링line buffering 방식을 사용
  - 문자 하나를 입력해도 반응을 보이지 않다가
  - [enter] 키를 누르면 그제서야 이전에 입력한 문자마다 입력이 실행
  - 입력한 문자는 임시 저장소인 버퍼(buffer)에 저장되었다가
  - [enter] 키를 만나면 함수는 버퍼에서 문자를 읽기 시작
- 즉각적interactive인 입력을 요구하는 시스템에서는 사용이 불가능



# 버퍼를 사용하지 않고 문자를 입력하는 함수 getche()

- getche()는 버퍼를 사용하지 않으므로 문자 하나를 입력하면 바로 함수 getche()가 실행
- 함수 getche()에서 입력된 문자는 바로 모니터에 표시
- 함수를 이용하려면 헤더파일 conio.h를 삽입
- 입력 문자가 'q'가 아니면 함수 putchar()에 의하여 문자가 바로 출력
  - 함수 getche()에 의하여 입력된 문자도 보이고 바로 putchar()에 의하여 출력
- 입력문자가 "inputq"
  - 화면에는 "iinnppuuttq" 표시
  - 화면에 보이는 행이 표준입력과 표준출력이 번갈아 가면서 나오게 되므로 한 문자가 두 번씩 표시

# 함수 getche()

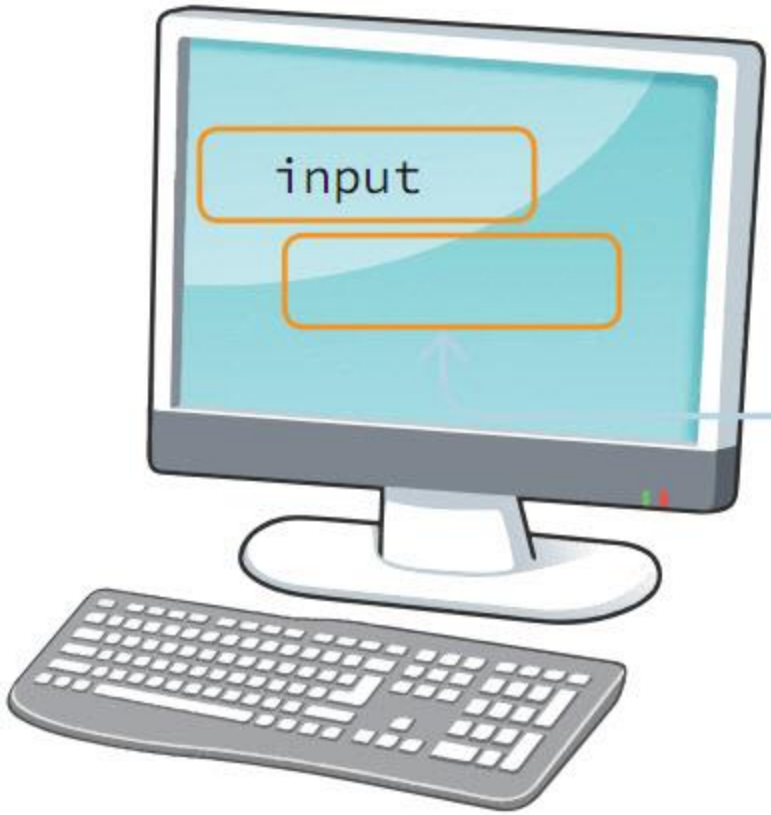

```
char ch;
while ( (ch = getche()) != 'q')
    putchar(ch);
```

입력으로 문자 q 이후를 입력할 수 없으며
한번 입력된 문자는 수정이 될 수 없다.

# 함수 getch()

- 문자 입력을 위한 함수 getch()
  - 입력한 문자가 화면에 보이지 않는 특성
  - 입력된 문자를 출력함수로 따로 출력하지 않으면 입력 문자가 화면에 보이지(echo) 않음
  - 버퍼를 사용하지 않는 문자 입력 함수
  - conio.h를 삽입
- 위 소스에서 문자 "inputq"를 입력으로 실행
  - "input"이 출력
  - 함수 putch(ch): 인자를 출력하는 함수

# 함수 getch()




```
char ch;
while ( (ch = getche()) != 'q')
    putchar(ch);
```

마지막에 입력한 문자 q 이전까지 문자 하나씩 출력되며, 입력한 문자는 수정할 수 없다. 마지막에 입력한 문자 'q'는 함수 putch()를 실행하지 않으므로 출력되지 않는다.

# 함수 getch()

| 함수 | scanf("%c", &ch) | getchar() | getche()<br>_getche() | getch()<br>_getch() |
|---|---|---|---|---|
| 헤더파일 | stdio.h | | conio.h | |
| 버퍼 이용 | 버퍼 이용함 | | 버퍼 이용 안함 | |
| 반응 | [enter] 키를 눌러야 작동 | | 문자 입력마다 반응 | |
| 입력 문자의 표시(echo) | 누르면 바로 표시 | | 누르면 바로 표시 | 표시 안됨 |
| 입력문자 수정 | 가능 | | 불가능 | |

# Source Code #04: getche.c

```c
// file: getche.c
#include <stdio.h>
#include <conio.h>

int main(void)
{
    char ch;

    printf("문자를 계속 입력하고 Enter를 누르면 >>\n");
    while ((ch = getchar()) != 'q')
        putchar(ch);

    printf("\n문자를 누를 때마다 두 번 출력 >>\n");
    while ((ch = _getche()) != 'q')
        putchar(ch);

    printf("\n문자를 누르면 한 번 출력 >>\n");
    while ((ch = _getch()) != 'q')
        _putch(ch);
    printf("\n");

    return 0;
}
```

- 세 개의 while 문의 입력으로 각각 "getchar()q", "getche()q", "getch()q"를 입력

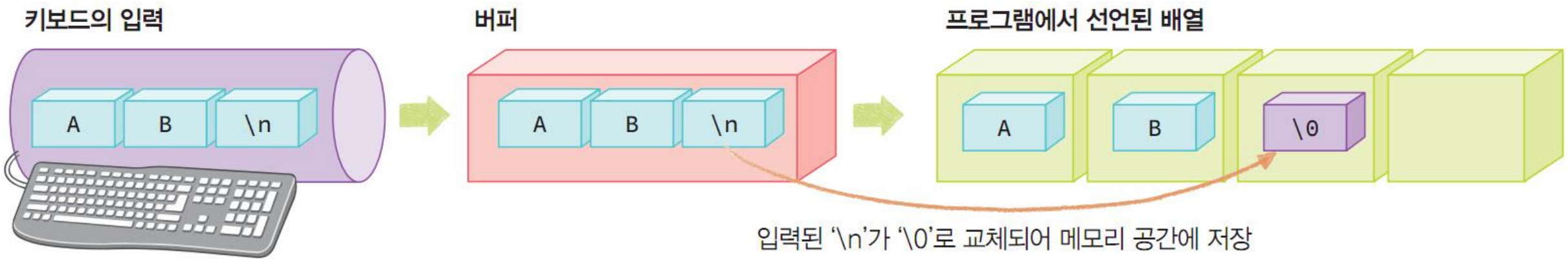
# Source Code #05: string.c

```c
// file: stringput.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    char name[20], dept[30]; //char *name, *dept; 실행 오류 발생

    printf("%s", "학과 입력 >> ");
    scanf("%s", dept);
    printf("%s", "이름 입력 >> ");
    scanf("%s", name);
    printf("출력: %10s %10s\n", dept, name);

    return 0;
}
```

