학번 : _____ 성명 : _____.

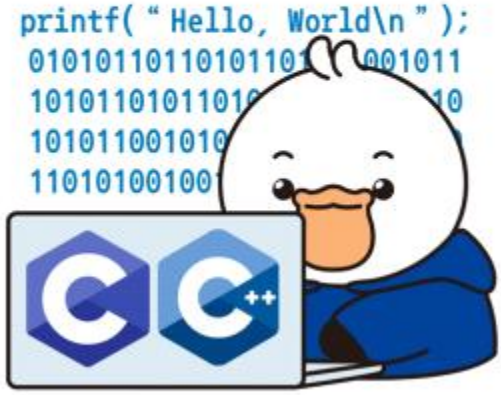
- 본 강의자료는 과학기술정보통신부 및 정보통신기획평가원에서 지원하는『소프트웨어중심대학』사업의 결과물입니다.
- 본 강의자료는 내용은 전재할 수 없으며, 인용할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원의 '소프트웨어중심대학'의 결과물이라는 출처를 밝혀야 합니다.

SW 대구가톨릭대학교 소프트웨어중심대학사업단

대구가톨릭대학교 컴퓨터소프트웨어학부 School of Computer Software, Daegu Catholic University

Part 10. 문자와 문자열

목차



10.1 문자 배열과 문자열

- 문자열
  - ✓ [ ]가 연속적으로 모여 있는 형태의 데이터
  - ✓ 큰 따옴표(" ")로 둘러쌓여 있는 연속적인 문자
  - ✓ C 언어에는 별도의 문자열 데이터 타입 존재 X
    - ➢ 문자(char형) 배열 이용
    - ➢ C 문자열 처리 (C string handling)

10.1 문자 배열과 문자열

- C언어의 문자 배열과 문자열
  - ✓ C 언어에서 문자열 처리를 위해 [ ] 이용
  - ✓ 문자 배열에 문자열 저장 후, 문자열의 끝 다음에 [ ] 문자 추가
  - ✓ 문자 배열에 '\0'으로 끝나는 데이터
    - ➢ 있다면 → 문자열을 처리하기 위한 문자 배열
    - ➢ 없다면 → 순수한 문자 배열

10.1 문자 배열과 문자열

- C언어의 문자 배열과 문자열
  - ✓ '\0' : [ ]
    - ➢ char형에서 0의 값을 저장하기 위한 기본값
    - ➢ 아스키 값 0에 해당하는 값
  - ✓ ex) "Hello" 문자열 저장
    - ➢ 6개의 원소를 가지는 문자 배열 사용
    - ➢ 다섯 글자에 '\0'를 추가하므로 실제로는 여섯 글자 저장
  - ✓ 한글 문자열의 경우, 한글 한 글자당 2개의 문자 저장 공간 사용

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]
H	e	l	l	o	'\0'

< "Hello" 문자열 저장하기 위한 문자 배열 arr[] >

10.1 문자 배열과 문자열

소스 10-1

```
1  #include<stdio.h>
2  int main(void) {
3     char arr1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
4     char arr2[6] = {'H', 'e', 'l', 'l', 'o', NULL};
5     char arr3[6] = {'H', 'e', 'l', 'l', 'o'};
6     char arr4[5] = {'H', 'e', 'l', 'l', 'o'};
7     printf("%s\n", arr1);
8     printf("%s\n", arr2);
9     printf("%s\n", arr3);
10    printf("%s\n", arr4);
11 }
```

✓ 문자열을 출력하기 위한 형식지정자 

출력 예	Hello Hello Hello HelloHello

10.1 문자 배열과 문자열

소스 10-2

```
1  #include <stdio.h>
2  int main(void)
3  {
4          char arr1[20] = "This is String";
5          char arr2[] = "This is String";
6          printf("%s\n", arr1);
7          printf("%s\n", arr2);
8          arr1[0] = 'C';
9          arr1[1] = 'h';
10         arr1[2] = 'a';
11         arr1[3] = 'n';
12         arr1[4] = 'g';
13         arr1[5] = 'e';
14         arr1[6] = 'd';
15         printf("%s\n", arr1);
16 }
```

