String

String Input/Output Functions

#include ⟨stdio.h⟩

formatted input/output functions

- scanf/fscanf
- printf/fprintf

special set of string-only functions

- get string (gets/fgets)
- put string (puts/fputs).

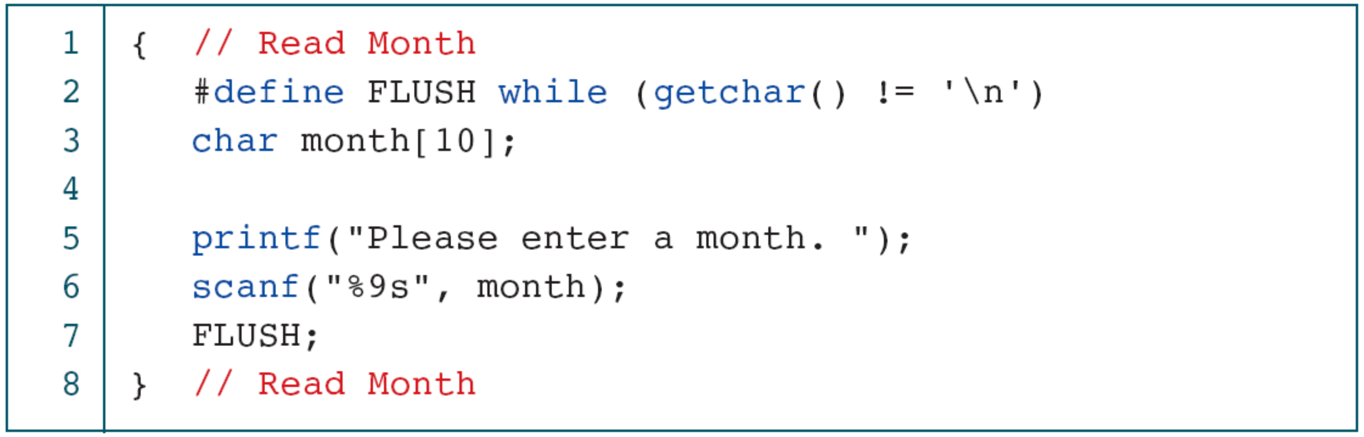
FLUSH

- 스트림 버퍼를 비우는 역할을 한다.
- Scanf 등 입력받는 함수 사용 시

→ The string conversion code(s) skips whitespace.

```
ex)
int a;  char b;
scanf(“%d”, &a);
scanf(“%c”,&b);
printf(“%d %c\n”, a, b)

fflush();
#define FLUSH while(getchar != ‘\n’)
```



```
{   // Read Month
    #define FLUSH while (getchar() != '\n')
    char month[10];

    printf("Please enter a month. ");
    scanf("%9s", month);
    FLUSH;
}   // Read Month
```