- 함수 scanf()는 공백으로 구분되는 하나의 문자열을 입력
  - 함수 scanf("%s", str)에서 형식제어문자 %s를 사용
  - 문자열 입력은 충분한 공간의 문자배열이 있어야 가능
  - 단순히 문자 포인터로는 문자열 저장이 불가능
- 가장 먼저 입력 받은 문자열이 저장될 충분한 공간인 문자 배열 str을 선언
- 함수 printf("%s", str)에서 %s를 사용하여 문자열을 출력
- 이름과 성을 분리하여 입력한다면 성만 name[]에 저장
- 함수 printf()에서 %10s는 폭이 10, 우측정렬로 문자열을 출력

# 함수 gets(): 한 행의 문자열 입력

- 헤더파일 stdio.h 를 삽입
- 함수 gets()는 [enter] 키를 누를 때까지 한 행을 버퍼에 저장 한 후 입력처리
  - 마지막에 입력된 '\n'가 '\0'로 교체되어 인자인 배열에 저장
  - 한 행을 하나의 문자열로 간주하고 프로그래밍할 수 있도록



# 문자열 입출력 함수

## 문자열 입출력 함수: 헤더파일 stdio.h 삽입

```
char * gets(char * buffer);
```

- 함수 gets()는 문자열을 입력 받아 buffer에 저장하고 입력 받은 첫 문자의 주소값을 반환한다.
- 함수 gets()는 표준입력으로 [enter] 키를 누를 때까지 공백을 포함한 한 행의 모든 문자열을 입력 받는다.
- 입력된 문자열에서 마지막 [enter] 키를 '\0' 문자로 대체하여 저장한다.

```
char * gets_s(char * buffer, size_t sizebuffer);
```

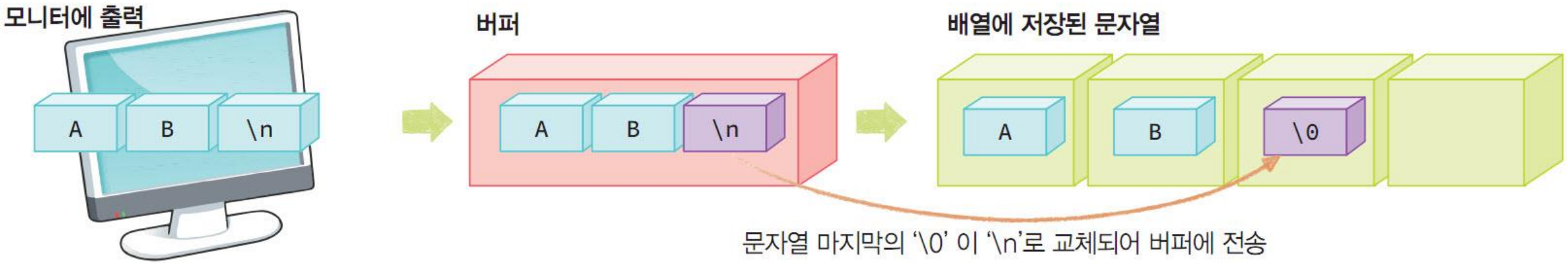
- 두 번째 인자인 sizebuffer는 정수형으로 buffer의 크기를 입력한다.
- Visual C++에서는 앞으로 gets() 대신 함수 gets_s()의 사용을 권장한다.

```
int puts(const char * str);
```

- 인자인 문자열 str에서 마지막 '\0' 문자를 개행 문자인 '\n'로 대체하여 출력한다.
- 함수 puts()는 일반적으로 0인 정수를 반환하는데, 오류가 발생하면 EOF를 반환한다.

# 함수 puts(): 한 행에 문자열을 출력

- 헤더파일 stdio.h 를 삽입
- 오류가 발생하면 EOF를 반환
  - 기호 상수 EOF(End Of File)
  - 파일의 끝이라는 의미로 stdio.h 헤더파일에 정수 -1로 정의
    - #define EOF (-1)
- 함수 gets()와 반대로 문자열의 마지막에 저장된 '\0'를 '\n'로 교체하여 버퍼에 전송
- 버퍼의 내용이 모니터에 출력되면 문자열이 한 행에 출력



# Source Code #06: gets.c

```c
// file: gets.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    char line[101]; //char *line 으로는 오류발생

    printf("입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시요.\n");
    while (gets(line))
        puts(line);
    printf("\n");

    while (gets_s(line, 101))
        puts(line);
    printf("\n");

    return 0;
}
```

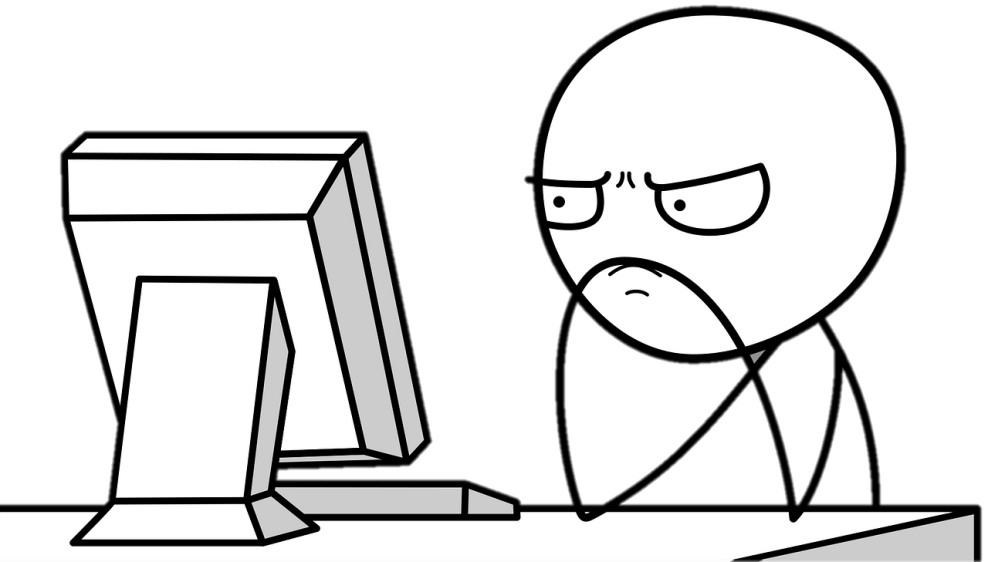
- 함수 gets()와 gets_s()를 사용하여 여러 줄을 입력 받아 출력
- while문을 사용하면 연속된 여러 행을 입력 받아 바로 행 별로 출력
- 다음 반복을 종료하려면 새로운 행 처음에 (ctrl + Z)를 입력
- 함수 printf()와 scanf()
  - 다양한 입출력에 적합
- 함수 puts()와 gets()
  - 처리 속도가 빠르다는 장점

# Lab #01: 한 행을 표준입력으로 받아 문자 하나 하나를 그대로 출력

- 함수 gets()를 사용하여 한 행의 표준입력을 받아 배열 s에 저장한 후,
  - 문자 포인터를 사용해서 이배열 s에서 문자 하나 하나를 이동하면서 출력
  - char 변수 p를 선언하면서 배열 s의 첫 원소를 가리키도록 저장
  - 포인터 변수 p는 주소값이며 *p는 p가 가리키는 곳의 문자
- 결과
  - int main(void)
  - int main(void)

```c
// lineprint.c:
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char s[100];
    //문자배열 s에 표준입력한 한 행을 저장
    gets(s);

    //문자배열에 저장된 한 행을 출력
    char *p = ;
    while ( )
        printf("%c", );
    printf("\n");

    return 0;
}
```

# 2. 문자열 관련 함수



# 다양한 문자열 라이브러리 함수

- 헤더파일 string.h에 함수원형으로 선언된 라이브러리 함수로 제공
  - 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리
  - 문자의 배열 관련 함수
  - 자료형 size_t
    - 비부호 정수형(unsigned int type)
  - 자료형 void *
    - 아직 정해지지 않은 다양한 포인터를 의미

| 함수 | scanf("%c", &ch) | getchar() | getche()<br>_getche() | getch()<br>_getch() |
|---|---|---|---|---|
| 헤더파일 | stdio.h | | conio.h | |
| 버퍼 이용 | 버퍼 이용함 | | 버퍼 이용 안함 | |
| 반응 | [enter] 키를 눌러야 작동 | | 문자 입력마다 반응 | |
| 입력 문자의 표시(echo) | 누르면 바로 표시 | | 누르면 바로 표시 | 표시 안됨 |
| 입력문자 수정 | 가능 | | 불가능 | |

