

14-01. 다차원 배열

지금까지 배운 내용은 1차원 배열이다. 이를 다수에게 적용시키기 위해 2차원 배열을 사용한다. 2차원 배열은 행렬의 구조와 동일하다. 지금부터 1차원 배열을 모아 만드는 다차원 배열에 대하여 학습을 해보자.

- 2차원 배열 선언과 요소 사용

형태가 같은 배열이 여러 개 필요한 경우 이를 모아 배열을 만들 수 있다. 이러한 배열을 2차원 배열이라고 한다.

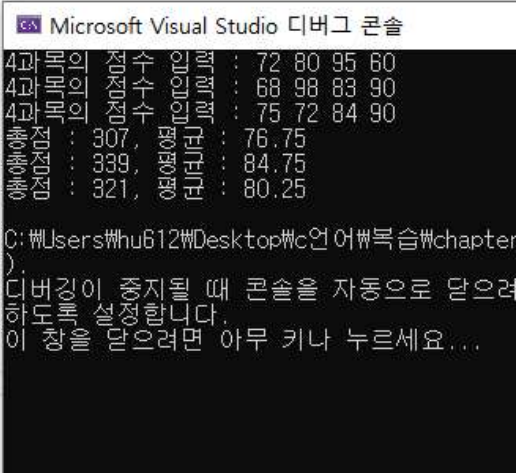
즉, 2차원 배열은 1차원 배열을 요소로 갖는 배열이다.

- 예시

```
int main(void)
{
    int score[3][4];
    int total;
    double avg;
    int i, j;

    for (i = 0; i < 3; i++)
    {
        printf("4과목의 점수 입력 : ");
        for (j = 0; j < 4; j++)
        {
            scanf("%d", &score[i][j]);
        }
    }

    for (i = 0; i < 3; i++)
    {
        total = 0;
        for (j = 0; j < 4; j++)
        {
            total += score[i][j];
        }
        avg = total / 4.0;
        printf("총점 : %d, 평균 : %.2f\n", total, avg);
    }
    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

```
4과목의 점수 입력 : 72 80 95 60
4과목의 점수 입력 : 68 98 83 90
4과목의 점수 입력 : 75 72 84 90
총점 : 307, 평균 : 76.75
총점 : 339, 평균 : 84.75
총점 : 321, 평균 : 80.25

C:\Users\whu612\Desktop\언어\책습\chapter
):
디버깅이 중지될 때 콘솔을 자동으로 닫으려
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

2차원 배열을 선언하는 방법 => int score[3][4]

첫 번째 인자는 행, 두 번째 인자는 열을 나타낸다. 따라서 score 배열은 12개의 int형 변수가 있는 셈이며, 배열명에 행 첨자와 열 첨자를 지정해 각 요소를 지목할 수 있다.

행 : 행의수 - 1, 열 : 열의수 - 1 이다. 0부터 시작하기 때문이다.

중간에 total = 0으로 초기화 시켜주었기 때문에 3번의 4과목의 점수들의 평균이 정확하게 나온 것이다.

- 메모리에서의 2차원 배열

2차원 배열은 논리적으로는 행렬의 구조를 가지고 있지만 물리적으로 1차원 배열의 형태로 메모리에 할당된다. 마치 파이썬의 flatten 함수를 이용한 것 같다.

위의 행첨자와 열첨자를 빠르게 아는 방법은 무엇일까?

행 첨자 : 1차원 배열로 계산했을 때 위치 / 열의 수

열 첨자 : 1차원 배열로 계산했을 때 위치 % 열의 수

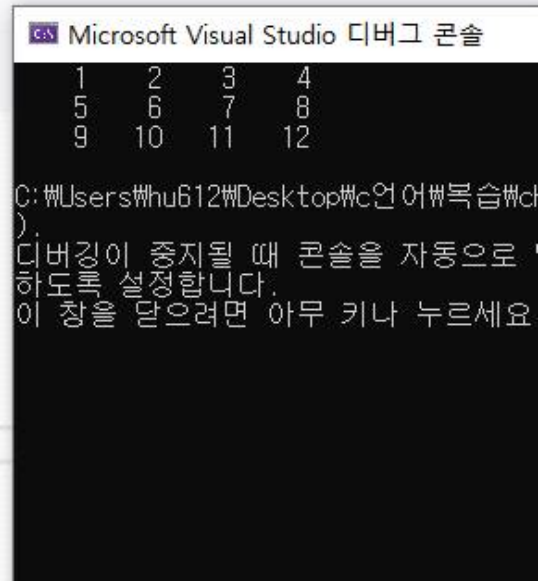
- 2차원 배열 초기화

2차원 배열을 함수 내에서 선언하면 자동 변수와 같이 메모리에 남아 있는 쓰레기 값을 지니게 된다. 따라서 배열의 저장 공간에 특정 값을 저장할 필요가 있을 때는 선언과 동시에 초기화를 해야 한다.

