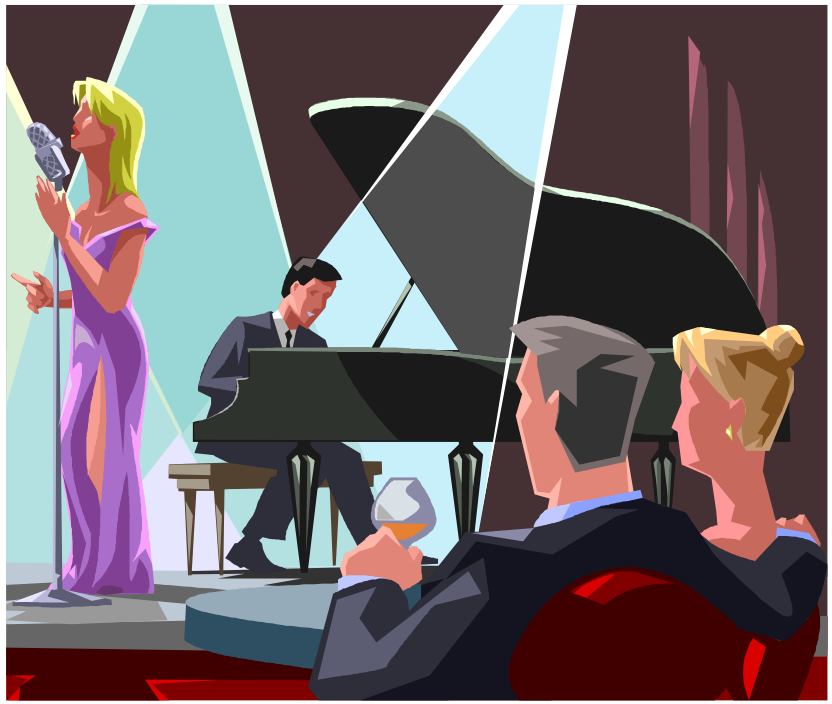
누구나 즐기는 C언어 콘서트

제10장 문자와 문자열



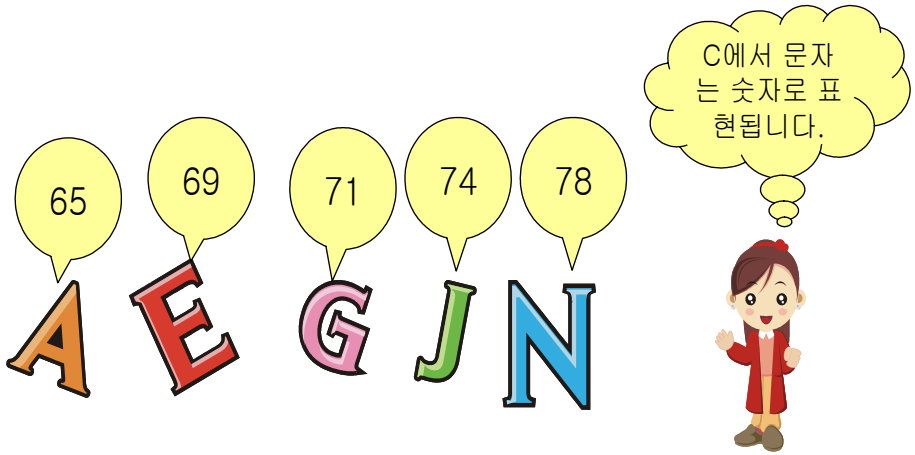
이번 장에서 학습할 내용

- 문자 표현 방법
- 문자열 표현 방법
- 문자열이란 무엇인가?
- 문자열의 입출력
- 문자처리 라이브러리 함수
- 표준입출력 라이브러리 함수

문자와 문자열 처리 방법에 대하여 살펴볼 것이다.

문자표현방법

- 컴퓨터에서는 각각의 문자에 숫자코드를 붙여서 표시한다.
- 아스키코드(ASCII code): 표준적인 8비트 문자코드
  - 0에서 127까지의 숫자를 이용하여 문자표현
- 유니코드(unicode): 표준적인 16비트 문자코드
  - 전세계의 모든 문자를 일관되게 표현하고 다룰 수 있도록 설계



문자 변수와 문자 상수

문자변수

문자상수

```
// 문자 상수
#include <stdio.h>

int main(void)
{
    char code1 = 'A';
    char code2 = 65;

    printf("code1=%c, code2=%c\n", code1,code2);
    return 0;
}
```

```
code1=A, code2=A
```

아스키 코드 출력

```
// 아스키 코드 출력
#include <stdio.h>
int main(void)
{
    unsigned char code;

    for(code = 32; code < 128; code++)
    {
        printf("아스키 코드 %d은 %c입니다.\n", code, code);
    }
    return 0;
}
```

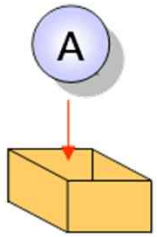
```
아스키 코드 32은 입니다.
아스키 코드 33은 !입니다.
...
아스키 코드 97은 a입니다.
아스키 코드 98은 b입니다.
아스키 코드 127은  입니다.
```

중간 점검

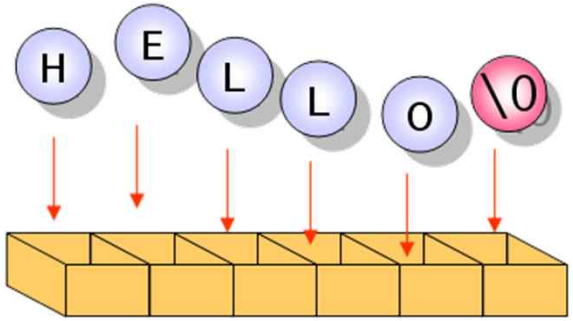
1. 컴퓨터에서는 문자를 어떻게 나타내는가?
2. C에서 문자를 가장 잘 표현할 수 있는 자료형은 무엇인가?
3. 컴파일러가 'A'를 만나면 어떻게 처리하는가?

문자열 표현 방법

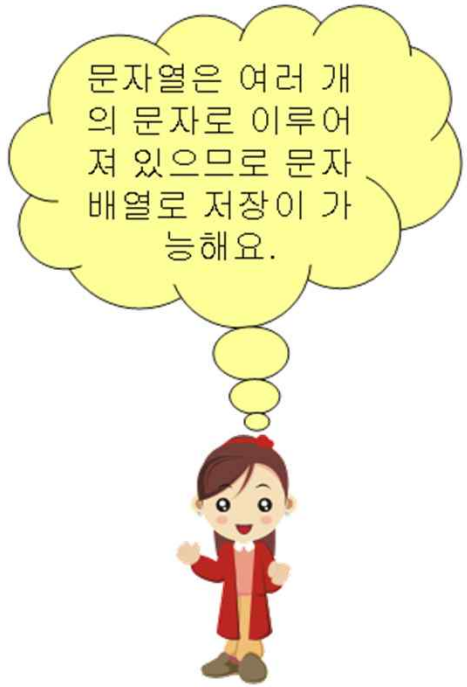
- *문자열(string):* 문자들이 여러 개 모인것
  - "A"
  - "Hello World!"



하나의 문자는 char형 변수로 저장



문자열은 char형 배열로 저장



문자열 상수와 변수

- *문자열 상수: 변경되지 않는 문자열을 저장*
  - (예) "Hello World"

- *문자열 변수: 변경되는 문자열을 저장*
  - (예) char str[100];

NULL 문자

- NULL 문자: 문자열의 끝을 나타낸다.

