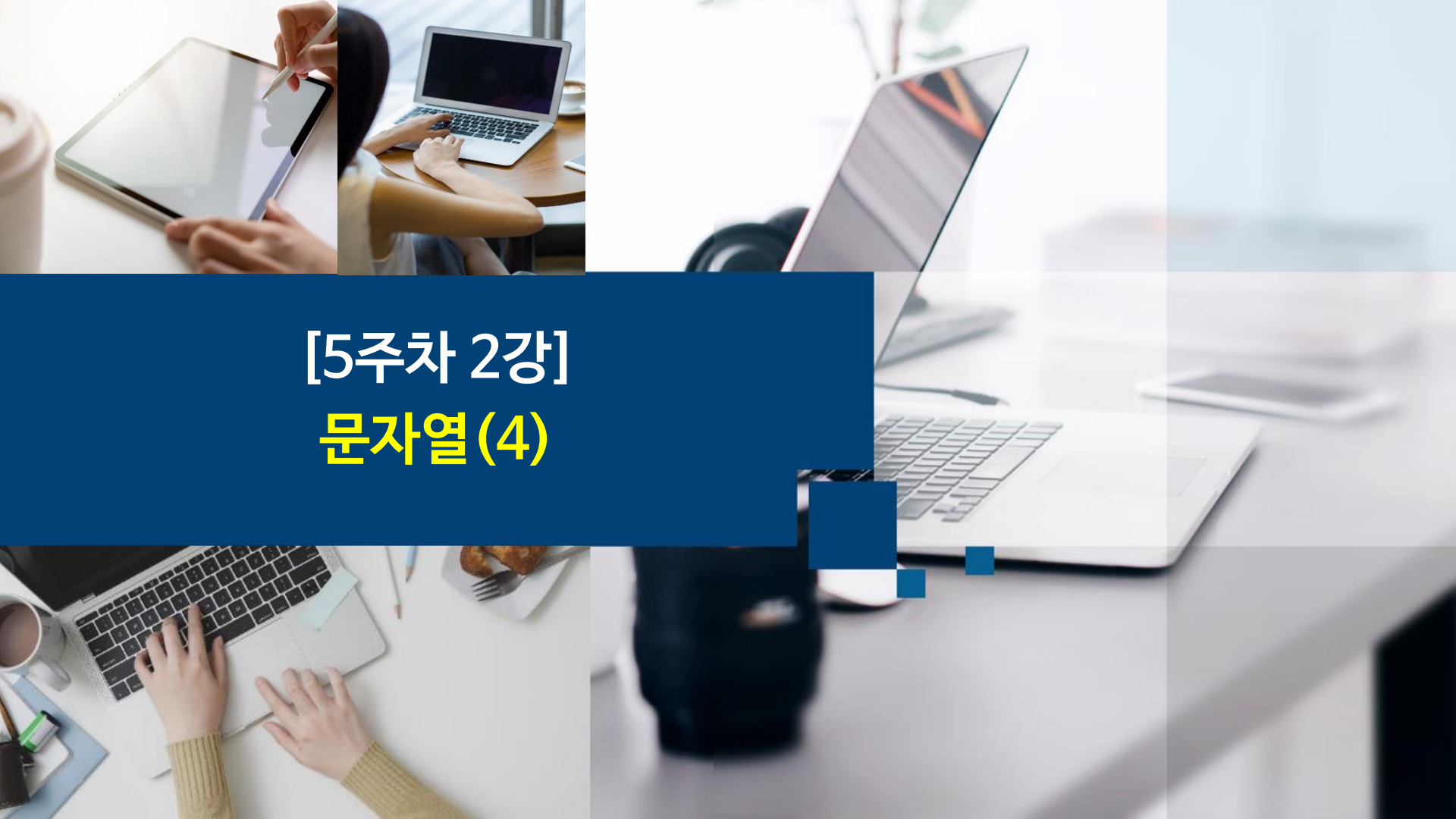


[5주차 2강]
문자열(4)

학습 내용

- 10.5 문자열 및 문자 처리 함수 (2)
- 10.6 문자열 및 문자 입출력

학습 목표

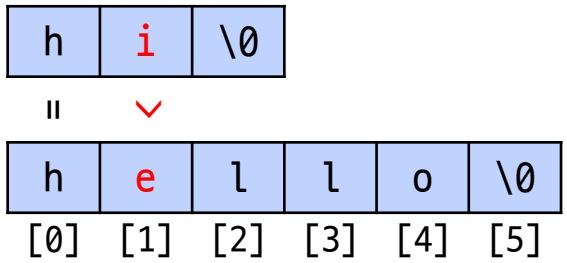
- 10.5 문자열 및 문자 처리 표준 함수의 사용법을 익힌다.
- 10.6 문자열 및 문자 입출력 표준 함수의 사용법을 익힌다.

