



혼.공.C Chapter14

2021210088 허지혜

목차

1. 2차원 배열
2. 3차원 배열
3. 포인터 배열

2차원 배열 선언과 요소 사용

```
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;
}
```

2차원 배열

: 형태가 같은 배열을 여러 개 모아 만든 배열
1차원 배열을 요소로 갖는 배열이다.

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
```

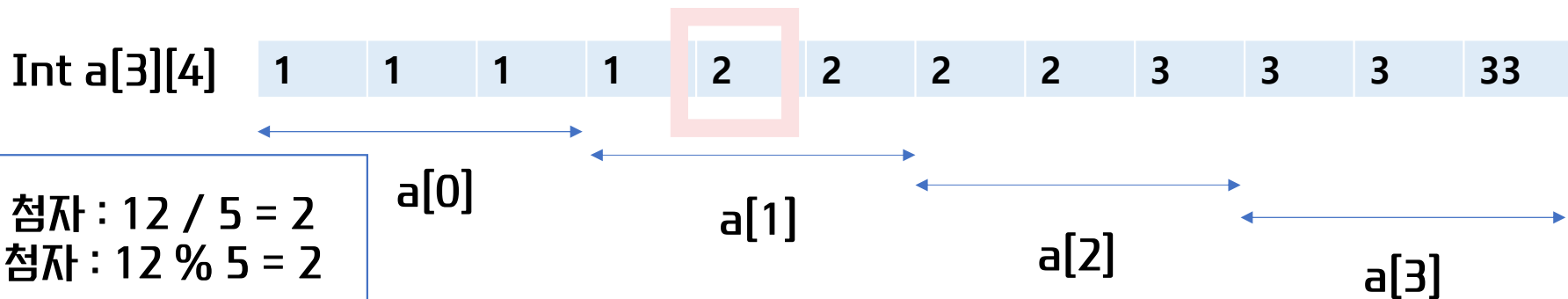
C:\Users\whu612\Desktop\c언어\복습\chapter
)
디버깅이 중지될 때 콘솔을 자동으로 닫으려
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

2차원 배열 선언과 요소 사용

2차원 배열 선언 방법 “int[행][열]”

2차원 배열은 행렬의 구조를 가지고 있지만 물리적으로 1차원 형태로 배열에 할당된다.
파이썬 flatten 함수를 이용한 것과 같다.
따라서 1차원 배열로 바뀌었을 때 행 첨자와 열 첨자의 수를 아는 방법은 다음과 같다.

행 첨자 : 1차원 배열로 계산했을 때 위치 / 열의 수
열 첨자 : 1차원 배열로 계산했을 때 위치 % 열의 수

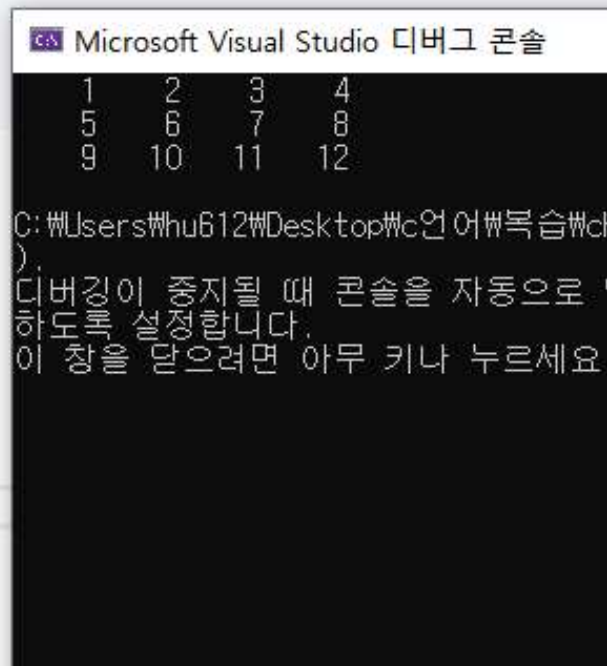


행 첨자 : $12 / 5 = 2$
열 첨자 : $12 \% 5 = 2$

2차원 배열 초기화

```
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;
}
```



Microsoft Visual Studio 디버그 콘솔