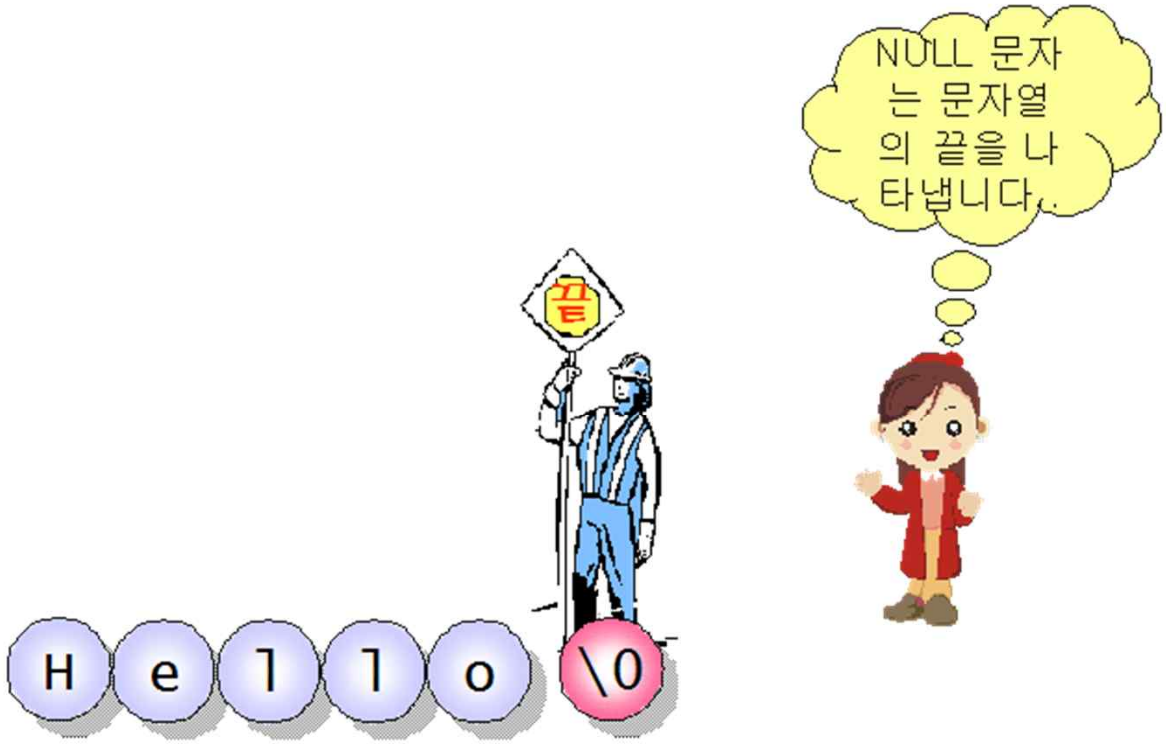


그림 10.3 문자열의 끝에는 항상 NULL 문자가 들어가야 한다.

왜 NULL 문자 가 필요한가?

- 문자열은 어디서 종료되는지 알수가 없으므로 표시를 해주어야 한다.

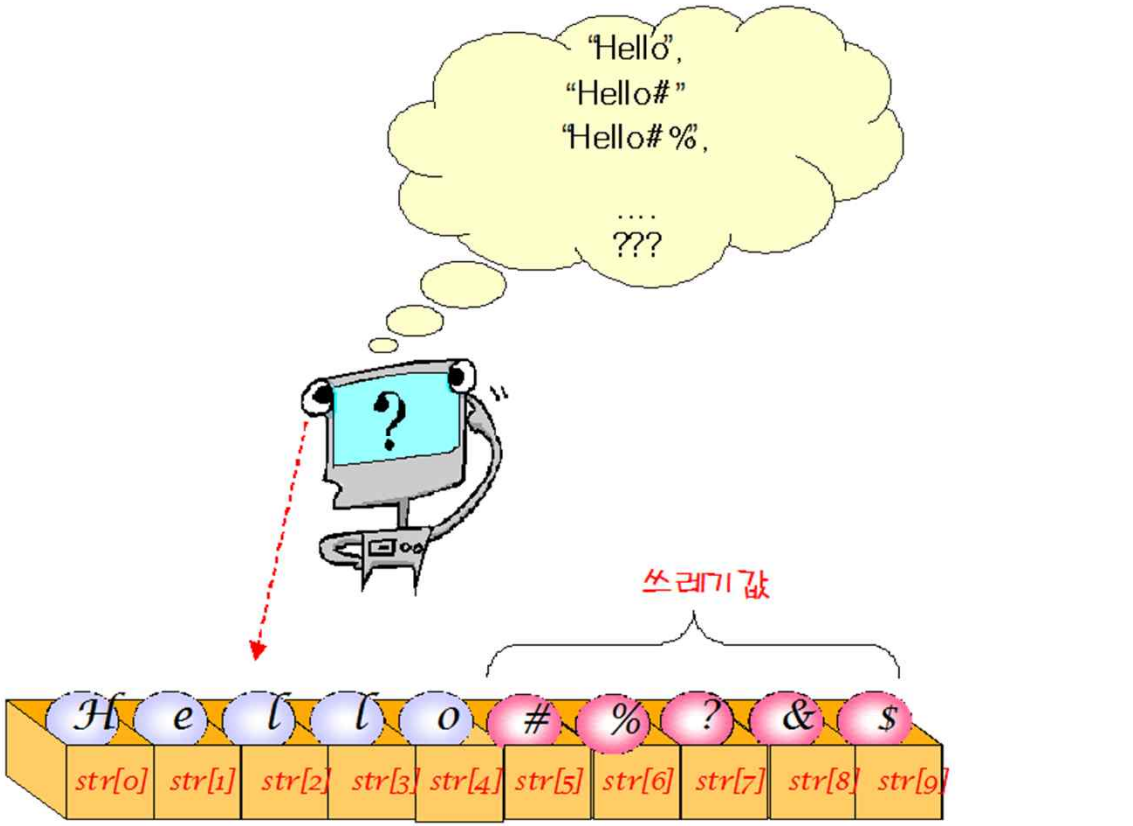


그림 10.4 NULL 문자의 필요성: 정상적인 데이터와 쓰레기값을 분리하기 위해서이다.

문자 배열의 초기화

- 문자 배열 원소들을 중괄호 안에 넣어주는 방법
  - char str[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };
- 문자열 상수를 사용하여 초기화하는 방법
  - char str[6] = "Hello";·

  만약 배열을 크기를 지정하지 않으면 컴파일러가 자동으로 배열의 크기를 초기화값에 맞추어 설정
  - char str[] = "C Bible"; // 배열의 크기는 7이 된다.

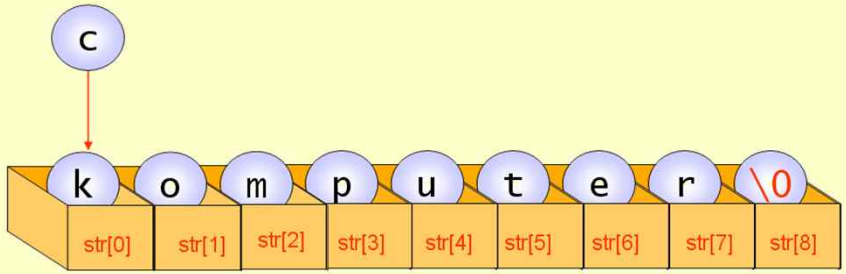
예제 #1

```
#include <stdio.h>

int main(void)
{
    char str1[6] = "Seoul";
    char str2[3] = { 'i', 's' };
    char str3[] = "the capital city of Korea.";

    printf("%s %s %s\n", str1, str2, str3);
}
```

```
Seoul is the capital city of Korea.
```

예제 #2

```
#include <stdio.h>

int main(void)
{
    char str[] = "komputer";

    printf("%s ", str);
    str[0] = 'c';
    printf("%s ", str);

    return 0
}
```



```
k o m p u t e r
c o m p u t e r
```

예제 #3

```
// 문자열의 길이를 구하는 프로그램
#include <stdio.h>

int main(void)
{
    char str[30] = "A barking dog never bites";
    int i = 0;

    while(str[i] != 0)
        i++;

    printf("문자열\"%s\"의 길이는 %d입니다.\n", str, i);

    return 0;
}
```

문자열 A barking dog never bites의 길이는 25입니다.

