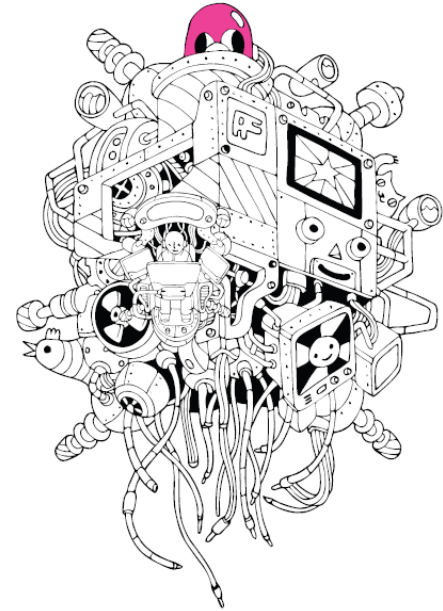


C 프로그래밍



배열과 문자열



문자 변수와 문자 상수



```
// 문자 상수
#include <stdio.h>

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

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

문자변수

문자상수



```
code1=A, code1=65
code2=A, code2=65
```

아스키 코드 출력



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

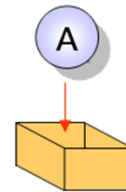
    for(code = 32; code < 128; code++)
    {
        printf("아스키 코드 %d은 %c입니다.\n", code, code);
    }
    return 0;
}
```



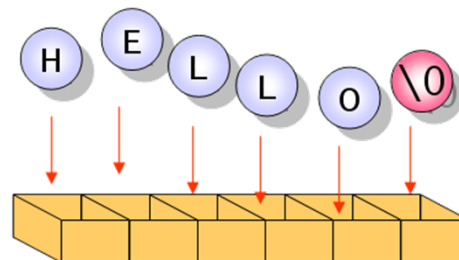
아스키 코드 32은 입니다.
아스키 코드 33은 !입니다.
...
아스키 코드 65은 A입니다.
아스키 코드 66은 B입니다.
...
아스키 코드 97은 a입니다.
아스키 코드 98은 b입니다.
...
아스키 코드 126은 ~입니다.
아스키 코드 127은 입니다.

문자열 표현 방법

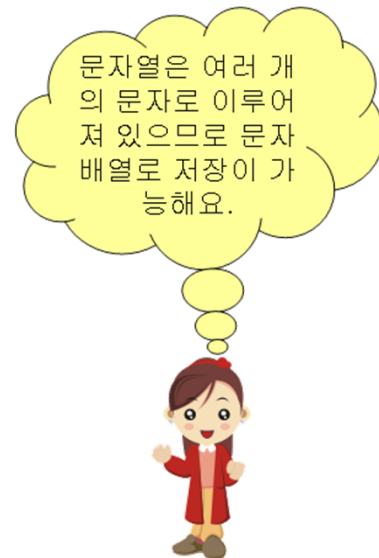
- ▶ **문자열(string)**: 문자들이 여러 개 모인것
 - ▶ "A"
 - ▶ "Hello World!"
 - ▶ "변수 score의 값은 %d입니다"
- ▶ **문자열 상수**
 - ▶ "Hello World"
 - ▶ "Hong"
 - ▶ "string!#\$"
 - ▶ "guest123"
 - ▶ "ascii code = %d"
- ▶ **문자열 변수**
 - ▶ char형 배열



하나의 문자는 char형 변수로 저장



문자열은 char형 배열로 저장



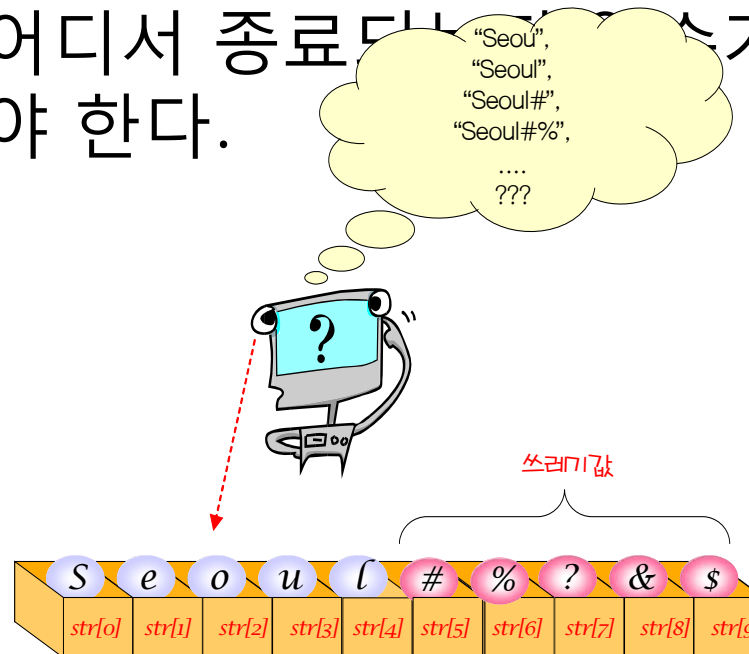
NULL 문자

- ▶ NULL 문자: 문자열의 끝을 나타낸다.

