chapter 06-4 참고자료

문자열의 처리

박 종 혁

Tel: 970-6702
Email: jhpark1@snut.ac.kr

출처: 뇌를 자극하는 C프로그래밍, 한빛미디어

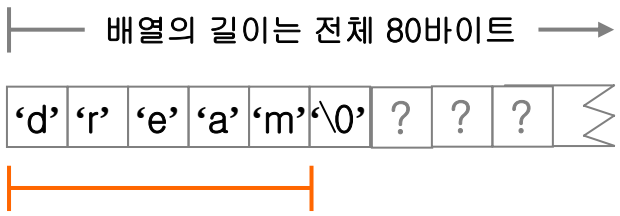
❑ 문자열의 연산

- 문자열은 배열의 형태로 구현된 응용자료형이므로 연산을 자유롭게 할 수 없다.

"straw" + "berry" → "strawberry" ??

- 배열에 저장된 문자열의 길이를 계산하는 작업도 간단하지 않다.

```
char str[80]=“dream”;
int count=0;
int i=0;
while(str[i] != ‘₩0’){
    count++;
    i++;
}
printf(“배열에 저장된 문자열의 길이 : %d₩n”, count);
```

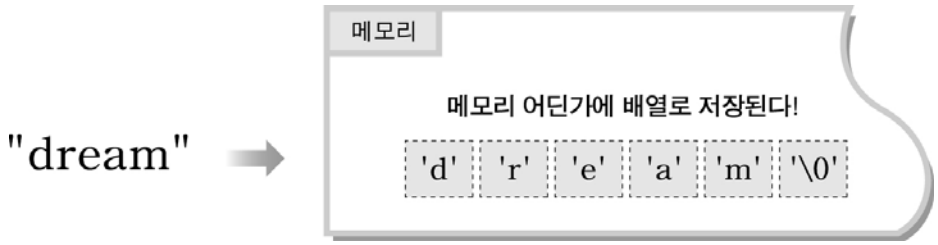


문자열의 길이는 5바이트!

- 문자열의 연산에는 문자열 복사, 길이 계산, 대소비교, 문자열 붙이기 등이 있다.

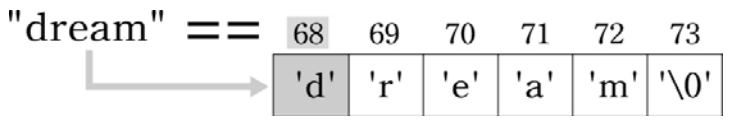
▶ 문자열상수는 포인터다!

- 프로그램에서 사용된 모든 문자열은 메모리에 배열의 형태로 저장된다.



- 문자열이 컴파일되면 첫 번째 문자를 가리키는 포인터로 치환된다. 결국 문자열상수는 char형을 가리키는 포인터이다!