중간 점검

1. C에서 문자열은 어떻게 정의되는가?
2. 문자열에서 NULL 문자의 역할은 무엇인가?
3. NULL 문자의 아스키 코드 값은 얼마인가?
4. NULL 문자로 끝나지 않는 문자열을 출력하면 어떻게 되는가?
5. B, 'B', "B"의 차이점을 설명하라.
6. 변경 가능한 문자열은 어디에 저장되는가?
7. 문자열의 크기보다 문자 배열의 크기를 하나 더 크게 하는 이유는 무엇인가?
8. 문자 배열을 문자열로 초기화하는 방법을 아는 대로 설명하라.

문자 입출력 라이브러리

입출력 함수	설명
int getchar(void)	하나의 문자를 읽어서 반환한다.
void putchar(int c)	변수 c에 저장된 문자를 출력한다.
int getch(void)	하나의 문자를 읽어서 반환한다(버퍼를 사용하지 않음).
void putch(int c)	변수 c에 저장된 문자를 출력한다(버퍼를 사용하지 않음).
scanf("%c", &c)	하나의 문자를 읽어서 변수 c에 저장한다.
printf("%c", c);	변수 c에 저장된 문자를 출력한다.

getchar(), putchar()

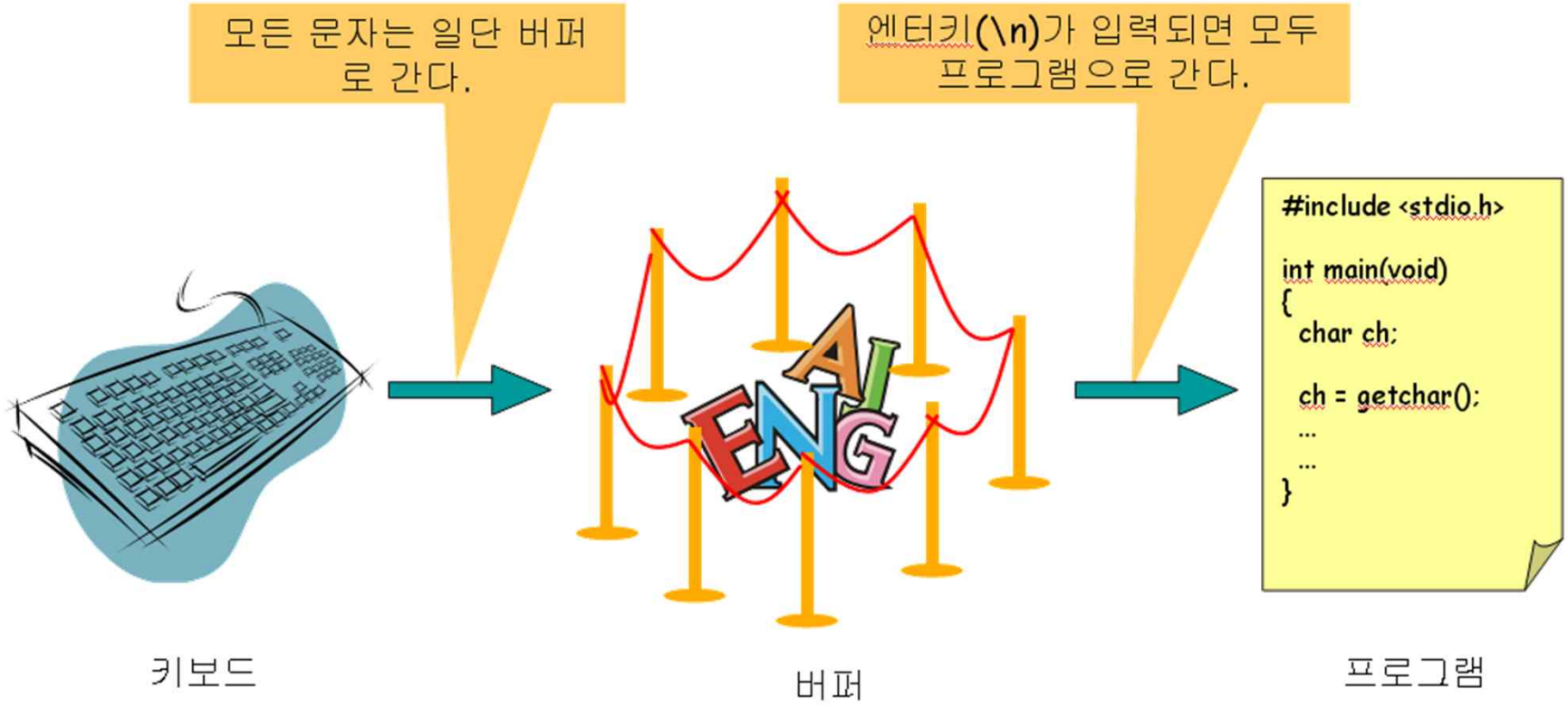
```c
// getchar()의 사용
#include <stdio.h>

int main(void)
{
	int ch;				// 정수형에 주의

	while(1)
	{
		ch = getchar();		// 문자를 입력받는다.
		if( ch == 'q' ) break;
		putchar(ch);
	}
	return 0;
}
```

반드시 엔터키가 눌려져야만이 입력을 받는다.

```
A
A
B
B
q
```

문자 입출력 버퍼

- 일반적으로 버퍼를 사용한다.



getch(), putch()

```
// getch()의 사용
#include <conio.h>

int main(void)
{
      int ch;              // 정수형에 주의

      while(1)
      {
            ch = getch();          // 문자를 입력받는다.
            if( ch == 'q' ) break;
            putch(ch);
      }
      return 0;
}
```

버퍼를 사용하지 않는다

```
ABCDEFGH
```

getch(), getche(), getchar()

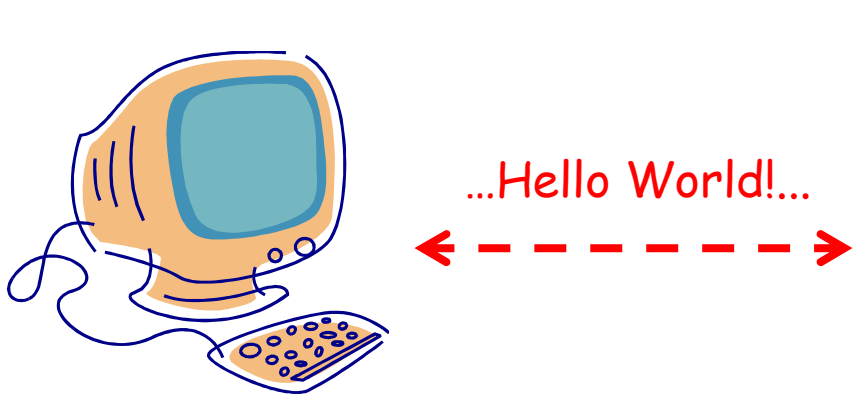
	헤더파일	버퍼사용여부	에코여부	응답성	문자수정여부
getchar()	<stdio.h>	사용함 (엔터키를 눌러입력됨)	에코	줄단위	가능
getch()	<conio.h>	사용하지 않음	에 코 하 지 않음	문자단위	불가능
getche()	<conio.h>	사용하지 않음	에코	문자단위	불가능

중간 점검

1. getchar()와 getch()가 다른 점은 무엇인가?
2. 하나의 문자를 입력받는 방법에는 몇 가지나 있는가?