# Source Code #07: memfun.c

```c
// file: strcomp.c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char src[50] = "https://www.visualstudio.com";
    char dst[50];

    printf("문자배열 src = %s\n", src);
    printf("문자열크기 strlen(src) = %d\n", strlen(src));
    memcpy(dst, src, strlen(src) + 1);
    printf("문자배열 dst = %s\n", dst);
    memcpy(src, "안녕하세요!", strlen("안녕하세요!") + 1);
    printf("문자배열 src = %s\n", src);

    char ch = ':';
    char *ret;
    ret = memchr(dst, ch, strlen(dst));
    printf("문자 %c 뒤에는 문자열 %s  이 있다.\n", ch, ret);

    return 0;
}
```

- 문자열의 길이를 반환하는 함수 strlen()
- 문자배열의 복사를 위한 함수 memcpy()
- 문자배열에서 문자 이후의 문자열을 찾는 함수 memchr()을 알아보는 예제

```
문자배열 src = https://www.visualstudio.com
문자열크기 strlen(src) = 28
문자배열 dst = https://www.visualstudio.com
문자배열 src = 안녕하세요!
문자 : 뒤에는 문자열 ://www.visualstudio.com  이 있다.
```

# 함수 strcmp()

- 문자열 관련 함수는 대부분 strxxx()로 명명
  - 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리
  - 헤더파일 string.h에 함수원형으로 선언된 라이브러리 함수로 제공

문자열 비교 함수: 헤더파일 string.h 삽입

```
int strcmp(const char * s1, const char * s2);
```

두 인자인 문자열에서 같은 위치의 문자를 앞에서부터 다를 때까지 비교하여 같으면 0을 반환하고, 앞이 크면 양수를, 뒤가 크면 음수를 반환한다.

```
int strncmp(const char * s1, const char * s2, size_t maxn);
```

두 인자 문자열을 같은 위치의 문자를 앞에서부터 다를 때까지 비교하나 최대 n까지만 비교하여 같으면 0을 반환하고, 앞이 크면 양수를, 뒤가 크면 음수를 반환한다.

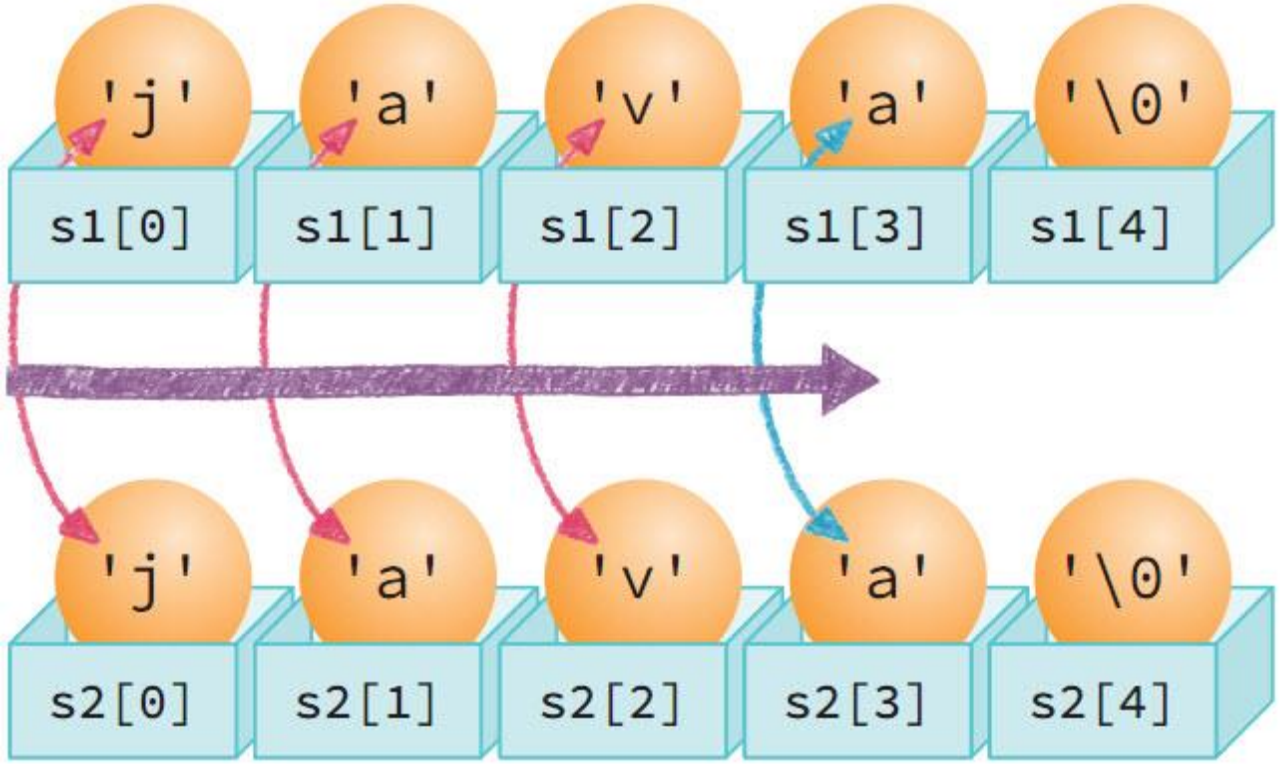
# Source Code #08: strcmp.c

```c
// file: strcomp.c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *s1 = "java";
    char *s2 = "java";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));

    s1 = "java";
    s2 = "jav";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
    s1 = "jav";
    s2 = "java";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
    printf("strncmp(%s, %s, %d) = %d\n", s1, s2, 3, strncmp(s1, s2, 3));

    return 0;
}
```

```
strcmp(java, java) = 0
strcmp(java, jav) = 1
strcmp(jav, java) = -1
strncmp(jav, java, 3) = 0
```

- 문자열 비교 함수
- 함수 strcmp()
  - 인자인 두 문자열을 사전(lexicographically) 상의 순서로 비교하는 함수
- 함수 strncmp()
  - 두 문자를 비교할 문자의 최대 수를 지정하는 함수
  - 비교 기준은 아스키 코드값
  - 두 문자가 같다면 계속 다음 문자를 비교하여 문자가 다를 때까지 계속 비교
  - 비교 앞 문자가 작으면 음수, 뒤 문자가 작으면 양수, 같으면 0을 반환

# 함수 strcmp()




- strcmp("a", "ab"): 음수
- strcmp("ab", "a"): 양수
- strcmp("ab", "ab"): 0

- strcmp("java", "javA"): 양수
  문자 a가 A보다 크므로 양수 반환

- strncmp("java", "javA", 3): 0
  인자 3인 문자 셋까지 비교하여 같으므로 0

# 함수 strcpy()

- 함수 strcpy()와 strncpy()
  - 문자열을 복사하는 함수
  - 함수 strcpy()는 앞 인자 문자열 dest에 뒤 인자 문자열 source를 복사
  - 첫 번째 인자인 dest는 복사 결과가 저장될 수 있도록 충분한 공간을 확보
  - 함수 strncpy()는 복사되는 최대 문자 수를 마지막 인자 maxn으로 지정하는 함수

# 함수 strcpy()

문자열 복사 함수

```
char * strcpy(char * dest, const char * source);
```

- 앞 문자열 dest에 처음에 뒤 문자열 null 문자를 포함한 source 를 복사하여 그 복사된 문자열을 반환한다.
- 앞 문자열은 수정되지만 뒤 문자열은 수정될 수 없다.

```
char * strncpy(char * dest, const char * source, size_t maxn);
```

- 앞 문자열 dest에 처음에 뒤 문자열 source에서 n개 문자를 복사하여 그 복사된 문자열을 반환한다.
- 만일 지정된 maxn이 source의 길이보다 길면 나머지는 모두 널 문자가 복사된다. 앞 문자열은 수정되지만 뒤 문자열은 수정될 수 없다.

```
errno_t strcpy_s(char * dest, size_t sizedest, const char * source);
errno_t strncpy_s(char * dest, size_t sizedest, const char * source, size_t maxn);
```

- 두 번째 인자인 sizedest는 정수형으로 dset의 크기를 입력한다.
- 반환형 errno_t는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수strcpy_s()와 strncpy_s()의 사용을 권장한다.

