

# 제 12 장 문자와 문자열

---

- 01 문자와 문자열
- 02 문자열 관련 함수
- 03 여러 문자열 처리

# 문자와 문자열의 개념

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

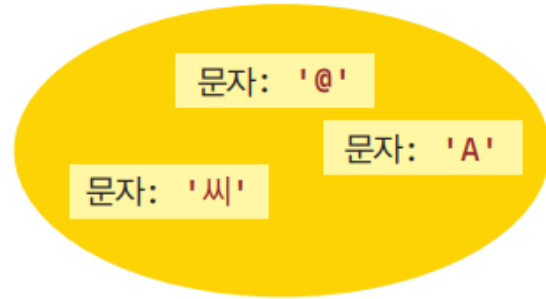


그림 12-1 문자 상수와 선언

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

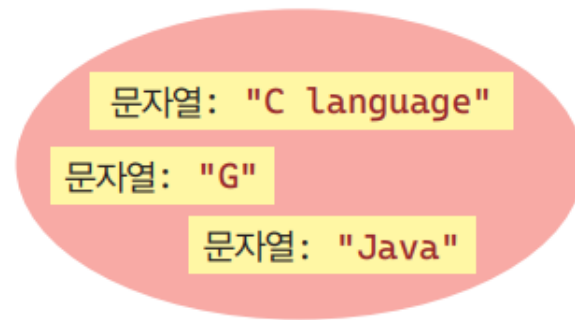


그림 12-2 문자열 상수와 선언

# 문자와 문자열의 선언

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

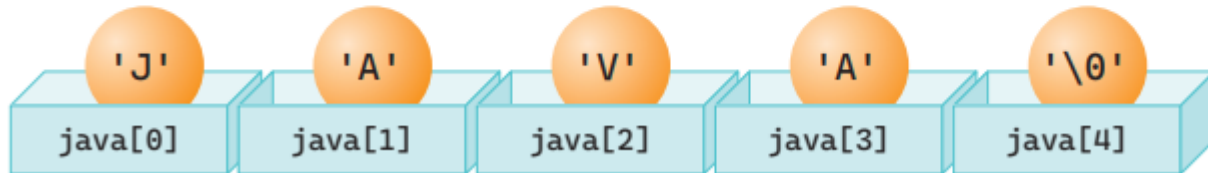
```
char csharp[3];
```

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

//문자 하나하나 저장 시 마지막에 '\\0' 문자 저장

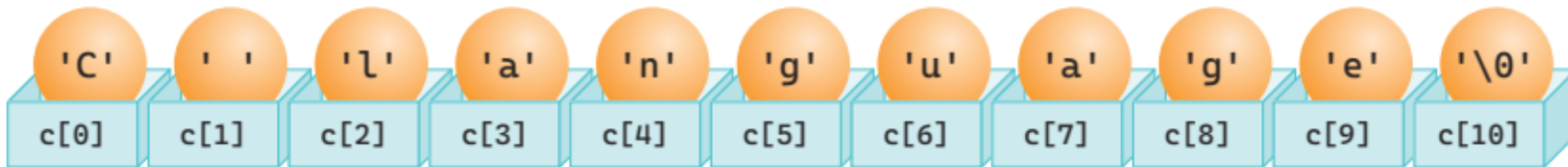
```
char java[] = {'J', 'A', 'V', 'A', '\\0'};
```

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



# 문자열을 선언하는 편리한 다른 방법

```
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
```

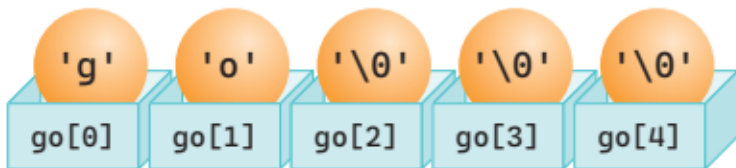


그림 12-5 문자열 상수로 문자배열 초기화

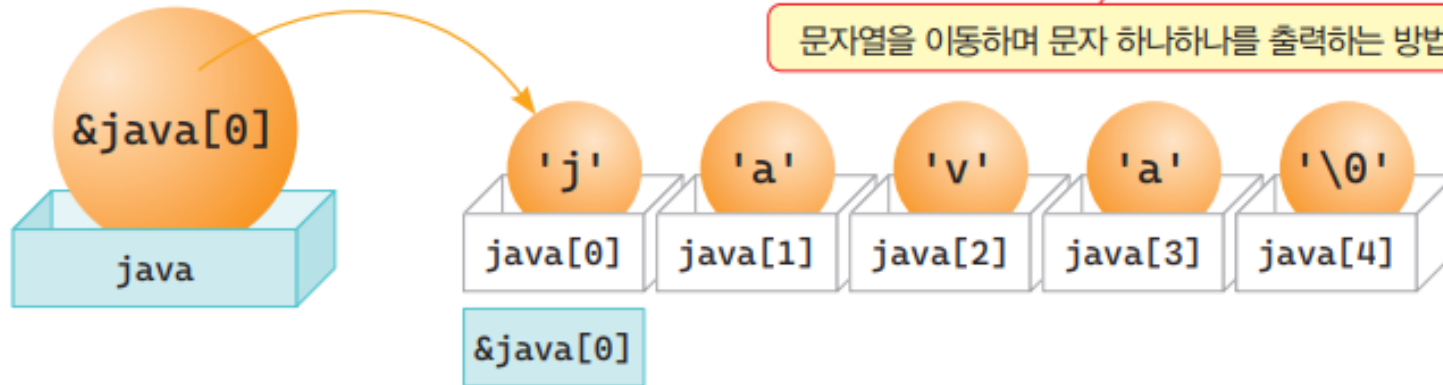
```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char ch = 'A';
06      printf("%c %d\n", ch, ch);
07
08      char java[] = { 'J', 'A', 'V', 'A', '\0' };
09      printf("%s\n", java);
10
11
12      char py[] = "Python";
13      printf("%s\n", py);
14
15
16      char csharp[5] = "C#";
17      printf("%s\n", csharp);
18
19
20      printf("%c %c\n", csharp[0], csharp[1]);
21
22      return 0;
23  }
```