출력 예	This is String This is String Changed String

10.1 문자 배열과 문자열

- 문자열 저장의 일반적인 방법 (소스 10-2)
  - ✓ 큰 따옴표에 문자열을 작성하여 [ ] 에 저장하는 방식
    - ➢ 문자열의 각 문자는 문자 배열의 [ ] 부터 차례대로 저장
  - ✓ 배열의 크기를 지정하지 않고 문자열을 저장하는 경우
    - ➢ 문자열의 문자 개수보다 1 크도록 배열을 생성
    - ➢ 문자열의 끝에는 항상 '\0' 값이 추가되기 때문

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
arr1	T	h	i	s		i	s		S	t	r	i	n	g	\0	\0	\0	\0	\0	\0
arr2	T	h	i	s		i	s		S	t	r	i	n	g	\0					
변경 후 arr1	C	h	a	n	g	e	d		S	t	r	i	n	g	\0	\0	\0	\0	\0	\0

<arr1과 arr2의 각 원소에 값이 저장되는 형태>

10.1 문자 배열과 문자열

소스 10-3

```
1  #include <stdio.h>
2  int main(void) {
3        char *str = "Constant String";
4        int i = 15;
5        printf("%s\n", str);
6        for (i = 0; i < 15; i++) {
7              printf("%c", str[i]);
8        }
9        printf("\n");
10       str[0] = 'B';
11 }
```

출력 예

```
Constant String
Constant String
Segmentation fault
```

세그멘테이션 폴트 에러 (Segmentation fault)

10.1 문자 배열과 문자열

- char형 포인터 변수를 이용한 문자열 처리 (소스 10-3)
  - ✓ [ ]를 이용하여 문자열을 저장하는 경우
    - ➢ 문자열 출력 및 각 원소 접근 가능
    - ➢ 인덱스 값 변경 시 세그멘테이션 폴트(Segmentation fault) 에러 발생

<포인터 변수를 이용하여 문자열을 저장하는 경우>

10.1 문자 배열과 문자열

소스 10-4

```
1  #include <stdio.h>
2  int main(void) {
3      char str[20];
4      int i;
5      scanf("%s", str);
6      printf("%s\n", str);
7      for (i=0; i<20; i++)
8          printf("%c", str[i]);
9      printf("\n");
10 }
```

✓ scanf() 함수를 이용하여 문자열을 입력 받을 때에는 [ ]가 포함된 문자열은 입력 받을 수 없음

입력 예 1	Inputyourstringhere.
출력 예 1	Inputyourstringhere. Inputyourstringhere.
입력 예 2	Separated string
출력 예 2	Separated Separated@@ى

10.1 문자 배열과 문자열

소스 10-5

```c
#include <stdio.h>
int main(void) {
    char str1[8], str2[80];
    gets(str1);
    gets(str2);
    printf("%s\n", str1);
    printf("%s\n", str2);
}
```

✓ 문자열을 저장하기 위한 배열 선언 시, 문자열을 저장하기에 충분한 크기의 배열을 선언하는 것이 중요

입력 예 1	test case test case
출력 예 1	test castest case test case
입력 예 2	test case test string
출력 예 2	test castest string test string

10.1 문자 배열과 문자열

소스 10-6

```c
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char *str;
    int i = 0;
    if((str=malloc(sizeof(char)*80)) == NULL) {
        printf("Memory allocation problem!\n");
        return -1;
    }
    gets(str);
    while (str[i] != '\0') i++;
    printf("String length is %d\n", i);
}
```

입력 예 1	abcdefghijklmnopqrstuvwxyz
출력 예 1	String length is 26
입력 예 2	C Programming
출력 예 2	String length is 13

10.1 문자 배열과 문자열

소스 10-7

```c
#include <stdio.h>
#include <string.h>
int main(void) {
   char nation[4][80] = {"대한민국", "미국", "중국", "일본"};
   char *capital[4] = {"서울", "워싱턴DC", "베이징", "도쿄"};
   int i;
   for (i=0; i<4 ; i++)
      printf("%s : %s\n", nation[i], capital[i]);
   return 0;
}
```

출력 예

```
대한민국 : 서울
미국 : 워싱턴DC
중국 : 베이징
일본 : 도쿄
```


10.2 문자열 활용

소스 10-8

```c
#include <stdio.h>
int main(void)
{
    char str[80];
    int i = 0, count = 0;
    gets(str);
    while (str[i] != '\0') {
        if (str[i] == 'D' || str[i] == 'd') count++;
        i++;
    }
    printf("Number of D or d is %d.\n", count);
}
```

입력 예 1	DDDDddddDD
출력 예 1	Number of D or d is 10.
입력 예 2	Destination
출력 예 2	Number of D or d is 1.

