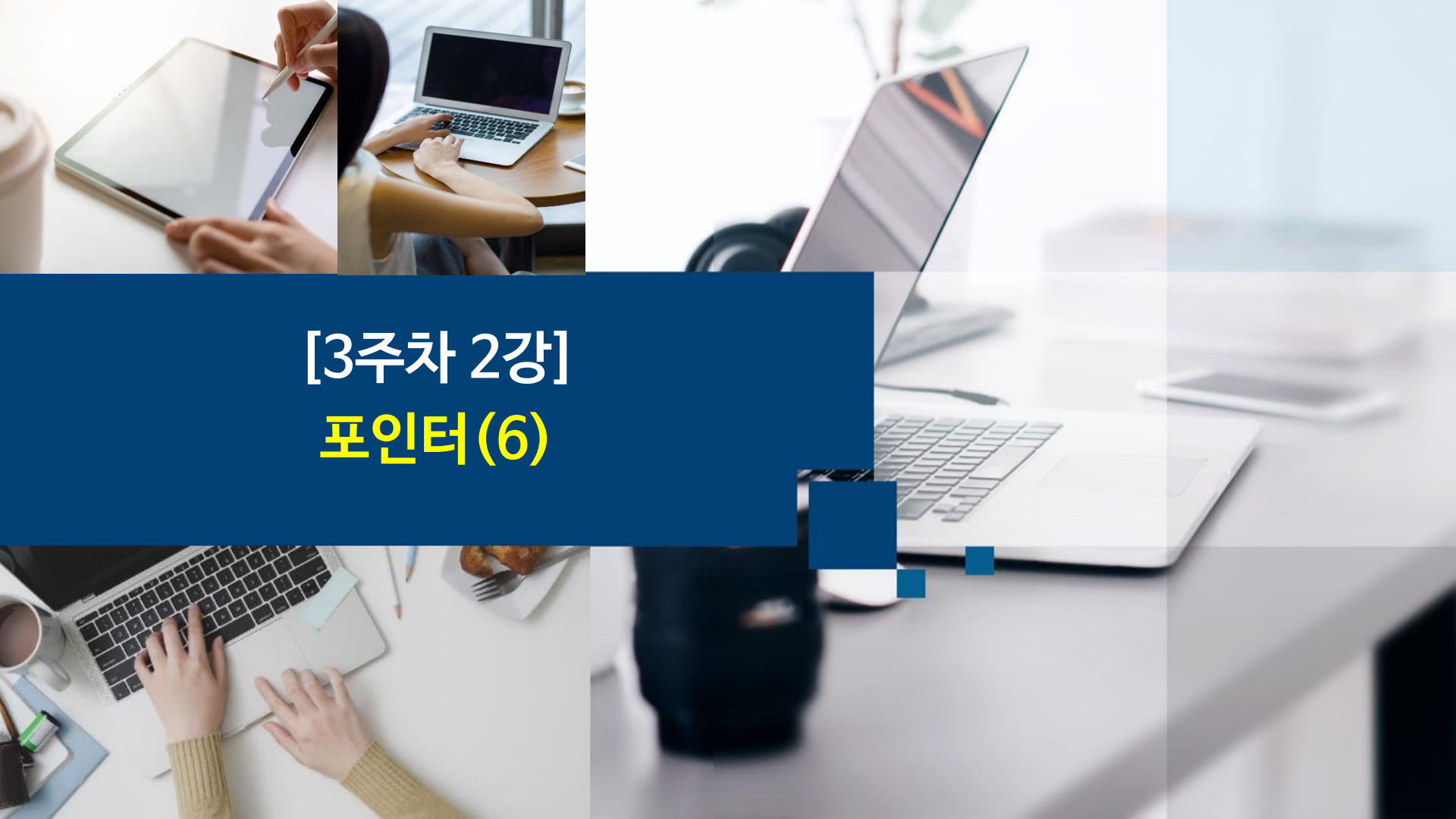


[3주차 2강]
포인터(6)

학습 내용

- 9.6 포인터 배열
- 9.7 다중 포인터

학습 목표

- 9.6 포인터 배열의 사용법을 익힌다.
- 9.7 다중 포인터가 무엇인지 이해한다.