# 문자열 구성하는 문자 참조

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

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

문자열을 이동하며 문자 하나하나를 출력하는 방법



## 문자 단위로 참조

텍스트 분석, 데이터 변환, 암호화, 유효성 검사, 파서 구현 등에 사용

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     char *java = "java";
06     printf("%s ", java);
07
08     //문자 포인터가 가리키는 문자 이후를 하나씩 출력
09     int i = 0;
10     while (java[i]) //while (java[i] != '\0')
11         printf("%c", java[i++]);
12     printf(" ");
13
14     i = 0;
15     while (*(java + i) != '\0') //java[i]는 *(java + i)와 같음
16         printf("%c", *(java + i++));
17     printf("\n");
18
19     //수정 불가능, 실행 결과에 문제 발생
20     java[1] = 'A';
21     printf("%c", java[1]);
22
23     return 0;
24 }
```

java[i]는 \*(java + i)와 동일한 표현 방식이므로  
java[i++]도 \*(java + i++)와 같다.





# '\0' 문자에 의한 문자열 분리

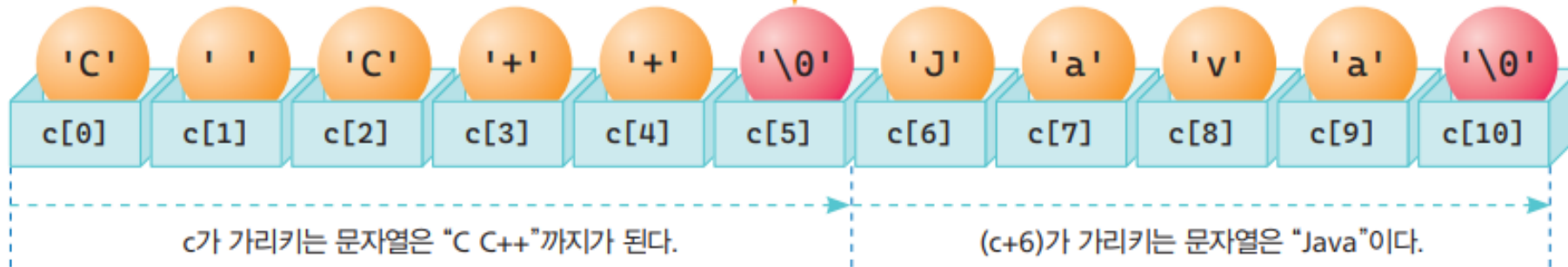
```
char c[] = "C C++ Java";  
c[5] = '\0';  
printf("%s\n%s\n", c, (c+6));
```

출력

C C++  
Java

'\0'

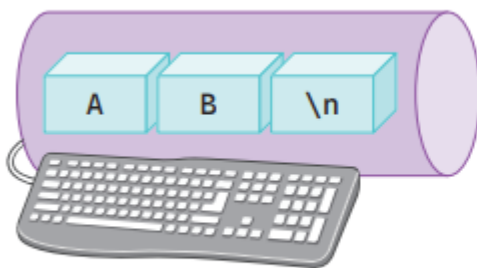
문장 c[5] = '\0'에 의해 문자열이 2개로 나뉘어진다.



# 문자 입출력 함수

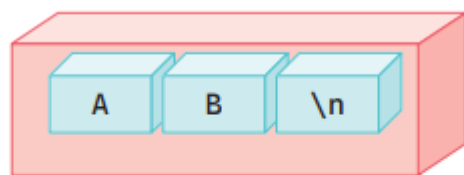
- 함수 `getchar()`, `putchar()`
  - 라인 버퍼링(line buffering) 방식

키보드 표준입력



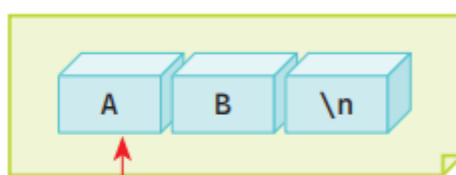
[enter] 키를 누르기 전까지 입력 자료의 편집이 가능하며 버퍼에 저장된다.

버퍼에 저장



임시 기억장소인 버퍼에 저장되었다가 [enter] 키를 만나면 프로그램으로 입력이 시작된다

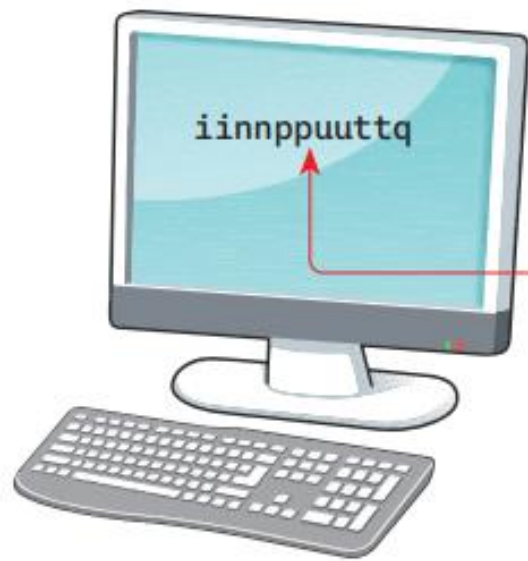
입력 처리



버퍼 자료를 이용하여 `getchar()`를 수행한다.

# 버퍼를 사용하지 않는 문자 입력 함수

- 함수 `_getche()`
  - `<conio.h>` 에 정의

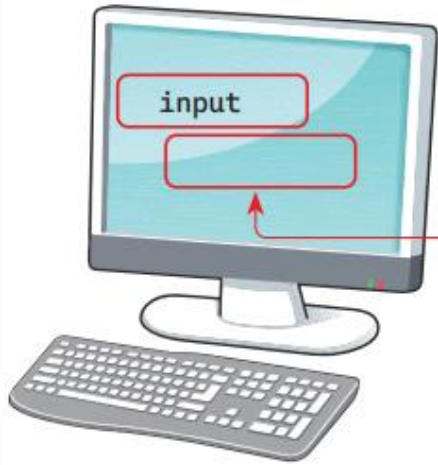


```
char ch;  
while ( (ch = _getche()) != 'q')  
    putchar(ch);
```

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

# 입력한 문자가 화면에 보이지 않는 함수

- 함수 `_getch()`



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

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

**표 12-1** 문자입력 함수 scanf(), getchar(), \_getche(), \_getch()의 비교

함수	scanf("%c", &ch)	getchar()	_getche()	_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)	입력하면 바로 표시		입력하면 바로 표시	표시 안됨
입력문자 수정	가능		불가능	

```
01 #include <stdio.h>
02 #include <conio.h>
03
04 int main(void)
05 {
06     char ch;
07
08     printf("getchar() 사용 (q 입력 시 종료):\n");
09     while ((ch = getchar()) != 'q')
10         putchar(ch);
11
12     printf("\n\n_getche() 사용 (q 입력 시 종료):\n");
13     while ((ch = _getche()) != 'q')
14         _putch(ch);
15
16     printf("\n\n_getch() 사용 (q 입력 시 종료):\n");
17     while ((ch = _getch()) != 'q')
18         _putch(ch);
19
20     printf("\n");
21     return 0;
22 }
```

Microsoft Visual Studio 디버그 × + ▾

Enter your password: \*\*\*\*\*

Your password is: hello

D:\ekryu\_vs\p1\Debug\p1.exe(프로세스

이 창을 닫으려면 아무 키나 누르세요..

