

10. 문자열 (string), 파일 입출력, 이진파일, 암호화



교수 김 영 탁

영남대학교 정보통신공학과

(Tel : +82-53-810-2497; Fax : +82-53-810-4742

<http://antl.yu.ac.kr/>; E-mail : ytkim@yu.ac.kr)

Outline

◆ 문자 입출력 라이브러리

- getchar(), putchar(), getch(), putch(), scanf("%c"), printf("%c")

◆ 문자열 입출력 라이브러리, 문자열 처리 라이브러리

- scanf("%s"), printf("%s"), gets(), puts(), strlen(), strcpy(), strcmp(),

◆ 문자열 수치변환

- atoi(), atof(), atox()

◆ 문자열의 배열, 문자열 탐색, 문자열 정렬

- stringSort(), strchr(), strstr()

◆ 문자열의 토큰 단위 분리

- strtok()

◆ 파일입출력

- fopen(), fscanf(), fprintf(), fclose()

◆ 랜덤 파일 입출력 (Random File Input, Output)

- fseek(), rewind(), ftell(), feof()

◆ 이진파일 (Binary File) dump

◆ 암호화



문자 (character)

문자 (character) 표현방법

- ◆ 컴퓨터에서는 각각의 문자에 숫자코드를 붙여서 표시.
- ◆ **아스키코드(ASCII code): 표준적인 8비트 문자코드**
 - 0에서 127까지의 숫자를 이용하여 문자표현
- ◆ **유니코드(unicode): 표준적인 16비트 문자코드**
 - 전세계의 모든 문자를 일관되게 표현하고 다룰 수 있도록 설계



프로그램/컴퓨터에서 문자는
문자코드 (숫자)로 표현 됩니다.



문자 입출력 라이브러리

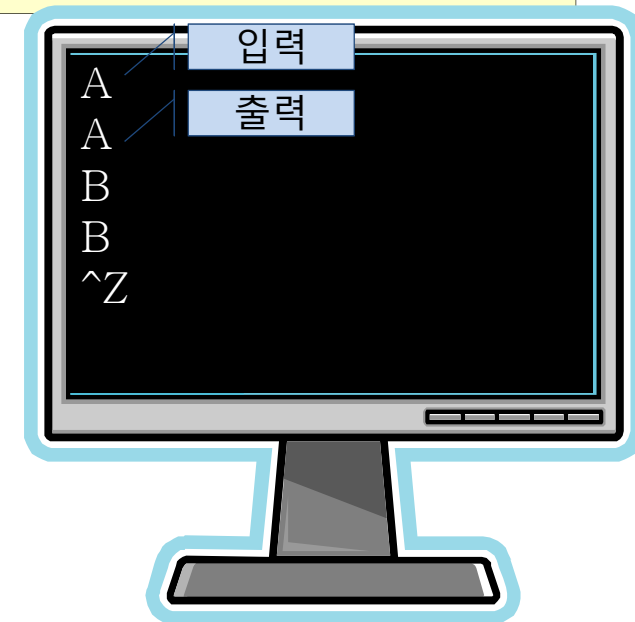
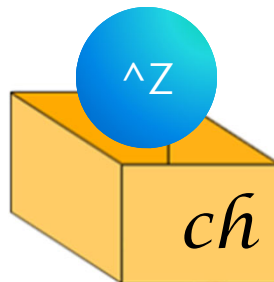
| 입출력 함수 | 설명 |
|---------------------|---|
| int getchar(void) | 하나의 문자를 읽어서 반환한다. #include <stdio.h> 필요 |
| void putchar(int c) | 정수형 인수 c에 저장된 문자를 출력한다. #include <stdio.h> 필요 |
| int getch(void) | 하나의 문자를 읽어서 반환한다(버퍼를 사용하지 않음). #include <conio.h> 필요 |
| void putch(int c) | 정수형 인수 c에 저장된 문자를 출력한다 (버퍼를 사용하지 않음). #include <conio.h> 필요 |
| scanf("%c", &c) | 하나의 문자를 읽어서 변수 c에 저장한다. #include <stdio.h> 필요 |
| printf("%c", c); | 정수형 인수 c에 저장된 문자를 출력한다. #include <stdio.h> 필요 |



버퍼를 사용하는 문자 입출력

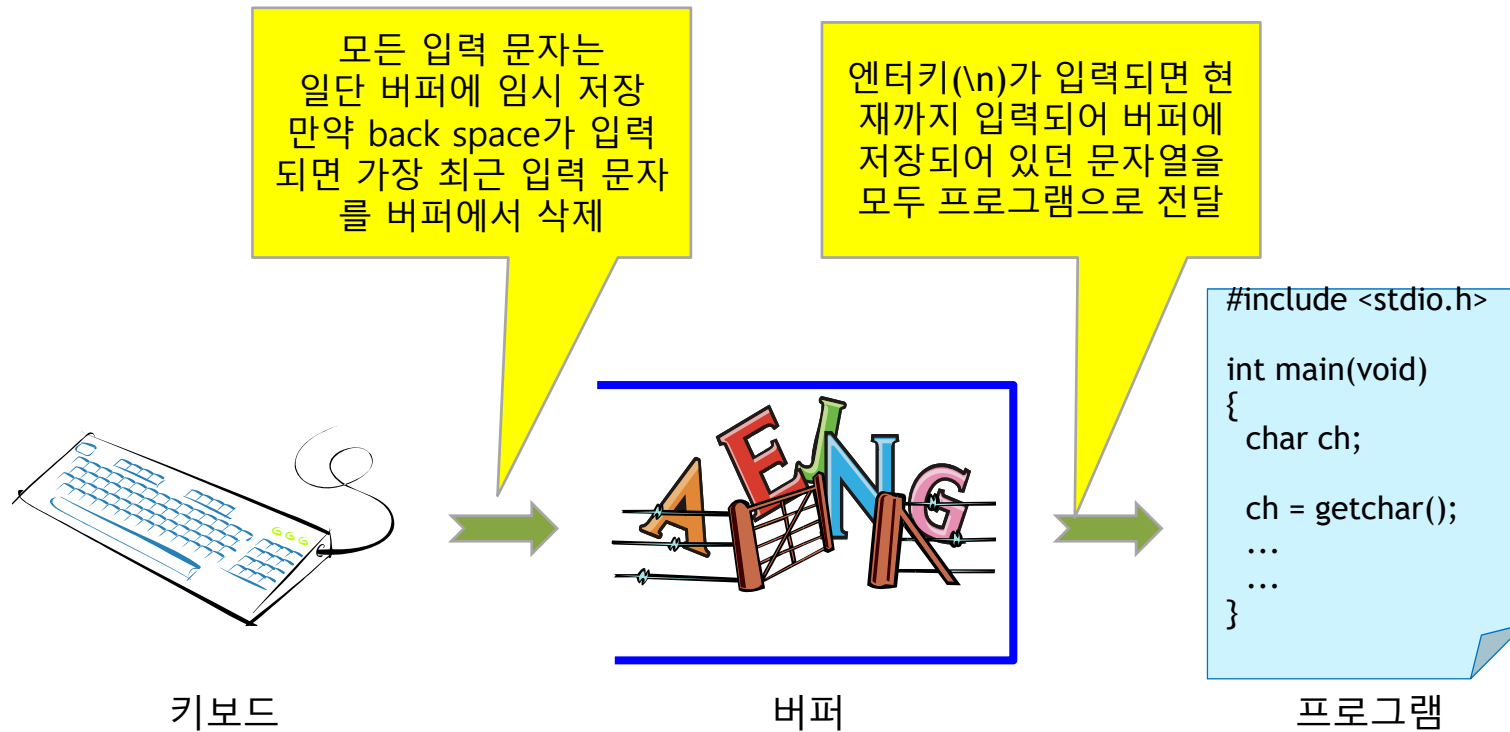
- getchar(), putchar()

```
// getchar()의 사용
#include <stdio.h>
int main(void)
{
    int ch;    // 정수형에 주의
    while( (ch = getchar()) != EOF ) // Windows에서 EOF는 ^Z
        putchar(ch);
    return 0;
}
```



버퍼링

◆ 엔터키를 쳐야만 입력을 받는 이유



버퍼를 사용하지 않는 문자 입출력 - getch(), putch()

```
#include <stdio.h>
#include <conio.h> // getch(), putch()를 위한 헤더파일

int main(void)
{
    int ch;
    while( (ch = getch()) != 'q' )
        putch(ch);
    return 0;
}
```

버퍼를 사용하지 않는
문자단위 입력



getch(), getche(), getchar()

| 문자 단위 입력 함수 | 헤더파일 | 버퍼사용여부 | 에코 여부 | 응답성 | 문자수정 여부 |
|----------------|-----------|-------------------------|-------------|-------|------------|
| getchar() | <stdio.h> | 사용함 (엔터키를 눌러 입력됨) | 에코 | 줄 단위 | 가능 |
| getch() | <conio.h> | 사용하지 않음 | 에코 하지 않음 | 문자 단위 | 불가능 |
| getche() | <conio.h> | 사용하지 않음 | 에코 | 문자 단위 | 불가능 |



용도에 맞는 것을 골라
사용하세요!

버퍼가 없이 바로 받으려면
getch()를 사용합니다.

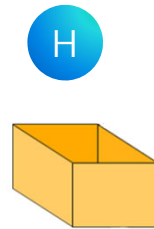


문자열 (string)

문자열 (string)

◆ 문자열(string): 문자들이 여러 개 모인 것

- "A"
- "Hello World!"
- "변수 score의 값은 %d입니다"

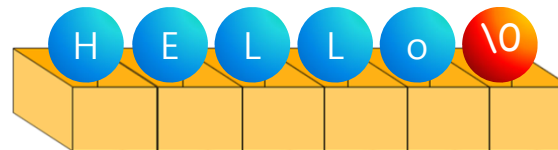


하나의 문자는 char형 변수로 저장

문자열은 여러 개의 문자로 이루어져 있으므로 문자 배열로 저장이 가능해요.

◆ 문자열의 저장

- 문자 배열 (char array) 사용
- 문자열의 마지막은 NULL 문자



문자열은 char형 배열로 저장



문자열 저장을 위한 문자 배열

◆ 각각의 문자 배열 원소에 원하는 문자를 개별적으로 대입하는 방법이다.

- `str[0] = 'H';`
- `str[1] = 'e';`
- `str[2] = 'l';`
- `str[3] = 'l';`
- `str[4] = 'o';`
- `str[5] = '\0';` // NULL 문자

◆ `strcpy()`를 사용하여 문자열을 문자 배열에 복사

- `strcpy(str, "Hello");`

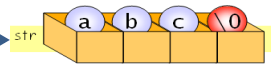
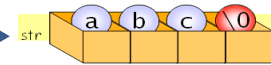
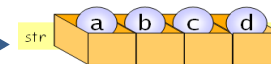
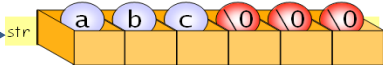
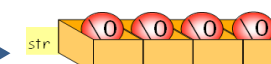
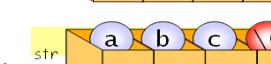
◆ 문자열 상수와 문자포인터 사용한 간접참조

- `const char *pStr = "Wonderful world !";`
// `pStr`을 사용하여 읽을 수는 있으나,
// 간접참조로 문자열을 내용을 변경할 수는 없도록 보호



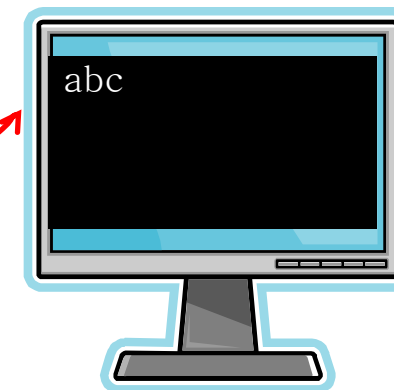
문자 배열의 초기화와 출력

◆ 문자열 (문자배열) 초기화

- `char str[4] = { 'a', 'b', 'c', '\0' };` 
- `char str[4] = "abc";` 
- `char str[4] = "abcdef";` 
- `char str[6] = "abc";` 
- `char str[4] = "";` 
- `char str[] = "abc";` 

◆ 문자열 출력

```
char str[] = "abc";  
printf("%s", str);
```

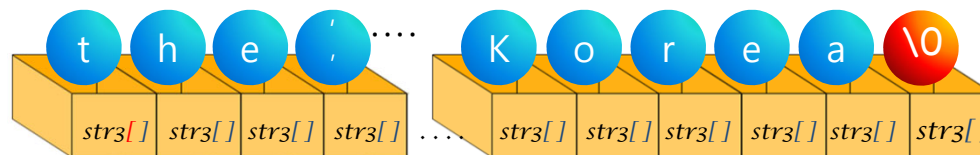
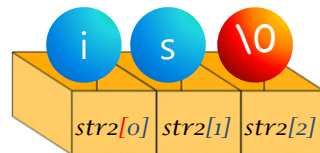
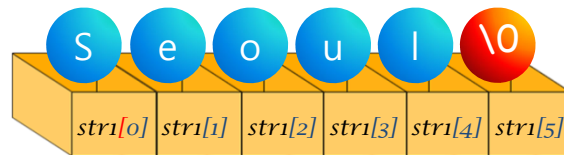