C 프로그래밍 및 실습

1. 포인터 개요
2. 포인터 선언과 사용
3. 배열과 포인터
4. 포인터 연산
5. 포인터 인자와 주소 반환
6. **포인터 배열**
7. 다중 포인터

6. 포인터 배열

포인터 배열 : 포인터 변수들의 묶음

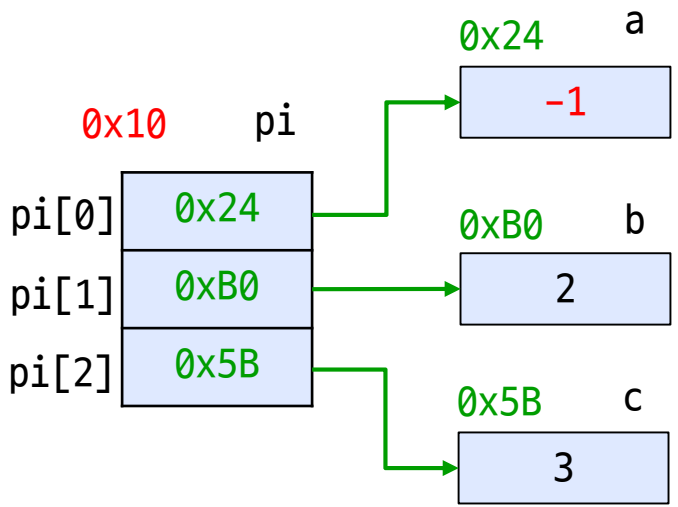
포인터 배열 선언

- 포인터 선언 + 배열 선언

```
void main(){
   int a=1, b=2, c=3, i;
   int *pi[3];          // 포인터 배열 선언

   pi[0] = &a;          // 배열 원소는 int 포인터 변수
   pi[1] = &b;  pi[2] = &c;

   *pi[0] = -1;       // *pi[0]은 *(pi[0]) ? (*pi)[0] ?

   for( i=0; i < 3 ; ++i )
      printf("%p %p %d\n", &pi[i], pi[i], *pi[i]);
}
```

포인터 배열 선언

- 메모리 그림



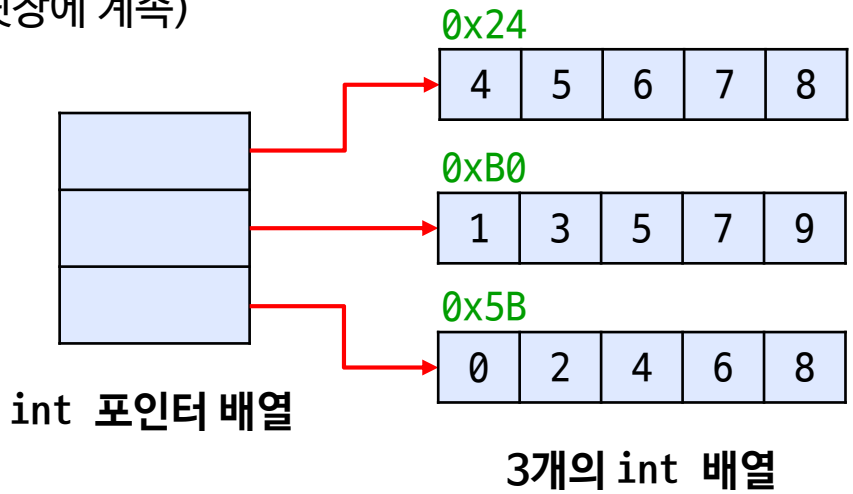
실행 결과

```
006FFD10 006FFD24 -1
006FFD14 006FFDB0 2
006FFD18 006FFD5B 3
```

※ 실습하기

[예제 9.9]
다음 프로그램을 작성하시오.

- ✓ 3개의 int 배열을 선언 (배열의 크기는 모두 5)하고 아래 그림과 같이 초기화
- ✓ 1개의 int 포인터 배열 선언 (배열의 크기는 3)
- ✓ 아래 그림과 같이 포인터 배열의 각 원소를 각 int 배열에 연결
- ✓ (뒷장에 계속)



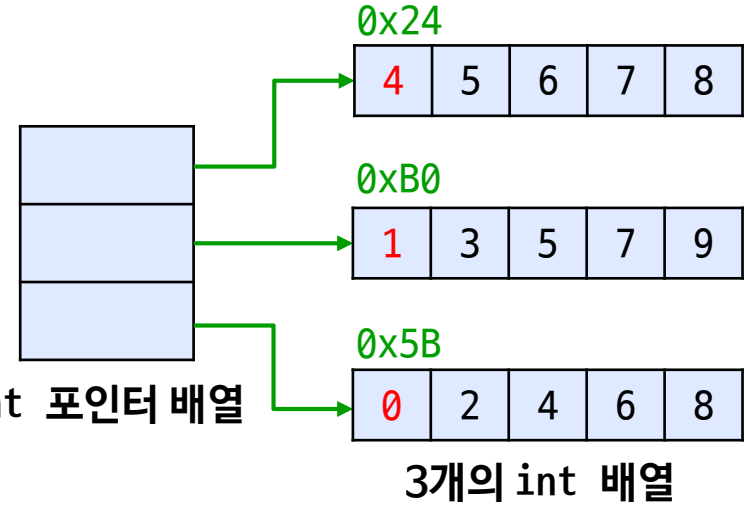
※ 실습하기

[예제 9.9]
다음 프로그램을 작성하시오.