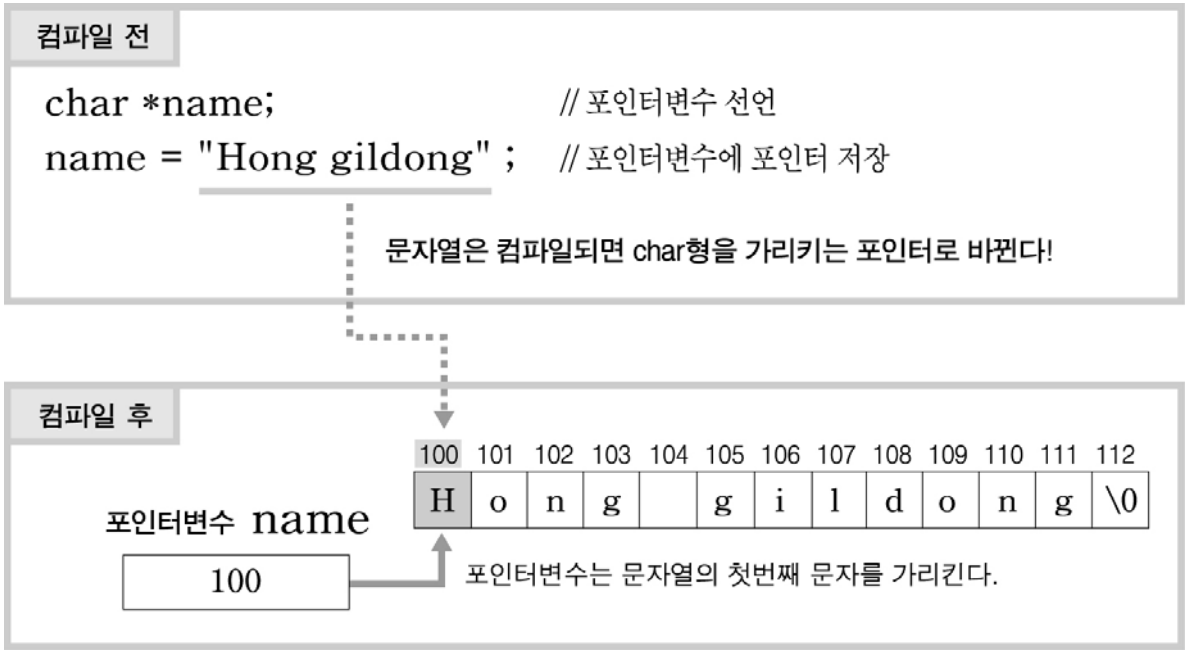
```
#include <stdio.h>

int main()
{
    printf("주소값을 출력 : %u₩n", "dream");
    printf("첫번째 문자를 출력 : %c₩n", *"dream");
    printf("세번째 문자를 출력 : %c₩n", "dream"[2]);
    return 0;
}
```




```
주소값을 출력 : 4350068
첫번째 문자를 출력 : d
세번재 문자를 출력 : e
```

▶ 포인터변수로 문자열 처리하기

- 문자열상수가 포인터이므로 포인터변수에 저장하여 사용할 수 있다.

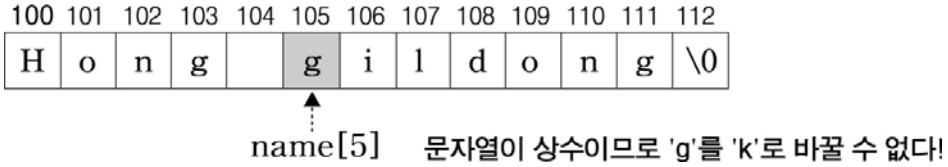



```
printf("이름 : %s₩n", name);
printf("여섯번째 문자 : %c₩n", name[5]);
```

▶ 포인터변수로 문자열 처리하기

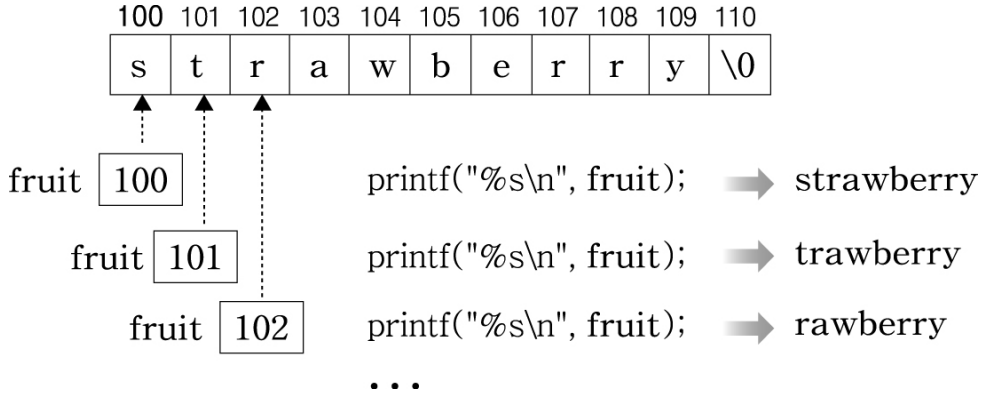
- 문자열상수는 상수이므로 포인터변수로 그 값을 바꿀 수 없다.