문자열 초기화와 출력 예제

```
#include <stdio.h>

int main(void)
{
    char str1[6] = "Seoul";
    char str2[3] = { 'i', 's', '\0' };
    char str3[] = "the capital city of Korea.";

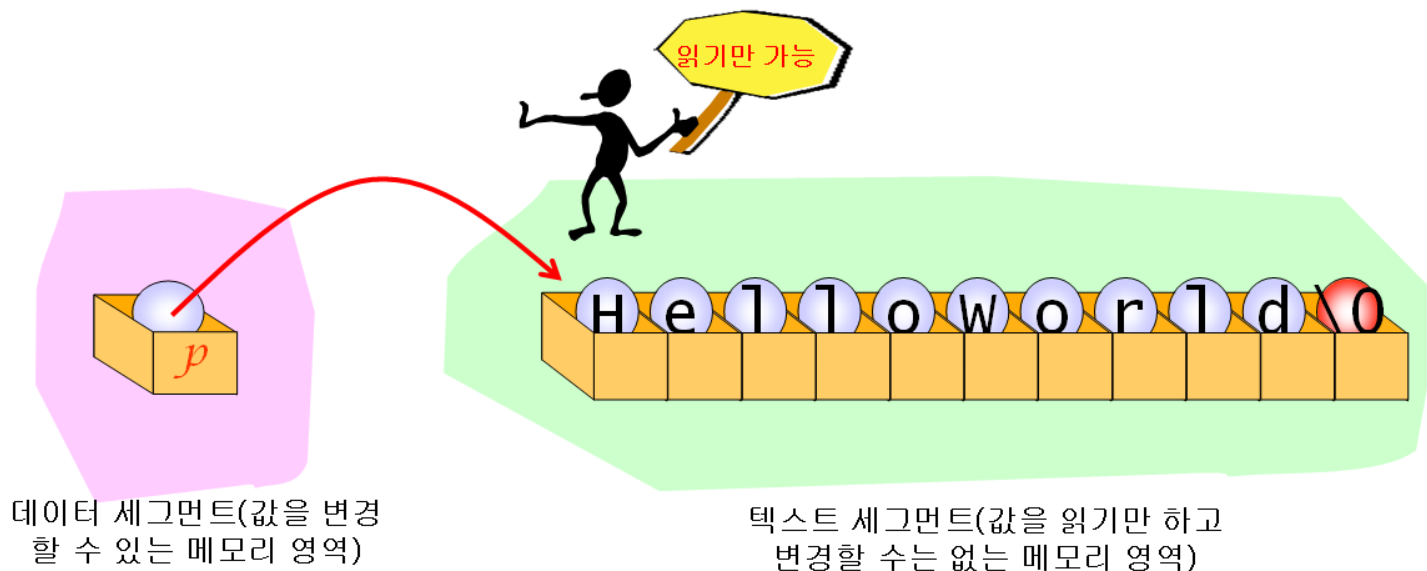
    printf("%s %s %s\n", str1, str2, str3);
}
```



문자열 상수와 포인터

- ◆ 문자열 상수 (string constant) : "HelloWorld"와 같이 프로그램 소스 안에 포함된 문자열
- ◆ 문자열 상수는 메모리 영역 중에서 텍스트 세그먼트(text segment)에 저장

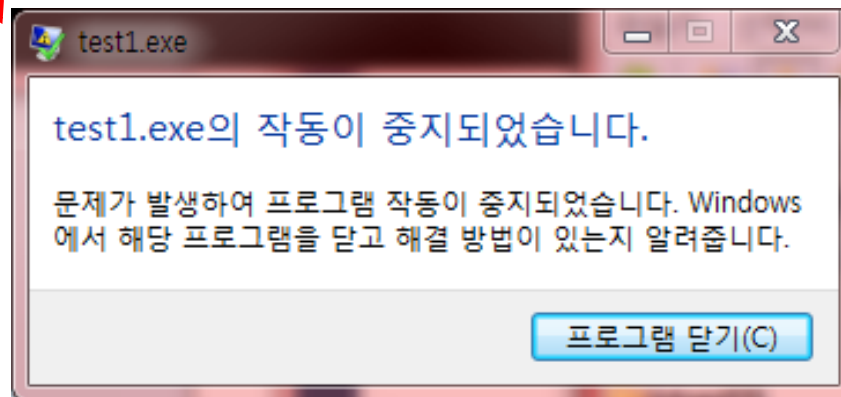
```
const char *p = "HelloWorld";
```



문자열 상수와 포인터

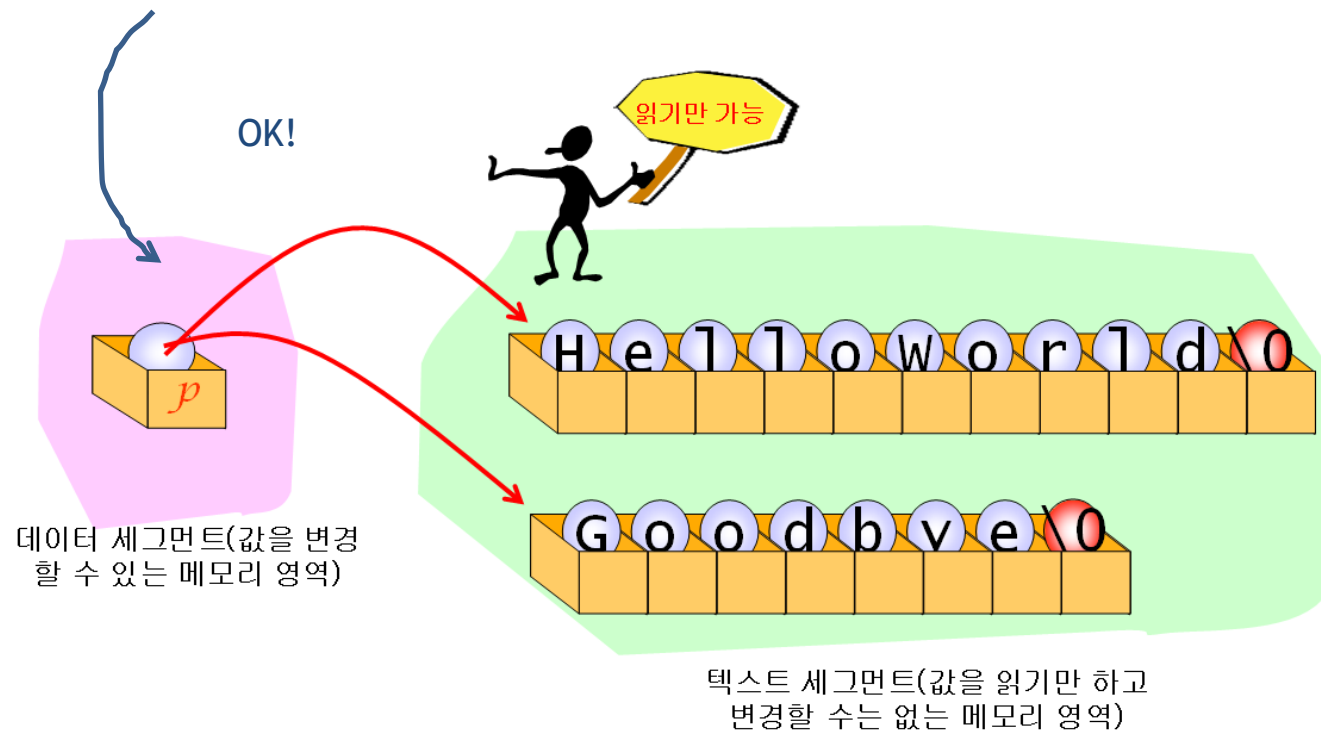
```
const char *p = "HelloWorld";  
p[0] = 'A'; // 또는 strcpy(p, "Goodbye");
```

p를 통하여 텍스트 세그먼트에 문자를
저장하려면 오류가 발생한다.



문자열 상수와 포인터

```
const char *p = "HelloWorld";  
p = "Goodbye";
```



문자열 입출력 라이브러리 함수

| 입출력 함수 | 설명 |
|--------------------------------------|---------------------------------|
| <code>int scanf("%s", s)</code> | 문자열을 읽어서 문자배열 s[]에 저장 |
| <code>int printf("%s", s)</code> | 배열 s[]에 저장되어 있는 문자열을 출력한다. |
| <code>char *gets(char *s)</code> | 한 줄의 문자열을 읽어서 문자 배열 s[]에 저장한다. |
| <code>int puts(const char *s)</code> | 배열 s[]에 저장되어 있는 한 줄의 문자열을 출력한다. |

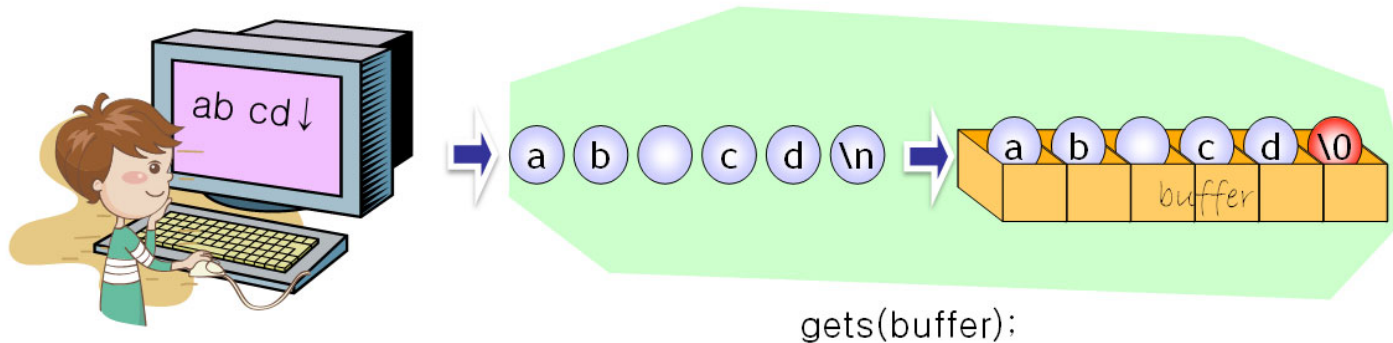


gets() - 문자열 입력

```
char *gets(char *buffer);  
int puts(const char *str);
```

◆ gets()

- 표준 입력으로부터 엔터키 (enter key)가 입력될 때까지 한 줄의 라인을 입력
- 문자열에 줄바꿈 문자('\n')는 포함되지 않으며 대신에 자동으로 NULL 문자('\0')를 추가한다.
- 입력받은 문자열은 buffer가 가리키는 주소에 저장된다.



puts() - 문자열 출력

```
char *gets(char *buffer);  
int puts(const char *str);
```

◆ puts()

- const char pointer str이 가리키는 문자열을 받아서 화면에 출력
- NULL 문자('\0')는 줄바꿈 문자('\n')로 변경

```
const char *menu = "File open: fopen; File close: fclose";  
puts(menu);
```



gets()와 puts()를 사용한 문자열 입출력 예제

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char name[100];
```

```
    char address[100];
```

```
    printf("이름을 입력하시오: ");
```

```
    gets(name);
```

```
    printf("현재 거주하는 주소를 입력하시오: ");
```

```
    gets(address);
```

```
    puts(name);
```

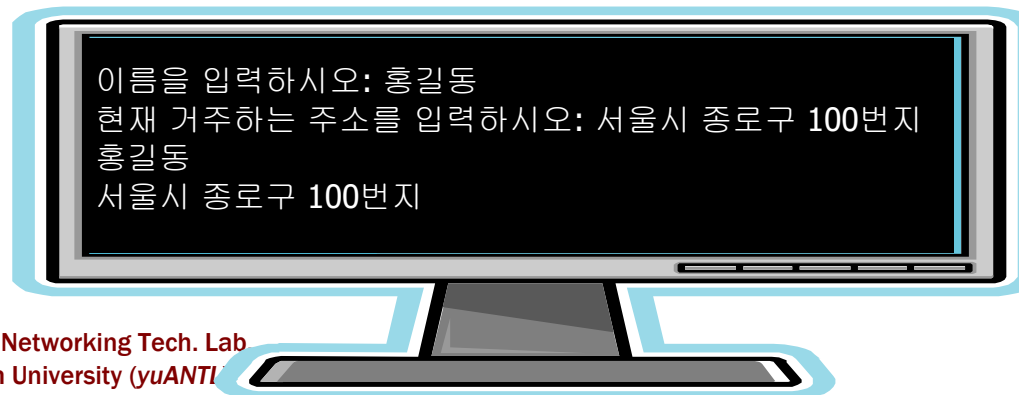
```
    puts(address);
```

```
    return 0;
```

```
}
```

문자열 (한 단어 이상)
을 입력 받을 때 사용

문자열 (한 단어 이상)
을 출력 할 때 사용



문자열 처리 라이브러리 함수 (1)

문자열 처리 라이브러리

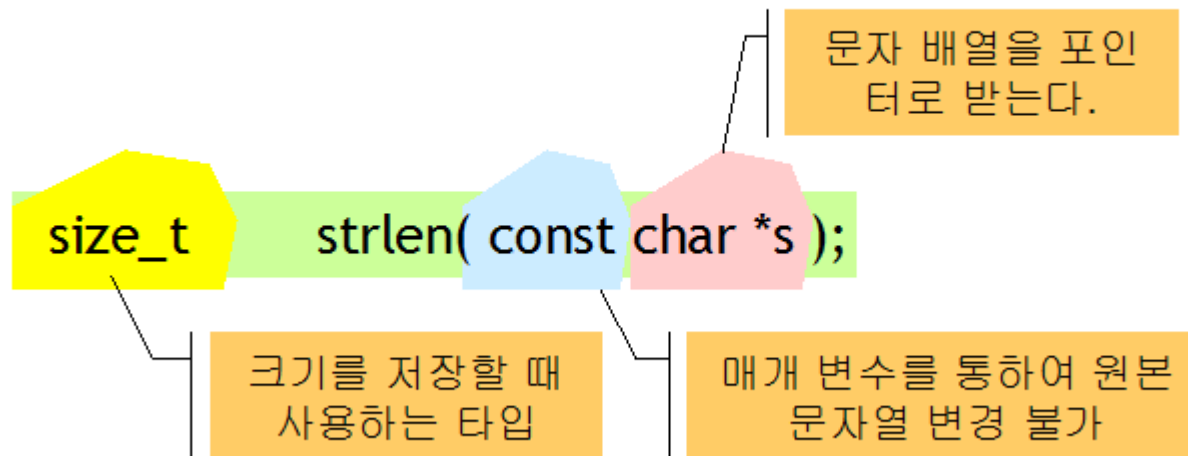
| 문자열 처리 함수 | 함수 설명 |
|--|---|
| <code>int strlen(const char *str);</code> | 문자열 str의 길이를 반환 |
| <code>strcpy(char *s1, const char *s2);</code> | 문자열 s2를 s1으로 복사 |
| <code>strncpy(char *s1, const char *s2, int n);</code> | 문자열 s2를 s1으로 최대 n개 문자까지 복사 |
| <code>strcat(char *s1, const char *s2);</code> | 문자열 s2를 s1에 이어 붙임 (concatenate) |
| <code>strncat(char *s1, char *s2, int n);</code> | 문자열 s2의 최대 n개 문자를 s1에 이어 붙임 (concatenate) |
| <code>strcmp(const char *s1, const char *s2);</code> | 문자열 s1과 s2를 비교 |
| <code>strncmp(const char *s1, const char *s2, int n);</code> | 문자열 s1과 s2를 최대 n개 문자까지 비교 |
| <code>strchr(const char *str, char c);</code> | 문자열 str에 문자c가 포함되어 있는가를 확인 |
| <code>strstr(const char *s1, const char *s2);</code> | 문자열 s1에 문자열 s2가 포함되어 있는가를 확인 |
| <code>char *strtok(char *str, const char *delimit);</code> | 문자열 str로 부터 delimit에 포함된 문자들을 기준으로 토큰 단위로 분리한다. |



문자열 길이 – strlen()

◆ 문자열의 길이

- strlen("Hello")는 5를 반환



문자열 길이 (string length) 계산 예제