```
1 2 3 4
5 6 7 8
9 10 11 12
```

C:\Users\whu612\Desktop\c언어\복습\cd...
)
디버깅이 중지될 때 콘솔을 자동으로
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요

2차원 배열을 함수 내에서 선언하면 메모리에 남아 있는 쓰레기 값을 지니게 된다.
따라서 배열 저장 공간에 특정 값을 저장할 때는 선언과 동시에 초기화를 해야 한다.

2차원 배열 초기화

- ① 2차원 배열에서 일부 초기값을 생략
- ② 행의 수 생략
- ③ 1차원 배열의 초기화 방식으로 초기화
- ④ 모든 배열을 0으로 초기화
- ⑤ 행의 첨자를 안적었을 때 1차원 배열 초기화 방식으로 초기화

- ① `int num[3][4] = {{1},{5,6},{9,10,11}};`
- ② `int num[][4] = {{1},{2,3},{4,5,6}};`
- ③ `int num[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};`
- ③ `int num[3][4] = {1,2,3,4,5,6};`
- ④ `int num[1][3] = {0};`
- ⑤ `int num[][4] = {1,2,3,4,5,6};`

- 2차원 배열의 요소는 1차원 배열

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

2차원 char 배열

```
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;
}
```

Microsoft Visual Studio 디버그 콘솔

```
dog
tigerrabbit
horse
cat
fuck
dogtigerrabbithorsecatfuck
C:\Users\hu612\Desktop\언어책
)
디버깅이 중지될 때 콘솔을 자동으로
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르
```

2차원 char 배열 초기화

Char 배열을 초기화 하는 방법은 두가지이다.

- 1) 다른 2차원 배열처럼 배열 요소를 하나씩 초기화한다.
- 2) 각 행의 단위를 문자열로 초기화하는 방법이다.

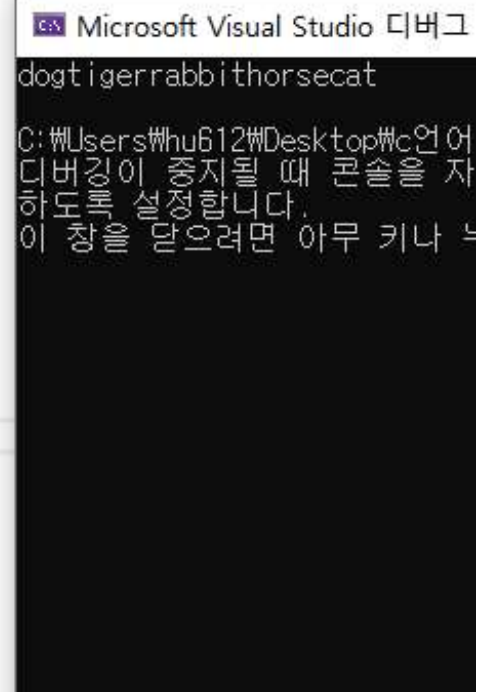
```
int main(void)
{
    char animal1[5][10] = {
        {'d','o','g','\0'},
        {'t','i','g','e','r','\0'},
        {'r','a','b','b','i','t','\0'},
        {'h','o','r','s','e','\0'},
        {'c','a','t','\0'}
    };

    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 디버깅

dogtigerrabbithorsecat

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

3차원 배열

3차원 배열은 3가지 요소로 이루어져 있다.
면, 행, 열

```
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 디버그 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\c언어책
)
디버깅이 중지될 때 콘솔을 자동으로
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르

다차원 배열 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.

다차원 배열 마무리

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

확인 문제 1

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. `int stock[25][200];`

② 신체검사를 받는 50명의 좌, 우 시력(sight)을 저장할 배열을 선언한다. `int sight[50][2];`

③ 15,000개의 단어(word)를 저장할 배열을 선언한다. 단, 가장 긴 단어의 길이는 45글자이다.
`int sight[15000][45];`

확인 문제 2

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"};`

②,⑤ 열의 수는 공석이면 안된다.

확인 문제 3

3-1. 다음 그림과 같이 2차원 배열에 문자 x가 저장되도록 반복문의 빈칸을 채웁니다.

X				
	X			
		X		
			X	
				X

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

확인 문제 3

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

X				X
	X		X	
		X		
	X		X	
X				X

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

포인터 배열

포인터는 주소를 저장하는 용도로 쓰이지만 일반 변수처럼 메모리에 저장 공간을 갖는 변수이다.
따라서 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];
```