10.2 문자열 활용

소스 10-9

```c
#include <stdio.h>
int main(void)
{
    char str[80];
    char ch;
    int i = 0;
    gets(str);
    scanf("%c", &ch);
    while (str[i] !='\0') {
        if (str[i] == ch) {
            printf("Index number of %c is %d.\n", ch, i);
            return 0;
        }
        i++;
    }
    printf("%c in not in %s\n", ch, str);
    return 0;
}
```

입력 예 1	marshmello h
출력 예 1	Index number of h is 4.
입력 예 2	Republic of Korea K
출력 예 2	Index number of K is 12.

10.2 문자열 활용

소스 10-10

```c
#include <stdio.h>
int main(void) {
    char str1[80], str2[80];
    int i;
    gets(str1);
    gets(str2);
    for(i = 0; i < 80; i++) {
        if (str1[i] != str2[i]) {
            printf("Different strings\n");
            return 0;
        }
        else if (str1[i] == '\0' && str2[i] == '\0') {
        break;
        }
    }
    printf("Same strings\n");
    return 0;
}
```

입력 예 1	Str Allocation Test. Str Allocation Test.
출력 예 1	Same strings
입력 예 2	This is Another Str1 This is Another Str2
출력 예 2	Different strings

10.2 문자열 활용

소스 10-11

```
1  #include<stdio.h>
2  int main(void)
3  {
4          char src[80], dest[80];
5          int i =0;
6          gets(src);
7          while (src[i] !='\0') {
8                  dest[i] = src[i];
9                  i++;
10         }
11         dest[i] ='\0';
12         printf("Source : %s\n", src);
13         printf("Destination : %s\n", dest);
14         return 0;
15 }
```

입력 예 1	This is test String
출력 예 1	Source : This is test String Destination : This is test String
입력 예 2	Copied String
출력 예 2	Source : Copied String Destination : Copied String

10.3 string.h의 함수

- 문자열 처리를 위한 C 표준 라이브러리
  - ✓ string.h 헤더 파일
    - ➢ [ ]
  - ✓ 문자열의 복사, 병합, 비교, 탐색 등 다양한 기능의 함수 제공

10.3 string.h의 함수

< string.h 에 정의된 함수의 종류 >

	함수	설명
복사	void * memcpy (void * destination, const void * source, size_t num);	source이 가리키는 곳부터 num 바이트의 데이터를 destination에 복사
	void * memmove (void * destination, const void * source, size_t num);	source이 가리키는 곳부터 num 바이트의 데이터를 destination으로 이동
	char * strcpy (char * destination, const char * source);	source의 문자열을 destination에 복사
	char * strncpy (char * destination, const char * source, size_t num);	source에서 destination으로 처음 num 개의 문자들을 복사
병합	char * strcat (char * destination, const char * source);	source를 destination 뒤에 이어 붙여 destination에 저장하고, 이어 붙인 문자열을 반환
	char * strncat (char * destination, char * source, size_t num);	source에서 num개의 문자들을 destination 뒤에 이어 붙여 destination에 저장하고, 반환

10.3 string.h의 함수

< string.h 에 정의된 함수의 종류 >

	함수	설명
비교	int memcmp (const void * ptr1, const void * ptr2, size_t num);	ptr1이 가리키는 첫 num 바이트의 데이터와 ptr2가 가리키는 첫 num 바이트의 데이터를 비교
	int strcmp (const char * str1, const char * str2);	str1과 str2를 비교
	int strcoll (const char * str1, const char * str2);	strcmp와 비슷하지만 LC_COLLATE에 정의되어 있는 방식에 따라 해석된 후 비교
	int strncmp (const char * str1, const char * str2, size_t num);	str1의 처음 num 개의 문자를 str2의 처음 num 개의 문자와 비교
	size_t strxfrm (char * destination, const char * source, size_t num);	source를 현재 지역 정보에 따라 문자열을 변환한 후 변환한 문자열의 처음 num개 문자를 destination에 복사