```
// 문자열의 길이를 구하는 프로그램
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char str[30] = "C language is easy";
```

```
    int len = 0;
```

```
    while(str[len] != 0)
```

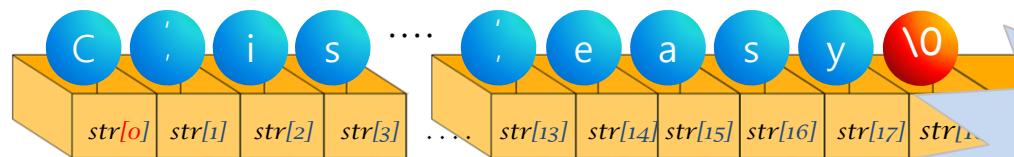
```
        len++;
```

```
    printf("문자열 \"%s\"의 길이는 %d입니다.\n", str, len);
```

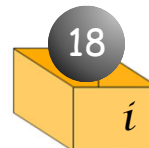
```
    return 0;
```

```
}
```

문자열 "C language is easy"의 길이는 18입니다.



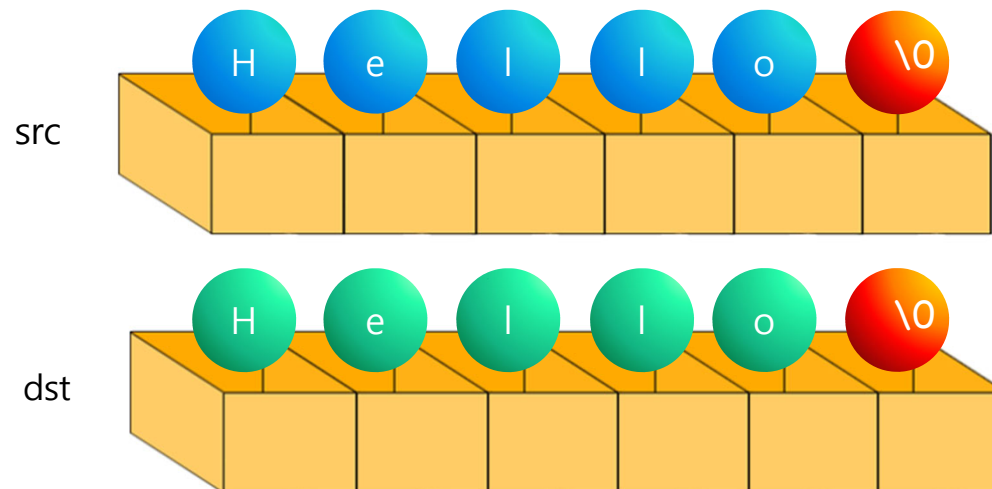
.....
str[18] == 0
이므로 카운터
종료
('\0'의 아스키
코드 값은 0이다)



문자열 복사 – strcpy()

◆ 문자열 복사


```
char dst[6];  
char src[6] = "Hello";  
strcpy(dst, src);
```



문자열 복사 (string copy) 예제

```
#include <stdio.h>
int main(void)
{
    char src[] = "The worst things to eat before you sleep";
    char dst[100];
    int i;
    printf("원본 문자열=%s\n", src);
    for(i=0 ; src[i] != NULL ; i++)
        dst[i] = src[i];
    dst[i] = NULL;
    printf("복사된 문자열=%s\n", dst);
    return 0;
}
```

← NULL 문자가 나올 때까지 반복하면서 각각의 문자들을 새로운 배열로 복사한다.



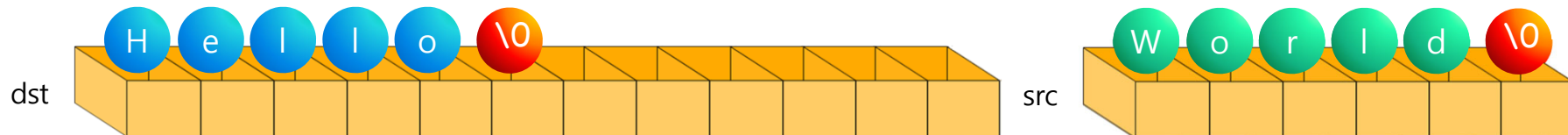
원본 문자열=The worst things to eat before you sleep
복사된 문자열=The worst things to eat before you sleep



문자열 연결 – strcat()

◆ 문자열 연결 (concatenation)

```
char dst[12] = "Hello";  
char src[6] = "World";  
strcat(dst, src);
```



예제

```
// strcpy와 strcat  
#include <string.h>  
#include <stdio.h>
```

```
int main( void )
```

```
{  
    char string[80];
```

```
    strcpy( string, "Hello world from" );
```

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

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

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

```
    printf( "string = %s\n", string );
```

```
    return 0;
```

```
}
```

복사하기

문자열 이어 붙이기
(concatenate)

string = Hello world from strcpy and strcat!



strcpy(), strcat를 사용한 파일 이름 편집 예제

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char filename[100];
    char s[100];
    int i;
    for(i=0; i < 6; i++){
        strcpy(filename, "image");
        sprintf(s, "%d", i);
        strcat(filename, s);
        strcat(filename, ".jpg");
        printf("%s \n", filename);
    }
    return 0;
}
```



```
image0.jpg
image1.jpg
image2.jpg
image3.jpg
image4.jpg
image5.jpg
```



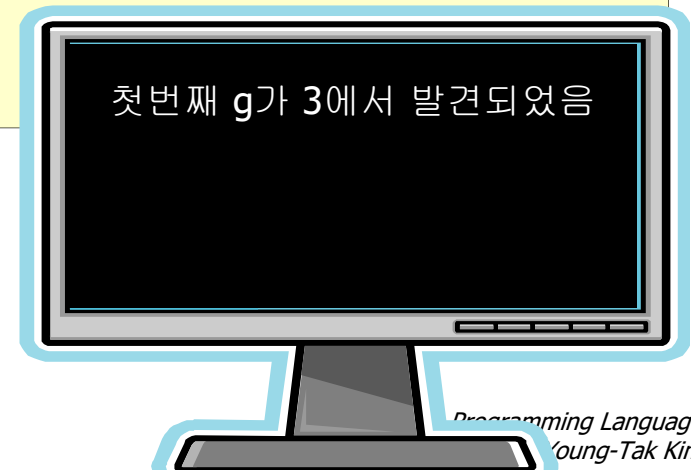
문자 검색 (char search) - strchr()

```
#include <string.h>
#include <stdio.h>
int main( void )
{   char s[] = "language";
    char c = 'g';
    char *p;
    int loc;

    p = strchr(s, c);
    loc = (int)(p - s);
    if ( p != NULL )
        printf( "첫번째 %c가 %d에서 발견되었음\n", c, loc );
    else
        printf( "%c가 발견되지 않았음\n", c );
    return 0;
}
```

s 안에서 문자 c를 찾는다.

첫번째 g가 3에서 발견되었음



문자열 검색 (string search) - strstr()

```
#include <string.h>
#include <stdio.h>
int main( void )
{
    char s[] = "A joy that's shared is a joy made double";
    char sub[] = "joy";
    char *p;
    int loc;
    p = strstr(s, sub);
    loc = (int)(p - s);
    if ( p != NULL )
        printf( "첫번째 %s가 %d에서 발견되었음\n", sub, loc );
    else
        printf( "%s가 발견되지 않았음\n", sub );
}
```

s 안에서 문자열 sub를 찾는다.

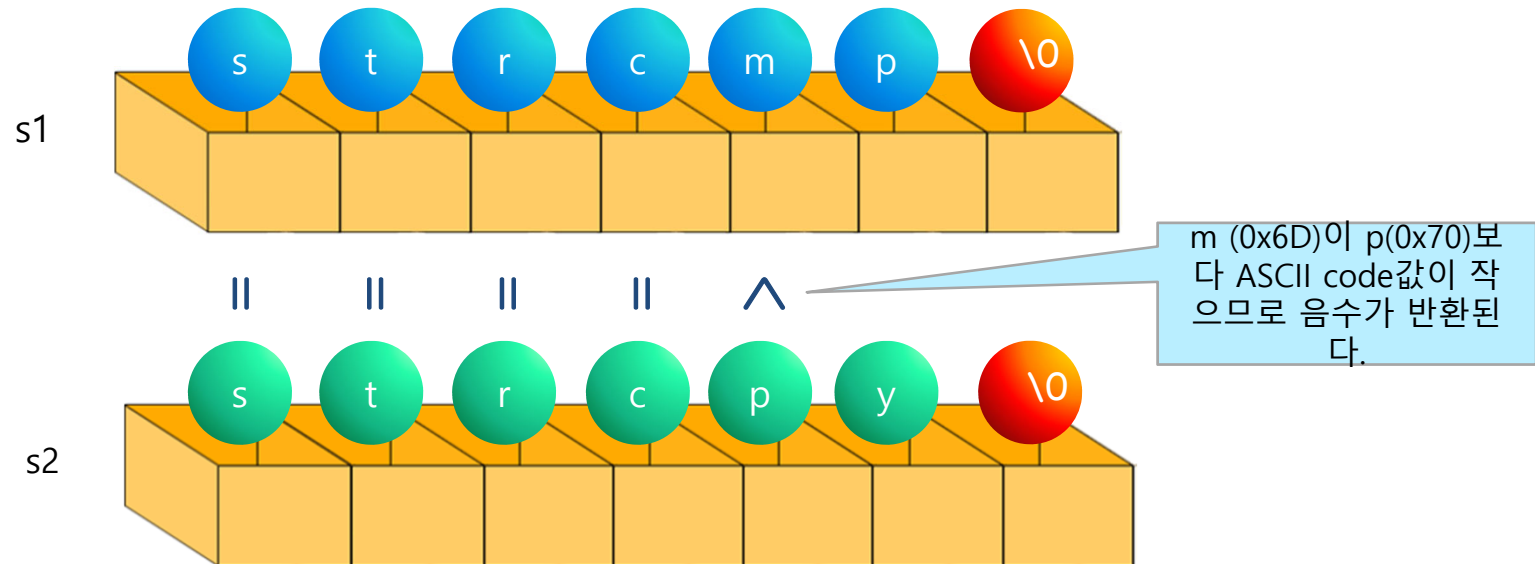
첫번째 joy가 2에서 발견되었음



문자열 비교 - strcmp()

```
int strcmp( const char *s1, const char *s2 );
```

| 반환값 | s1과 s2의 관계 |
|-----|--------------|
| < 0 | s1이 s2보다 작다 |
| 0 | s1이 s2와 같다. |
| > 0 | s1이 s2보다 크다. |




예제

```
// strcmp() 함수
#include <string.h>
#include <stdio.h>

int main( void )
{
    char s1[80]; // 첫번째 단어를 저장할 문자배열
    char s2[80]; // 두번째 단어를 저장할 문자배열
    int result;

    printf("첫번째 단어를 입력하시오:");
    scanf("%s", s1);
    printf("두번째 단어를 입력하시오:");
    scanf("%s", s2);

    result = strcmp(s1, s2);
    if( result < 0 )
        printf("%s가 %s보다 앞에 있습니다.\n", s1, s2);
    else if( result == 0 )
        printf("%s가 %s와 같습니다.\n", s1, s2);
    else
        printf("%s가 %s보다 뒤에 있습니다.\n", s1, s2);
    return 0;
}
```



첫번째 단어를 입력하시오:Hello
두번째 단어를 입력하시오:World
Hello가 World보다 앞에 있습니다.



문자열 배열의 정렬 – sorting of string array

```
void stringSort(char **words, int num_words, int max_word_len)
{
    char *temp_word;
    int i, j, min;
    temp_word = (char *)calloc(max_word_len, sizeof(char));
    if (temp_word == NULL)
    {
        printf("Error in dynamic allocation of memory for temp_word !!\n");
        exit;
    }

    /* selection sorting of words */
    for (i = 0; i < num_words; i++)
    {
        strcpy(temp_word, words[i]);
        min = i;
        for (j = i + 1; j < num_words; j++)
        {
            if (strcmp(temp_word, words[j]) > 0)
            {
                min = j;
                strcpy(temp_word, words[j]);
            }
        }
        if (min != i)
        {
            /* temp_word contains words[min] already !!*/
            strcpy(words[min], words[i]);
            strcpy(words[i], temp_word);
        }
    }
}
```



문자열의 토큰 단위 분리를 위한 strtok()

| | |
|----|---|
| 형식 | <code>char *strtok(char *str, const char *delimit);</code> |
| 설명 | <p>strtok 함수는 문자열 str로 부터 delimit에 포함된 문자들을 기준으로 토큰 단위로 분리한다.</p> <p>토큰을 받아올 대상 문자열 (str)은 맨 처음 호출 시에는 인수로 전달하며, 이 후 이 str 문자열에 대하여 연속적으로 다음 토큰을 받아올 때에는 NULL을 전달</p> |

```
const char delimiters[] = “ ,\t\n”; // delimiter: space, comma, tab, new line
t1 = strtok(s, delimiters); // 첫 번째 토큰
t2 = strtok(NULL, delimiters); // 두 번째 토큰
t3 = strtok(NULL, delimiters); // 세 번째 토큰
t4 = strtok(NULL, delimiters); // 네 번째 토큰
```



문자열의 토큰 단위 분리 (string token separation)

```
// strtok 함수의 사용 예
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
char s[] = "Man is immortal, because he has a soul";
```

```
const char delimiters[] = " ,\t\n"; // space, comma, tab, newline
```

```
char *token;
```

delimiter
(분리자)

```
int main( void )
```

```
{
```

```
    // 문자열을 전달하고 다음 토큰을 얻는다.
```

```
    token = strtok( s, delimiters );
```

```
    while( token != NULL )
```

```
    {
```

```
        // 문자열 s에 토큰이 있는 동안 반복한다.
```


```
        printf( "토큰: %s\n", token );
```

```
        // 다음 토큰을 얻는다.
```

```
        token = strtok( NULL, delimiters ); //
```

```
    }
```

```
}
```



```
토큰: Man  
토큰: is  
토큰: immortal  
토큰: because  
토큰: he  
토큰: has  
토큰: a  
토큰: soul
```



```
/** StringTokenHandling.c (1) */
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#define MAX_STRING_LEN 256
```

```
int main()
```

```
{
```

```
    char delimiters[] = ":\t\n"; // token 구분자 : colon, comma, space, tab, new line
```

```
    char *token;
```

```
    char str[MAX_STRING_LEN] = { 0 };
```

```
    FILE *fp_in;
```

```
    int line_count = 1;
```

```
    fp_in = fopen("TestInput.txt", "r");
```

```
    if (fp_in == NULL)
```

```
    {
```

```
        printf("Error in file open - TestInput.txt !!\n");
```

```
        exit(-1);
```

```
    }
```



```

/** StringTokenHandling.c (2) */

printf("Testing fgets() and strtok() ...\n");
while (fgets(str, MAX_STRING_LEN, fp_in) != NULL)
{
    printf("Line %2d : %s", line_count++, str);
    token = strtok(str, delimiters);
    int i = 1;
    printf("\t");
    while (token != NULL)
    {
        printf("(Token %d) %s ", i, token);
        token = strtok(NULL, delimiters);
        i++;
    }
    printf("\n");
}

fclose(fp_in);
}

