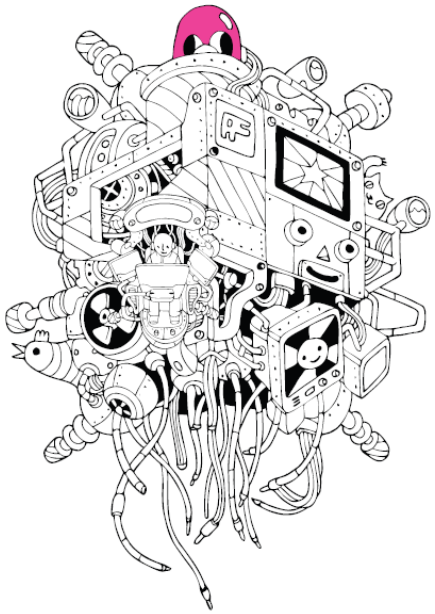


# Chapter 17. 포인터의 포인터

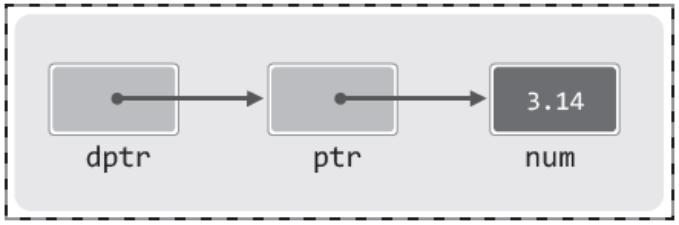
# Chapter 17-1. 포인터의 포인터에 대한 이해

# 실습 I

```
int main(void)
{
    double num = 3.14;
    double *ptr = &num;
    double **dptr = &ptr;
    double *ptr2;

    printf("%9p %9p \n", ptr, *dptr);
    printf("%9g %9g \n", num, **dptr);
    ptr2 = *dptr;   // ptr2 = ptr 과 같은 문장
    *ptr2 = 10.99;
    printf("%9g %9g \n", num, **dptr);
    return 0;
}
```

# 포인터 변수를 가리키는 이중 포인터 변수

```
int main(void)
{
    double num=3.14;
    double * ptr=&num;
    double ** dptr =&ptr;
    . . . . .
}
```

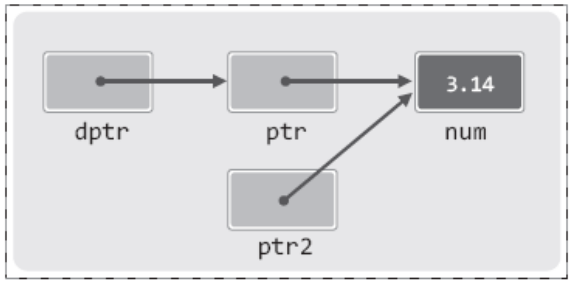



포인터 변수의 주소 값을 저장하는 것이 이중 포인터 변수(더블 포인터 변수)이다.

**위의 상황에서 *dptr은 포인터 변수 ptr을...**

***(*dptr)은 변수 num을 의미하게 된다.**

```
int main(void)
{
    double num = 3.14;
    double *ptr = &num;
    double **dptr = &ptr;
    double *ptr2;

    printf("%9p %9p \n", ptr, *dptr);
    printf("%9g %9g \n", num, **dptr);
    ptr2 = *dptr;   // ptr2 = ptr 과 같은 문장
    *ptr2 = 10.99;
    printf("%9g %9g \n", num, **dptr);
    return 0;
}
```




**이 상황에서 변수 num에 접근하는 네 가지 방법은?**

```
0032FD00    0032FD00
    3.14        3.14
   10.99       10.99
```

실행결과

# 실습 2

```
void SwapIntPtr(int *p1, int *p2)
{
    int * temp=p1;
    p1=p2;
    p2=temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(ptr1, ptr2);
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

# 실습 3

```
void SwapIntPtr(int **dp1, int **dp2)
{
    int *temp = *dp1;
    *dp1 = *dp2;
    *dp2 = temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(&ptr1, &ptr2);  // ptr1과 ptr2의 주소 값 전달!
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