문자열 입출력 라이브러리 함수

입출력 함수	설명
int scanf("%s", s)	문자열을 읽어서 문자배열 s[]에 저장
int printf("%s", s)	배열 s[]에 저장되어 있는 문자열을 출력한다.
char *gets(char *s)	한 줄의 문자열을 읽어서 문자 배열 s[]에 저장한다.
int puts(const char *s)	배열 s[]에 저장되어 있는 한 줄의 문자열을 출력한다.



scanf(), printf() 문자열 입출력

- scanf()의 사용법
  - char str[10];
  - scanf("%s", str);

- scanf()는 한 번에 두개 이상의 문자열도 받아들일 수 있다.
  - char s1[10];
  - char s2[10];
  - char s3[10];
  - scanf("%s%s%s", s1,s2,s3);
  - // 사용자가 one two three와 같이 입력하면 s1에는 one이, s2에는 two가, s3에는 three가 할당된다.

예 제

```
1    #include <stdio.h>
2
3    int main(void)
4    {
5            char name[100];
6            char address[100];
7
8            printf("이름을 입력하시오: ");
9            scanf("%s", name);
10           printf("현재 거주하는 도시를 입력하시오: ");
11           scanf("%s", address);
12
13           printf("당신은 %s에 사는 %s입니다.\n", address, name);
14           return 0;
15   }
```

배열의 이름이 배열의 주소이므로 &name와 같이 하면 안 된다.

실행 결과

```
이름을 입력하시오: 홍길동
현재 거주하는 도시를 입력하시오: 서울
당신은 서울에 사는 홍길동입니다.
```

gets()와 puts() 문자열 입출력

형식	char *gets(char *buffer); int puts(const char *str);
설명	사용자로부터 한 줄을 입력받거나 출력한다.

- gets()
  - 표준 입력으로부터 엔터키가 나올 때까지 한 줄의 라인을 입력
  - 문자열에 줄바꿈 문자('\n')는 포함되지 않으며 대신에 자동으로 NULL 문자('\0')를 추가한다.
  - 입력받은 문자열은 buffer가 가리키는 주소에 저장된다.

- puts()
  - str이 가리키는 문자열을 받아서 화면에 출력
  - NULL 문자('\0')는 줄바꿈 문자('\n')로 변경

예 제

```
#include <stdio.h>

int main(void)
{
	char name[100];
	char address[100];

	printf("이름을 입력하시오: ");
	gets(name);
	printf("현재 거주하는 주소를 입력하시오: ");
	gets(address);

	puts(name);
	puts(address);
	return 0;
}
```

```
이름을 입력하시오: 홍길동
현재 거주하는 주소를 입력하시오: 서울시 종로구 100번지
홍길동
서울시 종로구 100번지
```

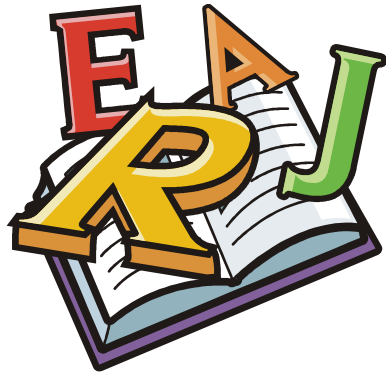
중간 점검

1. 한줄의 텍스트를 입력받는 문장을 작성하라.
2. 사용자로부터 하나의 단어를 입력받는 문장을 작성하라.

문자 처리 라이브러리 함수

- 문자를 검사하거나 문자를 변환한다.

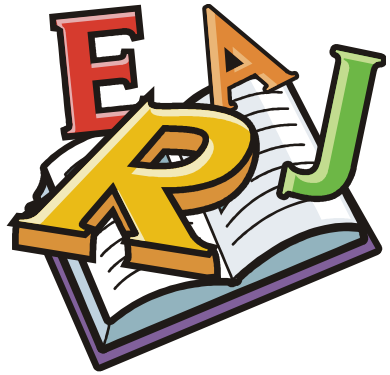
함수	설명
isalpha(c)	c가 영문자인가?(a-z, A-Z)
isupper(c)	c가 대문자인가?(A-Z)
islower(c)	c가 소문자인가?(a-z)
isdigit(c)	c가 숫자인가?(0-9)
isalnum(c)	c가 영문자이나 숫자인가?(a-z, A-Z, 0-9)
isxdigit(c)	c가 16진수의 숫자인가?(0-9, A-F, a-f)
isspace(c)	c가 공백문자인가?(' ', '\n', '\t', '\v', '\r')
ispunct(c)	c가 구두점 문자인가?
isprint(c)	C가 출력가능한 문자인가?
iscntrl(c)	c가 제어 문자인가?
isascii(c)	c가 아스키 코드인가?



문자 처리 라이브러리 함수

- 문자를 검사하거나 문자를 변환한다.

함수	설명
toupper(c)	c를 대문자로 바꾼다.
tolower(c)	c를 소문자로 바꾼다.
toascii(c)	c를 아스키 코드로 바꾼다.



예 제

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

int main( void )
{
      int c;

      while((c = getchar()) != EOF)
      {
            if( islower(c) )
                  c = toupper(c);
            putchar(c);
      }
      return 0;
}
```

소문자인지 검사

대문자로 변환

```
abcdef
ABCDEF
^Z
```

예 제

```c
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

int main( void )
{
	int c;