name[5] = 'k'



- 그러나 포인터변수의 값은 바꿀 수 있으므로 포인터변수가 문자열상수의 중간을 가리키게끔 할 수 있다.

```
#include <stdio.h>

int main()
{
    char *fruit="strawberry";
    while(*fruit != '₩0'){
        printf("%s₩n", fruit);
        fruit++;
    }
    return 0;
}
```



▶ 포인터변수로 문자열 처리하기

- 포인터변수는 여러 개의 문자열을 선택하여 처리할 수 있다.

```
#include <stdio.h>

int main()
{
    int age;
    char *greeting;

    printf("나이를 입력하세요 : ");
    scanf("%d", &age);

    if(age>30) greeting="처음 뵙겠습니다.";
    else greeting="반가워요.";     // 나이에 따라 서로 다른 문자열을 저장한다.

    printf("인사말 : %s₩n", greeting);
    return 0;
}
```

▶ 문자열상수의 두 얼굴

- 문자열상수를 포인터변수에 초기화하는 것과 배열에 초기화하는 것은 서로 다른 의미를 가진다.

```
char *fruit = “strawberry”;        // 포인터변수에 포인터를 초기화한다.
char fruit[20] = “strawberry”;     // 배열에 문자열상수의 데이터를 복사한다.
```

- 배열에 문자열상수로 초기화하는 것은 특별한 경우로 선언과 동시에 초기화할 때만 가능하다.

```
char fruit[20];

fruit  =  "strawberry" ;
```

배열명 주소상수 — 문자열 주소상수

⇨ 상수를 상수에 대입하는 것이므로 불가능하다!!

▶ 문자배열에 문자열상수를 저장하려면?

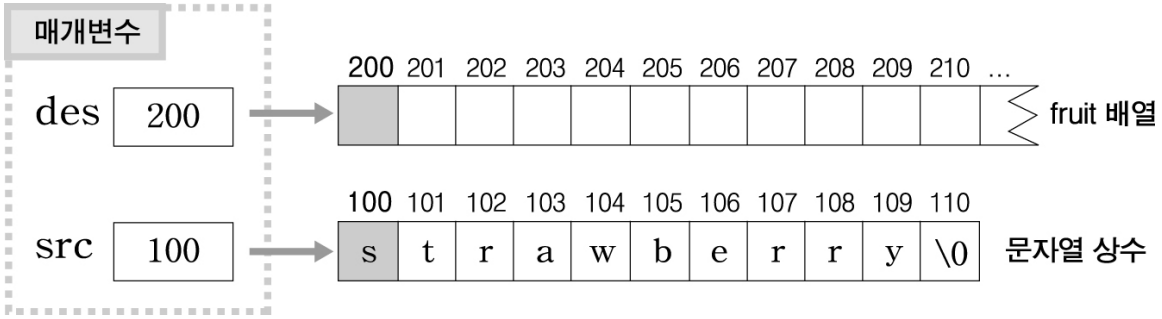
- 문자배열에 문자열상수를 저장하려면 문자열상수의 문자들을 직접 하나씩 배열에 복사해야 한다.
- 문자열을 복사하는 함수를 만들어보자.

```
user_strcpy(fruit, "strawberry");   // 함수의 호출
```

