# 3주차 2차시 포인터

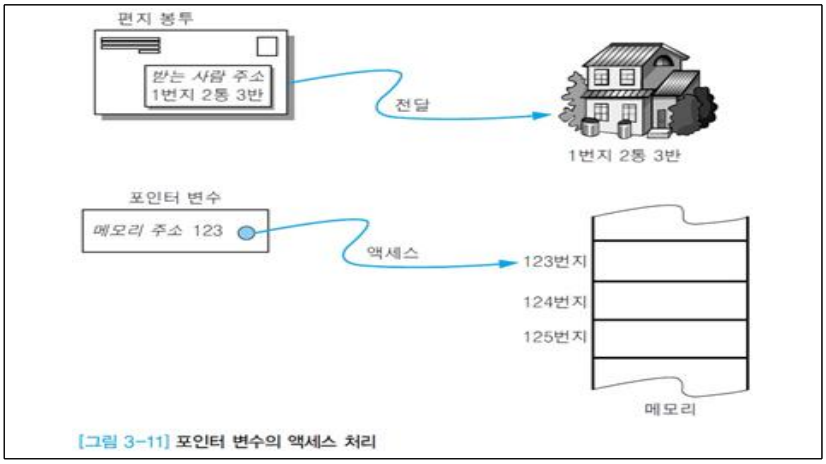
## 【학습목표】

1. 포인터의 의미를 설명할 수 있다.
2. 포인터의 구현방법을 설명할 수 있다.

## 학습내용1 : 포인터

### 1. 정의

- 변수의 메모리 주소값
- 포인터 변수
    - 주소값을 저장하는 특별한 변수
    - 포인터 변수가 어떤 변수의 주소를 저장하고 있다는 것은 포인터 변수가 그 변수를 가리키고 있다(포인트하고 있다)는 의미
    - 포인터 변수를 이용하여 연결된 주소의 변수 영역을 액세스 함
    - 포인터 변수를 간단히 포인터라고 함
- 포인터 변수의 의미
    - 편지봉투에 받는 사람의 집주소를 쓰면, 그 주소로 편지가 전달되어 집주인이 편지를 받게 된다
    - 편지봉투 ⇒ 포인터 변수
    - 편지봉투에 쓰는 받는 사람 주소 ⇒ 포인터 변수에 저장된 변수의 메모리 주소
    - 주소에 해당하는 집주인이 편지를 받는 것 ⇒ 포인터 변수를 통한 변수의 액세스

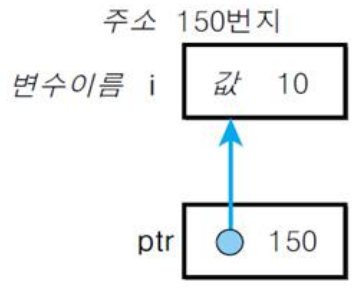


[그림 3-11] 포인터 변수의 액세스 처리

## 2. 포인터 사용 예

```
int i;  .....❶
int *prt = &i; ....❷
```

❶에서 int형 변수 i를 선언했을 때에 저장된 메모리 번지를 150번지라고 한다면,
❷에서 변수 i의 주소를 포인터 변수 ptr에 저장하면 ptr에는 메모리 주소 150이 저장되므로, 포인터 변수 ptr은 150번지의 변수 i를 가리키는 상태가 됨