```

```

Testing fgets() and strtok() ...
Line 1 : Test input file.
        (Token 1) Test (Token 2) input (Token 3) file.
Line 2 : The second line.
        (Token 1) The (Token 2) second (Token 3) line.
Line 3 : The third line with numbers : 0123456789.
        (Token 1) The (Token 2) third (Token 3) line (Token 4) with (Token 5) numbers (Token 6) 0123456789.

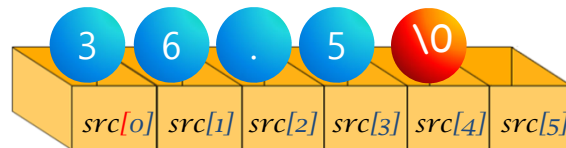
```



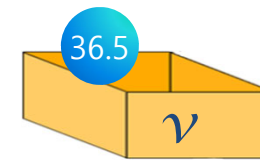
문자열 처리 라이브러리 함수 (2)

문자열 숫자 변환 – scanf()

◆ 문자열과 수치

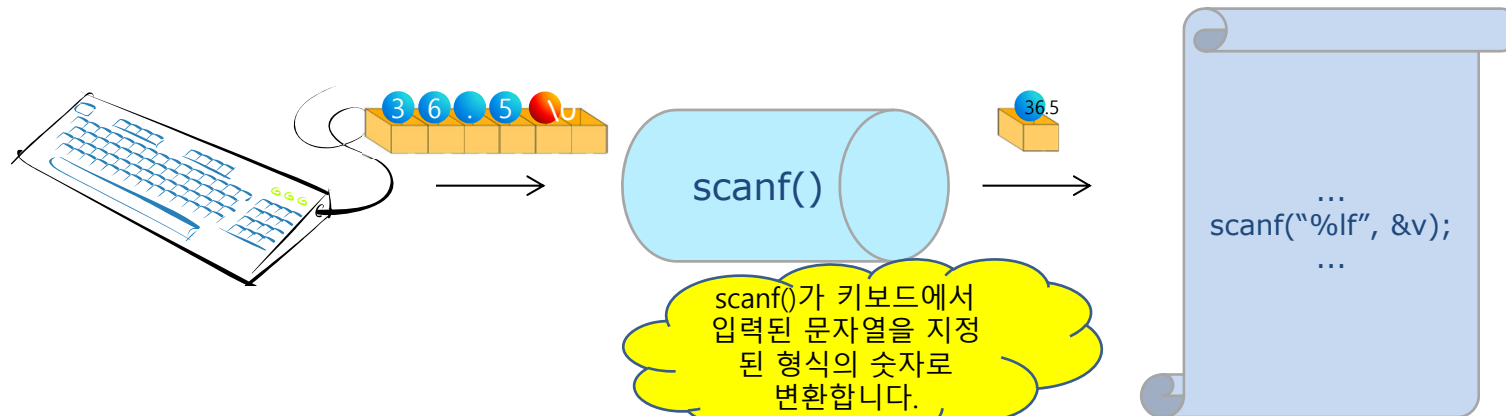


문자열



수치 (값)

◆ scanf() 함수는 문자열을 지정된 포맷의 수치로 변환



scanf()가 키보드에서
입력된 문자열을 지정
된 형식의 숫자로
변환합니다.



sscanf(), sprintf()

◆ 문자열 버퍼로의 포맷 지정 출력 및 포맷지정 입력

- sscanf() : 지정된 포맷에 맞추어 문자열 버퍼로부터 입력
- sprintf() : 지정된 포맷에 맞추어 문자열 버퍼로 출력

| 문자열의 포맷지정 입출력 함수 | 함수 설명 |
|---|----------------------------------|
| int sscanf(const char* str_buf, const char* format, ...); | 문자열 버퍼 str_buf로부터 지정된 포맷에 맞추어 입력 |
| int sprintf(char* str_buf, const char* format, ...); | 문자열 버퍼 str_buf로 지정된 포맷에 맞추어 출력 |



sscanf(), sprintf() 예제

```
#include <stdio.h>
int main( void )
{
    char s1[] = "100 200 300";
    char s2[30];
    int value;

    sscanf(s1, "%d", &value);
    printf("%d\n", value);
    sprintf(s2, "%d", value);
    printf("%s\n", s2);

    return 0;
}
```



문자열을 숫자로 변환하는 전용함수

◆ 문자열 전용 함수

- 다양한 형식을 처리하는 scanf()보다 크기가 작으며, 간편하게 사용할 수 있다.
- stdlib.h에 원형 정의 - 반드시 포함 (#include <stdlib.h>)

| 문자열의 수치변환 함수 | 함수 설명 |
|-------------------------------|---------------------------|
| int atoi(const char *str); | 주어진 문자열 str을 정수 데이터로 변환 |
| long atol(const char *str); | 주어진 문자열 str을 long 데이터로 변환 |
| double atof(const char *str); | 주어진 문자열 str을 더블형 데이터로 변환 |



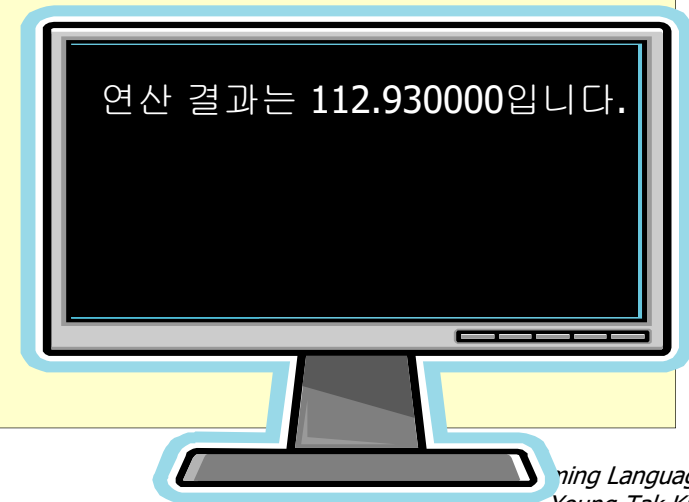
문자열 수치 변환

```
#include <stdio.h>
#include <stdlib.h> // atoi(), atof() 함수
int main( void )
{
    char s1[] = "100";
    char s2[] = "12.93";
    char buffer[100];
    int i;
    double d, result;

    i = atoi(s1);
    d = atof(s2);
    result = i + d;

    sprintf(buffer, "%lf", result);
    printf("연산 결과는 %s입니다.\n", buffer);

    return 0;
}
```



16진수 문자열을 정수로 변환 – atoi()

```
unsigned int atoi(char *hexStr)
{
    unsigned char uc, hexChar;
    unsigned int hexVal = 0, hexSum = 0;

    for (int i = 0; i < MAX_HEX_STR_LEN; i++)
    {
        hexChar = hexStr[i];
        if (hexChar == NULL)
            break;
        if ((hexChar >= '0') && (hexChar <= '9'))
            hexVal = hexChar - '0';
        else if ((hexChar >= 'A') && (hexChar <= 'F'))
            hexVal = hexChar - 'A' + 10;
        else if ((hexChar >= 'a') && (hexChar <= 'f'))
            hexVal = hexChar - 'a' + 10;
        else
        {
            printf("Error in atoi() :: given hexStr (%x) is
                not hex code character !!\n", hexChar);
            continue;
        }
        hexSum = hexSum * 16 + hexVal;
    }
    return hexSum;
}
```

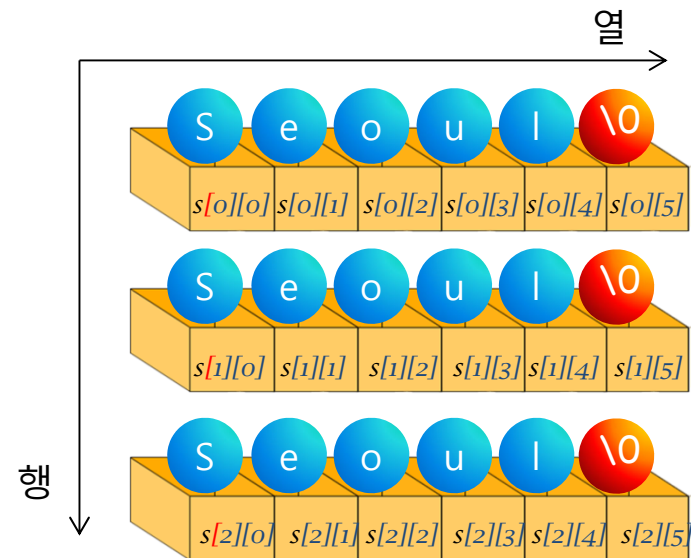
문자열의 배열

◆ (Q) 문자열이 여러 개 있는 경우에는 어떤 구조를 사용하여 저장하면 제일 좋을까?

- (A) 여러 개의 문자 배열을 각각 만들어도 되지만 문자열의 배열을 만드는 것이 여러모로 간편하다.

◆ 문자열이 문자 배열에 저장되므로 문자열의 배열은 배열의 배열, 즉 2차원 문자 배열이 된다.

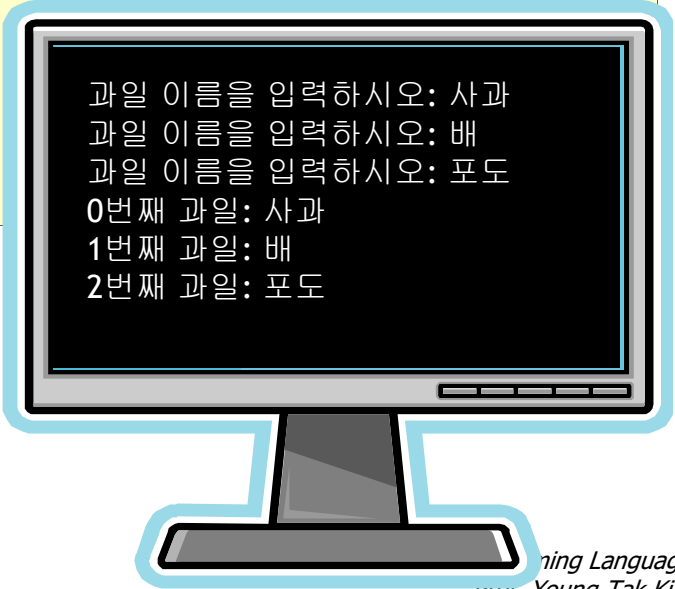
```
char s[3][6] = {  
    "init",  
    "open",  
    "close"  
};
```



문자열 배열 구성

- 문자열을 입력받아 2차원 문자 배열에 저장

```
#include <stdio.h>
#define MAX_STR_LEN 20
int main( void )
{
    int i;
    char fruits[3][MAX_STR_LEN];
    for(i = 0; i < 3; i++) {
        printf("과일 이름을 입력하시오: ");
        scanf("%s", fruits[i]);
    }
    for(i = 0; i < 3; i++)
        printf("%d번째 과일: %s\n", i, fruits[i]);
    return 0;
}
```

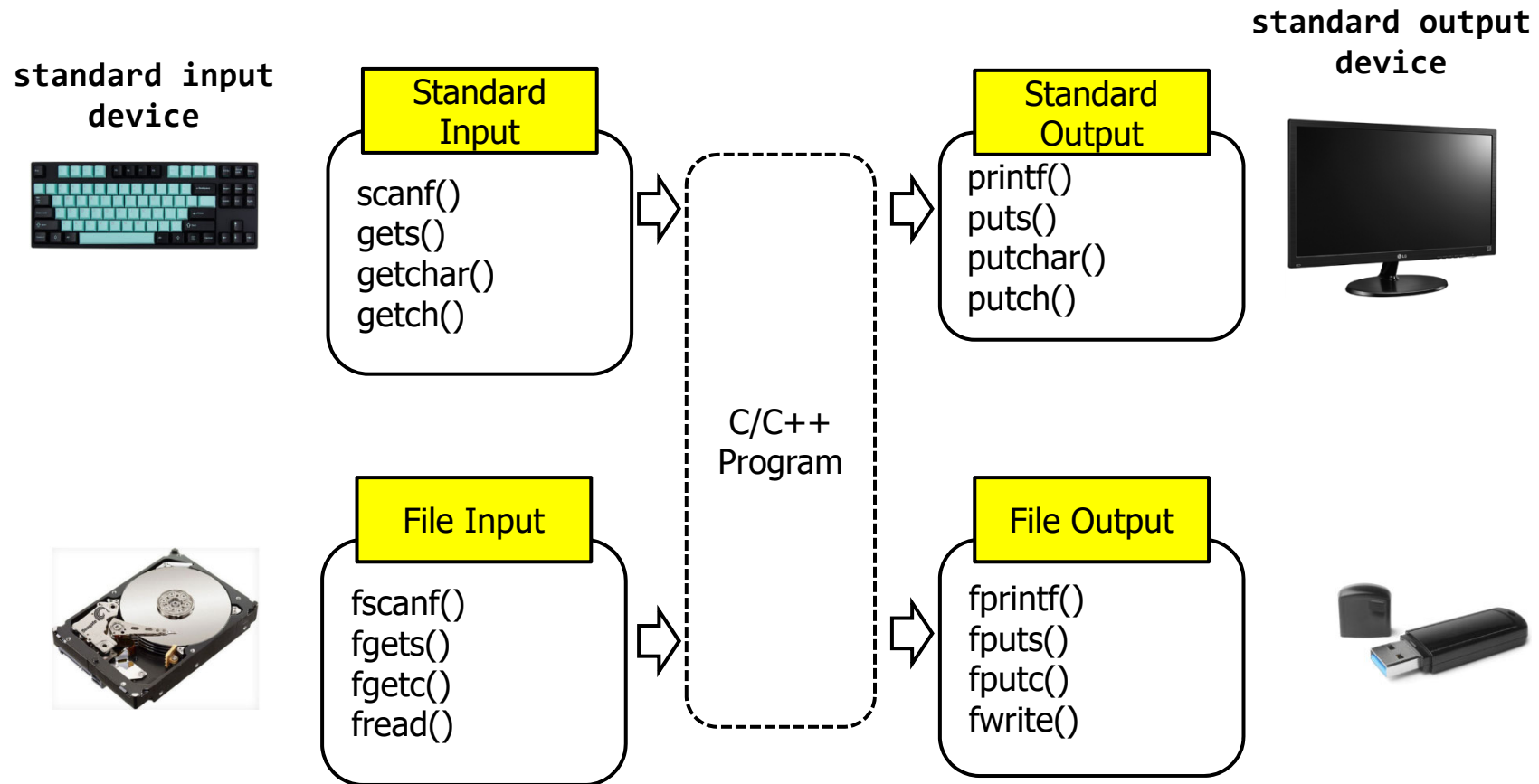


과일 이름을 입력하시오: 사과
과일 이름을 입력하시오: 배
과일 이름을 입력하시오: 포도
0번째 과일: 사과
1번째 과일: 배
2번째 과일: 포도



