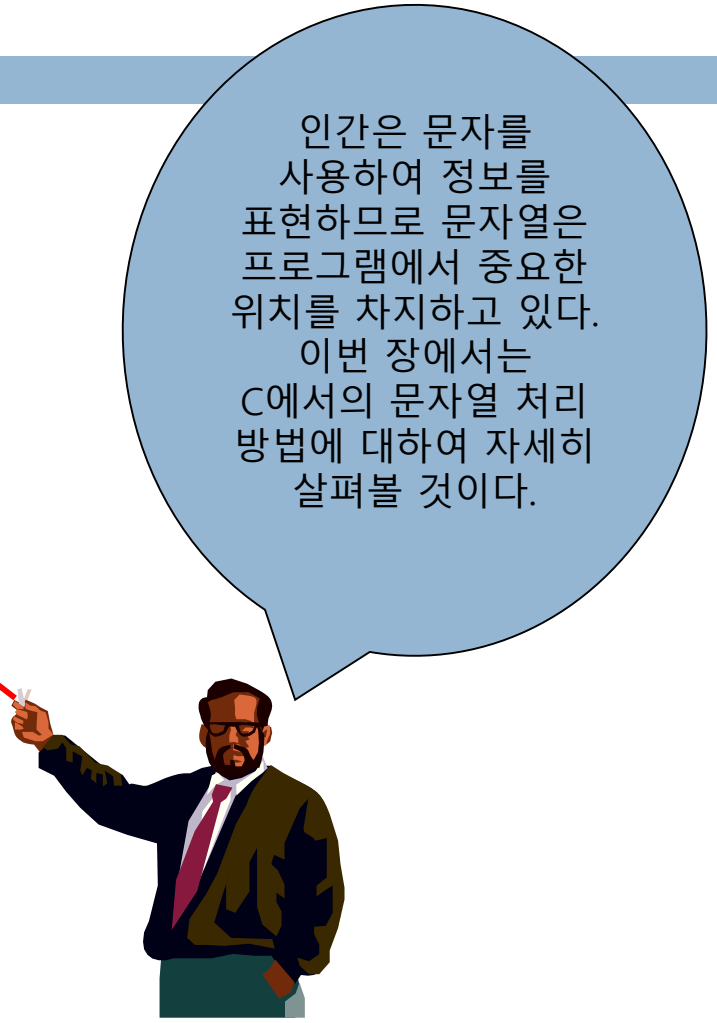
제12장 문자와 문자열

이번 장에서 학습할 내용

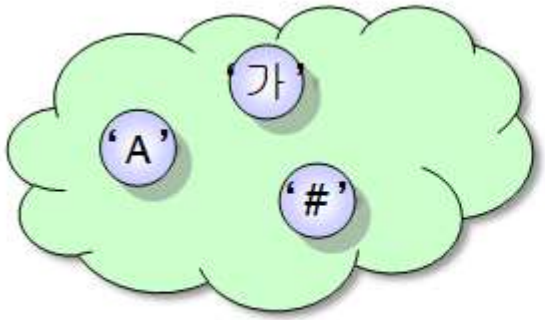
- 문자 표현 방법
- 문자열 표현 방법
- 문자열이란 무엇인가?
- 문자열의 입출력
- 문자처리 라이브러리 함수
- 표준입출력 라이브러리 함수



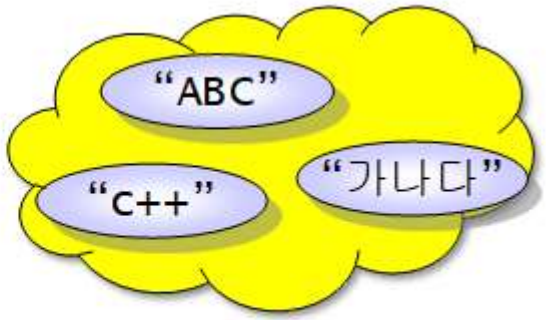
문자의 중요성

- 인간한테 텍스트는 대단히 중요하다.

문자와 문자열



문자



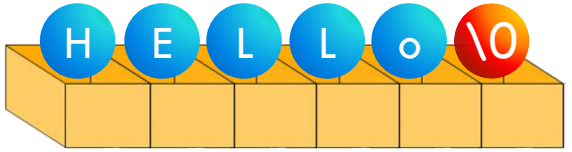
문자열

문자열 표현 방법

- *문자열(string):* 문자들이 여러 개 모인 것
  - "A "
  - "Hello World!"
  - "변수 score의 값은 %d입니다"

- *문자열 변수*
  - 변경가능한 문자열을 저장할 수 있는 변수
  - 어디에 저장하면 좋은가?