# Source Code #09: strcpy.c

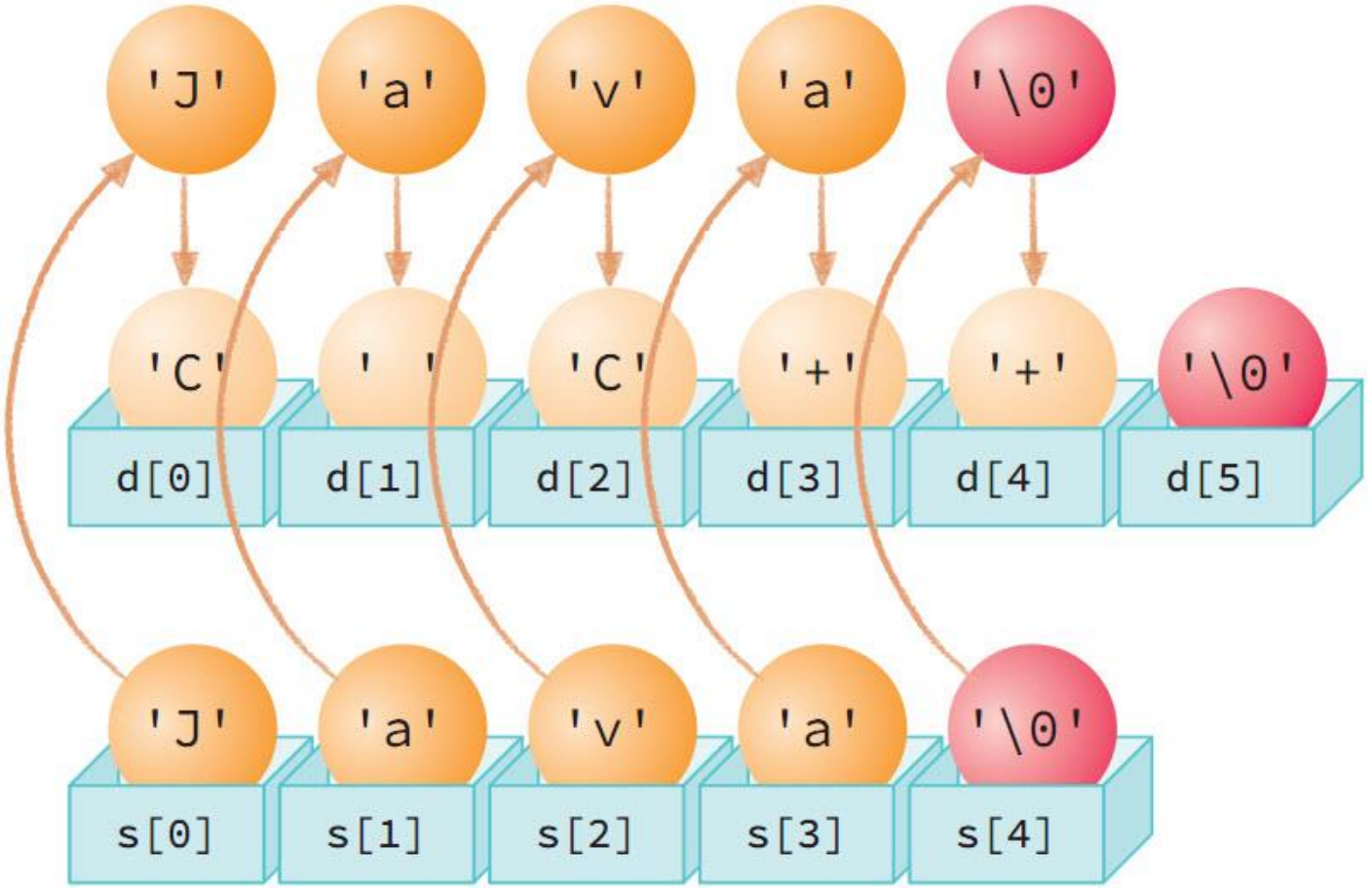
```c
// file: strcpy.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
    char dest[80] = "Java";
    char source[80] = "C is a language.";

    printf("%s\n", strcpy(dest, source));
    //printf("%d\n", strcpy_s(dest, 80, source));
    //printf("%s\n", dest);
    printf("%s\n", strncpy(dest, "C#", 2));

    printf("%s\n", strncpy(dest, "C#", 3));
    //printf("%d\n", strncpy_s(dest, 80, "C#", 3));
    //printf("%s\n", dest);

    return 0;
}
```

- 문자열 복사 사용

```
C is a language.
C#is a language.
C#
```

# 함수 strcpy()

- 항상 문자열은 마지막 NULL 문자까지 포함하므로 다음 부분 소스에서 문자배열 d에는 NULL 문자까지 복사




결과는 d에도 "java"가 저장된다.

```
char d[] = "C C++";
char s[] = "Java";
strcpy(d, s);
```

# 함수 strcpy()

문자열 연결 함수

```
char * strcat(char * dest, const char * source);
```

- 앞 문자열 dest에 뒤 문자열 source를 연결(concatenate)해 저장하며, 이 연결된 문자열을 반환하고 뒤 문자열은 수정될 수 없다.

