10주차 2차시 포인터의 포인터

【학습목표】

1. 3차원 배열에 대해 설명할 수 있다.
2. 포인터의 포인터에 대해 설명할 수 있다.

학습내용1 : 3차원 배열

1. 3차원 배열의 논리적 구조

3차원 배열은 2차원 배열에 높이의 개념이 추가된 것이다.
int arr1[2][3][4]; 높이 2, 세로 3, 가로 4인 int형 3차원 배열(세로 3, 가로 4인 배열이 두 개 겹친 형태)
double arr2[5][5][5]; 높이, 세로, 가로가 모두 5인
double형 3차원 배열(세로 5, 가로 5인 배열이 5개 겹친 형태)

```
int main(void)
{
    int arr1[2][3][4];
    double arr2[5][5][5];
    printf("높이2, 세로3, 가로4 int형 배열: %d \n", sizeof(arr1));
    printf("높이5, 세로5, 가로5 double형 배열: %d \n", sizeof(arr2));
    return 0;
}
```

실행 결과

```
높이2, 세로3, 가로4 int형 배열: 96
높이5, 세로5, 가로5 double형 배열: 1000
```

2. 3차원 배열의 선언과 접근

```
int main(void)
{
    int mean=0, i, j;
    int record[3][3][2]={
        {
            {70, 80},   // A 학급 학생 1의 성적
            {94, 90},   // A 학급 학생 2의 성적
            {70, 85}    // A 학급 학생 3의 성적
        },
        {
            {83, 90},   // B 학급 학생 1의 성적
            {95, 60},   // B 학급 학생 2의 성적
            {90, 82}    // B 학급 학생 3의 성적
        },
        {
            {98, 89},   // C 학급 학생 1의 성적
            {99, 94},   // C 학급 학생 2의 성적
            {91, 87}    // C 학급 학생 3의 성적
        }
    };
```

```
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)
            mean += record[0][i][j];
    printf("A 학급 전체 평균: %g \n", (double)mean/6);

    mean=0;
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)
            mean += record[1][i][j];
    printf("B 학급 전체 평균: %g \n", (double)mean/6);

    mean=0;
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)
            mean += record[2][i][j];
    printf("C 학급 전체 평균: %g \n", (double)mean/6);
    return 0;
}
```

실행결과

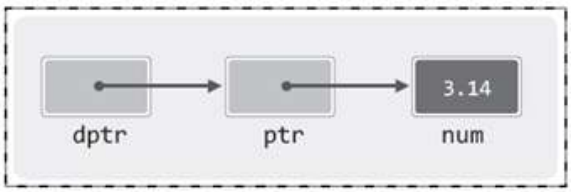
```
A 학급 전체 평균: 81.5
B 학급 전체 평균: 83.3333
C 학급 전체 평균: 93
```

학습내용2 : 포인터의 포인터에 대한 이해

1. 포인터 변수를 가리키는 이중 포인터 변수

포인터 변수의 주소값을 저장하는 것이 이중 포인터 변수(더블 포인터 변수)이다.

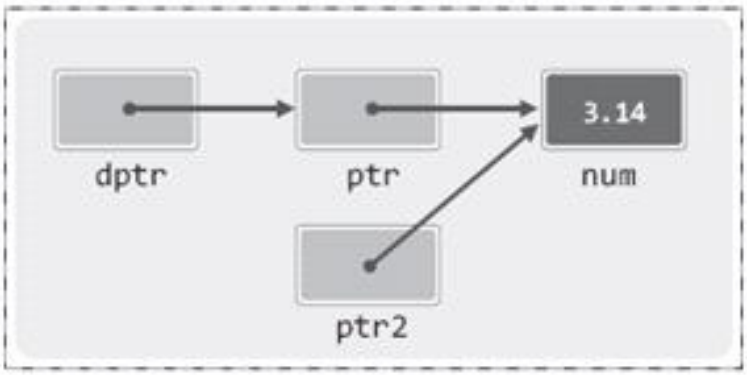
```
int main(void)
{
    double num=3.14;
    double * ptr=&num;
    double ** dptr =&ptr;
    .....
}
```



위의 상황에서 *dptr은 포인터 변수 ptr을... *(*dptr)은 변수 num을 의미하게 된다.

```c
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;
}
```