```

1  #include <stdio.h>
2  #include <conio.h>
3
4  #define MAX_LEN 100
5
6  int main() {
7      char pwd[MAX_LEN];
8      int i = 0;
9      char c;
10
11     printf("Enter your password: ");
12     while (1) {
13         c = _getch();
14         if (c == '\r' || c == '\n') {
15             break;
16         }
17
18         if (i < MAX_LEN - 1) {
19             pwd[i++] = c;
20             printf("*");
21         }
22     }
23     pwd[i] = '\0';
24     printf("\nYour password is: %s\n", pwd);
25
26     return 0;
27 }

```

```

Microsoft Visual Studio 디버그
Enter your password: *****
Your password is: hello

D:\ekryu_vs\p1\Debug\p1.exe(프로세스
이 창을 닫으려면 아무 키나 누르세요...

```

Windows: Enter 키 → `\r`과 `\n` 제어 문자가 발생  
(Unix/Linux → `\n` 제어 문자 발생)

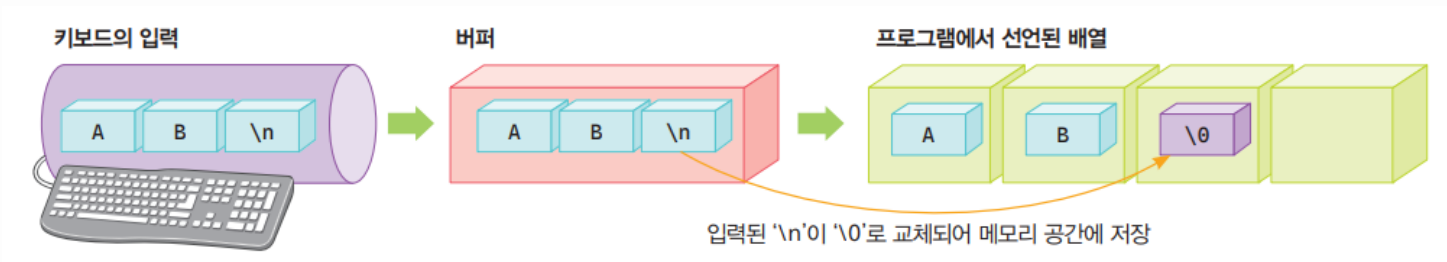


# 문자열의 입출력

- 함수 scanf()는 공백으로 구분되는 하나의 문자열을 입력
  - 입력 받은 문자열이 저장될 충분한 공간인 문자 배열 str을 선언
  - 함수 scanf("%s", str)
    - 형식제어문자 %s를 사용하여 문자열 입력
  - 함수 printf("%s", str)
    - %s를 사용하여 문자열을 출력
- gets()와 puts()
  - 함수 gets()는 한 행의 문자열 입력에 유용한 함수
  - 함수 puts()는 한 행에 문자열을 출력하는 함수
  - 함수 gets(), puts(), gets\_s()를 사용하려면 헤더파일 stdio.h 를 삽입

# 함수 gets()

- [enter] 키를 누를 때까지 한 행을 버퍼에 저장한 후 입력처리
  - 함수 gets()는 마지막에 입력된 '\n'이 '\0'로 교체되어 인자인 배열에 저장



## 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.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {
    char str[100];

    printf("문자열 입력: ");
    gets(str); // 비추천 함수

    printf("입력한 문자열: %s\n", str);

    return 0;
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {
    char name[30];

    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin);

    printf("Hello, %s!\n", name);
    return 0;
}
```

학번 이름 학과 입력 >>

20222007 컴퓨터정보공학과 나윤희 ← 공백으로 구분해 snum, dept, name에 저장

출력: 20222007 컴퓨터정보공학과 나윤희

한 행에 학번 이름 학과 입력한 후 [enter]를 누르고 새로운 행에서 (ctrl + Z)를 누르십시오.

20222007 컴퓨터정보공학과 나윤희 ← 한 행 모두가 배열 line에 저장

20222007 컴퓨터정보공학과 나윤희

^Z ←

새로운 행에서 ctrl + Z 입력하면 while 반복 종료

20222007 컴퓨터정보공학과 나윤희

20222007 컴퓨터정보공학과 나윤희

^Z

```

01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 int main(void)
05 {
06     char name[20], dept[30]; //char *name, *dept; 실행 오류 발생
07     int snum;
08     printf("학번 이름 학과 입력 >>\n");
09     scanf("%d %s %s", &snum, dept, name);
10     printf("출력: %d %s %s\n", snum, dept, name);
11
12     char line[101]; //char *line 으로는 오류발생
13     printf("한 행에 학번 이름 학과 입력한 후 [enter]를 누르고 ");
14     printf("새로운 행에서 (ctrl + Z)를 누르십시오.\n");
15     while (gets(line))
16     {
17         puts(line);
18         printf("\n");
19     }
20
21     while (gets_s(line, 101))
22     {
23         puts(line);
24         printf("\n");
25     }
26
27     return 0;
28 }

```

이름과 학과가 저장될 공간인 문자배열 name[20]과 dept[30]을 선언, 배열크기 20, 30은 이름과 학과가 저장될 충분한 크기여야 하며, 특히 한글은 크기 2개에 하나의 글자가 입력됨

한 줄에 입력되는 모든 문자열이 입력되도록 충분한 크기의 문자배열 line[101] 선언

함수 gets(line) 호출로 한 줄 전체를 입력 받음, 새로운 행에서 ctrl + Z 입력하면 while 반복 종료

함수 gets\_s(line, 101) 호출로 한 줄 전체를 입력 받음, 새로운 행에서 ctrl + Z 입력하면 while 반복 종료

학번 이름 학과 입력 >>

20222007 컴퓨터정보공학과 나윤희

공백으로 구분해 snum, dept, name에 저장

출력: 20222007 컴퓨터정보공학과 나윤희

한 행에 학번 이름 학과 입력한 후 [enter]를 누르고 새로운 행에서 (ctrl + Z)를 누르십시오.