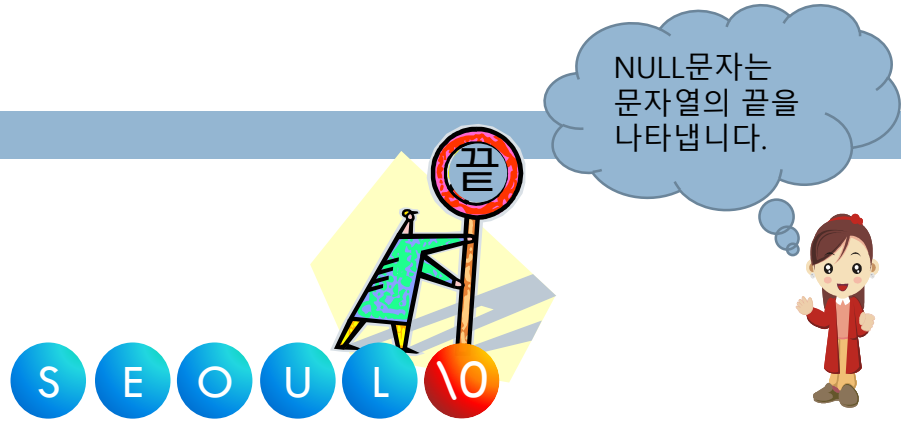
하나의 문자는 char형 변수로 저장



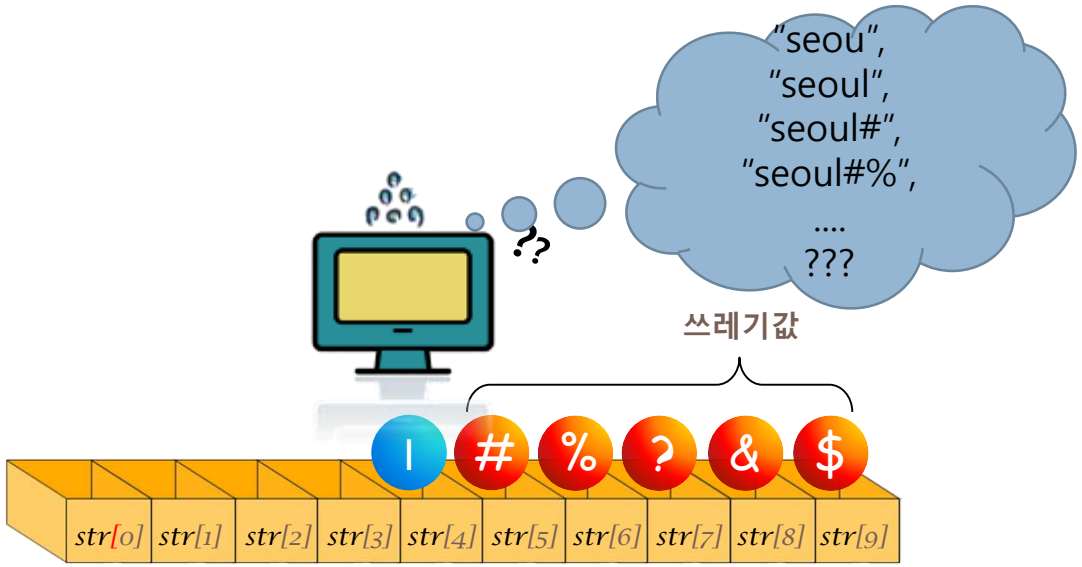
문자열은 char형 배열로 저장

NULL 문자

- NULL 문자: 문자열의 끝을 나타낸다.



- 문자열은 어디서 종료되는지 알수가 없으므로 표시를 해주어야 한다.



예제 #1

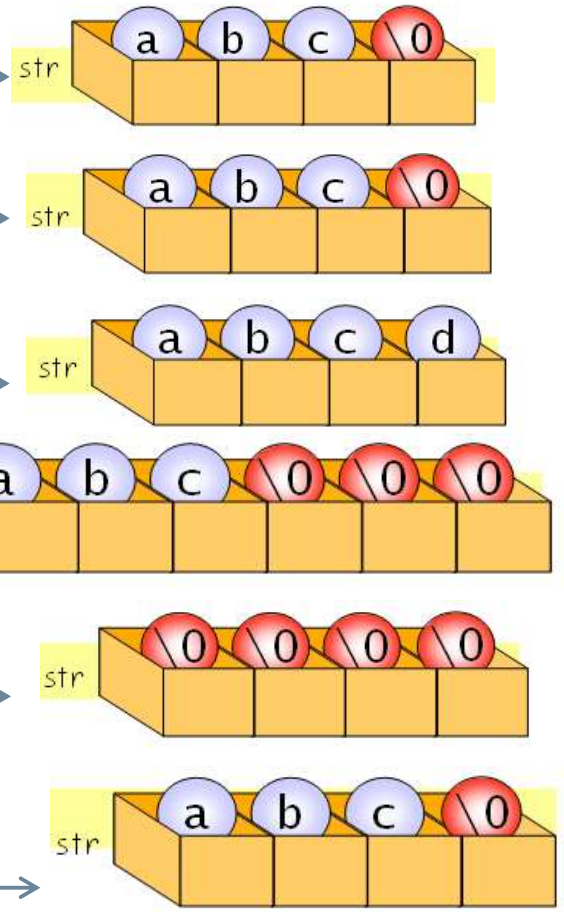
string1.c

```
#include <stdio.h>
int main(void)
{
 int i;
 char str[4];
 str[0] = 'a';
 str[1] = 'b';
 str[2] = 'c';
 str[3] = '\0';

 i = 0;
 while(str[i] != '\0') {
       printf("%c", str[i]);
       i++;
 }
 return 0;
}
```

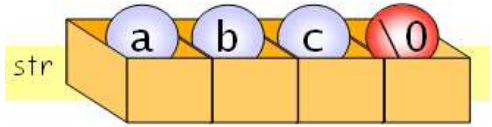
문자 배열에 들어 있는 문자들을 하나씩 출력하여 보자. 문자 배열에 저장된 문자열을 출력할 때는 %s를 사용하면 되지만 여기서는 문자열 처리의 기본적인 방법을 실감하기 위하여 문자 배열에 들어 있는 문자들을 하나씩 화면에 출력하다가 NULL 문자가 나오면 반복을 종료하도록 하였다.

문자 배열의 초기화

- char str[4] = { 'a', 'b', 'c', '\0' };
- char str[4] = "abc“;
- char str[4] = "abcdef";
- char str[6] = "abc";
- char str[4] = "";
- char str[] = "abc";



문자열의 출력



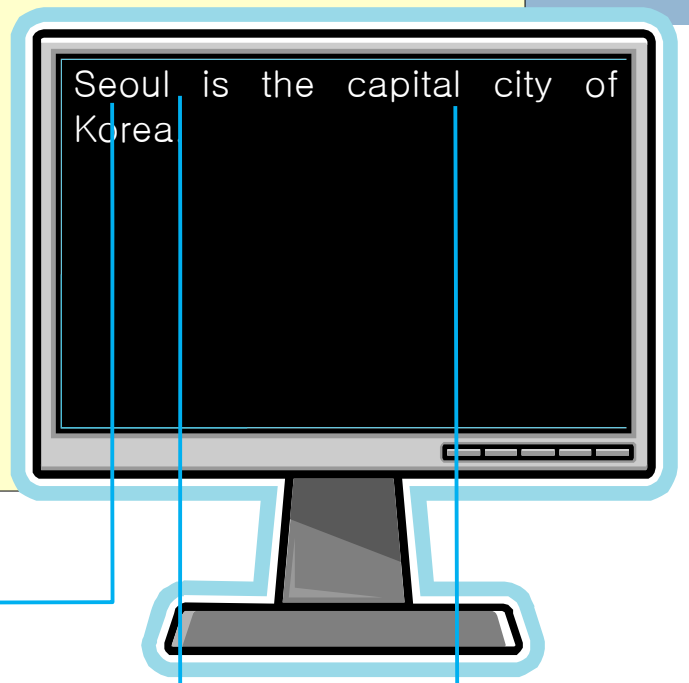

```
char str[] = “abc”;
printf(“%s”, str);
```

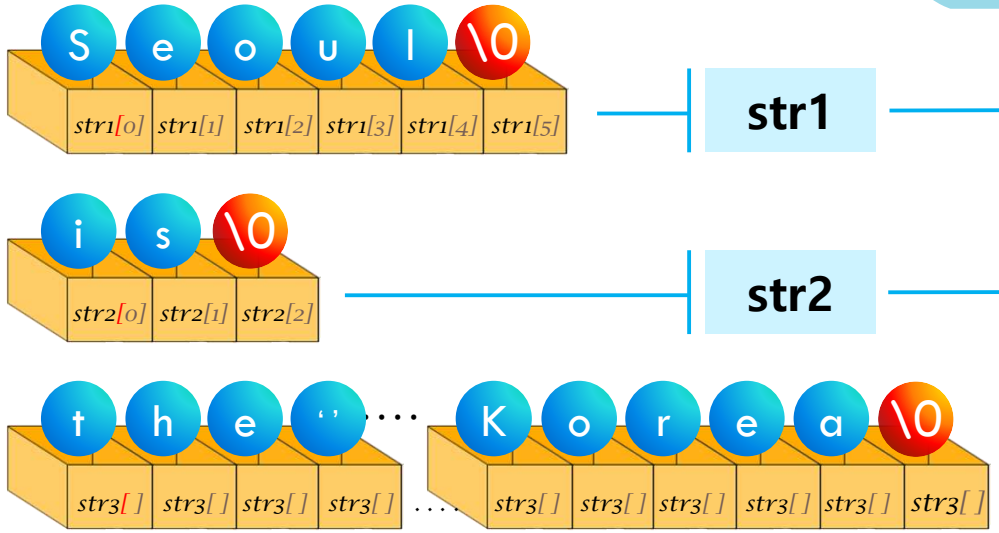
```
char str[] = “abc”;
printf(str);
```