배열명 ------ 문자열상수

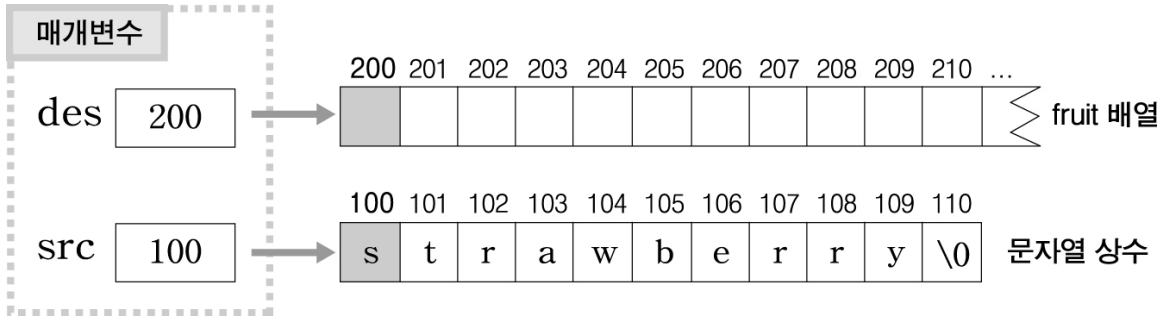
```
void user_strcpy(char *des, char *src);   // 함수의 원형 선언
```

복사될 배열 (destination) ------ 복사할 문자열 (source)



▶ 함수로 문자열을 복사하는 프로그램

```
#include <stdio.h>

void user_strcpy(char *, char *);
int main()
{
    char fruit[20];                    // fruit = “strawberry”; 와 같이 사용할 수 없으므로
    user_strcpy(fruit, “strawberry”);   // 함수를 만들어 문자열을 일일이 배열에 복사한다.
    printf(“배열에 저장된 문자열 : %s₩n”, fruit);
    return 0;
}

void user_strcpy(char *des, char *src)
{
    while(*src!=‘₩0’){
        *des = *src;
        src++;
        des++;
    }
    *des=‘₩0’;
}
```



❑ 중요한 문자열 처리 함수

- 문자열을 처리하는 함수는 라이브러리로 컴파일러와 함께 제공된다.
- 문자열 처리 함수를 사용할 때는 string.h헤더파일을 include한다.
- 문자열 복사함수 : strcpy(string copy)
- 문자열의 길이 계산 함수 : strlen(string length)
- 문자열 비교 함수 : strcmp(string compare)
- 두 개의 문자열을 붙이는 함수 : strcat(string concatenation)

▶ 문자열 복사함수(strcpy)

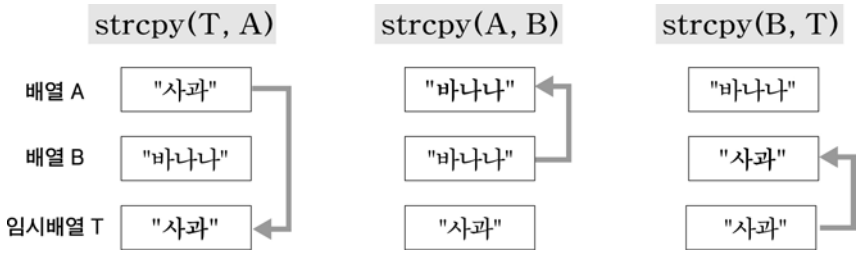
- 문자열상수나 배열에 저장된 문자열을 다른 배열에 복사한다.

```
char *strcpy(char *, char *);   // 문자열의 복사, string copy
```

  – 두 번째 전달인자로 주어지는 문자열을 첫번째 전달인자의 위치에 복사한다.

- 두 배열에 저장된 문자열을 바꾸는 프로그램

```
#include <stdio.h>
#include <string.h>      // 헤더파일 포함
int main()
{
    char str1[20]="apple";
    char str2[20]="banana";
    char temp[20];
    strcpy(temp, str1);    // "apple" -> temp
    strcpy(str1, str2);    // "banana" -> str1
    strcpy(str2, temp);    // "apple" -> str2
    printf("str1 : %s₩n", str1);
    printf("str2 : %s₩n", str2);
    return 0;
}
```



▶ 문자열의 길이를 계산하는 함수(strlen)

- 문자열상수나 배열에 저장된 문자열을 다른 배열에 복사한다.

```
unsigned int strlen(char *);  // 문자열의 길이 계산, string length
```

  - 전달인자로 주어지는 문자열의 길이를 계산하여 리턴한다(널문자 제외).

```
char fruit[80] = “apple”;
int len;
len = strlen(fruit);
printf(“문자열의 길이 : %d₩n”, len);    // 문자열의 길이 : 5
```

