

# 컴퓨터 프로그래밍 및 실습

## 강의자료 6

- 동적 할당
- 문자열

# 메모리 구성과 영역별로 저장되는 데이터의 유형

코드 영역

실행할 프로그램의 코드가 저장되는 메모리 공간

데이터 영역

전역변수와 **static** 변수가 할당되는 영역  
프로그램 시작과 동시에 할당되어 종료시까지 남아있음

힙 영역

동적 메모리 할당

스택 영역

함수 수행시 메모리 할당이 일어나고  
함수를 빠져 나가면 소멸됨

# 동적 메모리 할당 연산자 **malloc, free**

- 메모리를 동적으로 할당하기 위해서 malloc, free를 사용한다

```
#include <stdlib.h>
void * malloc(size_t size);    // 힙 영역으로의 메모리 공간 할당
void free(void * ptr);        // 힙 영역에 할당된 메모리 공간 해제
```

➔ malloc 함수는 성공 시 할당된 메모리의 주소 값, 실패 시 NULL 반환

```
int main(void)
{
    void * ptr1 = malloc(4);    // 4바이트가 힙 영역에 할당
    void * ptr2 = malloc(12);   // 12바이트가 힙 영역에 할당
    . . . .
    free(ptr1);    // ptr1이 가리키는 4바이트 메모리 공간 해제
    free(ptr2);    // ptr2가 가리키는 12바이트 메모리 공간 해제
    . . . .
}
```

# 동적 메모리 할당 연산자 **malloc, free**

malloc 함수의 일반적인 호출형태

```
void * ptr1 = malloc(sizeof(int));  
void * ptr2 = malloc(sizeof(double));  
void * ptr3 = malloc(sizeof(int)*7);  
void * ptr4 = malloc(sizeof(double)*9);
```

sizeof 연산 이후 실질적인 malloc의 호출

```
void * ptr1 = malloc(4);  
void * ptr2 = malloc(8);  
void * ptr3 = malloc(28);  
void * ptr4 = malloc(72);
```

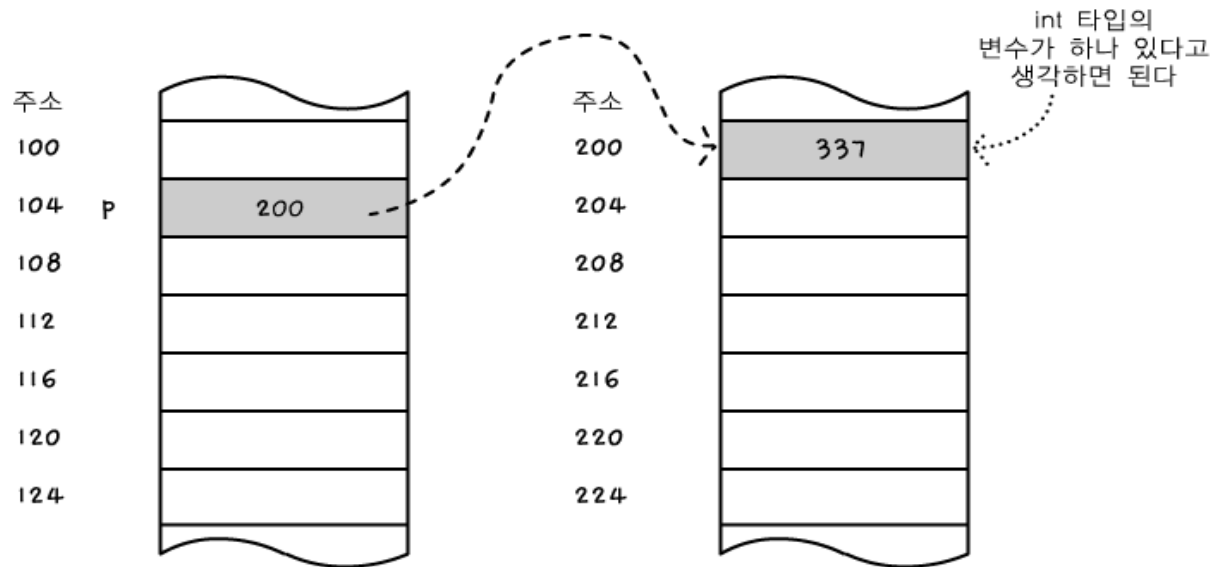
malloc 함수는 인자로 숫자만 하나 전달받을 뿐이니 할당하는 메모리의 용도를 알지 못한다. 따라서 메모리의 포인터 형을 결정짓지 못한다. 따라서 다음과 같이 형 변환의 과정을 거쳐서 할당된 메모리의 주소 값을 저장해야 한다.

```
int * ptr1 = (int *)malloc(sizeof(int));  
double * ptr2 = (double *)malloc(sizeof(double));  
int * ptr3 = (int *)malloc(sizeof(int)*7);  
double * ptr4 = (double *)malloc(sizeof(double)*9);
```