S E O U L \0



- ▶ 문자열은 어디서 종료되는지 알 수 없으므로 표시를 해주어야 한다.



문자 배열의 초기화

1. 문자 배열 원소들을 중괄호 안에 넣어주는 방법

▶ `char str[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };`

2. 문자열 상수를 사용하여 초기화하는 방법

▶ `char str[6] = "Hello";`

3. 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.



char형 배열의 문자열 저장과 널 문자

```
char str[14]="Good morning!";
```

배열에 문자열 저장



저장결과



문자열의 끝에 널 문자라 불리는 \0가 삽입되었음에
주목! 널 문자는 문자열의 끝을 의미한다.

```
int main(void)
{
    char str[]="Good morning!";
    printf("배열 str의 크기: %d \n", sizeof(str));
    printf("널 문자 문자형 출력: %c \n", str[13]);
    printf("널 문자 정수형 출력: %d \n", str[13]);

    str[12]='?'; // 배열 str에 저장된 문자열 데이터는 변경 가능!
    printf("문자열 출력: %s \n", str);
    return 0;
}
```

실행결과

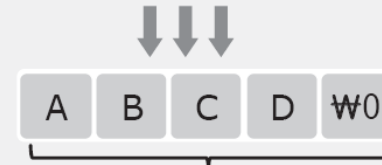
배열 str의 크기: 14
널 문자 문자형 출력:
널 문자 정수형 출력: 0
문자열 출력: Good morning?

char형 배열을 이용한 문자열의 표현

■ 예제 18-2.c

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      char str1[5]="AAA";
6.      char str2[ ]="BBB";
7.      char str3[ ]={'A', 'B', 'C'};
8.      char str4[ ]={'A', 'B', 'C', '\0'};
9.
10.     printf("str1 : %s \n", str1);
11.     printf("str2 : %s \n", str2);
12.     printf("str3 : %s \n", str3);
13.     printf("str4 : %s \n", str4);
14.
15.     str1[0]='C';
16.     str1[1]='B';
17.     printf("str1 : %s \n", str1);
18.     return 0;
19. }
```

char str[5] = "ABCD";



① 배열 생성

② 요소 초기화

널 문자와 공백 문자의 비교

```
int main(void)
{
    char nu = '\0';    // 널 문자 저장
    char sp = ' ';     // 공백 문자 저장
    printf("%d %d", nu, sp);    // 0과 32 출력
    return 0;
}
```

NULL 문자

\0

널 문자를 %c를 이용해서 출력 시 아무것도 출력되지 않는다. 그렇다고 해서 널 문자가 공백 문자는 아니다.

널 문자의 아스키 코드 값은 0이고, 공백 문자의 아스키 코드 값은 32이다.

널 문자는 모니터 출력에서 의미를 갖지 않는다. 그래서 아무것도 출력이 되지 않을 뿐이다.



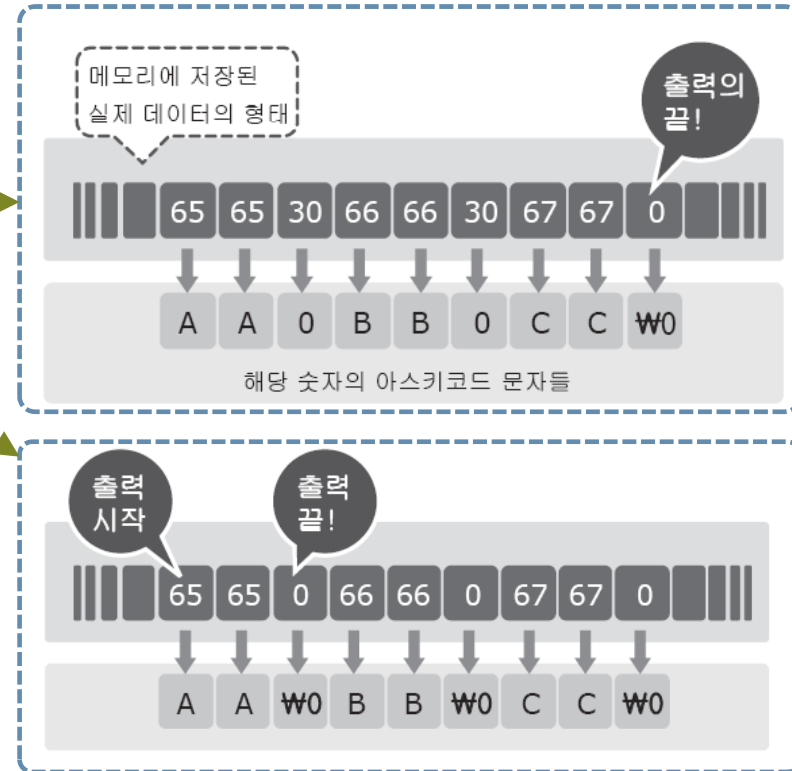
문자열의 끝에 붙는 널(null) 문자

■ 예제 18-1.c