10.3 string.h의 함수

< string.h 에 정의된 함수의 종류 >

	함수	설명
탐색	void * memchr (const void * ptr, int value, size_t num);	ptr이 가리키는 메모리의 처음 num 바이트 중 처음으로 value와 일치하는 값(문자)의 주소를 반환
	char * strchr (const char * str, int character);	str에서 처음으로 character와 일치하는 문자의 주소를 반환
	size_t strcspn (const char * str1, const char * str2);	str1에서 str2의 각 원소 문자 중 처음으로 일치하는 데 문자를 찾는데 소요된 문자의 개수를 반환
	char * strpbrk (const char * str1, const char * str2);	str1에서 str2의 각 원소 문자 중 처음으로 일치하는 문자의 주소 반환
	char * strrchr (const char * str, int character);	str에서 character와 일치하는 마지막 문자의 주소를 반환
	size_t strspn (const char * str1, const char * str2);	str1에서 str2에 속하는 문자가 아닌 문자를 처음 발견하기까지의 문자의 수 반환
	char * strstr (const char * str1, const char * str2);	str1에서 str2가 처음 검색되는 곳의 주소 반환

10.3 string.h의 함수

< string.h 에 정의된 함수의 종류 >

	함수	설명
탐색	char * strtok (char * str, const char * delimiters);	str을 delimiters의 문자들로 분리 (원본 문자열 변경)
	char *strtok_r(char *str, const char * delim, char **pos);	str을 주어진 delim문자들로 분리 (원본문자열 변경)
	char *strsep(char **str, const char * delim);	str을 delim 문자들로 분리 (원본문자열 변경)
기타	void * memset (void * ptr, int value, size_t num);	ptr이 가리키는 메모리의 처음 num 바이트를 value 값(문자)으로 변경
	char * strerror (int errnum);	errnum(에러 번호)을 해석한 뒤 그에 해당하는 에러 문자열의 포인터를 반환
	size_t strlen (const char * str);	str의 길이를 반환

10.3 string.h의 함수

- [ ] 함수
  - ✓ 문자 포인터 변수 또는 문자 배열 두개를 인자로 받아 두 번째 인자의 문자열을 첫 번째 인자로 복사하는 함수
- [ ] 함수
  - ✓ 두 문자열을 인자로 받아, 두 번째 인자를 첫 번째 인자에 이어 붙이고, 이를 반환하는 함수
  - ✓ 연결된 문자열은 첫 번째 인자에 저장됨

10.3 string.h의 함수

소스 10-12

```c
#include <stdio.h>
#include <string.h>
int main(void)
{
   char src[] = "strcpy()";
   char dest[10];
   strcpy(dest, src);
   printf("Source : %s\n", src);
   printf("Destination : %s\n", dest);
   return 0;
}
```

출력 예	Source : strcpy() Destination : strcpy()

10.3 string.h의 함수

소스 10-13

```c
#include <stdio.h>
#include <string.h>
int main(void) {
        char str1[10] ="str";
        char str2[10] ="cat()";
        strcat(str1, str2);
        printf("str1 : %s\n", str1);
        printf("str2 : %s\n", str2);
        return 0;
}
```

출력 예

```
str1 : strcat()
str2 : cat()
```


10.3 string.h의 함수

- [ ] 함수
  - ✓ 문자열 두 개를 인자로 받아, 두 문자열을 비교하여 같다면 [ ] 반환
  - ✓ 같지 않다면, 달라지는 시점의 문자의 [ ] 을 이용한 값 반환
    - ➢ **첫 번째 문자열의 문자 아스키코드 – 두 번째 문자열의 문자 아스키코드** 를 수행한 결과값 반환
    - ➢ 첫 번째 인자의 문자가 두 번째 인자의 문자보다 알파벳 순으로 앞에 위치하면 음수, 반대라면 양수

10.3 string.h의 함수

소스 10-14

```c
#include <stdio.h>
#include <string.h>
int main(void) {
    char str1[80], str2[80];
    gets(str1);
    gets(str2);
    if (!strcmp(str1, str2)) {
        printf("str1 == str2\n");
    }
    else if (strcmp(str1, str2) > 0) {
        printf("str1 != str2, %d\n", strcmp(str1, str2));
    }
    else if (strcmp(str1, str2) < 0) {
        printf("str1 != str2, %d\n", strcmp(str1, str2));
    }
    return 0;
}
```

입력 예 1	Same Same
출력 예 1	str1 == str2
입력 예 2	Compare Company
출력 예 2	str1 != str2, 4
입력 예 3	Different Difficult
출력 예 3	str1 != str2, -4