malloc 함수의  
가장모범적인 호출형태

# 동적 메모리 할당 연산자 **malloc, free**

```
// int 하나를 담을 수 있는 크기의 메모리 공간을 할당한다.  
int* p = (int *)malloc(sizeof(int));  
  
// 메모리에 값을 넣어본다.  
*p = 337;  
  
// 사용이 끝난 메모리를 해제한다.  
free(p);
```



# 할당된 동적 메모리 접근

```
int main(void)
{
    int * ptr1 = (int *)malloc(sizeof(int));
    int * ptr2 = (int *)malloc(sizeof(int)*7);
    int i;
    *ptr1 = 20;
    for(i=0; i<7; i++)
        ptr2[i]=i+1;
    printf("%d \n", *ptr1);
    for(i=0; i<7; i++)
        printf("%d ", ptr2[i]);

    free(ptr1);
    free(ptr2);
    return 0;
}
```

```
int * ptr = (int *)malloc(sizeof(int));
if(ptr==NULL)
{
    // 메모리 할당 실패에 따른 오류의 처리
}
```

동적 메모리 접근은 포인터를 통해서 이뤄진다.

실행결과

20

1 2 3 4 5 6 7

‘동적할당’이라 하는 이유!

컴파일시 할당에 필요한 메모리 공간이 계산되지 않고, 실행시 할당에 필요한 메모리 공간이 계산되므로!

# calloc

```
#include <stdlib.h>
void * calloc(size_t elt_count, size_t elt_size);
```

➔ 성공 시 할당된 메모리의 주소 값, 실패 시 NULL 반환

elt\_count × elt\_size 크기의 바이트를 동적 할당한다.  
즉, elt\_size 크기의 블록을 elt\_count의 수만큼 동적할당!  
그리고 malloc 함수와 달리 모든 비트를 0으로 초기화!

사용방법 :

포인터변수이름 = (자료형 \*) calloc (개수, 크기)

사용 예:

```
int* ptr;
```

```
ptr = (int *)calloc(5,sizeof(int));
```

# 동적 메모리 배열 할당과 해제 - 예

- 동적으로 메모리를 할당하기

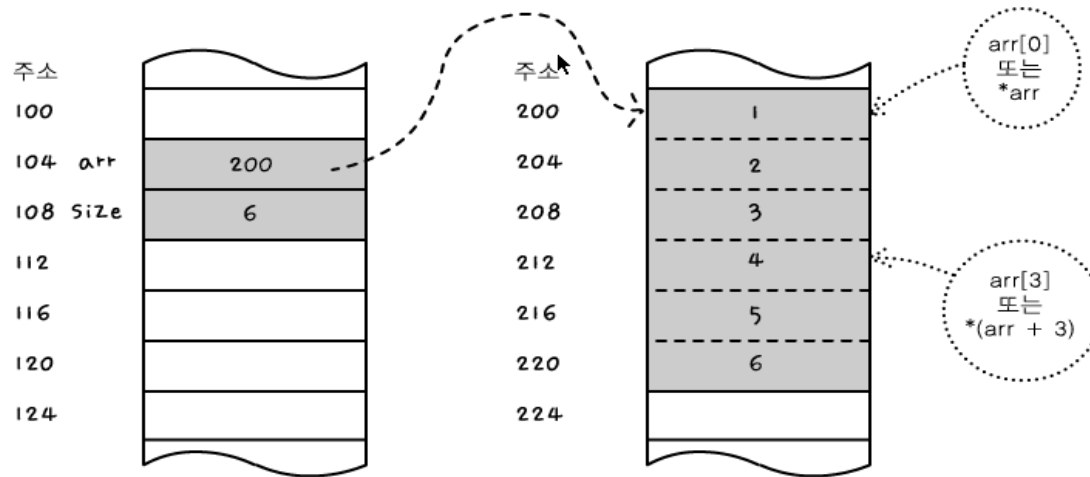
```
int size;  
scanf("%d", &size);
```

```
int* arr = (int *)malloc(sizeof(int)*size);
```



