

10. 배열과 포인터 복습

Autumn 2019

Today

1. 지난 시간 복습 - 포인터

2. 실습

[복습] 포인터 변수

- 변수의 주소를 저장하는 변수
- 포인터 변수 생성
 - 데이터형에 * 을 붙여서 생성
 - int*, float*, double*, char*
- 다음 코드에 int 형 포인터 변수 p, q를 선언해서 각각 a, b의 주소를 저장한 뒤 다음의 scanf 와 printf 문을 바꿔보기

```
#include <stdio.h>

int main()
{
    int a, b;

    puts("두 정수를 입력하세요.");
    scanf("%d %d", &a, &b);
    printf("두 수의 합은 %d입니다.\n", a + b);
}
```

```
#include <stdio.h>

int main(){
    int a, b;
    int *p, *q;

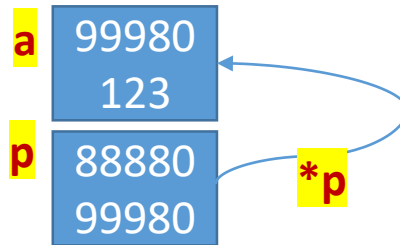
    p = &a;
    q = &b;

    puts("두 정수를 입력하세요.");
    scanf("%d %d", p, q);
    printf("두 수의 합은 %d입니다. \n", *p + *q);
}
```

[복습] 포인터 변수

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

```
int main(){  
    int a = 123;  
    int *p;
```



```
    p = &a;    ← 포인터 변수 p 에 a의 주소를 할당한다.
```

```
    *p = 1;    ← 포인터 변수 p 가 가리키는 값을 1로 할당한다.
```

```
    p = 1;     ← ???
```

```
}
```

[복습] 배열과 포인터

- 배열의 변수명은 주소를 나타낸다!!

```
int i = 0;
int num1[4] = { 10, 20, 30, 40 };
int num2[4] = { 11, 22, 33, 44 };
int *p1, *p2;
p1 = &num1;
p2 = &num2;

for (i=0; i<4; i++)
    printf("%d, %d \n", p1+i, p2+i);

for (i = 0; i < 4; i++)
    printf("%d, %d \n", *(p1 + i), *(p2 + i));
```

```
2018965672, 2018965720
2018965676, 2018965724
2018965680, 2018965728
2018965684, 2018965732
10, 11
20, 22
30, 33
40, 44
```

num1 == &num1 == &num1[0]

p1 = &num1

***p1 = num1**

5672	5676	5680	5684	5688	5692
10	11	12	13		
num1[0]	num1[1]	num1[2]	num1[3]		

[복습] 문자열과 포인터

```
int main(){
    char s[8] = "Basic-C";
    char* p;
    p = &s;
    printf("%s\n", p);

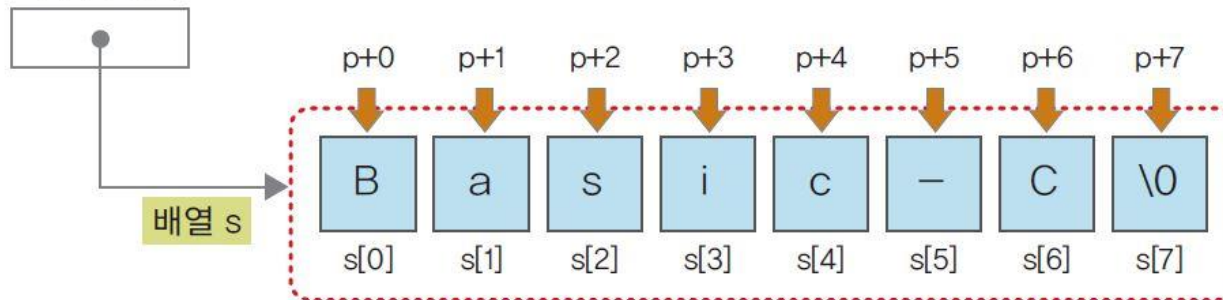
    printf("&s[3]: %s \n", &s[3]);
    printf("p+3: %s \n", p+3);

    printf("s[3]: %c \n", s[3]);
    printf("*(p+3): %c \n", *(p + 3));
}
```

p = s 와 동일함,

배열의 변수명은 주소!!