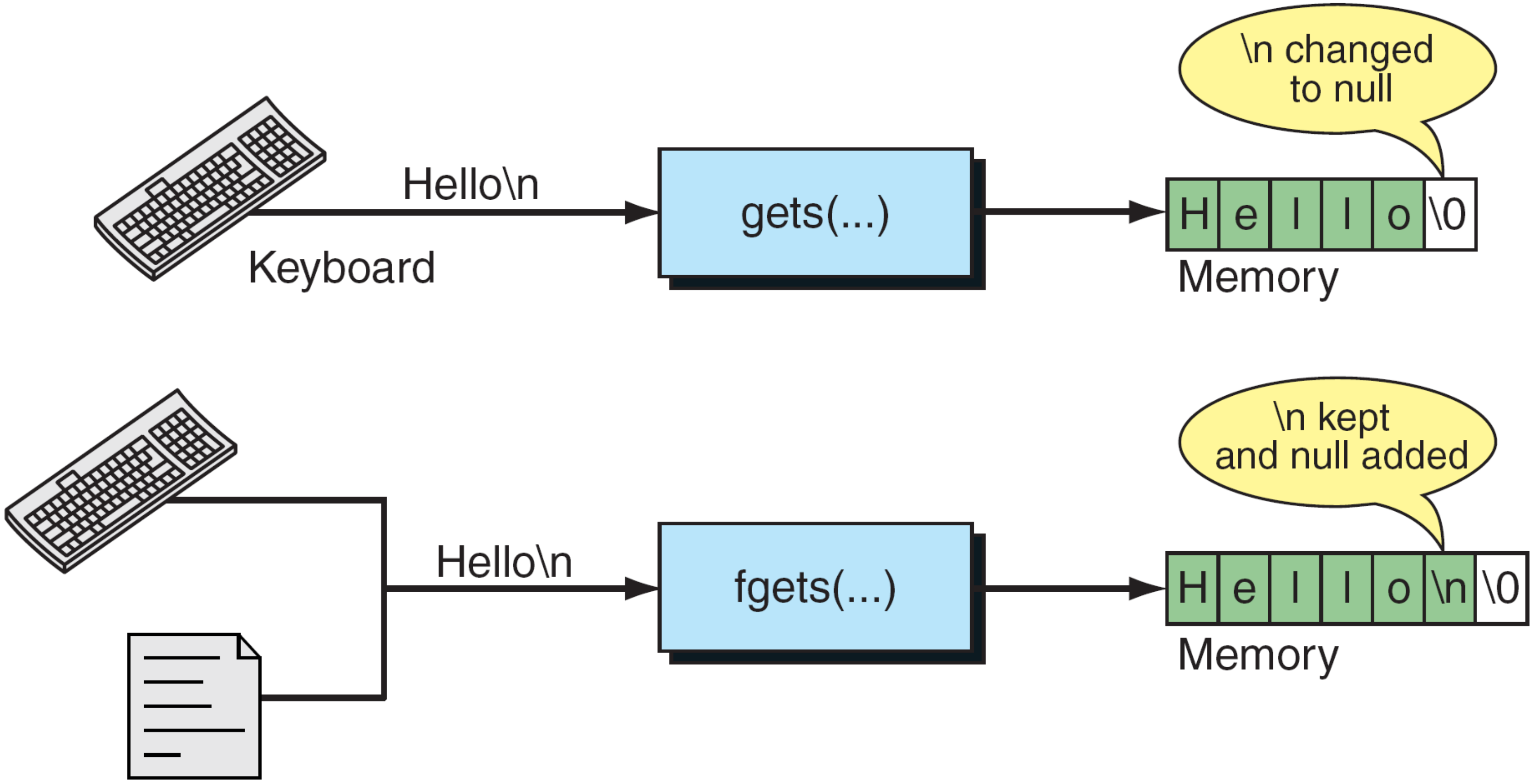
Formatted String Input

```
char str[10];
scanf("%9s", str);
```

- str – 배열 포인터 → &가 붙지 않음

- 입력 문자 개수를 반드시 정의해 주자

String-only Input: gets/fgets



gets(…)

char * gets(char *str);

Reads characters from stdin, until either a newline or EOF

Params

- str: pointer to an array of chars

→ 길이 제한 없음 (보안상 취약)

Return value

- Success: returns str (incl. ‘₩0’ at the end)
- EOF: feof() set, 여태까지 읽은 str return
- 아무것도 못 읽으면: NULL

fgets(…)

char * fgets (char *str, int num, FILE *stream);

Reads characters from stream, until (num-1) characters have been read OR either a newline or the EOF

Params

- str : pointer to an array of chars
- num : Maximum number of characters (incl. null) → 문자는 최대 (num-1)
- stream : Pointer to a FILE that identifies an input stream

Return value

- Success: returns str (incl. '₩0' at the end)
- EOF: sets EOF(feof), 여태까지 읽은 str 리턴
- 아무것도 못 읽으면: NULL

