10주차 3차시 다차원 배열과 포인터 포인터 실습

【학습목표】

1. 실습을 통하여 다차원 배열을 학습하고 실행할 수 있다.
2. 실습을 통하여 포인터 포인터를 학습하고 실행할 수 있다.

학습내용1 : 다차원 배열 실습

1. 다차원 배열의 이해와 활용

√프로그램 사용자로부터 층별로 두 가구가 사는 4층짜리 빌라의 가구별 거주인원 수를 입력받는 예제를 작성해 보세요. 단 층별로 두 가구가 사는 4층짜리 빌라이니, 가로가 2이고 세로가 4인 2차원 배열로 표현하세요.

```
int main(void)
{
    int villa[4][2];
    int popu, i, j;
    /* 가구별 거주인원 입력 받기 */
    for(i=0; i<4; i++)
    {
        for(j=0; j<2; j++)
        {
            printf("%d층 %d호 인구수: ", i+1, j+1);
            scanf("%d", &villa[i][j]);
        }
    }
    /* 빌라의 층별 인구수 출력하기 */
    for(i=0; i<4; i++)
    {
        popu=0;
        popu += villa[i][0];
        popu += villa[i][1];
        printf("%d층 인구수: %d \n", i+1, popu);
    }
    return 0;
}
```

```
1층 1호 인구수: 2
1층 2호 인구수: 4
2층 1호 인구수: 3
2층 2호 인구수: 5
3층 1호 인구수: 2
3층 2호 인구수: 6
4층 1호 인구수: 4
4층 2호 인구수: 3
1층 인구수: 6
2층 인구수: 8
3층 인구수: 8
4층 인구수: 7
```

√2차원 배열의 초기화에 대한 예제 실습 교제(10주차 2차시 예제 참조)

```
int main(void)
{
    int i, j;

    /* 2차원 배열 초기화의 예 1 */
    int arr1[3][3]={
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    /* 2차원 배열 초기화의 예 2 */
    int arr2[3][3]={
        {1},
        {4, 5},
        {7, 8, 9}
    };
    /* 2차원 배열 초기화의 예 3 */
    int arr3[3][3]={1, 2, 3, 4, 5, 6, 7};
```

```
    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr1[i][j]);
        printf("\n");
    }
    printf("\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr2[i][j]);
        printf("\n");
    }
    printf("\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr3[i][j]);
        printf("\n");
    }
    return 0;
}
```

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

1 0 0
4 5 0
7 8 9

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

√가로 길이가 9, 세로의 길이가 3인 int형 2차원 배열을 선언하여 구구단 중 2단, 3단, 4단을 다음과 같이 저장하자.

	1열	2열	3열	4열	5열	6열	7열	8열	9열
1행	2	4	6	8	10	12	14	16	18
2행	3	6	9	12	15	18	21	24	27
3행	4	8	12	16	20	24	28	32	36

그리고 제대로 저장이 되었는지 확인하기 위한 출력을 진행하는 예제를 작성해 보자.

학습내용2 : 포인터 포인터 실습

1. 포인터의 포인터

√포인터 변수 대상의 Call-by-reference 활용한 포인터 Swap 성공 예제 실습
(10주차 2차시 강의안 참조)

```
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
```

√포인터 배열과 포인터 배열의 포인터 형 실습
(10주차 2차시 강의안 참조)

```
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
```

【학습정리】

1. const 선언은 추가적인 기능을 제공하기 위한 것이 아니라, 코드의 안전성을 높이기 위한 것이다. 따라서 이러한 const의 선언을 소홀히 하기 쉬운데, const의 선언과 같이 코드의 안전성을 높이는 선언은 가치가 매우 높은 선언이다.

2. 배열의 크기를 알려주지 않고 초기화 하면 컴파일러가 숫자를 결정하지 못한다.

3. 포인터의 필요성은 함수 내에서 함수 외부에 선언된 변수의 접근을 허용하기 위함이고, 메모리의 동적 할당을 이해하기 위함이다.