포인터 배열

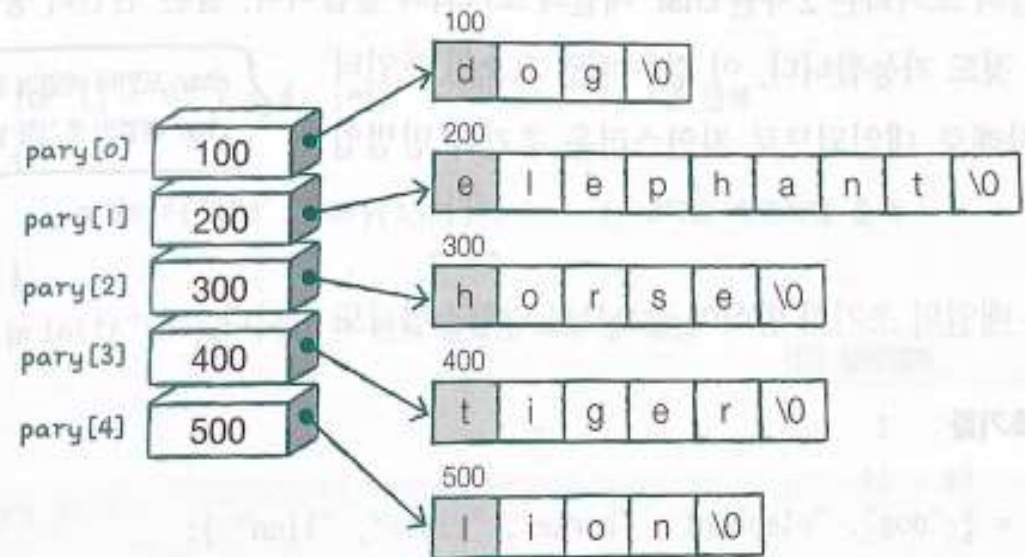
```
int main(void)
{
    char* pary[5];
    int i;

    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 \
dog
elephant
horse
tiger
lion
C:\Users\WuhuB
디버깅이 중지
하도록 설정함
이 창을 닫으
```



포인터 배열을 선언하여 각 포인터 배열에 문자열 상수를 대입한다.

문자열 상수는 변경이 불가능한 메모리 영역에 보관되고 첫 번째 문자의 주소가 된다.

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

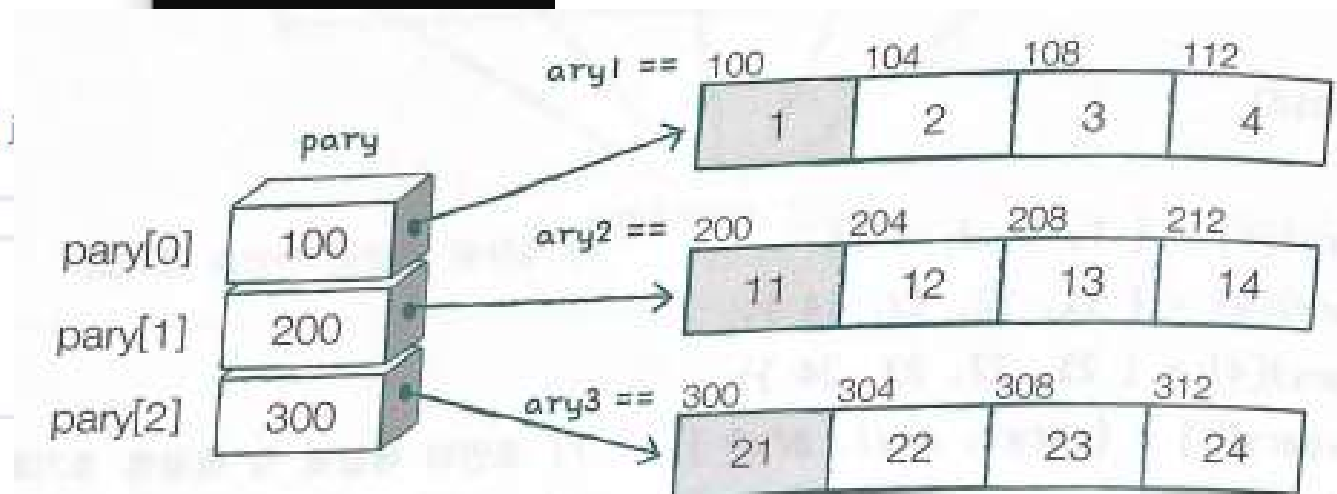
```
int main(void)
{
    int ary1[4] = { 1,2,3,4 };
    int ary2[4] = { 11,12,13,14 };
    int ary3[4] = { 21,22,23,24 };
    int* pary[3] = { ary1, ary2, ary3 };
    int i, j;

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

    return 0;
}
```

Microsoft Visual Studio

1	2	3	4
11	12	13	14
21	22	23	24



포인터 배열을 2차원 배열처럼 사용할 수 있는 이유

포인터 연산 때문이다.

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)`

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

확인 문제 1

1. 다음 5개의 문자열 상수를 저장하기 위한 포인터 배열을 선언하고 초기화하세요.

“apple”, “pear”, “peach”, “banana”, “melon”

```
char *pa[5] = {"apple", "pear", "peach", "banana", "melon"}
```

확인 문제 2

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;
}
```

hope

확인 문제 3

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";

④ Pary[4]는 존재하지 않는다.

끝