포인터 변수 p



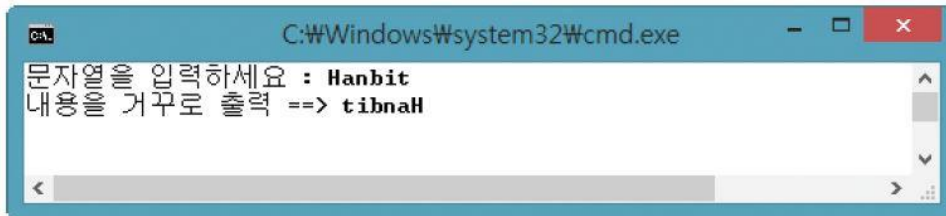
```
Basic-C
&s[3]: ic-C
p+3: ic-C
s[3]: i
*(p+3): i
```

그림 9-15 [응용 9-9]의 변수와 포인터의 관계

[실습20, 21]

예제 설명 문자열 배열을 이용해서 입력받은 문자열을 반대 순서로 출력하는 프로그램이다.

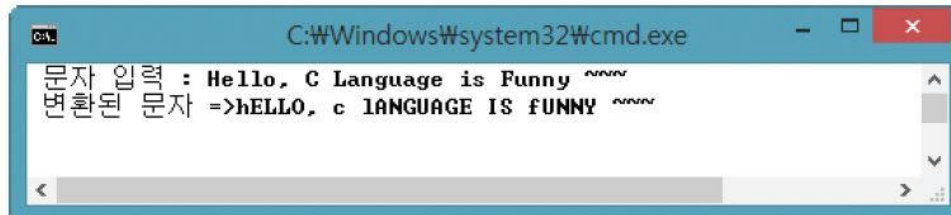
실행 결과



```
C:\Windows\system32\cmd.exe
문자열을 입력하세요 : Hanbit
내용을 거꾸로 출력 ==> tibnaH
```

예제 설명 입력된 문자열이 대문자이면 소문자로, 소문자이면 대문자로 변환하고 그 외의 문자는 그대로 출력하는 프로그램이다.

실행 결과

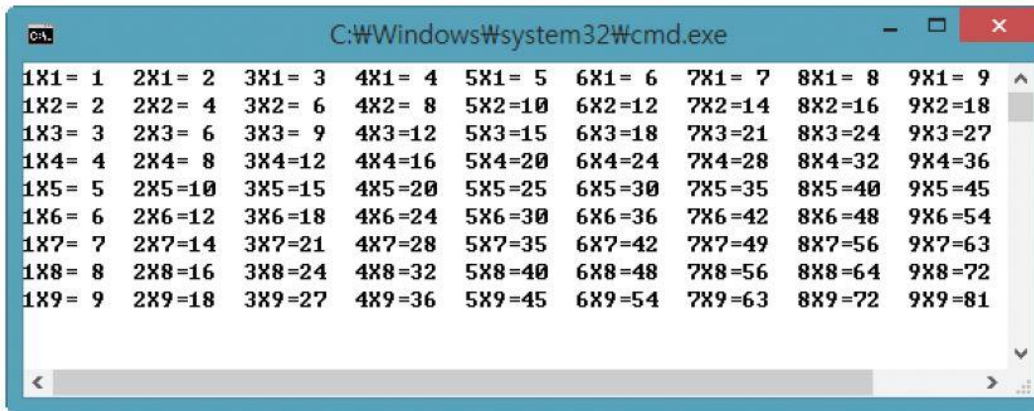


```
C:\Windows\system32\cmd.exe
문자 입력 : Hello, C Language is Funny ~~~~
변환된 문자 =>hELLO, c LANGUAGE IS FUNNY ~~~~
```

[실습22, 23]

예제 설명 구구단의 결과를 2차원 배열에 저장한 후 출력하는 프로그램이다.

