

**Robotics**



대경혁신인재양성프로젝트  
**HuStar**

# C프로그래밍4

**Robotics**



대경혁신인재양성프로젝트



**HuStar**

**정렬 (Sort)**

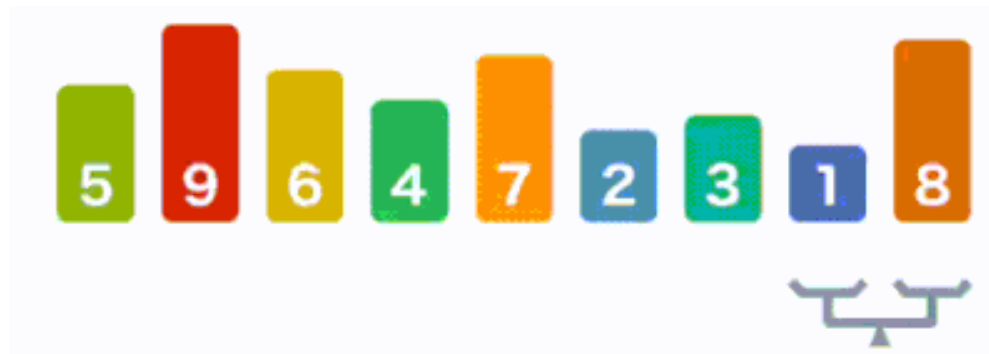
# 정렬(sort)

- 정렬
  - 데이터를 특정 규칙에 따라 재배열 하는 것
  - 정렬을 위한 알고리즘
    - Bubble sort(버블 정렬)
    - Selection sort(선택 정렬)
    - Insertion sort(삽입 정렬)
    - ...

# 버블정렬

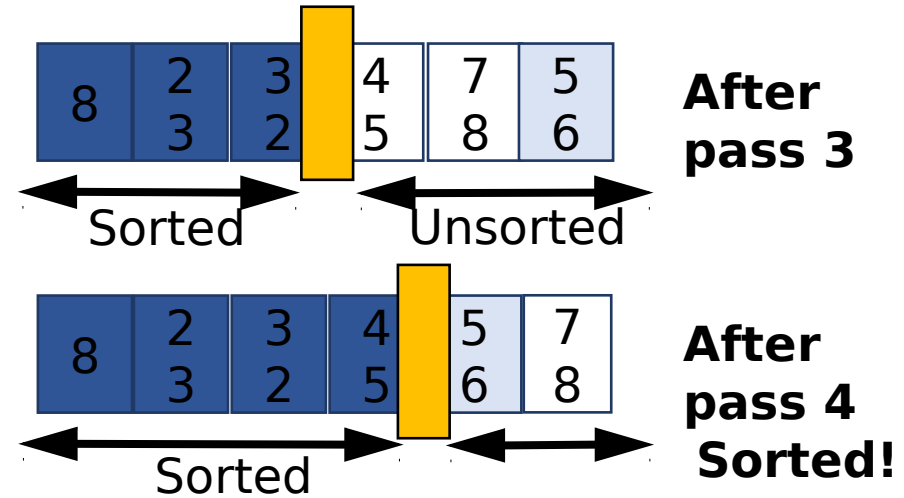
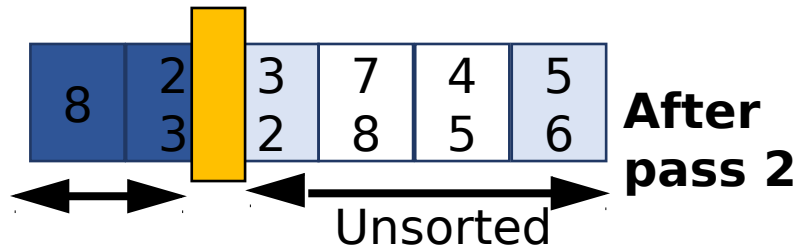
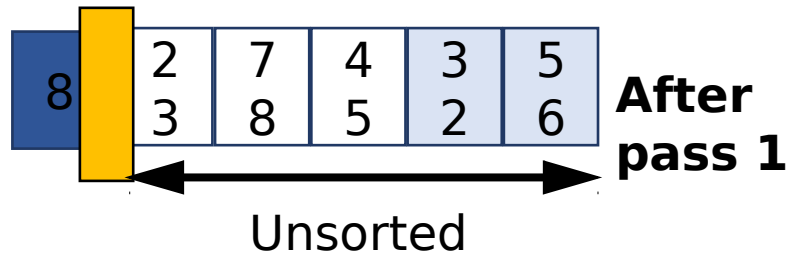
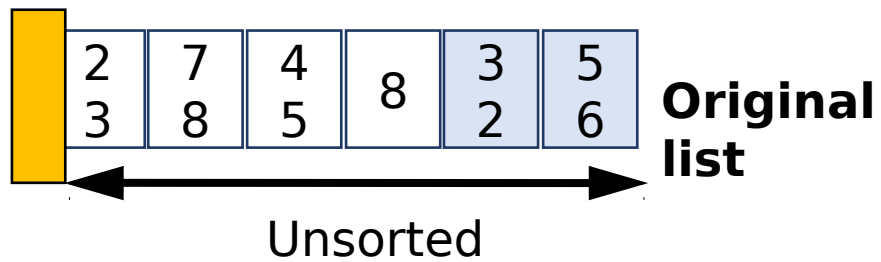
- 버블정렬의 기본 개념

- 인접하는 두 항을 비교하여 뒷 항이 앞 항보다 작으면 두항을 교환
- 원리는 간단하나 교환 횟수가 많은 단점
- 시간 복잡도:  $O(n^2)$



# 버블정렬 알고리즘

```
int a[] = {23, 78, 45, 8, 32, 56};
```



# 버블 정렬 예시

```
#include <stdio.h>
#define N 6

int main(){
    int a[]={80, 41, 35, 90, 40, 20};
    int temp;

    for(int i=0; i<N-1; i++){
        for(int j=N-1; j>=i; j--){
            if(a[j-1]>a[j]){
                temp=a[j];
                a[j]=a[j-1];
                a[j-1]=temp;
            }
        }
    }

    for(int i=0; i<N; i++)
        printf("%d ", a[i]);

    return 0;
}
```

20 35 40 41 80 90  
\_ \_ \_ \_ \_

# 버블 정렬 실습1

- 앞에서 제시된 버블 정렬을 수정
  - 배열의 크기와 데이터를 사용자로부터 입력 받을 것
  - 정렬을 void bubble() 함수에서 하도록 프로그램을 수정할 것

```
input array size:5
input numbers:12 35 23 543 0
0 12 23 35 543
```

# 버블 정렬 실습2

- 버블 정렬 알고리즘을 수정
  - 오름차순과 내림차순을 사용자가 설정할 수 있도록 수정
  - 사용자가 입력에서 오름차순(i) 내림차순(d)을 정할 수 있음
    - Ex) i를 입력하면 오름차순으로 정렬, d를 입력하면 내림차순으로 정렬
    - 배열의 크기와 함께 입력되도록 할 것

```
input array size and increment/decrement(5i):5i 오름차순
input numbers:12 54 23 12 0
0 12 12 23 54
```

```
input array size and increment/decrement(5i):5d 내림차순
input numbers:45 67 12 23 12
67 45 23 12 12
```