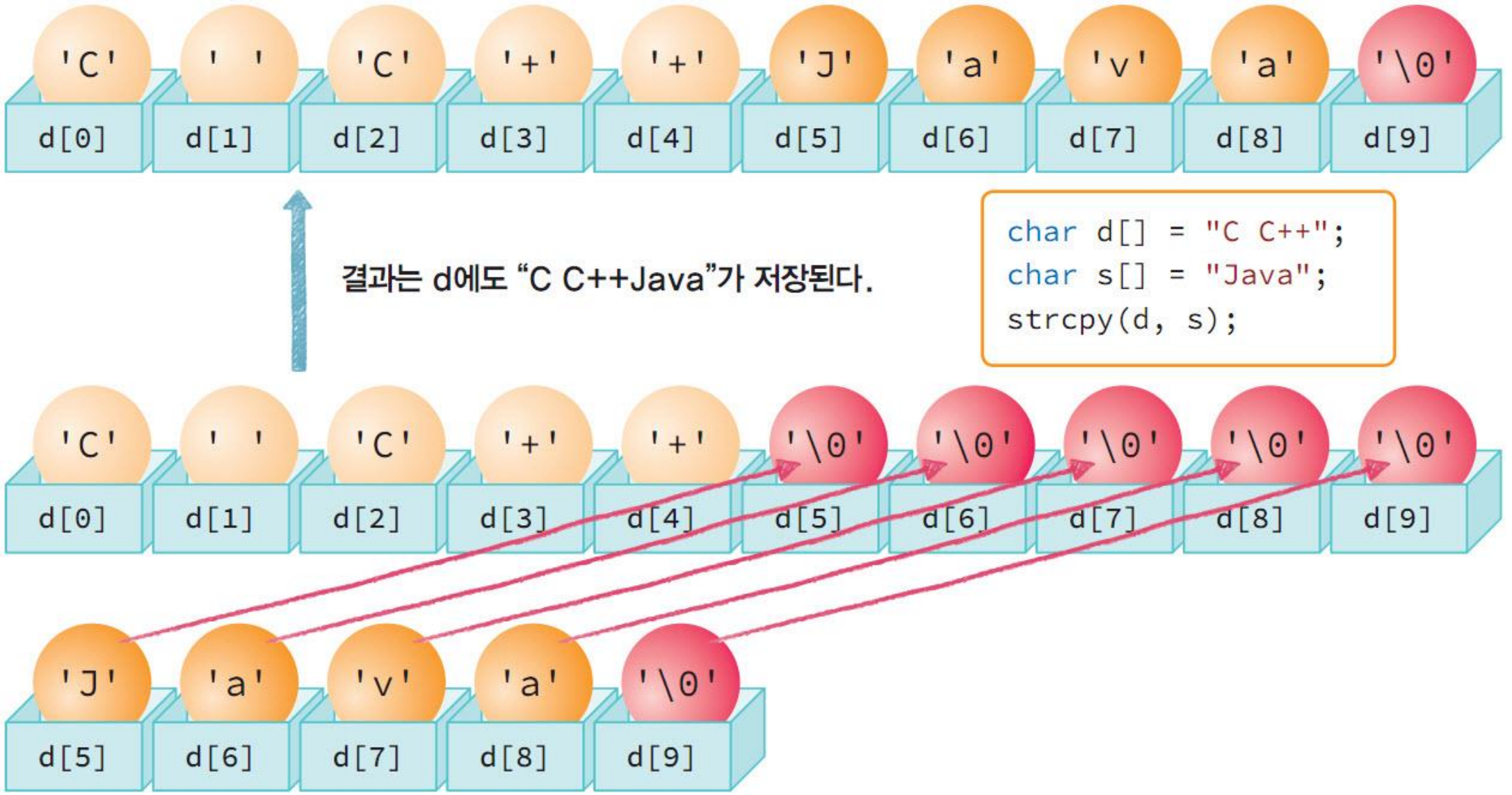
```
char * strncat(char * dest, const char * source, size_t maxn);
```

- 앞 문자열 dest에 뒤 문자열 source중에서 n개의 크기만큼을 연결(concatenate)해 저장하며, 이 연결된 문자열을 반환하고 뒤 문자열은 수정될 수 없다.
- 지정한 maxn이 문자열 길이보다 크면 null 문자까지 연결한다.

```
errno_t strcat_s(char * dest, size_t sizedest, const char * source);
errno_t strncat_s(char * dest, size_t sizedest, const char * source, size_t maxn);
```

- 두 번째 인자인 sizedest는 정수형으로 dest의 크기를 입력한다.
- 반환형 errno_t는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수strcat_s()와 strncat_s()의 사용을 권장한다.

# 함수 strcpy()


'C' ' ' 'C' '+' '+' 'J' 'a' 'v' 'a' '\0'
d[0] d[1] d[2] d[3] d[4] d[5] d[6] d[7] d[8] d[9]
결과는 d에도 "C C++Java"가 저장된다.
char d[] = "C C++";
char s[] = "Java";
strcpy(d, s);
'C' ' ' 'C' '+' '+' '\0' '\0' '\0' '\0' '\0'
d[0] d[1] d[2] d[3] d[4] d[5] d[6] d[7] d[8] d[9]
'J' 'a' 'v' 'a' '\0'
d[5] d[6] d[7] d[8] d[9]

# Source Code #10: strcat.c

```c
// file: strcat.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
    char dest[80] = "C";

    printf("%s\n", strcat(dest, " is "));
    //printf("%d\n", strcat_s(dest, 80, " is "));
    //printf("%s\n", dest);
    printf("%s\n", strncat(dest, "a java", 2));
    //printf("%d\n", strncat_s(dest, 80, "a proce", 2));
    //printf("%s\n", dest);
    printf("%s\n", strcat(dest, "procedural "));
    printf("%s\n", strcat(dest, "language."));

    return 0;
}
```

- 문자열 연결 함수 사용

```
C is
C is a
C is a procedural
C is a procedural language.
```

# 함수 strcat()

- 함수 strcpy()와 strcat()를 이용 시
  - 첫 인자로 문자열 포인터변수는 사용 불가능
  - 첫 번째 인자인 dest는 복사 또는 연결 결과가 저장될 수 있도록 충분한 공간을 확보
  - 문자열 관련 함수에서 단순히 문자열 포인터를 수정이 가능한 문자열의 인자로 사용 불가능
- 다음은 모두 오류 발생

```
char dest[5] = "C";
char *destc = "C";

strcpy(dest, "Java language");          //실행 시 오류발생
strcpy(destc, " Java language");        //실행 시 오류발생
strcat(dest, " is a language.");        //실행 시 오류발생
strcat(destc, " is a language.");       //실행 시 오류발생
```

# 함수 strtok()

- 문자열에서 구분자(delimiter)인 문자를 여러 개 지정하여 토큰을 추출하는 함수
  - 첫 번째 인자인 str은 토큰을 추출할 대상인 문자열
  - 두 번째 인자인 delim은 구분자로 문자의 모임인 문자열
  - 첫 번째 인자인 str은 문자배열에 저장된 문자열을 사용
  - str은 문자열 상수를 사용 불가능

문자열 분리 함수

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

- 앞 문자열 str에서 뒤 문자열delim을 구성하는 구분자를 기준으로 순서대로 토큰을 추출하여 반환하는 함수이며, 뒤 문자열 delim은 수정될 수 없다.

```
char * strtok_s(char * str, const char * delim, char ** context);
```

- 마지막 인자인 context는 함수 호출에 사용되는 위치 정보를 위한 인자이며, Visual C++에서는 앞으로 함수 strtok_s()의 사용을 권장한다.

# 함수 strtok()

문자열: "C and C++\t language are best!"

- 구분자 delim이 " "인 경우의 토큰: C, and, C++\t, language, are, best! 총 6개
- 구분자 delim이 " \t"인 경우의 토큰: C, and, C++, language, are, best! 총 6개
- 구분자 delim이 " \t!"인 경우의 토큰: C, and, C++, language, are, best 총 6개

# Source Code #11: strtok.c

```c
// file: strtok.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
    char str1[] = "C and C++\t language are best!";
    char *delimiter = " ,\t!";
    //char *next_token;

    printf("문자열 \"%s\"을 >>\n", str1);
    printf("구분자[%s]를 이용하여 토큰을 추출 >>\n", delimiter);
    char *ptoken = strtok(str1, delimiter);
    //ptoken = strtok_s(str, delimiter, &next_token);
    while (ptoken) //(ptoken != NULL)
    {
        printf("%s\n", ptoken);
        ptoken = strtok(NULL, delimiter); //다음 토큰을 반환
        //ptoken = strtok_s(NULL, delimiter, &next_token); //다음 토큰을 반환
    }

    return 0;
}
```

- 문자열 분리 함수 사용
- 문장 ptoken = strtok(str, delimiter); 으로 첫 토큰을 추출
  - 결과를 저장한 ptoken이 NULL이면 더 이상 분리할 토큰이 없는 경우
  - 계속 토큰을 추출하려면 while 반복으로 추출된 토큰이 있는지를 (ptoken != NULL) 로 검사
  - NULL을 첫 번째 인자로 다시 strtok(NULL, delimiter)를 호출하면 그 다음 토큰을 반환

```
문자열 "C and C++        language are best!"을 >>
구분자[ ,       !]를 이용하여 토큰을 추출 >>
C
and
C++
language
are
best
```