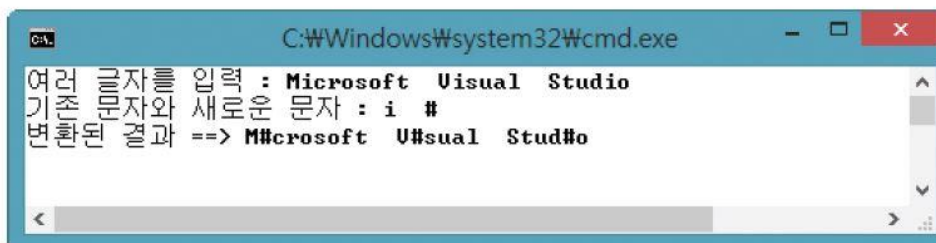
실행 결과



```
C:\Windows\system32\cmd.exe
1X1= 1  2X1= 2  3X1= 3  4X1= 4  5X1= 5  6X1= 6  7X1= 7  8X1= 8  9X1= 9
1X2= 2  2X2= 4  3X2= 6  4X2= 8  5X2=10  6X2=12  7X2=14  8X2=16  9X2=18
1X3= 3  2X3= 6  3X3= 9  4X3=12  5X3=15  6X3=18  7X3=21  8X3=24  9X3=27
1X4= 4  2X4= 8  3X4=12  4X4=16  5X4=20  6X4=24  7X4=28  8X4=32  9X4=36
1X5= 5  2X5=10  3X5=15  4X5=20  5X5=25  6X5=30  7X5=35  8X5=40  9X5=45
1X6= 6  2X6=12  3X6=18  4X6=24  5X6=30  6X6=36  7X6=42  8X6=48  9X6=54
1X7= 7  2X7=14  3X7=21  4X7=28  5X7=35  6X7=42  7X7=49  8X7=56  9X7=63
1X8= 8  2X8=16  3X8=24  4X8=32  5X8=40  6X8=48  7X8=56  8X8=64  9X8=72
1X9= 9  2X9=18  3X9=27  4X9=36  5X9=45  6X9=54  7X9=63  8X9=72  9X9=81
```

예제 설명 문자열을 입력받고 그 문자열에서 변환할 기존 문자와 새로운 문자를 각각 입력받은 뒤 변환된 문자열을 반환하는 프로그램이다.

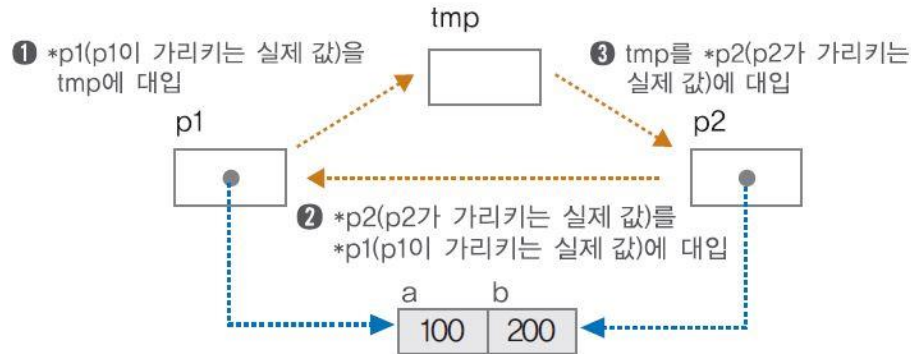
실행 결과



```
C:\Windows\system32\cmd.exe
여러 글자를 입력 : Microsoft Visual Studio
기존 문자와 새로운 문자 : i #
변환된 결과 ==> M#crosoft U#sual Stud#o
```


[실습25] 포인터를 이용한 두 값 교환

예제 설명 입력한 두 값을 포인터를 활용하여 교환하는 프로그램이다. 값을 바꾸는 개념은 다음과 같다.



실행 결과

```
C:\Windows\system32\cmd.exe

a 값 입력 : 100
b 값 입력 : 200
바뀐 값 a는 200, b는 100
```

```
int main()
{
    int a, b, temp;

    puts("a와 b값을 입력하세요.");
    scanf("%d %d", &a, &b);

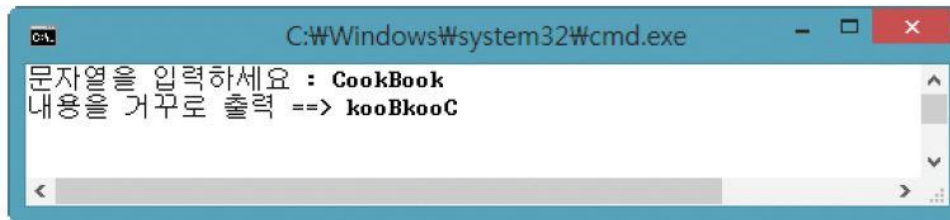
    temp = a;
    a = b;
    b = temp;

    printf("a = %d, b = %d\n", a, b);
}
```

[실습24] 포인터를 이용해서 문자열

예제 설명 8장의 [예제모음 20]에서처럼 입력한 문자열을 반대 순서로 출력해보자. 이번에는 포인터를 활용하여 작성한 프로그램이다.