예제 #2

string2.c

```
#include <stdio.h>

int main(void)
{
    char str1[6] = "Seoul";
    char str2[3] = { 'i', 's' , '\0'};
    char str3[] = "the capital city of Korea.";
    printf("%s %s %s\n", str1, str2, str3);
    return 0;
}
```





예제 #3

string3.c

```
#include <stdio.h>
int main(void)
{
        char src[] = "The worst things to eat before you sleep";
        char dst[100];
        int i;
        printf("원본 문자열=%s\n", src);
        for(i=0 ; src[i] != '\0' ; i++)
                dst[i] = src[i];
        dst[i] = '\0';
        printf("복사된 문자열=%s\n", dst);
        return 0;
}
```

NULL과 '\0'은 같다.

문자열 길이 계산 예제

string4.c

```c
// 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.
```

문자 배열을 실행 시간에 변경

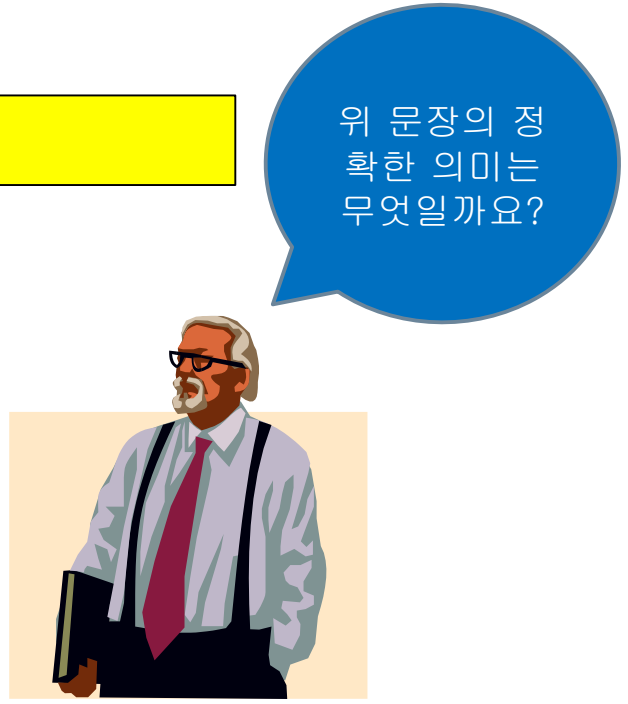
1. 문자 배열의 각각의 원소를 개별적으로 변경
   - str[0] = 'W';
   - str[1] = 'o';
   - str[2] = 'r';
   - str[3] = 'l';
   - str[4] = 'd';
   - str[5] = '\0';

2. strcpy()를 사용하여 문자열을 문자 배열에 복사
   - strcpy(str, "World");

문자열 상수

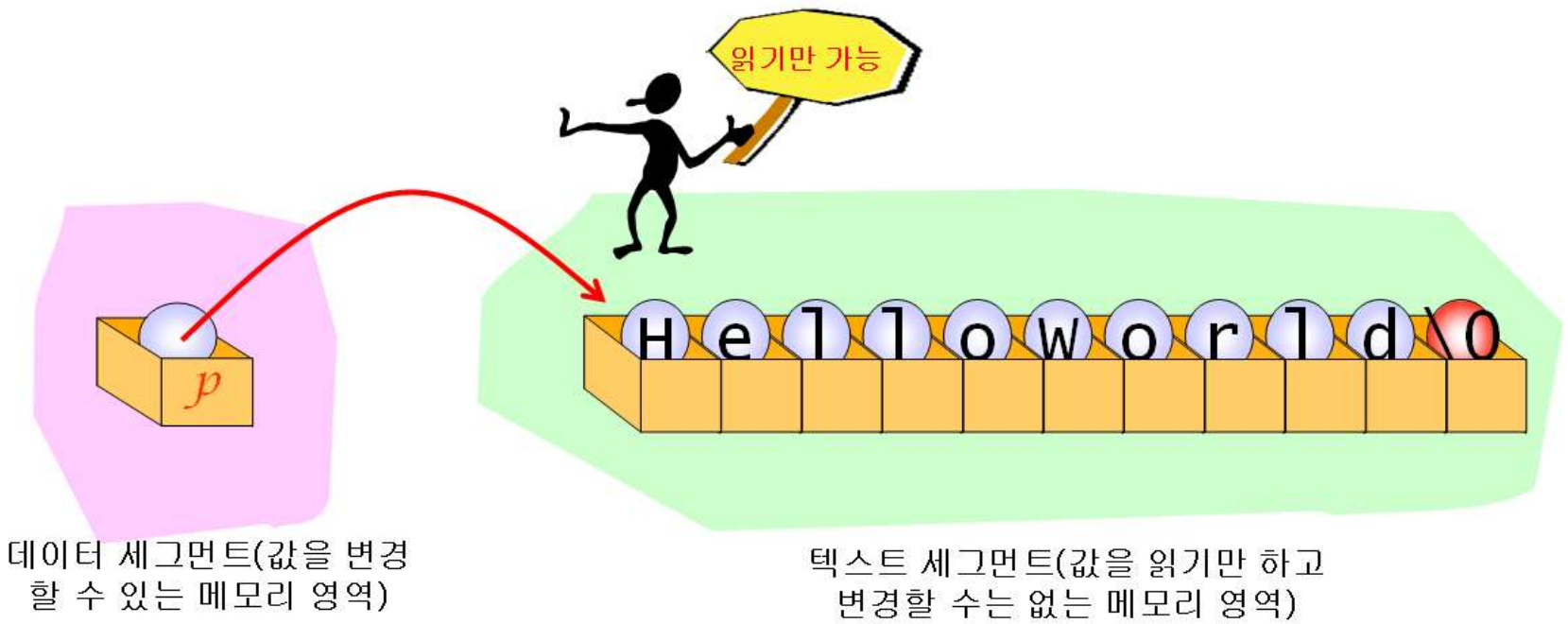
- 문자열 상수: “HelloWorld”와 같이 프로그램 소스 안에 포함된 문자열
- 문자열 상수는 메모리 영역 중에서 **텍스트 세그먼트(text segment)** 에 저장

```
char *p = "HelloWorld";
```