20222007 컴퓨터정보공학과 나윤희

한 행 모두가 배열 line에 저장

20222007 컴퓨터정보공학과 나윤희

^Z

새로운 행에서 ctrl + Z 입력하면 while 반복 종료

20222007 컴퓨터정보공학과 나윤희

20222007 컴퓨터정보공학과 나윤희

^Z


```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    char name[20], dept[30];
    int snum;

    printf("학번 이름 학과 입력 >>\n");
    scanf("%d %s %s", &snum, dept, name);
    printf("%d %s %s\n", snum, dept, name);

    char line[101];
    gets_s(line, 101);
    puts(line);
    printf("\n");

    return 0;
}
```



```
char line[MAX_LEN];

printf("\nEnter text (Ctrl + Z to end):\n");
while (fgets(line, MAX_LEN, stdin) != NULL) {
    line[strlen(line) - 1] = '\0';
    printf("%s\n", line);
}
printf("\nEnd of input.\n");
```



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char s[100];
    fgets(s, sizeof(s), stdin);

    char *p = s;
    while (*p)
        printf("%c", *p++);

    printf("\n");
    return 0;
}
```

# 문자와 문자열

01 문자와 문자열

**02 문자열 관련 함수**

03 여러 문자열 처리

# 다양한 문자열 라이브러리 함수

- 헤더파일 **string.h** 라이브러리 함수

- 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리 함수
- `size_t`
  - `unsigned __int64`
- `void *`
  - 아직 데이터형이 정해지지 않은 포인터를 의미

표 12-2 문자열 배열에 관한 다양한 함수

함수원형	설명
<code>size_t strlen(const char *str)</code>	포인터 <code>src</code> 위치에서부터 널 문자를 제외한 문자열의 길이 반환
<code>void *memcpy(void *dest, const void *src, size_t n)</code>	포인터 <code>src</code> 위치에서 <code>dest</code> 에 <code>n</code> 바이트를 복사한 후 <code>dest</code> 위치 반환
<code>void *memchr(const void *str, int c, size_t n)</code>	메모리 <code>str</code> 에서 <code>n</code> 바이트까지 문자 <code>c</code> 를 찾아 그 위치를 반환
<code>int memcmp(const void *str1, const void *str2, size_t n)</code>	메모리 <code>str1</code> 과 <code>str2</code> 를 첫 <code>n</code> 바이트를 비교 검색하여 같으면 0, 다르면 음수 또는 양수 반환
<code>void *memmove(void *dest, const void *src, size_t n)</code>	포인터 <code>src</code> 위치에서 <code>dest</code> 에 <code>n</code> 바이트를 복사한 후 <code>dest</code> 위치 반환
<code>void *memset(void *str, int c, size_t n)</code>	포인터 <code>src</code> 위치에서부터 <code>n</code> 바이트까지 문자 <code>c</code> 를 지정한 후 <code>src</code> 위치 반환

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

```
// 문자열 길이 계산
```

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

```
// 문자열 복사
```

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

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

```
// 문자열 연결
```

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

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

```
// 문자열 비교
```

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

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

```
// 문자열 검색 (문자 위치)
```

```
char* strchr(const char* str, int c);    // 처음 등장 위치
```

```
char* strrchr(const char* str, int c);    // 마지막 등장 위치
```

```
// 문자열 검색 (부분 문자열 위치)
```

```
char* strstr(const char* haystack, const char* needle);
```

```
// 문자열 분할 (토큰화)
```

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

# 함수 strcmp()와 strncmp()

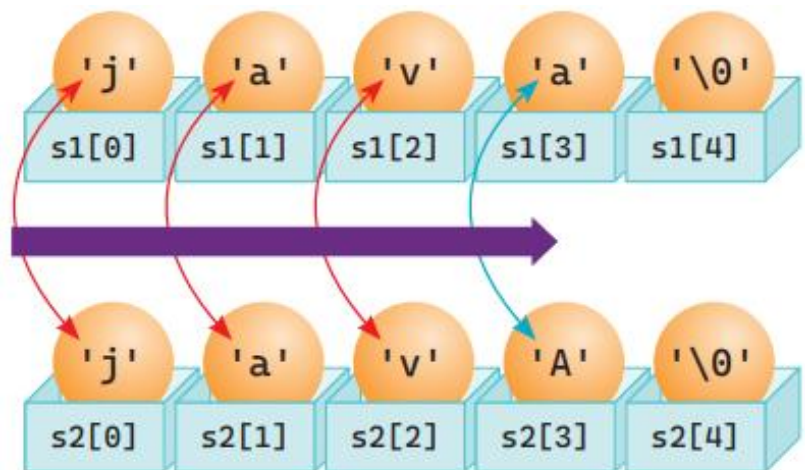
문자열 비교 함수: 헤더파일 string.h 삽입

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

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



- `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

```
#include <stdio.h>

int my_strcmp(const char* s1, const char* s2);

int main() {
    printf("%d\n", my_strcmp("apple", "apple"));    // 0
    printf("%d\n", my_strcmp("apple", "apricot")); // 음수
    printf("%d\n", my_strcmp("banana", "apple"));  // 양수
    return 0;
}
```

```
int my_strcmp(char *s1, char *s2) {
    while (*s1 && *s1 == *s2) {
        s1++;
        s2++;
    }
    return *s1 - *s2;
}
```

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

int main(void)
{
    char src[20] = "C Python";
    char dst[20];

    printf("%s\n", src);
    printf("%zu\n", strlen(src));
    memcpy(dst, src, strlen(src) + 1);
    printf("%s\n", dst);

    memcpy(dst, "안녕하세요!", strlen("안녕하세요!") + 1);
    printf("%s\n", dst);

    return 0;
}
```



```
int src[5] = { 1, 2, 3, 4, 5 };  
int dst[5];  
  
// 배열 복사  
memcpy(dst, src, 5 * sizeof(int));  
  
for (int i = 0; i < 5; i++) {  
    printf("%d ", dst[i]);  
}  
printf("\n");
```

**void \*memcpy(void \*dest, const void \*src, size\_t n);**

메모리 블록을 빠르게 복사

배열이나 구조체의 데이터를 복사할 때 유용

```
// 간단한 비밀번호 인증
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main() {
    char password[] = "secure";
    char input[50];

    printf("Enter password: ");
    scanf("%49s", input); // 최대 49자까지 입력받음

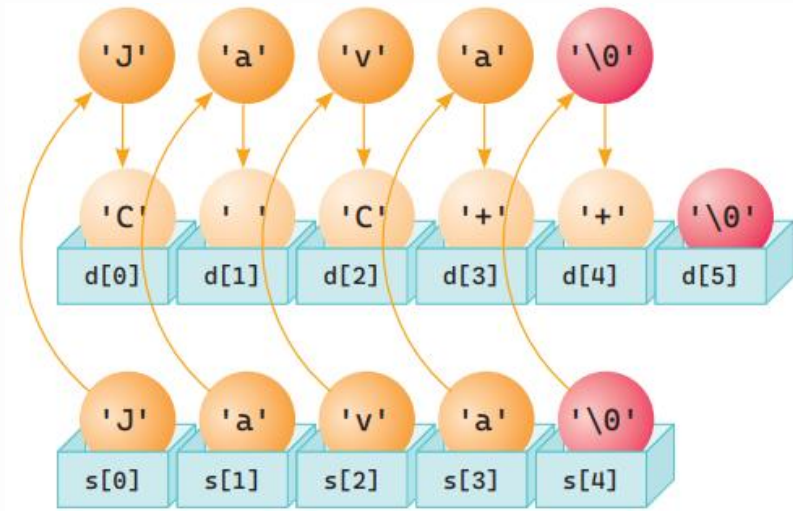
    if (strlen(input) > 10) {
        printf("Password is too long!\n");
    }
    else if (strcmp(password, input) == 0) {
        printf("Access granted.\n");
    }
    else {
        printf("Access denied.\n");
    }

    return 0;
}
```

# 함수 strcpy()

- 함수 strcpy()

- 문자열을 복사하는 함수



결과는 d에도 "java"가 저장된다.