[그림 3-12] 포인터 변수로 일반 변수를 액세스하는 방법

## 3. 포인터 변수 선언

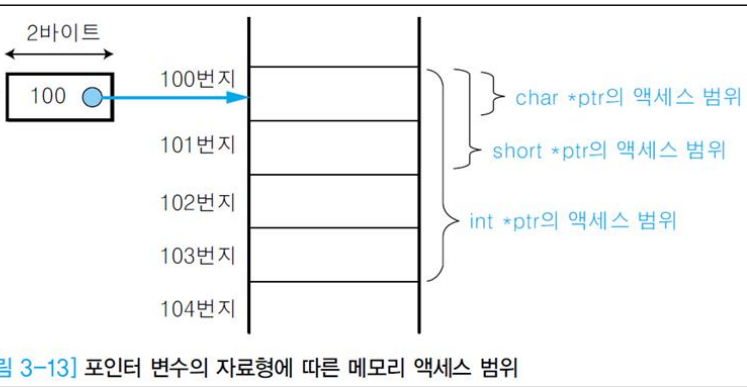
1) 포인터 선언 형식

```
자료형 *포인터 변수이름;
❶      ❷
```

❶ 자료형 : 포인터 변수 자체의 자료형이 아니라, 포인터 변수에 저장할 수 주소에 있는 일반 변수의 자료형
❷ 포인터 변수 이름 : 일반 변수이름과 구별하여 변수이름 앞에 '*'를 표시하여 포인터 변수임을 표시

2) 포인터 변수의 자료형에 따른 메모리 액세스 범위

```
char * ptr; .....❶
short *ptr; .....❷
int *ptr;  .....❸
```

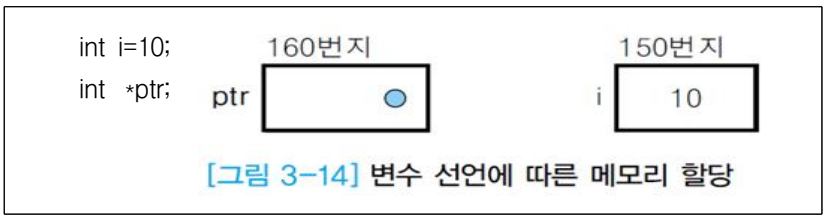



[그림 3-13] 포인터 변수의 자료형에 따른 메모리 액세스 범위

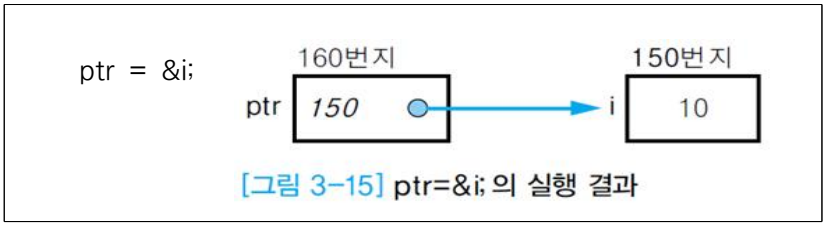
## 4. 포인터 연산

* 주소 연산자 : &
- 변수의 주소를 구하기 위하여 사용
- 변수 앞에 &를 사용하여 그 변수의 주소를 사용
    - 포인터 변수 = &변수 ;
    - 사용할 주소 영역의 변수와 포인터 변수는 같은 자료형으로 선언

