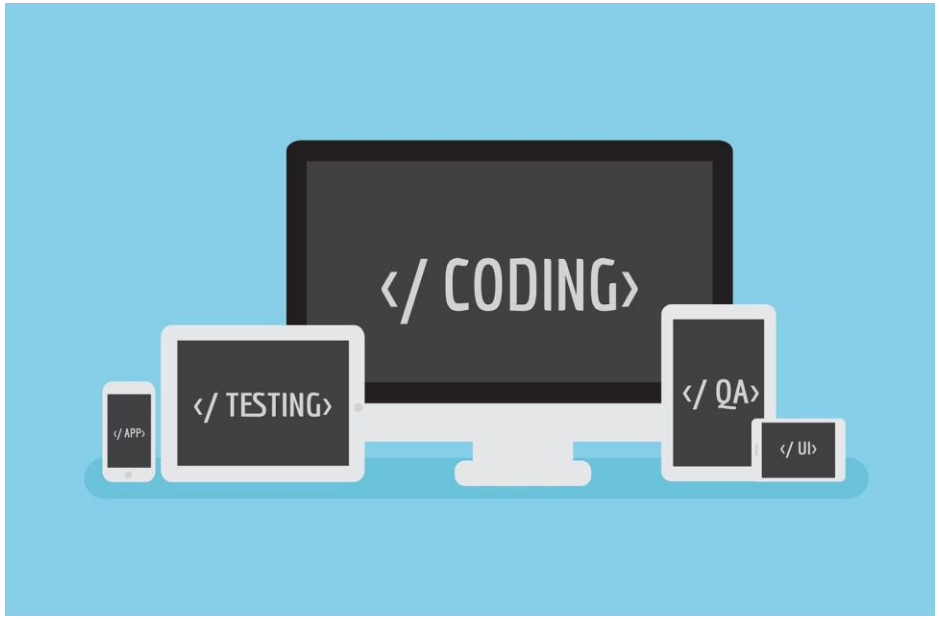
고급응용 C프로그래밍

11장 – String



서강대학교
SOGANG UNIVERSITY

String

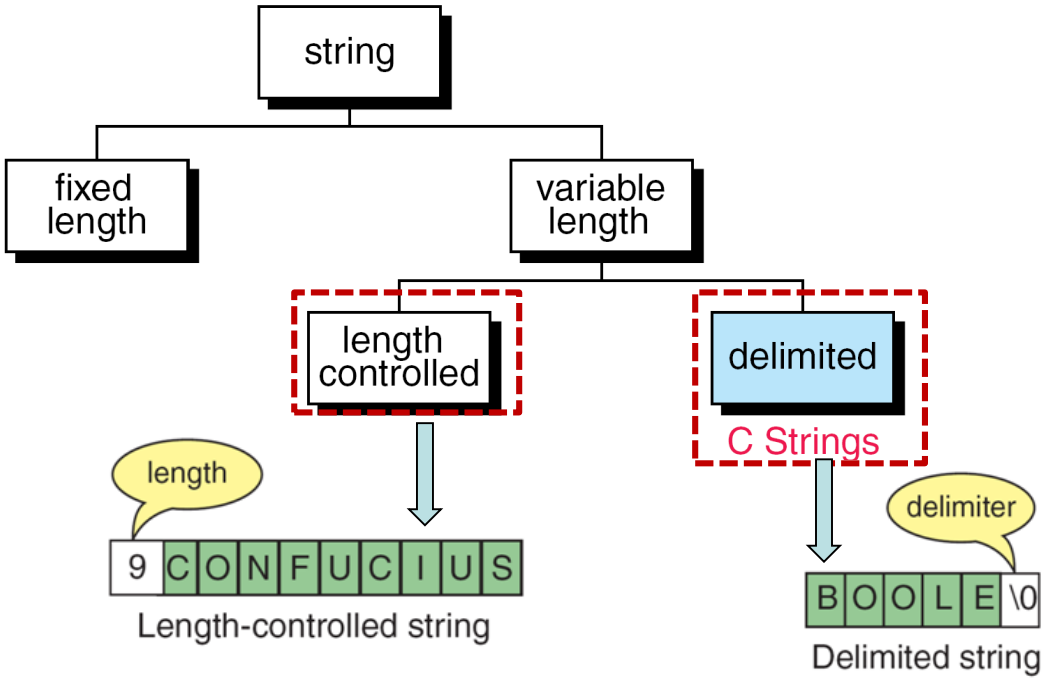
- 문자열의 개념
- **C** 문자열
- 문자열 입출력 함수
- 문자열의 배열
- 문자열 조작 함수