# 동적 메모리 배열 할당과 해제

- 메모리를 할당했을 때의 메모리 상태



- 동적으로 할당한 메모리를 해제하기  
`free(arr);`

# 동적 메모리 할당의 규칙

- malloc과 free 쌍을 맞춰서 사용하자.

(1) free 함수를 호출하지 않으면?

- 할당된 메모리 공간은 메모리라는 중요한 리소스를 계속 차지하게 된다

(2) free 함수를 호출하지 않으면 프로그램 종료 후에도 메모리를 차지하는가?

- 프로그램이 종료되면 프로그램 실행시 할당된 모든 자원이 반환된다

(3) fopen, fclose가 늘 쌍을 이루듯 malloc, free도 쌍을 이루게 하자!

# 동적 메모리 할당의 응용 - 배열 할당

## ● 입력 받은 정수들의 합과 평균을 구하는 예

```
// 몇 개의 정수를 입력할지 물어본다.
int size;
printf("몇 개의 정수를 입력하시겠습니까? \n");
scanf("%d", &size);

// 필요한 만큼의 메모리를 할당한다 (배열 할당).
int* arr = (int *)malloc(sizeof(int)*size);

// 정수를 입력받는다.
printf("정수를 입력하십시오.\n");
for (int i = 0; i < size; ++i)
    scanf("%d", &arr[i]);

// 평균을 계산하고 출력한다.
int sum = 0;
for (i = 0; i < size; ++i)
{
    sum += arr[i];
}
float ave = (float)sum / (float)size;
printf("합 = " %d 평균 = %f \n", sum, ave);

// 사용한 메모리를 해제한다.
free(arr);
```

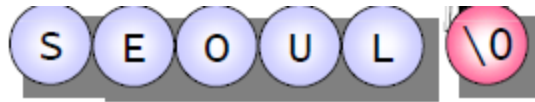
문자열

# 문자열

- 문자열(string): 문자들이 여러 개 모인 것
  - "A"
  - "Hello World!"
  - "변수 score의 값은 %d입니다"
- 문자열 상수
  - "Hello World"
  - "Hong"
  - "string!#\$"
  - "guest123"
- 문자열 변수
  - char형 배열

# NULL 문자

- NULL 문자 문자열의 끝을 나타낸다



- 문자열은 어디서 종료되는지 알 수가 없으므로 표시를 해주어야 한다.

# 문자 배열의 초기화

1. 문자 배열 원소들을 중괄호 안에 넣어주는 방법

– `char str[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };`

2. 문자열 상수를 사용하여 초기화하는 방법

– `char str[6] = "Hello";`

3. 만약 배열을 크기를 지정하지 않으면 컴파일러가 자동으로 배열의 크기를 초기화 값에 맞추어 설정

– `char str[] = "C Bible";` // 배열의 크기는 8이 된다.

# 문자 배열에 문자를 저장

1. 각각의 문자 배열 원소에 원하는 문자를 개별적으로 대입하는 방법 이다.

- `str[0] = 'W';`
- `str[1] = 'o';`
- `str[2] = 'r';`
- `str[3] = 'l';`
- `str[4] = 'd';`
- `str[5] = '\0';`

