

WEEK 11

문자와 문자열

학습목표

- I. 문자와 문자열, 문자열 처리에 관련된 기본 개념 이해
- II. 문자 입출력과 문자열 입출력 함수 이해
- III. 다양한 문자열 처리 함수 및 여러 문자열 처리 방법 이해

학습목차

- I. 제 1교시 문자와 문자열
- II. 제 2교시 문자와 문자열 입출력
- III. 제 3교시 문자열 관련 함수

1. 문자와 문자열

1. 문자와 문자열의 표현
2. 문자열의 초기화
3. 문자 포인터로 문자열 처리

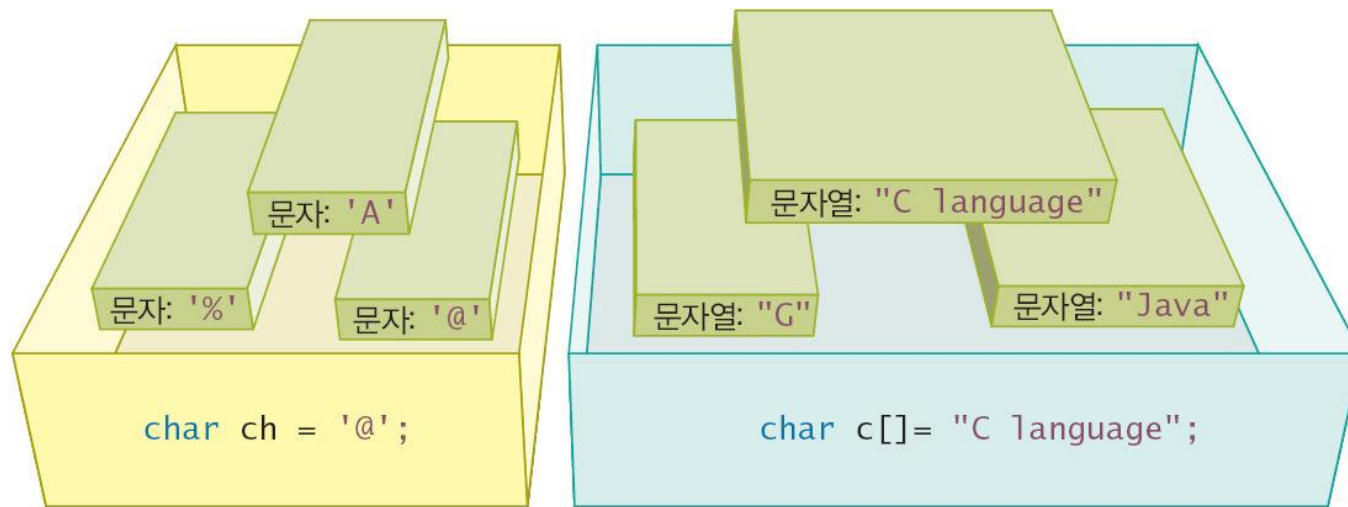
1. 문자와 문자열의 표현

❖ 문자

- 문자 하나를 작은따옴표로 둘러싸서 'A'와 같이 표기

❖ 문자열(string)

- 문자의 모임인 일련의 여러 문자
- 일련의 문자 앞 뒤로 큰따옴표로 둘러싸서 "java"로 표기



1. 문자와 문자열의 표현

◆ 자료형 char와 char[]

❖ 문자 저장

- char형 변수를 사용

❖ 문자열 저장

- 문자의 모임인 문자 배열을 사용
- 문자열은 문자열의 마지막을 의미하는 널 문자'\0'가 마지막에 저장되어야 함
 - 그러므로 배열 크기는 반드시 저장될 문자 수보다 1 커야 함

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

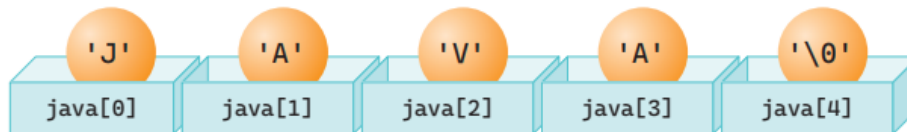
2. 문자열의 초기화

❖ 배열 선언 시 문자열 초기화 방법

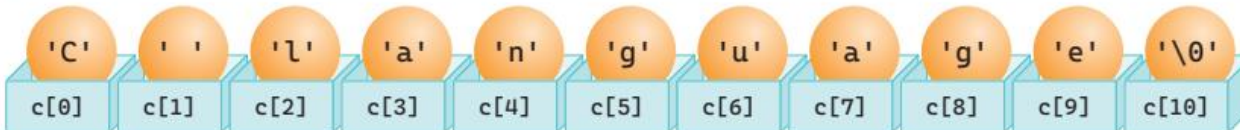
- 배열 초기화 시 배열 크기는 지정하지 않는 것이 더 수월
 - 만일 지정한다면 마지막 문자인 '\0'을 고려해 문자 수보다 1 더 크게 배열 크기를 정함

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

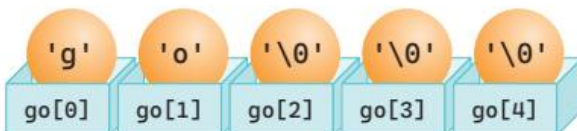
마지막에 '\0'을 빼면, 대입 시에는 문제가 없으나 출력 등에서 문제가 발생한다.



```
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
```



◆ 문자와 문자열 출력

❖ 문자열 출력

- 형식제어문자
%s로 출력

Prj01 01chstrprt.c 문자와 문자열 출력 난이도: ★

```

01 #include <stdio.h>
02
03 int main(void)
04 {
05     //문자 선언과 출력
06     char ch = 'A';
07     printf("%c %d\n", ch, ch);
08
09     //문자열 선언 방법1
10     char java[] = { 'J', 'A', 'V', 'A', '\0' };
11     printf("%s\n", java);
12
13     //문자열 선언 방법2
14     char py[] = "Python"; //배열크기를 생략하는 것이 간편
15     printf("%s\n", py);
16
17     //문자열 선언 방법3
18     char csharp[5] = "C#";
19     printf("%s\n", csharp);
20
21     //문자 배열에서 문자 출력
22     printf("%c %c\n", csharp[0], csharp[1]);
23
24     return 0;
25 }