- 예시

```
int main(void)
{
    int num[3][4] = {
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12}
    };
    int i, j;

    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 4; j++)
        {
            printf("%5d", num[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



- 2차원 배열에서 일부 초기값 생략

```
int num[3][4] = {{1},{5,6},{9,10,11}};
```

요렇게 초기값 일부를 생략하면 0으로 자동 초기화한다.

- 행의 수 생략

```
int num[][4] = {{1},{2,3},{4,5,6}};
```

위 경우 행의 중괄호의 개수로 행의 수를 결정하므로 문제가 되지 않는다. 하지만 한 행의 크기는 열의 수로 결정하기 때문에 열의 개수는 생략될 수 없다.

- 1차원 배열의 초기화 방식으로 초기화

```
int num[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

물리적으로 1차원 배열의 나열이므로 행 초기화 괄호를 생략해도 된다.

```
int num[3][4] = {1,2,3,4,5,6};
```

이 경우 나머지 부분은 0으로 자동 초기화 한다.

- 모든 배열을 0으로 초기화

```
int num[1][3] = {0};
```

- 행의 첨자를 안적했을 때 1차원 배열 초기화 방식으로 초기화

```
int num[][4] = {1,2,3,4,5,6};
```

열의 수에 맞게 초깃값을 끊어 행의 수를 결정한다.

- 2차원 배열의 요소는 1차원 배열

따라서 부분배열명이 생긴다. 부분배열명은 1차원 배열의 배열명이며 각 행의 단위를 독립적으로 처리할 때 배열명의 역할을 수행한다.

- 2차원 char 배열

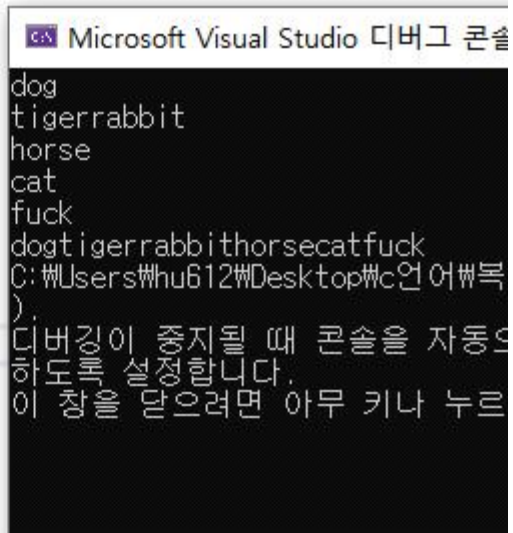
2차원의 char 배열은 여러 개의 문자열을 처리할 때 사용한다.

```
int main(void)
{
    char animal[5][20];
    int i;
    int count;

    count = sizeof(animal) / sizeof(animal[0]);
    for (i = 0; i < count; i++)
    {
        scanf("%s", animal[i]);
    }

    for (i = 0; i < count; i++)
    {
        printf("%s", animal[i]);
    }

    return 0;
}
```



긴 동물의 이름까지 넉넉하게 지정하기 위해 열 20 지정

count = 배열 전체의 크기 / 부분배열 하나의 크기 이다.

- 2차원 char 배열 초기화

char 배열을 초기화 하는 방법은 두가지이다. 첫 번째는 다른 2차원 배열처럼 배열 요소를 하나씩 초기화하는 방법이고 두 번째는 각 행의 단위를 문자열로 초기화하는 방법이다.

```

int main(void)
{
    char animal1[5][10] = {
        {'d','o','g','\0'},
        {'t','i','g','e','r','\0'},
        {'r','a','b','b','i','t','\0'},
        {'h','o','r','s','e','\0'},
        {'c','a','t','\0'}
    };

    char animal2[][10] = { "dog","tiger","rabbit","horse","cat" };
    int i;

    for (i = 0; i < 5; i++)
    {
        printf("%s", animal1[i]);
    }
    printf("\n");

    for (i = 0; i < 5; i++)
    {
        printf("%s", animal2[i]);
    }

    return 0;
}

```

Microsoft Visual Studio 디버거

dogtigerrabbithorsecat

C:\Users\whu612\Desktop\언어 디버깅이 중지될 때 콘솔을 자동으로 설정합니다. 이 창을 닫으려면 아무 키나 누르십시오.