2. `strcpy()`를 사용하여 문자열을 문자 배열에 복사

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



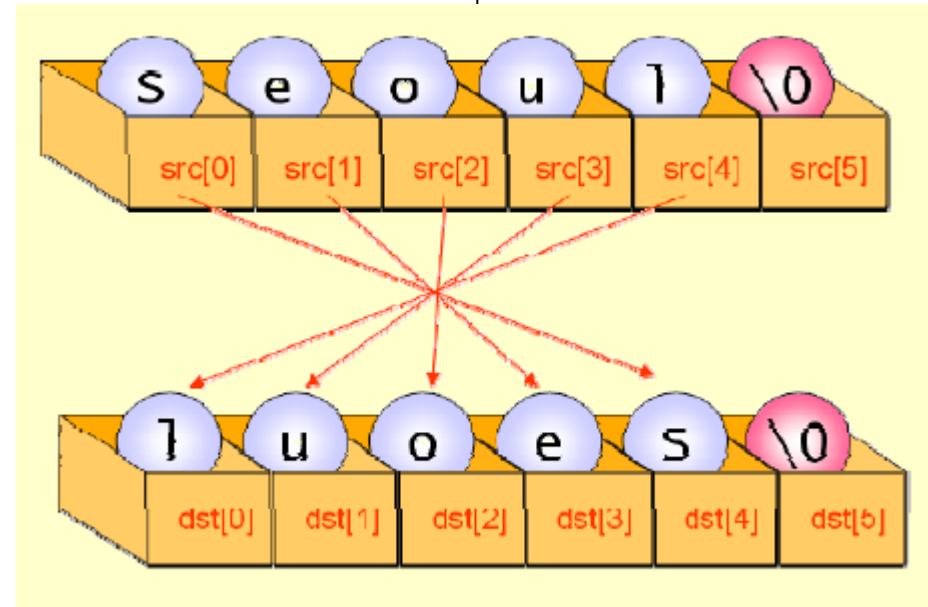
# 예제

```
#include <stdio.h>
int main(void)
{
    char str[] = "komputer";
    int i;
    for(i=0;i<8;i++)
        printf("%c ", str[i]);
    str[0] = 'c';
    printf("\n");
    for(i=0;i<8;i++)
        printf("%c ", str[i]);
    return 0;
}
```

k o m p u t e r  
c o m p u t e r

# 문자열 역순 예

```
#include <stdio.h>
int main(void)
{
    char src[] = "Seoul";
    char dst[6];
    int i;
    printf("원래 문자열=%s\n", src);
    i = 0;
    while(src[i] != '\0')
    {
        dst[i] = src[4 - i];
        i++;
    }
    dst[i] = '\0';
    printf("역순 문자열=%s\n", dst);
    return 0;
}
```



원래 문자열=Seoul  
역순 문자열=luoeS

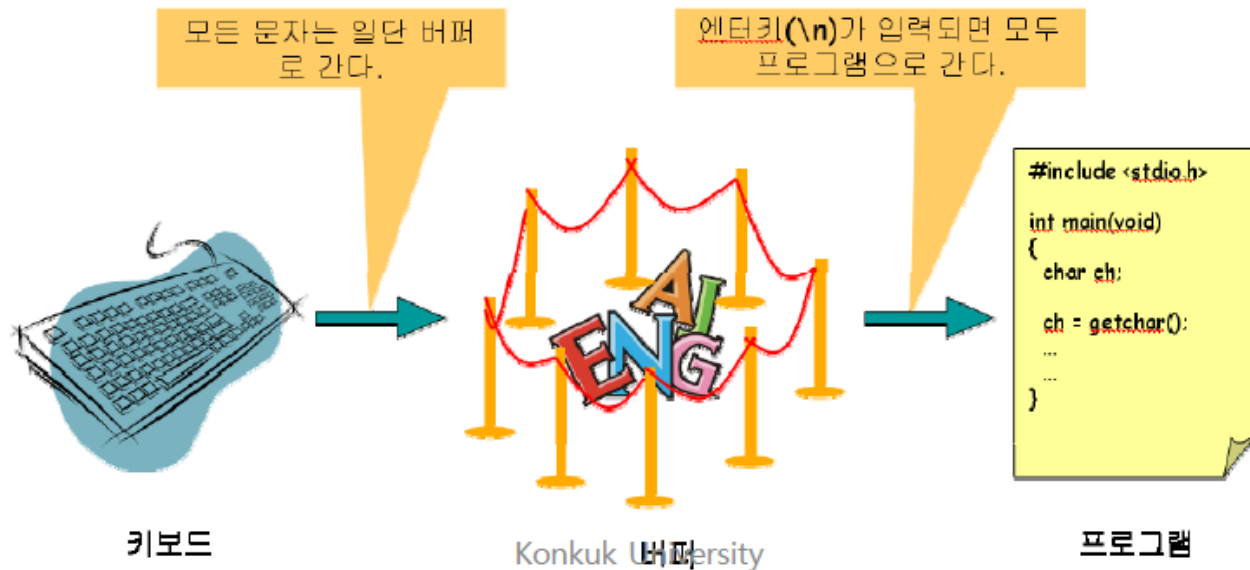
# 문자열 길이 구하는 예

```
// 문자열의 길이를 구하는 프로그램
#include <stdio.h>
int main(void)
{
    char str[30] = "C language is easy";
    int i = 0;
    while(str[i] != 0)
        i++;
    printf("문자열 \"%s\"의 길이는 %d입니다.\n", str, i);
    return 0;
}
```