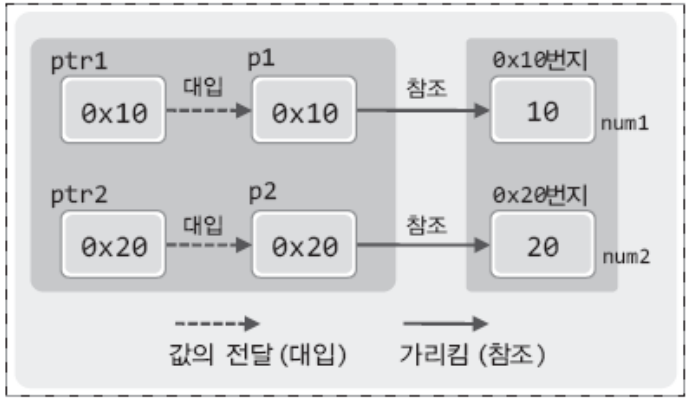
# 포인터 변수의 Swap 1

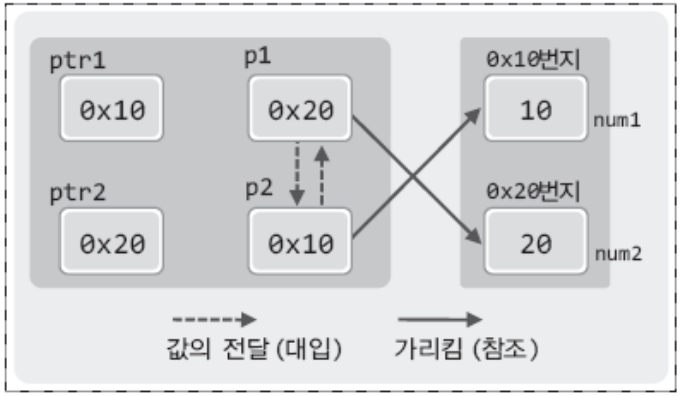
```
void SwapIntPtr(int *p1, int *p2)
{
    int * temp=p1;
    p1=p2;
    p2=temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(ptr1, ptr2);
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

ptr1과 ptr2의 swap은 성공하는가? 문제점은?

```
*ptr1, *ptr2: 10 20
*ptr1, *ptr2: 10 20
```

실행결과



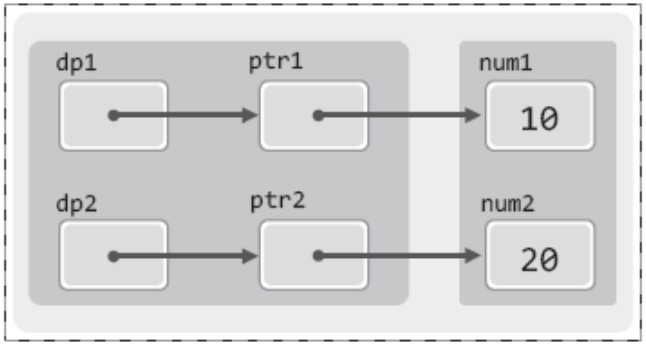



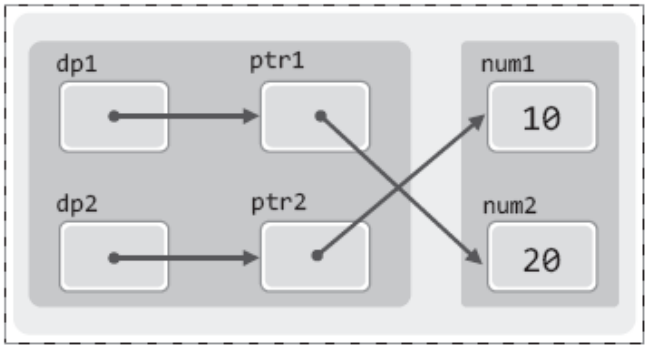

왼편 예제의 실행결과! 결과적으로 ptr1과 ptr2에 저장된 값은 서로 바뀌지 않는다.

# 포인터 변수의 Swap 2

```
void SwapIntPtr(int **dp1, int **dp2)
{
    int *temp = *dp1;
    *dp1 = *dp2;
    *dp2 = temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(&ptr1, &ptr2);  // ptr1과 ptr2의 주소 값 전달!
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

포인터 변수에 저장된 값의 변경이 목적이므로 포인터 변수의 주소 값을 함수에 전달해야 한다.