Difference btw. Input functions

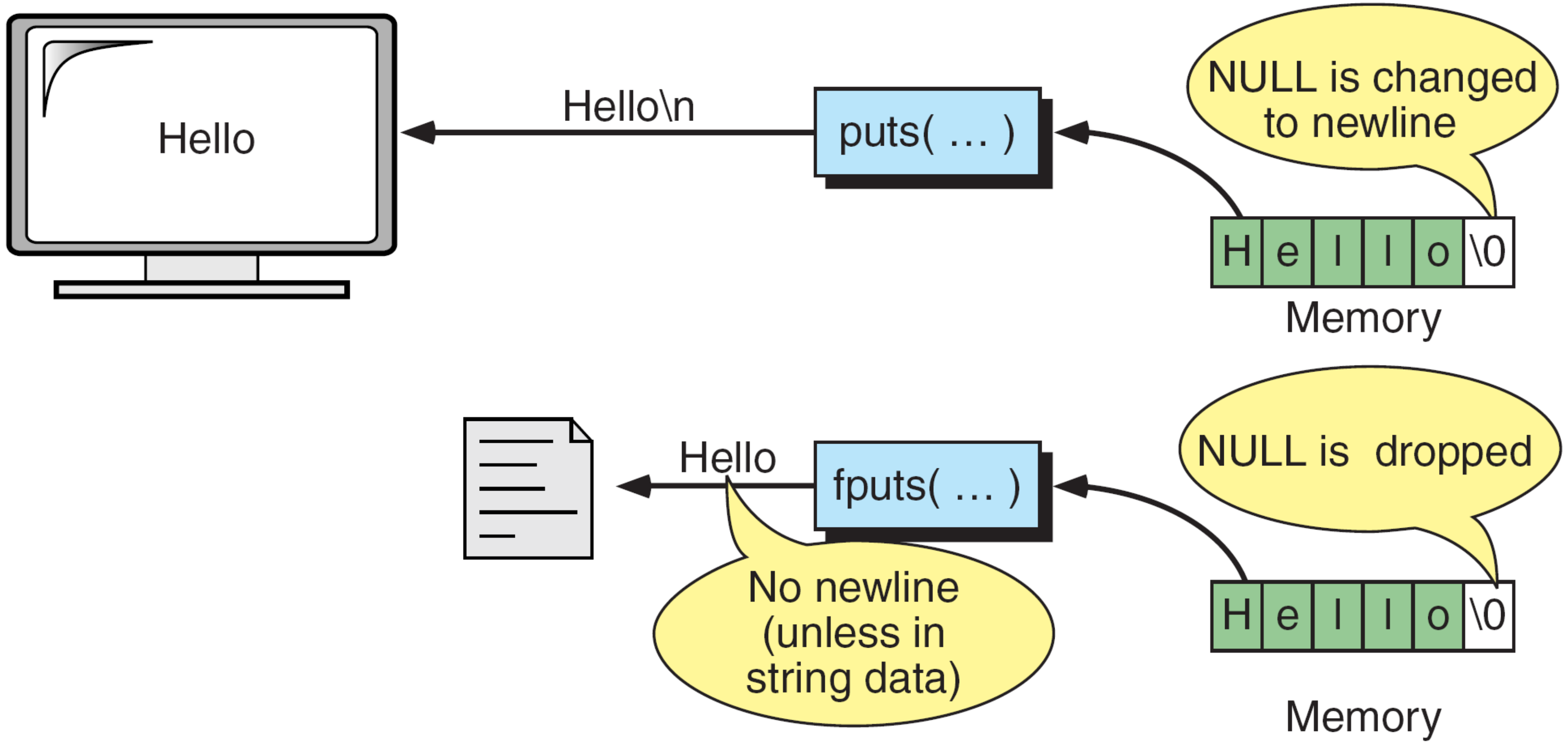
gets/fgets

- gets does not include newline (‘₩n’), fgets does.
- gets does not allow to specify a maximum size for *str* (which can lead to buffer overflows).

Scanf/gets

- scanf는 단어 단위로(space), gets는 줄 단위로(newline)

String-only Output: puts/fputs



puts(…)

int puts (const char * str);

Writes str to stdout and appends a newline character ('₩n') until null('0₩').
Terminating null-character is not copied to the stream.

Params

- str: pointer to an array of chars

Return value

- Success: returns a non-negative value
- Error: returns EOF(-1) and sets the error indicator (ferror)

fputs(…)

int fputs (const char * str, FILE * stream);

Writes *str* to the stream until null ('₩0'). Terminating null-character is not copied to the stream.