문자열 "C language is easy"의  
길이는 18입니다.

# 문자 입출력 라이브러리

입출력 함수	설명
<code>int getchar(void)</code>	하나의 문자를 읽어서 반환한다.
<code>void putchar(int c)</code>	변수 <code>c</code> 에 저장된 문자를 출력한다.
<code>int getch(void)</code>	하나의 문자를 읽어서 반환한다(버퍼를 사용하지 않음).
<code>void putch(int c)</code>	변수 <code>c</code> 에 저장된 문자를 출력한다(버퍼를 사용하지 않음).
<code>scanf("%c", &amp;c)</code>	하나의 문자를 읽어서 변수 <code>c</code> 에 저장한다.
<code>printf("%c", c);</code>	변수 <code>c</code> 에 저장된 문자를 출력한다.



# getchar(), putchar()

```
// getchar()의 사용
#include <stdio.h>

int main(void)
{
    int ch;                // 정수형에 주의

    while(1)
    {
        ch = getchar();    // 문자를 입력 받는다.
        if( ch == 'q' ) break;
        putchar(ch);
    }
    return 0;
}
```

```
A
A
B
B
q
```

# getch(), putch()

```
// getch()의 사용  
#include <conio.h>
```

```
int main(void)
```

```
{  
    int ch;           // 정수형에 주의
```

버퍼를 사용하지 않는다

```
    while(1)
```

```
    {  
        ch = getch(); // 문자를 입력받는다.
```

```
        if( ch == 'q' ) break;
```

```
        putch(ch);
```


```
    }
```

```
    return 0;
```

```
}
```

ABCDEFGH

# getch(), getche(), getchar()

	헤더 파일	버퍼 사용 여부	에코 여부	응답성	문자 수정 여부
getchar()	<stdio.h>	사용함 (엔터키를 눌러 입력됨)	에코	줄 단위	가능
getch()	<conio.h>	사용하지 않음	에코하지 않음	문자 단위	불가능
getche()	<conio.h>	사용하지 않음	에코	문자 단위	불가능

# scanf(), printf() 문자열 입출력

- scanf()의 사용법
  - char str[10];
    - scanf("%s", str);
- scanf()는 한 번에 두개 이상의 문자열도 받아들일 수 있다.
  - char s1[10];
  - char s2[10];
  - char s3[10];
  - scanf("%s%s%s", s1,s2,s3);
  - // 사용자가 one two three와 같이 입력하면 s1에는 one이, s2에는 two가, s3에는 three가 할당된다.



# gets()와 puts() 문자열 입출력

- gets()

- 표준 입력으로부터 엔터키가 나올 때까지 한 줄의 라인을 입력
- 문자열에 줄바꿈 문자('\n')는 포함되지 않으며 대신에 자동으로 NULL 문자('\0')를 추가한다.
- 입력 받은 문자열은 buffer가 가리키는 주소에 저장된다.

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

- puts()

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

```
char *menu = "파일열기: open, 파일닫기: close";  
puts("메뉴에서 하나를 선택하십시오.");  
puts(str);
```

# 예제

```
#include <stdio.h>

int main( void )
{
    char buffer[21]; // 20개의 문자와 '\0'을 저장할 수 있다.

    printf("문자열을 입력하십시오.\n");
    gets( buffer );

    printf("입력된 라인은 다음과 같습니다.\n");
    puts(buffer);
    return 0;
}
```

문자열을 입력하십시오.

Hello!