이중 포인터를 이용해서 두 포인터 변수의 swap에 성공한다.

```
*ptr1, *ptr2: 10 20
*ptr1, *ptr2: 20 10
```

실행결과

# 실습 4

```
int main(void)
{
    int num1=10, num2=20, num3=30;
    int *ptr1=&num1;
    int *ptr2=&num2;
    int *ptr3=&num3;

    int * ptrArr[]={ptr1, ptr2, ptr3};
    int **dptr=ptrArr;

    printf("%d %d %d \n", *(ptrArr[0]), *(ptrArr[1]), *(ptrArr[2]));
    printf("%d %d %d \n", *(dptr[0]), *(dptr[1]), *(dptr[2]));
    return 0;
}
```

# 포인터 배열과 포인터 배열의 포인터 형

```
int * arr1[20];
double * arr2[30];
```

**Ch 13의 후반에 학습한 포인터 변수로 이뤄진 배열(포인터 배열)**

**int arr1[3]; 에서 arr1의 포인터 형은 int *  double arr2[3]; 에서 arr2의 포인터 형은 double ***

**이렇듯 1차원 배열이름의 포인터 형은 배열 이름이 가리키는 대상을 기준으로 결정된다.**

**따라서 int * arr1[20]; 에서 arr1의 포인터 형은 int ****

**double * arr2[30]; 에서 arr2의 포인터 형은 double ****

```
int main(void)
{
    int num1=10, num2=20, num3=30;
    int *ptr1=&num1;
    int *ptr2=&num2;
    int *ptr3=&num3;

    int * ptrArr[]={ptr1, ptr2, ptr3};
    int **dptr=ptrArr;

    printf("%d %d %d \n", *(ptrArr[0]), *(ptrArr[1]), *(ptrArr[2]));
    printf("%d %d %d \n", *(dptr[0]), *(dptr[1]), *(dptr[2]));
    return 0;
}
```

```
10 20 30
10 20 30
```

실행결과

# Chapter 17-2. 다중 포인터 변수와 포인터의 필요성

# 이중 포인터를 가리키는 삼중 포인터

```
int ***tptr;
```

**삼중 포인터 변수!**

**이중 포인터 변수의 주소 값을 담는 용도로 선언된다.**

```
int main(void)
{
    int num=100;
    int *ptr=&num;
    int **dptr=&ptr;
    int ***tptr=&dptr;

    printf("%d %d \n", **dptr, ***tptr);
    return 0;
}
```

실행결과

```
100 100
```

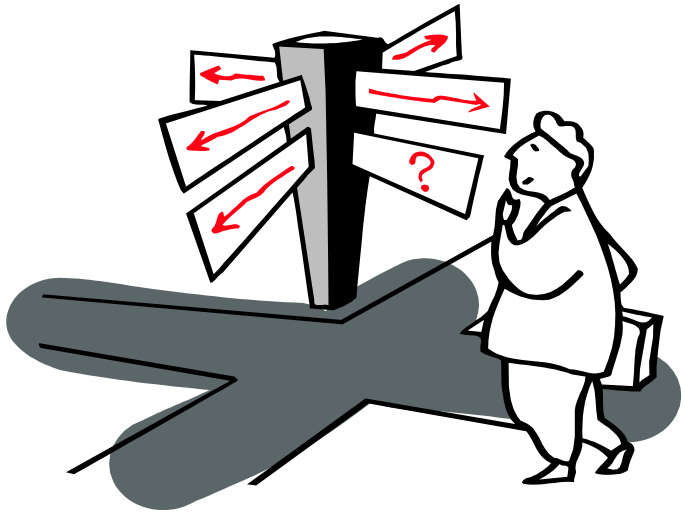
**이중 포인터 변수의 개념을 그대로 확장해서 이해할 수 있는 것이 삼중 포인터 변수이다!**

# 포인터의 필요성은 어디서 찾아야 하는가?

. scanf 함수와 같이 함수 내에서 함수 외부에 선언된 변수의 접근을 허용하기 위해서.

. 메모리의 동적 할당 등등 PART 04에서 공부하는 내용을 통해서 포인터의 필요성을 다양하게 이해하게 된다.

. 향후에 자료구조라는 과목을 공부하게 되면 보다 넓게 필요성을 이해할 수 있게 된다.



Chapter 17이 끝났습니다. 질문 있으신지요?