Params

- str: pointer to an array of chars
- stream: Pointer to a FILE object that identifies an output stream.

Return value

- Success: returns a non-negative value
- Error: returns EOF(-1) and sets the error indicator (ferror)

Difference btw. Output functions

- puts appends a newline(‘₩n) at the end automatically
- fputs does not write additional characters

입력한 문자열에서 입력한 문자 개수 세기

```
#include <stdio.h>
main()
{
    char str[30],ch;
    int i=0, cnt=0;
// scanf("%s",str); // scanf쓰면 apple의 마지막엔터가 ch에 들어가므로 쓰면안됨
    gets(str);
// scanf("%c", &ch); // 한문자입력은  scanf, getchar 모두 가능
    ch=getchar();
    while(str[i]!='₩0') {
            if(str[i++]==ch)
                    cnt++;
    }
    printf("%s에서 ₩'%c₩'문자가 %d개 입니다.₩n", str, ch, cnt);
}
```

String Manipulation Functions

- #include ⟨string.h⟩

- String Length and String Copy
- String Compare and String Concatenate
- Character in String
- Search for a Substring and Search for Character in Set
- String Span and String Token
- String to Number

String Length

unsigned int strlen (const char * str);

Returns the length of the C string *str*, until null(‘₩0’).

Params

- str: pointer to an array of chars

Return value

- The length of string.

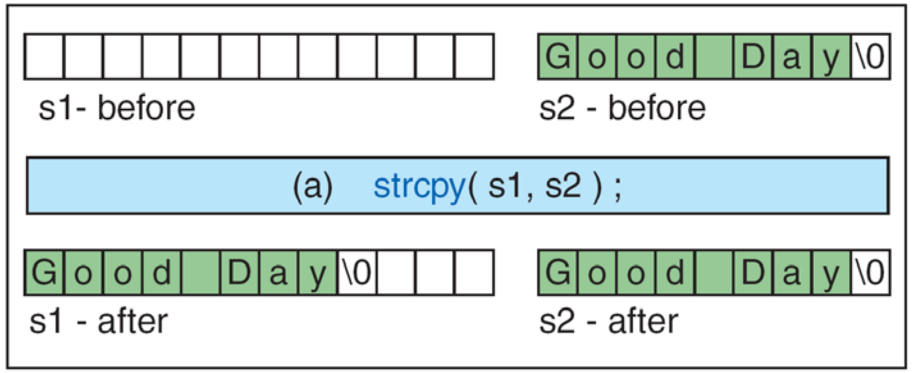
Example

- char mystr[100]="test string";

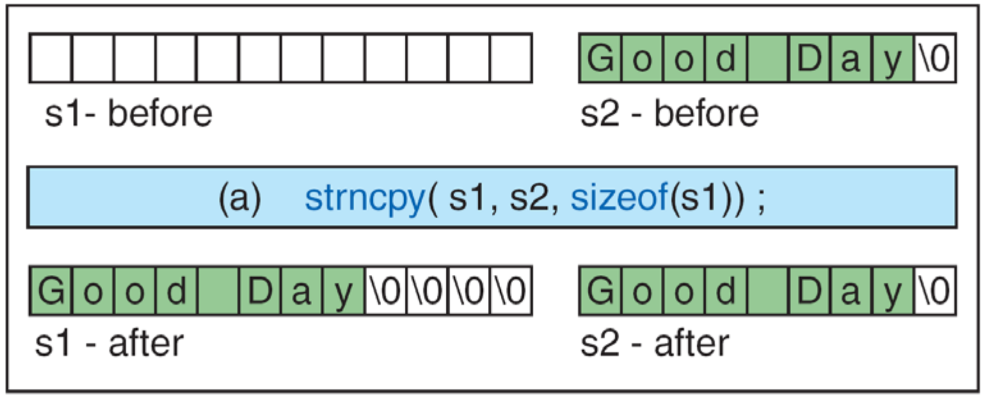
- sizeof(mystr) evaluates to 100
- strlen(mystr) returns 11.