텍스트 파일 입출력

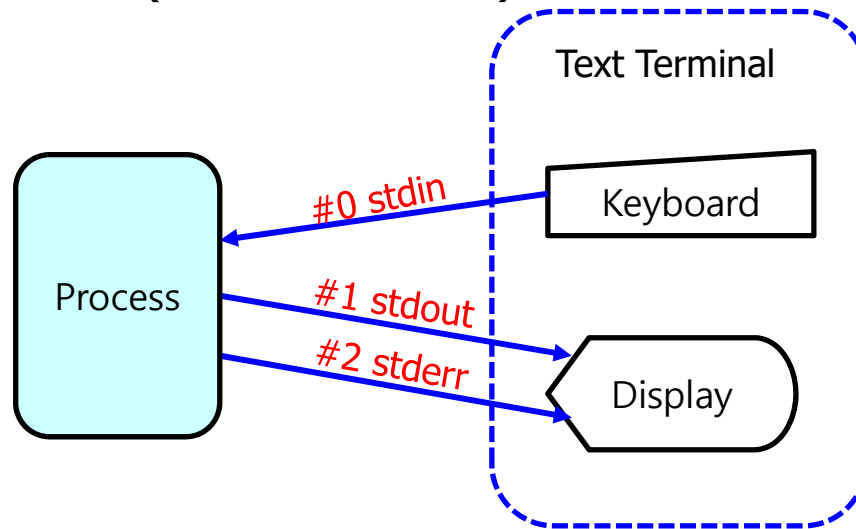
C/C++ 프로그램의 입력과 출력



표준 입출력 파일 서술자

◆ 표준 File Descriptor

- stdin (standard input)
- stdout (standard output)
- stderr (standard error)



파일 입출력 관련 함수

| 유형 | 파일 입출력 함수 | 설명 |
|-----------------------|---|---|
| 파일 open /close | FILE *fopen (const char *name, const char *mode); | name으로 지정된 파일을 지정된 mode (r, w, a)로 사용할 수 있도록 open하며, FILE 포인터를 반환 |
| | int fclose (FILE *fp); | FILE 포인터 fp로 지정된 파일을 close |
| 문자 (char) 단위 | int fgetc(FILE *fp) | 문자 하나 읽기 |
| | int fputc(int c, FILE *fp) | 문자 하나 쓰기 |
| 문자열 (string) 단위 | char *fgets(char *buf, int max_size, FILE *fp) | 문자열을 최대 max_size 만큼 읽어 버퍼에 저장. |
| | int fputs(const char *buf, FILE *fp) | 버퍼에 저장되어 있는 문자열을 파일에 쓰기 |
| 포맷이 지정된 입출력 | int fscanf(FILE *fp, const char *format, ...) | 지정된 포맷으로 파일을 읽어 해당 항목으로 복사 |
| | int fprintf(FILE *fp, const char *format, ...) | 지정된 포맷으로 해당 항목들을 파일로 출력 |
| 버퍼링 | fflush(FILE *fp) | 파일의 입력 버퍼 또는 출력 버퍼에 쌓여있는 데이터의 입출력이 완료되게 함 |
| 이진 데이터 파일 | size_t fread(char *buf, int record_size, int count, FILE *fp) | 이진 데이터 파일로부터 record_size 만큼 읽고, 이를 버퍼에 저장 |
| | size_t fwrite(char *buf, int record_size, int count, FILE *fp) | 버퍼에 저장되어 있는 데이터 레코드를 이진 파일로 출력 |



File open (1)

◆ File open의 의미

- 파일에서 데이터를 읽거나 쓸 수 있도록 모든 준비를 마치는 것을 의미
- 파일을 연 다음에는 데이터를 읽기, 쓰기 가능
- File open → File read & write → File close 순으로 진행
- FILE 구조체를 통하여 파일에 접근
 - FILE 구조체를 가리키는 포인터를 파일 포인터 (file pointer)라고 한다
 - 각각의 파일마다 하나의 파일 포인터가 필요

```
FILE *fopen (const char *name, const char *mode);
```

name : 파일의 이름을 나타내는 문자열
mode : 파일을 여는 방식



File open (2)

◆ File mode

| 모드 | 설명 |
|------|---|
| "r" | 읽기 모드로 파일을 연다. |
| "w" | 쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 이미 존재하면 기존의 내용이 지워진다. |
| "a" | 추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일이 만들어진다. |
| "r+" | 읽기와 쓰기 모드로 파일을 연다. 파일이 반드시 존재해야 한다. |
| "w+" | 읽기와 쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 존재하면 새 데이터가 기존 파일의 데이터에 덮어 쓰인다. |
| "a+" | 읽기와 추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 읽기는 어떤 위치에서나 가능하다. 파일이 없으면 새로운 파일을 만든다. |
| "b" | 이진 파일 모드로 파일을 연다. |



File close

◆ fclose()

- 열린 파일을 닫는 함수
- stdio.h에 정의
- 성공적으로 파일을 닫는 경우에는 0이 반환
- 만약 실패한 경우에는 -1이 반환

```
int fclose (FILE *stream);
```



File open/close

◆ Sample.txt 생성

- Project -> sample.txt 생성

◆ File open/close 예제

```
#include <stdio.h>

int main()
{
    FILE *fp = NULL;    //FILE 포인터fp를 생성하고NULL로 초기화
    fp = fopen("sample.txt", "w"); //파일을 쓰기모드로 열고, 그 주소를 fp에 저장

    if (fp == NULL)
        printf("파일열기실패\n");
    else
        printf("파일열기성공\n");

    fclose(fp);    //파일 닫기

    return 0;
}
```



텍스트 파일 읽기와 쓰기 (1)

◆ 형식화된 입출력

- 정수나 실수 데이터를 파일에 문자열로 바꾸어서 저장
- fprintf()
 - 사용방법은 printf()와 비슷하나, 화면이 아닌 파일에 출력

```
int fprintf( FILE *fp, const char *format, ...);
```

▪ fprintf() 사용 예제

```
#include <stdio.h>
int main()
{
    int i = 23;
    float f = 1.2345;
    FILE *fp = NULL;          //FILE 포인터fp를 생성하고NULL로 초기화
    fp = fopen("sample.txt", "w"); //파일을 쓰기모드로 열고, 그 주소를 fp에 저장

    if(fp != NULL)
        fprintf(fp, "%10d %16.4f", i, f);
    fclose(fp);               //파일 닫기
    return 0;
}
```



텍스트 파일 읽기와 쓰기 (2)

- fscanf()

- scanf()와 사용법은 비슷하지만 입력 대상이 키보드가 아닌 파일

```
int fscanf( FILE *fp, const char *format, ...);
```

- fscanf() 사용 예제

```
#include <stdio.h>

int main()
{
    int i;
    float f;
    FILE *fp = NULL;    //FILE 포인터fp를 생성하고 NULL로 초기화
    fp = fopen("sample.txt", "r"); //파일을 읽기모드로 열고, 그 주소를 fp에 저장

    if(fp != NULL)
        fscanf(fp, "%d %f", &i, &f);
    printf(" %d\n %f\n", i, f); //화면에 출력
    fclose(fp);    //파일 닫기

    return 0;
}
```



파일 입력, 출력 예제 프로그램

```
/** SimpleFileInputOutput.cpp */

#include <stdio.h>
#include <string.h>

#define MAX_WORD_LEN 50
#define NUM_WORDS 100

int main()
{
    FILE *pFin = NULL;
    FILE *pFout = NULL;

    char str[80];
    char wordList[NUM_WORDS][MAX_WORD_LEN];
    int word_count;

    pFin = fopen("input.txt", "r");
    if (pFin == NULL)
    {
        printf("Error in input data file open !!\n");
        return 0;
    }
}
```



```

/** FileInputOutput.cpp (2) */

pFout = fopen("output.txt", "w");
if (pFout == NULL)
{
    printf("Error in output data file creation !!\n");
    return 0;
}

word_count = 0;
while (fscanf(pFin, "%s", str) != EOF)
{
    printf("%2d-th input word: %s\n", word_count, str);
    strcpy(wordList[word_count], str);
    word_count++;
}

printf("Number of words: %d\n", word_count);

for (int i=0; i<word_count; i++)
{
    fprintf(pFout, "wordList[%2d]: %s (length: %d)\n",
            i, wordList[i], strlen(wordList[i]));
}
fprintf(pFout, "\n");
fclose(pFin);
fclose(pFout);
}

```



◆ input.txt

```
one two three four five six seven eight nine ten
January February March April May June July August September October
Sunday Monday Tuesday Wednesday Thursday Friday Saturday week day month
China India UnitedStates Indonesia Brazil Pakistan Nigeria Russia Bangladesh
Japan Mexico Philippines Vietnam Ethiopia Germany Egypt Iran Turkey Congo
Thailand France UnitedKingdom Italy
```

◆ output.txt

```
wordList[ 0]: one (length: 3)
wordList[ 1]: two (length: 3)
wordList[ 2]: three (length: 5)
wordList[ 3]: four (length: 4)
wordList[ 4]: five (length: 4)
wordList[ 5]: six (length: 3)
wordList[ 6]: seven (length: 5)
wordList[ 7]: eight (length: 5)
wordList[ 8]: nine (length: 4)
wordList[ 9]: ten (length: 3)
wordList[10]: January (length: 7)
wordList[11]: February (length: 8)
wordList[12]: March (length: 5)
wordList[13]: April (length: 5)
wordList[14]: May (length: 3)
wordList[15]: June (length: 4)
wordList[16]: July (length: 4)
wordList[17]: August (length: 6)
wordList[18]: September (length: 9)
wordList[19]: October (length: 7)
```

```
wordList[20]: Sunday (length: 6)
wordList[21]: Monday (length: 6)
wordList[22]: Tuesday (length: 7)
wordList[23]: Wednesday (length: 9)
wordList[24]: Thursday (length: 8)
wordList[25]: Friday (length: 6)
wordList[26]: Saturday (length: 8)
wordList[27]: week (length: 4)
wordList[28]: day (length: 3)
wordList[29]: month (length: 5)
wordList[30]: China (length: 5)
wordList[31]: India (length: 5)
wordList[32]: UnitedStates (length: 12)
wordList[33]: Indonesia (length: 9)
wordList[34]: Brazil (length: 6)
wordList[35]: Pakistan (length: 8)
wordList[36]: Nigeria (length: 7)
wordList[37]: Russia (length: 6)
wordList[38]: Bangladesh (length: 10)
wordList[39]: Japan (length: 5)
```

```
wordList[40]: Mexico (length: 6)
wordList[41]: Philippines (length: 11)
wordList[42]: Vietnam (length: 7)
wordList[43]: Ethiopia (length: 8)
wordList[44]: Germany (length: 7)
wordList[45]: Egypt (length: 5)
wordList[46]: Iran (length: 4)
wordList[47]: Turkey (length: 6)
wordList[48]: Congo (length: 5)
wordList[49]: Thailand (length: 8)
wordList[50]: France (length: 6)
wordList[51]: UnitedKingdom (length: 13)
wordList[52]: Italy (length: 5)
```



랜덤파일 입출력

랜덤 파일 입출력

◆ 랜덤 파일 접근 관련 함수

- fseek() 함수를 사용하여 임의의 파일 위치 (offset로 지정)로 파일 포인터 이동한 후 파일 입력 또는 출력 가능
- rewind()는 파일 포인터를 맨 처음으로 이동시킴
- 구조체로 표현된 data record 단위로 이동

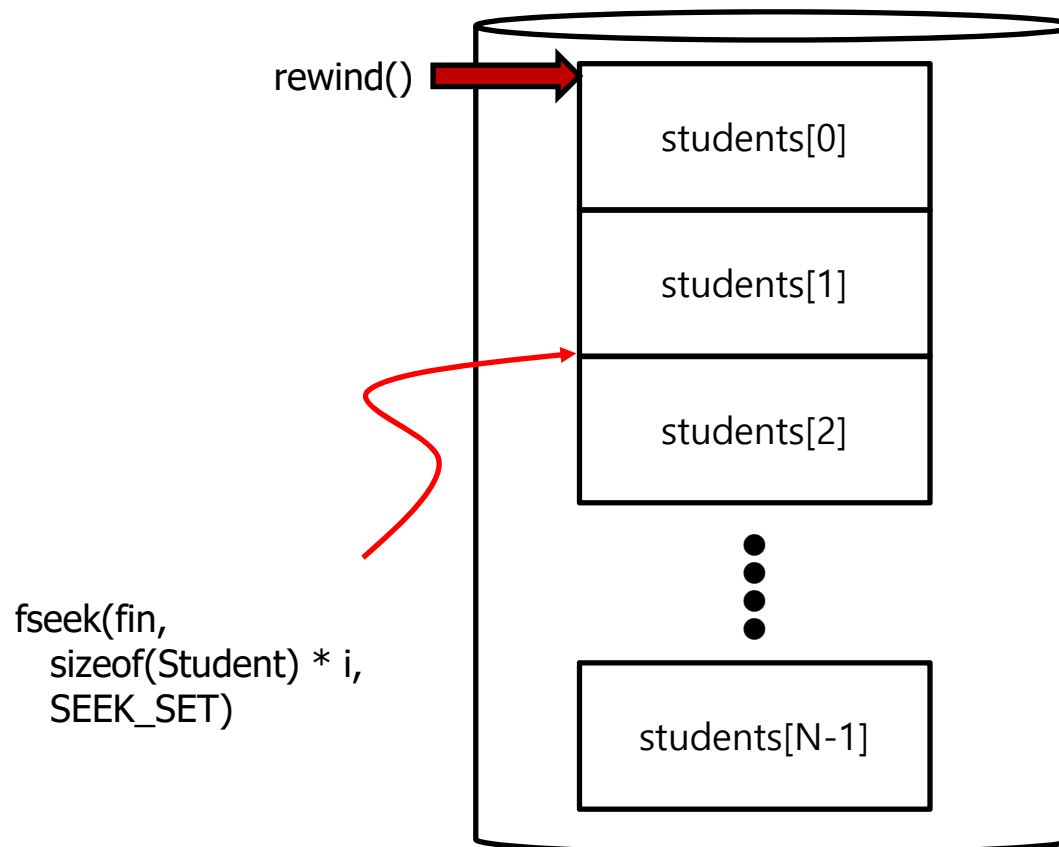
| 랜덤 파일 접근 관련 함수 | 의미 |
|---|---|
| int fseek(FILE *fp, long offset, int origin) | 지정된 시작위치 (origin)으로 부터 지정된 offset 만큼 이동 Origin으로 설정가능한 Macro: - SEEK_SET : 파일의 처음 - SEEK_END : 파일의 끝 - SEEK_CUR : 현재의 커서 위치 |
| void rewind(FILE *fp) | 파일 포인터를 파일의 시작 위치로 이동 |
| long ftell(FILE *fp) | 파일 포인터의 현재 위치를 반환 |
| int feof(FILE *fp) | 파일의 끝에 도달하였는가를 알려줌. 이진 파일 입출력에서는 주로 사용 |



