[5주차 1강]
문자열(3)

학습 내용

- 10.4 문자열의 배열
- 10.5 문자열 및 문자 처리 함수 (1)

학습 목표

- 10.4 다수의 문자열을 처리하는 방법을 학습한다.
- 10.5 문자열 및 문자 처리 표준 함수의 사용법을 익힌다.

C 프로그래밍 및 실습

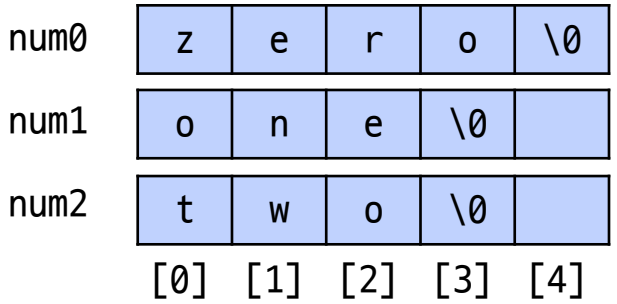
1. 문자열 개요
2. 문자열 저장 및 기본 입출력 (1)
3. 문자열과 포인터
4. **문자열의 배열**
5. 문자열 및 문자 처리 함수
6. 문자열 및 문자 입출력

4. 문자열의 배열

다수의 문자열 처리하기 : 문자 배열을 여러 개 사용

```
char num0[5] = "zero";
char num1[5] = "one";
char num2[5] = "two";

printf("%s\n", num0);
printf("%s\n", num1);
printf("%s\n", num2);
```



- 문자열이 많아지면 불편

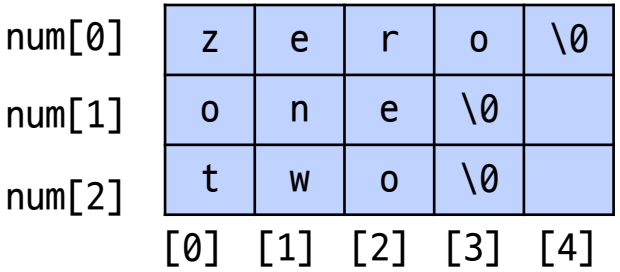
4. 문자열의 배열

다수의 문자열 처리하기: 문자열의 배열 (문자 배열을 배열로 묶기)

- 2차원 문자 배열 이용
  - ✓num[0], num[1], num[2]의 **자료형은 char***

```
int i;
char num[3][5] = {"zero", "one", "two"};

for( i=0; i < 3; ++i )
   printf("%s\n", num[i]);
```

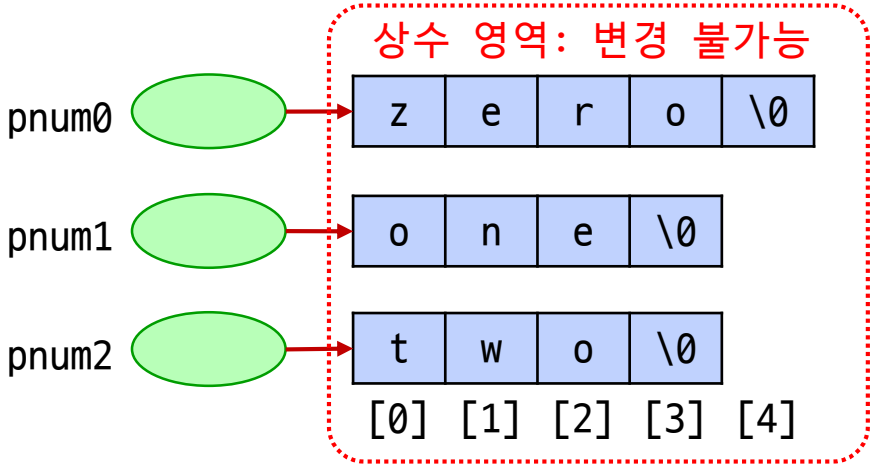


4. 문자열의 배열

다수의 문자열 처리하기 : 문자형 포인터를 여러 개 사용

```
char *pnum0 = "zero";
char *pnum1 = "one";
char *pnum2 = "two";

printf("%s\n", pnum0);
printf("%s\n", pnum1);
printf("%s\n", pnum2);
```