**Robotics**



대경혁신인재양성프로젝트



**HuStar**

**문자열(String)**

# 문자열 출력 예시

- 문자열 리터럴(literal) 출력

“(큰따옴표)로 묶여 있는 경우  
하나의 문자열을 나타냄

```
printf( "%s", "me\tand\nyou" );
```

me and  
you

연속된 문자열을 알리는 \

```
printf("Hello!, \n  
everybody");
```

Hello, everybody

문자배열과 문자열

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

```
char str  
[11];
```

G	o	o	d					y	\0	?	?
---	---	---	---	--	--	--	--	---	----	---	---

배열의 일부분  
문자열의 일부분은 아님

# 문자열의 저장- 배열형

- 문자열 저장
  - char 형 배열을 사용
  - 문자열의 종료는 NULL 문자로 나타냄
  - 문자열 저장은 문자의 배열을 사용하기 때문에 문자 배열을 문자열 변수라고 부를 수 있음

```
[Ex] char word[100]; /* word 배열은 99개의 문자를 저장할 수 있음*/
```

```
[Ex] word[0] = 'a';      /* 배열명은 배열의 첫번째 주소를 가리킨다*/  
    word[1] = 'b';  
    word[2] = 'c';  
    word[3] = '\0';      /* 종료 문자*/
```

```
[Ex] printf("%s", word); /* word배열에 저장된 문자열 출력*/
```

# 문자열 초기화- 배열형

- 문자열을 초기화 하는 방법

문자배열의 크기가 정해진 경우

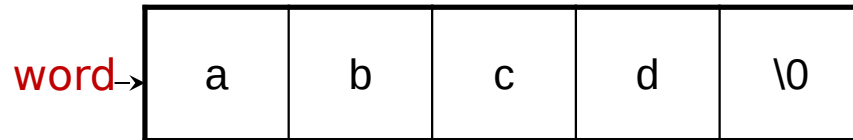
```
char word[5] = "abcd"; //자동으로 NULL이 입력되면서 초기화
```

```
char word[5] = {'a', 'b', 'c', 'd', '\0'};
```

문자배열의 크기가 없는 경우

```
char word[] = {'a', 'b', 'c', 'd', '\0'}; //배열의 크기가 자동으로 할당됨
```

```
char word[] = "abcd"; //NULL 문자가 없는 경우 자동으로  
추가되면서 배열이 할당됨
```

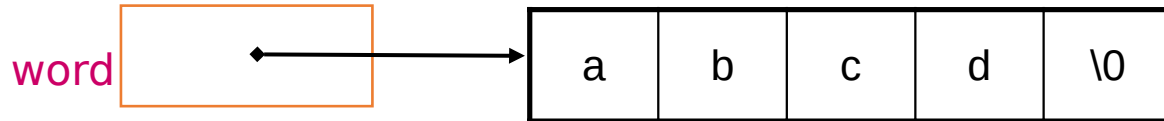


배열의 크기: 5  
문자 개수( 4 ) + null 문자('\0')

# 문자열 저장-포인터

- 포인터를 이용하여 문자열 저장
  - Char 형 pointer 변수를 이용하여 문자열 저장

```
[Ex] char *word = "abcd"; /* 포인터 변수 word*/
```



Word변수는 “abcd”리터럴 문자열을 가리킨다.

```
[Ex] char *word; /* 초기화 없는 문자 포인터형 변수*/
```



아무 의미 없는 값(dummy)을 가지고 있음

# 문자 포인터 변수의 장점

배열을 이용한 경우

```
char s[8] = "Hello  
!";
```

```
char s[9] = "Hello  
!";
```

```
char s[7] = "Hello  
!";
```

s

H	e	l	l	o		!	\0
---	---	---	---	---	--	---	----

s

H	e	l	l	o		!	\0	\0
---	---	---	---	---	--	---	----	----

s

H	e	l	l	o		!
---	---	---	---	---	--	---

배열의 크기가 부족할 수 있음

포인터를 이용한 경우

```
char *s  
= "Hello !";
```

s

--

↓

H	e	l	l	o		!	\0
---	---	---	---	---	--	---	----

메모리에 리터럴 문자열(Hello !)을  
저장하고 해당 주소로 포인터 변수 **s**  
를 할당

# 문자 배열과 문자 포인터의 비교

- 문자 배열과 문자 포인터의 차이점
  - 배열
    - 배열명을 l-value와 같은 형태로 사용할 수 없음
    - 배열명은 상수로 지정된 메모리 주소만 가리킴
  - 포인터
    - 포인터명은 l-value와 같은 형태로 사용할 수 있음
    - 포인터명은 변수로 선언되었기 때문

```
[Ex] char word[] = "abc";  
      word = "def";          /* Error */
```

```
[Ex] char *wp = "abc";  
      wp = "def";           /* OK */
```

```
[Ex] char *p_word = "abc";  
      printf("%u\n", p_word);  
      p_word = "def"; /* ok */  
      printf("%u\n", p_word);
```

P\_word는  
문자열 "abc"의  
주소를 가지고  
있음

P\_word는  
새로운 문자열  
"def"의 위치로  
메모리 주소 변경

4325412  
4325404

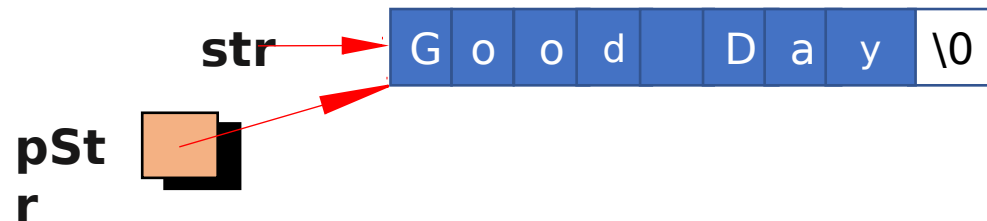
# 문자 배열과 문자 포인터의 비교

- 문자 배열과 문자 포인터의 차이점
  - 배열
    - 배열의 각 요소를 수정할 수 있음
  - 포인터
    - 문자열의 각 요소를 수정할 수 없음, 해당 문자열은 상수 형인 리터럴로 표현되었기 때문

```
[Ex]    char *p = "abc", q[] = "hi";  
        q[0] = 'H';           /* OK */  
        p[0] = 'A';           /* Error */  
        p[1] = 'B';           /* Error */  
        p[2] = 'C';           /* Error */  
        p[3] = '\0';          /* Error */
```

[Ex] //문자 배열을 포인터로 할당

```
char str[9] = "Good Day";  
char *pStr = NULL;  
pStr = str;
```