# 문자열의 길이와 위치 검색

- 함수 strlen()
  - NULL 문자를 제외한 문자열 길이를 반환하는 함수
- 함수 strlwr()
  - 인자를 모두 소문자로 변환하여 반환
- 함수 strupr()
  - 인자를 모두 대소문자로 변환하여 반환

# 다양한 문자열 관련 함수

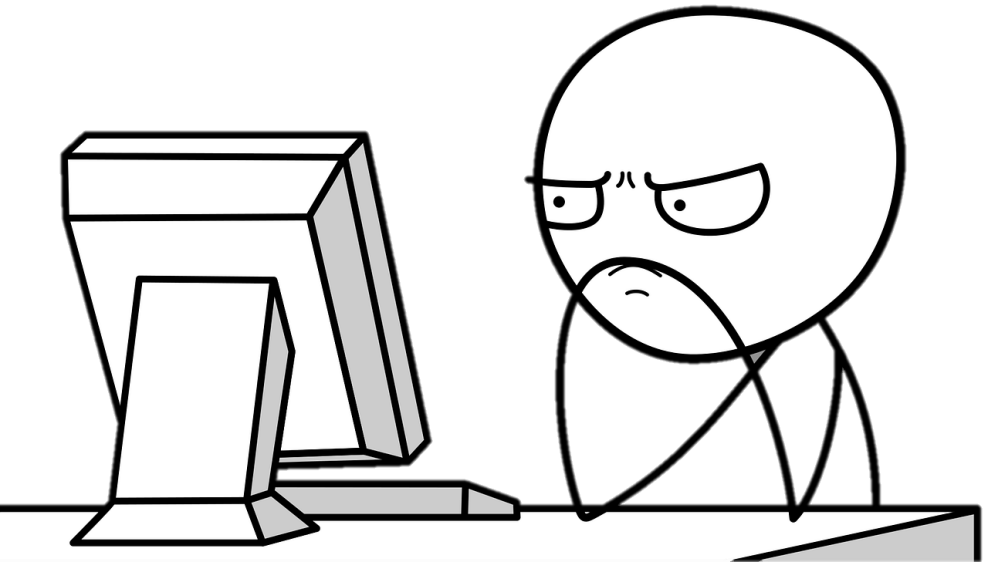
| 함수원형 | 설명 |
| --- | --- |
| `char * strlwr(char * str);`<br>`errno_t _strlwr_s(char * str, size_t strsize); //Visual C++ 권장함수` | 문자열 str을 모두 소문자로 변환하고 변환한 문자열을 반환하므로 str은 상수이면 오류가 발생하며, errno_t는 정수형의 오류번호이며, size_t도 정수형으로 strsize는 str의 길이 |
| `char * strupr(char * str);`<br>`errno_t _strupr_s(char * str, size_t strsize); //Visual C++ 권장함수` | 문자열 str을 모두 대문자로 변환하고 변환한 문자열을 반환하므로 str은 상수이면 오류가 발생하며, errno_t는 정수형의 오류번호이며, size_t도 정수형으로 strsize는 str의 길이 |
| `char * strpbrk(const char * str, const char * charset);` | 앞의 문자열 str에서 뒤 문자열 charset에 포함된 문자가 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환 |
| `char * strstr(const char * str, const char * strsearch);` | 앞의 문자열 str에서 뒤 문자열 strsearch이 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환 |
| `char * strchr(const char * str, char ch);` | 앞의 문자열 str에서 뒤 문자 ch가 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환 |

# Lab #02: 문자열을 역순으로 저장하는 함수 reverse() 구현

- 함수 memcpy()를 사용
  - 문자열 상수를 배열에 저장하여 출력한 후,
  - 함수 reverse()를 호출
    - 문자열을 역순으로 저장한 후 그 결과를 출력
    - 문자열 배열을 역순으로 저장하는 함수
  - char 일차원 배열 s[50]를 선언하여 함수 memcpy()로 문자열 "C Programming!"을 저장
  - 함수 reverse(char str[])는 문자열 str을 역순으로 다시 저장
- 다음 결과와 같이 원 문자열과 역순 문자열을 출력
  - C Programming!
  - !gnimmargorP C

```c
// strreverse.c:
#include <stdio.h>
#include

void reverse(char str[]);

int main(void)
{
    char s[50];
    (s, "C Programming!", strlen("C Programming!") + 1);
    printf("%s\n", s);

    reverse(s);
    printf("%s\n", s);

    return 0;
}

void reverse(char str[])
{
    for (int i = 0, j = strlen(str) - 1; i < j; i++, j--)
    {
        char c = str[i];

        str[j] = c;
    }
}
```

# 3. 여러 문자열 처리



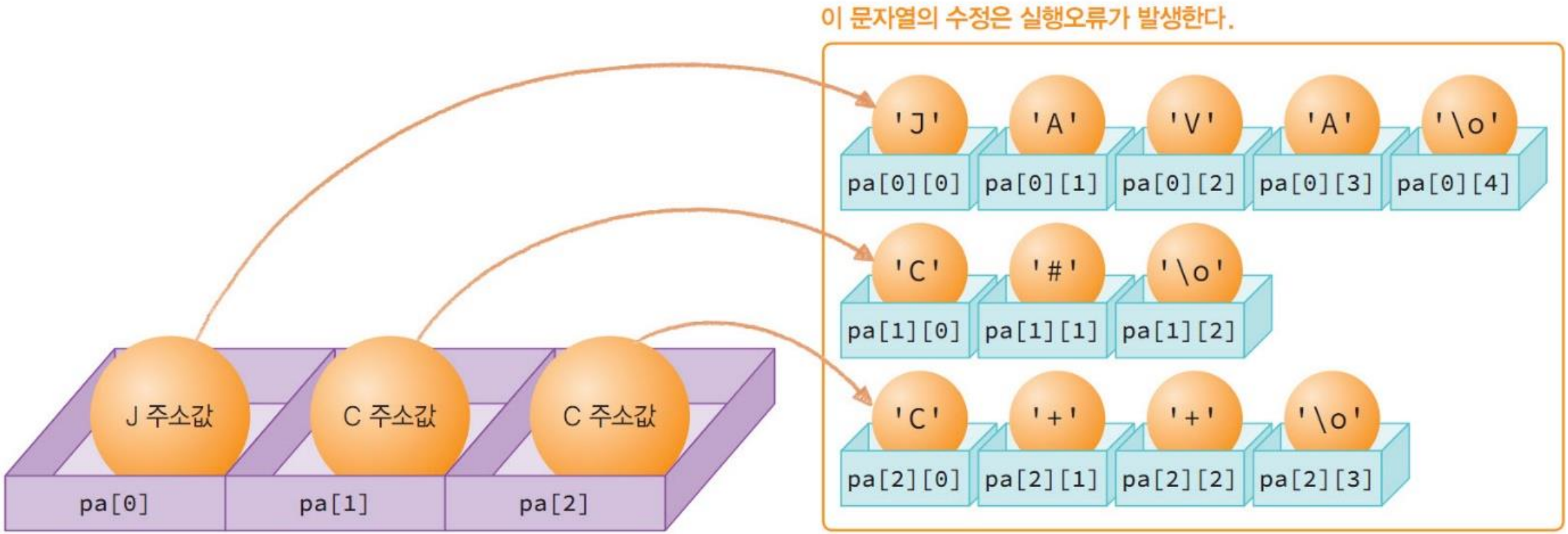
# 문자 포인터 배열

- 문자 포인터 배열을 이용하는 방법
  - 여러 개의 문자열을 처리하는 하나의 방법
  - 하나의 문자 포인터가 하나의 문자열을 참조 가능
  - 문자 포인터 배열은 여러 개의 문자열을 참조 가능
- 장 단점
  - 문자 포인터 배열 이용 방법은 각각의 문자열 저장을 위한 최적의 공간을 사용
  - 문자 포인터를 사용해서는 문자열 상수의 수정은 불가능
  - 문장 pa[0][2] = 'v';와 같이 문자열의 수정은 실행오류가 발생

# 문자 포인터 배열

```
char *pa[] = {"JAVA", "C#", "C++"};
//각각의 3개 문자열 출력
printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n", pa[2]);
```

배열의 크기는 문자열 개수인 3을 지정하거나 빈 공백으로 한다.