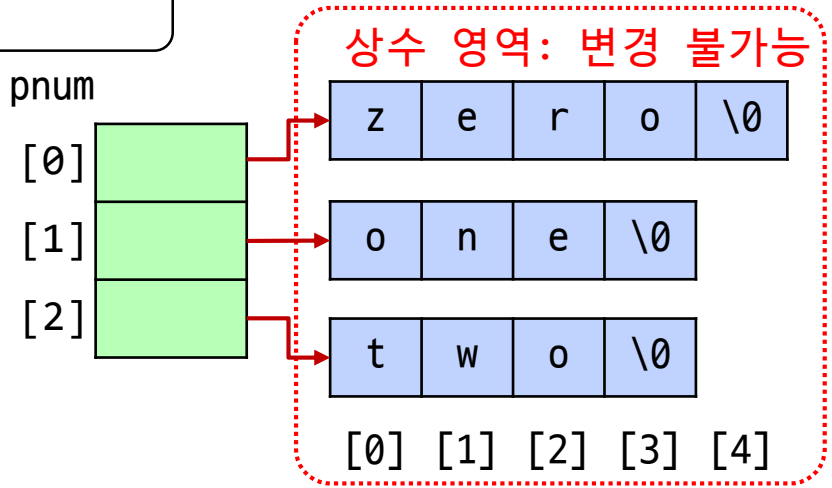


4. 문자열의 배열

다수의 문자열 처리하기 : 문자 포인터 배열(포인터를 배열로 묶기)

```
int i;
char *pnum[3] = {"zero", "one", "two"};

for( i=0; i < 3; ++i )
   printf("%s\n", pnum[i]);
```



※ 실습하기

[예제 10.4] 다음 프로그램을 작성하시오.

- 3 X 20 크기의 2차원 문자배열을 선언하고, 다음 문자열로 초기화
  - ✓ "Time is gold"
  - ✓ "No pain no gain"
  - ✓ "No sweat no sweet"
- 2중 반복문을 사용하여, 각 문자열에서 영어 소문자 'a' 가 몇 번 나오는 지 출력

※ 실습하기

[예제 10.5] 다음 프로그램을 작성하시오.

- 이전 프로그램을 2차원 문자 배열 대신 **문자 포인터 배열**을 사용하여 구현하시오.

C 프로그래밍 및 실습

5. 문자열 및 문자 처리 함수

문자열 처리 표준 함수

- C언어에서는 문자열 처리에 관련된 다양한 표준 함수 제공
- 대부분 〈string.h〉 헤더 파일에 함수의 원형 선언되어 있음

  ✓이 헤더파일을 include 시켜야 함

  #include 〈string.h〉

- 대부분 문자열 처리 함수의 코드를 작성하는 것은 어렵지 않지만, 이미 구현되어 있는 표준 함수를 사용하는 것이 편리

- 다만, 정확한 사용법을 익혀야 함

문자열의 길이 구하기 1 (직접 구현)

- 널 문자와 반복문을 이용하여 구할 수 있음

```
char str[20] = "Hello World";
int i = 0;

while ( str[i] )  // 널문자가 아닌 동안
    ++i;          // i 값 증가
printf("length: %d\n", i);
```

결과:

```
length: 11
```

5. 문자열 및 문자 처리 함수

문자열의 길이 구하기 2 (표준 함수 strlen 이용)

- 원형 : unsigned int strlen(char *s)
- 기능 : 문자열 s의 길이 반환

```
#include<stdio.h>
#include<string.h>  // strlen() 함수가 선언된 헤더 파일

int main(){
    char str[20] = "Hello World";
    printf("length: %d\n", strlen(str));
    return 0;
}
```