문자열(String)의 개념

◈ 문자열**(string)**은 문자**(character)**의 연속이다

◈ 문자열은 길이가 가변적이기 때문에 구현 방법도 다양하다

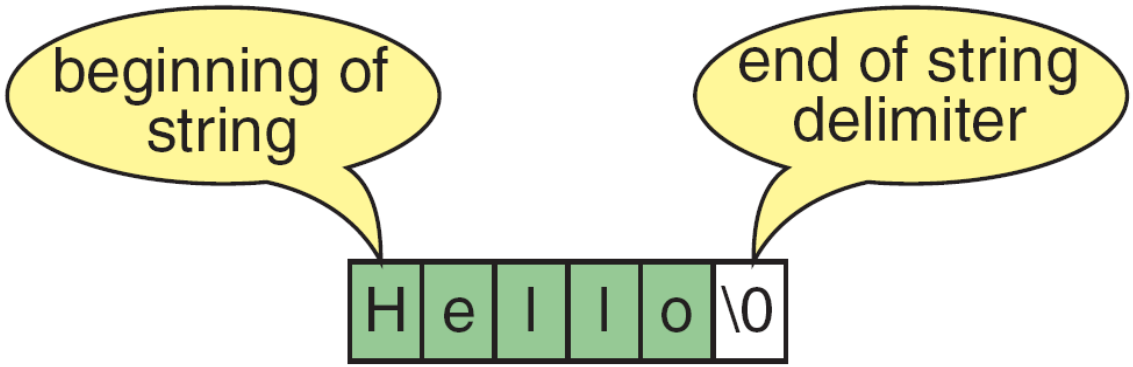
- 실제로 언어마다 그 구현 방법이 다르다



C 문자열

◈ C에서의 문자열

- C에서 문자열은 구성되는 문자들의 배열의 형태로 메모리에 저장
- 가변길이이며, null character('₩0')으로 끝난다

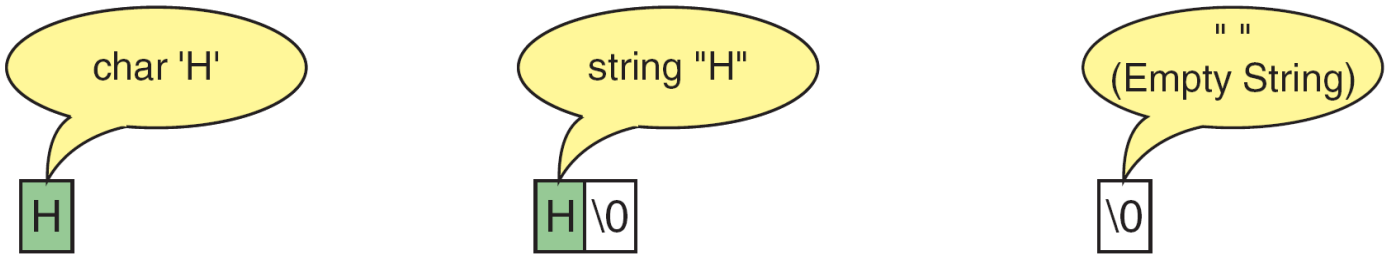


C 문자열

◈ 문자열과 문자의 저장

질문) '₩0' 와 "" 의 차이는 ?

- 문자열과 문자의 차이
  - "H" 문자열은 'H'와 함께 문자열의 끝을 표시하는 null character를 저장하여 2byte에 저장된다
  - null character는 비어있는 문자열임을 표시한다 (1byte에 저장된다)



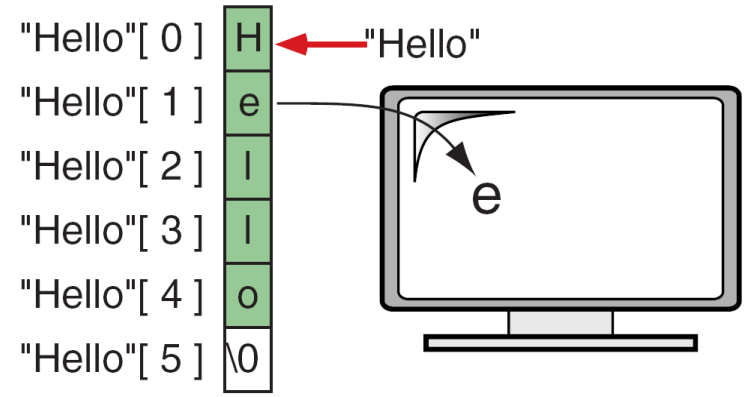
  - 문자열은 character 배열과는 달리 null이라는 delimiter를 갖는다

C 문자열

◈ 문자열 상수는 포인터이다

- 문자열 상수가 사용되면, C는 자동적으로 character 배열을 생성해서 이를 저장하고, 그 배열의 시작 주소를 저장한다
- 문자열 상수 자체를 포인터처럼 사용하여 문자열 상수의 한 character에 접근할 수 있다
  - 즉, 문자열 상수를 배열의 이름으로 활용 가능하다

```
#include <stdio.h>
int main (void)
{
  printf("%c\n", "Hello"[1];
  return 0;
} // main
```

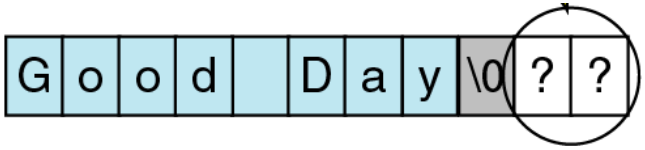


C 문자열

◈ 문자열 초기화의 4가지 방법

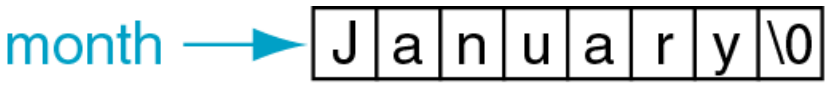
- 고정된 길이의 배열을 선언

```
char str[11] = "Good Day";
```



- 배열 크기를 비워 둠
  - 문자열의 길이에 딱 맞는 배열이 생성된다

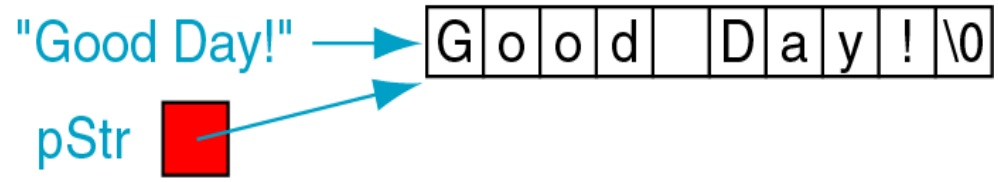
```
char month[] = "January";
```



C 문자열

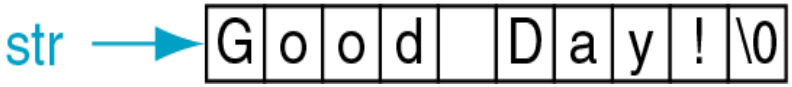
- 포인터를 이용
  - 문자열 상수가 임의의 배열에 저장되고, 이 배열의 시작 주소를 포인터 변수가 갖는다

```
char *pStr = "Good Day!";
```



- 배열 선언 후 한 문자씩 입력하는 방법

```
char str[10] =
     {'G','o','o','d',' ','D','a','y','!','\0'};
```