```

%c로 문자 변수 ch 출력, %d를 사용하면 문자 코드 값 출력
 문자열 출력을 위해 배열이름과 형식제어문자 %s를 이용

A 65

JAVA

Python

C#

C #

[출처] 강환수 외, Perfect C 3판, 인피니티북스

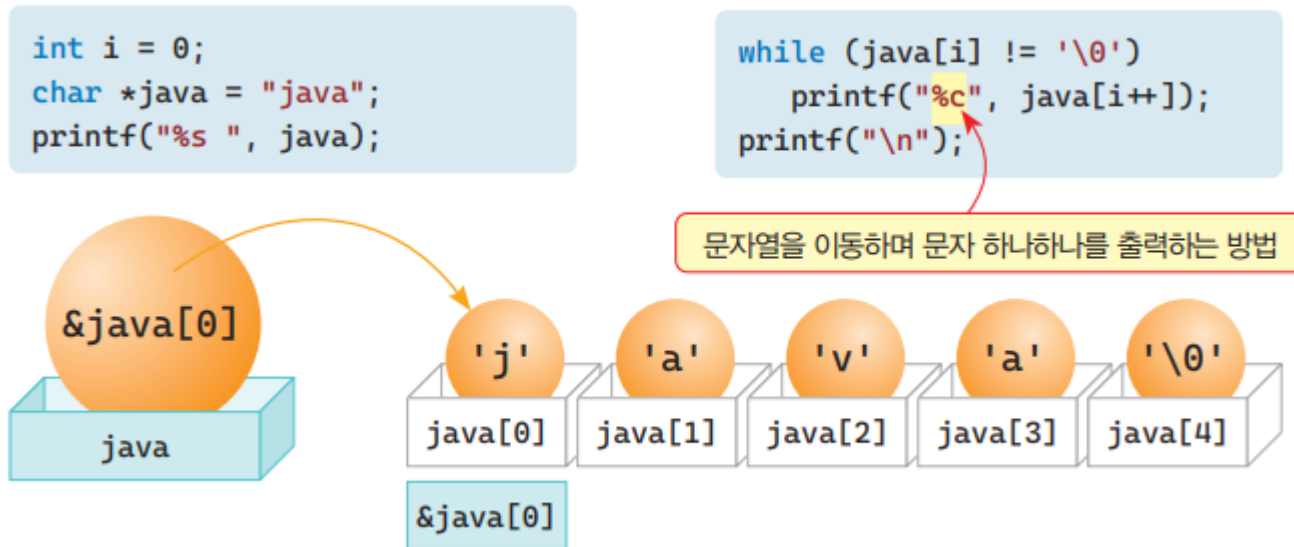
3. 문자 포인터로 문자열 처리

❖ 문자열을 문자 포인터로 처리하는 다른 방법

- 문자열 상수를 문자 포인터에 저장하는 방식

❖ 주의점

- 변수 java가 가리키는 문자열은 상수이므로 수정 불가능
 - 그러므로 변수 java를 사용하여 문자를 수정하거나 수정될 수 있는 함수의 인자로 사용하면 실행 오류가 발생



3. 문자 포인터로 문자열 처리

- 변수 java
 - 문자열 상수를 저장하는 문자 포인터
 - 계속 java를 사용해 문자를 참조
 - 사용 상 주의가 필요
변수 java가 가리키는 문자열은 상수이므로 수정 불가능
- 반복문을 이용
 - 문자가 '\0'이 아니면 문자를 출력하도록 하는 방식
✓ 출력할 문자열의 끝을 '\0' 문자로 검사하면 편리