	while((c = getch()) != 'z')
	{
		printf("----------------------\n");
		printf("isdigit(%c) = %d\n", c, isdigit(c));
		printf("isalpha(%c) = %d\n", c, isalpha(c));
		printf("islower(%c) = %d\n", c, islower(c));
		printf("ispunct(%c) = %d\n", c, ispunct(c));
		printf("isxdigit(%c) = %d\n", c, isxdigit(c));
		printf("isprint(%c) = %d\n", c, isprint(c));
		printf("----------------------\n\n");
	}
	return 0;
}
```

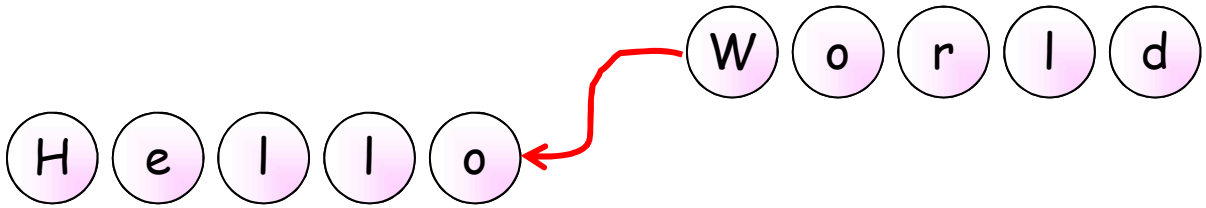
```
----------------------
isdigit(') = 0
isalpha(') = 0
islower(') = 0
ispunct(') = 16
isxdigit(') = 0
isprint(') = 16
----------------------
...
```

중간 점검

1. 문자 처리 라이브러리 함수를 사용하려면 포함시켜야 하는 헤더 파일은 무엇인가?
2. ispunct('.')의 반환값은 무엇인가?
3. toupper('a')의 반환값은 무엇인가?

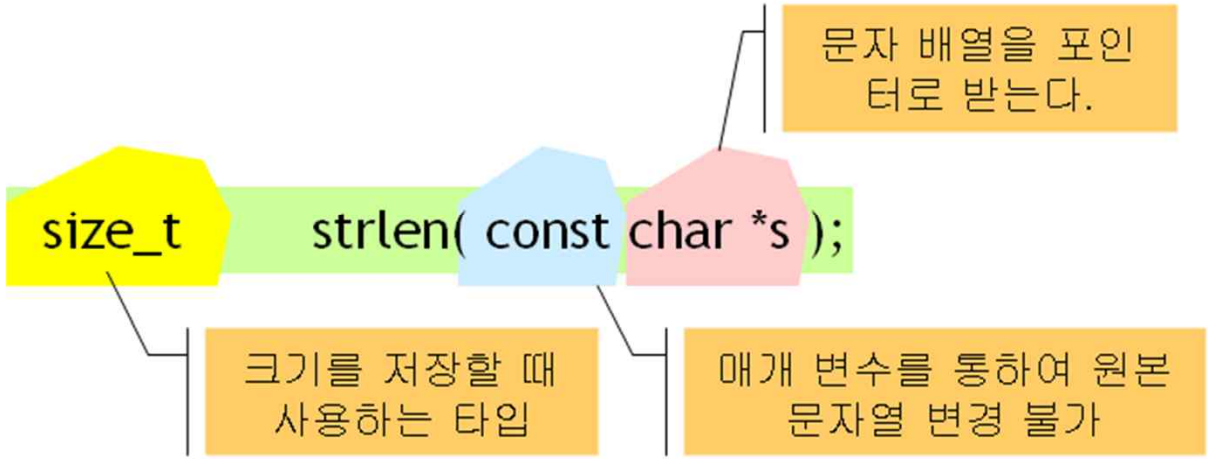
문자열 처리 라이브러리

함수	설명
strlen(s)	문자열 s의 길이를 구한다.
strcpy(s1, s2)	s2를 s1에 복사한다.
strcat(s1, s2)	s2를 s1의 끝에 붙여넣는다.
strcmp(s1, s2)	s1과 s2를 비교한다.
strncpy(s1, s2, n)	s2의 최대 n개의 문자를 s1에 복사한다.
strncat(s1, s2, n)	s2의 최대 n개의 문자를 s1의 끝에 붙여넣는다.
strncmp(s1, s2, n)	최대 n개의 문자까지 s1과 s2를 비교한다.
strchr(s, c)	문자열 s안에서 문자 c를 찾는다.
strstr(s1, s2)	문자열 s1에서 문자열 s2를 찾는다.



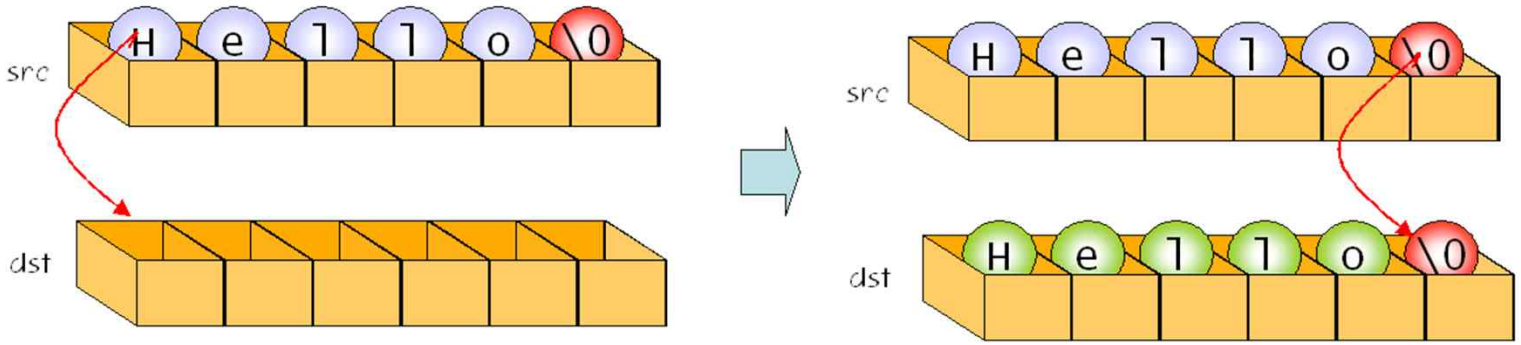
문자열 길이

형식	size_t strlen(const char *s);
설명	문자열의 길이를 반환한다. NULL 문자는 제외된다.
예	size = strlen("Hello"); // 5를 반환한다.

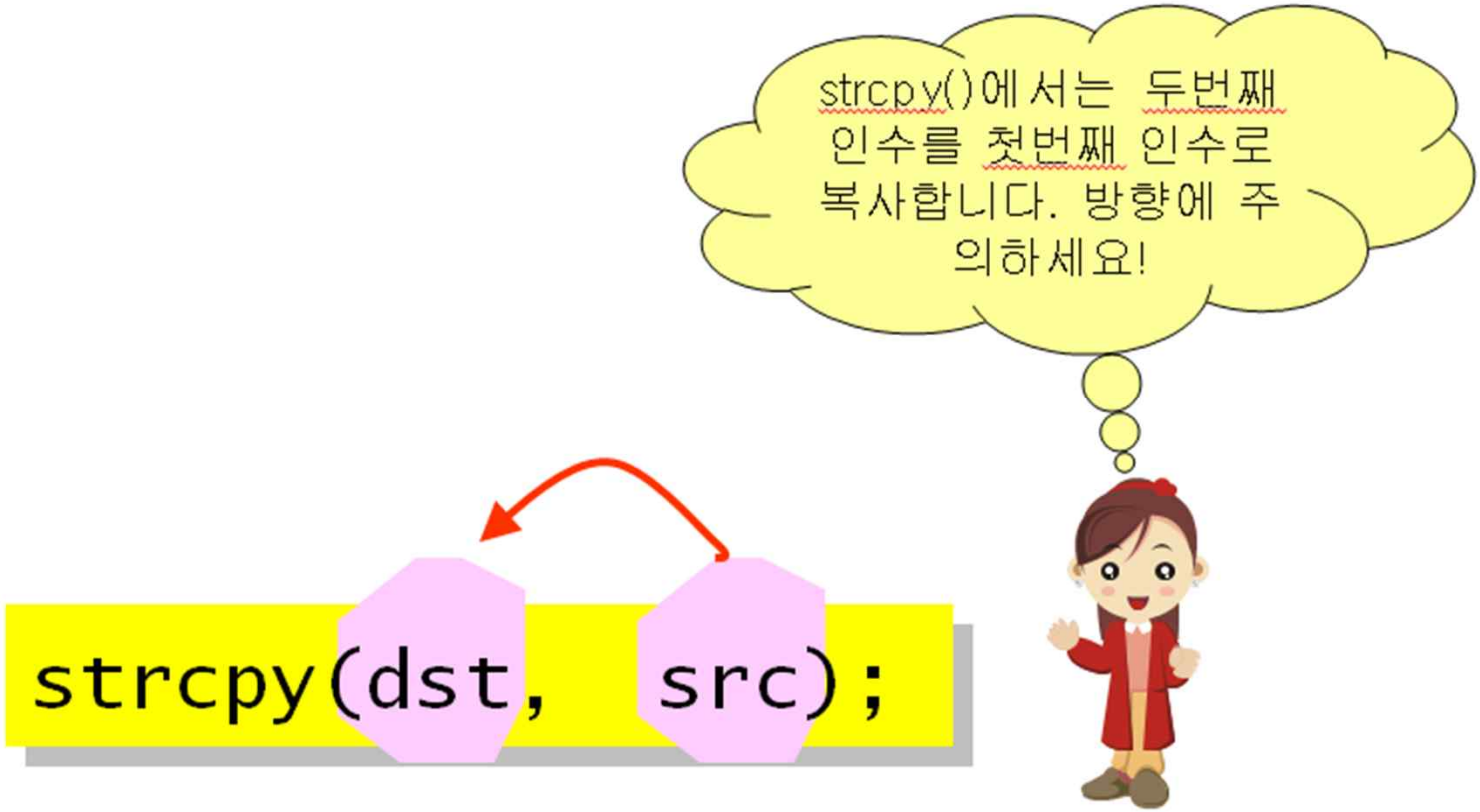


문자열 복사

형식	char *strcpy(char *dst, const char *src);
설명	src를 dst로 복사한다.
예	char dst[6]; char src[6] = “Hello"; strcpy(dst, src); // "Hello"가 dst로 복사된다.

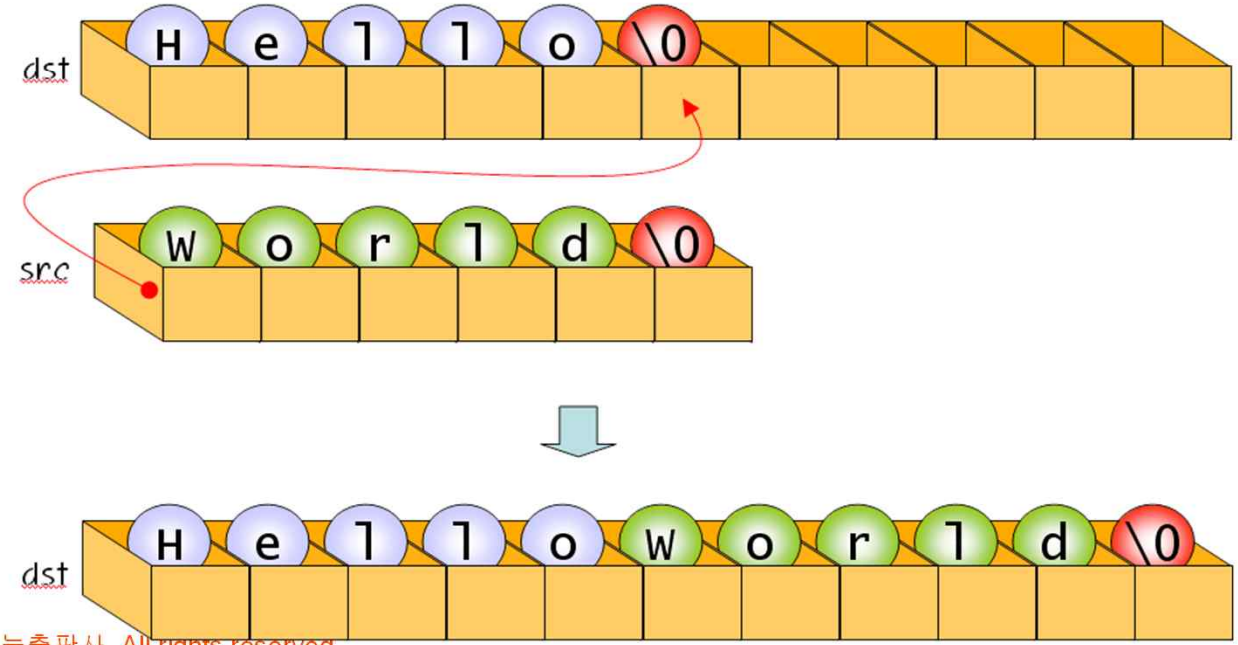


strcpy 사용시 주의 점



문자열 연결

형식	char *strcat(char *dst, const char *src);
설명	src를 dst에 붙인다.
예	char dst[12]= "Hello"; char src[6] = "World"; strcat(dst, src); // dst가 "HelloWorld"가 된다.



예 제