- int 포인터 배열을 이용하여 각 int 배열의 0번 원소의 주소와 값을 출력하시오. (앞 예제 참조)
  - ✓ 단, int 배열의 이름은 사용 금지

실행 결과 (예시)

```
006FFD24 4
006FFDB0 1
006FFD5B 0
```



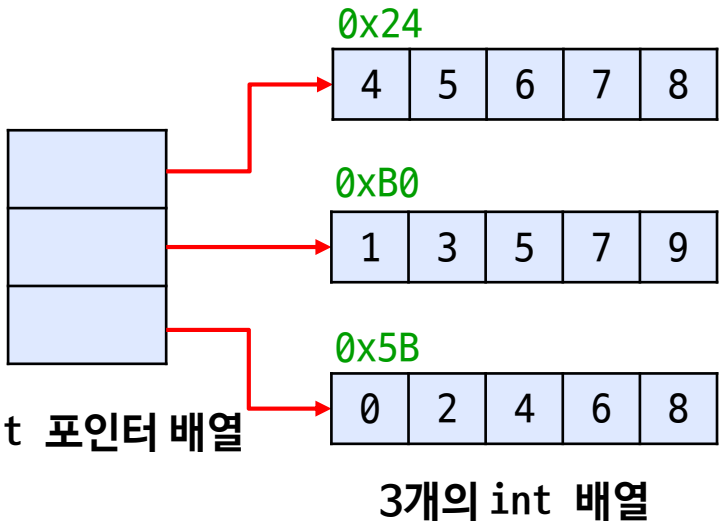
※ 실습하기

[예제 9.10] 이전 실습에서 int 포인터 배열을 이용하여 int 배열의 모든 원소의 값을 출력하시오.

- ✓ 원소 값 출력 코드에서 int 배열의 이름 사용 금지
- ✓ (힌트!!) 2중 반복문, 9.3절 배열과 포인터의 관계

실행 결과

```
4 5 6 7 8
1 3 5 7 9
0 2 4 6 8
```



※ 실습하기

[예제 9.10] 이전 실습에서 int 포인터 배열을 이용하여 int 배열의 모든 원소의 값을 출력하시오.

- 임시변수 x 사용 안 함 → 대신 pi[i] 사용

```
void main(){
   ...                  // int 배열 선언과 초기화 (생략)
   int *pi[3];          // int 포인터 배열 선언

   ...                  // pi 배열에 int 배열 연결 (생략)

   for( i=0 ; i<3 ; ++i ){
      x = pi[i];        // 임시 변수 x에 pi 배열의 원소 값 대입

      for(j=0; j<5; j++)
         printf(" %d", pi[i][j]);  // 2차원 배열처럼
      printf("\n");
   }
}
```

C 프로그래밍 및 실습

1. 포인터 개요
2. 포인터 선언과 사용
3. 배열과 포인터
4. 포인터 연산
5. 포인터 인자와 주소 반환
6. 포인터 배열
7. **다중 포인터**

7. 다중 포인터

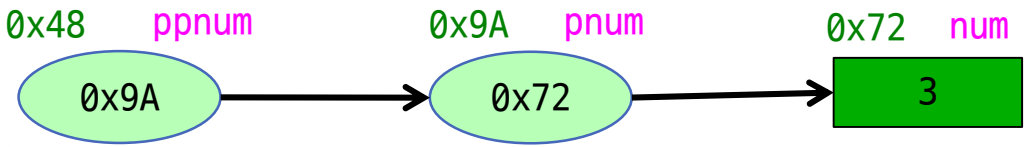
이중 포인터

- 'int 포인터 변수의 주소'를 저장하는 변수? 가능
- 'int 포인터 변수의 주소'를 저장하는 변수의 자료형은? (int **)