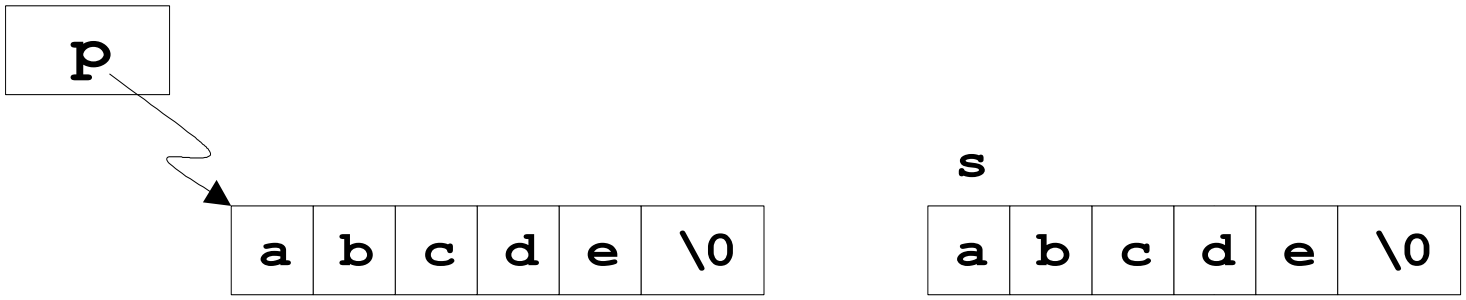
C 문자열

◈ 문자열을 초기화 할 때 배열 사용과 포인터 사용의 차이

```
char  *p = "abcde";     // const char *p = “abcde”와 차이점은?
char  s[ ] = "abcde";   /* char  s[ ] = {'a', 'b', 'c', 'd', 'e', '₩0'}; 와 동일*/
```

포인터 변수 p는 하나의 변수로서 별도의 저장공간을 갖지만
배열 s는 단지 배열의 시작주소로서 상수이다

- 배열의 이름은 포인터처럼 활용할 수 있지만, 다른 주소 값을 대입할 수 없다



C 문자열

◈ 배열 단위의 복사를 허용하지 않는다

- 컴파일 에러가 발생한다
- 다음은 str2를 초기화하기 위해 str2에 문자열 str1을 저장하고자 했다. 하지만 문자열간에는 대입연산자를 사용할 수 없으므로 아래 예제는 세 번째 줄에서 compile error가 발생한다

```
char str1[11] = "Hello";
char str2[11];
str2 = str1;                    /* Compile error */
```

C 문자열

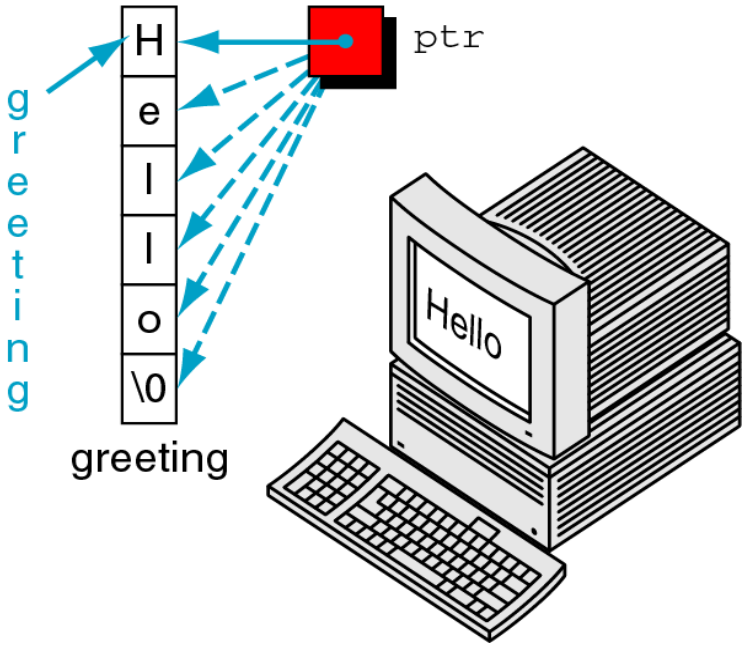
◈ 문자열과 포인터

- 문자열은 배열에 저장되고 배열의 이름은 포인터이므로, 이를 이용해서 다음과 같이 문자열을 출력하는 것이 가능하다

```
#include  <stdio.h>

int main() {
    char greeting[] = "Hello";
    char * ptr;

    ptr = greeting;
    while (*ptr!='\0')
    {
        printf("%c", *ptr);
        ptr++;
    } /* while */

    printf("\n");

    return 0;
} /* Print String */
```

null을 만날 때까지 반복

한 문자씩 출력



문자열 입출력 함수

◈ **scanf**와 **printf** 함수를 이용하여 문자열의 입출력이 가능하다

◈ **scanf()** 를 이용한 문자열 입력

- 문자열 변환 명세(%s)

```
scanf("%s", month);
```

month가 입력 받을 데이터를 저장하기에 충분한 크기인지를 고려할 필요가 있다

- 길이 9의 문자열을 입력 받아 배열에 저장

```
char month[10];
scanf("%9s", month);
```

null까지 저장을 해야 하므로 길이 10인 배열을 만든다

9글자 이상의 문자를 입력 받더라도 최대 9글자까지만을 month에 저장

문자열 입출력 함수

- 스캔 세트 변환 코드([…])
  - [] 사이에 있는 문자들만 문자열에 포함될 수 있다
  - ^(탈자 기호)를 이용하면 포함하지 않을 문자를 명시할 수도 있다
  - %s와 달리 입력에서 공백 문자를 제거하지 않는다