```
// strcpy와 strcat
#include <string.h>
#include <stdio.h>

int main( void )
{
    char string[80];

    strcpy( string, "Hello world from " );
    strcat( string, "strcpy " );
    strcat( string, "and " );
    strcat( string, "strcat!" );
    printf( "string = %s\n", string );
    return 0;
}
```

```
string = Hello world from strcpy and strcat!
```

문자열 비교

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

반환값	s1과 s2의 관계
<0	s1이 s2보다 앞에 있다.
0	s1이 s2와 같다.
>0	s1이 s2보다 뒤에 있다.

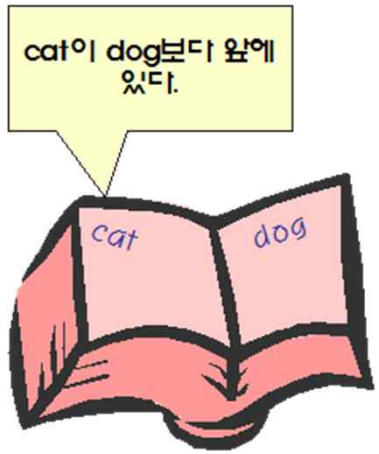


그림 10.10 사전적인 순서

예제

```
// strcmp() 함수
#include <string.h>
#include <stdio.h>

int main( void )
{
    char s1[80];     // 첫번째 단어를 저장할 문자배열
    char s2[80];     // 두번째 단어를 저장할 문자배열
    int result;

    printf("첫번째 단어를 입력하시오:");
    scanf("%s", s1);
    printf("두번째 단어를 입력하시오:");
    scanf("%s", s2);
```

예제

```
    result = strcmp(s1, s2);

    if( result < 0 )
        printf("%s가 %s보다 앞에 있읍니다.\n", s1, s2);
    else if( result == 0 )
        printf("%s가 %s와 같습니다.\n", s1, s2);
    else
        printf("%s가 %s보다 뒤에 있습니다.\n", s1, s2);

    return 0;
}
```

```
첫번째 단어를 입력하시오:Hello
두번째 단어를 입력하시오:World
Hello가 World보다 앞에 있읍니다.
```

문자 검색, 문자열 검색

- 문자열에서 문자 검색

```
char s[] = "language"; // 문자열
char c = 'g';                    // 찾고자 하는 문자
char *p;                 // 문자 포인터

p = strchr(s, c);        // str에서 c를 찾는다.
```

- 문자열에서 문자열 검색

```
char s[] = "A joy that's shared is a joy made double"; // 입력 문자열
char sub[] = "joy";                                    // 찾으려고 하는 문자열
char *p;                                               // 문자 검색 위치 저장 포인터

p = strstr(s, sub);                                    // s에서 sub를 찾는다.
```

문자열 토큰 분리

```
char *strtok( char *s,     const char *delimit );
```

입력 문자열

분리자(예를 들어서 스페이스나 탭)

```
char s[] = "A joy that's shared is a joy made double";// 입력 문자열

token = strtok(s, " ");                  // " "로 분리된 토큰 "A"를 얻는다.
token = strtok(NULL, " ");               // " "로 분리된 토큰 "joy"를 얻는다
```

같은 문자열에서 다음 토큰을 얻을 때는 NULL을 기입한다.

문자열 토큰 분리

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

char s[] = "Man is immortal, because he has a soul";
char seps[]   = " ,\t\n";
char *token;

int main( void )
{
    // 문자열을 전달하고 다음 토큰을 얻는다.
    token = strtok( s, seps );
    while( token != NULL )
    {
        // 문자열 s에 토큰이 있는 동안 반복한다.
        printf( "토큰: %s\n", token );
        // 다음 토큰을 얻는다.
        token = strtok( NULL, seps ); //
    }
    return 0;
}
```

```
토큰: Man
토큰: is
토큰: immortal
토큰: because
토큰: he
토큰: has
토큰: a
토큰: soul
```