문자열 상수

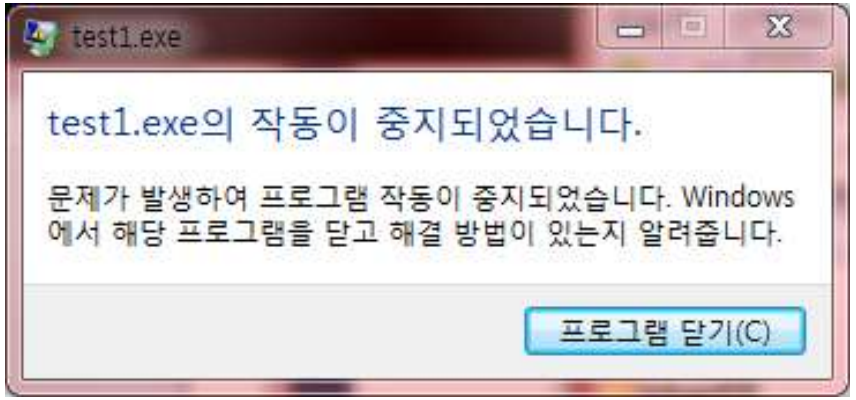

```
char *p = "HelloWorld";
```



문자열 상수

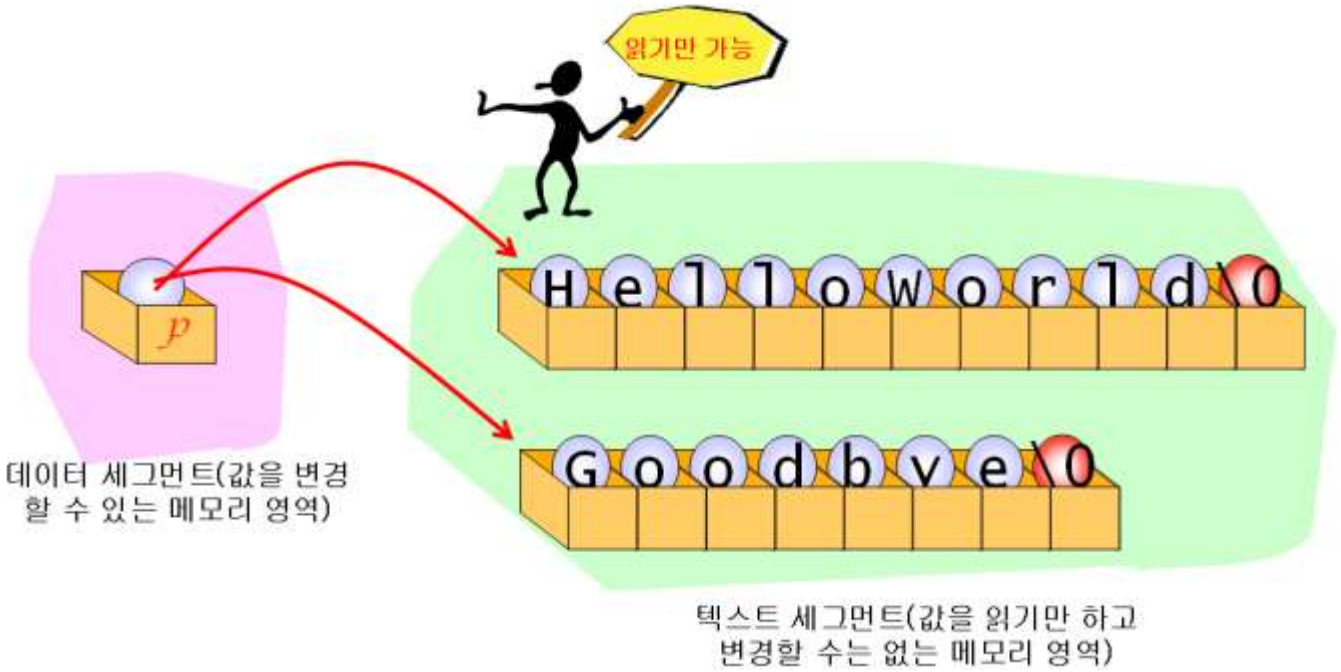

```
char *p = "HelloWorld";
strcpy(p, “Goodbye”);
```

p를 통하여 텍스트 세그먼트에 문자를 저장하려면 오류가 발생한다.



문자열 상수


```
char *p = "HelloWorld";
p = "Goodbye";
```



예제

stringconst.c

```
#include <stdio.h>

int main(void)
{
   char *p = "HelloWorld";
   printf("%s \n", p);

   p = "Welcome to C World!"; // 가능
   printf("%s \n", p);

   p = "Goodbye"; // 가능
   printf("%s \n", p);
   // p[0] = 'a'; // 오류가 발생한다.

   return 0;
}
```

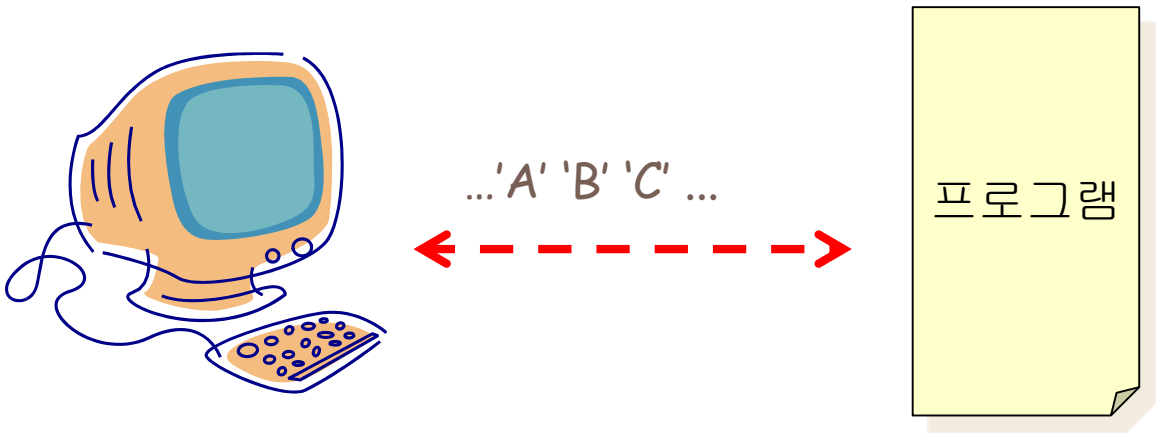
```
HelloWorld
Welcome to C World!
Goodbye
```

중간 점검

- C에서 문자열은 어떻게 정의되는가?
- 문자열에서 NULL 문자의 역할은 무엇인가?
- NULL 문자의 아스키 코드 값은 얼마인가?
- NULL 문자로 끝나지 않는 문자열을 출력하면 어떻게 되는가?
- B, 'B', "B"의 차이점을 설명하라.
- 변경 가능한 문자열은 어디에 저장되는가?
- 문자열의 크기보다 문자 배열의 크기를 하나 더 크게 하는 이유는 무엇인가?
- 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()

getchar.c

```
// getchar()의 사용
#include <stdio.h>
int main(void)
{
        int ch;          // 정수형에 주의
        while( (ch = getchar()) != EOF )
                putchar(ch);
        return 0;
}
```

End Of File을 나타내는 문자, EOF는 정수형이다.