Prj02 02charptr.c 문자 포인터로 문자열 저장과 출력 난이도: ★

```

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.

java java java

[출처] 강환수 외, Perfect C 3판, 인피니티북스

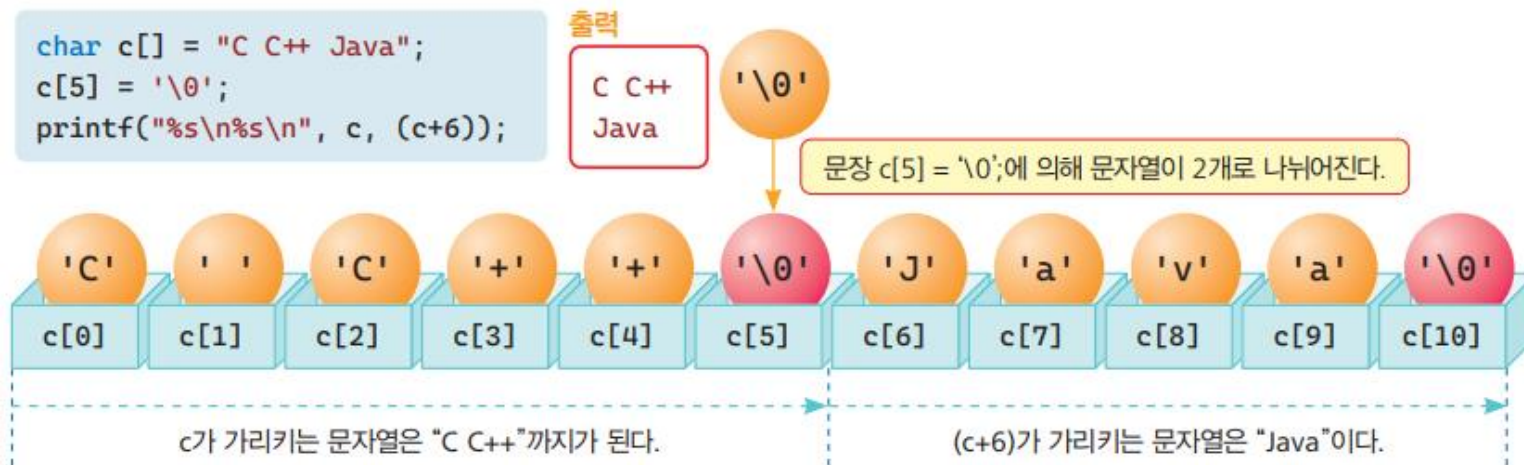


3. 문자 포인터로 문자열 처리

◆ 문자열 종료 의미 '\0'

❖ '\0' 문자에 의한 문자열 분리

- 문자열은 시작 문자부터 '\0' 문자가 나올 때까지 하나의 문자열로 처리
 - 그러므로 (c+6)로 문자열을 출력하면 "Java"가 출력



3. 문자 포인터로 문자열 처리

❖ 다양한 문자열 처리

- 문자 배열과 문자 포인터로 문자열 처리

실습예제 12-3	Prj03	03strsplit.c	문자열 중간에 '\0'을 삽입해 문자열 분리	난이도: ★
	01	#include <stdio.h>		
	02			
	03	int main(void)		
	04	{		
	05	char c[] = "C C++ Java";		
	06	printf("%s\n", c);		
	07	c[5] = '\0'; // NULL 문자에 의해 문자열 분리		
	08	printf("%s\n%s\n", c, (c + 6));		
	09			
	10	//문자 배열의 각 원소를 하나하나 출력하는 방법		
	11	c[5] = ' '; //널 문자를 빈 문자로 바꾸어 문자열 복원		
	12	char *p = c;		
	13	while (*p) /*(*p != '\0')도 가능		
	14	printf("%c", *p++);		
	15	printf("\n");		
	16			
	17	return 0;		
	18	}		
결과	C C++ Java C C++ Java C C++ Java			

c[5]에 '\0'를 저장하면 "C C++\0Java\0"이 저장되어 두 개의 문자열 "C C++"과 "Java"로 나뉨

int i = 0;
while (c[i])
printf("%c", c[i++]);
printf("\n");

i = 0;
while (*(c+i))
printf("%c", *(c + i++));
printf("\n");

1. 문자와 문자열

1교시 수업을 마치겠습니다.



II. 문자와 문자열 입출력

1. 문자 입출력 함수
2. 문자열 입출력 함수

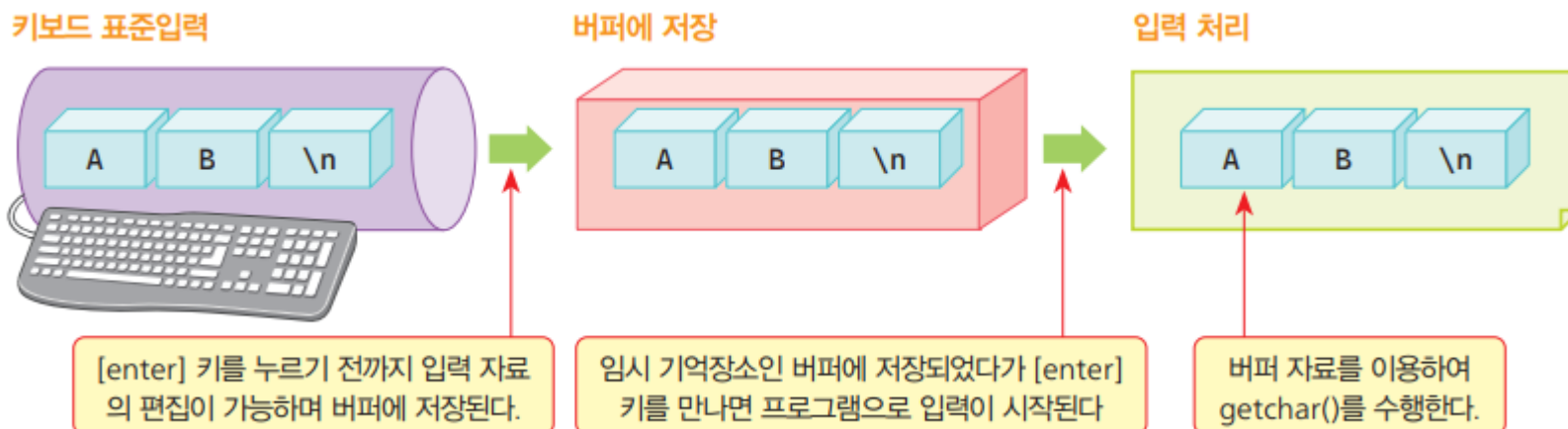
1. 문자 입출력 함수

❖ 함수 `getchar()`와 `putchar()`

- 함수 `getchar()`는 문자의 입력에 사용
- `putchar()`는 문자의 출력에 사용

❖ 버퍼링

- 함수 `getchar()`는 [enter]키를 입력해야 이전에 입력한 문자가 입력
- 이러한 작동 원인은 함수 `getchar()`가 라인 버퍼링(line buffering) 방식을 사용하기 때문

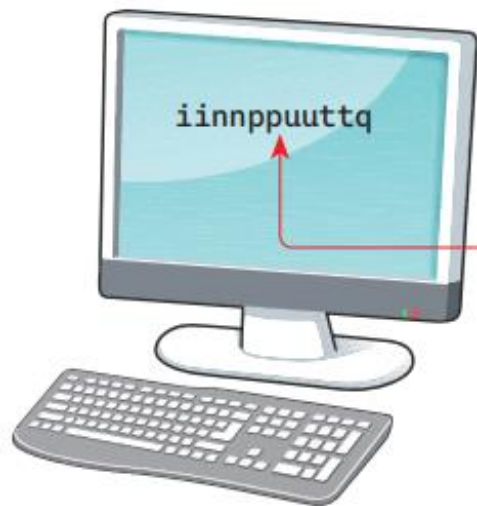


[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 문자 입출력 함수

◆ 함수 `_getche()`

- ❖ 버퍼를 사용하지 않고 문자 하나를 바로 입력하는 함수
 - 헤더 파일 `conio.h` 를 첨가
- ❖ 입력 문자가 'q'가 아니면
 - 함수 `putchar()`에 의하여 문자가 바로 출력
 - 입력된 문자로 보이고, 바로 `putchar()`에 의하여 출력



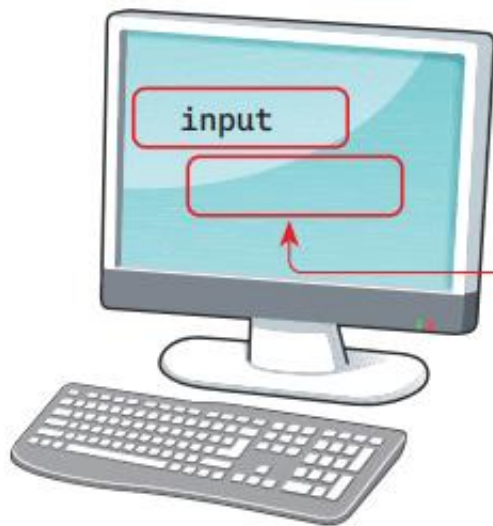
```
char ch;
while ( (ch = _getche()) != 'q')
    putchar(ch);
```

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

1. 문자 입출력 함수

◆ 함수 `_getch()`

- ❖ 함수 `getch()`도 버퍼를 사용하지 않는 문자 입력 함수
 - 헤더 파일 `conio.h` 를 첨가
 - 함수 `getch()`는 입력한 문자가 화면에 보이지 않는 특성
- ❖ 함수 `putchar(ch)`는 인자를 출력하는 함수



```
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.

1. 문자 입출력 함수

◆ 문자 입력 함수 비교

❖ 버퍼링 유무

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	

1. 문자 입출력 함수

◆ 다양한 문자 입출력 함수 이용

❖ 04getch.c

- 세 개의 while 문에서
- 각각 "getchar() q"
- "_getche() q"
- "_getch() q"를 입력

Prj04
04getch.c
함수 getchar(), _getche(), _getch()의 차이를 알아보는 예제
난이도: ★

```

01 #include <stdio.h>
02 #include <conio.h>
03
04 int main(void)
05 {
06     char ch;
07
08     printf("문자를 계속 입력하고 Enter를 누르면 >>\n");
09     while ((ch = getchar()) != 'q')
10         putchar(ch); // stdio.h
11
12     printf("\n문자를 누른 것이 보이고, _putch에 의해 입력문자 출력 >>\n");
13     while ((ch = _getche()) != 'q')
14         _putch(ch); // conio.h
15
16     printf("\n문자를 누른 것이 안 보이고, _putch에 의해 입력문자 출력 >>\n");
17     while ((ch = _getch()) != 'q')
18         _putch(ch); // conio.h
19     printf("\n");
20
21     return 0;
22 }
    
```

함수 getchar()로 입력한 문자를 변수 ch에 저장하여 'q'가 아니면 반복문체인 putchar(ch)를 실행, 변수 ch에 'q'이면 반복이 종료되므로 출력이 안됨

함수 _getche()로 입력한 문자를 변수 ch에 저장하여 'q'가 아니면 반복문체인 _putch(ch)를 실행, 변수 ch에 'q'이면 반복 종료이므로 출력이 안됨

함수 _getch()로 입력한 문자를 변수 ch에 저장하여 'q'가 아니면 반복문체인 _putch(ch)를 실행, 변수 ch에 'q'이면 반복 종료이므로 출력이 안됨. 함수 _getch()는 입력한 문자마다 바로 처리하며, 입력한 문자도 보이지 않음