랜덤 파일 입출력을 위한 파일 포인터 이동

◆ `fseek()`, `rewind()`를 사용한 파일 포인터 이동

Random access of file with student records




```

/* main.c (1) */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Student.h"

#define NUM_STUDENTS 10
#define MAX_STRING_LEN 512
extern STUDENT students[NUM_STUDENTS];
#define TEST_SORTING
#define TEST_BST_Student

void main()
{
    FILE *fin, *fout;
    STUDENT *pST;
    int record_len;
    char student_record[MAX_STRING_LEN] = { 0 };

    fout = fopen("Sorted_Students.txt", "w");
    if (fout == NULL)
    {
        printf("Error in opening Sorted_Students.txt (write mode)!!\n");
        exit;
    }
    printf("Array of students at initialization : \n");
    printStudents(students, NUM_STUDENTS);
    printf("\n");
}

```



```

/* main.c (2) */

//Sorting students;
selectionSortStudents_by_ST_ID(students, NUM_STUDENTS);
printf("Storing sorted students by increasing order of student ID into
SortedStudent.txt ....\n");
fprintf(fout, students, NUM_STUDENTS);
printStudents(students, NUM_STUDENTS);
fprintf(fout, "\n");
fclose(fout);

fin = fopen("Sorted_Students.txt", "r");
if (fin == NULL)
{
    printf("Error in opening Sorted_Students.txt (read mode)!!\n");
    exit;
}

int cur_pos;
fgets(student_record, MAX_STRING_LEN, fin);
record_len = strlen(student_record);
rewind(fin);
printf("\nRandom access to Sorted_students.txt file (Student record length: %d) ... \n",
record_len);
for (int i = NUM_STUDENTS-1; i >= 0; i--)
{
    fseek(fin, (record_len + 1) * i, SEEK_SET);
    // record_len + 1 to include the CR character at the end of each line
    cur_pos = ftell(fin);
    printf("Current file_position : %3d\n", cur_pos);
    fgets(student_record, MAX_STRING_LEN, fin);
    printf("Student (%2d): %s", i, student_record);
}
fclose(fin);
}

```



```

Array of students at initialization :
Student [ID: 21711000, Kim, G-M , (1990, 10, 5), (tel: +82-053-0805-1234), GPA: 3.57]
Student [ID: 21611075, Yoon, S-M , (1990, 4, 5), (tel: +82-053-0811-1550), GPA: 4.37]
Student [ID: 21411015, Hwang, S-S, (1989, 1, 10), (tel: +82-053-0817-1005), GPA: 2.72]
Student [ID: 21611054, Lee, K-M , (1991, 5, 15), (tel: +82-010-9112-9876), GPA: 3.35]
Student [ID: 21611340, Hong, G-M , (1990, 2, 5), (tel: +82-055-0810-5678), GPA: 3.89]
Student [ID: 21712056, Jang, S-M , (1990, 3, 15), (tel: +82-010-9112-1600), GPA: 4.42]
Student [ID: 21411017, Park, S-S , (1989, 7, 10), (tel: +82-034-0817-1098), GPA: 4.12]
Student [ID: 21511053, Choi, Y-H , (1992, 9, 25), (tel: +82-053-0845-5764), GPA: 3.85]
Student [ID: 21411017, Shin, D-J , (1988, 10, 3), (tel: +82-031-0817-1038), GPA: 3.21]
Student [ID: 21511053, Kwak, S-B , (1994, 11, 15), (tel: +82-002-0897-8778), GPA: 4.45]

Storing sorted students by increasing order of student ID into SortedStudent.txt ....
Student [ID: 21411015, Hwang, S-S, (1989, 1, 10), (tel: +82-053-0817-1005), GPA: 2.72]
Student [ID: 21411017, Park, S-S , (1989, 7, 10), (tel: +82-034-0817-1098), GPA: 4.12]
Student [ID: 21411017, Shin, D-J , (1988, 10, 3), (tel: +82-031-0817-1038), GPA: 3.21]
Student [ID: 21511053, Choi, Y-H , (1992, 9, 25), (tel: +82-053-0845-5764), GPA: 3.85]
Student [ID: 21511053, Kwak, S-B , (1994, 11, 15), (tel: +82-002-0897-8778), GPA: 4.45]
Student [ID: 21611054, Lee, K-M , (1991, 5, 15), (tel: +82-010-9112-9876), GPA: 3.35]
Student [ID: 21611075, Yoon, S-M , (1990, 4, 5), (tel: +82-053-0811-1550), GPA: 4.37]
Student [ID: 21611340, Hong, G-M , (1990, 2, 5), (tel: +82-055-0810-5678), GPA: 3.89]
Student [ID: 21711000, Kim, G-M , (1990, 10, 5), (tel: +82-053-0805-1234), GPA: 3.57]
Student [ID: 21712056, Jang, S-M , (1990, 3, 15), (tel: +82-010-9112-1600), GPA: 4.42]

Random access to Sorted_students.txt file (Student record length: 89) ...
Current file_position : 810
Student ( 9): Student [ID: 21712056, Jang, S-M , (1990, 3, 15), (tel: +82-010-9112-1600), GPA: 4.42]
Current file_position : 720
Student ( 8): Student [ID: 21711000, Kim, G-M , (1990, 10, 5), (tel: +82-053-0805-1234), GPA: 3.57]
Current file_position : 630
Student ( 7): Student [ID: 21611340, Hong, G-M , (1990, 2, 5), (tel: +82-055-0810-5678), GPA: 3.89]
Current file_position : 540
Student ( 6): Student [ID: 21611075, Yoon, S-M , (1990, 4, 5), (tel: +82-053-0811-1550), GPA: 4.37]
Current file_position : 450
Student ( 5): Student [ID: 21611054, Lee, K-M , (1991, 5, 15), (tel: +82-010-9112-9876), GPA: 3.35]
Current file_position : 360
Student ( 4): Student [ID: 21511053, Kwak, S-B , (1994, 11, 15), (tel: +82-002-0897-8778), GPA: 4.45]
Current file_position : 270
Student ( 3): Student [ID: 21511053, Choi, Y-H , (1992, 9, 25), (tel: +82-053-0845-5764), GPA: 3.85]
Current file_position : 180
Student ( 2): Student [ID: 21411017, Shin, D-J , (1988, 10, 3), (tel: +82-031-0817-1038), GPA: 3.21]
Current file_position : 90
Student ( 1): Student [ID: 21411017, Park, S-S , (1989, 7, 10), (tel: +82-034-0817-1098), GPA: 4.12]
Current file_position : 0
Student ( 0): Student [ID: 21411015, Hwang, S-S, (1989, 1, 10), (tel: +82-053-0817-1005), GPA: 2.72]

```



이진 파일 (binary file) 입출력

이진 (Binary) 파일 입출력

◆ 이진 파일 (Binary File) 이란?

- 문자열 정보를 저장하는 텍스트파일 (.txt)와는 달리 영상, 음성 등의 멀티미디어 정보 (예: .jpg, .bmp, .mpg, .wma) 또는 프로그램 실행파일 (예: .exe, .lib 등)
- fscanf(), fgets(), fgetc() 함수로 파일을 읽을 수 없음
- 대신, 이진파일 입출력 함수인 fread(), fwrite() 함수를 사용

◆ 이진 파일 관련 함수

| 유형 | 이진 파일 입출력 함수 | 설명 |
|---------------------|--|--|
| 이진 데이터 파일 입출력 | size_t fread(char *buf, int record_size, int count, FILE *fp) | 이진 데이터 파일로부터 record_size 크기 단위를 count 갯수만큼 읽고, 이를 버퍼에 저장 |
| | size_t fwrite(char *buf, int record_size, int count, FILE *fp) | 버퍼에 저장되어 있는 record_size 크기의 데이터 레코드 count 개수를 이진 파일로 출력 |
| | int feof(FILE *fp) | 파일의 끝에 도달하였는가를 알려줌. 이진 파일 입출력에서는 주로 사용됨. |



dumpBinaryFile()

```
#define BUFFER_LEN DUMP_OCTETS_PER_LINE // 16
```

```
void dumpBinaryFile(FILE *fin, FILE *fout)
```

```
{
    int nbytes;
    unsigned char buffer[BUFFER_LEN] = { '\0' };
    for (int addr = 0; addr < MAX_FILE_SIZE; addr += BUFFER_LEN)
    {
        nbytes = fread(buffer, sizeof(unsigned char), BUFFER_LEN, fin);
        if (nbytes <= 0)
            break;
        fprintf(fout, "%08X: ", addr);
        for (int i = 0; i < nbytes; i++)
        {
            if (i == (BUFFER_LEN / 2))
                fprintf(fout, " ");
            fprintf(fout, "%02X ", buffer[i]);
        }
        fprintf(fout, " ");
        for (int i = 0; i < nbytes; i++)
        {
            if (isprint(buffer[i])) // if printable
                fprintf(fout, "%c", buffer[i]);
            else // if not-printable
                fprintf(fout, ".");
        }
        fprintf(fout, "\n");
    }
}
```

| 주소 | 이진 데이터 16바이트씩을 16진수 형식으로 출력 | 이진 데이터의 printable 코드 출력 |
|-----------|---|----------------------------|
| 00000000: | 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 | MZ..... |
| 00000010: | B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 |@..... |
| 00000020: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00000030: | 00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00 | |
| 00000040: | 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 |!..L.!Th |
| 00000050: | 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F | is program canno |
| 00000060: | 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 | t be run in DOS |
| 00000070: | 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 | mode...\$...... |
| 00000080: | 6B 69 55 C7 2F 08 3B 94 2F 08 3B 94 2F 08 3B 94 | kiU./././././. |
| 00000090: | 69 59 E6 94 2C 08 3B 94 69 59 DB 94 3C 08 3B 94 | iY...;iY...<./. |
| 000000A0: | 69 59 DA 94 28 08 3B 94 F2 F7 F0 94 2D 08 3B 94 | iY...(-...-./. |
| 000000B0: | 2F 08 3A 94 13 08 3B 94 22 5A DA 94 2D 08 3B 94 | /...;."Z...-./. |
| 000000C0: | 22 5A E0 94 2E 08 3B 94 22 5A E5 94 2E 08 3B 94 | "Z...;."Z...;. |
| 000000D0: | 52 69 63 68 2F 08 3B 94 00 00 00 00 00 00 00 00 | Rich/./..... |
| 000000E0: | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 000000F0: | 50 45 00 00 4C 01 07 00 8A A4 96 5A 00 00 00 00 | FE..L.....Z.... |
| 00000100: | 00 00 00 00 E0 00 02 01 0B 01 0C 00 00 40 00 00 |@..... |
| 00000110: | 00 3E 00 00 00 00 00 00 1D 11 01 00 00 10 00 00 | >..... |
| 00000120: | 00 10 00 00 00 00 40 00 00 10 00 00 00 02 00 00 |@..... |
| 00000130: | 06 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 | |
| 00000140: | 00 C0 01 00 00 04 00 00 00 00 00 00 03 00 40 81 |@. |
| 00000150: | 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 | |
| 00000160: | 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 | |
| 00000170: | 74 91 01 00 3C 00 00 00 00 A0 01 00 3C 04 00 00 | t...<.....<... |



```

/* main.c */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include "HandlingBinaryFile.h"

#define MAX_STR_LEN 100
void main()
{
    FILE *fin, *fout;

    const char *fname = "ExampleExeCodeFile.exe";

    if ((fin = fopen(fname, "rb")) == NULL)
    {
        printf("Error - binary input file (%s) cannot be opened !!\n", fname);
        exit;
    }

    if ((fout = fopen("Output.txt", "w")) == NULL)
    {
        printf("Error - Output.txt cannot be created !!\n");
        exit;
    }
    printf("Dumping binary file (%s) ... \n", fname);
    dumpBinaryFile(fin, fout);
    fclose(fin);
    fclose(fout);
}