- 배열에 저장된 문자열의 길이만을 계산해 준다.

```
sizeof(fruit)  ➡  80   // 배열 전체의 크기
strlen(fruit)  ➡  5    // 배열에 저장된 문자열의 크기
```

- 문자열상수나 문자열을 가리키는 포인터변수를 사용할 수도 있다.

```
char *strp = “apple”;
strlen(strp);
strlen(“banana”);
```

▶ 문자열을 비교하는 함수(strcmp)

- 두 문자열의 사전적 순서를 따진다.

```
int strcmp(char *str1, char *str2); // 문자열의 비교, string compare
```

- 두 문자열을 비교하여 리턴하는 값(사전의 뒤에 나오는 것이 큰 문자열이다).

크기 비교	str1 > str2	str1 < str2	str1 == str2
리턴값	1	-1	0

- 두 문자열의 순서를 따져서 사전 순서로 바꾸는 예

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

int main()
{
    char str1[20]="banana";
    char str2[20]="apple";
    char temp[20];
    int res;

    res=strcmp(str1, str2);
    if(res>0){          // str1이 더 크면 문자열을 바꾼다.
        strcpy(temp, str1);
        strcpy(str1, str2);
        strcpy(str2, temp);
    }
    printf("str1 : %s₩n", str1);
    printf("str2 : %s₩n", str2);
    return 0;
}
```

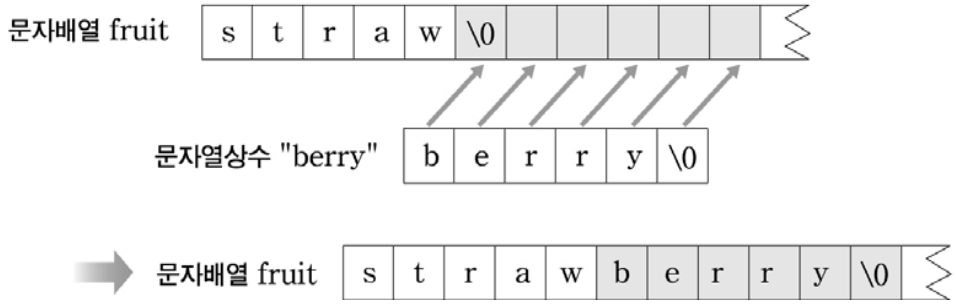
▶ 두 개의 문자열을 붙이는 함수(strcat)

- 두 문자열을 붙여서 하나의 문자열을 만든다.

```
char *strcat(char *str1, char *str2);  // string concatenation
```

- str2의 문자열을 str1의 문자열 뒤에 붙인다.

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

int main()
{
    char fruit[80]="straw";
    strcat(fruit, "berry");
    printf("연결된 문자열 : %s₩n", fruit);

    return 0;
}
```



❑ 문자열의 입출력

- scanf함수는 빈칸이 포함된 문자열은 입력할 수 없다.

```
char str[80];
printf("문자열을 입력하세요 : ");
scanf("%s", str);
printf("입력된 문자열 : %s₩n", str);
```

```
문자열을 입력하세요 : 백번 보는 것보다 한번 짜보는 것이 낫다. (엔터)
입력된 문자열 : 백번      // 첫번재 빈칸 이후의 문자열은 입력되지 않았다.
```

- 문자열의 입출력은 전용함수를 사용한다.
- 한 줄을 모두 입력하는 함수(gets)
- 문자열을 출력하는 함수(puts)

▶ 한 줄을 모두 입력하는 함수(gets)

- gets함수는 빈칸을 포함하여 한 줄을 입력할 수 있다.

```
char *gets(char *);  // 문자열의 입력, get string
```