문자를 계속 입력하고 Enter를 누르면 >>

pythonq ← 마지막에 입력한 문자 q가 입력 종료를 처리

python

문자를 누른 것이 보이고, _putch에 의해 입력문자 출력 >>

ppyytthhoonnq ← 마지막에 입력한 문자 q가 입력 종료를 처리

문자를 누른 것이 안 보이고, _putch에 의해 입력문자 출력 >>

python ← 입력을 pythonq와 같이 마지막에 q를 입력해야 종료되며, 마지막에 입력한 q는 출력되지 않음

2. 문자열 입출력 함수

◆ gets()와 puts()

❖ 함수 gets()

- 한 행의 문자열 입력에 유용한 함수

❖ 함수 puts()

- 한 행에 문자열을 출력하는 함수
- 함수 gets(), puts(), gets_s()를 사용하려면 헤더 파일 stdio.h를 삽입

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);
```

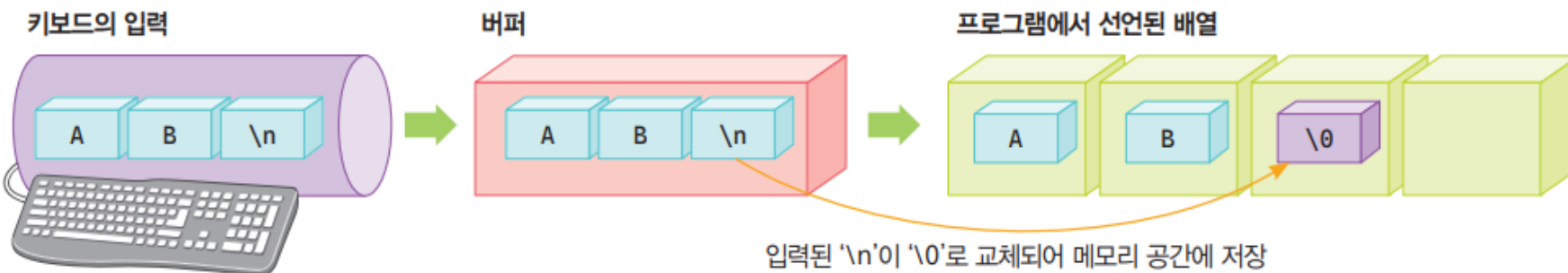
- 인자인 문자열 str에서 마지막 '\0' 문자를 개행 문자인 '\n'로 대체하여 출력한다.
- 함수 puts()는 일반적으로 0인 정수를 반환하는데, 오류가 발생하면 EOF를 반환한다.

2. 문자열 입출력 함수

◆ 함수 gets()

❖ 버퍼링

- 함수 gets()는 [enter] 키를 누를 때까지 한 행을 버퍼에 저장한 후 입력 처리
- 마지막에 입력된 '\n'이 '\0'로 교체되어 인자인 배열에 저장
- 함수 gets()의 인자로써 입력된 문자열을 저장할 수 있는 충분한 공간의 문자 배열을 사용
 - 그렇지 않으면 실행 오류가 발생

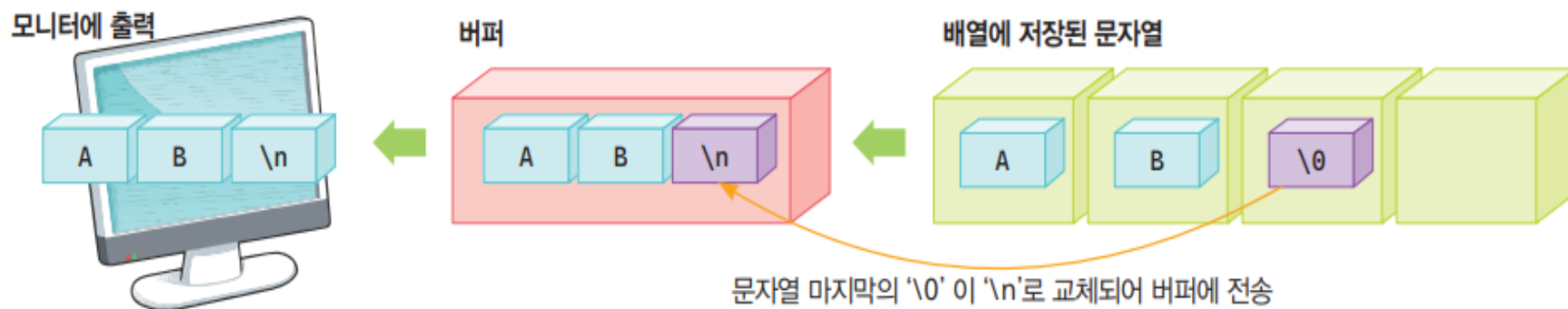


2. 문자열 입출력 함수

◆ 함수 puts()

❖ 문자열을 한 줄에 출력하는데 유용하게 사용

- 함수 puts()는 함수 gets()와 반대로 문자열의 마지막에 저장된 '\0'를 '\n'로 교체하여 버퍼에 전송
- 버퍼의 내용이 모니터에 출력되면 문자열이 한 행에 출력
 - 그러므로 문자열의 한 행 출력은 함수 puts()가 효과적