- 주소 연산자 사용 예



[그림 3-14] 변수 선언에 따른 메모리 할당



[그림 3-15] ptr=&i; 의 실행 결과

* 참조 연산자 : *
- 저장된 주소에 있는 값(변수에 저장된 값)을 액세스하는 연산자

- 사용 방법 1 : 지정한 값을 포인터가 가리키고 있는 주소에 저장

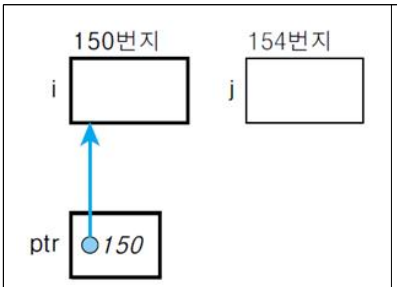
```
* 포인터 변수 = &변수 ;
```

- 사용 방법 2 : 포인터가 가리키는 주소에 있는 값을 변수에 저장

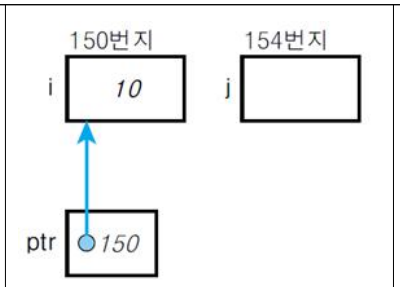
```
변수 = * 포인터 변수 ;
```

- 참조 연산자 사용 예

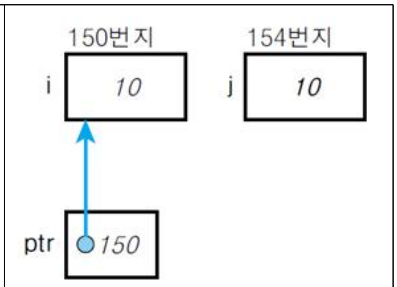
```
int I, j;
int *ptr;
ptr = &i;  .....❶
*ptr = 10; .....❷
j = *ptr ;  .....❸
```



[그림 3-16] ptr=&i; 의 실행 결과

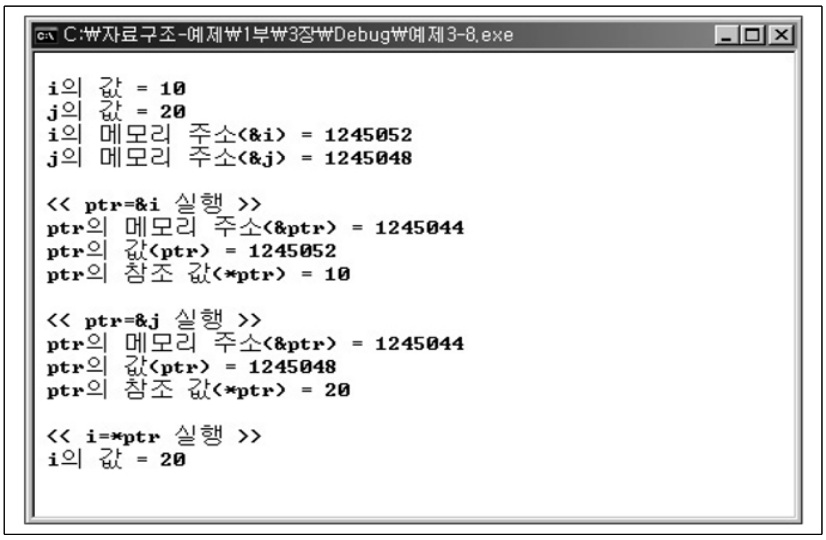


[그림 3-17] *ptr=10; 의 실행 결과



[그림 3-18] j=*ptr; 의 실행 결과

- [예제 3-8] 포인터 연산자 사용 예 프로그램
  - 실행화면



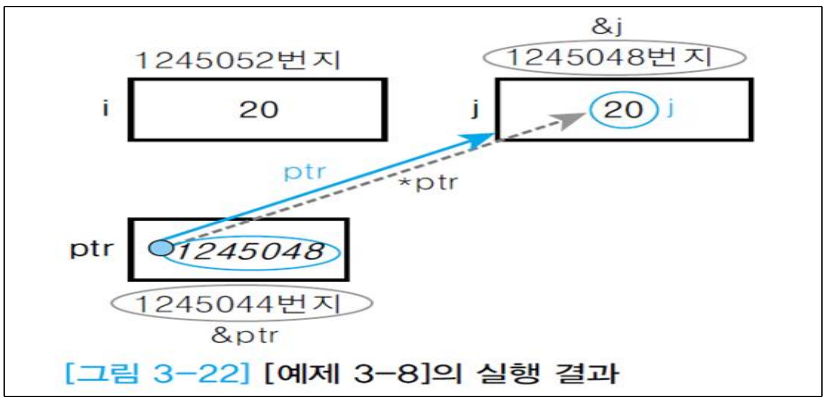

- [예제 3-8]의 실행 후 상태

## 5. 포인터의 초기화

**자료형 * 포인터 변수 = 초기값 주소 ;**

1) 포인터 초기화 방법 1

- 주소 연산자를 사용하여 변수의 주소 지정




[그림 3-22] [예제 3-8]의 실행 결과

**자료형 변수;**
**자료형 * 포인터 변수 = 초기값 주소 ;**

- 예) int i;
  int *ptr = &i;