입력된 라인은 다음과 같습니다.

Hello!

# 문자 처리 라이브러리 함수

문자를 검사하거나 문자를 변환한다.

# include ctype.h

함수	설명
isalpha(c)	c가 영문자인가?(a-z, A-Z)
isupper(c)	c가 대문자인가?(A-Z)
islower(c)	c가 소문자인가?(a-z)
isdigit(c)	c가 숫자인가?(0-9)
isalnum(c)	c가 영문자이나 숫자인가?(a-z, A-Z, 0-9)
isxdigit(c)	c가 16진수의 숫자인가?(0-9, A-F, a-f)
isspace(c)	c가 공백문자인가?(' ', '\n', '\t', '\v', '\r')
ispunct(c)	c가 구두점 문자인가?
isprint(c)	C가 출력가능한 문자인가?
isctrl(c)	c가 제어 문자인가?
isascii(c)	c가 아스키 코드인가?
toupper(c)	c를 대문자로 바꾼다.
tolower(c)	c를 소문자로 바꾼다.
toascii(c)	c를 아스키코드로 바꾼다

# 예제

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

```
int main( void )
{
    int c;

    while((c = getchar()) != EOF)
    {
        if( islower(c) )
            c = toupper(c);
        putchar(c);
    }
    return 0;
}
```

소문자인지 검사  
대문자로 변환

```
abcdef
ABCDEF
^Z
```

# 문자열 처리 라이브러리

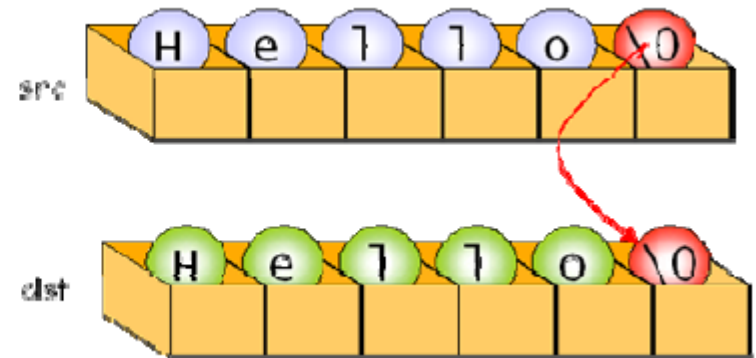
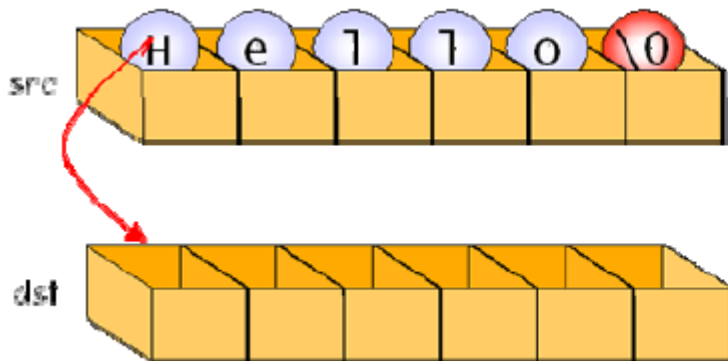
- #include <string.h>

함수	설명
strlen(s)	문자열 s의 길이를 구한다.
strcpy(s1, s2)	s2를 s1에 복사한다.
strcat(s1, s2)	s2를 s1의 끝에 붙여넣는다.
strcmp(s1, s2)	s1과 s2를 비교한다.
strncpy(s1, s2, n)	s2의 최대 n개의 문자를 s1에 복사한다.
strncat(s1, s2, n)	s2의 최대 n개의 문자를 s1의 끝에 붙여넣는다.
strncmp(s1, s2, n)	최대 n개의 문자까지 s1과 s2를 비교한다.
strchr(s, c)	문자열 s안에서 문자 c를 찾는다.
strstr(s1, s2)	문자열 s1에서 문자열 s2를 찾는다.