  - 숫자와 콤마, 마침표, 음수 부호, 달러만 사용하는 최대 길이 10인 문자열 읽기

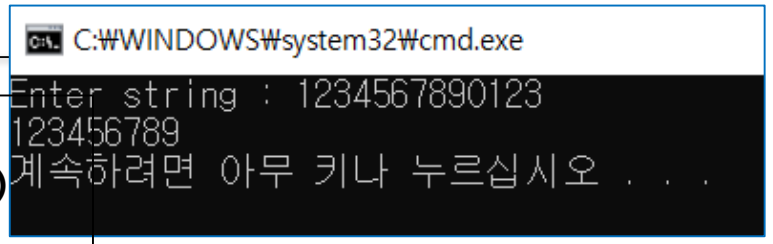
```
scanf ("%10[0123456789.,-$]", str);
```

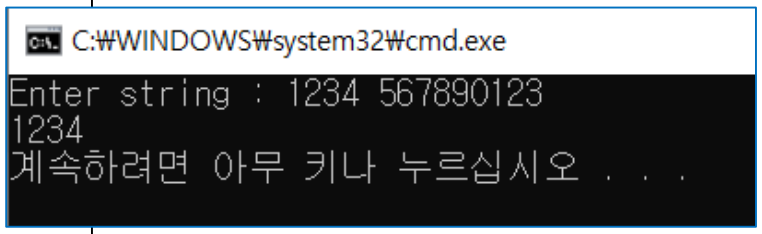
  - 특수 문자들을 제외한 길이 15의 문자열 읽기

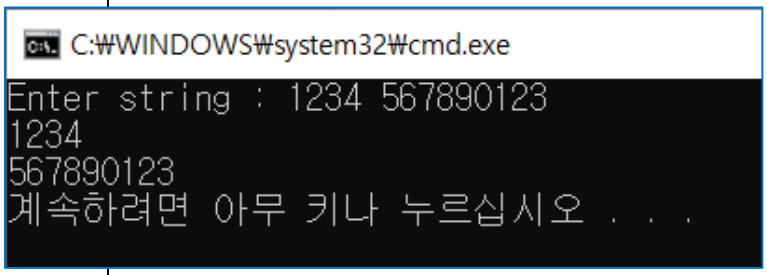
```
scanf ("%15[^~!@#$%^&*()_+]", str);
```

문자열 입출력 함수

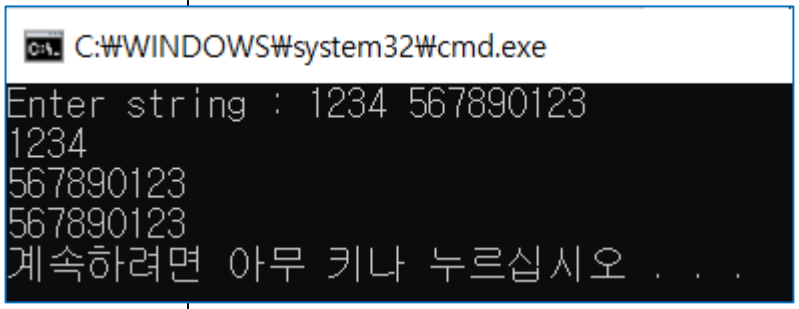
```
#include <stdio.h>
//#define FLUSH while (getchar() != '\n')
int main(void) {
    char  str[10];
    char* ptr ;

    printf("Enter string : ");
    scanf("%9s", str);

    ptr = str;

    while (*ptr != '\0') {
        printf("%c", *ptr);
        ptr++;
    }
    printf("\n");

    //FLUSH;
    //scanf("%s", str);
    //printf("%s\n", str);
    return 0;
}
```







FLUSH없이 주석 제거 후 실행



FLUSH포함 주석 제거 후 실행

문자열 입출력 함수

◈ printf() 를 이용한 문자열의 출력

- scanf에서와 같이 %s 를 이용하여 문자열을 출력한다
- 30칸에 맞춰서 출력. 오른쪽 정렬

```
    printf("|%30s|\n", "This is the string");
Output:
          |          This is the string|
```

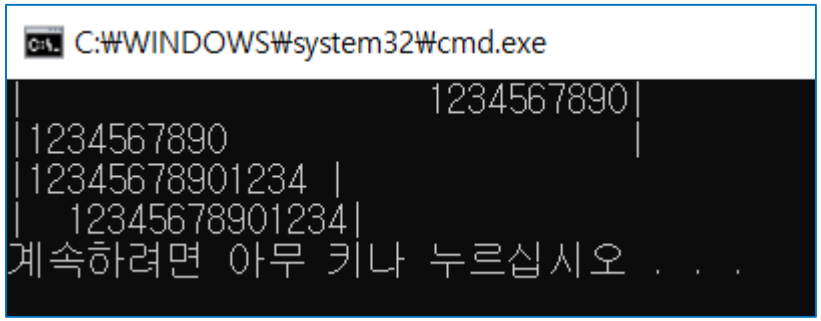
- 30칸에 맞춰서 출력. 왼쪽 정렬

```
    printf("|%-30s|\n", "This is the string");
Output:
          |This is the string          |
```

- 15칸에 맞춰서 출력. 14글자만 출력. 왼쪽 정렬

```
    printf("|%-15.14s|", "123456789012345678890");
Output:
          |12345678901234 |
```