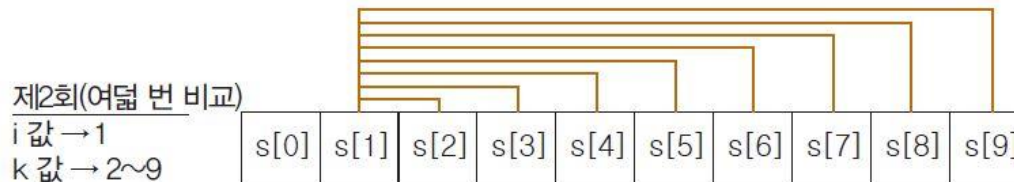
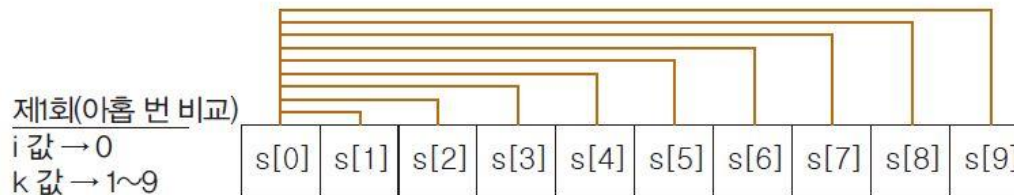
실행 결과



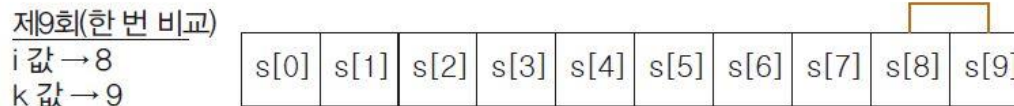
```
C:\Windows\system32\cmd.exe
문자열을 입력하세요 : CookBook
내용을 거꾸로 출력 ==> kooBkooC
```

[실습26] 포인터를 이용한 배열 정렬

예제 설명 포인터를 이용하여 배열에 들어 있는 값 10개를 정렬하는 프로그램이다. 다음 그림을 참고하여 두 값을 비교하고 작은 것을 앞으로 옮기는 선택 정렬을 사용한다.



⋮



실행 결과

```
C:\Windows\system32\cmd.exe
정렬 전 배열 s ==> 1 0 3 2 5 4 7 6 9 8
정렬 후 배열 s ==> 0 1 2 3 4 5 6 7 8 9
```

[실습] 배열

- ▶ 분반 3개가 있고, 각 분반은 학생 10명으로 구성되어 있다. 각 학생들의 3번의 시험에 대한 점수를 3차원 배열에 저장하고자 한다. 즉, 다음과 같이 선언되는 3차원 배열을 사용할 수 있다. 첫번째 index는 분반, 두번째 index는 학생, 세번째 index는 시험을 나타낸다.

```
int score[3][10][3];
```

- ▶ 0~100 사이의 정수인 random number를 생성하여 총 90개(=3×10×3)의 점수 데이터를 배열에 저장하세요 (for문 3번 중첩 사용). 그리고 각 시험에 대한 30명 학생의 평균점수를 계산하여 출력하세요.

시험1 평균 : 52.0
시험2 평균 : 53.7
시험3 평균 : 56.3

[실행 예]

Today

1. 포인터 활용

2. 실습

[복습] 포인터란?

- 포인터(pointer)
 - 변수의 메모리 주소
- 주소를 이용해서 변수에 접근하거나 수정할 수 있음
- 변수의 주소를 찾는 방법은?
 - & 기호를 사용해서 찾을 수 있음
 - t 변수의 주소를 담는 변수는 *t 로 쓰며, 데이터형에 *을 써도 동일한 의미임

정수변수 n의 주소
실수변수 pi의 주소

```
#include <stdio.h>

int main(){
    int n = 4;
    double pi = 3.14159;
    int *pn = &n;
    double *ppi = &pi;
}
```

==

```
#include <stdio.h>

int main(){
    int n = 4;
    double pi = 3.14159;
    int* pn = &n;
    double* ppi = &pi;
}
```

포인터는 무엇에 쓸까?