```
1.  #include <stdio.h>
2.
3.  int main(void)
4.  {
5.      printf("AA0BB0CC");
6.      printf("|");
7.      printf("AAW0BBW0CC");
8.      printf("|");
9.      printf("W0AA");
10.     printf("|");
11.     return 0;
12. }
```

AA0BB0CC | AA | |

실행 결과



- 널 문자 'W0'는 문자열의 끝을 표시하기 위해서 삽입
- 마지막에 'W0'이 삽입되어야 문자열로 인정

문자열의 끝에 널 문자가 필요한 이유

문자열의 시작은 판단할 수 있어도 문자열의 끝은 판단이 불가능하다! 때문에 문자열의 끝을 판단할 수 있도록 널 문자가 삽입이 된다.

```
int main(void)
{
    char str[50]="I like C programming";
    printf("string: %s \n", str);

    str[8]='\0';    // 9번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[6]='\0';    // 7번째 요소에 널 문자 저장
    printf("string: %s \n", str);

    str[1]='\0';    // 2번째 요소에 널 문자 저장
    printf("string: %s \n", str);
    return 0;
}
```

배열의 시작위치에 문자열이 저장되기 시작한다. 따라서 시작위치는 확인이 가능하다. 하지만 배열의 끝이 문자열의 끝은 아니므로 널 문자가 삽입되지 않으면 문자열의 끝은 확인이 불가능하다.

실행결과

```
string: I like C programming
string: I like C
string: I like
string: I
```

위 예제에서 보이듯이 printf 함수도 배열 str의 시작위치를 기준으로해서 널 문자를 만날 때까지 출력을 진행한다. 따라서 널 문자가 없으면 printf 함수도 문자열의 끝을 알지 못한다.



문자열 길이 계산 예제



// 문자열의 길이를 구하는 프로그램

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

```
int main(void)
```

```
{
```

```
    char str[30] = "C language is easy";
```

```
    int i = 0;
```

```
    while(str[i] != 0)
```

```
        i++;
```

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

```
    return 0;
```

```
}
```



문자열 "C language is easy"의 길이는 18입니다.



문자열 처리 주의사항

Hello World!₩0 => 13자

```
void printStringArr()
{
    char str[15] = "Hello World!";
    int i = 0;

    for(i=0; i<17 ; i++)
    {
        printf("문자값 %d : [%c:%d ] \n", i, str[i], str[i]);
    }
    printf("%s\n", str);
}
```

```
0 : [H:72 ]
1 : [e:101 ]
2 : [l:108 ]
3 : [l:108 ]
4 : [o:111 ]
5 : [ :32 ]
6 : [W:87 ]
7 : [o:111 ]
8 : [r:114 ]
9 : [l:108 ]
10 : [d:100 ]
11 : [!:33 ]
12 : [ :0 ]
13 : [ :0 ]
14 : [ :0 ]
15 : [?:-52 ]
16 : [?:-52 ]
Hello World!
```

문자열 처리 주의사항

Hello World!₩0 => 13자

```
void printStringArr()
{
    char str[10] = "Hello World!";
    int i = 0;

    for(i=0; i<17 ; i++)
    {
        printf("문자값 %d : [%c:%d ] \n", i, str[i], str[i]);
    }
    printf("%s\n", str);
}
```