문자열 입출력 함수



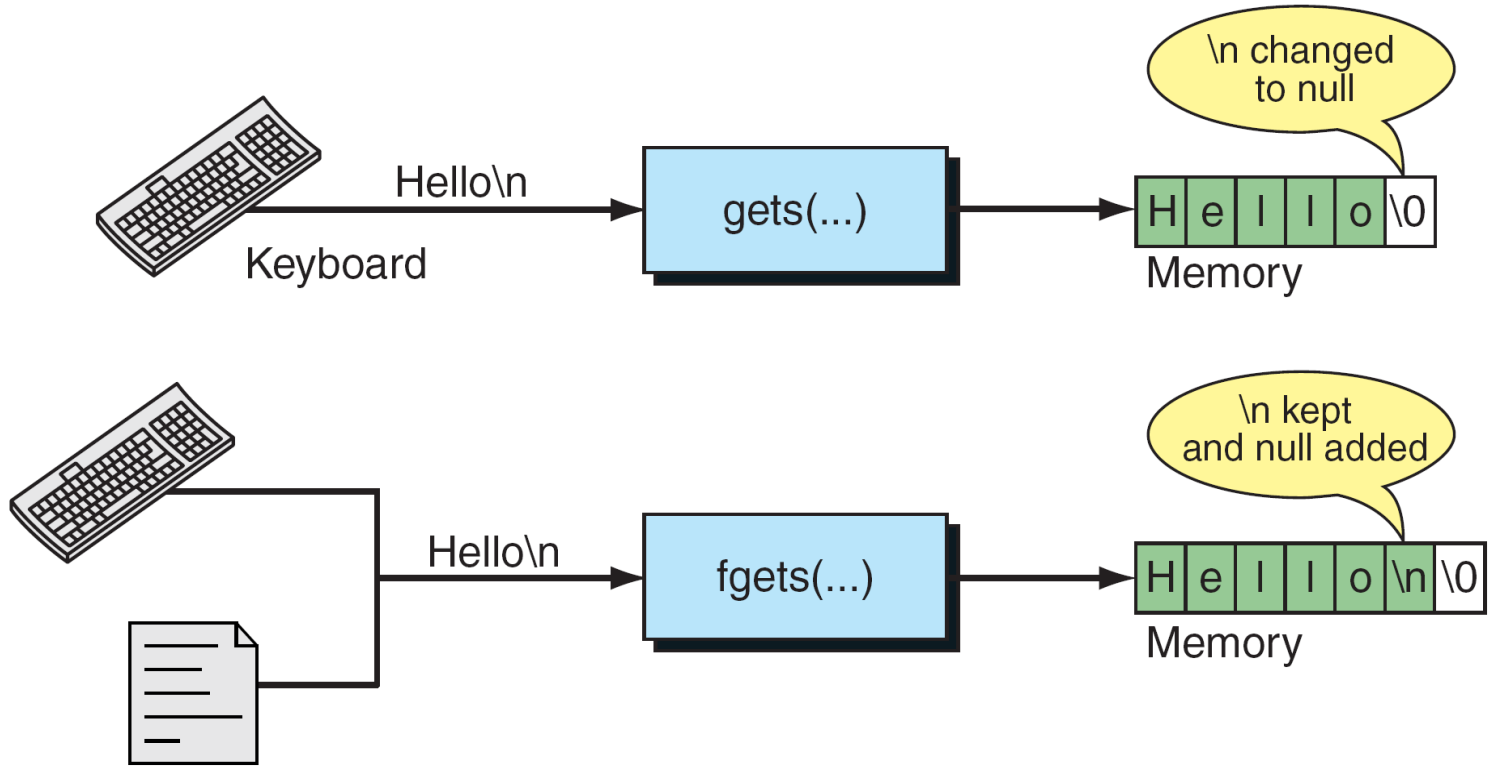

```
#include <stdio.h>
int main(void)
{
    printf("|%30s|\n", "1234567890");
    printf("|%-30s|\n", "1234567890");
    printf("|%-15.14s|\n", "12345678901234567890");
    printf("|%16.14s|\n", "12345678901234567890");
    return 0;
}
```

실습 #1

문자열 입출력 함수

◈ gets / fgets

- format에 따라 입력을 받는 scanf, fscanf 등과 달리 formatting 없이 한 line을 읽어 들이는 함수



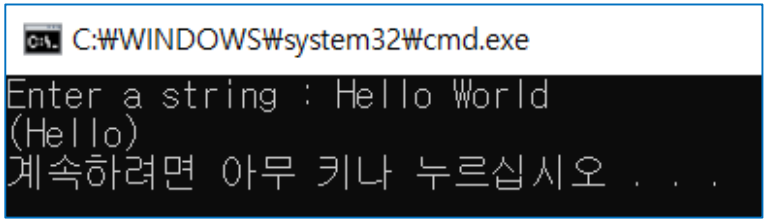
문자열 입출력 함수

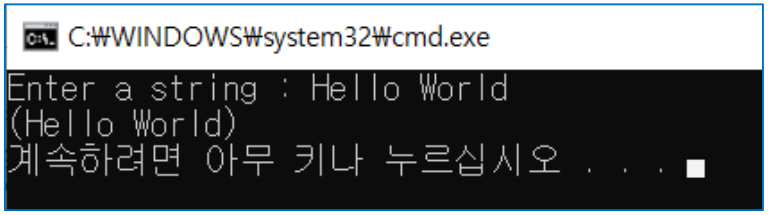
◈ scanf()와 gets() 비교

- scanf( ) 함수는 데이터를 읽을 때 탭(tab),공백(blank),개행 (newline) 문자 등에 의해 데이터가 구분
- gets( ) 함수는 개행문자 ('₩n') 앞에 있는 모든 문자 (공백, 탭(tab) 포함) 를 읽어 들임

```c
#include <stdio.h>
int main(void) {
    char str[80];

    printf("Enter a string : ");
    scanf("%s", str);
    printf("(%s)\n", str);
    return 0;
}
```

```c
#include <stdio.h>
int main(void) {
    char str[80];

    printf("Enter a string : ");
    gets(str);
    printf("(%s)\n", str);
    return 0;
}
```





문자열 입출력 함수

◈ gets / fgets 함수

- char* gets (char* strPtr);
  - 키보드로부터 한 line을 입력 받아 strPtr에 저장
  - 한 line은 '₩n'을 입력 받을 때까지를 의미, '₩n'은 '₩0'으로 치환되어 저장된다
  - 한 line의 길이가 strPtr의 길이보다 길면 strPtr 뒷부분의 메모리가 침범되어 Segmentation fault가 발생하므로 주의해야 한다