# 문자열 길이, 문자열 복사

- 문자열의 길이
  - `strlen("Hello")`는 5를 반환
- 문자열 복사

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



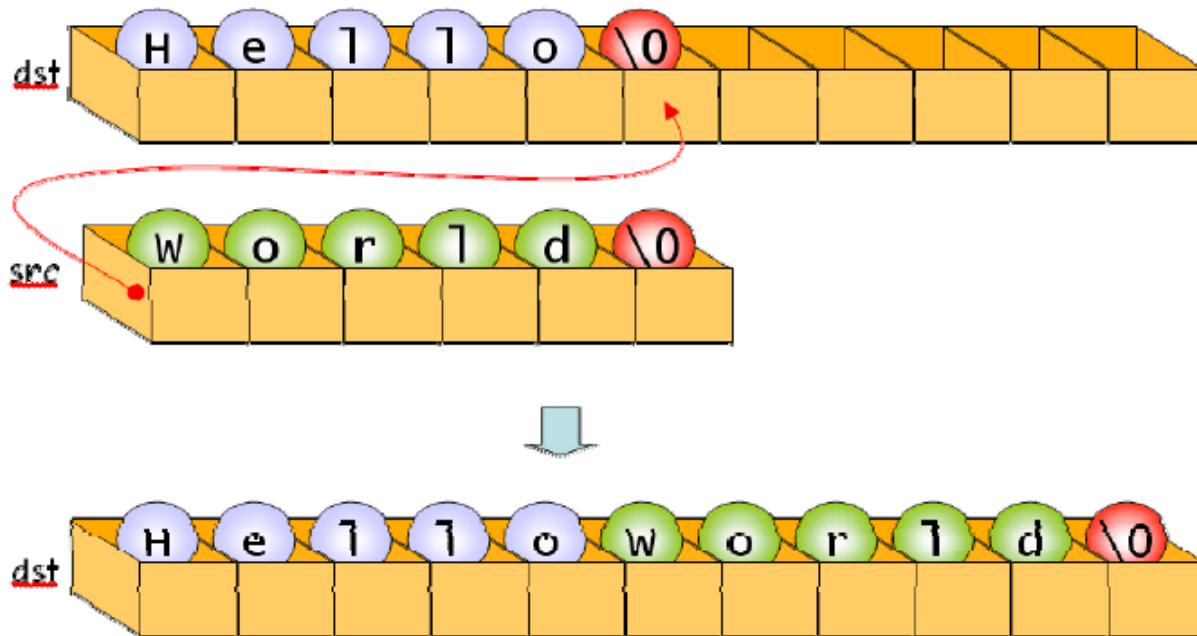
# 문자열 연결

- 문자열 연결

```
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;
}
```

string = Hello world from strcpy and strcat!



# 문자열 비교

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

반환값

<0

0

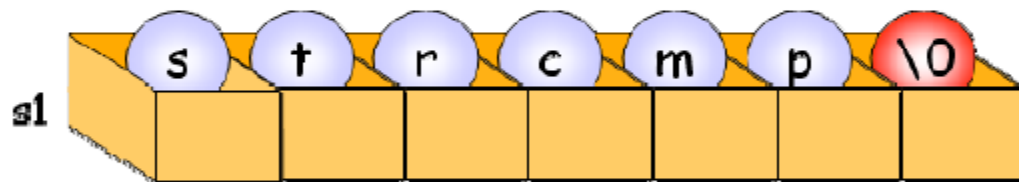
>0

s1과 s2의 관계

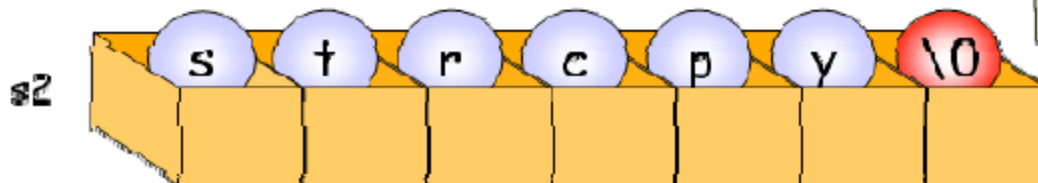
s1이 s2보다 작다

s1이 s2와 같다.

s1이 s2보다 크다.



|| || || || ^



m이 p보다 아스키 코  
드값이 작으므로 음수  
가 반환된다.