- 3차원 배열

```

int main(void)
{
    int score[2][3][4] = {
        {{72,80,95,60}, {68,98,83,90}, {75,72,84,90}},
        {{66,85,90,88}, {95,92,88,95}, {43,72,56,75}}
    };
    int i, j, k;

    for (i = 0; i < 2; i++)
    {
        printf("%d반 점수...\n", i + 1);
        for (j = 0; j < 3; j++)
        {
            for (k = 0; k < 4; k++)
            {
                printf("%5d", score[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }

    printf("\n");
}

```

Microsoft Visual Studio 디버거 콘솔

1반 점수...

72	80	95	60
68	98	83	90
75	72	84	90

2반 점수...

66	85	90	88
95	92	88	95
43	72	56	75

C:\Users\whu612\Desktop\언어 디버깅이 중지될 때 콘솔을 자동으로 설정합니다. 이 창을 닫으려면 아무 키나 누르십시오.

3차원은 면,행,렬로 구성되어 있다.

- 마무리

- ✓ 2차원 배열의 요소는 행 첨자와 열 첨자로 쓰며 0부터 시작한다.
- ✓ 2차원 배열의 초기화는 중괄호 쌍을 두 번 사용한다.
- ✓ 2차원 배열은 주로 2중 for문으로 처리하며 행의 수와 열의 수만큼 반복한다.
- ✓ 2차원 char 배열의 초기화는 중괄호 안에 여러 개의 문자열로 초기화할 수 있다.
- ✓ 3차원 배열은 2차원 배열에 면을 더해 면, 행, 열로 이뤄진다.

구분	기능	예
2차원 int 배열	선언 방법	int ary[3][4];
	두 번째 행 세 번째 열 요소	ary[1][2];
	초기화 방법	int ary[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
2차원 char 배열	선언 방법	char ary[5][20];
	두 번째 행의 문자열	ary[1];
	초기화 방법	char ary[5][20] = {"tiger", "dog", "lion", "giraffe", "cat"};

- 확인 문제

1. 괄호의 배열명을 참고해서 다음 문제에 맞는 2차원 배열을 선언하세요.

- ① A 사는 25개의 점포에 200가지의 제품을 공급하고 있다. 각 점포에 남아 있는 재고량(stock)을 저장할 배열을 선언한다.
- ② 신체검사를 받는 50명의 좌, 우 시력(sight)을 저장할 배열을 선언한다.
- ③ 15,000개의 단어(word)를 저장할 배열을 선언한다. 단, 가장 긴 단어의 길이는 45글자이다.

2. 다음 배열의 초기화 방법 중에서 잘못된 것을 고르세요.

- ① int a[4] = {{1,1,1,1},{2,2,2,2},{3,3,3,0}};
- ② int a[] = {{1,1,1,1},{2,2,2,2},{3,3,3,0}};
- ③ int a[4] = {1,1,1,2,2,2,2,3,3,3};
- ④ char a[6] = {"apple","pear","banana"};
- ⑤ char a[] = {"apple","pear","banana"};

3. 다음 그림과 같이 2차원 배열에 문자 x가 저장되도록 반복문의 빈칸을 채웁니다.

1)

x				
	x			
		x		
			x	
				x

2)

X				X
	X		X	
		X		
	X		X	
X				X

- 답

1. ① int stock[25][200];
- ② int sight[50][2];
- ③ int word[15000][45];

2. ②,⑤ 열의 수는 공석이면 안된다.

3.

1)

```
int main(void)
{
    char mark[5][5] = { 0 };
    int i, j;
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 5; j++)
        {
            if (i==j) mark[i][j] = 'X';
        }
    }
}
```

2)

```
int main(void)
{
    char mark[5][5] = { 0 };
    int i, j;
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 5; j++)
        {
            if (i== 5-j) mark[i][j] = 'X';
        }
    }
}
```

14-2. 포인터 배열

내가 처리할 데이터가 여기저기 흩어져 있더라도 그 주소만 따로 모아 놓으면 반복문으로 모든 데이터를 쉽게 처리할 수 있다. 이때 포인터 배열이 필요하다.