```

Prj05      05strinput.c      함수 scanf()와 printf(), gets()와 puts()의 문자열 입출력      난이도: ★
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          puts(line);
17      printf("\n");
18
19      while (gets_s(line, 101))
20          puts(line);
21      printf("\n");
22
23      return 0;
24  }

```

학번 이름 학과 입력 >>
 20222007 컴퓨터정보공학과 나윤희
 출력: 20222007 컴퓨터정보공학과 나윤희
 한 행에 학번 이름 학과 입력한 후 [enter]를 누르고 새로운 행에서 (ctrl + Z)를 누르십시오.

20222007 컴퓨터정보공학과 나윤희
 20222007 컴퓨터정보공학과 나윤희
 ^Z
 새로운 행에서 ctrl + Z 입력하면 while 반복 종료

20222007 컴퓨터정보공학과 나윤희
 20222007 컴퓨터정보공학과 나윤희
 ^Z

◆ gets(), puts()

- ❖ 함수 puts()
 - ❖ 함수 scanf()에서 제어문자 %s
 - 문자열을 입력
- ❖ 함수 gets()와 gets_s()를 사용
 - 여러 줄을 입력
- ❖ while문을 사용
 - 연속된 여러 행을 입력 받아 바로 행 별로 출력
 - 다음 반복을 종료
 - 새로운 행 처음에 (ctrl + Z)를 입력
- ❖ 함수 printf()와 scanf()
 - 다양한 입출력에 적합
- ❖ 함수 puts()와 gets()
 - 처리 속도가 빠르다는 장점

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

공백으로 구분해 snum, dept, name에 저장

한 행 모두가 배열 line에 저장

새로운 행에서 ctrl + Z 입력하면 while 반복 종료

Ⅱ. 문자와 문자열 입출력

2교시 수업을 마치겠습니다.



Ⅲ. 문자열 관련 함수

1. 기본적인 문자열 처리 함수
2. 다양한 문자열 관련 함수
3. 여러 개의 문자열 처리

1. 기본적인 문자열 처리 함수

- ❖ 헤더파일 string.h에 함수원형으로 선언된 라이브러리 함수로 제공
 - 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리 함수
- ❖ 함수에서 사용되는 자료형: 64비트 윈도우 시스템인 경우
 - size_t : 비부호 정수 long long형(unsigned __int64)
 - void * : 아직 정해지지 않은 다양한 포인터를 의미

표 12-2 문자열 배열에 관한 다양한 함수

함수원형	설명
size_t strlen(const char *str)	포인터 src 위치에서부터 널 문자를 제외한 문자열의 길이 반환
void *memcpy(void *dest, const void *src, size_t n)	포인터 src 위치에서 dest에 n 바이트를 복사한 후 dest 위치 반환
void *memchr(const void *str, int c, size_t n)	메모리 str에서 n 바이트까지 문자 c를 찾아 그 위치를 반환
int memcmp(const void *str1, const void *str2, size_t n)	메모리 str1과 str2를 첫 n 바이트를 비교 검색하여 같으면 0, 다르면 음수 또는 양수 반환
void *memmove(void *dest, const void *src, size_t n)	포인터 src 위치에서 dest에 n 바이트를 복사한 후 dest 위치 반환
void *memset(void *str, int c, size_t n)	포인터 src 위치에서부터 n 바이트까지 문자 c를 지정한 후 src 위치 반환

1. 기본적인 문자열 처리 함수

❖ 함수 strcmp()

- 헤더 파일 string.h에 함수원형이 정의

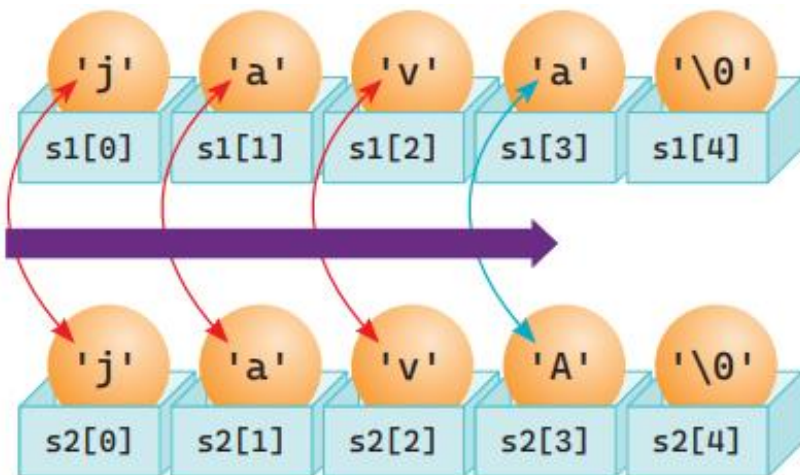
문자열 비교 함수: 헤더파일 string.h 삽입

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

두 인자인 문자열에서 같은 위치의 문자를 앞에서부터 다룰 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 다룰 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

```

01 #include <stdio.h>
02 #include <string.h>
03
04 int main(void)
05 {
06     char src[20] = "C Python";
07     char dst[20];
08
09     printf("%s\n", src);
10     printf("%zu\n", strlen(src));
11     memcpy(dst, src, strlen(src) + 1);
12     printf("%s\n", dst);
13     memcpy(dst, "안녕하세요!", strlen("안녕하세요!") + 1);
14     printf("%s\n\n", dst);
15
16     char* s1 = "C lang";
17     char* s2 = "C ";
18     printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
19     s1 = "C lang";
20     s2 = "C ";
21     printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
22     printf("strcmp(%s, %s) = %d\n", s2, s1, strcmp(s2, s1));
23     printf("strncmp(%s, %s, %d) = %d\n", s1, s2, 2, strncmp(s1, s2, 2));
24
25     return 0;
26 }

```

문자열 src에서 포인터 위치 dst에 strlen(src) + 1 수만큼 복사하여 문자열의 마지막이 널이 되도록

포인터 위치 dst에 문자열 상수 "안녕하세요!"를 복사하기 위해서는 (strlen("안녕하세요!") + 1)만큼 복사해야 문자열의 마지막이 널이 됨

비교 세 번째 문자가 각각 'l'과 '0'이므로 strcmp(s1, s2)는 양수임

비교 세 번째 문자가 각각 '0'과 'l'이므로 strcmp(s1, s2)는 음수임

C Python

8 ← 문자열 "C Python"의 길이는 8

C Python

안녕하세요!

strcmp(C lang, C lang) = 0

strcmp(C lang, C) = 1

strcmp(C , C lang) = -1

strncmp(C lang, C , 2) = 0

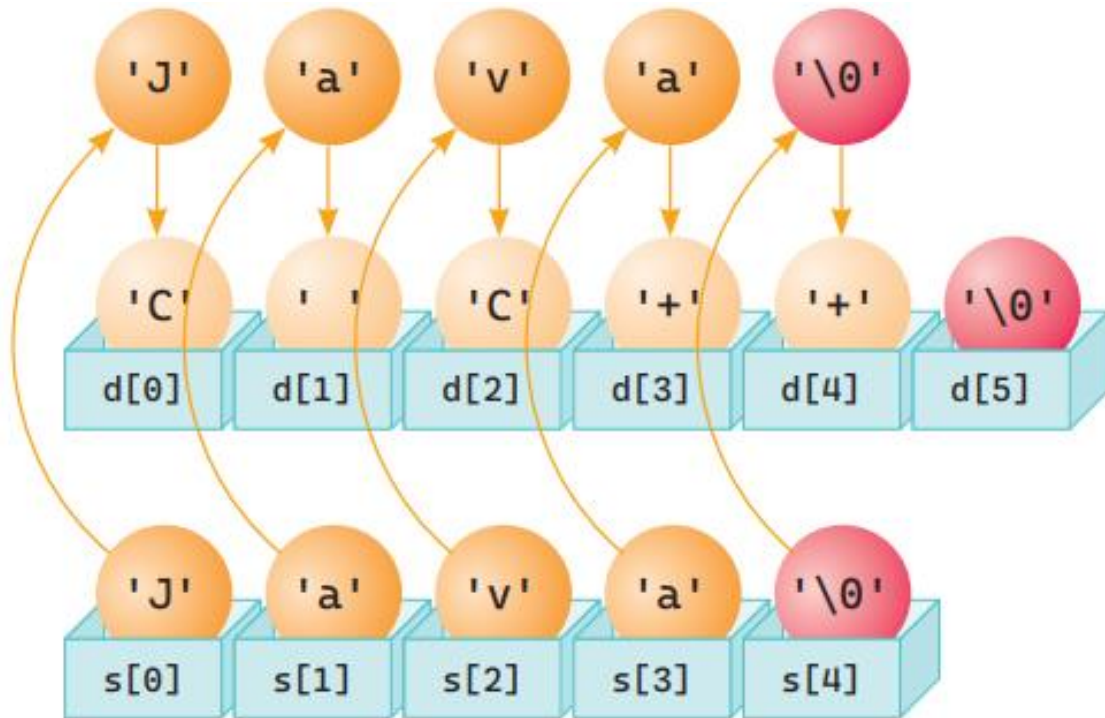
[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 기본적인 문자열 처리 함수

◆ 문자열 복사

❖ 함수 strcpy()

- 문자열을 복사하는 함수



결과는 d에도 "java"가 저장된다.

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

1. 기본적인 문자열 처리 함수

◆ 문자열 복사

❖ 함수 strncpy()

- 복사되는 최대 문자 수를 마지막 인자 maxn으로 지정하는 함수

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);
```