# 문자열 복사 실습

```
#include <stdio.h>
#define SIZE 10

(          ) //함수선언
(          ) //함수선언

int main( void ){
    char string1[ SIZE ];
    char *string2 = "Hello";
    char string3[ SIZE ];
    char string4[] = "Good Bye";

    copy1( string1, string2 );
    printf( "string1 = %s\
n", string1 );

    copy2( string3, string4 );
    printf( "string3 = %s\
n", string3 );
}
```

```
//array를 이용한 문자열 복사 s2를 s1으로
void copy1( char * s1, const char * s2
) {

}

}
```

```
//pointer를 이용한 문자열 복사 s2를 s1으로
void copy2( char *s1, const char *s2
){

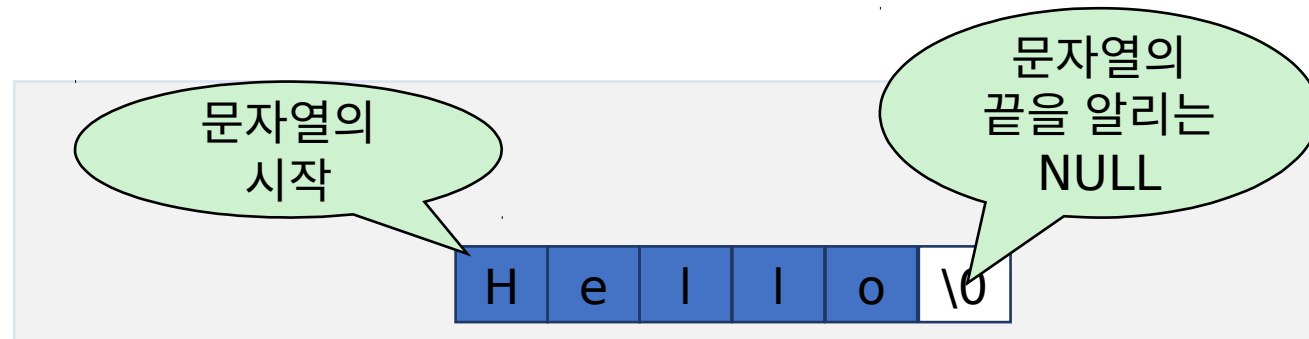
}

}
```

# 문자열 입출력

- %S

- 문자열 입출력을 위한 서식지정자
- 서식을 지정하여 입출력 하는 함수 scanf()와 printf()에서
- scanf()를 이용하여 문자열을 입력 받으면  
NULL(\0)이 자동으로 문자열의 끝에 추가



# 문자열 출력

- printf()를 이용한 출력
  - %s 서식 지정자를 사용
  - NULL 문자를 제외하고 출력됨

```
int printf(char*c, argument-list);    /* function prototype */
```

Return Value : int

- no. of chars written //출력 성공
- EOF(-1) // 출력 실패

[Ex]

```
char p[5]="hello"; //배열의 크기 부족 NULL 저장 안됨
printf("%u %s", p, p); //출력 시 오류: 문자열의 끝을 알 수 없음

printf("%d", printf("hello"));
```

hello 5

# 문자열 출력 예제

```
#include <stdio.h>

int main(){
    int nchars;
    char p[ ] = "Hello! the world";
    char *q = "Hi, everybody";

    nchars = printf("%s", p);
    printf("\nnum of chars=%d\n", nchars);
    printf("%.5s\n", p);           //5개의 문자만 출력
    printf("%s\n", &p[7]);         //p[7]의 위치에서 부터 문자열 출력
    printf("%s\n", &q[2]);

    return 0;
}
```

```
Hello! the world
num of chars=16
Hello
the world
, everybody
```

# 문자열 출력

- puts()
  - 문자열을 출력하는 함수
  - 문자열의 이름(배열명, 포인터변수)만 인수로 받음(인수 1개 필요)
    - 서식 지정자를 사용하지 않음( fast and simple)
  - 문자열 출력 뒤 자동으로 줄 바꿈

```
int puts(const char *str);      /*function prototype */
```

Return Value : int

- non-negative value
- EOF(-1)            //출력 오류의 경우

```
[Ex]    char p[ ] = "Hi !!";  
        puts(p);  
        puts("Hello!!");  
        puts(&p[3]);
```

Hi !!  
Hello!!  
!!

# 문자열 출력

- fputs()
  - 문자열 출력 함수
  - 출력을 위한 대상을 지정할 수 있음
  - puts함수는 호출되면 문자열 출력 후 자동으로 줄 바꿈이 이루어짐  
fputs함수는 호출되면 문자열 출력 후 자동으로 줄 바꿈이 이루어지지 않음

```
int fputs(const char *str, FILE * stream); /*function prototype */
```

Return Value : int

- non-negative value
- EOF(-1)                      //출력 오류의 경우

# 문자열 출력 예제

```
#include <stdio.h>

int main(){
    char *str="Simple String";
    printf("1. puts 실험:-----\n");
    puts(str);
    puts("간단한 문자열 출력");

    printf("2.fputs 실험 ----- \n");
    fputs(str, stdout); printf("\n");
    fputs("간단한 문자열 출력2",stdout); printf("\n");

    printf("3. end test-----\n");

    return 0;
}
```

```
1. puts 실험:-----
Simple String
간단한 문자열 출력
2.fputs 실험 -----
Simple String
간단한 문자열 출력2
3. end test-----
```

# 문자열 입력

- scanf()를 이용한 문자열 입력
  - %s(서식지정자) 사용
  - &는 사용하지 않음(배열명은 포인터 이기 때문)
  - 공백이 있는 문자열까지 한번에 읽음
  - NULL문자는 자동으로 문자열의 마지막에 추가됨

```
int scanf(char *format, argument_list);
```

Return Value : int

- no. of successfully matched and input items
- 0 if not



# 문자열 입력

## • 입력 예시

입력  
문자열

Handong Univ. ↵

```
[Ex] char name[80];  
  
scanf("%s", name);           /* name <= Handong */  
scanf("%s", &name[0]);      /* OK */
```

입력  
문자열

C-Program is ↵

```
[Ex] char name[80];  
  
scanf("%3s", name); /* name <= C-P */  
scanf("%8s", name); /* name <= rogram */
```

서식지정자 활용 3개의 문자입력

공백전까지 문자를 읽음

# 문자열 입력

- 저장 공간(배열크기)는 긴 문장을 저장할 수 있을 정도로 커야 함

[Ex]

```
char a[4], b[4];
```

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

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

```
printf("%s\n%s", a, b);
```

12345 abcdef↵

입력 문자열

Read "12345" and  
store them into *b*

Read "abcdef" and  
store them into *a*

/\* output until '\0' \*/

abcdef  
ef

a	
	a
	b
	c
b	d
	<del>1</del> e
	<del>2</del> f
	<del>3</del> \0
	4
	5
	\0

# 문자열 입력

- gets()
  - 문자열을 입력하기 위한 함수
  - 한 줄 단위(\n)로 문자열을 읽어 들임(줄바꿈 문자가 있는 곳까지 읽음)
  - \n => \0 줄바꿈 문자를 NULL문자로 변경하여 메모리에 저장
  - gets()함수는 C 표준 (2011) 라이브러리에서 삭제되었으니 사용을 지양할 것

```
char* gets(char *format);
```