- 포인터 배열 선언과 사용

포인터는 주소를 저장하는 용도로 쓰이지만 일반 변수처럼 메모리에 저장 공간을 갖는 변수이다. 따라서 같은 포인터가 많이 필요한 경우 배열을 사용하는 것이 좋다.

포인터 배열은 같은 자료형의 포인터를 모아 만든 배열이다.

<형태가 같은 포인터 3개>

```
int *pa;
```

```
int *pb;
```

```
int *pc;
```

<포인터 배열로 표현>

```
int *pary[3];
```

- 예시

```
int main(void)
{
    char* pary[5];
    int i;

    pary[0] = "dog";
    pary[1] = "elephant";
    pary[2] = "horse";
    pary[3] = "tiger";
    pary[4] = "lion";

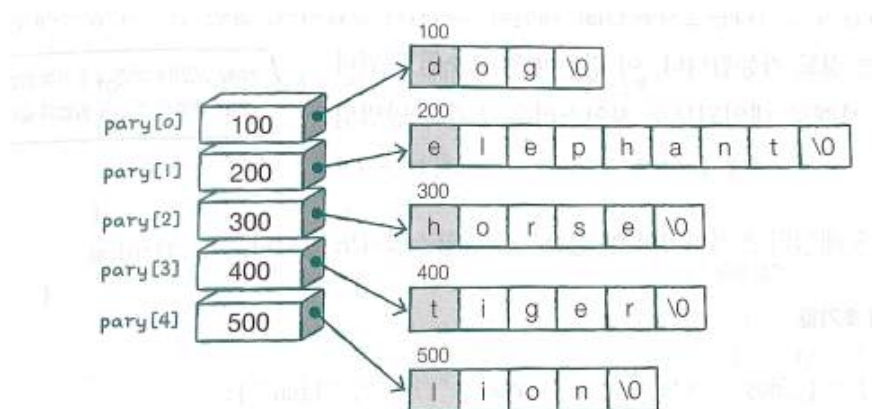
    for (i = 0; i < 5; i++)
    {
        printf("%s\n", pary[i]);
    }

    return 0;
}
```

Microsoft Visual Studio C

```
dog
elephant
horse
tiger
lion

C:\Users\whu612\Desktop\
);
디버깅이 중지될 때 콘솔
하도록 설정합니다.
이 창을 닫으려면 아무 키
```

포인터 배열을 선언했기 때문에 각 포인터 배열에 문자열 상수를 대입한다. 문자열 상수는 변경이 불가능한 메모리 영역에 보관되고 포인터 배열에는 그 첫 번째 문자의 주소가 저장된다.

- 포인터 배열의 초기화

`char` 포인터 배열의 초기화는 2차원 `char` 배열의 초기화와 같다. 선언과 동시에 문자열 상수로 초기화하는 것도 가능하다.

<char 포인터 배열 초기화>

```
char *pary[5] = {"dog", "elephant", "horse", "tiger", "lion"};
```

<2차원 char 배열 초기화>

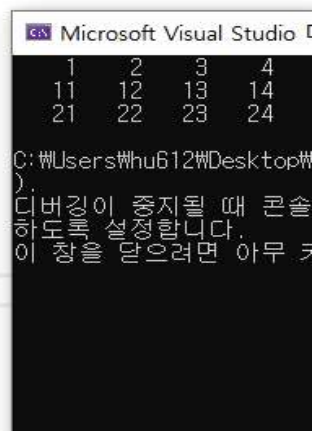
```
char animal[5][20] = {"dog", "elephant", "horse", "tiger", "lion"};
```