10.3 string.h의 함수

- [ ] 함수
  - ✓ 문자열 하나와 문자 하나를 인자로 받아, 문자열 내에 입력 받은 문자를 검색하여 처음 찾은 위치의 포인터를 반환하는 함수
- [ ] 함수
  - ✓ 두 개의 문자열을 인자로 받아, 첫 번째 문자열 내에 두 번째 문자열을 탐색하여 해당 위치에 포인터를 반환하는 함수

10.3 string.h의 함수

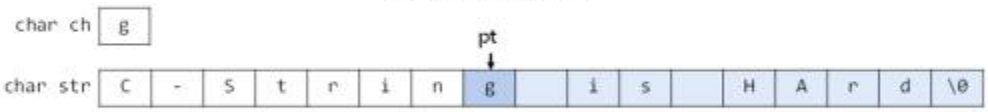
소스 10-15

```c
#include <stdio.h>
#include <string.h>
int main(void) {
    char str[80], ch, *pt;
    gets(str);
    scanf("%c", &ch);
    pt = strchr(str, ch);
    printf("Character : %c\n", *pt);
    printf("String : %s", pt);
}
```

입력 예 1	C-String is hard g
출력 예 1	Character : g String : g is hard.
입력 예 2	Strchr() func is useful.)
출력 예 2	Character :) String :) func is useful.

10.3 string.h의 함수

<입출력 1번 동작>



<해당 문자열에서 같은 문자로 더 검색하고자 할 때>

```c
pt = strchr(pt + 1, ch);
printf("string : %s", pt);
```

<문자 i를 통해 strchr() 함수를 호출했던 경우 포인터 주소의

✓ pt+1로 다시 검색할 시, 그 이후의 문자열을 이용하여 검색하므로 다음 문자 검색 가능

10.3 string.h의 함수

소스 10-16

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void) {
4      char str1[80], str2[10], *pt;
5      gets(str1);
6      gets(str2);
7      pt = strstr(str1, str2);
8      printf("str2 : %s\n", str2);
9      printf("Result : %s\n", pt);
10 }
```

입력 예 1	strstr() find string) fin
출력 예 1	str2 :) fin Result :) find string
입력 예 2	C string in string ring
출력 예 2	str2 : ring Result : ring in string

10.3 string.h의 함수

✓ strstr() 함수의 경우 strchr() 함수와 유사하게 동작

✓ strchr() 함수는 [ ]를 이용하여 검색하는데 반하여
strstr() 함수는 [ ]로 검색

< 연속적으로 검색을 하고자 할 때 >

```
pt = strstr(pt + 1, str);
printf("string : %s", pt);
```

10.3 string.h의 함수

- [ ] 함수
  - ✓ 두 개의 문자열을 인자로 받아, 첫 번째 문자열을 두 번째 문자열의 각 문자를 기준으로 쪼개어 여러 개의 문자열로 나누어 주는 함수
    - ➢ [ ]
    - ➢ 토큰(token) : 나누어 쪼개진 문자열
  - ✓ 반환값 : 토큰의 포인터
  - ✓ 첫 번째 인자로 NULL을 사용하면, 연속적으로 다음 토큰을 반환
- [ ] 함수
  - ✓ 인자로 받은 문자열의 길이를 반환하는 함수

10.3 string.h의 함수

소스 10-17

```
1  #include <stdio.h>
2  #include <string.h>
3  int main(void) {
4      char str1[80], str2[10], *pt;
5      gets(str1);
6      gets(str2);
7      pt = strtok(str1, str2);
8      while (pt != '\0') {
9          printf("%s\n", pt);
10         pt = strtok(NULL, str2);
11     }
12 }
```

입력 예 1	strtok()_is_very_useful _
출력 예 1	strtok() is very Useful
입력 예 2	ABCD EFG HIJK BFJ
출력 예 2	A CD E G HI K

10.3 string.h의 함수

소스 10-18

```
1 #include <stdio.h>
2 #include <string.h>
3 int main(void)
4 {
5     char str[80];
6     char *pt;
7     gets(str);
8     printf("String length : %lu", strlen(str));
9 }
```

입력 예 1	Strlen() Test string
출력 예 1	String Length : 20
입력 예 2	Abcdefg
출력 예 2	String Length : 7

10.4 Q&A

Q 35. 왜 아래 코드는 실행이 안 될까?

