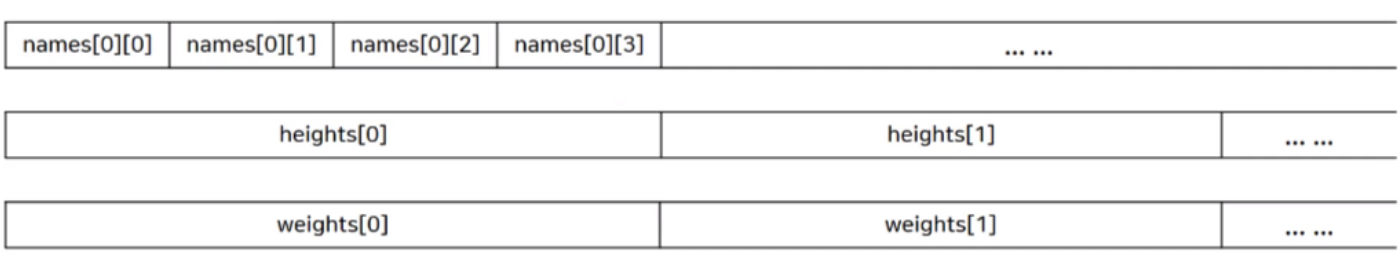
```c
/*
Patient
- name (string)
- height (cm)
- weight (kg)
*/

/* Variables */

char name1[MAX_NAME], name2[MAX_NAME], name3[MAX_NAME];// ... and more
float height1, height2, height3;// ... and more
float weight1, weight2, weight3;// ... and more

strcpy(name1, "John Wick");
height1 = 180;
weight1 = 90;

printf("%s %f %f\n", name1, height1, weight1);
```

- 환자 데이터 관리 프로그램
- 많은 수의 환자 데이터를 처리해야함 -> 환자 수에 따라 변수가 늘어남 -> 좋지 않음

```c
/* Arrays */

char names[MAX_PATIENTS][MAX_NAME];
float heights[MAX_PATIENTS];
float weights[MAX_PATIENTS];

strcpy(names[0], "John Wick");
heights[0] = 180.0f;
weights[0] = 90.0f;

printf("%s %f %f\n", names[0], heights[0], weights[0]);
```

배열은 자료형이 같은 데이터 오브젝트들이 나열된 형태

자료형이 서로 다르지만
함께 사용하면 편리한 데이터 오브젝트들 끼리
모아놓을 수는 없을까?

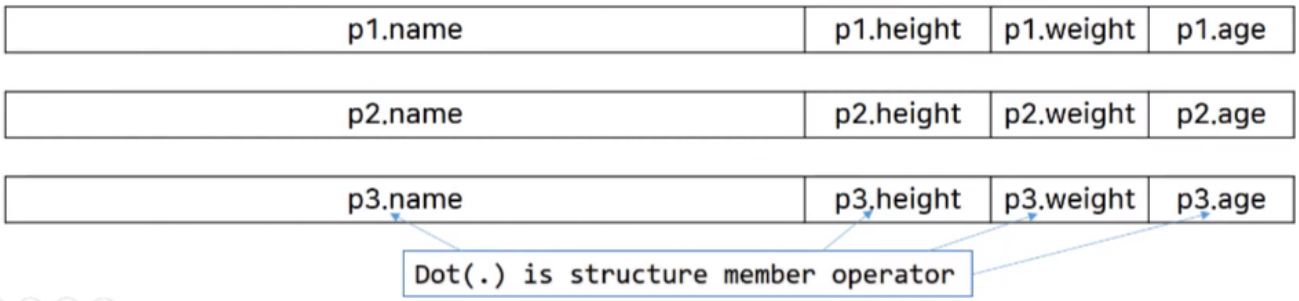


- 배열 사용
- 그러나 배열의 수도 늘어나고, 데이터가 찾을 때 속도 및 찾는 방법의 문제가 있을 수 있음

```c
/* Structures */

struct Patient
{
    char name[MAX_NAME];
    float height;
    float weight;
    int age;
};

struct Patient p1, p2, p3; //structure variables
//struct Patient pat[MAX_PARTIENTS];
```

```c
strcpy(p1.name, "John Wick");
p1.height = 180;
p1.weight = 90;

strcpy(p2.name, "Dwayne Johnson");
p2.height = 180;
p2.weight = 120;
```

* 메모리 패딩(padding)에 대해서는 뒤에 나와요



- 구조체 사용
- 여러 데이터들이 복합적으로 합쳐져 사용자가 직접 만든 자료형인 것처럼 사용가능