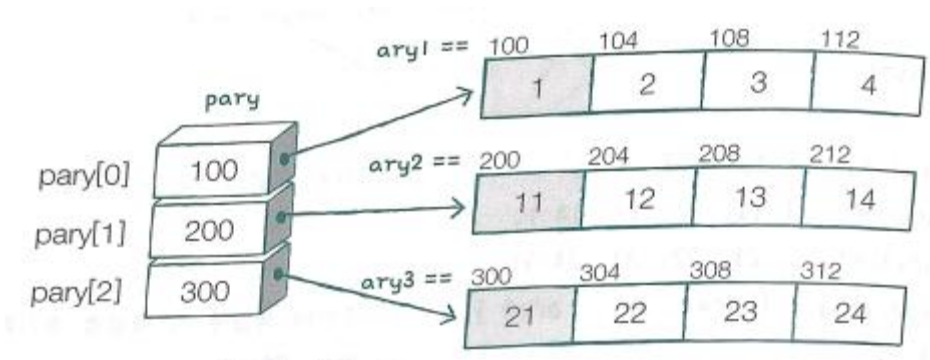
- 2차원 배열처럼 활용하는 포인터 배열

```
int main(void)
{
    int ary1[4] = { 1,2,3,4 };
    int ary2[4] = { 11,12,13,14 };
    int ary3[4] = { 21,22,23,24 };
    int* pary[3] = { ary1, ary2, ary3 };
    int i, j;

    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 4; j++)
        {
            printf("%5d", pary[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```





- 포인터 배열을 2차원 배열처럼 사용할 수 있는 이유
 => 포인터 연산이다.

포인터는 자신이 가리키는 변수의 형태를 알고 있으므로 정수 연산을 통해 원하는 위치를 찾아갈 수 있다. 따라서 포인터 배열은 포인터 연산을 통해 2차원 배열처럼 쓸 수 있다.

<배열 표현>

pary[2][2];

<포인터 표현>

*(pary[2]+2); = *(300+(2*(sizeof(int)))) = *(308)

- 마무리

- ✓ 포인터 배열을 선언하고 사용하는 방법은 일반 배열과 같다.
- ✓ char 포인터 배열을 이용하면 여러 개의 문자열을 다루기에 편하다.
- ✓ 포인터 배열을 사용하면 1차원 배열을 모아 2차원 배열처럼 쓸 수 있다.

	char 포인터 배열	int 포인터 배열
선언 방법	char *pary[5];	int *pary[3];
요소의 사용	pary[0] = "apple";	int ary[4]; pary[0] = ary;
초기화 방법	char *ary[5] = {"tiger", "dog", "lion", "giraffe", "cat"};	int ary1[4], ary2[4], ary3[4]; int *pary[3] = {ary1, ary2, ary3};

- 확인 문제

1. 다음 5개의 문자열 상수를 저장하기 위한 포인터 배열을 선언하고 초기화하세요.
 "apple", "pear", "peach", "banana", "melon"
2. 다음 프로그램의 실행결과를 적어보세요.

```

int main(void)
{
    char a[4][10] = { "horse", "fox", "hippo", "tiger" };
    char* pa[] = { a[0], a[1], a[2], a[3] };
    int count;
    int i;
    count = sizeof(pa) / sizeof(pa[0]);

    for (i = 0; i < count; i++)
    {
        printf("%c", pa[i][i]);
    }

    return 0;
}

```

3. 다음과 같은 코드가 있을 때 보기 중에서 배열 요소의 사용이 잘못된 것을 고르세요.

```

char *pa = "apple";
char ary[] = "banana";
char *pary[4];

```

- ① pary[1] = pa;
- ② pary[2] = ary;
- ③ pary[3] = "mango";
- ④ pary[4] = "orange";

- 정답

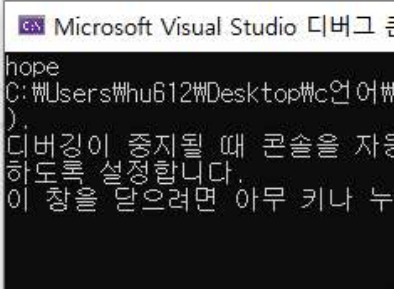
- 1. char *pa[5] = {"apple", "pear", "peach", "banana", "melon"}
- 2.

```
#include <stdio.h>

int main(void)
{
    char a[4][10] = { "horse", "fox", "hippo", "tiger" };
    char* pa[] = { a[0], a[1], a[2], a[3] };
    int count;
    int i;
    count = sizeof(pa) / sizeof(pa[0]);

    for (i = 0; i < count; i++)
    {
        printf("%c", pa[i][i]);
    }

    return 0;
}
```



The screenshot shows the Microsoft Visual Studio debugger console. The output of the program is 'hope'. Below the output, there is a warning message in Korean: '디버깅이 중지될 때 콘솔을 자동으로 설정합니다. 이 창을 닫으려면 아무 키나 누르십시오.' (When debugging is stopped, the console is automatically set. Press any key to close this window.)

3. ④ pary[4]는 존재하지 않음