```
strcat(string, '!');
```

A. 문자열과 문자는 큰 차이가 있다. strcat() 함수는 문자열을 이어주는 함수이다. '!'와 같은 문자 상수(character constant)는 한 문자를 나타낸다. 문자열(string literal)은 쌍따옴표로 둘러싼, 대개 여러 글자의 문자로 구성된다. "!"와 같은 문자열은 하나의 문자를 나타내는 것 같지만, 실제로는 ! 문자와, 문자열 끝을 나타내는 '\0', 두 개의 문자로 구성된다. 따라서 문자열에 !를 이어 붙이려면 다음과 같이 코드를 작성해야 한다.

```
strcat(string, "!");
```

10.4 Q&A

Q 36. 문자열이 어떤 특정한 값과 같은지 검사하려고 한다. 다음과 같이 코드를 작성했는데 왜 동작하지 않는가?

```
char *string;
if (string == "value") {
}
```

A. C 언어는 문자열을 문자의 배열로 처리한다. 그리고 C 언어에서는 배열 전체에 대해 어떤 연산을 (대입, 비교 등) 직접 할 수 있는 방법은 없다. 위의 코드에서 == 연산은 피연산자인 포인터의 값을 비교한다. - 즉 변수 string의 포인터 값과 문자열 "value"를 가리키는 포인터 값을 비교한다. - 따라서 두 개의 포인터가 같은 곳을 가리키는지를 비교한다. 대개의 경우 이 값이 같게 될 경우는 거의 없으므로, 이 비교는 거의 항상 같지 않다고 나온다.

두 문자열을 비교하는 방법으로 라이브러리 함수인 `strcmp()` 를 쓰는 것이 가장 좋다.

```
if (strcmp(string, "value") == 0) {
}
```


10.4 Q&A

Q 37. 아래 코드와 같이 할 수 있다면

```
char a[] = "Hello, world!";
```

왜 이렇게는 할 수 없는가?

```
char a[14];
a = "Hello, world!";
```

A. 문자열은 배열이다. 그리고 배열에는 직접 대입 연산을 쓸 수 없다.

`strcpy()` 함수를 사용하길 바란다.

```
strcpy(a, "Hello, world!");
```

10.4 Q&A

Q 38. 왜 strcat() 이 동작하지 않을까? 아래 코드처럼 했는데 동작하지 않는다.

```
char *s1 = "Hello, ";
char *s2 = "world!";
char *s3 = strcat(s1, s2);
```

A. 왜냐하면 문자열을 저장할만한 공간이 없기 때문이다. C 언어는 동적으로 문자열을 처리해주지 않는다. C 컴파일러는 소스에서 요구한 만큼만 메모리를 할당한다. (문자열의 경우엔 문자 배열과 문자열 상수를 포함한다). 프로그래머는 문자열 연결(concatenation)과 같은 실행 시간 연산(run-time operation)에 필요한 적절한 공간을 배열이나 malloc() 을 불러서 직접 만들어 주어야 한다.

10.4 Q&A

Q 39. 두 문자열이 같은지 비교하기 위해 다음과 같은 코드를 작성하였다.

```
if(!strcmp(s1, s2))
```

이것이 좋은 스타일인가?

A. 흔히들 그런 식으로 코드를 작성하기는 하지만 특별히 좋은 스타일이라고 말하기는 어렵다. 위 코드는 두 문자열이 같은 경우를 검사한다는 점에서 전혀 문제가 될 것이 없지만, ! 연산자는 대개 'not'을 의미 쓰이므로 혼동을 가져올 수 있기 때문이다.

다음과 같은 매크로를 쓰는 것이 도움이 될 경우도 있다.