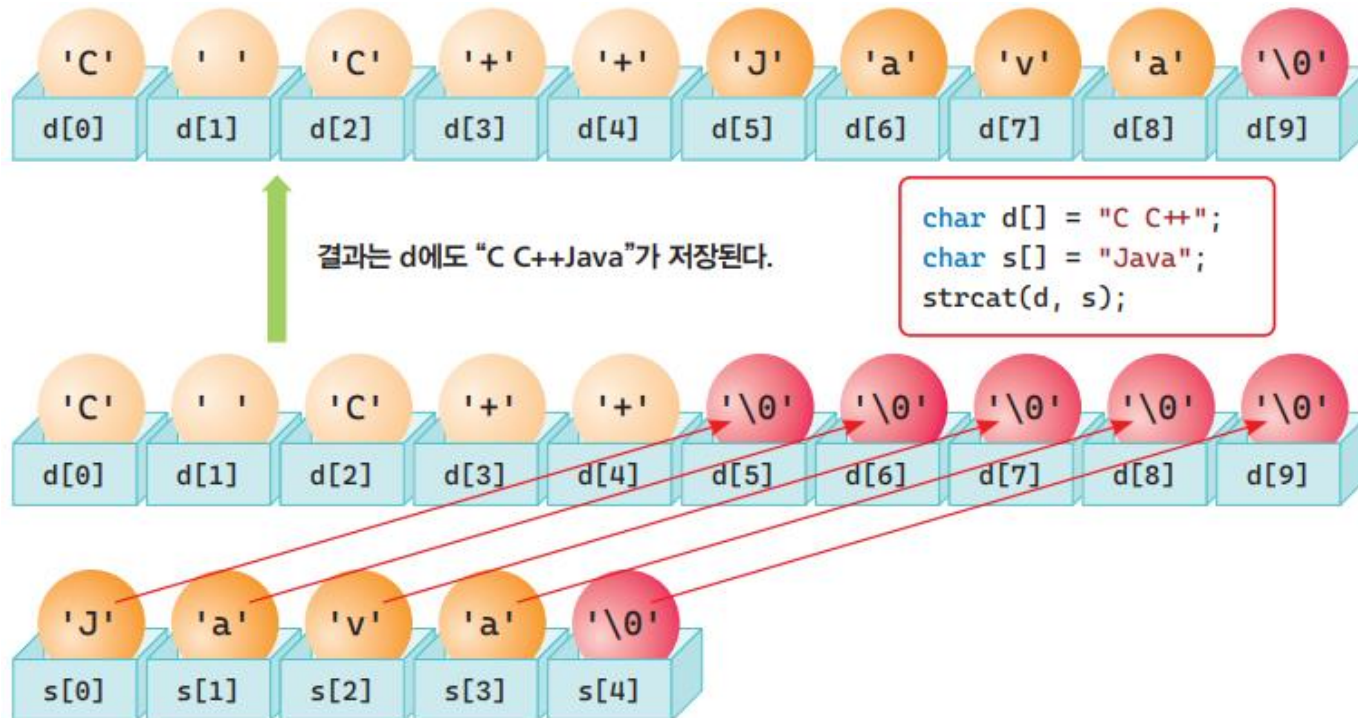
- 두 번째 인자인 sizedest는 정수형으로 dset 의 크기를 입력한다.
- 반환형 errno_t 는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수 strcpy_s()와 strncpy_s()의 사용을 권장한다.

1. 기본적인 문자열 처리 함수

◆ 문자열 연결

❖ 함수 strcat()

- 앞 문자열에 뒤 문자열의 널 문자까지 연결하여, 앞의 문자열 주소를 반환하는 함수



1. 기본적인 문자열 처리 함수

◆ 문자열 연결

❖ 함수 strcat()

- 전달 인자의 마지막에 연결되는 문자의 수를 지정하여 그 이상은 연결되지 않도록
 - 여기서 지정하는 문자 수는 널 문자를 제외한 수

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     char dest[80] = "Java";
08     char source[80] = "C is a language.";
09
10     printf("%s\n", strcpy(dest, source));
11     //printf("%d\n", strcpy_s(dest, 80, source));
12     //printf("%s\n", dest);
13     printf("%s\n", strcpy(dest, "C#", 2));
14
15     printf("%s\n\n", strncpy(dest, "C#", 3));
16     //printf("%d\n", strncpy_s(dest, 80, "C#", 3));
17     //printf("%s\n", dest);
18
19     char data[80] = "C";
20
21     printf("%s\n", strcat(data, " is "));
22     //printf("%d\n", strcat_s(data, 80, " is "));
23     //printf("%s\n", data);
24     printf("%s\n", strncat(data, "a java", 2));
25     //printf("%d\n", strncat_s(data, 80, "a proce", 2));
26     //printf("%s\n", data);
27     printf("%s\n", strcat(data, "procedural "));
28     printf("%s\n", strcat(data, "language."));
29
30     return 0;
31 }

```

비주얼 스튜디오에서 추천하는 strcpy_s() 사용 방법

"C#"을 dest에 2바이트인 "C#"까지만 복사

"C#"을 dest에 3바이트인 "C#\0"까지 복사

비주얼 스튜디오에서 추천하는 strncpy_s() 사용 방법

2개 문자인 "a"까지만 뒤에 연결되어 문자열은 "C is a"가 됨

```

C is a language.
C#is a language.
C#
C is
C is a
C is a procedural
C is a procedural language.

```

\0 이전인 C#만 출력

III. 문자열 관련 함수



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

```


2. 다양한 문자열 관련 함수

◆ 문자열 분리

❖ 함수 strtok()

- 문자열에서 구분자(delimiter)인 문자를 여러 개 지정하여 토큰을 추출하는 함수

문자열 분리 함수: string.h

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

앞 문자열 str에서 뒤 문자열 delim을 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.

2. 다양한 문자열 관련 함수

◆ 문자열 분리

- ❖ 문자열 "C and C++\t languages are best!"
 - 구분자를 공백문자 하나인 " "로 지정
 - 토큰을 분리하면 C, and, C++\t, languages, are, best! 총 6개의 토큰이
 - 추출공백문자 분리자를 이용하여 토큰을 분리
 - 구분자에 더 많은 문자를 삽입
 - 분리된 토큰이 많아지거나 하나 하나의 토큰에서 구분자가 사라짐

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

- 문장 ptoken = strtok(str, delimiter);
 - 첫 토큰을 추출
 - 결과를 저장한 ptoken
 - NULL이면 더 이상 분리할 토큰이 없는 경우
 - 계속 토큰을 추출하려면 (ptoken != NULL) 로 검사
 - while 반복으로 추출된 토큰이 있는지 검사
 - strtok(NULL, delimiter) 를 호출
 - NULL을 첫 번째 인자
 - 그 다음 토큰을 반환

Prj12
08strtok.c
문자열에서 지정한 분리자를 사용해 문자열 토큰을 분리
난이도: ★★

```

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 }