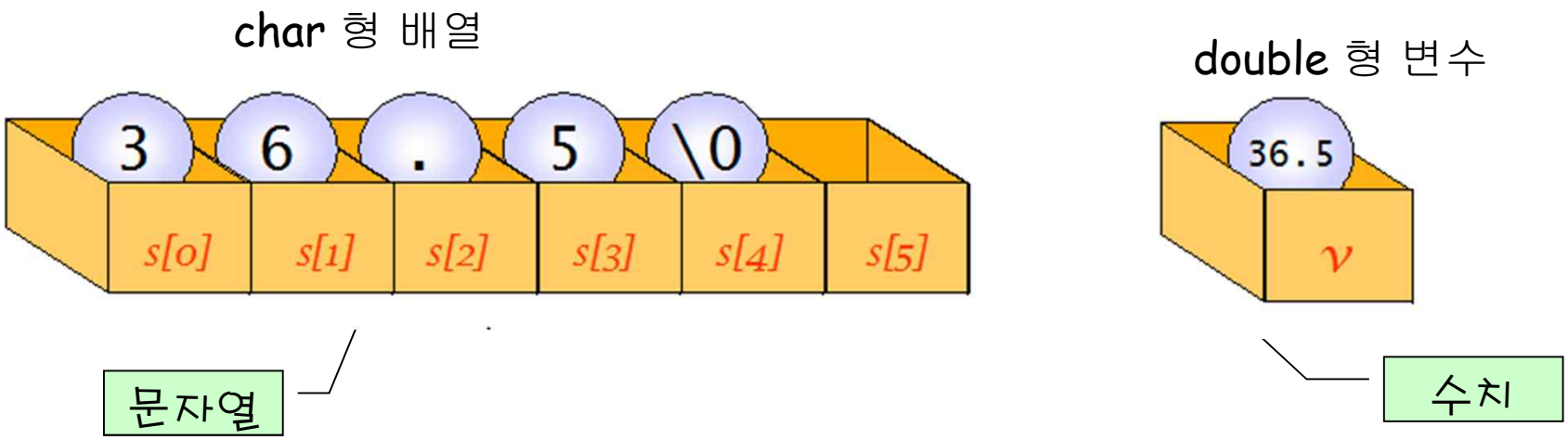
중간 점검

1. 문자열 s1를 문자열 s2로 복사하는 문장을 써라.
2. "String"을 저장하려면 최소한 어떤 크기 이상의 문자 배열이 필요한가?
3. 문자열을 서로 비교하는 함수는?
4. s1[]에 저장된 문자열 뒤에 s2[]를 붙이고 싶으면 어떤 라이브러리 함수를 어떻게 사용하여야 하는가?
5. strcmp("dog", "dog")의 반환값은 얼마인가?

문자열과 수치

- 문자열과 수치

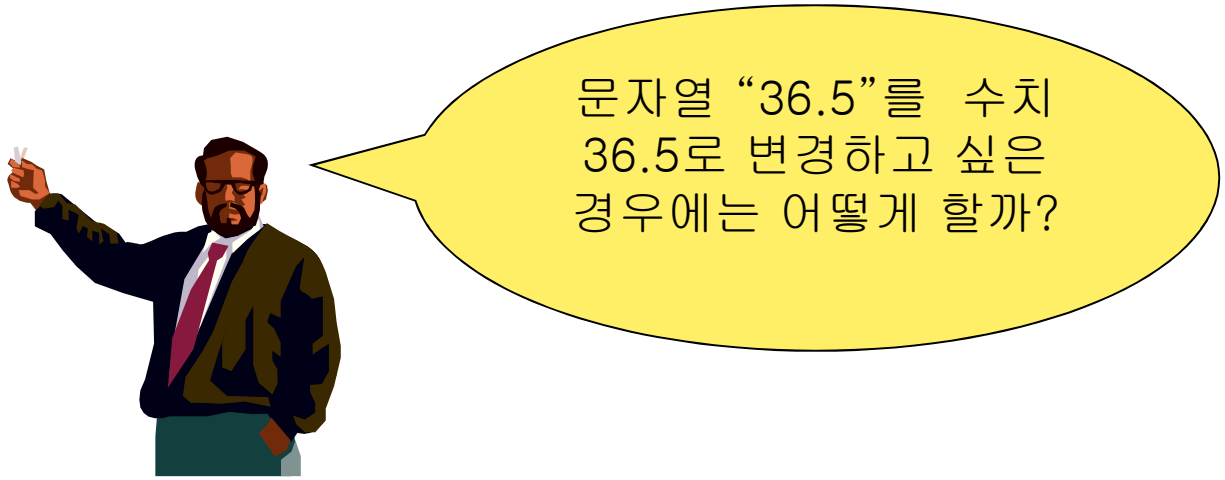
"36.5" 와 36.5



문자열과 수치

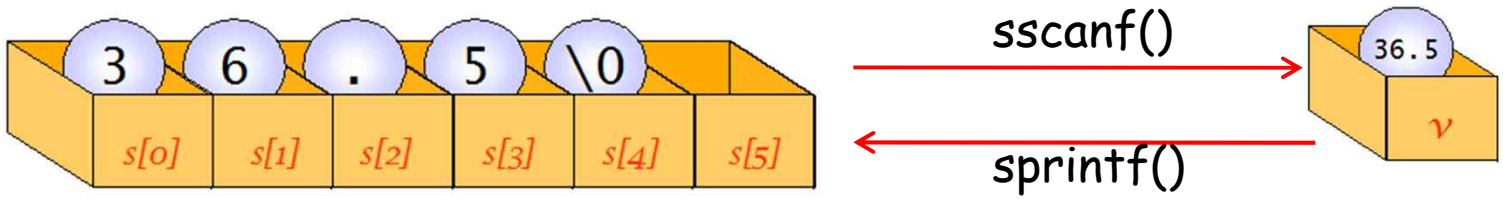
- 문자열과 수치

"36.5" ⟷ 36.5



문자열 <-> 수치

함수	설명
sscanf(s, ...)	문자열 s로부터 지정된 형식으로 수치를 읽어서 변수에 저장한다.
sprintf(s, ...)	변수의 값을 형식 지정자에 따라 문자열 형태로 문자 배열 s에 저장한다.



문자열<-> 수치

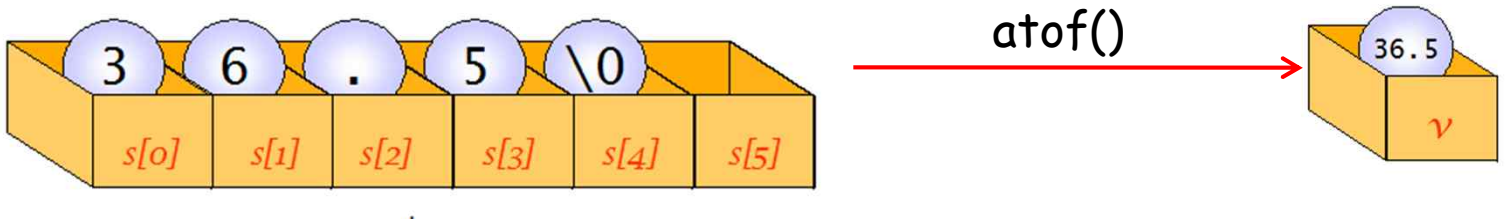
```
#include <stdio.h>

int main( void )
{
        char s1[] = "100";
        char s2[] = "12.93";
        char buffer[100];

        int i;
        double d;
        double result;

        sscanf(s1, "%d", &i);
        sscanf(s2, "%lf", &d);

        result = i + d;

        sprintf(buffer, "%f", result);
        printf("연산 결과는 %s입니다.\n", buffer);
        return 0;
}
```

```
연산 결과는 112.930000입니다.
```

문자열을 수치로 변환하는 전용함수

- 전용 함수는 scanf()보다 크기가 작다.
- stdlib.h에 원형 정의- 반드시 포함

함수	설명
int atoi(const char *str);	str을 int형으로 변환한다.
long atoi(const char *str);	str을 long형으로 변환한다.
double atof(const char *str);	str을 double형으로 변환한다.