성공 : strPtr의 주소를 return
실패 : NULL

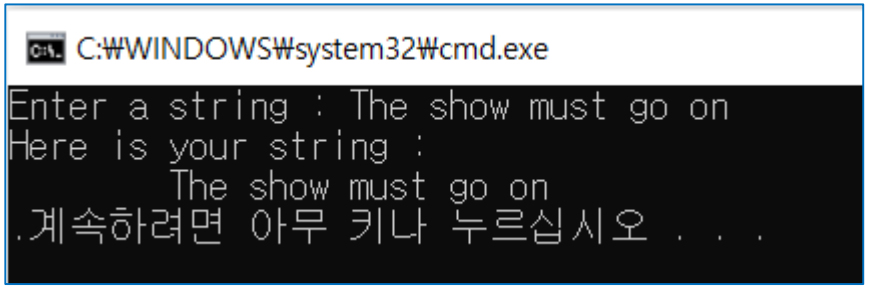
- char* fgets (char* strPtr, int size, FILE *fp);
  - file 포인터 fp로부터 한 line을 읽어 들여 strPtr에 저장
    - 읽어 들이는 문자의 최대 개수는 size-1개이다
  - 한 line은 '₩n'을 입력 받을 때까지를 의미, '₩n'까지 입력 받고 뒤에 '₩0'이 추가로 저장됨
  - File 포인터에 stdin을 주면 키보드로부터 입력을 받을 수 있다

성공 : strPtr의 주소를 return
실패, 파일의 끝(EOF) : NULL

일반적으로 gets(strPtr)보다 fgets(strPtr, sizeof(strPtr), stdin); 의 사용이 권장된다

문자열 입출력 함수

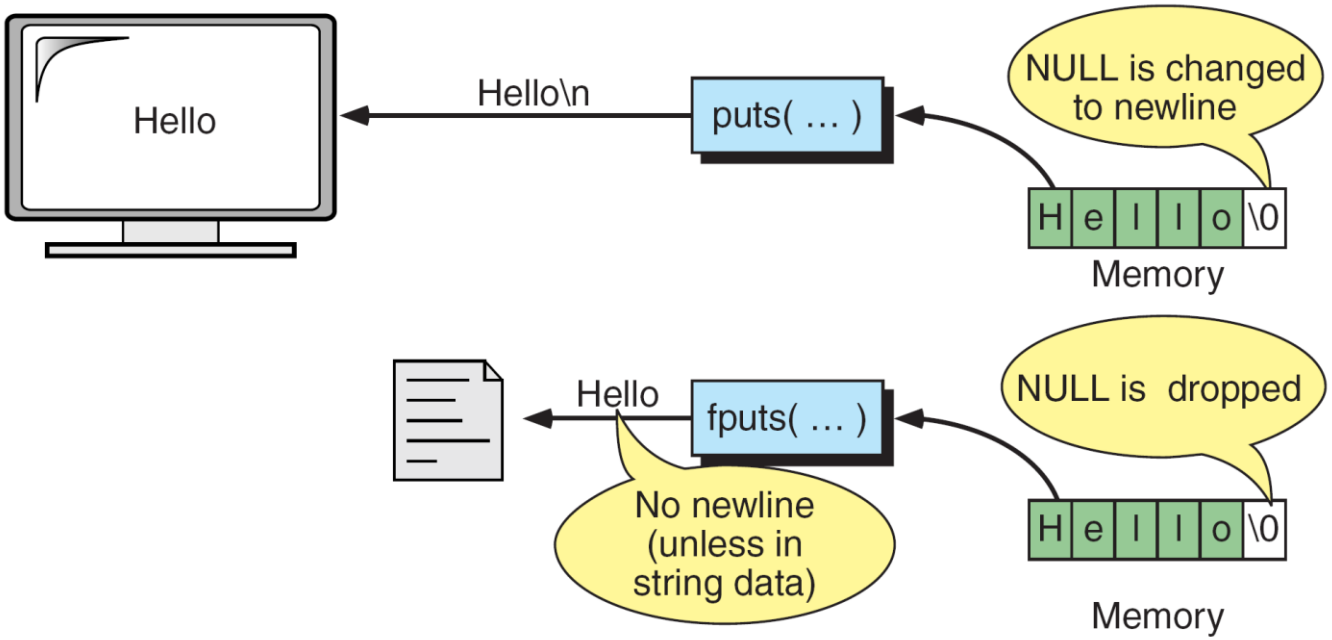
```
#include <stdio.h>
int main(void)
{
    char str[81];

    printf("Enter a string : ");
    fgets(str, sizeof(str), stdin);
    printf("Here is your string : \n\t%s.", str);
    return 0;
}
```



문자열 입출력 함수

◈ puts / fputs

- format에 따라 출력을 하는 printf, fprintf 등과 달리 formatting 없이 한 line을 출력하는 함수



문자열 입출력 함수

◈ puts / fputs 함수

- int puts (const char *strPtr);
  - strPtr의 문자열을 모니터에 한 line으로 출력
  - '₩0'이 '₩n'으로 치환되므로 문자열 끝에 '₩n'이 있다면 줄 바꿈이 두 번 일어난다
  - '₩0'까지만 출력하므로 문자열 중간에 '₩0'이 있으면 뒷부분은 무시된다

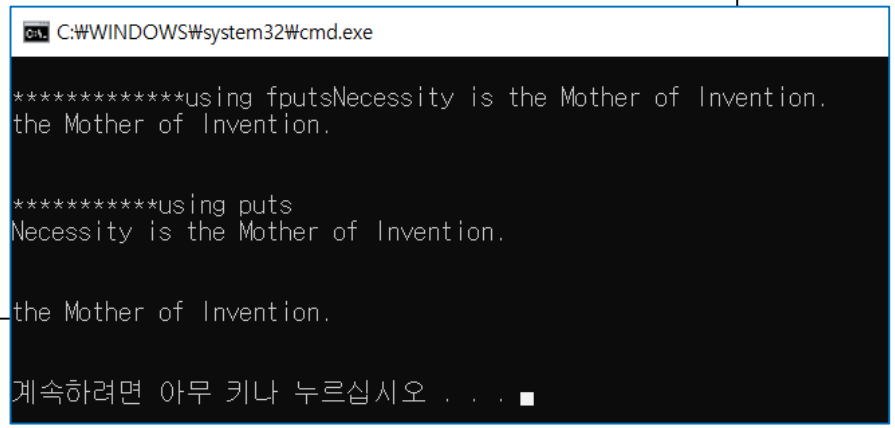
성공 : 음수가 아닌 정수
실패 : EOF(-1)