String Copy

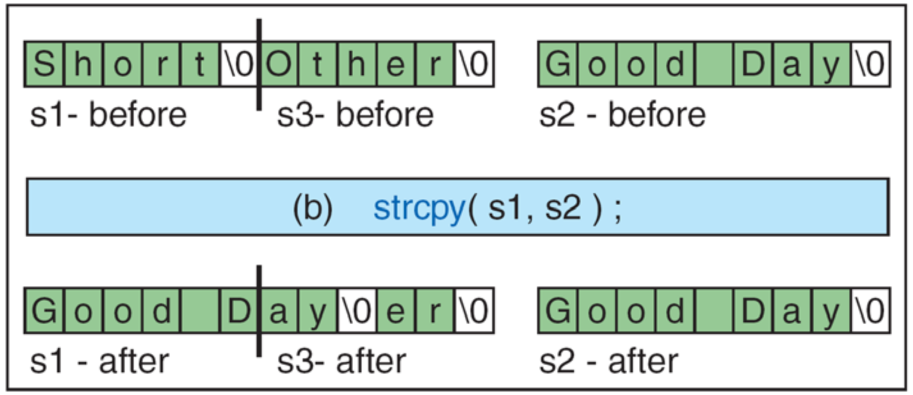
```
char * strcpy ( char * destination, const char * source );
char * strncpy ( char * destination, const char * source, unsigned int num );
```