# 이차원 문자 배열

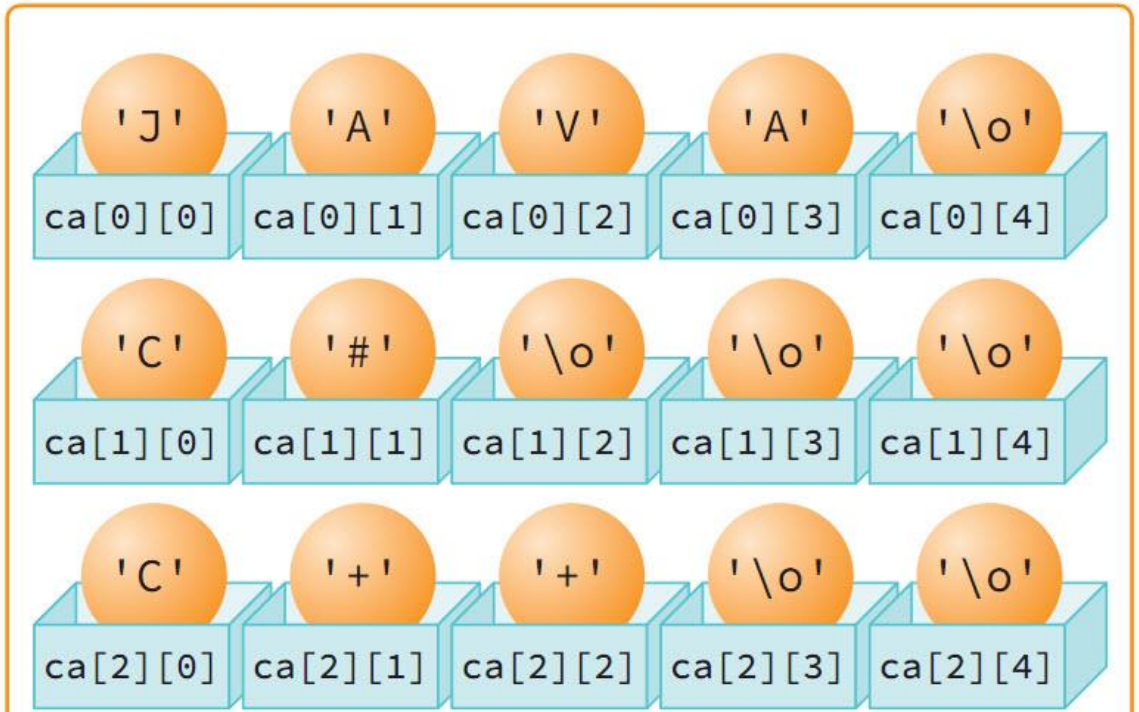
- 여러 개의 문자열을 처리하는 다른 방법
  - 문자의 이차원 배열을 이용하는 방법
  - 이차원 배열의 열 크기는 문자열 중에서 가장 긴 문자열의 길이보다 1 크게 지정
    - 가장 긴 문자열 "java"보다 1이 큰 5를 2차원 배열의 열 크기로 지정
  - 물론 이차원 배열의 행의 크기는 문자열 수
    - 3으로 지정하거나 공백으로 비워 둠
- 장 단점
  - 문자의 이차원 배열에서 모든 열 수가 동일하게 메모리에 할당
    - 열의 길이가 서로 다른 경우에는 '\0' 문자가 들어가 낭비
  - 문자열을 수정 가능
    - ca[0][2] = 'v';와 같이 원하는 문자 수정이 가능

# 이차원 문자 배열

```
char ca[][5] = {"JAVA", "C#", "C++"};
//각각의 3개 문자열 출력
printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);
```

첫 번째(행) 크기는 문자열 갯수를 지정하거나 빈 공백으로 두며, 두 번째(열) 크기는 문자열 중에서 가장 긴 문자열의 길이보다 1크게 지정한다.

이 문자열의 수정은 실행오류가 발생한다.



# Source Code #12: strarray.c

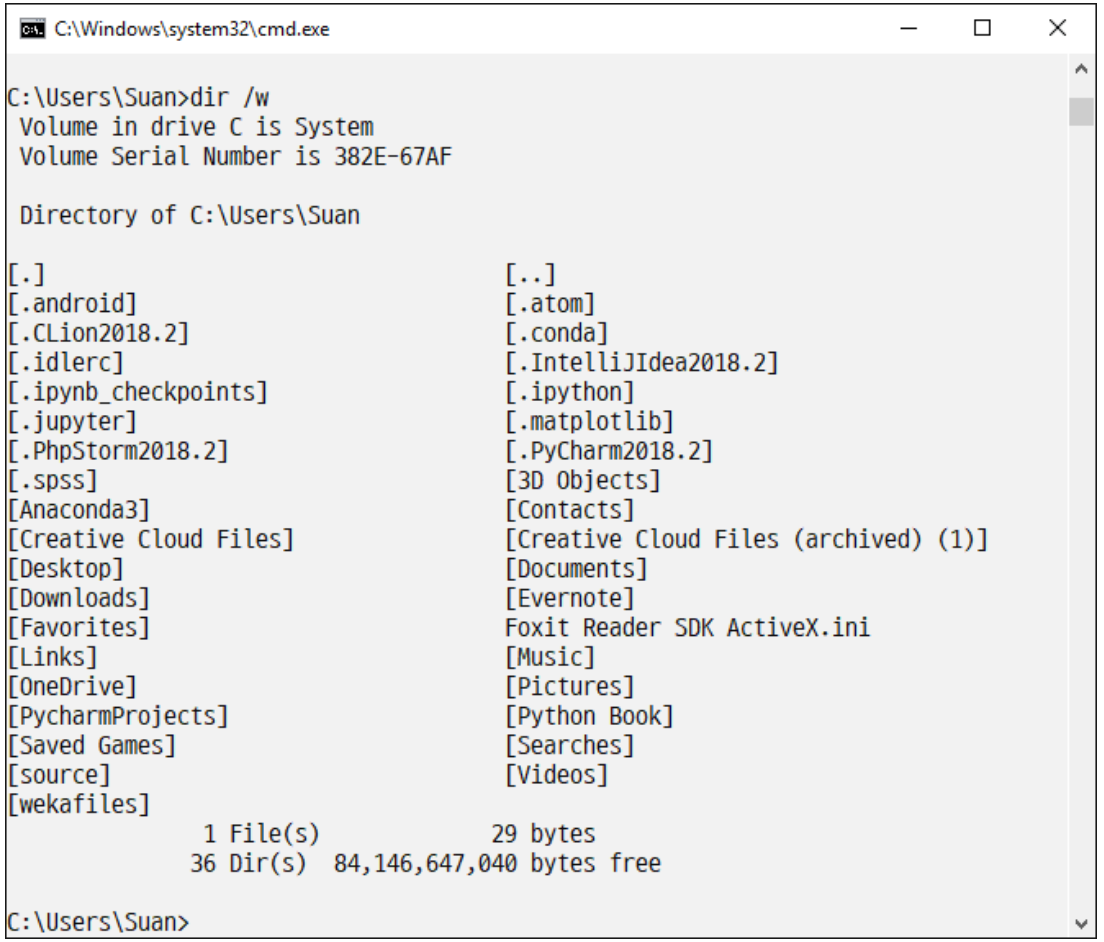
```c
// file: strarray.c
#include <stdio.h>

int main(void)
{
    char *pa[] = { "JAVA", "C#", "C++" };
    char ca[][5] = { "JAVA", "C#", "C++" };

    //각각의 3개 문자열 출력
    //pa[0][2] = 'v';   //실행 오류 발생
    //ca[0][2] = 'v';   //수정 가능
    printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n", pa[2]);
    printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);

    //문자 출력
    printf("%c %c %c\n", pa[0][1], pa[1][1], pa[2][1]);
    printf("%c %c %c\n", ca[0][1], ca[1][1], ca[2][1]);

    return 0;
}
```