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

- 함수 strncpy()

- 복사되는 최대 문자 수를 마지막 인자 maxn으로 지정하는 함수

```
char src[] = "Hello, World!";  
char dest[50];  
  
strcpy(dest, src);  
  
printf("%s\n", src);  
printf("%s\n", dest);
```

## 문자열 복사 함수

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

- 앞 문자열 dest에 처음에 뒤 문자열 null 문자를 포함한 source를 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 source 에서 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.

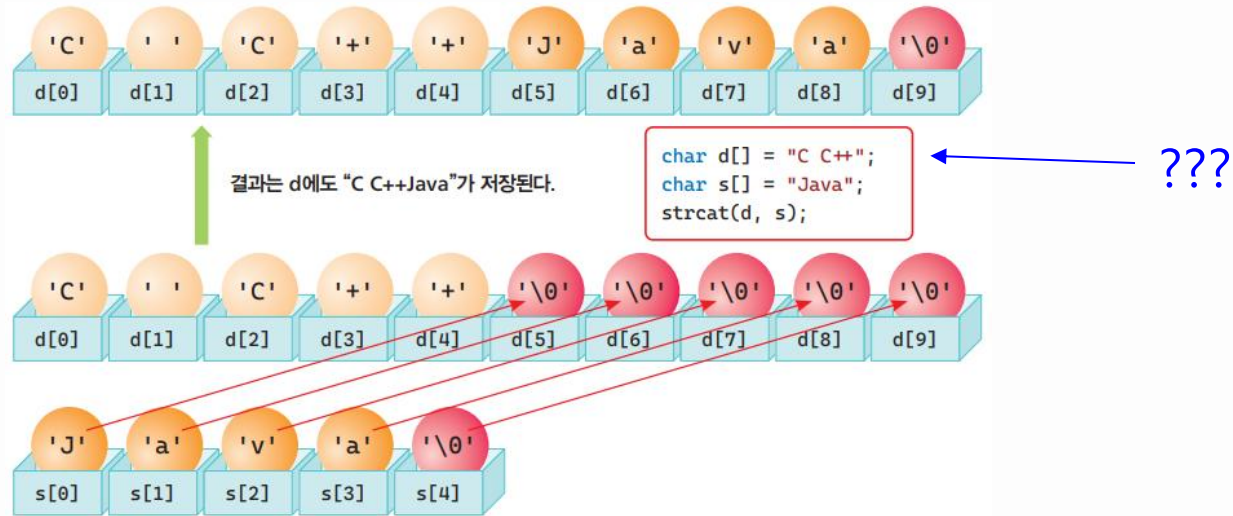
```
errno_t strcpy_s(char * dest, size_t sizedest, const char * source);
```

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

- 두 번째 인자인 sizedest는 정수형으로 dset 의 크기를 입력한다.
- 반환형 errno\_t 는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수 strcpy\_s()와 strncpy\_s()의 사용을 권장한다.

# 함수 strcat(), strncat()

- 하나의 문자열 뒤에 다른 하나의 문자열을 연이어 추가해 연결





## TIP 함수 strcpy()와 strcat()를 이용할 시 주의점

함수 strcpy()와 strcat()를 이용할 시 첫 번째 인자인 dest는 복사 또는 연결 결과가 저장될 수 있도록 충분한 공간을 확보해야 한다. 또한 문자열 관련 함수에서 단순히 문자열 포인터를 수정이 가능한 문자열의 인자로 사용할 수 없다. 즉 함수 strcpy()와 strcat()에서 첫 인자로 문자열 포인터변수는 사용할 수 없다. 그러므로 다음 소스는 모두 실행 시 바른 결과가 표시되지 않는다. 다양한 문자열 관련 함수에서 자료형이 (char \*)인 인자에는 문자열 상수를 사용할 수 없으며, 자료형이 (const char \*)인 인자에는 문자열 상수를 사용할 수 있다.

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

```
char *destc = "C";
```

```
strcpy(dest, "Java language"); //실행 시 문제 발생
```

```
strcpy(destc, " Java language"); //실행 시 문제 발생
```

```
strcat(dest, " is a language."); //실행 시 문제 발생
```

```
strcat(destc, " is a language."); //실행 시 문제 발생
```

## 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.

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04
05 int main(void)
06 {
07
08     char dest[80] = "Java";
09     char source[80] = "C is a language.";
10
11
12     printf("%s\n", strcpy(dest, source));
13     printf("%s\n", strncpy(dest, "C#", 2));
14     printf("%s\n\n", strncpy(dest, "C#", 3));
15
16
17
18     char data[80] = "C";
19
20
21     printf("%s\n", strcat(data, " is "));
22     printf("%s\n", strncat(data, "a java", 2));
23     printf("%s\n", strcat(data, "procedural "));
24     printf("%s\n", strcat(data, "language."));
25
26
27
28
29     return 0;
30
31 }
```



# 문자열 분리 함수 strtok()

- 문자열에서 구분자(delimiter)로 기준으로 분리

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

## strtok 함수

- 원본 문자열을 수정
- 원본 문자열을 유지하려면 문자열을 복사한 후 사용