결과:

```
length: 11
```

5. 문자열 및 문자 처리 함수

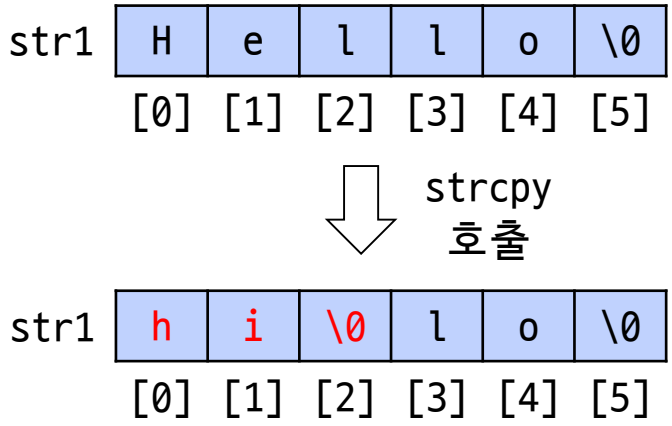
문자열 복사하기

- 원형 : char *strcpy(char *dest, char *src)
- 기능 : dest의 공간에 src의 문자열 **복사** (문자열 **대입**)
  src는 변화 없음

```
char str1[6] = "Hello";

strcpy( str1, "hi");

printf("str1: %s!!\n", str1);
결과:
str1: hi!!
```



- 참고) strncpy() 함수: 복사할 문자열의 길이를 지정하는 함수

strcpy(dest, src) 사용 시 주의사항

- dest의 공간이 **src의 문자열 길이+1 (널 문자) 이상** 이어야 함
  - ✓ 그렇지 않으면, 런타임 오류의 원인

```
char a[10], b[5] = "hi";
char *c = NULL;

strcpy( a, b);        // 정상 작동

strcpy( b, "Hello");      // 런타임 오류 유발
strcpy( c, "Hello");      // 런타임 오류 유발

c = a;
strcpy( c, "Hello");   // 정상 작동
```

5. 문자열 및 문자 처리 함수

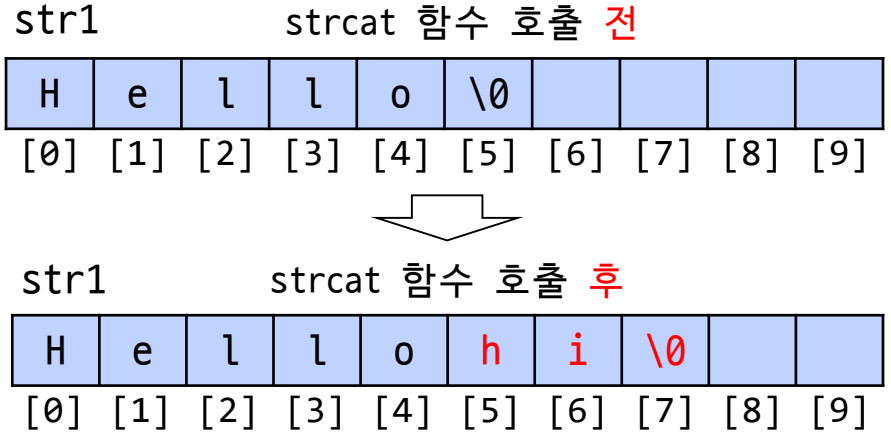
문자열 접합하기

- char *strcat(char *dest, char *src)
- 기능 : 문자열 dest 뒤에 src의 문자열 **접합**
  src는 변화 없음

```
char str1[10] = "Hello";

strcat( str1, "hi");

printf("str1: %s!!\n", str1);
결과:
str1: Hellohi!!
```



- 참고) strncat() 함수: 접합할 문자열의 길이를 지정하는 함수

strcat(dest, src) 사용 시 주의사항

- dest에 접합 결과를 저장하기에 충분한 공간이 할당되어 있어야 함
  - ✓ 그렇지 않으면, 런타임 오류 유발

```
char s1[10] = "Hello";
char s2[5] = "hi";
char *s3 = NULL;
char s4[20];

strcat( s2, s1);  // 런타임 에러 유발
strcat( s3, s1);  // 런타임 에러 유발
strcat( s4, s1);  // 런타임 에러 유발 (why?)
```

학습 정리

- 다수의 문자열을 저장하고 처리하기 위해서는 **문자열의 배열**을 사용하면 효율적
- 문자열의 배열은 **2차원 문자 배열** 또는 **문자 포인터 배열**로 구현 가능
- C 언어는 **strlen**(문자열 길이 계산), **strcpy**(문자열 복사), **strcat**(문자열 접합) 등의 문자열 처리를 위한 표준 함수가 지원됨