```



이진 파일 출력 예

| 주소 | 이진 데이터 16바이트씩을 16진수 형식으로 출력 | | | | | | | | | | | | | | | | 이진 데이터의 printable 코드 출력 |
|-----------|--------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------------------------|
| 00000000: | 4D | 5A | 90 | 00 | 03 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | FF | FF | 00 | 00 | MZ..... |
| 00000010: | B8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |@..... |
| 00000020: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000030: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | F0 | 00 | 00 | 00 | |
| 00000040: | 0E | 1F | BA | 0E | 00 | B4 | 09 | CD | 21 | B8 | 01 | 4C | CD | 21 | 54 | 68 |!...L.!Th |
| 00000050: | 69 | 73 | 20 | 70 | 72 | 6F | 67 | 72 | 61 | 6D | 20 | 63 | 61 | 6E | 6E | 6F | is program canno |
| 00000060: | 74 | 20 | 62 | 65 | 20 | 72 | 75 | 6E | 20 | 69 | 6E | 20 | 44 | 4F | 53 | 20 | t be run in DOS |
| 00000070: | 6D | 6F | 64 | 65 | 2E | 0D | 0D | 0A | 24 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | mode....\$..... |
| 00000080: | 6B | 69 | 55 | C7 | 2F | 08 | 3B | 94 | 2F | 08 | 3B | 94 | 2F | 08 | 3B | 94 | kiU./././././. |
| 00000090: | 69 | 59 | E6 | 94 | 2C | 08 | 3B | 94 | 69 | 59 | DB | 94 | 3C | 08 | 3B | 94 | iY.,.,.iY..<.;. |
| 000000A0: | 69 | 59 | DA | 94 | 28 | 08 | 3B | 94 | F2 | F7 | F0 | 94 | 2D | 08 | 3B | 94 | iY..(.;.....-.;. |
| 000000B0: | 2F | 08 | 3A | 94 | 13 | 08 | 3B | 94 | 22 | 5A | DA | 94 | 2D | 08 | 3B | 94 | /.:...;."Z..-.;. |
| 000000C0: | 22 | 5A | E0 | 94 | 2E | 08 | 3B | 94 | 22 | 5A | E5 | 94 | 2E | 08 | 3B | 94 | "Z....;."Z....;. |
| 000000D0: | 52 | 69 | 63 | 68 | 2F | 08 | 3B | 94 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | Rich/./;..... |
| 000000E0: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 000000F0: | 50 | 45 | 00 | 00 | 4C | 01 | 07 | 00 | 8A | A4 | 96 | 5A | 00 | 00 | 00 | 00 | PE..L.....Z.... |
| 00000100: | 00 | 00 | 00 | 00 | E0 | 00 | 02 | 01 | 0B | 01 | 0C | 00 | 00 | 40 | 00 | 00 |@.. |
| 00000110: | 00 | 3E | 00 | 00 | 00 | 00 | 00 | 00 | 1D | 11 | 01 | 00 | 00 | 10 | 00 | 00 | .>..... |
| 00000120: | 00 | 10 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 10 | 00 | 00 | 00 | 02 | 00 | 00 |@..... |
| 00000130: | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000140: | 00 | C0 | 01 | 00 | 00 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 03 | 00 | 40 | 81 |@. |
| 00000150: | 00 | 00 | 10 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 10 | 00 | 00 | |
| 00000160: | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000170: | 74 | 91 | 01 | 00 | 3C | 00 | 00 | 00 | 00 | A0 | 01 | 00 | 3C | 04 | 00 | 00 | t...<.....<... |

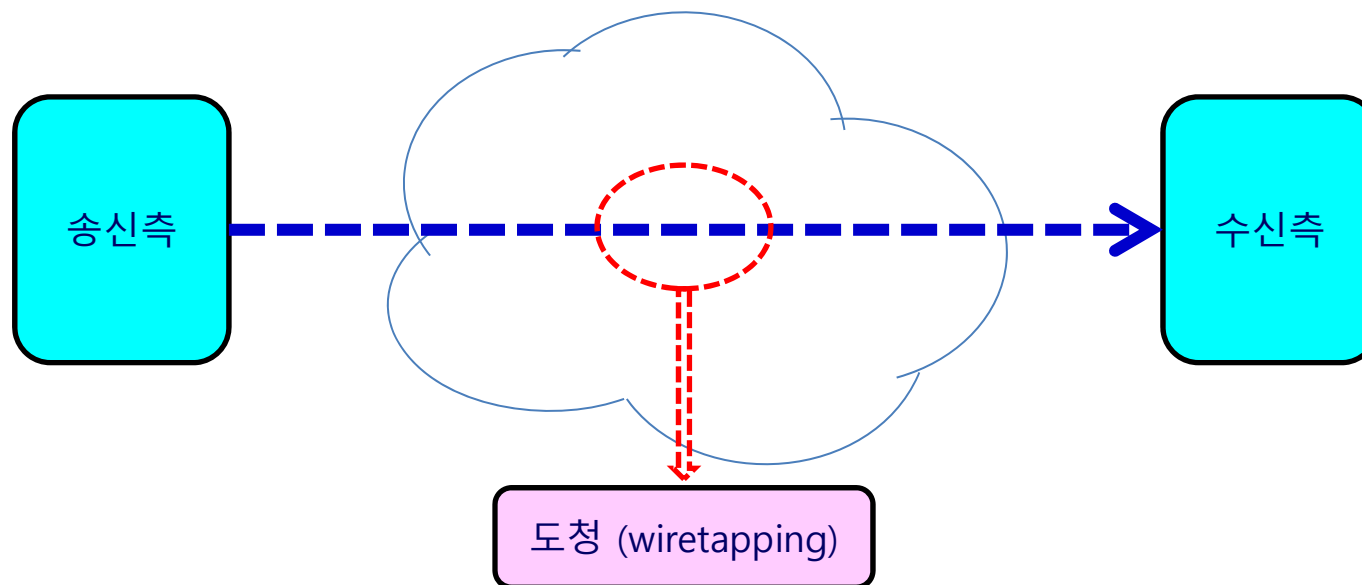


문자열의 암호화

문자열 처리 응용 - 암호화 된 메시지 전송

◆ 송신측, 전송선로, 수신측

- 전송선로상에서 도청이 될 수 있고, 중요한 개인 정보 (신상정보, 은행계좌 및 비밀번호 등)가 유출될 수 있음
- 군사작전, 정부 비밀 정보 등에 대한 높은 보안 등급의 비밀 유지 필요
- 메시지 전송 전에 암호화 시키고, 암호화된 메시지를 전송하며, 메시지 수신 후 암호해독을 함으로써 개인정보 비밀 유지 가능



암호화된 메시지 전송

◆ 송신측

- 전송되는 메시지를 사전에 약속된 암호화 코드로 변환시켜 전송
- 전송 선로상에서 도청이 이루어져도, 암호화 코드를 모르는 경우, 원래의 메시지로 복원할 수 없거나, 복원시키기 위하여 매우 오랜 시간이 걸리게 함

◆ 수신측

- 수신된 데이터를 사전에 약속된 암호화 코드로 역변환시켜 원래의 메시지를 복원

◆ 사전에 약속된 암호화 코드, 암호화 방식

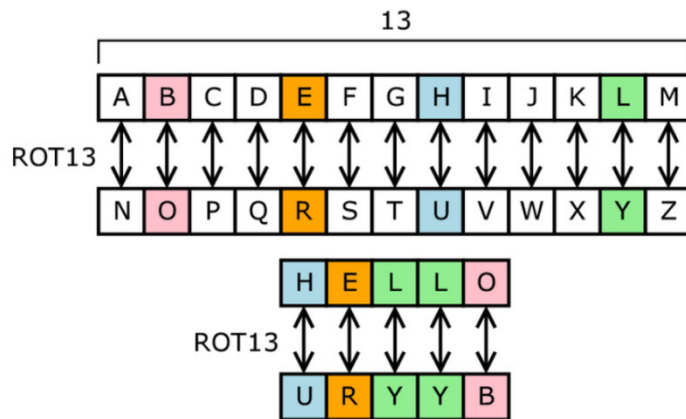
- 인터넷 은행 거래에서의 보안카드 번호 입력
- 인터넷 서버 접속 시 사용자 이름과 비밀번호 입력



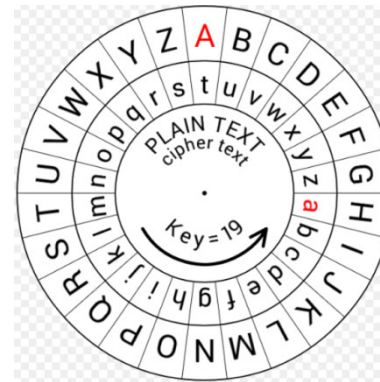
암호화 예 – Caesar Cipher (시저 암호)

◆ 시저 암호(Caesar cipher)

- 시저 암호는 암호학에서 다루는 간단한 치환암호의 일종
- 시저 암호는 약 기원전 100년경에 만들어져 로마의 장군인 시저 (Caesar)가 동맹군들과 소통하기 위해 만든 암호
- 암호화하고자 하는 내용을 알파벳 별로 일정한 거리만큼 밀어서 다른 알파벳으로 치환하는 방식이다. 예를 들어 13글자씩 밀어내는 시저 암호로 "HELLO"를 암호화하면 "URYJB"가 된다.



(a) 13자리씩 밀어 Caesar Cipher 암호화 하는 예



(b) Caesar Cipher 암호 해독 원판



간단한 암호화 개념을 사용하는 송신

◆ 송신 메시지의 문자열을 16진수 문자로 암호화

- ASCII code로 표현된 영문 한 글자 (character)의 8-비트 hexadecimal 값을 찾아 화면으로 16진수로 출력 (예: 0x12, 0xAF)
- 위에서 16진수의 상위 4비트 값과 하위 4비트 값을 각각 2개의 16진수 ASCII 문자 ('0' ~ '9', 'A' ~ 'F')로 출력
예: 'T' (0x54) → 0x5, 0x4 (0 ~ 15의 값을 가짐)
- 0 ~ 15의 값을 가지는 정수를 16진수 ASCII 문자로 출력 ('0' ~ 'F')
- 위 기능을 사용하여, 한 줄의 ASCII 코드 메시지를 각 문자별 16진수의 상위 4비트와 하위 4비트 값을 구하고, 이에 대한 16진수 ASCII 문자로 출력 ('0' ~ 'F')
예: "Test message" →
54657374206D657373616765206F6620
323031373A30333A32350A
- 참고: 0x54 : 'T', 0x65 : 'e', 0x73 : 's', 0x74 : 't'



간단한 암호화 개념을 사용하는 수신 및 복원

◆ 16진수 문자로 암호화된 메시지의 복원

- 16진수 문자로 암호화된 메시지로 부터 2문자 (실제로는 Hexadecimal 숫자) 단위로 읽고, 이에 대한 ASCII 코드를 찾아 변환

예: **54657374206D65737361676520
6F6620323031373A30333A32350A**

→ **Test message**

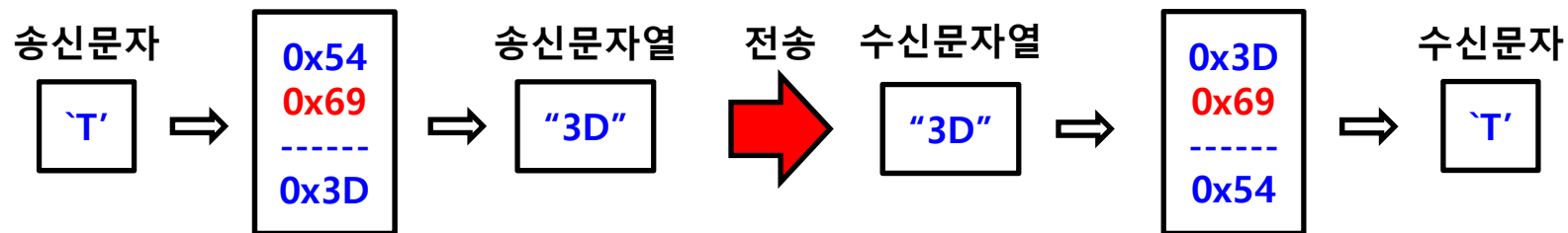
- 줄 바꿈 (0x0A)이 있는 경우, 이를 무시하고, 그 다음 16진수 문자를 2개 읽어 변환하여야 함



암호코드를 사용하는 16진수 문자로의 암호화

◆ 암호코드를 사용하는 방법

- 먼저 ASCII code로 표현된 영문자 (8-bit)를 주어진 **8-비트 암호코드**로 bit-wise exclusive OR를 실행한 후
- 암호화된 영문 한 글자 (character)의 8-비트 hexadecimal 값을 찾고, 이 16진수의 상위 4비트 값과 하위 4비트 값을 각각 2개의 16진수 ASCII 문자 ('0' ~ '9', 'A' ~ 'F')로 출력
- 수신단에서는 16진수 문자로 암호화된 메시지로 부터 2문자 (실제로는 Hexadecimal 숫자) 단위로 읽고, 이에 대한 ASCII 코드를 찾아 변환 ASCII 코드를 찾은 후, 이 ASCII 코드에 사전에 제공된 **8-비트 암호코드**로 bit-wise exclusive OR를 실행하여, 송신된 메시지를 복원



암호코드를 사용하는 16진수 문자로의 암호화

◆ 암호코드를 사용하여 전송하는 예 (**cipher_code** = 0x69)

T: 0101 0100 (0x54)

cc: 0110 1001 (0x69)

xor: 0011 1101 (0x3D)

◆ 암호코드 0x69를 알고 있어야 해독 가능

◆ 만약, 128비트 이상의 암호코드를 사용하여, 암호화를 하는 경우, 이를 암호코드 없이 해독하여야 하는 경우 슈퍼 컴퓨터로도 수 백년 이상 걸리게 됨



cipherText(), deCipherText() 응용 프로그램

```
void test_simple_cipher_text()
{
    FILE *fp_msg, *fp_tx, *fp_rx, *fp_dump_msg, *fout;

    fp_msg = fopen("Message.txt", "r");
    if (fp_msg == NULL)
    {
        printf("Error in file open - Message.txt !!\n");
        exit(-1);
    }

    fp_tx = fopen("Transmit.txt", "w");
    if (fp_tx == NULL)
    {
        printf("Error in file open - Transmit.txt !!\n");
        exit(-1);
    }

    fout = fopen("Output.txt", "w");
    if (fout == NULL)
    {
        printf("Error in file open - Output.txt !!\n");
        exit(-1);
    }
}
```



```

printf("Generating cipher text with cipher-code (%#04x) ..\n", CIPHER_CODE);
cipherText(fp_msg, fp_tx, CIPHER_CODE);
fclose(fp_tx);

rewind(fp_msg);
fprintf(fout, "Binary dump of message.txt file: \n");
dumpBinaryFile(fp_msg, fout);

fp_tx = fopen("Transmit.txt", "r");
if (fp_tx == NULL)
{
    printf("Error in file open - Transmit.txt !!\n");
    exit(-1);
}

fprintf(fout, "\n===== \n");
fprintf(fout, "Binary dump of ciphered document: \n");
dumpBinaryFile(fp_tx, fout);
rewind(fp_msg);

printf("Generating de-ciphered text with cipher-code (%#04x) ..\n", CIPHER_CODE);
fprintf(fout, "\n===== \n");
fprintf(fout, "Generating de-ciphered text with cipher-code (%#04x) ..\n",
    CIPHER_CODE);
deCipherText(fp_tx, fout, CIPHER_CODE);

fclose(fp_msg);
fclose(fp_tx);
fclose(fout);

```

```

}

```



◆ Message.txt

```
Date: 2019. 05. 13.  
Message to be ciphered  
The grand campaign will begin 2018:03:25 06:30AM, at Yeungnam Univ..  
The second line of message.  
The third line of message.
```

◆ Transmit.txt

```
1 2D081D0C53495B5958504749595C4749585A4763  
2 240C1A1A080E0C491D06490B0C490A0019010C1B0C0D63  
3 3D010C490E1B08070D490A08041908000E07491E000505490B0C0E0007495B59585153595A535B5C49595F535A5928244549081D49300C1C070E070804493C07001F474763  
4 3D010C491A0C0A06070D490500070C49060F49040C1A1A080E0C4763  
5 3D010C491D01001B0D490500070C49060F49040C1A1A080E0C47  
6
```