# 예제

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

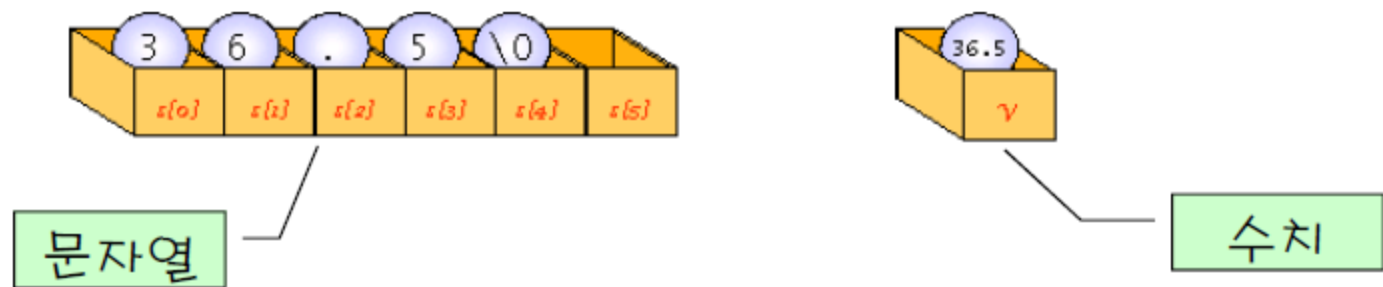
int main( void )
{
    char s1[80];           // 첫번째 단어를 저장할 문자배열
    char s2[80];           // 두번째 단어를 저장할 문자배열
    int result;

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

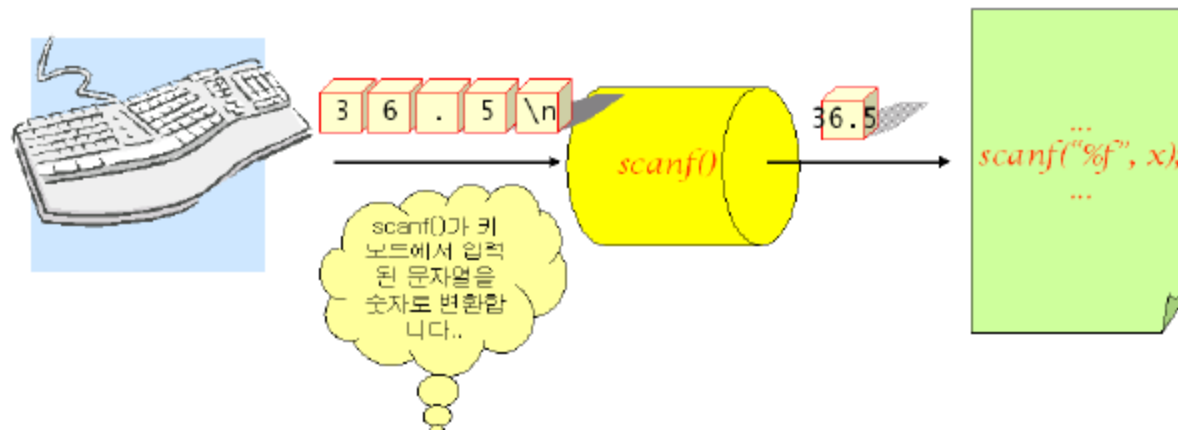
    result = strcmp(s1, s2);
    if( result < 0 )
        printf("%s가 %s보다 앞에 있습니다.\n", s1, s2);
    else if( result == 0 )
        printf("%s가 %s와 같습니다.\n", s1, s2);
    else
        printf("%s가 %s보다 뒤에 있습니다.\n", s1, s2);
    return 0;
}
```

# 문자열 수치 변환

- 문자열과 수치



- `scanf()` 함수는 문자열을 수치로 변환한다.



# 문자열을 수치로 변환하는 전용 함수

- `#include <stdlib.h>`

함수	설명
<code>int atoi( const char *str );</code>	<code>str</code> 을 <code>int</code> 형으로 변환한다.
<code>long atol( const char *str);</code>	<code>str</code> 을 <code>long</code> 형으로 바꾼다
<code>double atof( const char *str );</code>	<code>str</code> 을 <code>double</code> 형으로 변환한다.

```
char ch[10];  
strcpy(ch, "2000");
```

```
int a = atoi(ch);
```