```
#define Streq(s1, s2) (strcmp((s1), (s2)) == 0)
```

10.4 Q&A

Q 40. 수치는 문자로 바꾸고 싶은데 (atoi() 함수의 반대 기능), itoa() 라는 함수가 있는가?

A. 그냥 sprint()를 사용하면 된다. long이나 실수도 sprint()를 사용해서 바꿀 수 있다. 즉, sprintf를 atoi()이나 atof()의 반대 역할을 하는 함수라고 생각할 수 있다. 게다가, 어떤 식으로 출력할 것인지도 제어할 수 있다.

10.4 Q&A

Q 41. 왜 strncpy() 는 대상 문자열에 항상 '\0' 을 써 주지 않는가?

A. strncpy()의 원래 개발 목적은 고정 크기를 갖고, '\0' 으로 끝나지 않는 " 문자열 " 도 처리할 수 있게 하는것이었다. 따라서, 다른 목적으로 strncpy()를 쓸 때는 복사한 문자열이 저장된 버퍼의 끝에 수동으로 '\0' 을 채워주어야 할 필요가 있다. 따라서 strncpy()는 그렇게 자주 쓰이는 함수가 아니다. 대신, strncat()을 쓰기도 한다. 대상 버퍼가 비어있는 경우 strncat()은 strncpy()에서 기대한 작업을 수행한다.

```
*dest = '\0';
```

위 코드는 최대 n개의 문자를 복사하며, 항상 마지막에 '\0' 을 붙여준다.
또 다른 방법은 아래 코드와 같다.

```
sprintf (dest, "\%.*s", n, source);
```

위 코드는 n이 509 이하일때만 동작한다.
문자열이 아닌 특정한 크기의 바이트를 복사하고자 한다면 strncpy() 대신 memcpy() 를 사용하는 것이 더 효과적이다.

10.4 Q&A

Q 42. 다른 언어에서 제공하는 substr(문자열에서 부분 문자열 추출) 함수가 존재하는가?

A. 없다. 그 이유 중 하나로는, C 언어에서는 문자열을 취급하는 전용의 데이터 타입이 없기 때문이다. 원본 문자열의 POS번째에서 시작해서 길이가 LEN인 부분 문자열을 꺼내기 위해 아래와 같은 코드를 작성할 수 있다.

```
char dest[LEN +1];
strncpy(dest, &source[POS], LEN);
dest[LEN] = '\0';
```

아래와 같은 방식도 사용할 수 있다.

```
char dest[LEN +1] ="";
strncat(dest, &source[POS], LEN);
```

다음과 같이 포인터 형태로 할 수도 있다.

```
strncat(dest, source + POS, LEN);
```


10.4 Q&A

Q 43. 모든 문자열을 대문자로 또는 소문자로 바꾸려면 어떻게 해야 하는가?

A. 어떤 라이브러리들은 이 역할을 위해, strupr()와 strlwr(), 또는 strupper()와 strlower()를 제공하지만, 이식성도 없고, 표준도 아니다. 헤더 파일 <ctype.h>에 정의 되어 있는 매크로인 toupper()와 tolower()를 써서 이 기능을 수행하는 함수를 만들면 된다. (매우 간단하지만, 만들 함수가 전달받은 문자열을 고칠 것이냐, 아니면 따로 공간을 할당하고 변경된 문자열을 그곳에 저장할 것이냐는 미리 생각해 두어야 한다.)

10.4 Q&A

Q 44. 어떤 버전의 `toupper()`에 대해, 대문자를 인자로 줄 경우, 이상하게 동작한다. 이런 것을 예상해서인지 `toupper()`를 부르기 전에 `islower()`를 부르는 코드도 많이 보았다.

A. 오래된 버전의 `toupper()`와 `tolower()`의 경우, 변환이 필요없는 값이 인자로 전달되면, 제대로 동작하지 않는다. (즉 알파벳이 아닌 문자나, 또는 이미 변환이 된 알파벳인 경우). 그래서 오래된 (또는 이식성이 아주 높은) 코드는 `toupper()` 또는 `tolower()`를 부르기 전에, `islower()`와 `isupper()`를 부르는 경향이 있다.
그러나, C 표준은 이러한 함수들이 어떤 값이 전달되더라도 안전하게 동작한다고 말하고 있다. 즉, 변경이 필요없는 문자들은 변경되지 않는다.

10.5 실습 및 과제

- 실습 및 과제 진행
  - DCU Code : http://code.cu.ac.kr
  - 10 주차 실습
    - 코딩 10-1, 코딩 10-2, 코딩 10-3, 코딩 10-4, 코딩 10-5, 코딩 10-6 (p388-402)
  - 10 주차 과제
    - 10장 프로그래밍 연습 1-3번 (p410)

10.6 참고 문헌

- 위키백과 문자열 참조(https://ko.wikipedia.org/wiki/문자열)
- 위키백과 C 문자열 처리 참조(https://ko.wikipedia.org/wiki/C_문자열_처리)

E N D