  - 전달인자는 입력 받을 배열의 주소이다.
  - 입력될 문자열이 충분히 저장될 수 있도록 배열을 선언해야 한다.
  - 데이터를 입력한 후에 마지막에 널문자를 붙여서 문자열을 완성한다.

```
char str[80];
printf("문자열을 입력하세요 : ");
gets(str);
printf("입력된 문자열 : %s₩n", str);
```

```
문자열을 입력하세요 : 백번 보는 것보다 한번 짜보는 것이 낫다. (엔터)
입력된 문자열 : 백번 보는 것보다 한번 짜보는 것이 낫다.
```

▶ 문자열을 출력하는 함수(puts)

- puts함수는 문자열만을 출력하는 전용 함수이다.

```
int puts(char *);  // 문자열의 출력, put string
```

  - 문자열만을 출력하므로 printf함수보다 훨씬 작고 간편하다.
  - 출력할 문자열이 저장된 배열명을 전달인자로 준다.
  - 문자열을 출력한 후에는 자동으로 줄이 바뀐다.

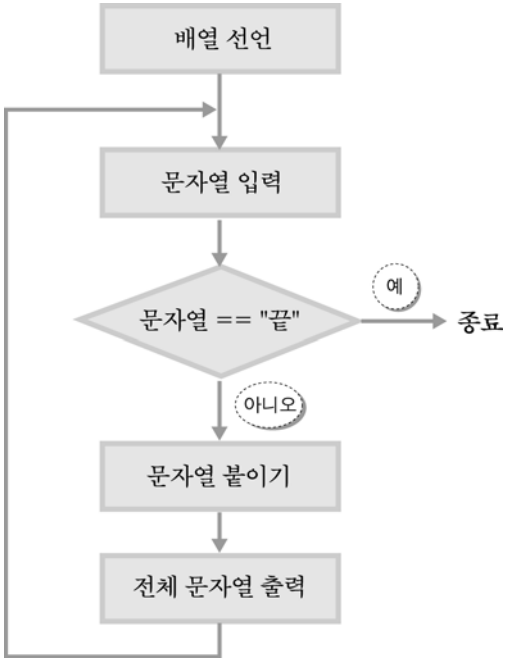
```
char str[80];
printf("문자열을 입력하세요 : ");
gets(str);
printf("입력된 문자열 : ");
puts(str)                    // printf("%s₩n", str);
```

```
문자열을 입력하세요 : 백번 보는 것보다 한번 짜보는 것이 낫다. (엔터)
입력된 문자열 : 백번 보는 것보다 한번 짜보는 것이 낫다.
_          // 커서의 위치!
```

▶ 소설 이어쓰기 프로그램(문자열 처리 함수 사용 예)

- 키보드로 하나의 문장들을 입력 받을 때마다 이미 입력 받은 문장들과 연결하여 전체를 출력한다. “끝”이 입력되면 프로그램을 종료한다.

```
문자열을 입력하세요 : 문자열은 (엔터)
현재까지의 줄거리 : 문자열은

문자열을 입력하세요 : 컴파일 되면 (엔터)
현재까지의 줄거리 : 문자열은 컴파일 되면

문자열을 입력하세요 : 주소를 (엔터)
현재까지의 줄거리 : 문자열은 컴파일 되면 주소를

문자열을 입력하세요 : 남긴다 (엔터)
현재까지의 줄거리 : 문자열은 컴파일 되면 주소를 남긴다.

문자열을 입력하세요 : 끝 (엔터)          // 프로그램 종료
```



▶ 소설 이어쓰기 프로그램(문자열 처리 함수 사용 예)

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

int main()
{
    char novel[800]={0};        // 전체 줄거리를 저장할 배열, 초기화가 필요하다!
    char str_in[80];            // 입력 문자열을 저장할 배열

    while(1){
        printf("문자열을 입력하세요 : ");
        gets(str_in);                                   // 한 줄을 입력한다.
        if(strcmp(str_in, "끝")==0) break;              // 입력된 문자열이 “끝”이면 반복문을 빠져 나간다.

        strcat(novel, str_in);                          // 입력된 문자열을 전체 줄거리에 붙인다.
        strcat(novel, " ");                             // 다음 문자열을 위해 한 칸 띄운다.
        printf("현재까지의 줄거리 : ");
        puts(novel);                                    // 전체 줄거리 출력
        puts("₩n");                                     // 한 줄을 띄운다.
    }