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 (2)**
6. 문자열 및 문자 입출력

문자열 비교하기

- 원형 : `int strcmp(char *lhs, char *rhs)`
- 기능 : 사전 순으로 lhs와 rhs를 **비교**하여
  문자열 lhs 〈 rhs 이면 **음수**,
  문자열 lhs == rhs 이면 **0**,
  문자열 lhs 〉 rhs 이면 **양수** 반환
  ✓ 어떤 음수, 어떤 양수를 반환하는 지는 컴파일러마다 다를 수 있음
- 문자열 비교는
  처음부터 문자 별로 비교



- 참고) strncmp() 함수: 비교할 문자열의 길이를 지정하는 함수

5. 문자열 및 문자 처리 함수

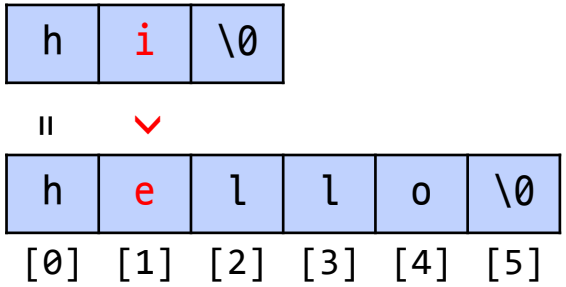
문자열 비교 예제

```
char s1[50] = "hi", s2[50] = "hello";
int cmp_result = strcmp(s1, s2);  // 문자열 비교

if( cmp_result < 0 )
   printf("%s가 %s보다 앞에 있습니다.\n", s1, s2);
else if( cmp_result == 0 )
   printf("%s가 %s와 같습니다.\n", s1, s2);
else // cmp_result > 0
   printf("%s가 %s보다 뒤에 있습니다.\n", s1, s2);
```

결과:

```
hi가 hello보다 뒤에 있습니다.
```



문자열 비교 결과의 추가 예시

✓문자는 단순히 아스키 코드 값 비교

```
char *str = "hi";
strcmp(str, str);        ⇨ 문자열 동일
strcmp(str, "hi");       ⇨ 문자열 동일
strcmp(str, "Hi");       ⇨ 소문자가 대문자보다 큼:  'h' > 'H'
strcmp("hi", ".");       ⇨ 첫 문자끼리 비교: 'h' > '.'
strcmp(str, "hi~");      ⇨ hi까지 동일. 다음 문자 '\0' < '~'
strcmp("hi", "high");    ⇨ hi까지 동일. 다음 문자 '\0' < 'g'
```

※ 실습하기

[예제 10.6] 사용자로부터 두 개의 문자열 A와 B를 입력 받아 다음 과정을 수행하는 프로그램을 작성하시오.

1) 문자열 A와 B의 길이를 각각 출력
2) A와 B 중 사전 순으로 빠른 문자열 출력
3) ABA 형태의 새로운 문자열 C를 생성하고 출력

- A와 B의 길이는 20 이내이고, 공백, 탭, 개행 문자는 없다고 가정
- 두 문자열은 서로 다르다고 가정

입력 예시

```
welcome
helloworld!!
```

출력 예시

```
7 12
helloworld!!
welcomehelloworld!!welcome
```

5. 문자열 및 문자 처리 함수

10진수로 표현된 문자열을 수로 변환

- int atoi(char *str) : int형으로 계산하여 반환
- long atol(char *str) : long형으로 계산하여 반환
- double atof(char *str) : double형으로 계산하여 반환
- <stdlib.h>에 원형 선언

```
printf("%d\n", atoi("123") );
printf("%d\n", atoi("-123") );

printf("%f\n", atof("-123") );
printf("%f\n", atof("123.45") );
```

실행 결과

```
123
-123

-123.000000
123.450000
```

5. 문자열 및 문자 처리 함수

주요 문자열 처리 함수 (요약)