2) 포인터 초기화 방법 2
- 동적 메모리를 할당하고 그 시작주소를 포인터 값으로 지정
- 예) char *ptr = (char *) malloc(100);

3) 포인터 초기화 방법 3
- 문자형 포인터에 문자열의 시작주소를 지정
- 예) char *ptr = “korea”;

4) 포인터 초기화 방법 4
- 배열의 이름을 이용하여 배열시작주소를 지정 : 배열 이름은 문자열과 마찬가지로 그 시작주소를 전달할 수 있음
- 예) char A[100];
  char *ptr = A;

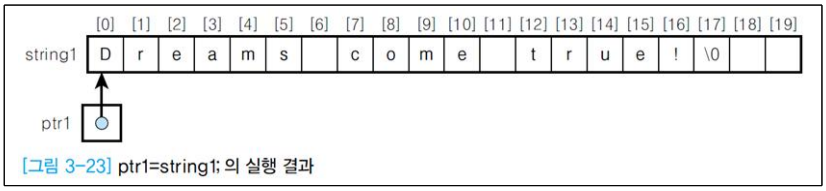
5) 포인터 초기화 방법 5
- 배열의 첫 번째 요소의 주소를 사용하여 배열 시작 주소를 지정
- 인덱스로 표시하는 배열의 각 요소는 그 이름만으로는 주소를 전달 할 수 없기 때문에 주소연산자 &를 사용
- 예) char A[100];
  char *ptr = &A[0];

## 학습내용2 : 문자배열 vs. 포인터배열

### 1. 포인터와 문자배열

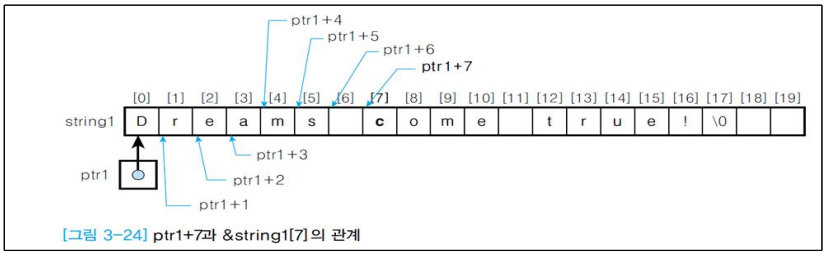
- 포인터를 사용하여 문자열 연산 처리
- [예제 3-9] 포인터를 이용한 문자열 처리 프로그램

* ptr1 = string1;



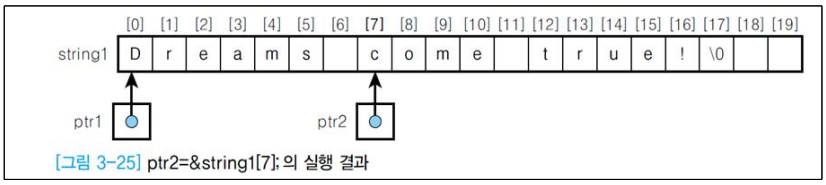

[그림 3-23] ptr1=string1; 의 실행 결과

* ptr1+ 7과 &string[7]의 관계



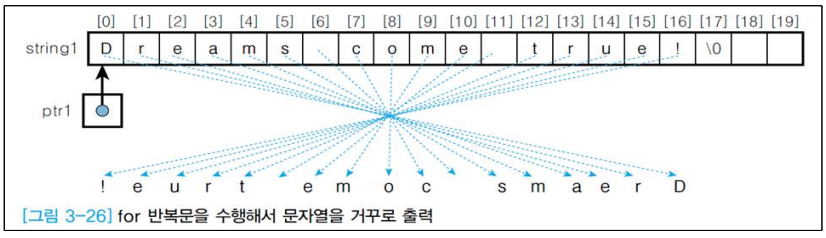

[그림 3-24] ptr1+7과 &string1[7]의 관계

* ptr2 = &string1[7];