```
문자값 0 : [H:72 ]
문자값 1 : [e:101 ]
문자값 2 : [l:108 ]
문자값 3 : [l:108 ]
문자값 4 : [o:111 ]
문자값 5 : [ :32 ]
문자값 6 : [W:87 ]
문자값 7 : [o:111 ]
문자값 8 : [r:114 ]
문자값 9 : [l:108 ]
문자값 10 : [?:-52 ]
문자값 11 : [?:-52 ]
문자값 12 : [?:-52 ]
문자값 13 : [?:-52 ]
문자값 14 : [?:-52 ]
문자값 15 : [?:-52 ]
문자값 16 : [?:-29 ]
Hello World!微微微?Q??
```


scanf 함수를 이용한 문자열의 입력

```
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str); // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```

scanf 함수의 호출을 통해서 입력 받은
문자열의 끝에도 널 문자가 존재함을 확
인하기 위한 문장

```
char arr1[ ] = {'H', 'i', '~'};
char arr2[ ] = {'H', 'i', '~', '\0'};
```

scanf 함수를 이용해서 문자열 입력 시
서식문자 %s를 사용한다.

`scanf("%s", str);`

위와 같이 배열이름 str의 앞에는
& 연산자를 붙이지 않는다.

실행결과

문자열 입력: Simple
입력 받은 문자열: Simple
문자 단위 출력: Simple

arr1은 문자열이 아닌 문자 배열, 반면 arr2는 문자열!
널 문자의 존재여부는 문자열의 판단여부가 된다.

scanf 함수의 문자열 입력 특성

```
int main(void)
{
    char str[50];
    int idx=0;

    printf("문자열 입력: ");
    scanf("%s", str); // 문자열을 입력 받아서 배열 str에 저장!
    printf("입력 받은 문자열: %s \n", str);

    printf("문자 단위 출력: ");
    while(str[idx] != '\0')
    {
        printf("%c", str[idx]);
        idx++;
    }
    printf("\n");
    return 0;
}
```

앞서 보인 원편의 예제를 실행할 때 다음과 같이 문자열을 입력하면

He is my friend

다음의 실행결과를 보인다.

입력 받은 문자열: He

문자 단위 출력: He

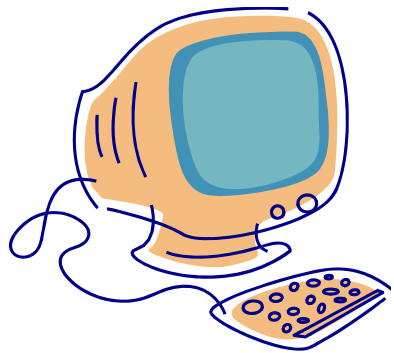
scanf 함수는 공백을 기준으로 데이터의 수를 구분한다. 따라서 공백을 포함하는 문자열을 한번의 scanf 함수호출을 통해서 읽어 들이지는 못한다.

공백을 포함하는 문자열의 입력에 사용되는 함수는 이후에 별도로 설명합니다.

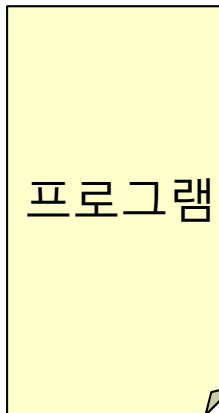


문자열 입출력 라이브러리 함수

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



...Hello World!...



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 one 이, s2에는 two가, s3에는 three가 할당된다.



예제



```
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

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

▶ 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

```
char *menu = "파일열기: open, 파일닫기: close";  
puts("메뉴에서 하나를 선택하십시오.");  
puts(str);
```

예제



```
#include <stdio.h>

int main( void )
{
    char buffer[21]; // 20개의 문자와 '\0'을 저장할 수 있다.

    printf("문자열을 입력하시오.\n");
    gets( buffer );

    printf("입력된 라인은 다음과 같습니다.\n");
    puts(buffer);
    return 0;
}
```



문자열을 입력하시오.
Hello!
입력된 라인은 다음과 같습니다.
Hello!



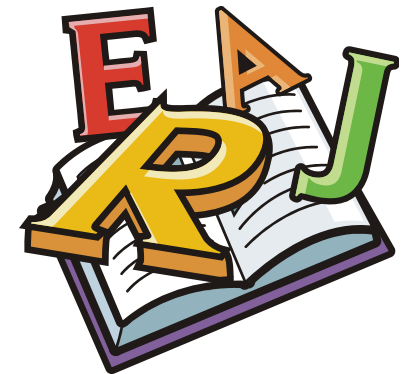
문자 라이브러리 함수

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?
isctrl(c)	c가 제어 문자인가?
isascii(c)	c가 아스키 코드인가?

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



예제