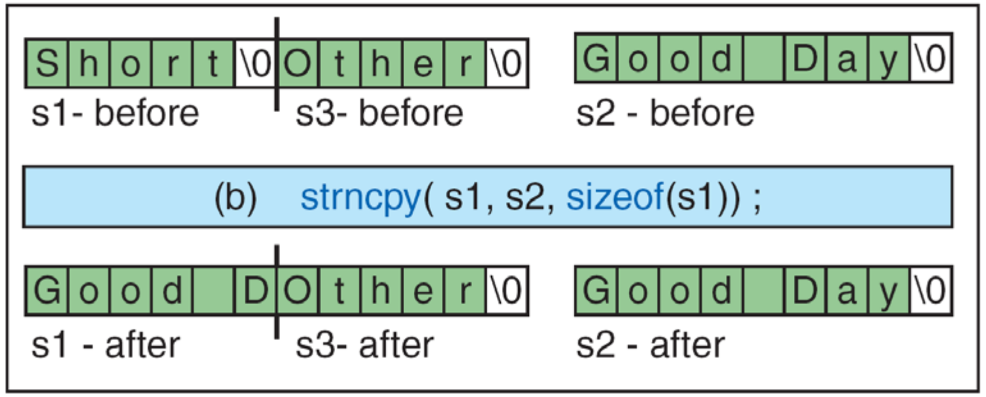
Copying Strings



Copying Strings

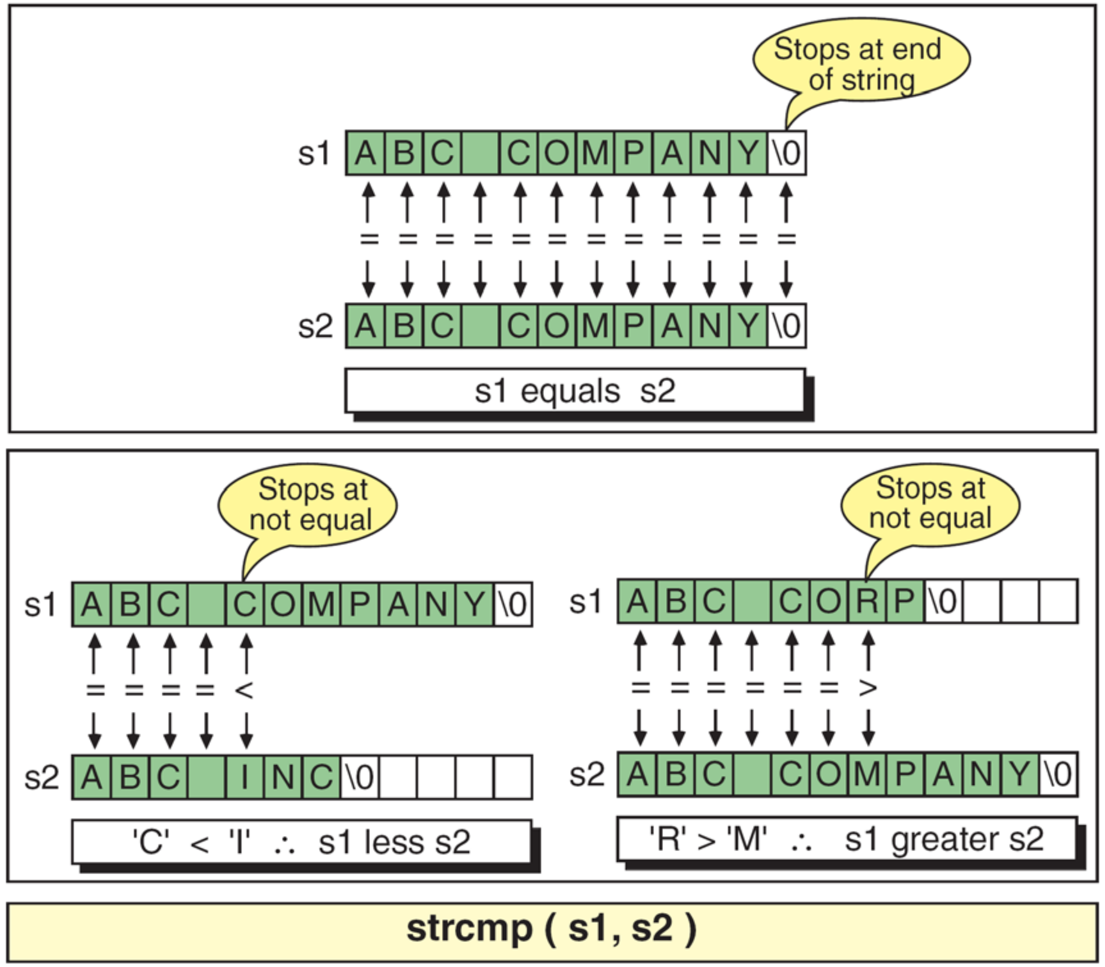


Copying Long Strings



Copying Long Strings

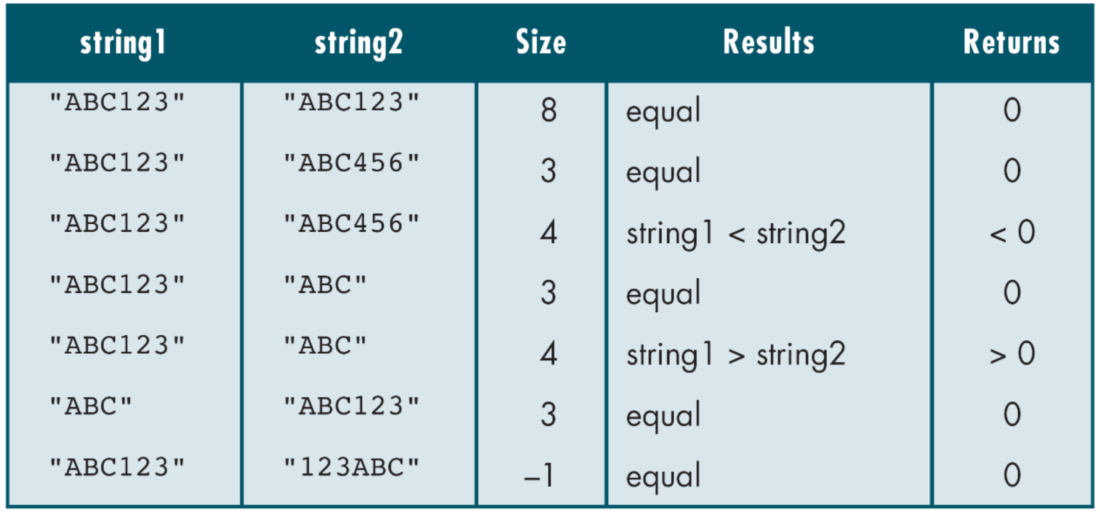
String Compare



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

Results for String Compare

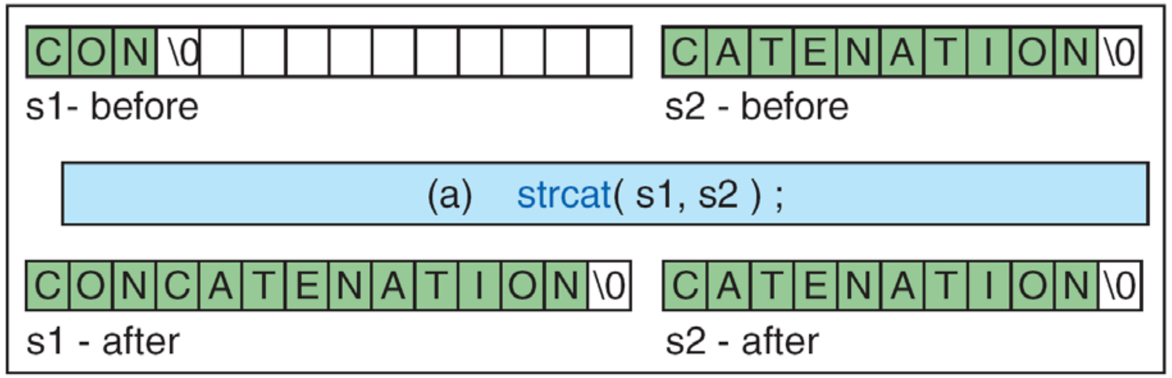
- 문자 비교 – 아스키 코드 순으로



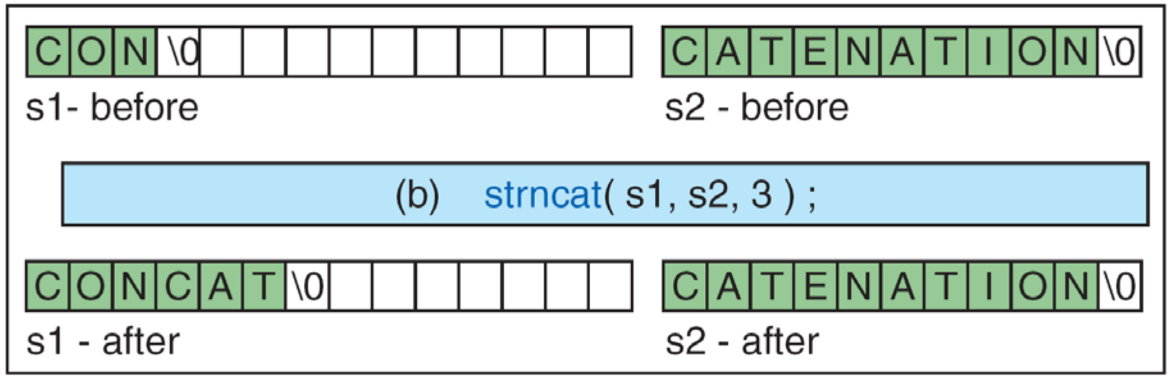
string1	string2	Size	Results	Returns
"ABC123"	"ABC123"	8	equal	0
"ABC123"	"ABC456"	3	equal	0
"ABC123"	"ABC456"	4	string1 < string2	< 0
"ABC123"	"ABC"	3	equal	0
"ABC123"	"ABC"	4	string1 > string2	> 0
"ABC"	"ABC123"	3	equal	0
"ABC123"	"123ABC"	–1	equal	0

String Concatenation

```
char * strcat ( char * destination, const char * source );
char * strncat ( char * destination, const char * source, unsigned int num );
```