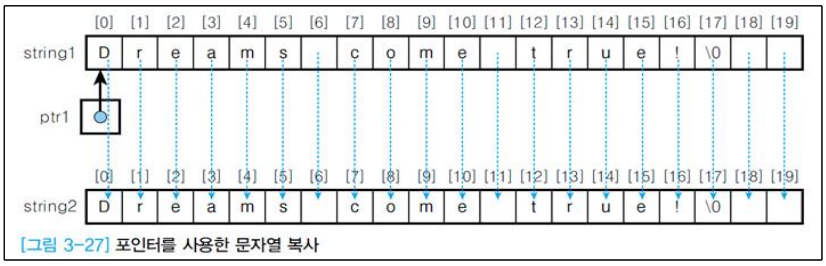

[그림 3-25] ptr2=&string1[7]; 의 실행 결과

* for(i=16; i>=0; I--){
  putchar(*(ptr1+1));
  }



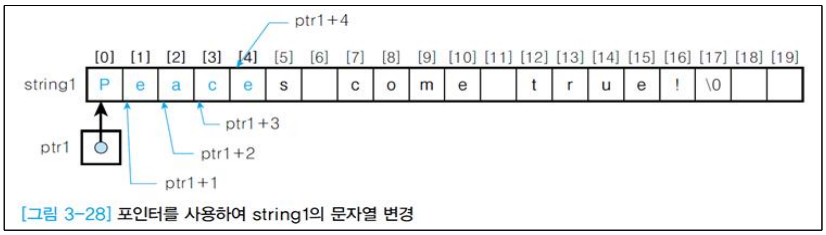

[그림 3-26] for 반복문을 수행해서 문자열을 거꾸로 출력

* 포인터를 사용한 문자열 복사



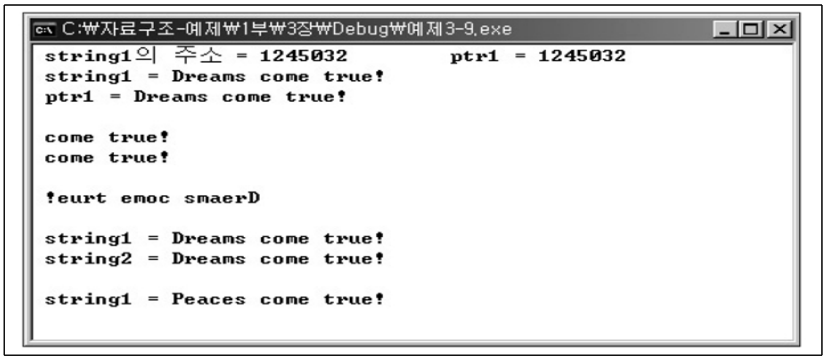
[그림 3-27] 포인터를 사용한 문자열 복사

* 포인터를 사용하여 string1의 문자열 변경



[그림 3-28] 포인터를 사용하여 string1의 문자열 변경

* 예제 3-9의 실행 결과




## 2. 포인터 배열

- 포인터 자료형을 배열로 구성
  - 여러 개의 포인터를 하나의 배열로 구성한 배열의 특징과 포인터의 특징을 모두 활용 가능
- 포인터 배열의 선언형식

**자료형 * 포인터배열이름[배열크기];**

- 포인터 배열에서 각 배열요소는 포인터
- 2차원 문자배열을 1차원 포인터배열로 표현
  - 2차원 배열의 행의 개수 : 포인터배열 크기
  - 포인터배열의 각 배열요소 : 각 문자열에 대한 시작주소를 가진 포인터