```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

```

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

```

```
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
```

2. 다양한 문자열 관련 함수

❖ 문자열의 길이, 대소문자 변환, 위치 검색 등

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

문자열 str에서 null 문자를 제외한 문자열의 길이를 반환한다.

```
char * strlwr(char * str);
```

```
errno_t _strlwr_s(char * str, size_t strsize); //Visual C++ 권장 함수
```

문자열 str을 모두 소문자로 변환하고 변환한 문자열을 반환한다. errno_t는 정수형의 오류 번호이며, size_t도 정수형으로 strsize는 str의 길이

```
char * strupr(char * str);
```

```
errno_t _strupr_s(char * str, size_t strsize); //Visual C++ 권장 함수
```

문자열 str을 모두 대문자로 변환하고 변환한 문자열을 반환한다. errno_t는 정수형의 오류 번호이며, 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 반환한다.

```
char * strstr(const char * str, const char * strsearch);
```

앞의 문자열 str에서 뒤 문자열 strsearch이 나타나는 처음 위치를 찾아 그 주소값을 반환한다. 만일 찾지 못하면 NULL 포인터를 반환한다.

```
char * strchr(const char * str, char ch);
```

앞의 문자열 str에서 뒤 문자 ch가 나타나는 처음 위치를 찾아 그 주소값을 반환한다. 만일 찾지 못하면 NULL 포인터를 반환한다.

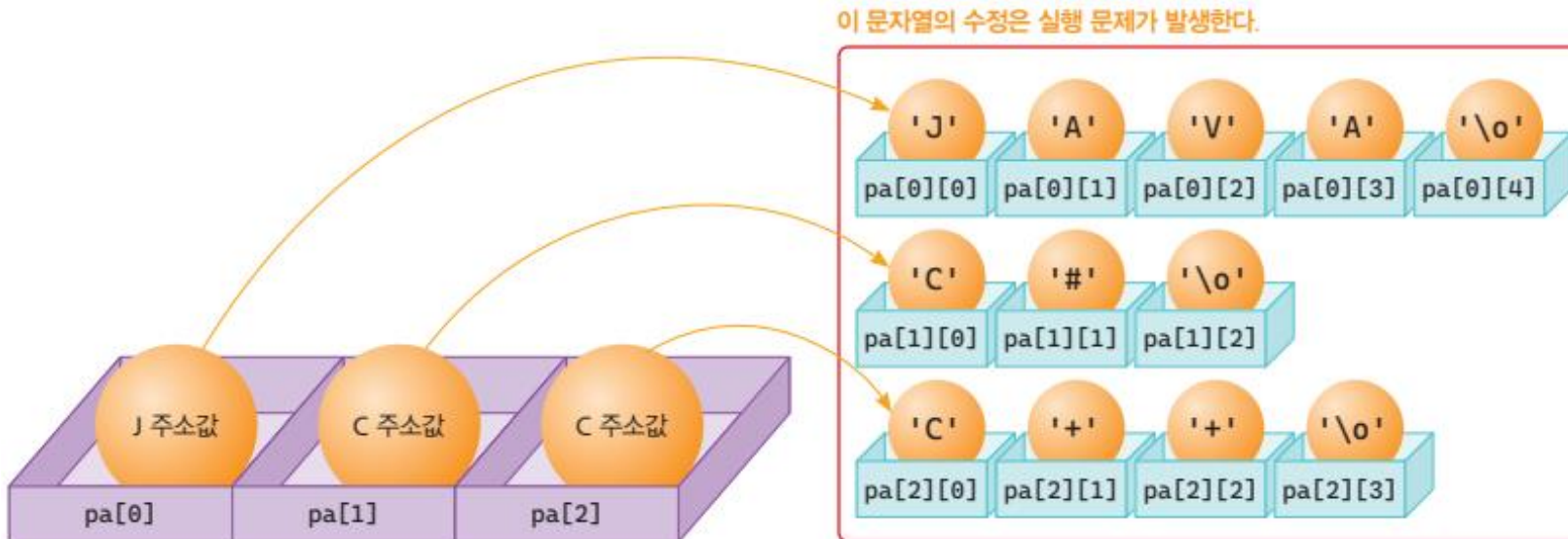
3. 여러 개의 문자열 처리

❖ 문자 포인터 배열

- 하나의 문자 포인터가 하나의 문자열을 참조할 수 있으므로 문자 포인터 배열은 여러 개의 문자열을 참조

```
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.



3. 여러 개의 문자열 처리

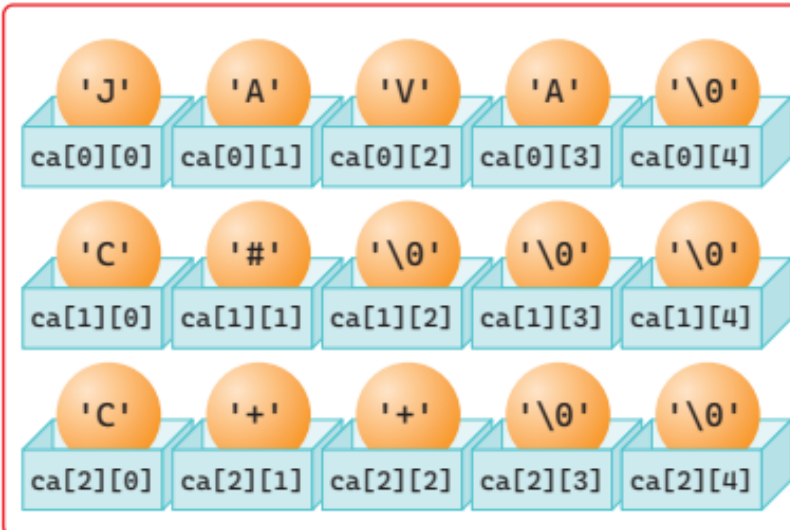
❖ 이차원 문자 배열

- 배열 선언에서 이차원 배열의 열 크기는 문자열 중에서 가장 긴 문자열의 길이보다 1 크게 지정

```
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 1 크게 지정한다.

이 문자열의 수정될 수 있다.



장 단점

- 이차원 배열에서 모든 열 수가 동일하게 메모리에 할당
 - 열의 길이가 서로 다른 경우에는 'w' 문자가 들어가 낭비
- 문자열을 수정 가능
 - ca[0][2] = 'v';와 같이 원하는 문자 수정이 가능


```

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

결과