◆ 화면출력

```
Generating cipher text with cipher-code (0x69) ..  
( 1-th Input string, length 20): Date: 2019. 05. 13.  
( 2-th Input string, length 23): Message to be ciphered  
( 3-th Input string, length 69): The grand campaign will begin 2018:03:25 06:30AM, at Yeungnam Univ..  
( 4-th Input string, length 28): The second line of message.  
( 5-th Input string, length 26): The third line of message.  
Generating de-ciphered text with cipher-code (0x69) ..
```

```
Received and deciphered message :  
Date: 2019. 05. 13.  
Message to be ciphered  
The grand campaign will begin 2018:03:25 06:30AM, at Yeungnam Univ..  
The second line of message.  
The third line of message.
```



◆ Output.txt (Cypher-code = 0x00)

```

1  Binary dump of message.txt file:
2  00000000: 44 61 74 65 3A 20 32 30 31 39 2E 20 30 35 2E 20 Date: 2019. 05.
3  00000010: 31 33 2E 0A 4D 65 73 73 61 67 65 20 74 6F 20 62 13..Message to b
4  00000020: 65 20 63 69 70 68 65 72 65 64 0A 54 68 65 20 67 e ciphered.The g
5  00000030: 72 61 6E 64 20 63 61 6D 70 61 69 67 6E 20 77 69 rand campaign wi
6  00000040: 6C 6C 20 62 65 67 69 6E 20 32 30 31 38 3A 30 33 ll begin 2018:03
7  00000050: 3A 32 35 20 30 36 3A 33 30 41 4D 2C 20 61 74 20 :25 06:30AM, at
8  00000060: 59 65 75 6E 67 6E 61 6D 20 55 6E 69 76 2E 2E 0A Yeungnam Univ...
9  00000070: 54 68 65 20 73 65 63 6F 6E 64 20 6C 69 6E 65 20 The second line
10 00000080: 6F 66 20 6D 65 73 73 61 67 65 2E 0A 54 68 65 20 of message..The
11 00000090: 74 68 69 72 64 20 6C 69 6E 65 20 6F 66 20 6D 65 third line of me
12 000000A0: 73 73 61 67 65 2E ssage.
13
14 =====
15 Binary dump of ciphered document:
16 00000000: 34 34 36 31 37 34 36 35 33 41 32 30 33 32 33 30 446174653A203230
17 00000010: 33 31 33 39 32 45 32 30 33 30 33 35 32 45 32 30 31392E2030352E20
18 00000020: 33 31 33 33 32 45 30 41 0A 34 44 36 35 37 33 37 31332E0A.4D65737
19 00000030: 33 36 31 36 37 36 35 32 30 37 34 36 46 32 30 36 361676520746F206
20 00000040: 32 36 35 32 30 36 33 36 39 37 30 36 38 36 35 37 2652063697068657
21 00000050: 32 36 35 36 34 30 41 0A 35 34 36 38 36 35 32 30 265640A.54686520
22 00000060: 36 37 37 32 36 31 36 45 36 34 32 30 36 33 36 31 6772616E64206361
23 00000070: 36 44 37 30 36 31 36 39 36 37 36 45 32 30 37 37 6D706169676E2077
24 00000080: 36 39 36 43 36 43 32 30 36 32 36 35 36 37 36 39 696C6C2062656769
25 00000090: 36 45 32 30 33 32 33 30 33 31 33 38 33 41 33 30 6E20323031383A30
26 000000A0: 33 33 33 41 33 32 33 35 32 30 33 30 33 36 33 41 333A32352030363A
27 000000B0: 33 33 33 30 34 31 34 44 32 43 32 30 36 31 37 34 3330414D2C206174
28 000000C0: 32 30 35 39 36 35 37 35 36 45 36 37 36 45 36 31 205965756E676E61
29 000000D0: 36 44 32 30 35 35 36 45 36 39 37 36 32 45 32 45 6D20556E69762E2E
30 000000E0: 30 41 0A 35 34 36 38 36 35 32 30 37 33 36 35 36 0A.5468652073656
31 000000F0: 33 36 46 36 45 36 34 32 30 36 43 36 39 36 45 36 36F6E64206C696E6
32 00000100: 35 32 30 36 46 36 36 32 30 36 44 36 35 37 33 37 5206F66206D65737
33 00000110: 33 36 31 36 37 36 35 32 45 30 41 0A 35 34 36 38 36167652E0A.5468
34 00000120: 36 35 32 30 37 34 36 38 36 39 37 32 36 34 32 30 6520746869726420
35 00000130: 36 43 36 39 36 45 36 35 32 30 36 46 36 36 32 30 6C696E65206F6620
36 00000140: 36 44 36 35 37 33 37 33 36 31 36 37 36 35 32 45 6D6573736167652E
37 00000150: 0A .
38
39 =====
40 Generating de-ciphered text with cipher-code (0000) ..
41 Date: 2019. 05. 13.
42 Message to be ciphered
43 The grand campaign will begin 2018:03:25 06:30AM, at Yeungnam Univ..
44 The second line of message.
45 The third line of message.

```



◆ Output.txt (Cypher-code = 0x69)

```

1  Binary dump of message.txt file:
2  00000000: 44 61 74 65 3A 20 32 30 31 39 2E 20 30 35 2E 20 Date: 2019. 05.
3  00000010: 31 33 2E 0A 4D 65 73 73 61 67 65 20 74 6F 20 62 13..Message to b
4  00000020: 65 20 63 69 70 68 65 72 65 64 0A 54 68 65 20 67 e ciphered.The g
5  00000030: 72 61 6E 64 20 63 61 6D 70 61 69 67 6E 20 77 69 rand campaign wi
6  00000040: 6C 6C 20 62 65 67 69 6E 20 32 30 31 38 3A 30 33 ll begin 2018:03
7  00000050: 3A 32 35 20 30 36 3A 33 30 41 4D 2C 20 61 74 20 :25 06:30AM, at
8  00000060: 59 65 75 6E 67 6E 61 6D 20 55 6E 69 76 2E 2E 0A Yeungnam Univ...
9  00000070: 54 68 65 20 73 65 63 6F 6E 64 20 6C 69 6E 65 20 The second line
10 00000080: 6F 66 20 6D 65 73 73 61 67 65 2E 0A 54 68 65 20 of message..The
11 00000090: 74 68 69 72 64 20 6C 69 6E 65 20 6F 66 20 6D 65 third line of me
12 000000A0: 73 73 61 67 65 2E ssage.
13
14 =====
15 Binary dump of ciphered document:
16 00000000: 32 44 30 38 31 44 30 43 35 33 34 39 35 42 35 39 2D081D0C53495B59
17 00000010: 35 38 35 30 34 37 34 39 35 39 35 43 34 37 34 39 58504749595C4749
18 00000020: 35 38 35 41 34 37 36 33 0A 32 34 30 43 31 41 31 585A4763.240C1A1
19 00000030: 41 30 38 30 45 30 43 34 39 31 44 30 36 34 39 30 A080E0C491D06490
20 00000040: 42 30 43 34 39 30 41 30 30 31 39 30 31 30 43 31 B0C490A0019010C1
21 00000050: 42 30 43 30 44 36 33 0A 33 44 30 31 30 43 34 39 B0C0D63.3D010C49
22 00000060: 30 45 31 42 30 38 30 37 30 44 34 39 30 41 30 38 0E1B08070D490A08
23 00000070: 30 34 31 39 30 38 30 30 30 45 30 37 34 39 31 45 041908000E07491E
24 00000080: 30 30 30 35 30 35 34 39 30 42 30 43 30 45 30 30 000505490B0C0E00
25 00000090: 30 37 34 39 35 42 35 39 35 38 35 31 35 33 35 39 07495B5958515359
26 000000A0: 35 41 35 33 35 42 35 43 34 39 35 39 35 46 35 33 5A535B5C49595F53
27 000000B0: 35 41 35 39 32 38 32 34 34 35 34 39 30 38 31 44 5A5928244549081D
28 000000C0: 34 39 33 30 30 43 31 43 30 37 30 45 30 37 30 38 49300C1C070E0708
29 000000D0: 30 34 34 39 33 43 30 37 30 30 31 46 34 37 34 37 04493C07001F4747
30 000000E0: 36 33 0A 33 44 30 31 30 43 34 39 31 41 30 43 30 63.3D010C491A0C0
31 000000F0: 41 30 36 30 37 30 44 34 39 30 35 30 30 30 37 30 A06070D490500070
32 00000100: 43 34 39 30 36 30 46 34 39 30 34 30 43 31 41 31 C49060F49040C1A1
33 00000110: 41 30 38 30 45 30 43 34 37 36 33 0A 33 44 30 31 A080E0C4763.3D01
34 00000120: 30 43 34 39 31 44 30 31 30 30 31 42 30 44 34 39 0C491D01001B0D49
35 00000130: 30 35 30 30 30 37 30 43 34 39 30 36 30 46 34 39 0500070C49060F49
36 00000140: 30 34 30 43 31 41 31 41 30 38 30 45 30 43 34 37 040C1A1A080E0C47
37 00000150: 0A .
38
39 =====
40 Generating de-ciphered text with cipher-code (0x69) ..
41 Date: 2019. 05. 13.
42 Message to be ciphered
43 The grand campaign will begin 2018:03:25 06:30AM, at Yeungnam Univ..
44 The second line of message.
45 The third line of message.

```



Homework 10

Homework 10

10.1 영문자를 처리하는 프로그램

- 1) 10 개의 ASCII 단어를 입력 파일 "words_input.txt"로 부터 차례대로 입력받아 문자 배열 char word[]에 저장하고, 이를 단어 배열 char words[][]에 차례로 저장하라. 각 단어는 최대 15 자 이내이며, 입력 데이터 파일의 내용은 다음과 같다.

one two three four five

six seven eight nine ten

- 2) `strlen(char *)` 함수를 이용하여 각 단어의 길이를 찾고, 이를 정수형 배열 `int word_len[]`에 저장하라.
- 3) 입력 단어들을 해당 단어 길이와 함께, 한 줄에 한 단어씩, 출력 파일 "words_output.txt"에 출력하라.
- 4) 입력된 단어들이 포함된 단어 배열 `char words[][]` 을 오름차순으로 정렬하여 "words_output.txt"에 출력하라.

```
Input word list:
( 0)th-word : one      (word_length: 3)
( 1)th-word : two      (word_length: 3)
( 2)th-word : three    (word_length: 5)
( 3)th-word : four     (word_length: 4)
( 4)th-word : five     (word_length: 4)
( 5)th-word : six      (word_length: 3)
( 6)th-word : seven    (word_length: 5)
( 7)th-word : eight    (word_length: 5)
( 8)th-word : nine     (word_length: 4)
( 9)th-word : ten      (word_length: 3)
Sorted word list:
eight   five    four    nine    one
seven   six     ten     three   two
```



10.2 ASCII 문자를 입력받아 16진수로 변환하기

- 1) 0 ~ 9, 'A' ~ 'F' 사이에 있는 16진수 ASCII 코드 문자열 (최대길이, MAX_HEX_STR_LEN: 8)을 매개변수로 전달 받은 후, 이를 16진수로 변환하여 정수 (integer)형으로 반환하는 함수 `int atoi(char *hxdStr)` 를 작성하라.
- 2) 하나의 정수 값 `hxd`와 `char pointer strBuf`를 인수로 전달받은 후, 전달된 정수 값을 16진수의 ASCII 코드로 변환하여, `strBuf`가 지정하는 곳에 저장하는 함수 `void xtoa(char *strBuf, int hxd)`를 작성하라.
- 3) 표준입력장치로부터 0 ~ 9, 'A' ~ 'F' 사이에 있는 ASCII 코드 문자로 구성된 16진수 데이터를 입력받은 후, 위의 `atoi()` 함수를 사용하여 정수값으로 변환하고, `printf("%d (decimal) %X (hexadecimal)", hxd, hxd)` 형식을 사용하여 변환된 정수값을 10진수와 16진수로 출력하라.
- 4) 이 변환된 정수값을 위에서 구현한 `xtoa()` 함수를 사용하여 16진수 ASCII 문자열로 변환한 후, 문자열로 출력하는 프로그램을 작성하고, 정확하게 동작하는 것을 확인하라.
- 5) 실행 결과 (예)

```
input hexadecimal number : 1000
hexadecimal (1000) = 4096 (decimal)
hexadecimal (1000) = 4096 (decimal) = 0X1000 (hexadecimal) = re-converted hexadecimal string (1000)
input hexadecimal number : FFFF
hexadecimal (FFFF) = 65535 (decimal)
hexadecimal (FFFF) = 65535 (decimal) = 0xFFFF (hexadecimal) = re-converted hexadecimal string (FFFF)
input hexadecimal number : FFFFFFFF
hexadecimal (FFFFFFFF) = -1 (decimal)
hexadecimal (FFFFFFFF) = -1 (decimal) = 0xFFFFFFFF (hexadecimal) = re-converted hexadecimal string (FFFFFFFF)
input hexadecimal number : 1234
hexadecimal (1234) = 4660 (decimal)
hexadecimal (1234) = 4660 (decimal) = 0X1234 (hexadecimal) = re-converted hexadecimal string (1234)
input hexadecimal number : 1234ABCD
hexadecimal (1234ABCD) = 305441741 (decimal)
hexadecimal (1234ABCD) = 305441741 (decimal) = 0X1234ABCD (hexadecimal) = re-converted hexadecimal string (1234ABCD)
input hexadecimal number : ABCD1234
hexadecimal (ABCD1234) = -1412623820 (decimal)
hexadecimal (ABCD1234) = -1412623820 (decimal) = 0XABCD1234 (hexadecimal) = re-converted hexadecimal string (ABCD1234)
input hexadecimal number : .
```



6) main() 함수 예

```
/* main_atox_xtoa.cpp */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "hexadecimal.h"

void main()
{
    char *hexStr;
    char *hexConvertStr;
    unsigned int value_int;
    while (1)
    {
        hexStr = (char*)calloc(MAX_HEX_STR_LEN+1, sizeof(char));
        hexConvertStr = (char*)calloc(MAX_HEX_STR_LEN+1, sizeof(char));
        printf("input hexadecimal number : ");
        scanf("%s", hexStr);
        if (strcmp(hexStr, ".") == 0)
            break;
        value_int = atox(hexStr);
        printf("hexadecimal (%s) = %d (decimal)\n", hexStr, value_int);
        xtoa(value_int, hexConvertStr);
        printf("hexadecimal (%s) = %d (decimal) = %#0X (hexadecimal) =  

            re-converted hexadecimal string (%s)\n",
            hexStr, value_int, value_int, hexConvertStr);
    }
}
```