- 문자 포인터 배열과 이차원 문자 배열

```
JAVA C# C++
JAVA C# C++
A # +
A # +
```

# 명령행 인자

- main(int argc, char *argv[])
- 프로그램 dir를 개발한다면 옵션에 해당하는 "/w"를 어떻게 인식할까?
  - 명령행 인자command line arguments를 사용하는 방법
  - 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법
- 프로그램에서 명령행 인자를 받으려면
  - 두 개의 인자 argc와 argv를 (int argc, char * argv[])로 기술
  - 매개변수 argc는 명령행에서 입력한 문자열의 수
  - argv[]는 명령행에서 입력한 문자열을 전달 받는 문자 포인터 배열
  - 실행 프로그램 이름도 하나의 명령행 인자에 포함



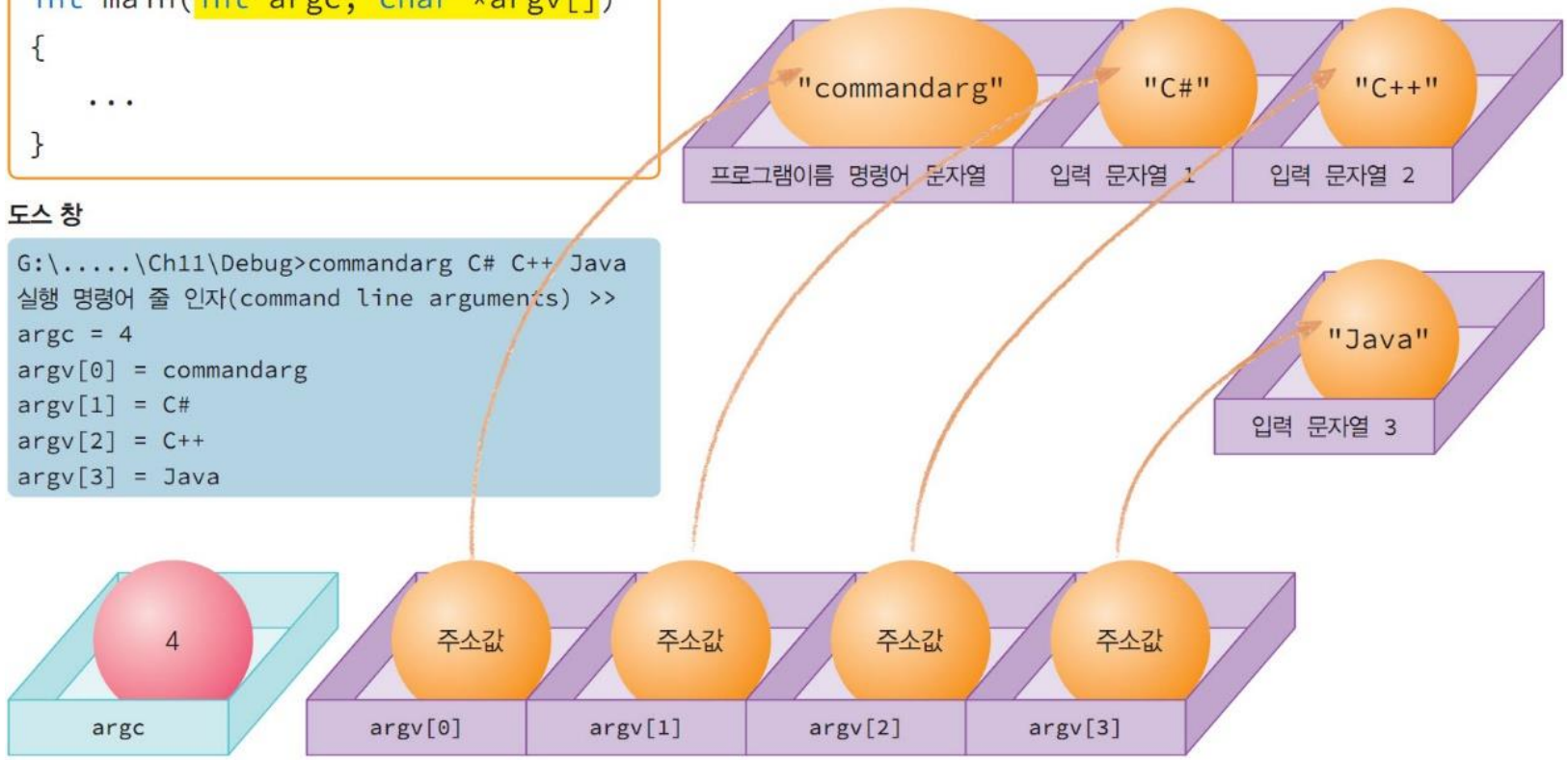
# 명령행 인자 실행 샘플

- 명령행에서 실행파일의 이름이 commandarg
  - 옵션으로 C# C++ Java: 프로그램을 실행한 결과
  - 명령행 인자로 프로그램을 실행하면 다음과 같은 구조의 문자열이 전달

```
int main(int argc, char *argv[])
{
    ...
}
```

도스 창

```
G:\.....\Ch11\Debug>commandarg C# C++ Java
실행 명령어 줄 인자(command line arguments) >>
argc = 4
argv[0] = commandarg
argv[1] = C#
argv[2] = C++
argv[3] = Java
```



# Source Code #13: commandarg.c

```c
// file: commandarg.c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int i = 0;

    printf("실행 명령행 인자(command line arguments) >>\n");
    printf("argc = %d\n", argc);
    for (i = 0; i < argc; i++)
        printf("argv[%d] = %s\n", i, argv[i]);

    return 0;
}
```

- 인자 argc, argv 활용
- 다음은 위 프로그램을 도스 창에서 실행한 결과
  - 도스 창에서 실행한 경우, 실행 결과의 첫 인자 값이 실행파일 이름

```
E:\컴퓨터 프로그래밍 (Computer Programming)\Source Code\Ch11\Debug>Project13 C언
어 진짜 재미있다.
실행 명령행 인자(command line arguments) >>
argc = 4
argv[0] = Project13
argv[1] = C언어
argv[2] = 진짜
argv[3] = 재미있다.
```

# Lab #03: 여러 문자열을 배열에 저장한 후 출력

- 문자 포인터 배열 pstr을 선언
  - 문자배열 이름을 초기화로 저장
- 변수 str1, str2, str3와 pstr을 사용
  - 저장된 문자열과 문자, 적절히 출력
  - str1, str2, str3을 선언하면서 문자열 "JAVA", "C#", "C++"를 저장
  - pstr을 선언하면서 문자열 포인터인 str1, str2, str3을 저장
- 다음과 같이 문자열과 문자를 출력
  - JAVA C# C++
  - J # +
  - A # +

```c
// file: strprocess.c
#include <stdio.h>

int main(void)
{
    char str1[] = "JAVA";
    char str2[] = "C#";
    char str3[] = "C++";

    char *pstr[] = {                    };

    //각각의 3개 문자열 출력
    printf("%s ", pstr[0]);
    printf("%s ",       );
    printf("%s\n", pstr[2]);

    //문자 출력
    printf("%c %c %c\n", str1[0], str2[1], str3[2]);
    printf("%c %c %c\n", pstr      , pstr      , pstr      );

    return 0;
}
```


I
PROGRAMMING