- 2차원 배열과 1차원 포인터배열

```
char  string[3][10] = {"Dreams", "come", "true!"};
```

| | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|---|---|---|---|---|---|---|---|---|---|---|
| string[0] | D | r | e | a | m | s | \0 | | | |
| string[1] | c | o | m | e | \0 | | | | | |
| string[2] | t | r | u | e | ! | \0 | | | | |

[그림 3-29] 2차원 문자 배열 string[3][10]의 구조

```
char  *ptr[3] = {{"Dreams"}, {"come"}, {"true!"}};
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ptr [0] → | D | r | e | a | m | s | \0 |
| ptr [1] → | c | o | m | e | \0 | | |
| ptr [2] → | t | r | u | e | ! | \0 | |

[그림 3-30] 1차원 포인터 배열 *ptr[3]의 구조

* 예제 3-10 : 포인터배열을 이용한 문자열 저장 프로그램

- 실행 결과

```
C:₩자료구조-예제₩1부₩3장₩Debug₩예제3-10.exe

Korea
Seoul
Mapo
152번지 2/3


Korea
Seoul
Jongno
152번지 2/3
```

## 학습내용3 : 포인터의 포인터

### 1. 정의

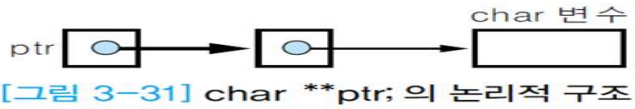
1) 포인터를 가리키고 있는 포인터, 즉  이중 포인터

### 2. 포인터의 포인터 선언 형식

**자료형 ** 포인터변수이름**

- 예)

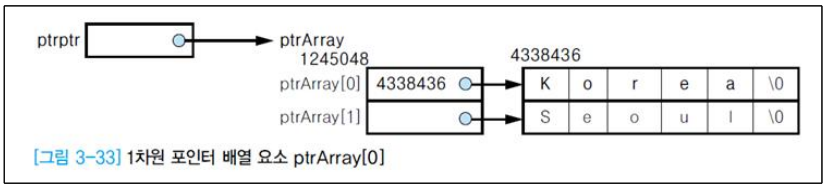
```
char  **ptr;
```



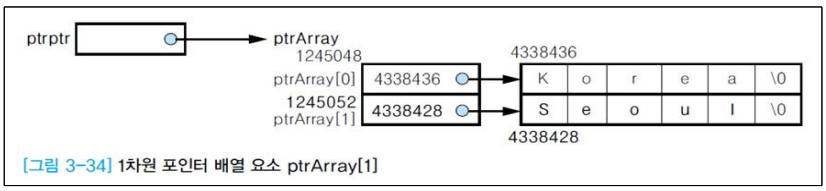

[그림 3-31] char **ptr; 의 논리적 구조

## 3. 예제 3-11 : 포인터 배열과 포인터의 포인터 예제 프로그램

char *ptrArray[2];
char **ptrptr;

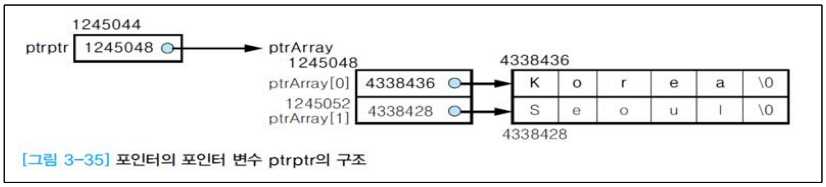


[그림 3-33] 1차원 포인터 배열 요소 ptrArray[0]



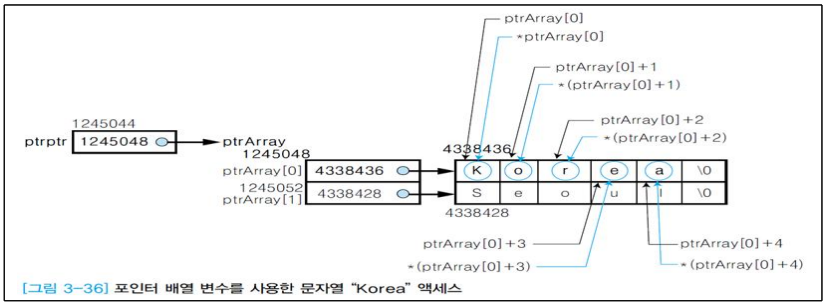
[그림 3-34] 1차원 포인터 배열 요소 ptrArray[1]

* ptrptr = ptrArray;



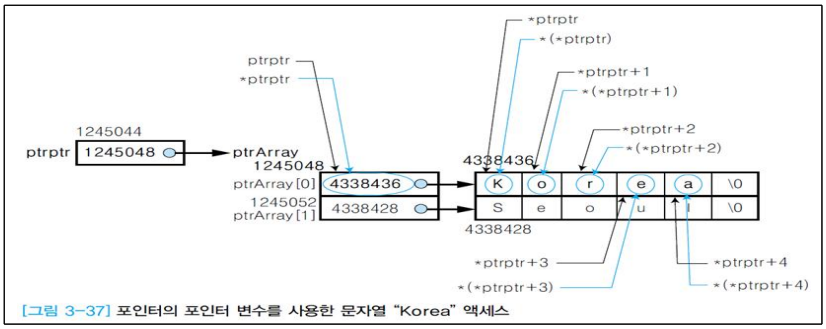
[그림 3-35] 포인터의 포인터 변수 ptrptr의 구조

* 포인터배열변수 ptrArray를 사용하여 문자여 “Korea” 액세스하기



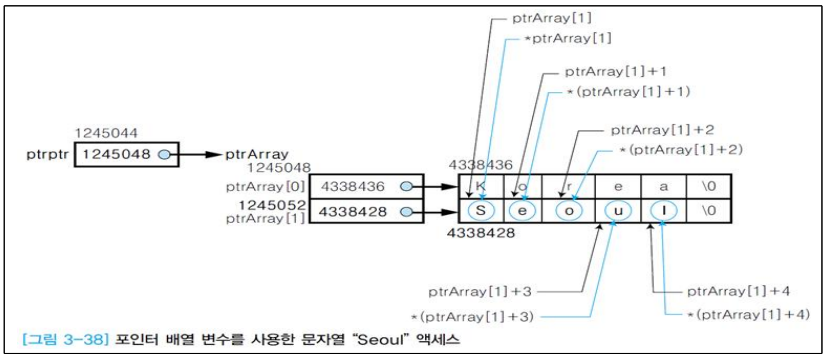
[그림 3-36] 포인터 배열 변수를 사용한 문자열 “Korea” 액세스

* 포인터의 포인터 변수 ptrptr을 사용하여 문자열 “Korea” 액세스 하기