- int fputs (const char *strPtr, FILE *fp);
  - strPtr의 문자열을 file 포인터 fp에 기록
  - strPtr은 '₩0'으로 끝나야 하며 NULL문자는 출력되지 않는다
  - '₩0'까지만 출력하므로 문자열 중간에 '₩0'이 있으면 뒷부분은 무시된다
  - File 포인터 fp에 stdout을 주면 모니터에 출력할 수 있다

성공 : 음수가 아닌 정수
실패 : EOF(-1)

문자열 입출력 함수

```
#include  <stdio.h>
int main(void)
{
    char str[] = "Necessity is the Mother of Invention.";
    char* pstr = str;

    fputs("\n************using fputs", stdout);
    fputs(pstr, stdout);
    fputs("\n", stdout);

    fputs(pstr + 13, stdout);
    fputs("\n", stdout);

    puts("\n\n**********using puts");
    puts(pstr);
    puts("\n");
    puts(pstr + 13);
    puts("\n");
    return 0;
}
```



문자열 입출력 함수

◈ **fgets / fputs** 사용 예제프로그램

```
#include <stdio.h>
#include <ctype.h>
int main(void) {
    char str[81];

    while (fgets(str, sizeof(str), stdin))
    {
        if (isupper(*str))
            fputs(str, stdout);
    }
    return 0;
}
```

키보드로부터 문자열의 입력 받는다

입력 받은 문자열이 대문자로 시작할 경우에만 출력한다.

EOF(Ctrl + z)를 입력 받을 때까지 반복

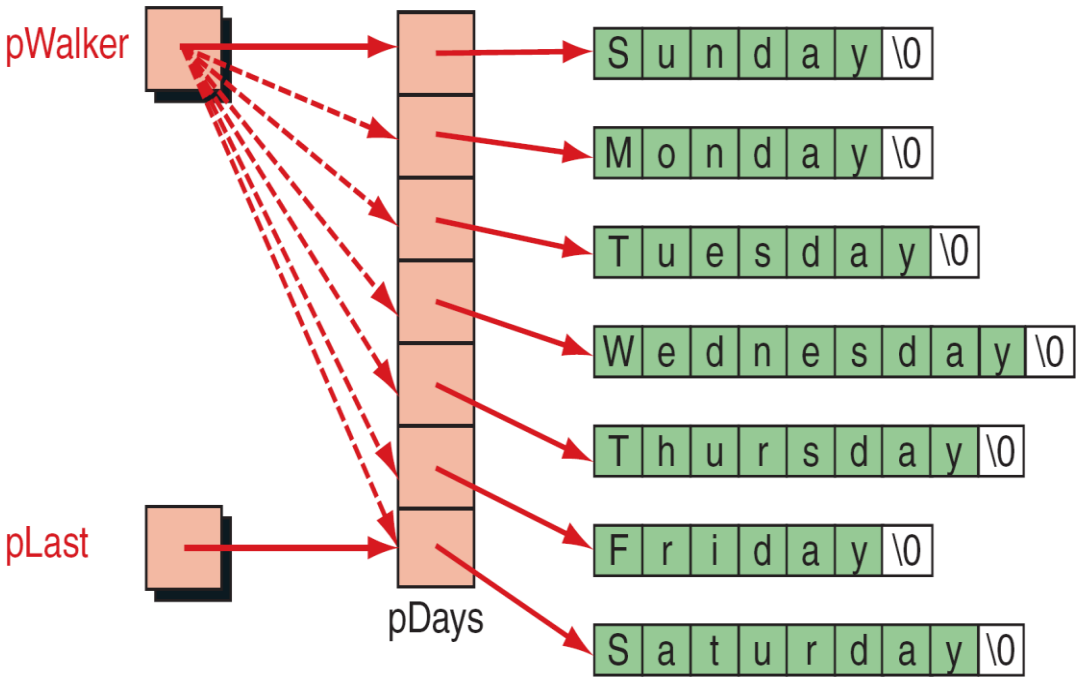
```
C:₩WINDOWS₩system32₩cmd.exe
The show must go on
The show must go on
C programming language
C programming language
is widely used by
system engineer
^Z
계속하려면 아무 키나 누르십시오 . . .
```

실습 #2

문자열의 배열

◈ 문자열을 원소로 갖는 배열을 만들어 사용할 수 있다

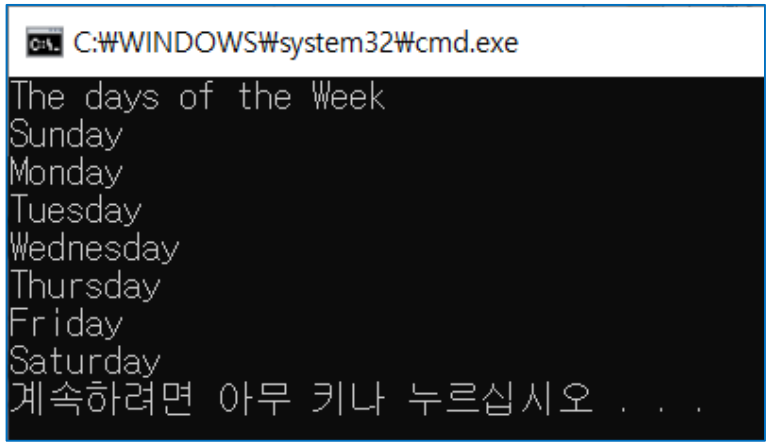
- char *타입의 배열을 만들면 각각의 원소(포인터)가 문자열을 포인팅하도록 할 수 있다
- 예제 프로그램 – 문자열의 배열을 이용하여 일주일의 요일을 출력