```
#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
```

예제



```
#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
-----
```

...

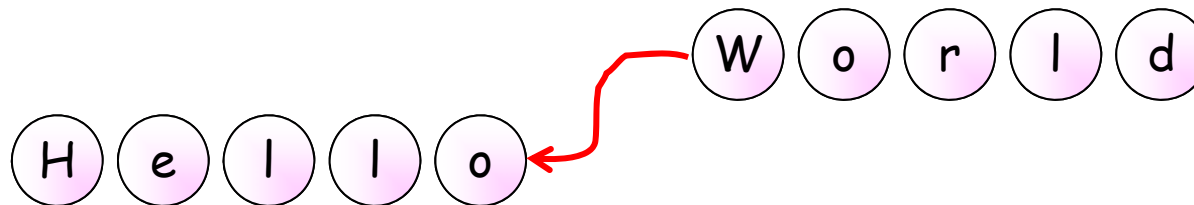
```
for (int i = 0, x = strlen(input); i < x; i++) {  
    if (isalpha(input[i])) {  
        input [i] = toupper(input[i]);  
    }  
}
```



문자열 처리 라이브러리

문자열 처리 라이브러리

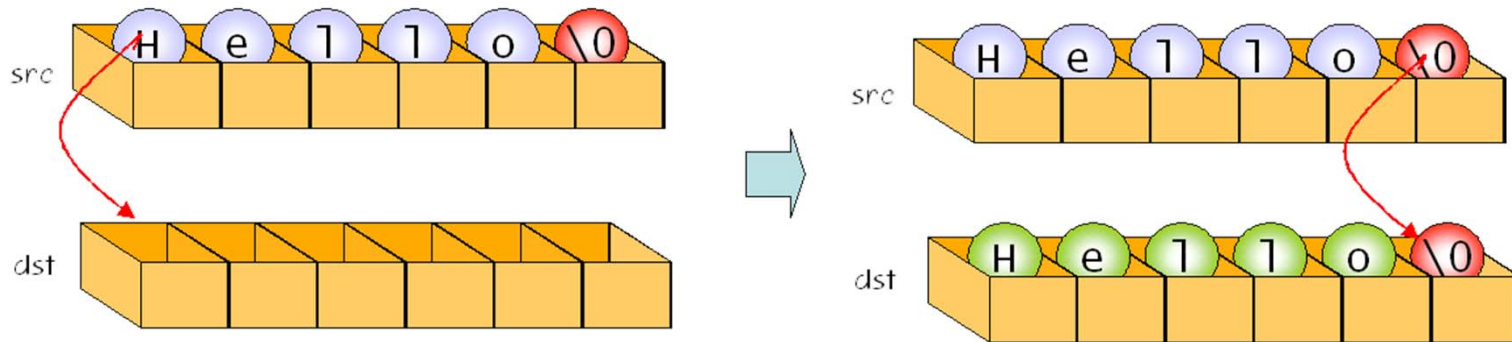
함수	설명
<code>strlen(s)</code>	문자열 <code>s</code> 의 길이를 구한다.
<code>strcpy(s1, s2)</code>	<code>s2</code> 를 <code>s1</code> 에 복사한다.
<code>strcat(s1, s2)</code>	<code>s2</code> 를 <code>s1</code> 의 끝에 붙여넣는다.
<code>strcmp(s1, s2)</code>	<code>s1</code> 과 <code>s2</code> 를 비교한다.
<code>strncpy(s1, s2, n)</code>	<code>s2</code> 의 최대 <code>n</code> 개의 문자를 <code>s1</code> 에 복사한다.
<code>strncat(s1, s2, n)</code>	<code>s2</code> 의 최대 <code>n</code> 개의 문자를 <code>s1</code> 의 끝에 붙여넣는다.
<code>strncmp(s1, s2, n)</code>	최대 <code>n</code> 개의 문자까지 <code>s1</code> 과 <code>s2</code> 를 비교한다.
<code>strchr(s, c)</code>	문자열 <code>s</code> 안에서 문자 <code>c</code> 를 찾는다.
<code>strstr(s1, s2)</code>	문자열 <code>s1</code> 에서 문자열 <code>s2</code> 를 찾는다.



문자열 길이, 복사

- ▶ 문자열의 길이
 - ▶ `strlen("Hello")`는 5를 반환
- ▶ 문자열 복사