```
01 #define _CRT_SECURE_NO_WARNINGS // strtok() 사용하기 위해
02 #include <stdio.h>
03 #include <string.h>
04
05 int main(void)
06 {
07     char str[] = "C and C++\t languages are best!";
08     char *delimiter = " !\t";
09     //char *next_token;
10
11     printf("문자열 \"%s\" 을 >>\n", str);
12     printf("구분자[%s]를 이용하여 토큰을 추출 >>\n", delimiter);
13     char* ptoken = strtok(str, delimiter);
14     //char* ptoken = strtok_s(str, delimiter, &next_token);
15     while (ptoken) //(ptoken != NULL)
16     {
17         printf("%s\n", ptoken);
18         ptoken = strtok(NULL, delimiter); //다음 토큰을 반환
19         //ptoken = strtok_s(NULL, delimiter, &next_token); //다음 토큰을 반환
20     }
21
22     return 0;
23 }
```

구분자가 공백문자, 느낌표 !, 수평탭 모두 3개

9, 14, 19행을 사용해 strtok\_S() 활용

두 번째 호출부터는 첫 인자를 NULL로 호출

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04
05 int main(void)
06 {
07     char str[] = "JAVA 2022 Python C";
08     printf("%zu\n", strlen("python")); //python 길이: 6
09     printf("%s, ", _strlwr(str));      //모두 소문자로 변환
10     printf("%s\n", _strupr(str));      //모두 대문자로 변환
11
12     //문자열 VA가 시작되는 포인터 반환: VA 2022 PYTHON C
13     printf("%s, ", strstr(str, "VA"));
14     //문자 A가 처음 나타나는 포인터 반환: AVA 2022 PYTHON C
15     printf("%s\n", strchr(str, 'A'));
16
17     return 0;
18 }
```

6

```
java 2022 python c, JAVA 2022 PYTHON C
VA 2022 PYTHON C, AVA 2022 PYTHON C
```

```
#include <ctype.h>

char* to_lowercase(char* str) {
    char* p = str;
    while (*p) {
        *p = tolower(*p);
        p++;
    }
    return str;
}
```

## 예제: 문자열을 역순으로 저장하는 함수 reverse() 구현

Microsoft Visual Studio 디버그 콘솔

```
Python C  
C nohtyP
```

## lab2reversestr.c

```
01  #include <stdio.h>
02  #include <string.h>
03
04  void reverse(char []);
05
06  int main(void)
07  {
08      char s[50];
09      char* str = "Python C";
10      memcpy(s, , strlen(str) + 1);
11      printf("%s\n", s);
12
13      reverse(s);
14      printf("%s\n", s);
15
16      return 0;
17  }
18
19  void reverse(char str[])
20  {
21      for (int i = 0, j = (int) strlen(str) - 1; i < j; i++, j--)
22      {
23          char c = str[i];
24          
25          str[j] = c;
26      }
27  }
```

Microsoft Visual Studio 디버그 콘솔

```
Python C
C nohtyP
```

# 문자와 문자열

01 문자와 문자열

02 문자열 관련 함수

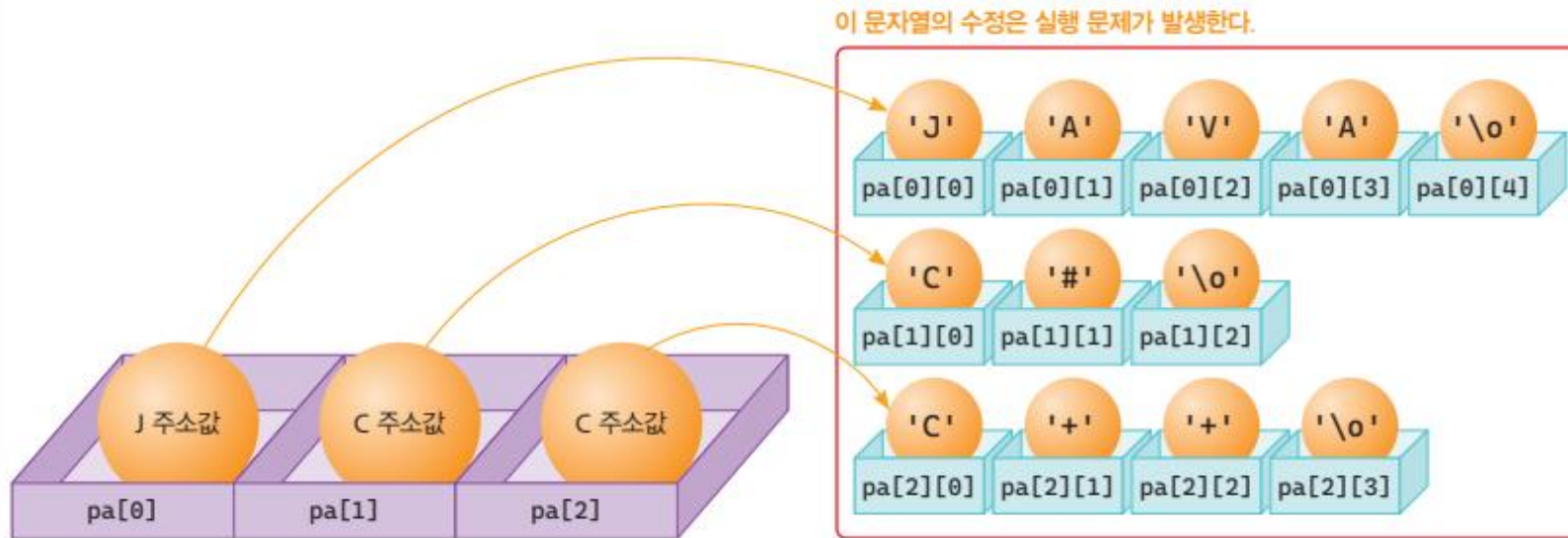
**03 여러 문자열 처리**

# 문자 포인터 배열

- 여러 개의 문자열을 처리하는 하나의 방법

```
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.

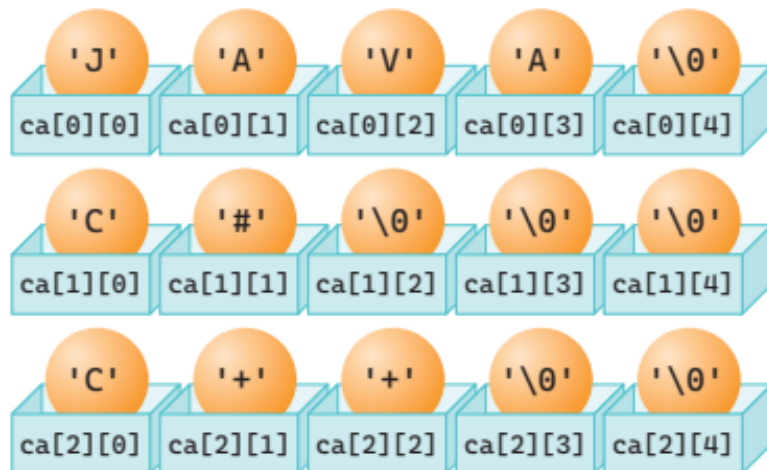


# 문자 이차원 배열을 이용하는 방법

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

이 문자열의 수정될 수 있다.