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

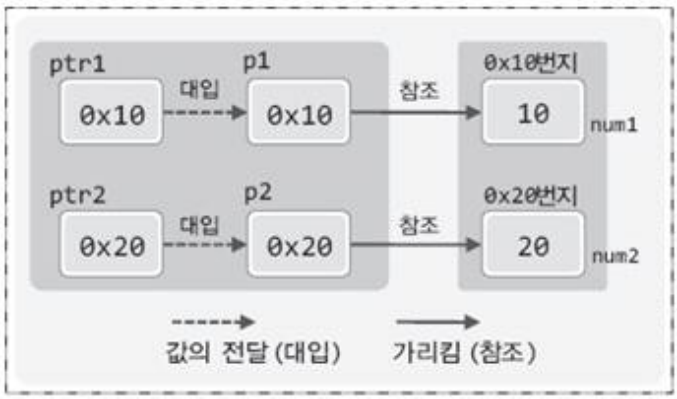


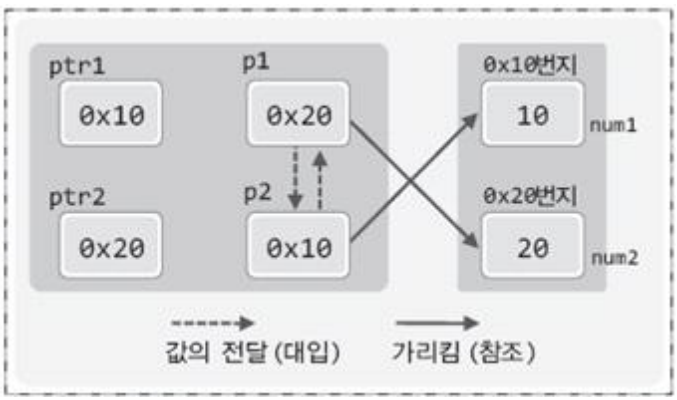
2. 포인터 변수의 Swap 1

```
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;
}
```

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

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

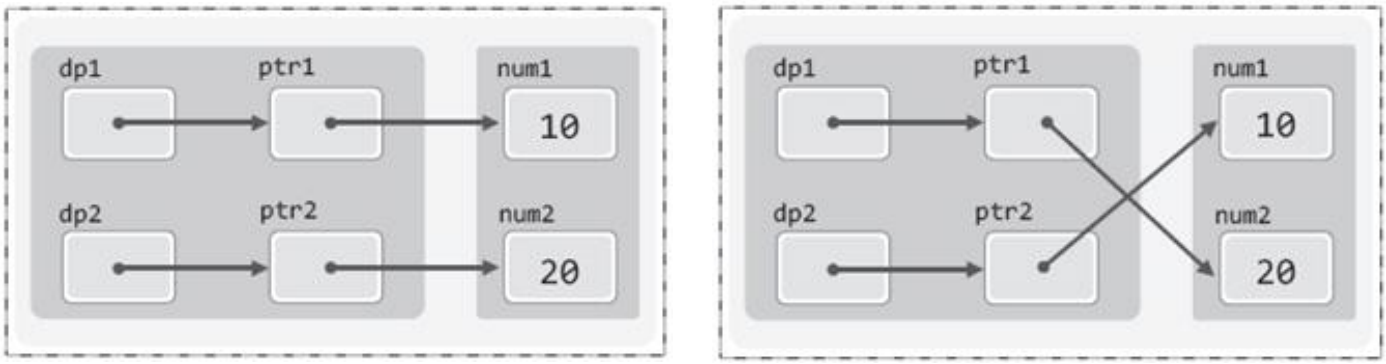




3. 포인터 변수의 Swap 2

```
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;
}
```

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



이중 포인터를 이용해서 두 포인터 변수의 swap에 성공한다.

4. 포인터 배열과 포인터 배열의 포인터 형

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

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
```

5. .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;
}
```

```
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 포인터의 필요성을 다양하게 이해하게 된다.
  ✓향후에 자료구조라는 과목을 공부하게 되면 보다 넓게 필요성을 이해할 수 있게 된다.

【학습정리】

1. 포인터의 필요성은 scanf 함수와 같이 함수 내에서 함수 외부에 선언된 변수의 접근을 허용하기 위함이고, 메모리의 동적 할당, 향후에 자료구조라는 과목을 공부하게 되면 보다 넓게 필요성을 이해할 수 있게 된다.

2. 삼중 포인터 변수는 이중 포인터 변수의 주소 값을 담는 용도로 선언된다.