선언

- **참조 연산자**를 **두 개** 사용

```
int num=3, *pnum, **ppnum;

pnum = &num;       // 변수 num의 주소가 pnum에 저장됨
ppnum = &pnum;    // 변수 pnum의 주소가 ppnum에 저장됨

printf("%p %p %d\n",ppnum, pnum, num);
```

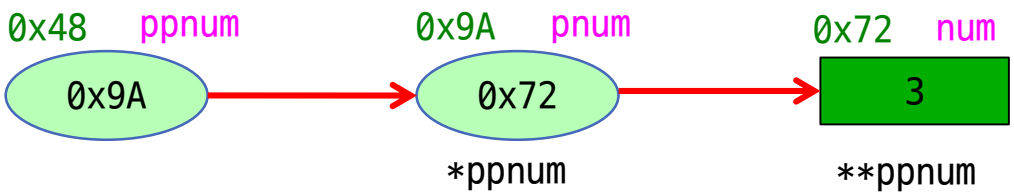


7. 다중 포인터

이중 포인터의 참조 연산

- 참조연산자를 한 개 사용하면 간접 참조를 한 번 (*)
- 참조연산자를 두 개 사용하면 간접 참조를 두 번 (**)

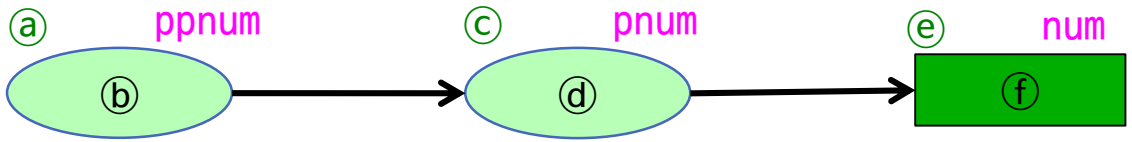
```
int num = 3, *pnum = &num, **ppnum = &pnum;

printf("%p %p %d\n",ppnum, *ppnum, **ppnum);

결과:
001EA09A 001EA072 3
```



※ 실습하기

[예제 9.11] 다음 프로그램의 각 출력이 아래 메모리 그림에서 어느 부분을 출력하는 건지 생각해보고, 프로그램을 실행시켜 확인해보자.

```
int num=3, *pnum = &num, **ppnum = &pnum;

printf("%p %d\n", &num, num);
printf("%p %p %d\n", &pnum, pnum, *pnum);
printf("%p %p %p %d\n", &ppnum, ppnum, *ppnum, **ppnum);
```

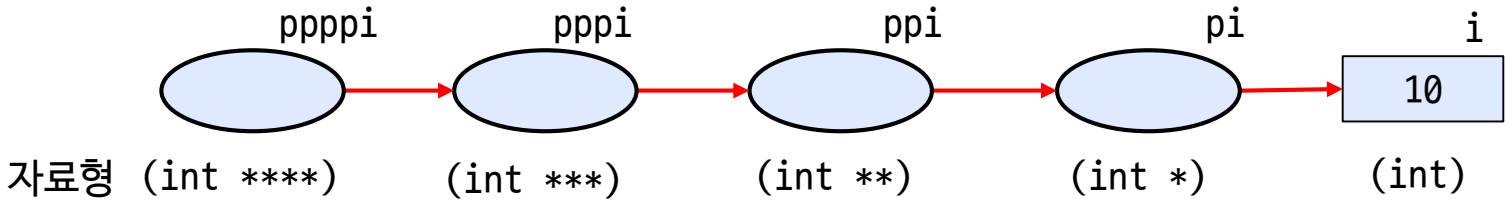


7. 다중 포인터

이중 포인터와 비슷하게, 삼중, 사중, .. 포인터도 가능

- 주의!! 각 포인터의 자료형은 모두 다르다.

```
int i = 10, *pi, **ppi, ***pppi, ****ppppi;

pi = &i              //(int)형인 i의 주소는 (int *)형
ppi = &pi;           //(int *)형인 pi의 주소는 (int **)형
pppi = &ppi;         //(int **)형인 ppi의 주소는 (int ***)형
ppppi = &pppi;       //(int ***)형인 pppi의 주소는 (int ****)형

printf("%d %d %d %d %d\n", ****ppppi, ***pppi, **ppi, *pi, i);
```



학습 정리

- **포인터 배열**은 포인터 변수를 묶은 배열
- int 포인터 변수의 자료형은 **(int *)**이고, int 포인터 변수의 주소의 자료형은 **(int **)**
- **다중 포인터**란 포인터 변수를 가리키는 포인터를 의미함