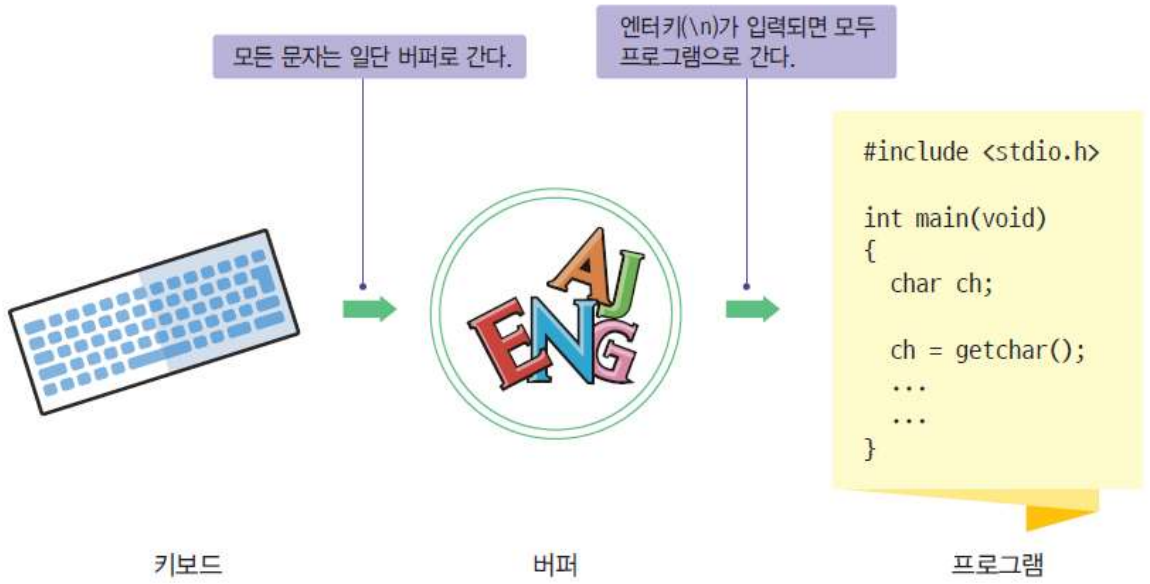
```
a
a
b
b
^Z
```

getchar() 출력

putchar() 출력

버퍼링

- 엔터키를 쳐야만 입력을 받는 이유



_getch(), _putch()

getch.c

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

int main(void)
{
        int ch;
        while( (ch = _getch()) != 'q' )
                _putch(ch);
        return 0;
}
```

버퍼를 사용하지 않는다,
에코도 없음!

```
abc
```

_getch(), _getche(), getchar()

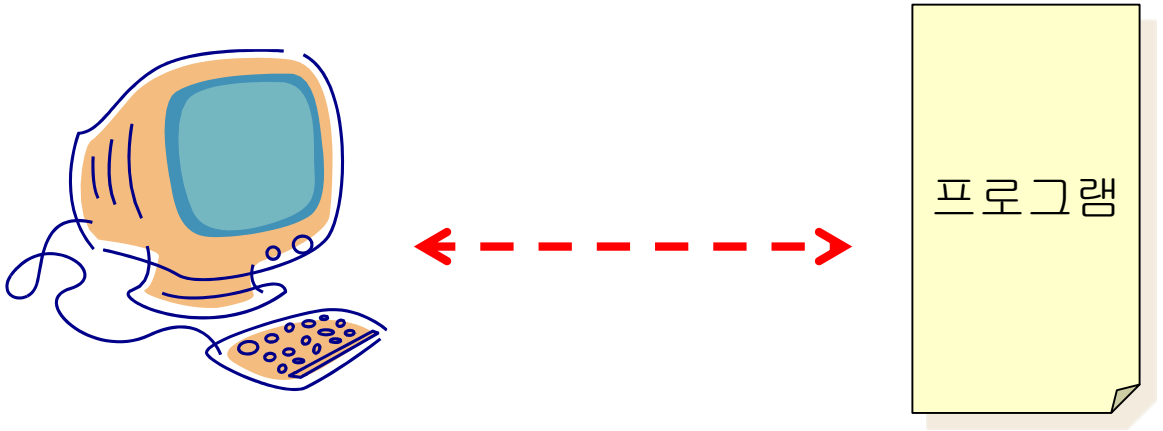
	헤더파일	버퍼사용여부	에코여부	응답성	문자수정여부
getchar()	<stdio.h>	사용함 (엔터키를 눌러입력됨)	에코	줄단위	가능
_getch()	<conio.h>	사용하지 않음	에코하지 않음	문자단위	불가능
_getche()	<conio.h>	사용하지 않음	에코	문자단위	불가능

중간 점검

- getchar()와 _getch()가 다른 점은 무엇인가?
- 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_s(char *s, int size)	빈칸을 포함한 한 줄의 문자열을 읽어서 문자 배열 s[]에 저장한다.
int puts(const char *s)	배열 s[]에 저장되어 있는 한 줄의 문자열을 출력한다.



gets_s()와 puts() 문자열 입출력

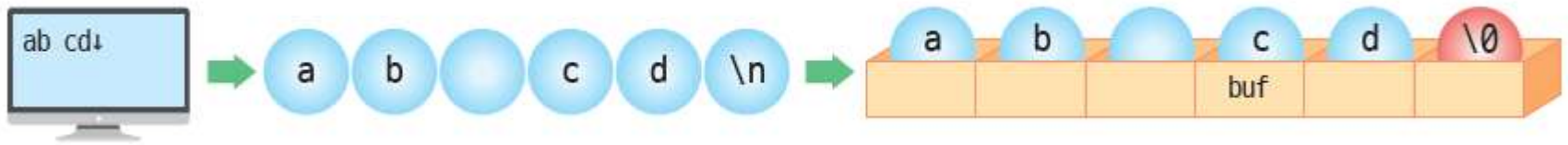
Syntax: getchar()

예

