
# 파이썬 - HW1

임베디드스쿨2기

Lv2과정

2020. 04. 16

박태

# 1. 배열 (1)

1) 배열을 사용하는 이유

- int 형 변수가 1000개 필요하면 ???
  - ㄴ 일일히 int a, b, c, d, e, f, g, h, ... zzz 까지 적기도 힘들다.
  - ㄴ 여러개의 변수를 한번에 선언한다.

- 배열 선언 방법
  1. 먼저 다발로 **활용할 데이터 타입**을 적는다.
  2. 변수명이 있듯이 **배열의 이름**을 적는다.
  3. **얼마만큼의 공간**을 활용할지 **숫자를 대괄호 내부에** 적는다.
     (여기서 대괄호를 비워두면 입력되는 요소에 따라 자동으로 개수가 정해진다)
     입력이 없을 경우엔 문제가 될 수 있으니
     **필요한 개수를 설정해놓는 것을 권장한다.**
     혹은 입력할 데이터를 미리 설정해놓는것도 좋다.

- 선언 예시

```
// arr
// [0]   [1]   [2]
//  1     2     3
// 배열의 시작 인덱스는 0부터 시작하므로 주의해야 한다.
// 선언할 때는 사용할 개수를 적지만
// 활용할 때는 적은 개수 - 1까지 활용이 가능하다는 것을 주의하라!
int arr[] = { 2, 4, 7 };
```

# 1. 배열 (2)

```
int arr[] = { 2, 4, 7 };
int len = sizeof(arr) / sizeof(int);

printf("arr len = %d\n", len);

printf("arr:\n");

for (i = 0; i < len; i++)
{
        printf("%2d", arr[i]);
}

printf("\n");

return 0;
```

2, 4, 7 값을 가지는 int 형 배열을 선언한다.
배열의 길이는 배열 크기 / 배열 형태로 구한다.

배열의 길이를 프린트하고,

For 문을 통해 배열 길이 만큼 반복문을 실행해서

배열의 값을 프린트 한다.

```
arr len = 3
arr:
 2 4 7
```

# 2. Continue

```c
#include <stdio.h>

int main(void)
{
        int i, num;

        printf("1 ~ n까지 출력합니다. (n을 선택하세요): ");
        scanf("%d", &num);

        for (i = 1; i <= num; i++)
        {
                if (!(i % 3))
                {
                        // 다시 위로 돌아감(증감부를 수행하게됨)
                        // 결국 아래의 printf를 실행하지 않고 스킵하게 됨
                        continue;
                }

                printf("i = %3d\n", i);
        }

        return 0;
}
```

```
1 ~ n까지 출력합니다. (n을 선택하세요): 9
i =   1
i =   2
i =   4
i =   5
i =   7
i =   8
```

Continue 명령어는 반복문을
실행하지 않고 다시 조건으로 돌아가는
명령어 이다.

왼쪽의 코딩을 보면
n항의 숫자로 입력 받고,

그 항 만큼 for 문을 통해 반복을 하게 된다.

그 와중에 if문은 3의 배수가 나오면
참 값이 되어 들어가게 되고

continue를 만나 아래의
printf는 실행하지 않게 됩니다.

# 3. 이중 배열 (1)

이중배열에서 주의 할 점은
이중 배열은 실제 **이들은 차원을 가지고 있지 않다는 점**이다.
이중 배열은 순차적으로 배치 되어 있게 된다.

예를 들어 **int arr[2][2]** 라는 배열이 있다면

| [0] | [1] |
| --- | --- |
| [0][0] [0][1] | [1][0] [1][1] |

또 다른 예로 **int arr[3][3]**

| [0] | [1] | [2] |
| --- | --- | --- |
| [0][0] [0][1] [0][2] | [1][0] [1][1] [1][2] | [2][0] [2][1] [2][2] |

하나더 하면 int **arr[2][4]**

| [0] | [1] |
| --- | --- |
| [0][0] [0][1] [0][2] [0][3] | [1][0] [1][1] [1][2] [1][3] |

# 3. 이중 배열 (2)

```
int i, j;
// 이중 배열
// 실제 이들은 차원을 가지고 있지 않으며 순차적으로 배치되어 있다.

//      [0]          [1]
// [0][0] [0][1] [1][0] [1][1]
//   0      20     10     30
int arr[2][2];

//          [0]                 [1]                 [2]
// [0][0] [0][1] [0][2] [1][0] [1][1] [1][2] [2][0] [2][1] [2][2]
int arr2[3][3];

//             [0]                         [1]
// [0][0] [0][1] [0][2] [0][3] [1][0] [1][1] [1][2] [1][3]
int arr3[2][4];
```

```
for (i = 0; i < 2; i++)
{
        for (j = 0; j < 2; j++)
        {
                arr[i][j] = i * 10 + j * 20;
                printf("arr[%d][%d] = %d\n", i, j, arr[i][j]);
        }
}

return 0;
```

arr[i][j]
의 배열 구조를 for문 두개를 만들어서
출력하는 구조 이다.

위의 int arr[2][2]의 구조에
각각의 값이 0 20 10 30 이 들어 가게 된다.

```
arr[0][0] = 0
arr[0][1] = 20
arr[1][0] = 10
arr[1][1] = 30
```

# 4. 더블 포인터

```c
#include <stdio.h>

int main(void)
{
        int num = 3;
        int *p_num = &num;
        int **pp_num = &p_num;

        // p7_num;

        printf("num = %d\n", num);
        printf("*p_num = %d\n", *p_num);
        printf("**pp_num = %d\n", **pp_num);

        printf("&num = 0x%x\n", &num);
        printf("p_num = 0x%x\n", p_num);

        printf("&p_num = 0x%x\n", &p_num);
        printf("pp_num = 0x%x\n", pp_num);

        printf("&pp_num = 0x%x\n", &pp_num);

        return 0;
}
```

```
num = 3
*p_num = 3
**pp_num = 3
&num = 0xb835694
p_num = 0xb835694
&p_num = 0xb835698
pp_num = 0xb835698
&pp_num = 0xb8356a0
```

Num = 3 초기화
***p_num은 num 변수의 주소 값**
****pp_num은 p_num 포인터 변수의 주소값**이다.

printf의 *p_num은 p_num 포인터 변수가 가르키는 주소의 값을 출력한다.

마찬가지로 **p_num은 p_num의 포인터 변수가 가르키는 주소의 값 = num의 주소 값 이므로 다시 한번 들어가면 num이 가르키는 값을 출력하게 된다.