Return Value : char pointer

- the address of the string
- NULL if EOF (end-of-file)

입력 문자열

^^Hong Gil-  
Dong↵

```
char name[20];
```

```
scanf("%s", name); // “^^Hong” 입력 됨
```

-----

```
gets(name); // “^^Hong Gil-Dong” 입력됨
```

# 문자열 입력

[Ex]     char data[81], \*P;

```
while( *(p = gets(data)) != NULL) {  
    printf("%s\n", data);  
}
```

while 반복문은 <blank line>  
이 입력될 때까지 반복

while( \*(gets(data)) != '\0')  
같음

gets(p) => !!!실행오류  
gets(/scanf)의 인수는 반드시  
배열명 이어야 함

[Ex]     char data[81], \*P;

```
while( gets(data) != NULL) {  
    printf("%s\n", data);  
}
```

<[ctrl] + z> 가 입력될 때까지  
반복

while( gets(data) != 0)  
같음

# 문자열 입력

- getchar()함수를 이용한 한 줄 단위 문자열 입력

```
int read_line(char str[], int n){  
    char ch;  
    int i=0;  
  
    while((ch=getchar())!='\n')  
        if(i < n) str[i++] = ch;  
  
    str[i]='\0';           /* 문자열 종료 */  
    return i;             /* 입력된 문자의 수를 되돌려 줌 */  
}
```

```
//호출 문장  
char s[10];  
int n;  
  
n = read_line(s,5);  
  
printf("%d  %s\n", n, s);
```

# 문자열에서 문자 확인

/\* 공백 문자 개수를 확인하는 함수. \*/

```
[Ex]   int count_s(const char s[]){           /* 배열 이용*/
        int ct=0, i;
        for(i=0; s[i] != '\0'; i++)
            if(s[i] == ' ') ct++;
        return ct;
    }
```

```
[Ex]   int count_s(const char *s){           /* 포인터 연산 이용*/
        int ct=0;
        for(; *s != '\0'; s++)
            if (*s == ' ') ct++;
        return ct;
    }
```

# 문자열 다루기 예시

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

void convertToUppercase( char *sPtr );           // prototype

int main( void ){
    char string[] = "cHaRaCters and $32.98";    // initialize char array

    printf( "The string before conversion is: %s", string );
    convertToUppercase( string );
    printf( "\nThe string after conversion is: %s\n", string );
}

// convert string to uppercase letters

void convertToUppercase( char *sPtr )
{
    while ( *sPtr != '\0' ) {                    // current character is not '\0'
        *sPtr = toupper( *sPtr );               // convert to uppercase
        ++sPtr;                                  // make sPtr point to the next character
    }
}
```

```
for( ; *sPtr != '\0' ; sPtr++)
    *sPtr = toupper( *sPtr );
```

**Robotics**



문자열 배열



# 문자열 배열

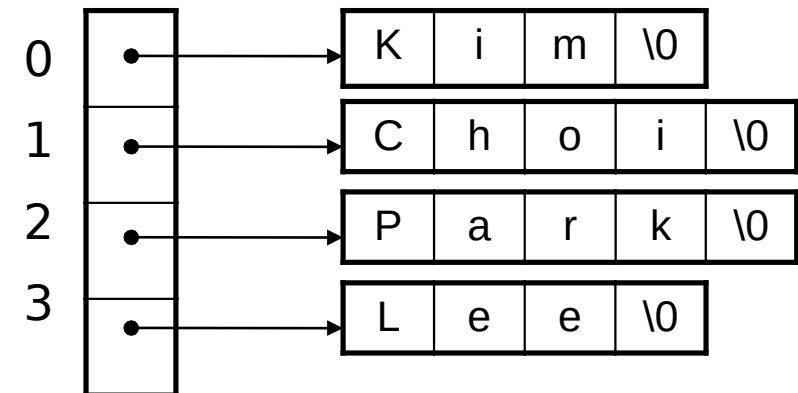
- 2차원 배열을 이용하여 문자열의 리스트 저장

```
char names[][5]={"Kim", "Choi", "Park", "Lee"};
```

	0	1	2	3	4
0	K	i	m	\0	\0
1	C	h	o	i	\0
2	P	a	r	k	\0
3	L	e	e	\0	\0

- 포인터 1차원 배열을 이용한 문자열 리스트

```
char *names[]={"Kim", "Choi", "Park", "Lee"};
```



# 문자열 배열

```
char names[][5]={“Kim”, “Choi”, “Park”, “Lee”};
```

```
for (i=0;i<4;i++)  
    if(*names[i]=='K')  
        names[i]="No";           //error : names[i] 는 상수 문자이기 때문  
.
```

```
for (i=0 ; i<4 ; i++)  
    if(names[i][0]=='K')  
        printf(“%s begins with K\n”, names[i]);
```

# 문자열 배열

```
char *names[]={“Kim”, “Choi”, “Park”, “Lee”};
```

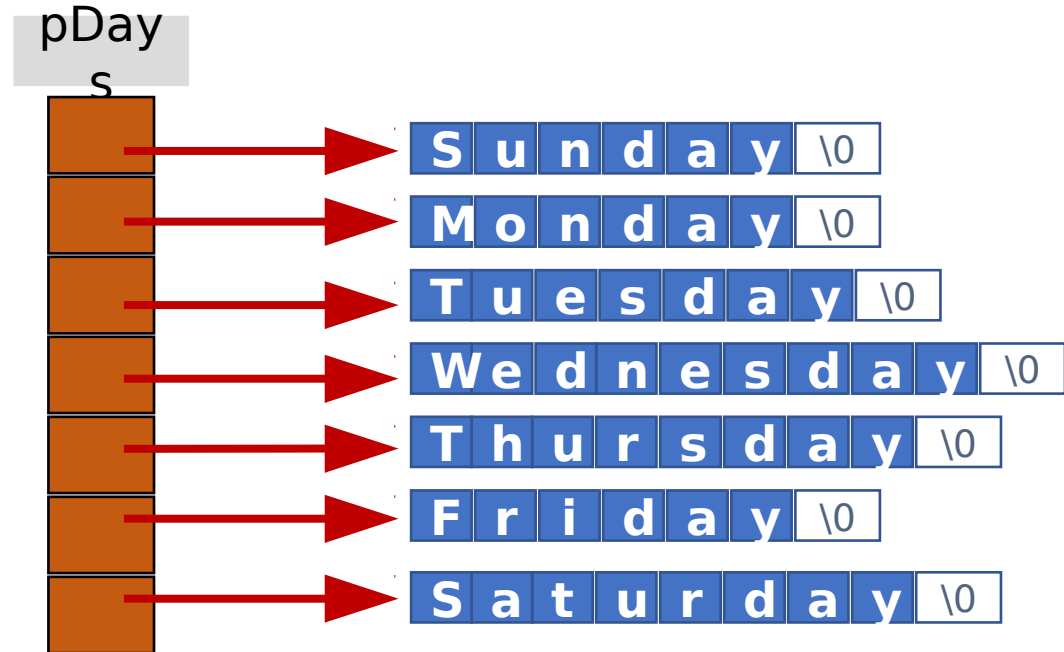
```
for (i=0;i<4;i++)  
    if(*names[i]==‘K’)  
        names[i]=“No”;           //OK : names[i] 포인터 변수이기 때문.
```