```

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

```


3. 여러 개의 문자열 처리 : 명령행 인자

❖ main(int argc, char *argv[])

- 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법이 명령행 인자(command line arguments)를 사용하는 방법

❖ 프로그램에서 명령행 인자를 받으려면

- 두 개의 인자 argc와 argv를 (int argc, char * argv[])로 기술
 - 매개변수 argc : 명령행에서 입력한 문자열의 수
 - argv[] : 명령행에서 입력한 문자열을 전달 받는 문자 포인터 배열
 - 실행 프로그램 이름도 하나의 명령행 인자에 포함

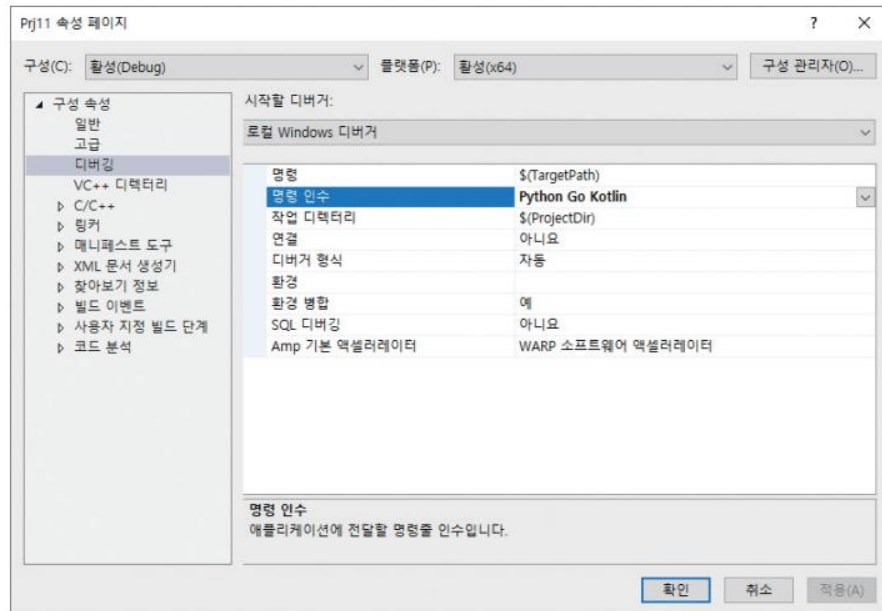
도스 창에서 프로그램이 저장된 폴더의 경로와 >을 합쳐 프롬프트(prompt)라 하고, 명령어 프롬프트가 나타나는 이 줄에 명령어를 입력할 수 있으며, 이 줄을 명령행(command line)이라 한다.

3. 여러 개의 문자열 처리 : 명령행 인자

[출처] 강환수 외, Perfect C 3판, 인피니티북스

◆ 비주얼 스튜디오

- ❖ 메뉴 [프로젝트/{프로젝트이름} 속성...]를 누르거나
 - 단축 키 Alt+F7을 눌러 다음 대화상자에서 설정
 - 대화상자 [{프로젝트이름} 속성 페이지]의 항목 [디버깅]을 누르고
 - 中间的 [명령 인수]의 입력 상자에 인자를 기술
 - 이 입력 상자에는 실행파일 이름 뒤의 옵션만을 기술



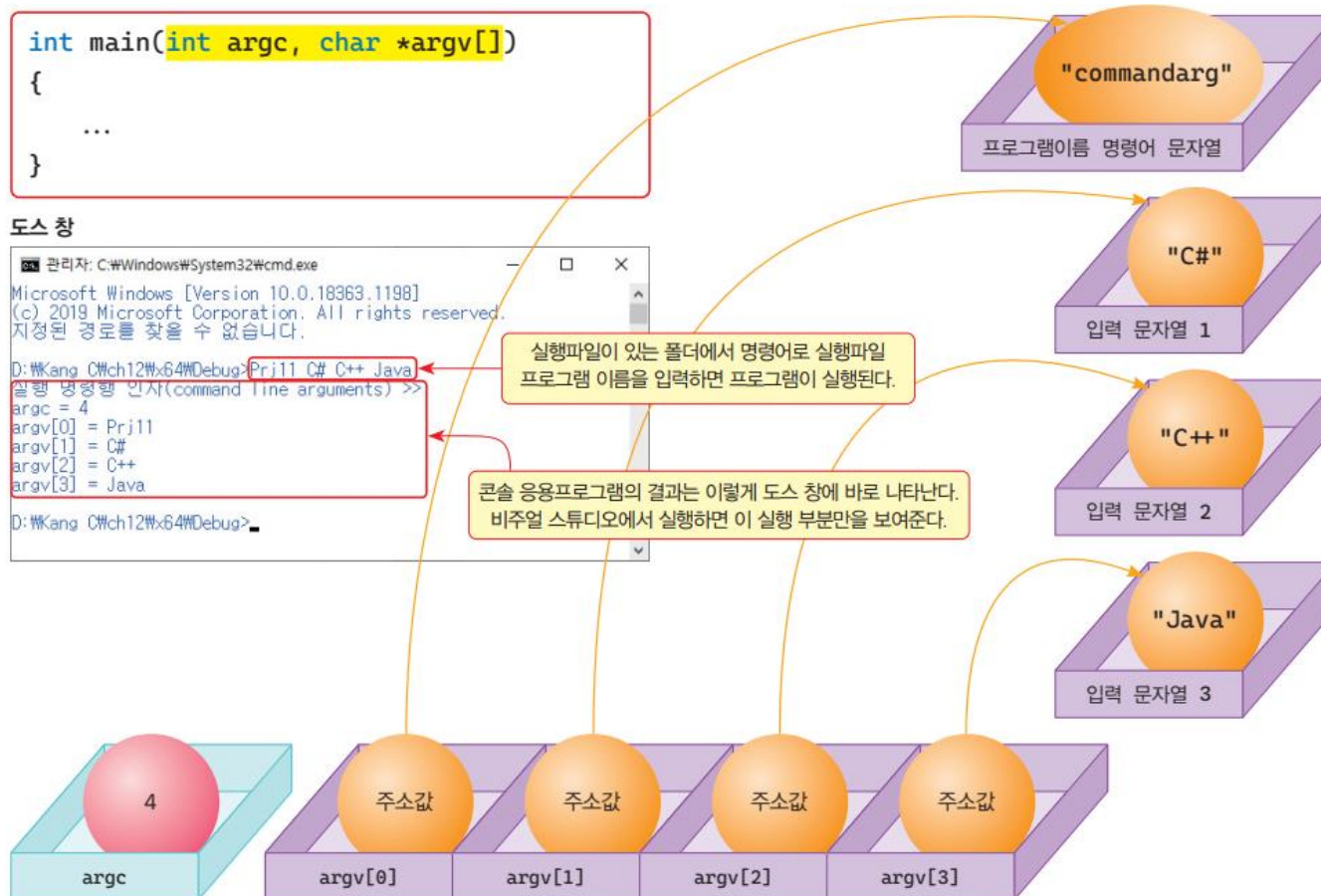
3. 여러 개의 문자열 처리 : 명령행 인자

실습예제 12-11	Prj11	11cmdarg.c	명령행 인자 출력	난이도: ★★
	<pre> 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 }</pre> <div data-bbox="950 511 1694 596"> <p>argc(argument count)에 인자의 수가, argv(argument variables)에는 인자인 여러 개의 문자열의 포인터가 저장된 배열이 전달</p> </div>			
결과	<pre> 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</pre> <div data-bbox="1123 1139 1638 1225"> <p>비주얼 스튜디오에서 실행하면 전체 경로를 포함한 실행파일의 이름이 첫 번째 인자로 표시된다.</p> </div>			

3. 여러 개의 문자열 처리 : 명령행 인자

❖ Wch12Wx64WDebug 폴더

- 비주얼 스튜디오의 실행파일이 생성된 폴더에서 명령 프롬프트를 실행



Ⅲ. 문자열 관련 함수

3교시 수업을 마치겠습니다.