```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char *pa[] = { "JAVA", "C#", "C++" };
06      char ca[][5] = { "JAVA", "C#", "C++" };
07
08      //pa[0][2] = 'v';    //실행 문제 발생
09      //ca[0][2] = 'v';    //수정 가능
10      //각각의 3개 문자열 출력
11      printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n", pa[2]);
12      printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);
13
14      //문자 출력
15      printf("%c %c %c\n", pa[0][1], pa[1][1], pa[2][1]);
16      printf("%c %c %c\n", ca[0][1], ca[1][1], ca[2][1]);
17
18      return 0;
19  }
```

문자열을 구성하는 각각의 문자를 출력하려면  
pa[i][j]와 ca[i][j]을 형식제어문자 %c로 출력

사용자로부터 최대 99자의 문자열을 5개 입력받아 배열에 저장하고, 저장된 문자열을 모두 출력하는 프로그램 작성하시오.

문자열 5개를 입력하세요 :

문자열 1: apple

문자열 2: banana

문자열 3: hello

문자열 4: world

문자열 5: test

입력한 문자열들 :

1: apple

2: banana

3: hello

4: world

5: test

```

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

#define MAX_STRINGS 3
#define MAX_LENGTH 100

int main() {
    char* strings[MAX_STRINGS];

    for (int i = 0; i < MAX_STRINGS; i++) {
        strings[i] = (char*)malloc(MAX_LENGTH);
        if (strings[i] == NULL) {
            printf("메모리 할당 실패\n");
            return 1;
        }
    }
}

```

```

printf("문자열 %d개를 입력하세요:\n", MAX_STRINGS);
for (int i = 0; i < MAX_STRINGS; i++) {
    printf("문자열 %d: ", i + 1);
    fgets(strings[i], MAX_LENGTH, stdin);
}

printf("\n입력한 문자열들:\n");
for (int i = 0; i < MAX_STRINGS; i++) {
    printf("%d: %s", i + 1, strings[i]);
    free(strings[i]);
}

return 0;
}

```

문자열 3개를 입력하세요:

문자열 1: Hello

문자열 2: World!

문자열 3: Test string

입력한 문자열들:

1: Hello

2: World!

3: Test string

```
const char* temp_strs[] = {
    "Hello",
    "World",
    "Welcome",
    "To",
    "C programming"
};
int num_strings = sizeof(temp_strs) / sizeof(temp_strs[0]);

// 동적 메모리 할당
char** strs = (char**)malloc(num_strings * sizeof(char*));
if (strs == NULL) {
    fprintf(stderr, "메모리 할당 실패\n");
    return 1;
}

// 문자열 복사
for (int i = 0; i < num_strings; i++) {
    strs[i] = (char*)malloc((strlen(temp_strs[i]) + 1) * sizeof(char));
    if (strs[i] == NULL) {
        fprintf(stderr, "메모리 할당 실패\n");
        return 1;
    }
    strcpy(strs[i], temp_strs[i]);
}

// 문자열 출력
for (int i = 0; i < num_strings; i++) {
    printf("%s\n", strs[i]);
    free(strs[i]);
}
free(strs);
```

# 명령행 인자

실습예제 12-11

Prj11

11cmdarg.c

명령행 인자 출력

난이도: ★★

```
01 #include <stdio.h>
02
03 int main(int argc, char* argv[])
04 {
05     int i = 0;
06
07     printf("실행 명령행 인자(command line arguments) >>\n");
08     printf("argc = %d\n", argc);
09     for (i = 0; i < argc; i++)
10         printf("argv[%d] = %s\n", i, argv[i]);
11
12     return 0;
13 }
```

argc(argument count)에 인자의 수가, argv(argument variables)에는 인자인 여러 개의 문자열의 포인터가 저장된 배열이 전달

결과

```
실행 명령행 인자(command line arguments) >>
argc = 4
argv[0] = C:\Kang C\ch12\x64\Debug\Prj11.exe
argv[1] = Python
argv[2] = Go
argv[3] = Kotlin
```

비주얼 스튜디오에서 실행하면 전체 경로를 포함한 실행파일의 이름이 첫 번째 인자로 표시된다.

# 명령행 인자

```
관리자: C:\Windows\System32\cmd.exe
D:\Kang C\ch12\64\Debug>dir /w
D 드라이브의 볼륨: DATA_DRIVE
볼륨 일련 번호: 0AEE-91E2

D:\Kang C\ch12\64\Debug 디렉터리

[.]          [...]      Lab1.exe      Lab1.ilc      Lab1.pdb      Lab2.exe      Lab2.ilc      Lab2.pdb
Lab3.exe      Lab3.ilc      Lab3.pdb      Lab4.exe      Lab4.ilc      Lab4.pdb      Prj01.exe     Prj01.ilc
Prj01.pdb     Prj02.exe     Prj02.ilc     Prj02.pdb     Prj03.exe     Prj03.ilc     Prj03.pdb     Prj04.exe
Prj04.ilc     Prj04.pdb     Prj05.exe     Prj05.ilc     Prj05.pdb     Prj06.exe     Prj06.ilc     Prj06.pdb
Prj07.exe     Prj07.ilc     Prj07.pdb     Prj08.exe     Prj08.ilc     Prj08.pdb     Prj09.exe     Prj09.ilc
Prj09.pdb     Prj10.exe     Prj10.ilc     Prj10.pdb     Prj11.exe     Prj11.ilc     Prj11.pdb     Prj12.exe
Prj12.ilc     Prj12.pdb     Prj13.exe     Prj13.ilc     Prj13.pdb     Prj14.exe     Prj14.ilc     Prj14.pdb
Prj15.exe     Prj15.ilc     Prj15.pdb     Prj16.exe     Prj16.ilc     Prj16.pdb     Prj17.exe     Prj17.ilc
Prj17.pdb     test01.exe    test01.ilc    test01.pdb    test02.exe    test02.ilc    test02.pdb    test03.exe
test03.ilc    test03.pdb
72개 파일          22,968,768 바이트
2개 디렉터리 1,825,650,806,784 바이트 남음