```
for (i=0 ; i<4 ; i++)  
    if(names[i][0]==‘K’)  
        printf(“%s begins with K\n”, names[i]);
```

# 문자열 배열: 포인터 배열

```
Char *pDays[7];
```

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



# 문자열 출력 예제

```
int i;  
char names[][5]={“Kang”, “Kim”, “Kong”, “Cho”, “Choi”, “Han”,  
“Hong”, “Lee”};  
for (i=0;i<8;i++)  
    if(*names[i]==‘K’ )  
        puts(names[i]);
```

Kang  
Kim  
Kong

```
int i;  
char names[][5]={“Kang”, “Kim”, “Kong”, “Cho”, “Choi”, “Han”,  
“Hong”, “Lee”};  
for (i=0;i<8;i++)  
    if(*names[i]==‘K’ )  
        puts(names[i]+i);
```

Kang  
im  
ng

# 프로그램 실행 시 main() 인수 전달

- main()

인자가 없는 경우(void)

```
[Ex]  
/* Without command-line arguments*/  
int main()  
{  
    :  
}
```

2개의 인자

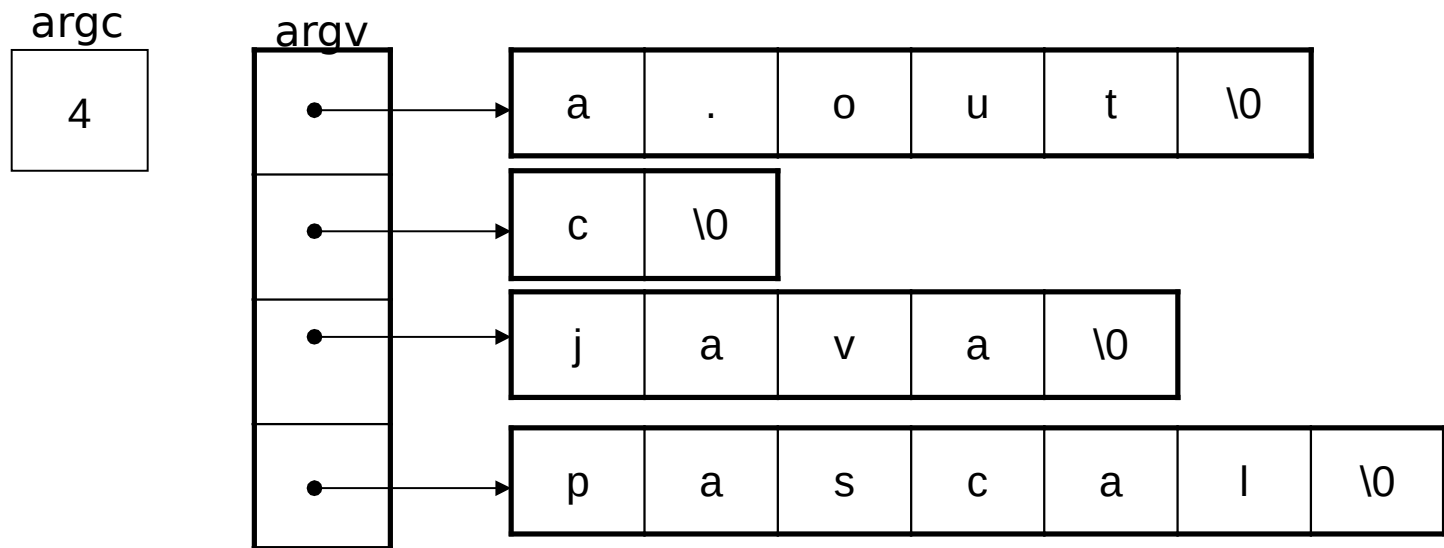
```
[Ex]  
/* 프로그램 실행 시 인수 전달*/  
int main(int argc, char *argv[])  
{  
    :  
}
```

# 프로그램 실행 시 main() 인수 전달

[Ex] a.out c java pascal

//a.out 은 실행 프로그램 파일명.

argc = 4  
argv[0] => "a.out"  
argv[1] => "c"  
argv[2] => "java"  
argv[3] => "pascal"

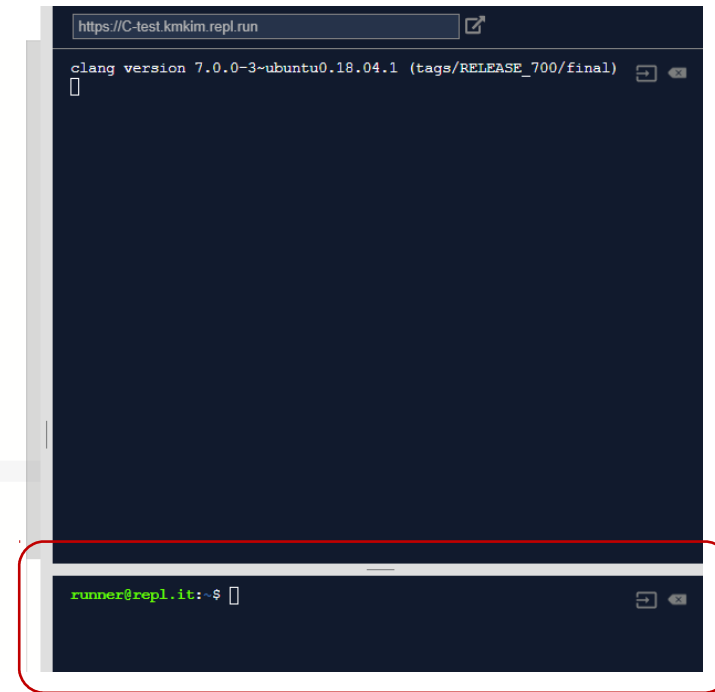


*argv* 배열은 몇 개의 요소를 가짐. 첫번째 요소는 실행파일 명을 가리킴 (a.out). 이것은 프로그램에서 자동으로 지원되는 것임.

# 프로그램 실행 시 main() 인수 전달

- Repl.it에서 실습
- 코드에 클릭한 상태에서 F1 key 누른 후, 나타나는 창에 open shell 이라고 치고 enter 하면 다음과 같이 실행 창 아래에 shell 창이 나타난다

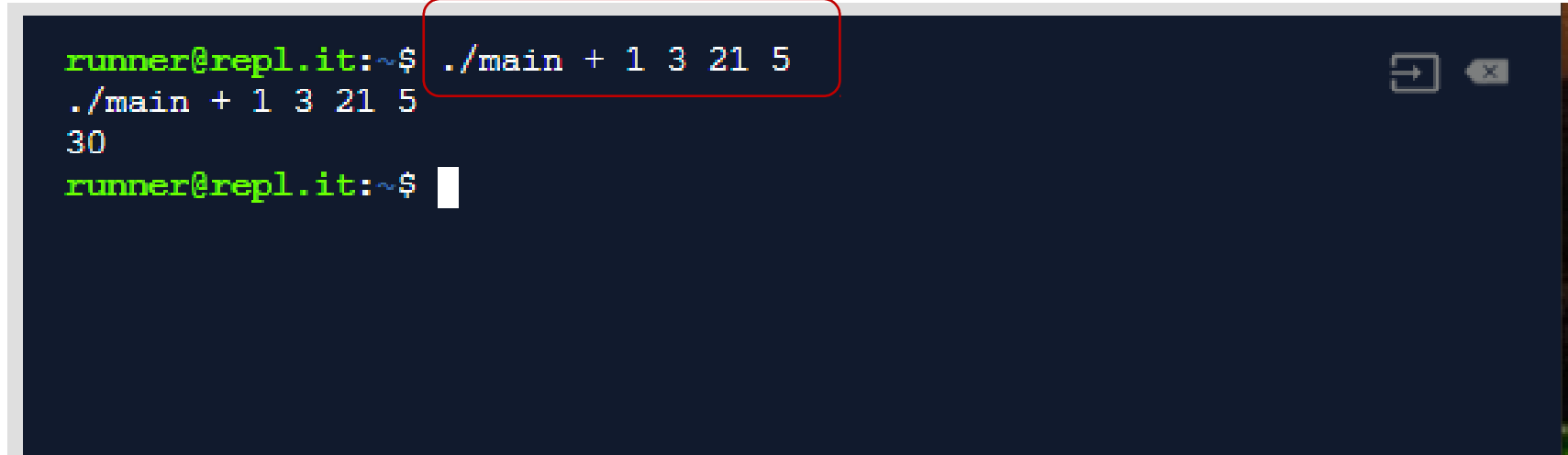
```
main.c  history
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main (int argc, char *argv[]){
5
6      int i, s;
7      switch(* argv[1]){
8          case '+': s=0;
9              for(i=2;i<argc;i++) s+=atoi(argv[i]);
10             break;
11         // case '-': s=atoi(argv[2]) - atoi(argv[3]);
12         //         break;
13         case 'x': s=1;
14             for(i=2;i<argc ; i++) s*=atoi(argv[i]);
15             break;
16         // case '/' : s=atoi(argv[2]) / atoi(argv[3]);
17         //         break;
18     }
19
20     printf("%d\n", s);
21
22 }
23
24
```





# 프로그램 실행 시 main() 인수 전달