```
char buf[100];
gets_s(buf, 100);
puts(buf);
```

사용자로부터 한 줄을 입력받는다.
최대 입력 개수는 100이다.

한 줄을 출력한다.

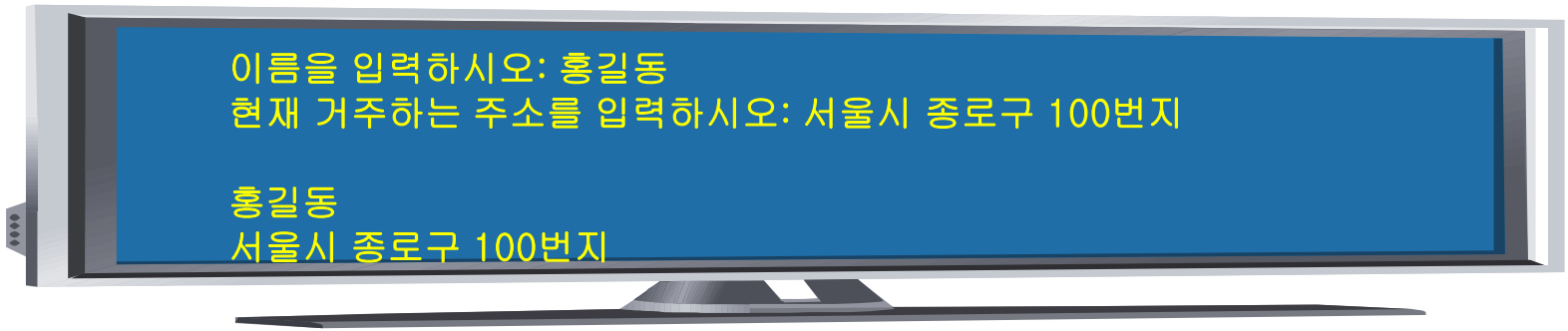


예제

gets.c

```c
#include <stdio.h>
int main(void)
{
        char name[100];
        char address[100];
        printf("이름을 입력하시오: ");
        gets_s(name, 100);
        printf("현재 거주하는 주소를 입력하시오: ");
        gets_s(address, 100);
        puts(name);
        puts(address);
        return 0;
}
```

한 단어 이상을 입력 받을 때에 사용한다.



중간 점검

- 빈칸을 포함한 한 줄의 텍스트를 입력받는 문장을 작성하라.
- 사용자로부터 하나의 단어를 입력받는 문장을 작성하라.

문자 처리 라이브러리 함수

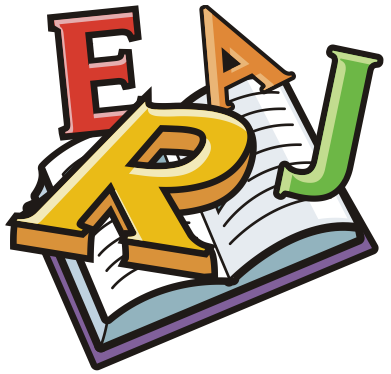
- 문자를 검사하거나 문자를 변환한다.

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.



예제

char_process.c

```
#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
```

EOF를 키보드에서 입력하려면 ^Z

lab: 단어 세기

- 문자열 안에 들어 있는 단어의 개수를 세는 프로그램을 작성하여 보자. 문자열이 “the c book... ” 이라면 다음과 같은 출력이 나온다.

예제

wc.c

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

int count_word(char *s);
int main( void )
{
        int wc = count_word("the c book...");
        printf("단어의 개수: %d \n", wc);

        return 0;

}
```

예제

wc.c

```
int count_word (  char * s )
{
        int i, wc = 0, waiting = 1;

        for( i = 0; s[i] != NULL; ++i)            // s의 각 글자 조사
                if( isalpha(s[i]) )               // s의 글자가 알파벳이면
                {
                        if( waiting )    // 단어를 기다리고 있으면
                        {
                                wc++;    // 카운터를 증가
                                waiting = 0;// 단어를 처리하는 중
                        }
                }
                else                              // 알파벳이 아니면
                        waiting = 1;     // 단어를 기다린다.


        return wc;
}
```

중간 점검

- 문자 처리 라이브러리 함수를 사용하려면 포함시켜야 하는 헤더 파일은 무엇인가?
- getchar()와 getch()가 다른 점은 무엇인가?
- ispunct('.')의 반환값은 무엇인가?
- 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

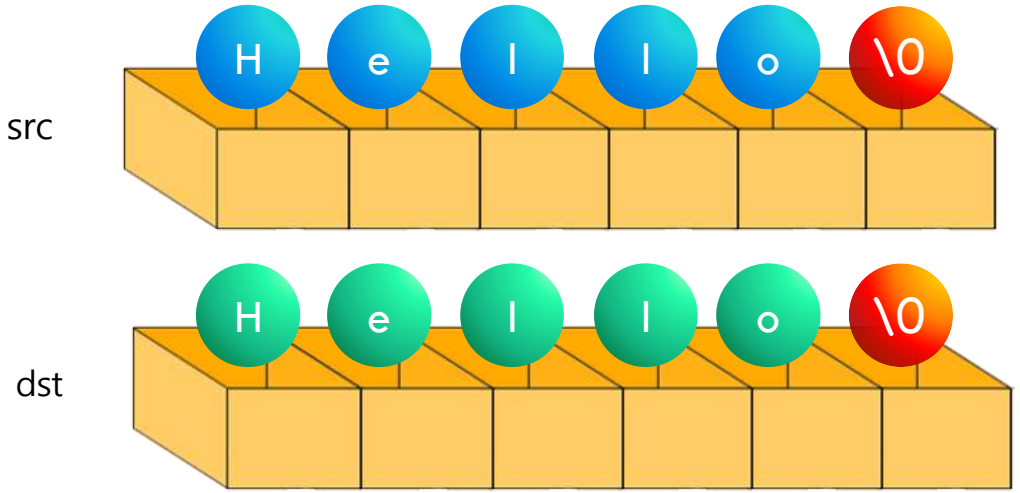
- 문자열 길이

```
strlen(char *str)
char str = “Hello”;
len = strlen(str);
len = strlen(“abcdef”);
```

문자열 복사

- 문자열 복사

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



문자열 연결

Syntax: strcat()

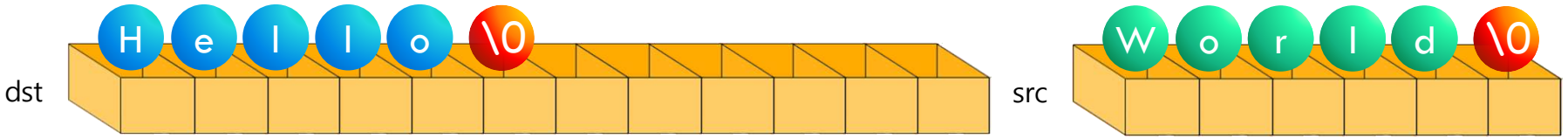
예

```
char dst[12]= "Hello";
char src[6] = "World";
strcat(dst, src);       // dst가 "HelloWorld"가 된다.
```

문자열 연결

- 문자열 연결

```
char dst[12] = "Hello";
char src[6] = "World";
strcat(dst, src);
```



예제

strcpy_strcat.c

```c
// 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

Syntax: strcmp()

```
예    int result = strcmp("dog", "dog");      // 0이 반환된다.
```

strcmp()는 문자열 s1과 s2를 비교하여 사전적인(lexicographic) 순서에서 s1이 앞에 있으면 음수가 반환되고, 같으면 0이, 뒤에 있으면 양수가 반환된다.

반환값	s1과 s2의 관계
< 0	s1이 s2보다 앞에 있다.
0	s1 == s2
> 0	s1이 s2보다 뒤에 있다.

예제

strcmp.c

```c
// 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;
}
```

알파벳 순으로 즉 사전에서 앞에 있다는 의미

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

문자 검색

Syntax: strchr()

예

```
char *p = strchr("dog", 'g');
```

'g' 문자의 주소를 반환한다.

문자 검색

strchr.c

```
#include <string.h>
#include <stdio.h>
int main( void )
{       char s[] = "language";
        char c = 'g';
        char *p;
        int loc;

        p = strchr(s, c);
        loc = (int)(p - s);
        if ( p != NULL )
                printf( "첫번째 %c가 %d에서 발견되었음\n", c, loc );
        else
                printf( "%c가 발견되지 않았음\n", c );
        return 0;
}
```

s 안에서 문자 c를 찾는다.