```
char dst[6];  
char src[6] = "Hello";  
strcpy(dst, src);
```



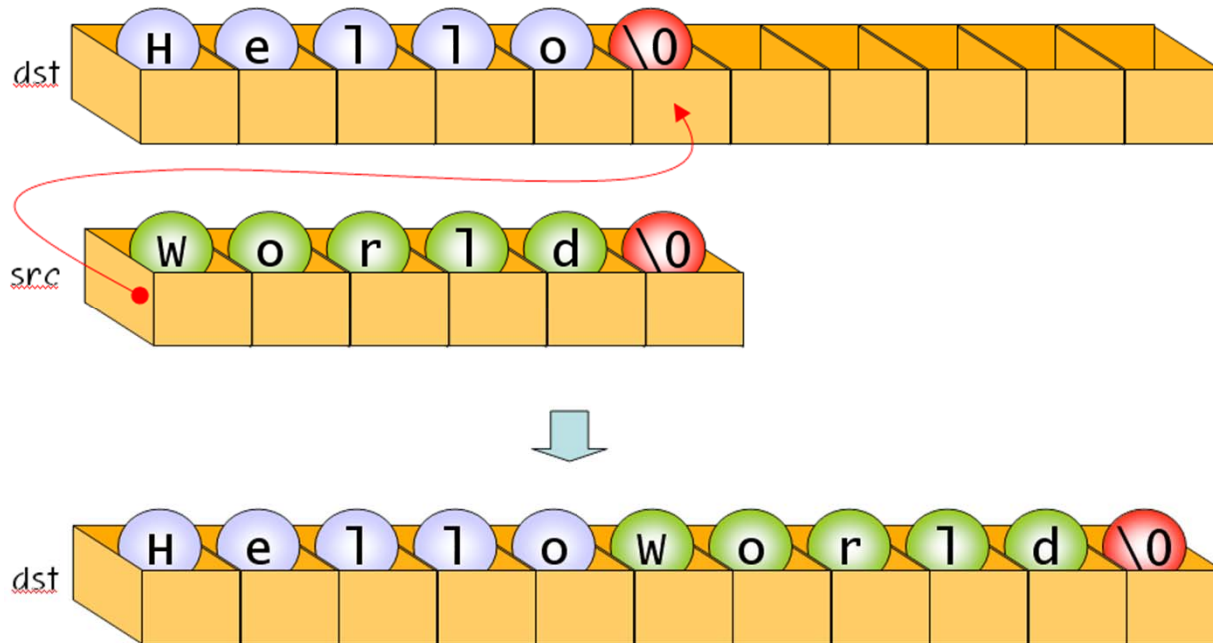
문자열 연결

▶ 문자열 연결

```
char dst[12] = "Hello";
```

```
char src[6] = "World";
```

```
strcat(dst, src);
```



예제



```
// 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;  
}
```

strcpy : string copy
strcat : string concatenate



string = Hello world from strcpy and strcat!

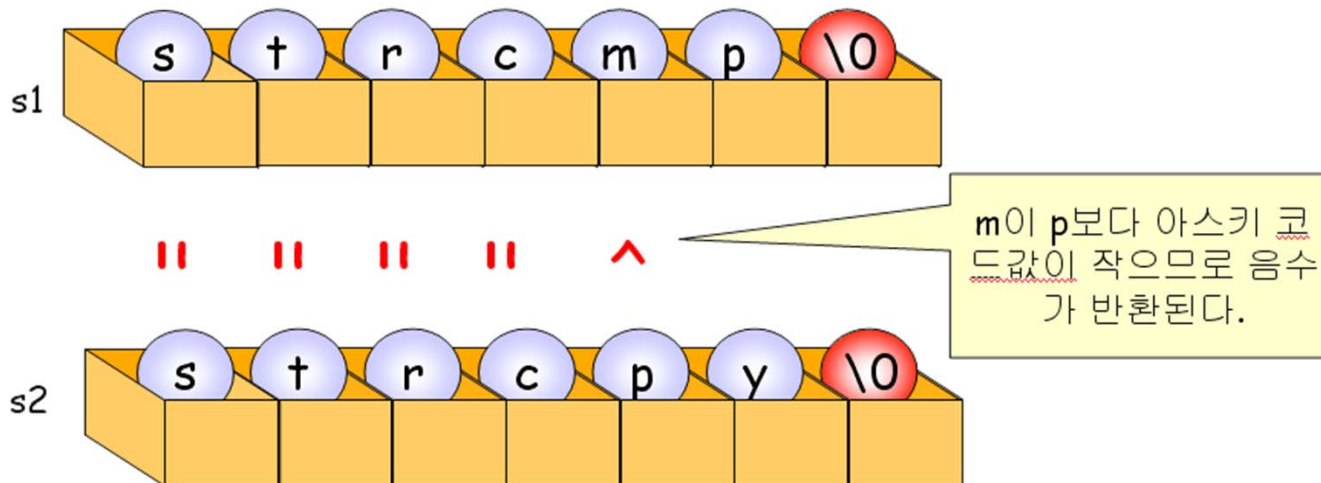


문자열 비교

strcmp : string compare

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

반환값	s1과 s2의 관계
<0	s1이 s2보다 작다
0	s1이 s2와 같다.
>0	s1이 s2보다 크다.



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);

    result = strcmp(s1, s2);
    if( result < 0 )
        printf("%s가 %s보다 앞에 있습니다.\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 있습니다.

문자 검색, 문자열 검색

▶ 문자열에서 문자 검색

strchr : string **character**

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

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

strstr : string **string**

▶ 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

▶ 문자열을 토큰으로 분리