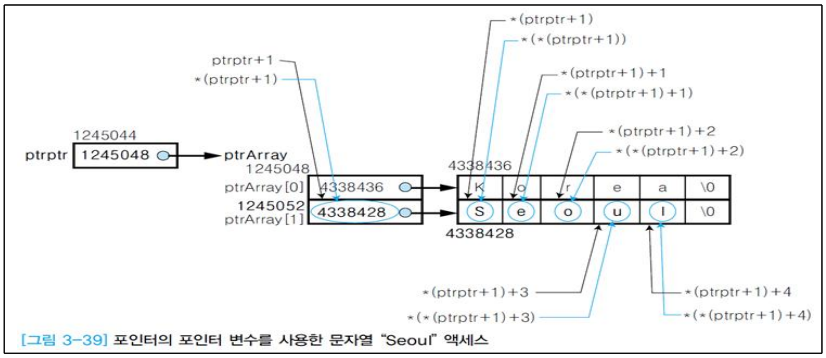
[그림 3-37] 포인터의 포인터 변수를 사용한 문자열 “Korea” 액세스

* 포인터배열변수 ptrArray를 사용하여 문자열 “Seoul” 액세스 하기



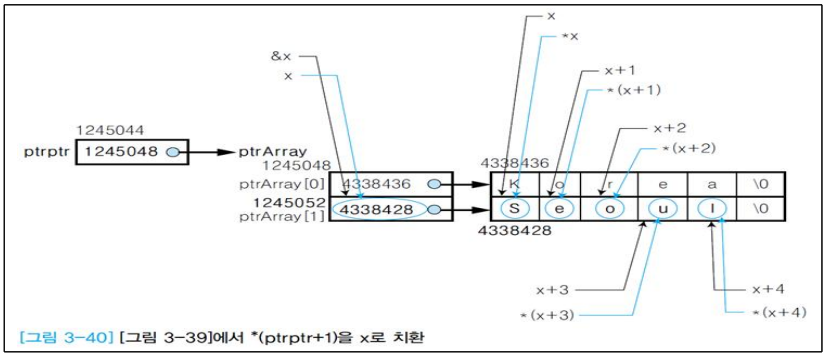
[그림 3-38] 포인터 배열 변수를 사용한 문자열 "Seoul" 액세스

* 포인터의 포인터 변수 ptrptr을 사용하여 문자열 “Seoul” 액세스 하기



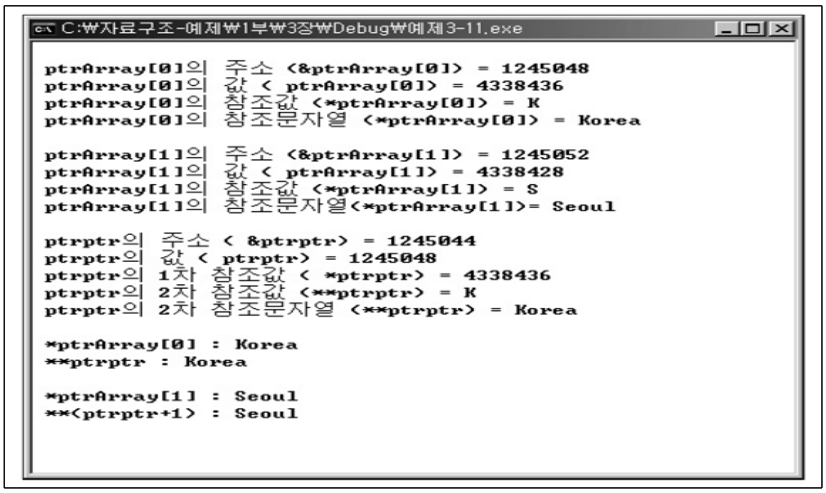
[그림 3-39] 포인터의 포인터 변수를 사용한 문자열 "Seoul" 액세스

* *(ptrptr+1)을 x로 치환하여 [그림 3-39] 다시보기



[그림 3-40] [그림 3-39]에서 *(ptrptr+1)을 x로 치환

* 예제 3-11의 실행 결과




## 【학습정리】

1. 포인터 변수는 주소값을 저장하는 특별한 변수이다.
- 포인터의 연산자로는 주소 연산자 &를 사용하여 변수의 주소를 저장할 수 있고, 참조 연산자 *를 사용하여 포인터가 가리키는 변수의 값을 사용할 수 있다.

2. 포인터 변수에 주소를 지정하는 방법으로는 다음의 5가지가 있다.
 - 주소 연산자를 사용하여 변수의 주소를 지정 방법
 - 동적 메모리를 할당하고 그 시작주소를 포인터 값으로 지정 방법
 - 문자형 포인터에 문자열의 시작주소를 지정 방법
 - 배열이름으로 배열의 시작 주소를 지정 방법
 - 주소 연산자를 사용하여 첫 번째 배열 요소의 시작 주소를 지정 방법