

다양한 문자열 라이브러리 함수

- ◆ `#include <string.h>`
함수원형으로 선언
- ◆ 메모리 내용 비교, 복사, 연결 등과 같은 다양한 문자열 처리 함수 제공
- ◆ 자료형 `size_t`
 - `unsigned int` 타입
 - 문자열 길이 표현에 사용
- ◆ 자료형 `void *`
 - 타입을 특정하지 않은 다양한 포인터 형
 - 메모리 번지수 전달에 사용
- ◆ 함수 예
 - `void *memchr(const void *str, int c, size_t n);`
 - `int memcmp(const void *str1, const void *str2, size_t n);`
 - `void *memcpy(void *dest, const void *src, size_t n);`
 - `void *memset(void *str, int c, size_t n);`
 - `size_t strlen(const char *str);`

문자열 관련 함수 활용 (memfun.c)

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char src[50] = "https://www.visualstudio.com";
    char dst[50];
    printf("문자배열 src = %s\n", src);
    printf("문자열크기 strlen(src) = %d\n", strlen(src));
    memcpy(dst, src, strlen(src) + 1);
    printf("문자배열 dst = %s\n", dst);

    char ch = ':';
    char *ret = memchr(dst, ch, strlen(dst));
    printf("문자 %c 뒤에는 문자열 %s 이 있다.\n", ch, ret);
    return 0;
}
```

[결과]

문자배열 src = https://www.visualstudio.com

문자열크기 strlen(src) = 28

문자배열 dst = https://www.visualstudio.com

문자 : 뒤에는 문자열 ://www.visualstudio.com 이 있다.

문자열 관련 함수

- ◆ `#include <string.h>`
함수원형으로 선언. 대부분 `strxxx()`로 명명.
- ◆ 대상 문자열(`dest`)에 충분한 공간이 있다는 가정하에 동작함.
- ◆ 문자열 비교, 복사, 연결 등과 같은 다양한 문자열 처리

- ◆ 함수 예
 - `int strcmp(const char *s1, const char *s2);`
 - `int strncmp(const char *s1, const char *s2, size_t maxn);`

 - `char *strcpy(char *dest, const char *source);`
 - `char *strncpy(char *dest, const char *source, size_t maxn);`
 - `char *strcpy_s(char *dest, size_t sizedest, const char *source);`
 - `char *strncpy_s(char *dest, size_t sizedest, const char *source, size_t maxn);`

 - `char *strcat(char *dest, const char *source);`
 - `char *strncat(char *dest, const char *source, size_t maxn);`
 - `char *strcat_s(char *dest, size_t sizedest, const char *source);`
 - `char *strncat_s(char *dest, size_t sizedest, const char *source, size_t maxn);`

문자열 비교 (strcmp.c)

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

int main(void) {
    char *s1 = "java";
    char *s2 = "java";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));

    s1 = "java";
    s2 = "jav";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
    s1 = "jav";
    s2 = "java";
    printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
    printf("strncmp(%s, %s, %d) = %d\n", s1, s2, 3, strncmp(s1, s2, 3));

    return 0;
}
```

[결과]

```
strcmp(java, java) = 0
strcmp(java, jav) = 1
strcmp(jav, java) = -1
strncmp(jav, java, 3) = 0
```

문자열 비교 (strcpy.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void) {
    char dest[80] = "Java";
    char source[80] = "C is a language.";

    printf("%s\n", strcpy(dest, source));
    //printf("%d\n", strcpy_s(dest, 80, source));
    //printf("%s\n", dest);
    printf("%s\n", strncpy(dest, "C#", 2));

    printf("%s\n", strncpy(dest, "C#", 3));
    //printf("%d\n", strncpy_s(dest, 80, "C#", 3));
    //printf("%s\n", dest);
    return 0;
}
```

[결과]

C is a language.

C#is a language.

C#

문자열 비교 (strcat.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void) {
    char dest[80] = "C";

    printf("%s\n", strcat(dest, " is "));
    //printf("%d\n", strcat_s(dest, 80, " is "));
    //printf("%s\n", dest);
    printf("%s\n", strncat(dest, "a java", 2));
    //printf("%d\n", strncat_s(dest, 80, "a proce", 2));
    //printf("%s\n", dest);
    printf("%s\n", strcat(dest, "procedural "));
    printf("%s\n", strcat(dest, "language."));

    return 0;
}
```

[결과]

C is

C is a

C is a procedural

C is a procedural language.

문자열 분리 (strtok())

- ◆ 문자열에서 구분자(delimiters)인 문자(들)를 지정하여 토큰을 추출하는 함수
- ◆ 함수 예
 - `char *strtok(char *str, const char *delim);`
 - `char *strtok_s(char *str, const char *delim, char **context);`
- ◆ 함수 사용 방법
 - strtok()를 while문 이용하여 NULL이 반환될때까지 반복 호출.
- ◆ 인자 설명
 - 첫 번째 인자 str :
첫 호출 시에는 토큰을 추출할 대상 문자열. (내용이 바뀔 것이므로 문자열 상수는 사용 불가능.)
두 번째 이후 호출 시에는 NULL.
 - 두 번째 인자 delim : 구분자로 사용될 문자의 모임
- ◆ 반환값
 - == NULL : 더 이상 분리할 토큰이 없는 경우
 - != NULL : 다음 토큰 문자열

문자열 비교

(strtok.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
```

```
int main(void) {
    char str1[] = "C and C++\t language are best!";
    char *delimiter = ",\t!";

    printf("문자열 %s"을 >>\n", str1);
    printf("구분자[%s]를 이용하여 토큰을 추출 >>\n", delimiter);
    char *ptoken = strtok(str1, delimiter);
    while (ptoken) //(ptoken != NULL) {
        printf("%s\n", ptoken);
        token = strtok(NULL, delimiter); //다음 토큰을 반환
    }
    return 0;
}
```

[결과]

문자열 "C and C++ language
구분자[, !]를 이용하여 토큰을

C

and

C++

language

are

best

다양한 문자열 관련 함수 (strfun.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
```

```
int main(void) {
    char str[] = "JAVA 2017 go c#";
    printf("%d\n", strlen("java"));
    printf("%s, ", _strlwr(str));
    printf("%s\n", _strupr(str));

    printf("%s, ", strstr(str, "VA"));
    printf("%s\n", strchr(str, 'A'));

    return 0;
}
```

[결과]

4

java 2017 go c#, JAVA

2017 GO C#

VA 2017 GO C#, AVA 2017

GO C#

모두 소문자로 변환

모두 대문자로 변환

문자열 "VA"가 시작되는 포인터 반환

문자 A가 처음 나타나는 위치 반환