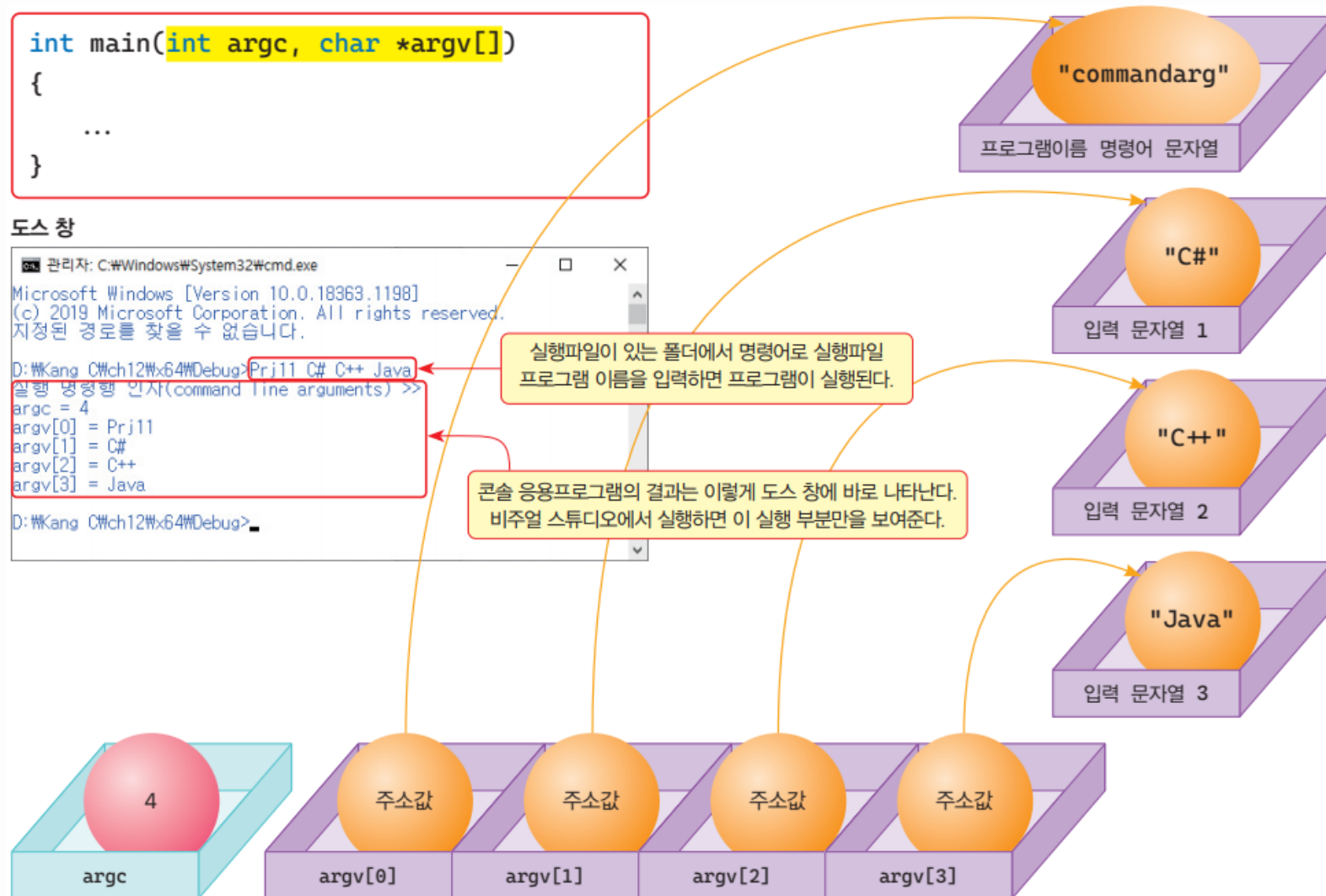
D:\Kang C\ch12\64\Debug>
```

도스 창에서 프로그램이 저장된 폴더의 경로와 >을 합쳐 프롬프트(prompt)라 하고, 명령어 프롬프트가 나타나는 이 줄에 명령어를 입력할 수 있으며, 이 줄을 명령행(command line)이라 한다.

## 명령행 인자

- 프로그램의 동작을 제어하거나
- 데이터를 입력받을 수 있음

# 명령행 인자 실행과 명령행 인자 전달





```
01 #include <stdio.h>
02
03 int main(int argc, char* argv[])
04 {
05     int i = 0;
06
07     printf("실행 명령행 인자(command line arguments) >>\n");
08     printf("argc = %d\n", argc);
09     for (i = 0; i < argc; i++)
10         printf("argv[%d] = %s\n", i, argv[i]);
11
12     return 0;
13 }
```

argc(argument count)에 인자의 수가, argv(argument variables)에는 인자인 여러 개의 문자열의 포인터가 저장된 배열이 전달

## 결과

실행 명령행 인자(command line arguments) >>

argc = 4

argv[0] = C:\Kang C\ch12\x64\Debug\Prj11.exe

argv[1] = Python

argv[2] = Go

argv[3] = Kotlin

비주얼 스튜디오에서 실행하면 전체 경로를 포함한 실행파일의 이름이 첫 번째 인자로 표시된다.



# LAB 여러 문자열 처리

Lab 12-3

lab3strprocess.c

난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      char str1[] = "Python";
06      char str2[] = "Kotlin";
07      char str3[] = "Tensorflow";
08
09      char *pstr[] = {  };
10
11      //각각의 3개 문자열 출력
12      printf("%s ", pstr[0]);
13      printf("%s ", );
14      printf("%s\n", pstr[2]);
15
16      //문자 출력
17      printf("%c %c %c\n", str1[0], str2[1], str3[2]);
18      printf("%c %c %c\n", pstr,
19                pstr, pstr);
20
21      return 0;
22  }
```

정답

```
09  char *pstr[] = { str1, str2, str3 };
13  printf("%s ", pstr[1]);
18  printf("%c %c %c\n", pstr[0][1], pstr[1][1], pstr[2][1]);
```

# 문자와 문자열

- ▶ 문자와 문자열을 이해
  - 문자와 **문자열의 표현과 저장** 방법
- ▶ 문자와 **문자열 입출력**을 이해
  - scanf(), printf(), getchar(), putchar(), getche(), getch(), putch()를 사용하여 문자 입출력
  - scanf(), printf(), gets(), puts()를 사용하여 문자열 입출력
- ▶ **문자열 관련 함수**를 이해
  - 문자열 비교 함수 strcmp(), strncmp()를 사용하여 문자열 비교
  - 문자열 연결 함수 strcat(), strncat()를 사용하여 문자열 연결
  - 문자열 토큰 추출 함수 strtok()를 사용하여 문자열에서 토큰 추출
  - 문자열 관련 함수 strlen(), strspn(), strcspn()의 사용 방법 이해
  - 문자열 관련 함수 strlwr(),strupr()의 사용 방법 이해
  - 문자열 관련 함수 strstr(), strchr()의 사용 방법 이해
- ▶ **여러 개의 문자열을 처리 방법**에 대해 이해
  - 문자 포인터 배열 방법과 2차원 문자 배열 방법의 차이
  - 명령행 인자의 필요성과 구현 방법 이해