```
// strtok 함수의 사용예
#include <string.h>
#include <stdio.h>

char s[] = "Man is immortal, because he has a soul";
char seps[] = " ,!+\\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 ); //
    }
}
```

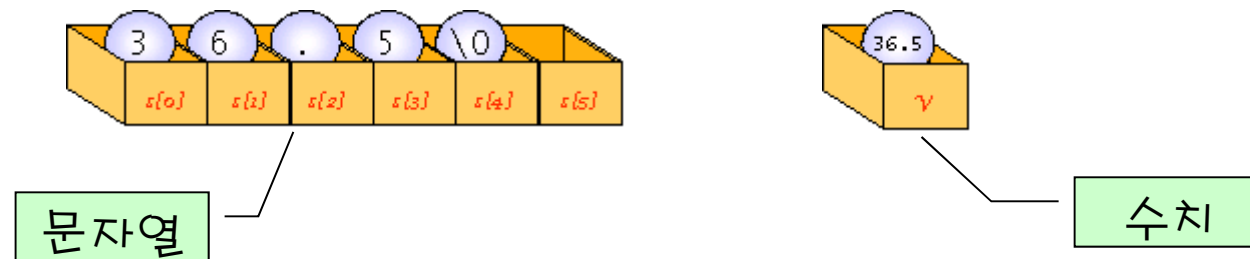
strtok : string token



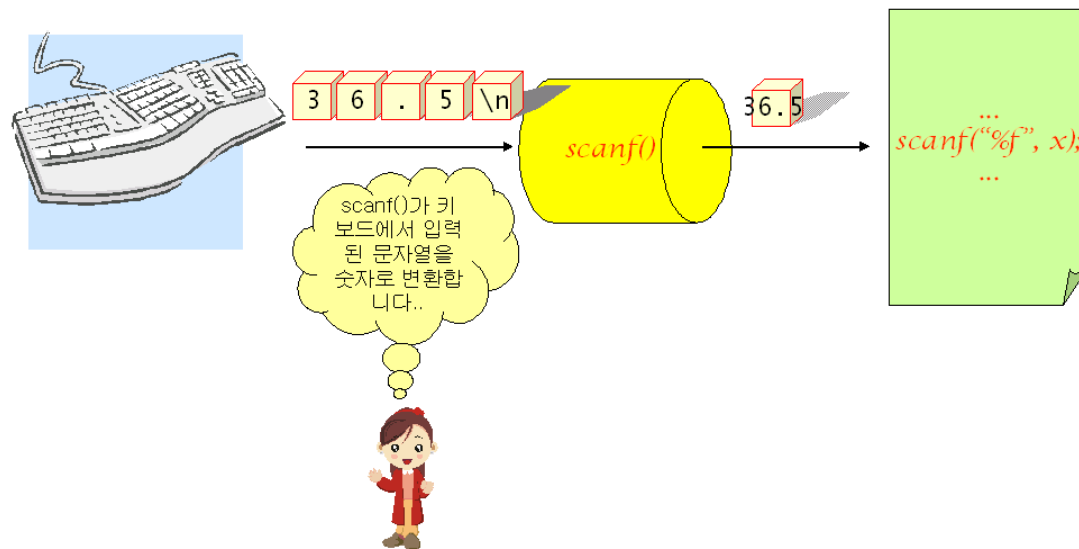
토큰: Man
토큰: is
토큰: immortal
토큰: because
토큰: he
토큰: has
토큰: a
토큰: soul

문자열 수치 변환

▶ 문자열과 수치



- `scanf()` 함수는 문자열을 수치로 변환한다.



문자열을 수치로 변환하는 전용 함수

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

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

atoi: ascii string **integer**



문자열 토큰 분리



