

문자열

학습내용

- 문자열 이해
- 문자열 입출력

학습목표

- 문자열의 정의에 대해 설명할 수 있다.
- 표준입출력을 통한 문자열 처리를 구현할 수 있다.



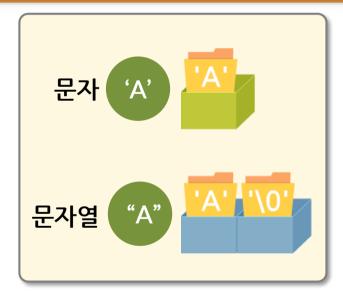
1 문자열 기초



문자열

연속된 문자들의 모임

- 1 문자열은 <mark>큰 따옴표("")</mark>로 표현함
- **2** 문자열의 끝에는 <mark>널 문자('\0')를</mark> 함께 저장함





1 문자열 기초

문자열 상수

- 값이 변경되지 않는 문자열
 - "A"나 "Hello World" 등
 - 문자열 리터럴이라고도 함

문자열 변수

- 프로그램 수행 중에 변경될 수 있는 문자열
 - 문자 배열



2 문자열 선언과 초기화

선언

문자 배열의 크기는 <mark>저장할 문자 수 +1</mark> 크기로 지정

초기화

" " 문자를 대입연산자에 이용



"Hello"를 저장

char a[6] = "Hello";

a[0]='H';

a[1]='e';

a[2]='l';

a[3]='l';

a[4]='o';

a[5]='₩0';

Н

e

0

\0



2 문자열 선언과 초기화

배열 크기 - 1개의 문자를 초기화 가능

```
char a[10]="Hello";
char b[12]="Hello World";
char c[]="Hello";
char d[4]="Good";
char e[10];
e = "Good";
```

- 🚺 문자열 표준 입력
 - 1 scanf()

01 입력된 문자열을 str 배열에 저장

02 공백 또는 enter 전까지 입력된 문자열을 저장

03 함수 형식: int scanf("%s",str);



"Hello World"를 입력

char str[15]; scanf("%s", str);

H e I I o \(0 \) \(0 \) \(0 \) \(0 \) \(0 \) \(0 \) \(0 \) \(0 \) \(0 \)

- 🚺 문자열 표준 입력
 - 2 gets()

01

공백을 포함한 문자열 입력

02

char* gets(char* str);

03

enter를 '₩0'으로 변환하여 저장

정상 수행 시

• 문자열 리턴

오류 시

- Null
- 읽을 문자열이 없는 경우 : EOF 리턴



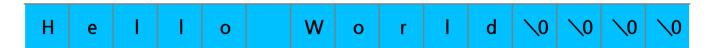
🚺 문자열 표준 입력

2 gets()



"Hello World"를 입력

char str[15];
gets(str);



- 2 문자열 표준 출력
 - 1 printf()

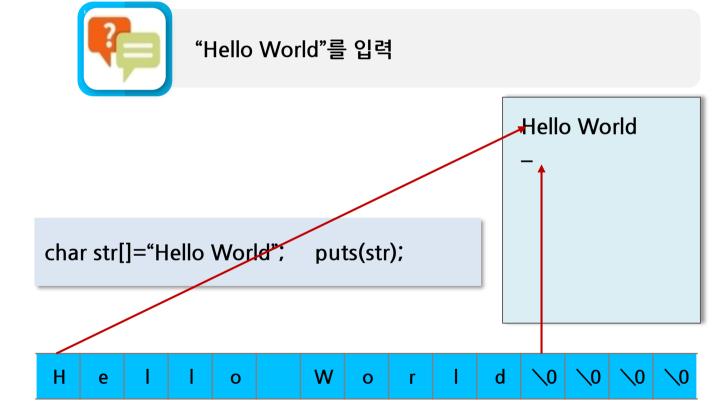
형식 문자열 "%s"를 이용

char a[]="Hello"; printf("%s", a);

문자 배열명을 printf함수의 첫 번째 인자로 가능

char a[]="Hello"; printf(a);

- 2 문자열 표준 출력
 - 2 puts()
 - 1 int puts(const char* str);
 - 2 str 배열의 문자열을 표준 출력에 출력
 - 3 배열의 '₩0'을 '₩n'로 변환하여 출력



학습정리

1. 문자열 이해



- 문자열은 null로 끝나는 문자들의 집합임
- 문자열 저장은 문자배열을 이용함
- 문자열을 저장하는 문자배열의 크기는 저장할 문자 수 + 1 이상 크기가 필요함
- 문자 배열은 선언과 동시에 " "를 이용하여 초기화가 가능함

2. 문자열 입출력



- •표준 입력을 통한 문자열 입력은 scanf(), gets()함수를 이용할 수 있음
- •scanf()는 공백이나 enter 전까지를 입력받음
- gets()는 공백을 포함한 문자열 입력이 가능함