```
첫번째 g가 3에서 발견되었음
```

문자열 검색

Syntax: strchr()

예

```
char *p = strstr("dog", "og");
```

strstr() 함수는 문자열 s안에서 부분 문자열(substring) sub를 검색하는 함수이다. 만약 부분 문자열이 발견되면 그 위치의 주소를 반환한다. 만약 부분 문자열을 찾지 못하면 NULL값이 반환된다.

문자열 검색

strstr.c

```c
#include <string.h>
#include <stdio.h>
int main( void )
{
        char s[] = "A joy that's shared is a joy made double";
        char sub[] = "joy";
        char *p;
        int loc;
        p = strstr(s, sub);
        loc = (int)(p - s);
        if ( p != NULL )
                printf( "첫번째 %s가 %d에서 발견되었음\n", sub, loc );
        else
                printf( "%s가 발견되지 않았음\n", sub );
}
```

s 안에서 문자열 sub를 찾는다.

```
첫번째 joy가 2에서 발견되었음
```

문자열 토큰 분리

Syntax:

예

```
char s[] = "Hello World";
char delimit[] = " ";
char *p = strtok(s, delimit);
```

문자열을 스페이스문자를 사용하여 단어들로 분리한다.

문자열 토큰 분리

strtok.c

```
// strtok 함수의사용예
#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 ); //
   }
}
```

분리자

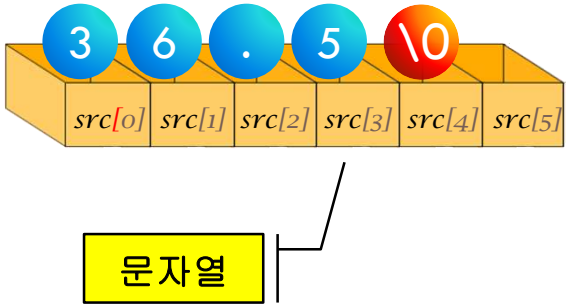
```
토큰: Man
토큰: is
토큰: immortal
토큰: because
토큰: he
토큰: has
토큰: a
토큰: soul
```

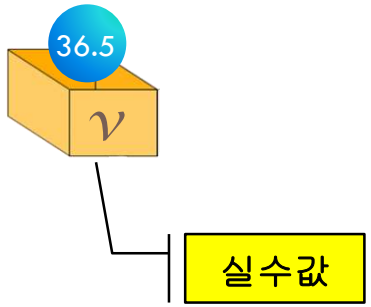
중간 점검

- 문자열 s1를 문자열 s2로 복사하는 문장을 써라.
- “String"을 저장하려면 최소한 어떤 크기 이상의 문자 배열이 필요한가?
- 문자열을 서로 비교하는 함수는?
- strcpy()와 strncpy()의 차이점은 무엇인가?
- s1[]에 저장된 문자열 뒤에 s2[]를 붙이고 싶으면 어떤 라이브러리 함수를 어떻게 사용하여야 하는가?
- strcmp("dog", "dog")의 반환값은 얼마인가?

문자열 수치 변환

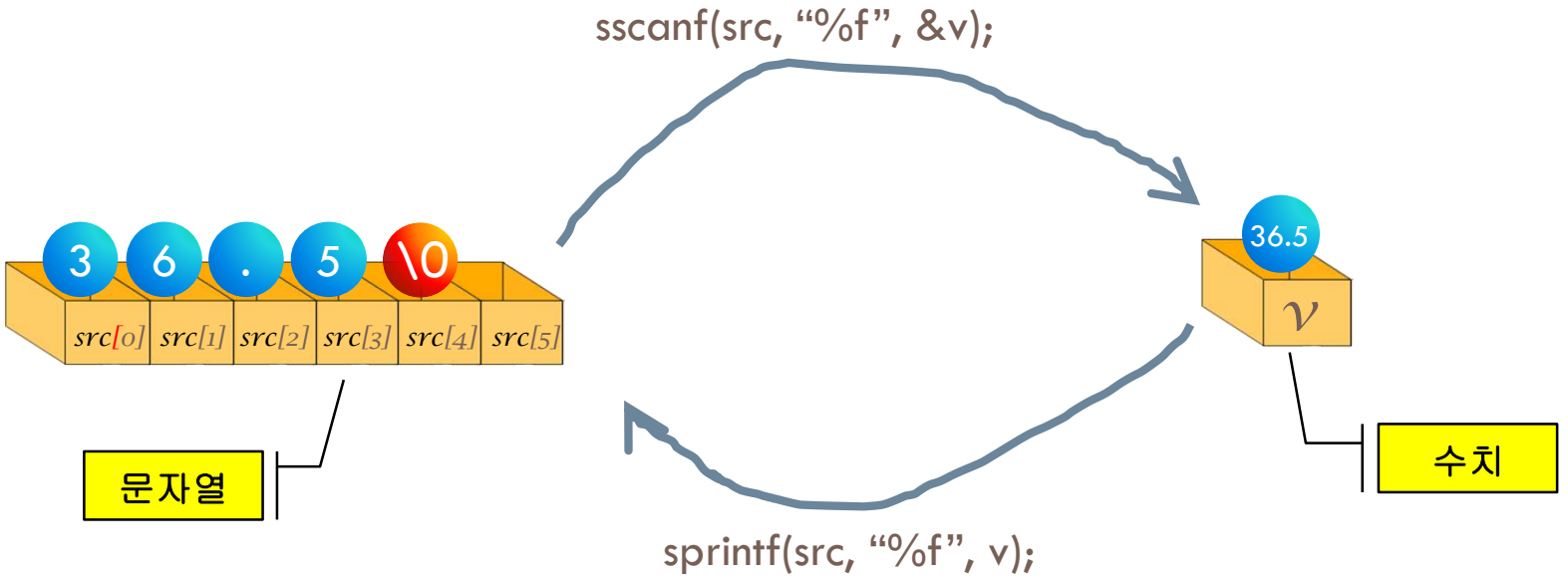
- 문자열 “36.5”와 수치값 36.5는 컴퓨터 안에서 상당히 다르게 저장된다.





sprintf()와 sscanf()

- 앞에 붙은 s는 string 을 의미한다.



예제

sscanf.c

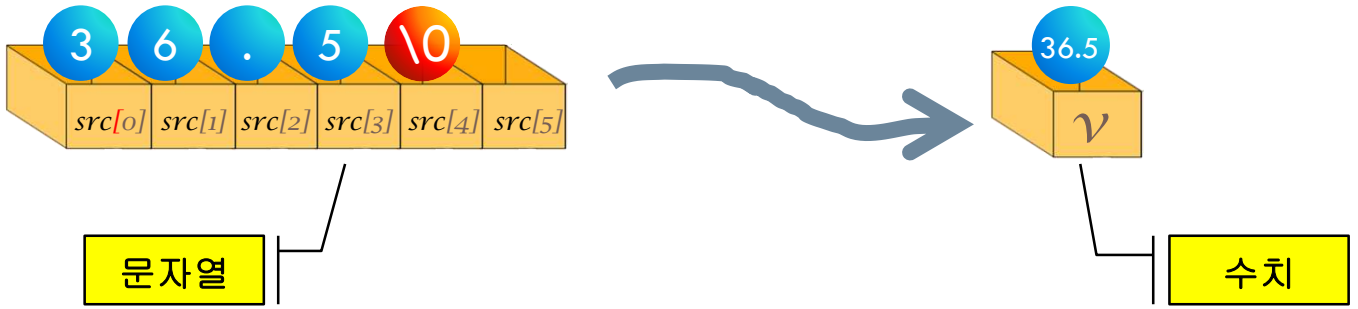
```
#include <stdio.h>