```
runner@repl.it:~$ ./main + 1 3 21 5
./main + 1 3 21 5
30
runner@repl.it:~$
```



# 프로그램 실행 시 main() 인수 전달

```
#include <stdio.h>

int main (int argc, char * argv[]){
    int count, i, s=0;
    printf("#=%d, argv[0]=%s\n", argc, argv[0]);

    if(argc>1)
        for(count=1; count<argc; count++)
            printf("argv[%d] = %s\n", count, argv[count]);
    else
        puts("No comand line arguments");

    for(i=2;i<argc;i++)
        s+=atoi(argv[i]);

    return 0;
}
```

Point to the program filename

```
runner@repl.it:~$ ./main + 2 3 5 7
#=6, argv[0]=./main
argv[1] = +
argv[2] = 2
argv[3] = 3
argv[4] = 5
argv[5] = 7
17
```

# atoi()

- 문자열을 정수로 변환

```
int atoi(const char *str);    /* function prototype */
```

```
printf("%d %d %d ", atoi("a"), atoi("100.78"), atoi("12A"));
```

0 100 12

# atof()

- 문자열을 실수(double)로 변환

```
#include <stdio.h>
#include <stdlib.h>          // gets()
#include <string.h>         // strcat()

int main(){  char num1[10], num2[10];
  int i;
  float f;

  printf("Input a number : ");
  gets(num1);
  printf(" Input a number : ");
  gets(num2);
  i=atoi(num1);
  f=atof(num2);
  printf("\n%d, %f\n", i, f);
  printf("strcat(num1,num2) = %s\n", strcat(num1,num2));
  printf("num1 + num2 = %f\n", i+f);
}
```

```
Input a number : 11
Input a number : 12
```

```
11, 12.000000
strcat(num1,num2) = 1112
num1 + num2 = 23.000000
```

# 문자열의 사용법

- 문자열 배열

NULL 문자가 나올 때 까지 문자들의 나열

```
char string[15] = "C PROGRAMMING";
```

C		P	R	O	G	R	A	M	M	I	N	G	\0	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Index를 이용하여 각 문자에 접근가능

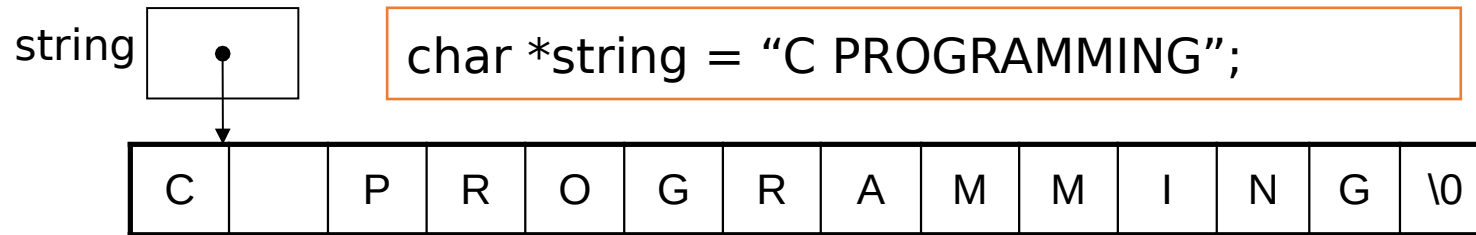
```
string[13] = '!';  
puts(&string[2]);           // print : PROGRAMMING!
```

C		P	R	O	G	R	A	M	M	I	N	G	!	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

# 문자열의 사용법

- 포인터를 이용하여 문자열 나타내기

포인터로 문자열 리터럴을 가리키게 함



문자열의 내용을 수정할 수 없음

```
string+=2;           /* string points to char 'P' */
puts(string);        /* Print : PROGRAMMING */
*string='A';         /* Error */
string[0]='A';       /* Error */
```

# 문자열 처리를 위한 포인터의 사용

- 포인터를 이용하여 개별 요소 출력

[Ex] /\* Print Strings -using pointer\*/

```
int main() {  
    char *p;  
    char *buffer = "Hello!";  
  
    for(p=buffer; *p != '\0'; p++)  
        printf("%c", *p);  
  
    return 0;  
}
```

Equivalent to :  
printf("%s", buffer);

[Ex] /\* Print Strings -using index\*/

