구조체 (Struct)

구조체의 정의와 typedef 선언

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

typedef int INT;
typedef int* PTR_INT;

int main() {
    INT num;      // == int num;
    INT* ptr1;    // == int* ptr1;
    PTR_INT ptr2; // == int* ptr2;
}
```

- `typedef`? → type에 이름을 정의하겠다.
  - 대상 자료형에 새로 원하는 이름을 부여할 수 있음

```
struct point {
    int xpos;
    int ypos;
};
typedef struct point Point;

typedef struct point {
    int xpos;
    int ypos;
} Point;
```

- 구조체의 선언과 `typedef`
  - 구조체 point 정의 후 `struct point`에 `Point`라는 이름을 부여
  - 구조체 정의와 `typedef` 선언을 함께 할 수 있음
    - 이 경우 `point` 생략 가능 → 왜?

함수로의 구조체 변수 전달과 반환

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

typedef struct point {
    int xpos;
    int ypos;
    char name[100];
} Point;

void ShowPosition(Point pos) {
    printf("%s : [%d, %d]\n", pos.name, pos.xpos, pos.ypos);
}

Point GetCurrentPosition() {
    Point cen;
    printf("Input current pos and name: ");
    scanf("%d %d %s", &cen.xpos, &cen.ypos, cen.name);
    return cen;
}

int main() {
    Point curPos = GetCurrentPosition();
    ShowPosition(curPos);
    return 0;
}
```

- 함수의 인자로 전달될 수 있음
- 함수의 return문에 의해 반환 될 수 있음
- 배열까지도 복사 되어서 전달 할 수 있음

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

typedef struct point {
```

```
    int xpos;
    int ypos;
} Point;

void Trans(Point* ptr) {
    ptr->xpos = (ptr->xpos) * -1;
    ptr->ypos = (ptr->ypos) * -1;
}

void ShowPoint(Point pos) {
    printf("[%d, %d]\n", pos.xpos, pos.ypos);
}

int main() {
    Point* pos =(Point*)malloc(sizeof(Point));
    pos->xpos=7;
    pos->ypos=-5;
    Trans(&pos);
    ShowPoint(*pos);
    Point pos2 = *pos;
    ShowPoint(pos2);
    //pos-pos2;ㅡ ->안됨
    free(pos);
    return 0;
}
```

- 구조체 기반의 Call-by-reference 예제임
- 구조체는 복사가 가능함
- 복사는 가능하지만 사칙연산 같은 기능은 없음

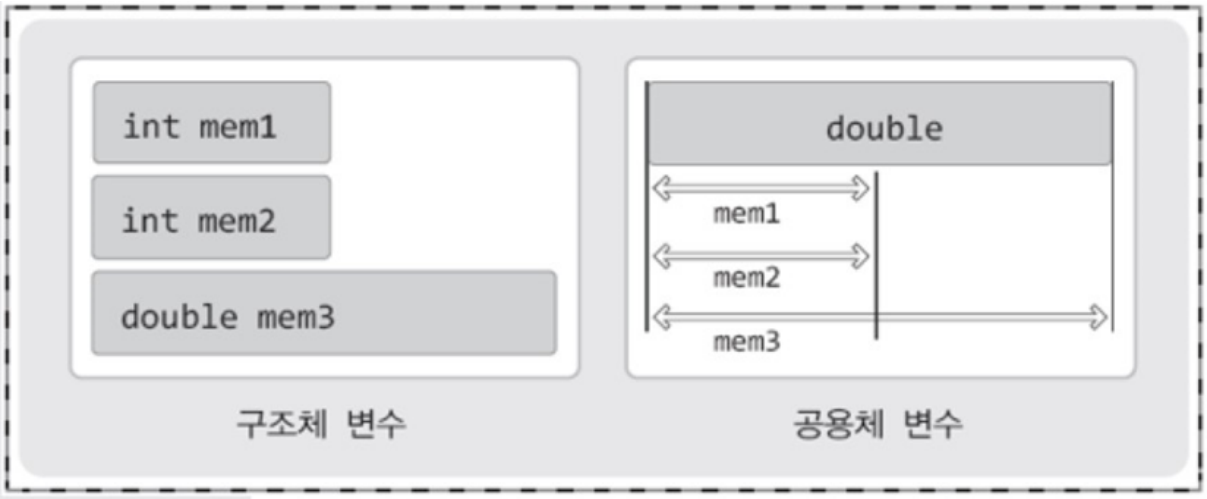
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>

struct comgong {
char name[20];
char id[20];
```

```
};

int main(void) {
    int n;
    scanf("%d", &n);
    struct comgong* arr = (struct comgong*)malloc(sizeof(st
ruct comgong) * n);
    for (int i = 0; i < n; i++) {
    scanf("%s %s", arr[i].name, arr[i].id);
    }
    for (int i = 0; i < n; i++) {
        printf("%s %s\\n", arr[i].name, arr[i].id);
    }
    free(arr);
    return 0;
}
```

공용체와 열거형

```
typedef struct sbox {
    int mem1;
    int mem2;
    double mem3;
} SBox;

typedef union ubox {
    int mem1;
    int mem2;
    double mem3;
} UBox;
```

- 공용체의 선언은 struct를 union으로 바꾸기만 하면 됨
- 구조체와 공용체는 메모리적 차이가 있음



- 공용체를 사용하는 이유
  - 하나의 메모리 공간을 여러 방식으로 접근할 수 있음
  - 메모리는 공유하고 해석은 다르게 할 수 있음

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

typedef enum syllable {
    Do = 1, Re = 2, Mi = 3, Fa = 4
} Syllable;

void Sound(Syllable sy) {
    switch (sy)
    {
    case Do:
        printf("도"); return;
    case Re:
        printf("레"); return;
    case Mi:
        printf("미"); return;
    case Fa:
        printf("파"); return;
    default:
        break;
    }
```

```
}

int main() {
    Syllable tone;
    for (tone = Do; tone <= Fa; tone += 1) {
        Sound(tone);
    }
    return 0;
}
```

- 열거형의 선언은 struct를 enum으로 바꾸기만 하면 됨
- Do를 1을 의미하는 상수로 정의함
    - 이 값은 Syllable 변수에 저장이 가능함
    - Do, Re, Mi, Fa를 열거형 상수라고 함
    - 열거형 상수는 선언 이후 어디서든 쓸 수 있는 상수가 됨