문자열의 배열

```
#include <stdio.h>
#include <string.h>
int main(void) {
     char *pDays[7];
     char **pLast,  **pWalker;

     pDays[0] = "Sunday";
     pDays[1] = "Monday";
     pDays[2] = "Tuesday";
     pDays[3] = "Wednesday";
     pDays[4] = "Thursday";
     pDays[5] = "Friday";
     pDays[6] = "Saturday";

     printf("The days of the Week\n");
     pLast = pDays + 6;

     for (pWalker=pDays; pWalker<=pLast ; pWalker++) {
          printf("%s\n", *pWalker);
     }
     return 0;
}
```

pDays에 각 문자열을 저장
문자열은 임의의 장소에 저장되며 pDays[0]등의 배열 원소는 그 문자열의 시작 주소를 갖는다



pDays의 내용을 순차적으로 출력

%c를 이용해 pWalker의 내용을 출력하면?

문자열 조작 함수

◈ **C**에서는 문자열을 관리하는 여러 함수들을 제공한다

- strlen() – 문자열의 길이 계산하는 함수
- strcpy(), strncpy() – 문자열을 복사하는 함수
- strcmp(), strncmp() – 두 문자열을 비교하는 함수
- strcat(), strncat() – 두 문자열을 결합하는 함수
- strtok() – 문자열을 자르는 함수

◈ 주교재 **11-5** 읽기

문자열 변환 함수

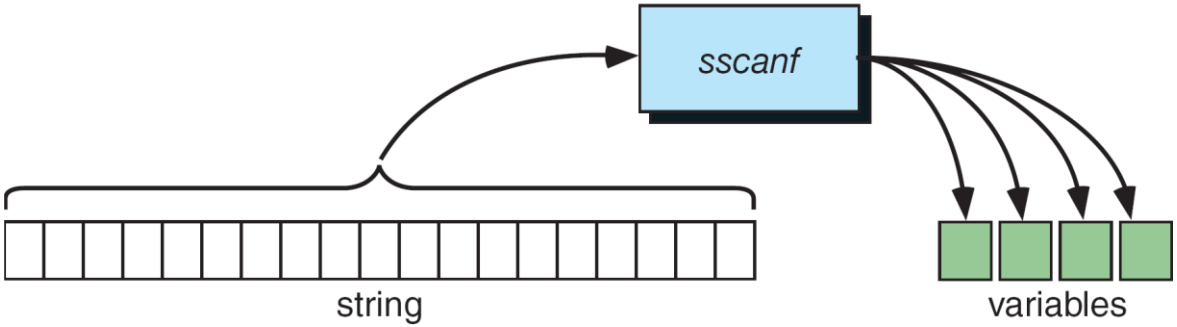
◈ 문자열 변환 함수

- int atoi(const char *s) – 문자열의 내용을 int형으로 변환하는 함수
- long atol(const char *s) – 문자열의 내용을 long형으로 변환하는 함수
- double atof(const char *s) – 문자열의 내용을 double형으로 변환하는 함수

실습 #3

문자열/데이터 변환

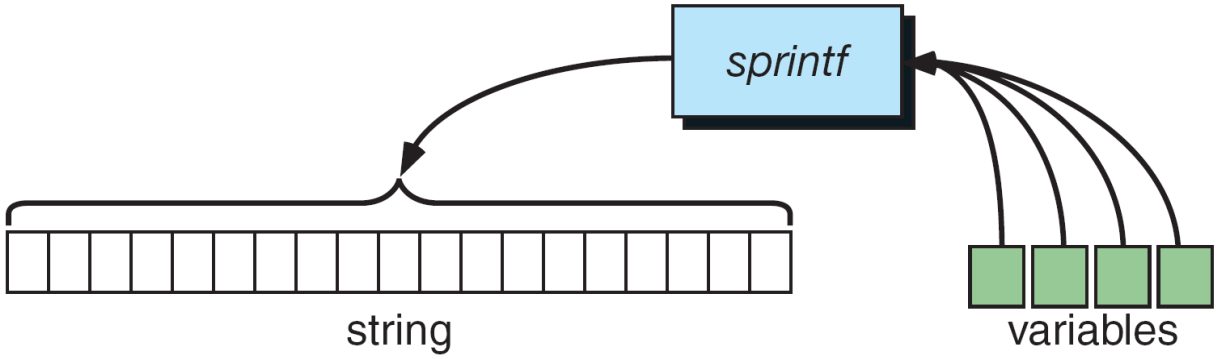
- 문자열로부터 데이터 형으로 변환하면서 읽거나, 데이터를 문자열에 쓴다
- **sscanf**




```
int sscanf(char* str, const char frmt_str, …);
```

문자열/데이터 변환

◈ sprintf




```
int sprintf(char* out_str,
            const char frmt_str, …);
```

문자열/데이터 변환

```
#include <stdio.h>
int main(void) {
   char name[30];
   char stuNo[5];
   int score;
   char grade;
   char strOut[80];
   char strIn[80] = "Einstein, Albert; 1234 97 A";

   printf("String contains: \"%s\"\n", strIn);
   sscanf(strIn, "%29[^;]%*c%4s%d%*[^ABCDF]%c", name, stuNo, &score, &grade);

   printf("Reformatted data: \n");
   printf("Name :\t\t\"%s\"\n", name);
   printf("Student No:\t\"%s\"\n", stuNo);
   printf("Score :\t\t%d\n", score);
   printf("Grade :\t\t%c\n", grade);

   sprintf(strOut, "%s %s %d %c", name, stuNo, score, grade);
   printf("New string:\"%s\"\n", strOut);
   return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
String contains: "Einstein, Albert; 1234 97 A"
Reformatted data:
Name :          "Einstein, Albert"
Student No:     "1234"
Score :         97
Grade :         A
New string:"Einstein, Albert 1234 97 A"

C:₩Users₩user₩Desktop₩C언어₩Project1₩x64₩Debug₩Projec
이 창을 닫으려면 아무 키나 누르세요...
```