```
int main() {  
    char *buffer = "Hello!";  
    int i;  
  
    for(i=0; buffer[i] != '\0'; i++)  
        printf("%c", buffer[i]);  
  
    return 0;  
}
```

Hello!

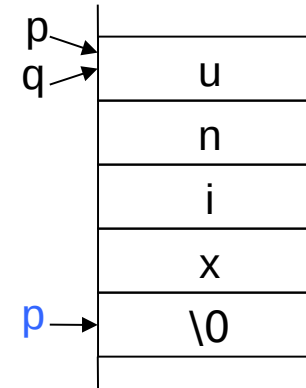
# 문자열 처리를 위한 포인터의 사용

- 역순으로 문자열 출력

[Ex]

```
int main(void) {  
    char *p= "unix", *q;  
    q=p;  
  
    while(*p) p++;  
    while(p>q) {  
        p--;  
        putchar (*p);  
    }  
    putchar('\n');  
  
    return 0;  
}
```

xinu





# 문자열 처리를 위한 포인터의 사용

[Ex] #define MAXLINE 100

```
read_in(char s[ ]) {  
    int c, i=0;  
    while((c = getchar( )) != EOF && c != '\n')  
        s[i++] = c;  
    s[i] = '\0';  
}
```

```
int main() {
```

```
    char line[MAXLINE], *change(char *);
```

```
    printf("\nWhat is your favorite line? ");
```

```
    read_in(line);
```

```
    printf("%s\n\n%s\n",  
        "Here it is after being changed:", change(line));
```

```
    return 0;  
}
```

리턴 및 매개 변수 유형이  
모두 문자 포인터 인 함수 프로토 타입

```
char *change(const char *s) {  
    static char new_string[MAXLINE];  
    char *p = new_string;
```

```
    *p++ = '\t';  
    for( ; *s != '\0'; ++s)  
        if(*s == 'e') *p++ = 'E';  
    else if(*s == ' ') {  
        *p++ = '\n';  
        *p++ = '\t';  
    }  
    else *p++ = *s;  
    *p = '\0';  
    return new_string; }
```

What is your favorite line? she sells sea shells  
Here it is after being changed:

shE  
sElls  
sEa  
shElls

# 문자열 관련 함수- strlen()

- strlen()
  - 문자의 개수를 확인
  - 문자열의 길이를 돌려줌
  - 문자열의 길이에는 NULL문자는 포함되지 않음

```
size_t strlen(const char *s1);           // size_t defined as unsigned int
```

```
size_t strlen(const char *s){  
    size_t n ;  
    for(n=0; *s != '\0'; s++) n++; //counts the number of characters of string  
    return n;  
}
```

# 문자열 관련 함수- strlen() 예시

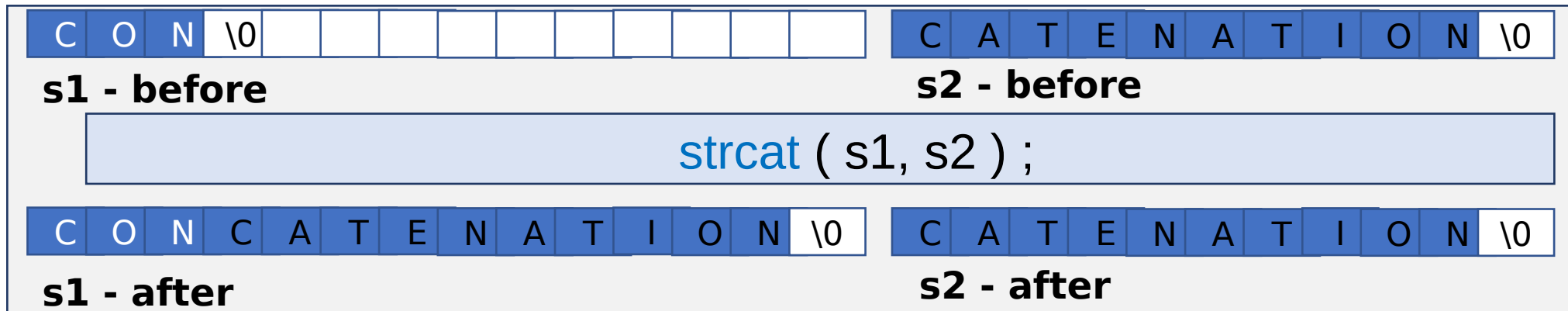
```
char str1[100] = "handong", *str2 = "handong";  
if(str1 == str2)          /* false */  
    printf("same address\n");  
if(strcmp(str1, str2) == 0) /* true */  
    printf("equal\n");  
printf("length=%d\n", strlen(str1));
```

equal  
length = 7

# 문자열 관련 함수- strcat()

- strcat()
  - 두개의 문자열을 합치는 함수
  - 두번째 문자열을 복사하여 첫번째 문자열의 뒤에 연결
  - 첫번째 문자열이 새로운 버전으로 변경됨
  - 첫번째 인수를 되돌려 줌

```
char *strcat(char *s1, const char *s2);
```



## String Concatenate

# 문자열 관련 함수- strcat() 예시

[Ex] //Append string s2 to string s1

```
char *strcat (char *s1, const char * s2){
```

```
    char *p;
```

```
    p=s1;
```

```
    while(*p != '\0') p++;
```

```
    while(*s2 != '\0'){
```

```
        *p=*s2; p++; s2++;
```

```
    }
```

```
    *p = '\0' ;
```

```
    return s1;
```

```
}
```

while(\*p) p++;

while(\*p++  
+=\*s2++);

# 문자열 관련 함수-strcpy()

- strcpy()
  - NULL문자를 포함하여 문자열을 복사
  - String s2의 내용을 string s1으로 할당
  - String s1은 반드시 배열 변수이어야 함
  - String s2는 문자 배열 변수 이거나 문자열 상수
  - S1의 값을 되돌려 줌

```
char *strcpy(char *s1, const char *s2);
```

# 문자열 관련 함수-strcpy()

- Strcpy()와 관련한 프로그래밍 오류

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char str1[5]="1234"; //size is not enough
    char str2[5]="abcd";

    printf("%s, %s\n", str1, str2);
    strcat(str1, "5678");
    printf("%s, %s\n", str1, str2);
}
```

str1

str2

1234, abcd  
12345678, 678

1	
2	
3	
4	
5	\0
6	a
7	b
8	c
9	d
10	\0

# 문자열 관련 함수- strncpy()

- strncpy()
  - 문자열 복사 함수
  - 문자열 s2의 최대 n개의 문자를 배열 s1에 복사

```
char *strncpy(char *s1, const char *s2, size_t n);
```

```
char s1[] = "Happy ";  
char s2[] = "New Year ";  
char s3[ 40 ] = "";  
printf( "s1 = %s\ns2 = %s\n", s1, s2 );  
printf( "strcpy( s1, s2 ) = %s\n", strcpy( s1, s2 ) );  
printf( "strncpy( s3, s1, 3 ) = %s\n", strncpy( s3, s1, 3 ) );
```

```
s1 = Happy  
s2 = New Year  
strcpy( s1, s2 ) = New Year  
strncpy( s3, s1, 3 ) = New
```



# 문자열 관련 함수-strchr()

- strchr()
  - 문자열 검색 함수
  - 문자열 내에서 c1에 입력된 문자가 처음 나타나는 위치를 찾음
  - 문자열에서 c1에 입력된 문자가 처음 나타나는 주소를 반환
    - 검색결과가 없는 경우 NULL 포인터 반환