                                // novel배열을 널문자로 초기화하지 않으면 쓰레기 문자들이 있으므로
    return 0;                   // 처음에 문자열 붙일 때 쓰레기 문자들이 남을 가능성이 있다!!
}
```

❑ 문자 입출력 함수

- 하나의 문자만을 전용으로 입출력하는 함수들이 있다.
- 문자들을 연속으로 입출력하면 문자열의 입출력이 된다.
- 문자 입력 함수 : getchar
- 문자 출력 함수 : putchar

▶ 하나의 문자를 입출력 하자.

- 하나의 문자를 입출력할 때는 문자 전용 입출력함수를 사용하는 것이 효율적이다.

```
int getchar();        // 하나의 문자를 입력, get character
int putchar(int);     // 하나의 문자를 출력, put character
```

- getchar함수는 키보드로부터 문자를 입력 받아서 리턴한다.
- putchar함수는 전달인자로 주어지는 문자를 화면에 출력한다.
- 두 함수 모두 stdio.h 헤더파일을 include하면 사용할 수 있다.

```
#include <stdio.h>

int main()
{
    int ch;
    printf("문자 하나를 입력하세요 : ");
    ch=getchar();
    printf("입력된 문자 : ");
    putchar(ch);
    return 0;
}
```

▶ 문자열을 입출력 하자.

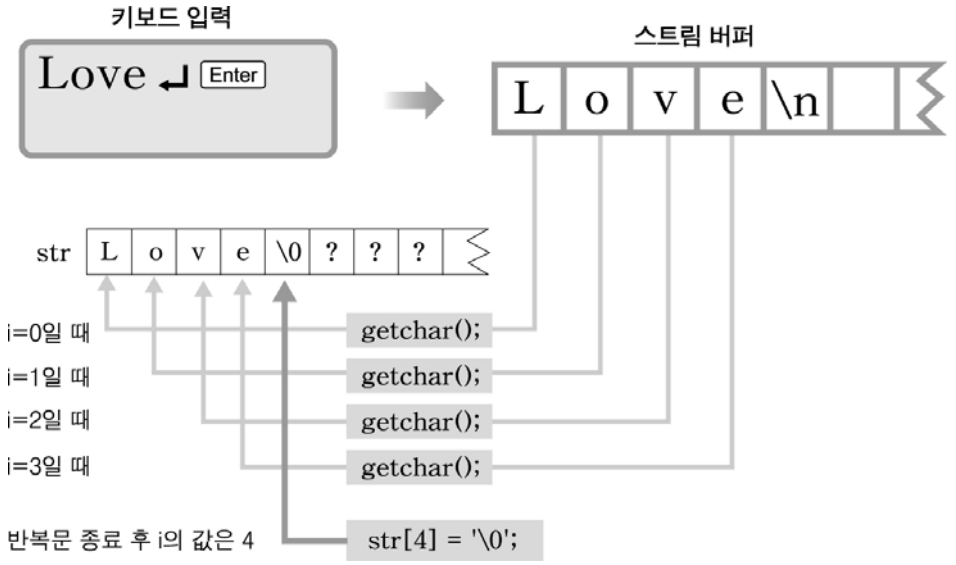
- 하나의 문자를 반복적으로 입출력하면 문자열의 입출력이 된다.
  - 문자열을 입력할 때는 마지막에 반드시 널문자를 채워준다.
  - 배열에 “Love”문자열을 입력하는 예