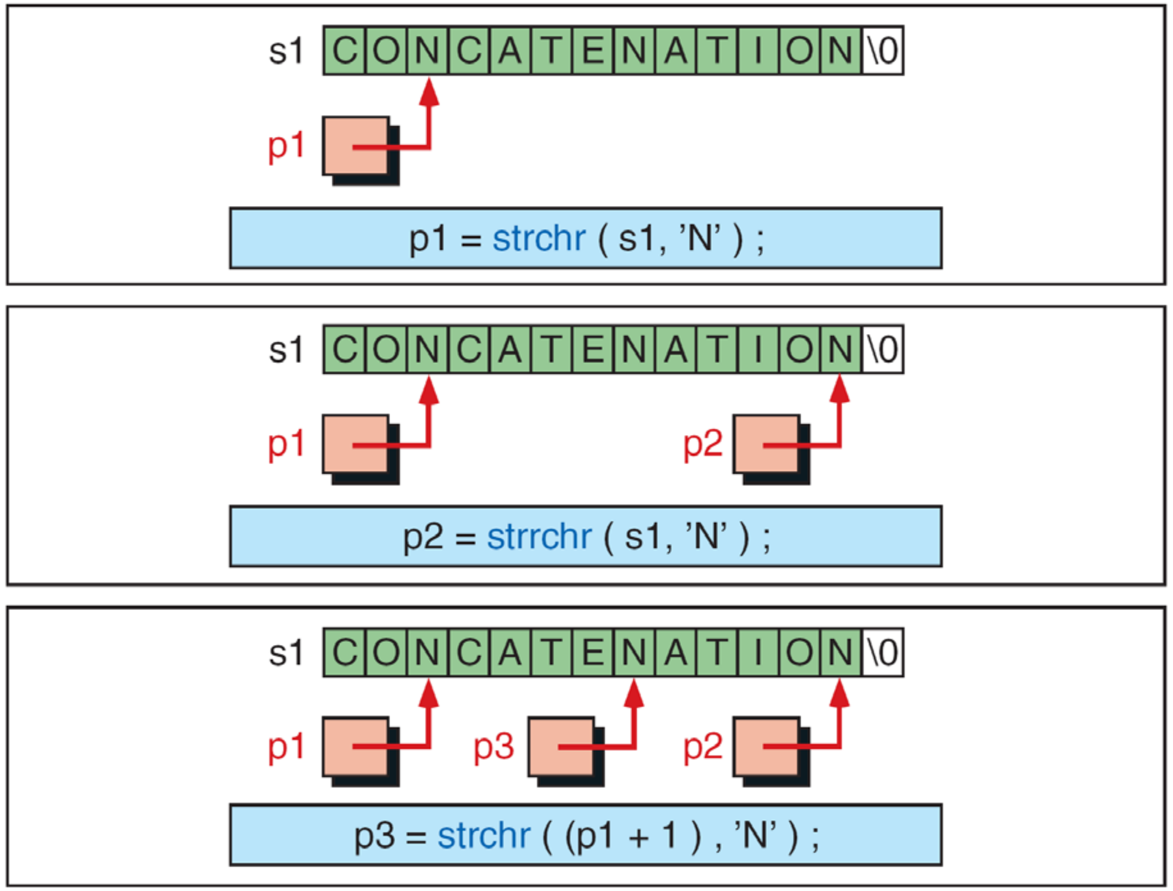
String Concatenate



String N Concatenate

Character in String

char * strchr (const char * str, int character);


s1
C O N C A T E N A T I O N \0
p1
p1 = strchr (s1, ’N’) ;
s1
C O N C A T E N A T I O N \0
p1
p2
p2 = strrchr (s1, ’N’) ;
s1
C O N C A T E N A T I O N \0
p1
p3
p2
p3 = strchr ((p1 + 1) , ’N’) ;

Return Value
A pointer to the first occurrence of *character* in *str*.

If the *character* is not found, the function returns a null pointer.

String/Data Conversion

- #include ‹stdio.h›

- String to Data Conversion
- Data to String Conversion

Assn #4 – Prob 1: String 처리 함수 구현하기

- int mystrlen(char *str);
- char *mystrcpy(char *toStr, char *fromStr);
- char *mystrcmp(char *str1, char *str2);
- char *mystrcat(char *str1, char *str2);
- char *mystrchr(char *str, char ch);
- int myatoi(char *str); → 이건 저번 어싸인에서 했고

- 위에서 기본적인 원리는 다 했죠? ^^