DATA STRUCTURE

Week 6

- String 2 -

김지오

INDEX

I. 문자열과 포인터

문자형 포인터 : 문자열을 활용하는 또 다른 형태

- 문자열을 가리키는 포인터
- 포인터 변수를 선언하고, 문자열을 가리키도록 초기화 할 수 있음

```
char *str = "Hello";

printf("%s", str);
```

I. 문자열과 포인터

문자형 포인터를 배열처럼 활용하기

- 배열과 포인터의 관계를 이용

```c
int i;
char *str = "Hello";

for (i=0 ; i<5 ; i++) {

  printf("%c", str[i]); // 문자 출력

}
```

I. 문자열과 포인터

변경 가능 여부

- “Hello” 는 문자열 상수로, 사용자 프로그램에서 변경 불가능
- 반면, str은 사용자 변수로 값을 변경할 수 있음

```
char *str = "Hello";

str[0] = 'h'; // 변경 불가능 (에러 발생)

str = "World"; // str에 저장된 값 변경 (가능)
```

실습 문제 #1

다음 프로그램을 작성하시오.

✓ 문자 포인터 변수 pc를 선언하고, 다음 문자열로 초기화

“ To be, or not to be : that is the question”

✓ 반복문을 사용하여 영어 소문자 ‘t’ 가 몇 번 나오는지 출력

→ 결과 : 6이 출력되어야 함

II. 문자열 처리 함수

문자열의 길이 구하기 1 : 직접 작성

- NULL 문자와 반복문을 이용하여 구할 수 있음

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

for( i=0 ; str[i] != '\0' ; ++i );

printf("length: %d\n", i);
```

II. 문자열 처리 함수

문자열의 길이 구하기 2 : 표준 함수 이용

strlen 함수

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

```c
#include<string.h>

int main() {

    char str[20] = "Hello World";
    printf("length: %d\n", strlen(str));
```

II. 문자열 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

II. 문자열 처리 함수

char *strcpy (char *s1, char *s2)

- 기능 : s1의 공간에 s2의 문자열 복사 (문자열 대입)
- s2는 변화 없음

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

strcpy(str1, "hi");

printf("str1: %s..\n", str1);
```

결과 : str1 : hi..

II. 문자열 처리 함수

strcpy(s1, s2) 사용 시 주의사항

- s2의 문자열 길이 + 1 이상의 공간이 s1에 할당되어 있어야 함
- 그렇지 않으면, 런타임 오류 발생

```c
char s1[10], s2[5] = "hi";
char *s3 = NULL;

strcpy(s1, s2); // 정상 작동

strcpy(s2, "Hello"); // 런타임 에러 유발
strcpy(s3, "Hello"); // 런타임 에러 유발

s3 = s1;

strcpy(s3, "Hello"); // 정상 작동
```

Thank You