- 주소변수를 이용해서 접근하거나 수정할 수 있음
 - 간접 기호 * 를 이용해서 역으로 참조할 수 있음

```
int n = 4;  
double pi = 3.14159;  
int* pn = &n;  
double* ppi = &pi;
```

```
int n = 4;  
double pi = 3.14159;  
int* pn = &n;  
double* ppi = &pi;
```

```
printf("pi = %lf \n", pi); == printf("pi = %lf \n", *ppi);
```

- ```
*ppi = *ppi + *pn;
 printf("*ppi: %lf\n", *ppi);
```

```
*ppi = *ppi + *pn;
printf("*ppi: %lf\n", *ppi);
printf("pi: %lf\n", pi);
```

# 함수에 포인터 넘겨주기

- 두 경우의 차이는?

- 매개변수로 값을 넘겨주는 경우 vs. 매개변수로 주소를 넘겨주는 경우
- (Call by value) (Call by reference)

```
void swap(int x, int y) {
 int temp = x;

 x = y;
 y = temp;
}

int main() {

 int a = 5, b = 7;

 swap(a, b);
 printf("a: %d, b:%d \n", a, b);
}
```

```
void swap(int *x, int *y){
 int temp = *x;

 *x = *y;
 *y = temp;
}

int main(){

 int a = 5, b = 7;

 swap (&a, &b);
 printf("a: %d, b:%d \n", a, b);
}
```



# [복습] 배열과 포인터

- 배열의 변수명은 주소를 나타낸다!!

```
int arr[8];

int a = arr[0];
int *pa = &arr[0]; == int *pa = arr;
```

# 배열과 포인터

```
int* swap(int x, int y) {
 int temp = x;

 x = y;
 y = temp;

 int rslt[2] = { 0, 0 };
 rslt[0] = x;
 rslt[1] = y;

 return rslt;
}

int main() {

 int a = 5, b = 7;

 int *s = swap(a, b);
 a = *s;
 b = *(s+1);
 printf("a: %d, b:%d \n", a, b);
}
```

```
void swap(int *x, int *y){
 int temp = *x;

 *x = *y;
 *y = temp;
}

int main(){

 int a = 5, b = 7;

 swap (&a, &b);
 printf("a: %d, b:%d \n", a, b);
}
```

# 포인터 연산

- 다음의 예에서,

```
int arr[8];

int *pa = arr;
```

- 포인터변수는 **주소를 담고** 있기 때문에 **정수가 아니지만**,  
더하거나 빼는 연산으로 원하는 주소로 접근이 가능:  
**pa + i** 는 **arr[i]** 를 가리키고 있음
- arr[0]의 주소가 100 이라면, arr[2]의 주소는 108이 됨 (int = 4byte)

- 문자열 포인터

```
char str[] = "This is a string."
char *pc = str;

*(pc + 10) = 'S';
puts(str);
```

# [실습1]

- ▶ **int**형 배열에 **0**에서 **999**까지 **random number**를 **10**개 생성하여 저장한 후, 최소값을 계산하는 프로그램을 작성해 보세요.

```
0: 807
1: 249
2: 73
3: 658
4: 930
5: 272
6: 544
7: 878
8: 923
9: 709
최소값은 73입니다.
```

[실행 예]

## [실습2]

- ▶ **int**형 배열에 0에서 999까지 **random number**를 100개 생성하여 저장한 후, 100단위별로 생성된 숫자들의 빈도수를 출력해 보세요.
- ▶ **Hint.** 앞의 예제를 확장하여 빈도수를 저장할 크기가 10인 **int**형 배열 **bb**를 생성하여 모두 0으로 초기화한다. 다음으로 **for**문을 사용하여 100개의 숫자들을 하나씩 반복하면서 각각 10개의 구간에 속하는 지를 체크하여, 속하는 구간의 빈도수를 1씩 증가시킨다.

|            |     |
|------------|-----|
| 0 - 99:    | 10개 |
| 100 - 199: | 14개 |
| 200 - 299: | 8개  |
| 300 - 399: | 7개  |
| 400 - 499: | 9개  |
| 500 - 599: | 12개 |
| 600 - 699: | 9개  |
| 700 - 799: | 8개  |
| 800 - 899: | 12개 |
| 900 - 999: | 11개 |

[실행 예]