문자열<-> 수치

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

int main( void )
{
        char s1[] = "100";
        char s2[] = "12.93";
        char buffer[100];

        int i;
        double d;
        double result;

        i = atoi(s1);
        d = atof(s2);

        result = i + d;
        sprintf(buffer, "%f", result);
        printf("연산 결과는 %s입니다.\n", buffer);
        return 0;
}
```

```
연산 결과는 112.930000입니다.
```

중간 점검

1. 실수값 3.141592와 문자열 "3.141592"가 차지하는 메모리 공간을 비교하라.
2. 문자열 "3.141592"를 실수값을 변환하고자 할 때 사용할 수 있는 함수는 어떤 것들이 있는가?
3. printf()와 sprintf()가 다른 점은 무엇인가?

문자열의 배열

- (Q)문자열이 여러 개 있는 경우에는 어떤 구조를 사용하여 저장하면 제일 좋을까?

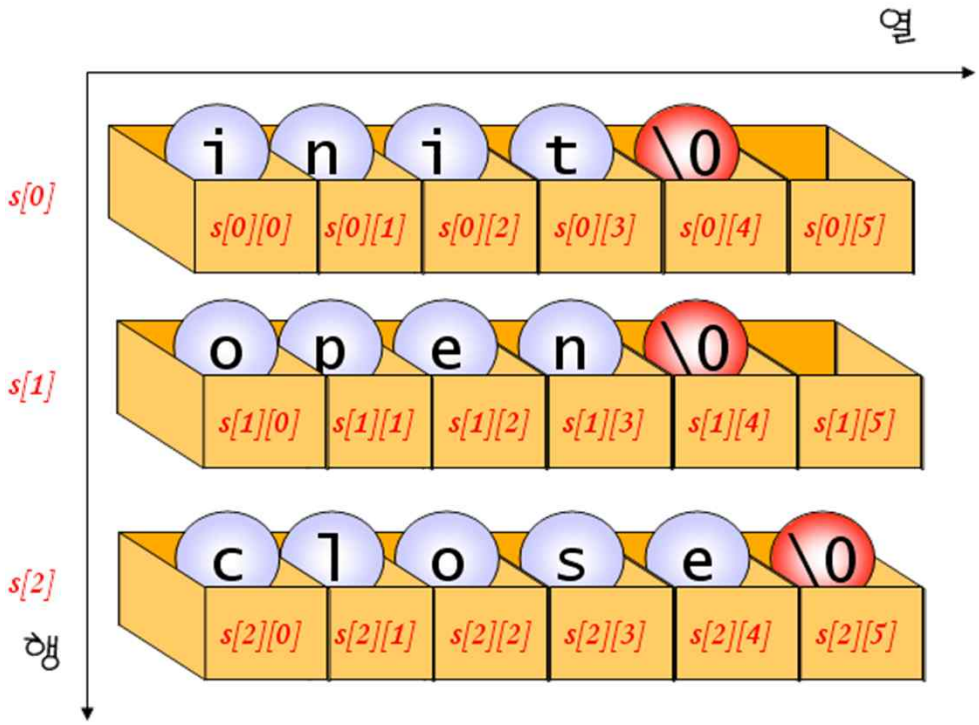
- (A) 여러 개의 문자 배열을 각각 만들어도 되지만 문자열의 배열을 만드는 것이 여러모로 간편하다.

```
char s[3][6] = {
        "init",
        "open",
        "close"
};
```

문자열의 배열

- 여러 개의 문자열은 2차원 문자 배열에 저장된다.

```
char s[3][6] = {
      "init",
      "open",
      "close"
};
```



메뉴 디스플레이

```c
#include <stdio.h>

int main( void )
{
	int i;
	char menu[5][10] = {
		"init",
		"open",
		"close",
		"read",
		"write"
	};

	for(i = 0; i < 5; i++)
		printf("%d 번째 메뉴: %s \n", i, menu[i]);

	return 0;
}
```

```
0 번째 메뉴: init
1 번째 메뉴: open
2 번째 메뉴: close
3 번째 메뉴: read
4 번째 메뉴: write
```

한영 사전 구현

```
#include <stdio.h>
#include <string.h>
#define WORDS 5

int main( void )
{
        int i, index;
        char dic[WORDS][2][30] = {
                {"book", "책"},
                {"boy", "소년"},
                {"computer", "컴퓨터"},
                {"lanuguage", "언어"},
                {"rain", "비"},
        };
        char word[30];
```

한영 사전 구현

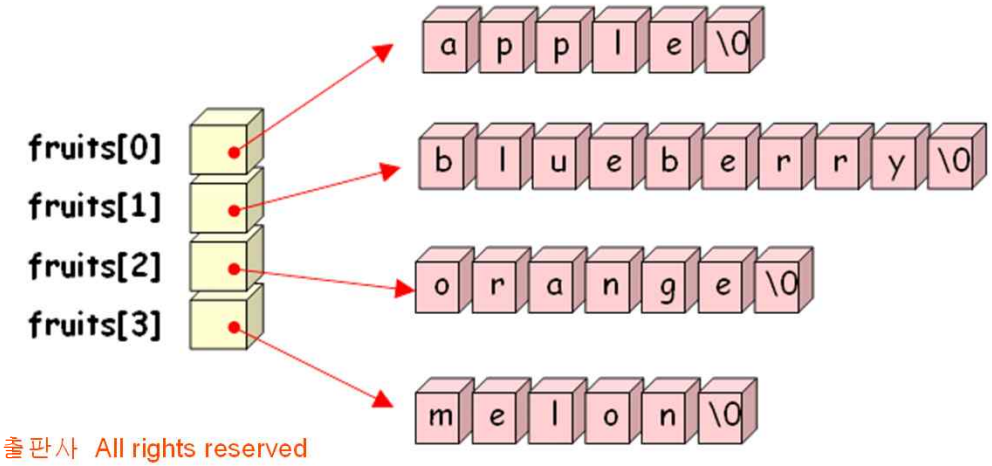
```
        printf("단어를 입력하시오:");
        scanf("%s", word);

        index = 0;
        for(i = 0; i < WORDS; i++)
        {
                if( strcmp(dic[index][0], word) == 0 )
                {
                        printf("%s: %s\n", word, dic[index][1]);
                        return 0;
                }
                index++;
        }
        printf("사전에서 발견되지 않았습니다.\n");

        return 0;
}
```

```
단어를 입력하시오:rain
rain: 비
```

래그드 배열

- 포인터 배열을 선언하여서 문자열 상수 저장
- 문자열 상수를 효율적으로 저장할 수 있다.

```
char *fruits[ ] = {
            "apple",
            "blueberry",
            "orange",
            “melon"
};
```



중간 점검

1. "C", "JAVA", "C++", "BASIC" 등을 저장할 수 있는 방법을 가능한 한 많이 제시하라.
2. 2차원 문자 배열 s[][]에서 첫번째 문자열을 printf()를 이용하여 화면에 출력하는 문장을 작성하라.?

Q & A