```
#include <stdio.h>
int main()
{
    char str[80];                       // 문자열을 저장할 배열
    int ch;                             // getchar함수의 리턴값을 저장할 변수
    int i;                              // 반복 제어변수
    printf(“문자열을 입력하세요 : ”);
    for(i=0; i<4; i++){                 // i는 0부터 3까지 변하면선 4번 반복
        ch=getchar();                   // 키보드로부터 문자를 입력 받아 리턴한다.
        str[i]=ch;                      // 리턴된 문자를 배열에 차례로 저장
    }
    str[i]=‘₩0’;                        // 마지막에 널문자를 저장하여 문자열을 완성한다.
    printf(“입력된 문자열 : ”);
    i=0;                                // 제어변수를 다시 0으로 초기화
    while(str[i]=‘₩0’){                 // 배열요소가 널문자가 아닌 동안 반복
        putchar(str[i]);                // 화면에 문자 출력
        i++;                            // 다음 문자로 이동
    }
    return 0;
}
```

▶ 데이터의 입력은 버퍼를 사용한다.

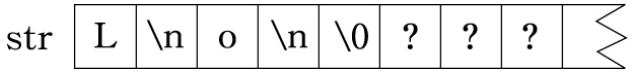
- 키보드에서 입력되는 데이터는 일단 버퍼에 저장되고 getchar함수는 버퍼로부터 데이터를 가져오므로 문자열은 한번에 입력한다.

```
문자열을 입력하세요 : Love (엔터)
입력된 문자열 : Love
```



- 문자를 하나씩 입력하면 새줄문자로 입력 되므로 문제가 발생한다.

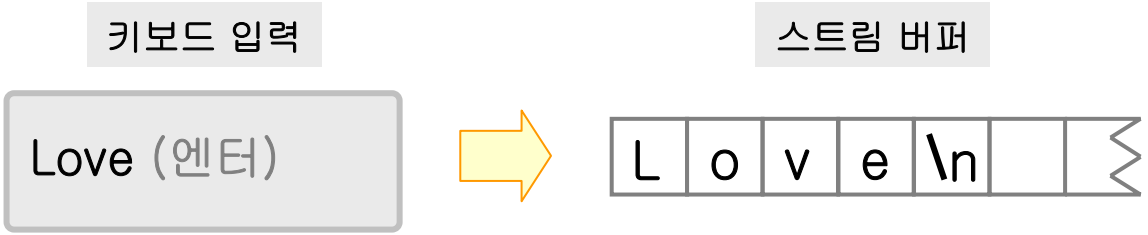
```
문자열을 입력하세요 : L (엔터)
o (엔터)        // 두 번째 문자를 입력할 때 입력이 끝난다.
입력된 문자열 : L
o
```



▶ 한 줄을 입력 하자.

- getchar함수가 새줄문자(‘₩n’)도 하나의 문자로 입력하므로 이 문자를 이용하여 한 줄을 입력 받을 수 있다.

```
printf(“문자열을 입력하세요 : ”);
while(1){                          // 무한 반복
    ch=getchar();                  // 퍼버로부터 문자 하나를 입력한다.
    if(ch==‘₩n’) break;            // 그 문자가 새줄문자이면 입력 종료
    str[i]=ch;                     // 배열요소에 입력한 문자를 저장
    i++;                           // 다음 배열요소로 이동
}
str[i]=‘₩0’;                       // 마지막에 널문자를 넣어 문자열을 완성한다.
```



▶ 여러 줄을 입력 하자.

- 새줄문자도 하나의 문자로 입력하면 여러 줄을 하나의 문자열로 입력할 수 있다. 입력의 종료는 Ctrl + Z키를 누른다.
  - getchar함수는 키보드에서 Ctrl + Z키가 눌려지면 -1을 리턴하므로 이 값을 비교하여 입력을 종료할 수 있다.

```
printf("문자열을 입력하세요 : ");
while(1){
    ch=getchar();
    if(ch==-1) break;     // getchar함수가 -1을 리턴하면 입력을 종료한다.
    str[i]=ch;
    i++;
}
str[i]='₩0';
printf("입력된 문자열 : ");
puts(str);
```

```
문자열을 입력하세요 : 10분 더 공부하면 (엔터)
배우자의 직업이 바뀐다. (엔터)
^Z (엔터)
입력된 문자열 : 10분 더 공부하면
배우자의 직업이 바뀐다.
```