함수 원형	함수 기능 설명
`unsigned int strlen(s)`	문자열 s의 길이 반환
`char *strcpy(s1, s2)`	문자열 s1에 s2를 복사
`char *strcat(s1, s2)`	문자열 s1의 끝에 s2를 접합
`int strcmp(s1, s2)`	문자열 s1과 s2를 사전 순으로 비교
`int atoi(s)`	문자열(s)로 표현된 수를 int형, long형, double형으로 계산하여 반환 예) atoi("12")는 정수 12 반환
`long atol(s)`	
`double atof(s)`	

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 (2)
6. **문자열 및 문자 입출력**

6. 문자열 및 문자 입출력

입출력 함수

- printf 와 scanf : 다양한 기능을 가진 범용 입출력 함수
  - ✓ 함수의 크기가 크고, 속도 느림
- C언어에서는 문자열과 문자에 특화된 입출력 함수 제공
  - ✓ 속도 빠르고, 문자 또는 문자열 입출력에 적합
  - ✓ 문자열 입출력함수 : puts, gets (gets_s, fputs)
  - ✓ 문자 입출력 함수 : putchar, getchar
- 위 함수들은 모두 〈stdio.h〉에 선언되어 있음

6. 문자열 및 문자 입출력

문자열 출력 함수 : int puts(char *str)

- str이 가리키는 문자열을 화면에 출력하고, **마지막에 '\n' 출력**
- 반환 값 : 출력에 성공하면 음수가 아닌 값, 실패하면 EOF
  - ✓ 참고) EOF (End Of File) : 파일의 끝을 나타내는 상수 (14장에서 학습)

```
char str[10] = "Hi World";
int ret=1;

ret = puts(str);
printf("return: %d\n", ret);
```

실행 결과

```
Hi World
return: 0
```

개행문자 '\n'이 출력되어 줄이 바뀜

- 위 코드에서 puts 대신 printf를 사용하여 str을 출력해보자. 차이점이 있는가?

6. 문자열 및 문자 입출력

문자열 입력 함수 : char *gets(char *s)

- 사용자로부터 문자열을 입력 받아, s가 가리키는 메모리 영역에 저장하고, 포인터 s를 리턴
  - ✓ 엔터('\n')가 입력될 때까지 입력된 모든 문자들을 저장
  - ✓ 마지막에 입력된 '\n'은 무시하고, 맨 뒤에 '\0'를 붙임
  - ✓ 문자열을 저장할 충분한 메모리 공간이 확보되어 있어야 함

```
char str[10];

gets(str); // 또는 gets_s(str, 10);
printf("str: %s!!", str );
```

실행 예시

```
Hi World      ← 입력
str: Hi World!!
```

  - ✓ 참고) gets_s() 함수 : gets() 함수의 보안 버전으로, 문자열을 저장할 배열 크기를 인자로 전달

(참고) 보안 상의 문제로 gets() 함수는 표준에서 제외되고, gets_s() 함수가 표준에 추가됨

- Visual Studio의 경우 2015 버전부터 gets() 함수 지원 안 함
  - ✓ gets_s 사용 ()
- gcc의 경우, 아직 gets()를 지원하고, gets_s()는 지원 하지 않음
  - ✓ gcc를 사용하는 온라인 채점 시스템에서는 gets() 사용
- 또는 대안으로 fgets 함수 사용(14장에서 학습)
  - ✓ 개행 문자도 문자열에 저장

```
char str[10];
fgets(str, 10, stdin);      // 사용법
printf("str: %s!!", str );
```

6. 문자열 및 문자 입출력

문자 입출력 함수 :

- `int putchar(int c)` : 인자 c의 문자를 화면에 출력
- `int getchar(void)` : 사용자로부터 입력된 문자 반환
- 성공하면 입출력된 문자 반환, 실패하면 EOF 반환
  - ✓ 참고) EOF (End Of File) : 파일의 끝을 나타내는 상수로 정수 -1의 값을 가짐(14장에서 학습)

```
int c;

c = getchar();
putchar(c);
```

실행 예시

```
H                ← 입력
H
```

학습 정리

- **strcmp**는 문자열의 두 문자열을 사전 순으로 비교하는 표준함수
- puts(), gets(), gets_s(), fgets()는 C 언어에서 자주 사용하는 '**문자열**' 입출력 함수
- putchar(), getchar()는 '**문자**' 입출력 함수