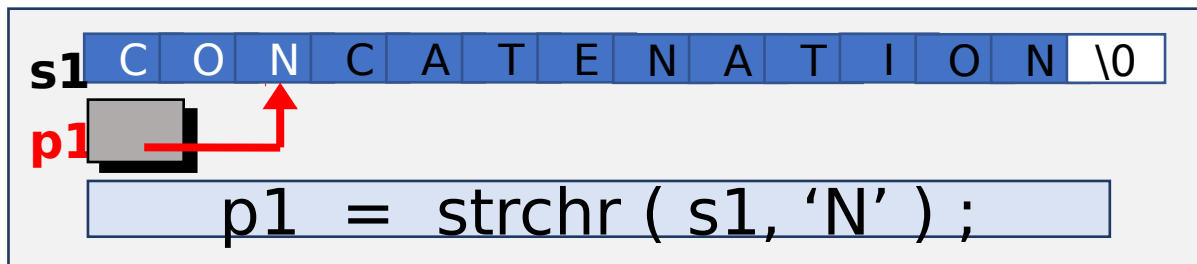
```
char* strchr(const char *s1, char c1);
```

# 문자열 관련 함수- strrchr()

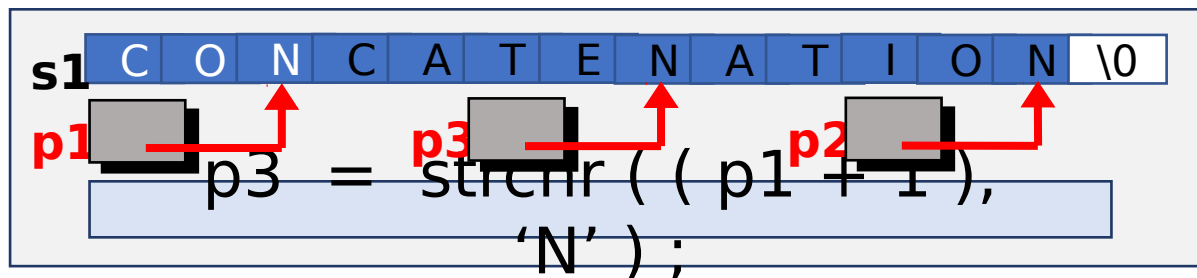
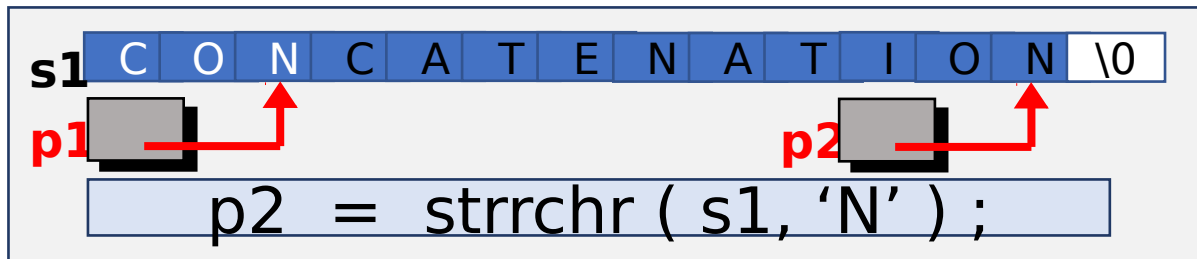
- strrchr()

- 문자열에서 문자를 검색하되 가장 마지막으로 나타나는 위치를 찾음

```
char* strrchr(const char *s1, char c1);
```



문자열 찾는 함수 비교



# 문자열 관련 함수- strchr(), strrchr() 예제

```
#include <stdio.h>
#include <string.h>
int main() {
    char *s1 = "Happy New Year";
    char *p, ch='a';

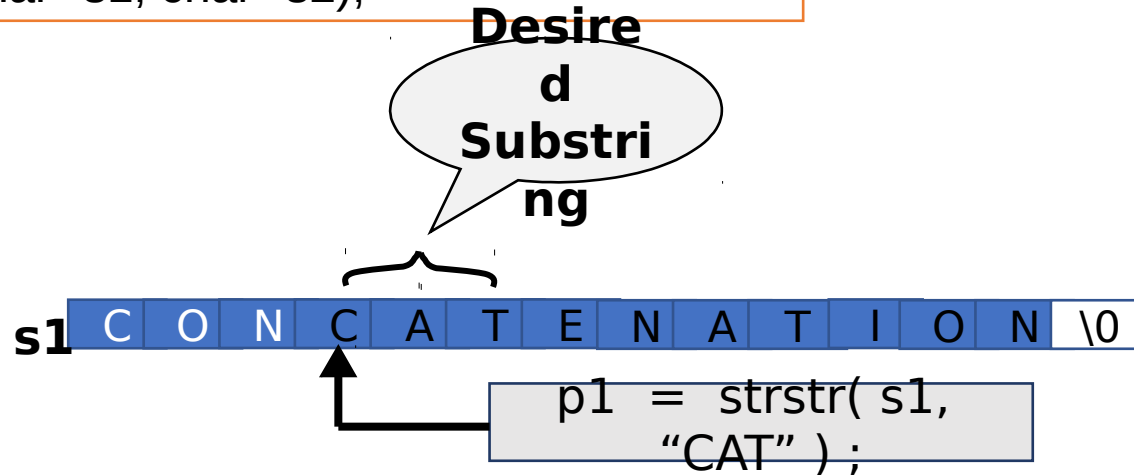
    p=strchr(s1, ch);
    printf("%c-- %s\n", *p, p);
    p=strrchr(s1, ch);
    printf("%c-- %s\n", *p, p);
}
```

a-- appy New Year  
a-- ar

# 문자열 관련 함수- strstr()

- strstr()
  - 문자열 안에서 문자열을 검색
  - strstr(대상문자열, 검색할문자열);
  - 문자열을 찾았으면 문자열로 시작하는 문자열의 포인터를 반환, 문자열이 없으면 NULL을 반환

```
char* strstr(const char *s1, char* s2);
```



# 문자열 관련 함수- strstr() 예제

```
#include <stdio.h>
#include <string.h>
int main() {
    char *s1 = "Happy New Year";
    char *p, ch='a';

    p=strstr(s1, "py");
    printf("%c-- %s", *p, p); }
```

p-- py New Year

# 문자열 관련 함수 예제 - 모든단어 위치 찾기

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

int main(int argc, char *argv[])
{
    char *str = "About C Programming\nProcedural Language - Instructions in a C program are executed step by

    if (argc ≤ 1)
    {
        printf("usage: %s [word]\n", argv[0]);
        return 0;
    }
    char *word = argv[1];
    puts(str);

    char *p = str;
    while ((p = strstr(p, word)) ≠ NULL)
    {
        int i = p - str;
        printf("%d, ", i);
        p += strlen(word);
    }
    printf("\n");

    return 0;
}
```