int main( void )
{
        char s[] = "100";
        int value;

        sscanf(s, "%d", &value);
        printf("%d \n", value);
        value++;
        sprintf(s, "%d", value);
        printf("%s \n", s);
        return 0;
}
```

```
100
101
```

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 변환

atoi.c

```c
#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.
```

중간 점검

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

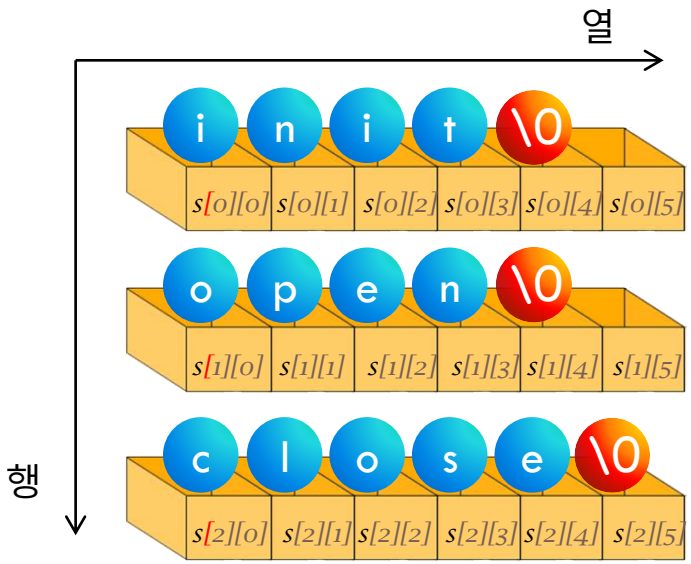
문자열의 배열

- (Q)문자열이 여러 개 있는 경우에는 어떤 구조를 사용하여 저장하면 제일 좋을까? (예) “init”, “open”, “close”, …

1. 문자열의 배열
2. 문자 포인터 배열

1. 문자열의 배열


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



2. 문자 포인터 배열


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

예제

stringarray1.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;
}
```

2차원 배열로 입력

stringarray2.c

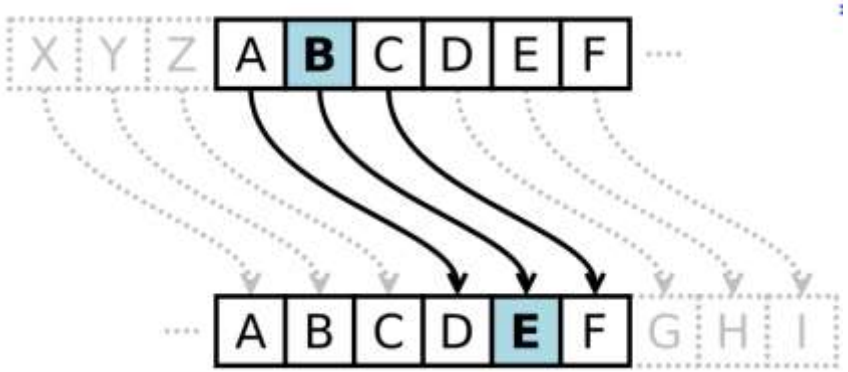
```
#include <stdio.h>
int main( void )
{
        int i;
        char fruits[3][20];
        for(i = 0; i < 3; i++) {
                printf("과일 이름을 입력하시오: ");
                scanf("%s", fruits[i]);
        }
        for(i = 0; i < 3; i++)
                printf("%d번째 과일: %s\n", i, fruits[i]);
        return 0;
}
```

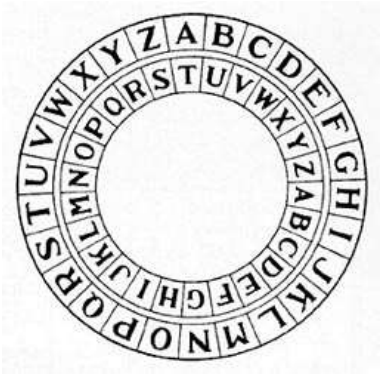
앞에 &을 붙이면 안됨!

```
과일 이름을 입력하시오: 사과
과일 이름을 입력하시오: 배
과일 이름을 입력하시오: 포도
0번째 과일: 사과
1번째 과일: 배
2번째 과일: 포도
```

lab: 메시지 암호화

- 메시지를 암호화하는 간단한 기법 중의 하나는 줄리어스 시저가 사용한 암호화 기법
- 평문에 단순히 더하기(즉, 영어의 알파벳을 왼쪽으로 이동하던지 오른쪽으로 이동하는 것)





실행 결과

코드

encrypt.c

```c
#include <stdio.h>

void encrypt(char cipher[], int shift);

int main (void) {
        char cipher[50];
        int shift=3;
        printf("문자열을 입력하시오: ");
        gets(cipher); // 한줄 전체 입력
        encrypt (cipher, shift);
        return 0;
}
```

코드


```
void encrypt (char cipher[], int shift) {
        int i = 0;
        while (cipher[i] != '\0') {
                if( cipher[i] >= ‘a' && cipher[i] <= 'z'){
                        cipher[i] += shift;
                        if( cipher[i] > 'z' )
                                cipher[i] -= 26;
                }
                i++;
        }
        printf("암호화된 문자열: %s", cipher);
}
```

아스키 코드값을 이동한다.

Q & A