```
// atoi() 함수
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
    char s[30];
    char t[] = "36.5";
    int i;
    double v;

    printf("정수를 입력하시오:");
    gets(s);
    i = atoi(s);
    printf("입력된 정수: %d \n", i);

    v = atof(t);
    printf("변환된 실수: %f", v);

    return 0;
}
```



정수를 입력하시오:89
입력된 정수: 89
변환된 실수: 36.500000



sscanf(), sprintf()

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



```
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.

atoi(), atof()

함수	설명
<code>int atoi(const char *str);</code>	<code>str</code> 을 <code>int</code> 형으로 변환한다.
<code>double atof(const char *str);</code>	<code>str</code> 을 <code>double</code> 형으로 변환한다.



```
#include <stdio.h>
#include <stdlib.h>
int main( void )
{
    char s1[] = "100";
    char s2[] = "12.93";
    char buffer[100];
    int i;
    double d, result;
    i = atoi(s1);
    d = atof(s2);
    result = i + d;
    sprintf(buffer, "%f", result);
    printf("연산 결과는 %s입니다.\n", buffer);
    return 0;
}
```

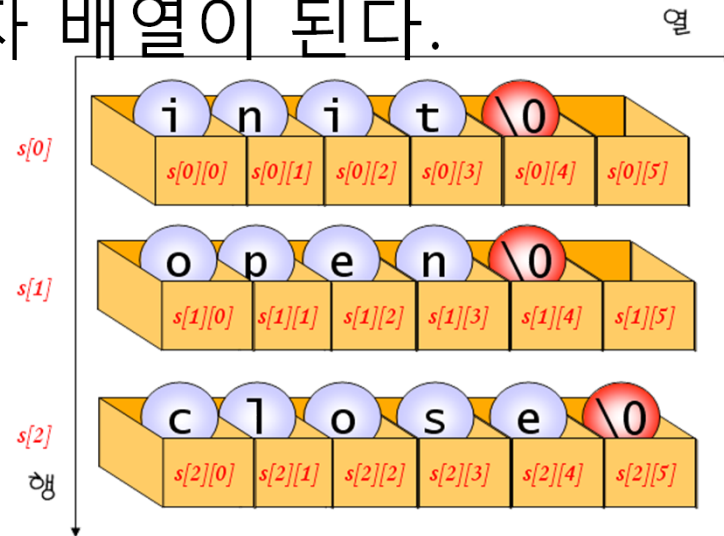


연산 결과는 112.930000입니다.

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.
- ▶ 문자열이 문자 배열에 저장되므로 문자열의 배열은 배열의 배열, 즉 2차원 문자 배열이 된다.

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



메뉴 디스플레이



```
#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

메뉴 선택



```
#include <stdio.h>

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

    printf("메뉴를 입력하시오:");
    scanf("%s", buffer);

    for(i = 0; i < 5; i++)
        if( strcmp(buffer, menu[i]) == 0 )
            printf("%d번째 메뉴를 입력하였습니다.\n", i);

    return 0;
}
```



메뉴를 입력하시오:open
1번째 메뉴를 입력하였습니다.

하영 사전 구현



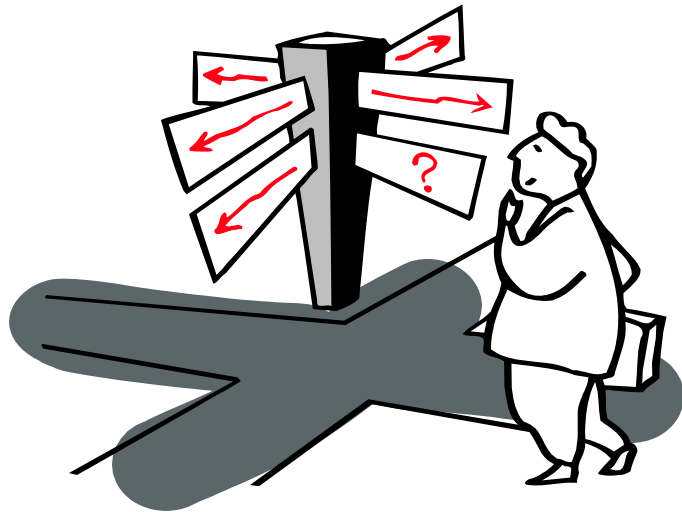
```
#define ENTRIES 5
```

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

    char word[30];

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

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



Chapter 11이 끝났습니다. 질문 있으신지요?