

C Programming Assignment / Week 12

정리노트 #8

구조체

- 서로 다른 자료형을 하나의 자료로 구조화한 자료형이다.
- 키워드 **struct** 를 사용하여 정의하고, **중괄호 안에 정의하며** 마지막 중괄호 다음엔 **세미콜론**을 붙여야 한다.
- **struct** 안의 각 구성요소는 **멤버**라 칭한다.

```
1  #include <stdio.h>
2
3  struct list {
4      char name; // 문자형 멤버
5      char gender; // 문자형 멤버
6      int age; // 정수형 멤버
7  }; // list 구조체의 정의
8
9  int main(void)
10 {
11     struct list st1 = { 'T', 'M', 25 }; // list 구조체의 선언
12
13     printf("구조체 list의 크기는 %d이다.\n", sizeof(struct list));
14     printf("구조체 객체 st1의 크기는 %d이다.\n", sizeof(st1));
15
16     return 0;
17 }
```

Microsoft Visual Studio 디버그 콘솔

구조체 list의 크기는 8이다.
구조체 객체 st1의 크기는 8이다.

구조체 변수의 선언은 아래의 사진처럼 int main()이 아닌 struct list에서 정의와 선언을 같이 할 수 있다. 또한, 초기화도 할 수 있다.

```
1  #include <stdio.h>
2
3  struct list {
4      char name; // 문자형 멤버
5      char gender; // 문자형 멤버
6      int age; // 정수형 멤버
7  } st1 = { 'T', 'M', 25 }; // list 구조체의 정의 및 선언
8
9  int main(void)
10 {
11
12     printf("구조체 list의 크기는 %d이다.\n", sizeof(struct list));
13     printf("구조체 객체 st1의 크기는 %d이다.\n", sizeof(st1));
14
15     return 0;
16 }
```

Microsoft Visual Studio 디버그 콘솔

구조체 list의 크기는 8이다.
구조체 객체 st1의 크기는 8이다.

구조체의 접근과 사용

- 구조체의 멤버 변수 접근하려면 도트 연산자를 사용하여야 한다.
- 위의 사진을 예로 멤버 변수의 형식을 예로 들면, st1.name, st1.gender, st1.age 가 된다.
- 구조체 변수의 정의, 선언 후 도트 연산자를 사용한다.
- 구조체에서는 할당 연산자도 사용이 가능하다.

```
1  #include <stdio.h>
2
3  struct list {
4      char name; // 문자형 멤버
5      char gender; // 문자형 멤버
6      int age; // 정수형 멤버
7  } st1, st2; // list 구조체의 정의 및 선언
8
9  int main(void)
10 {
11     // 도트 연산자를 이용
12     // name, gender, age에 J, M, 24 대입 후 할당 연산자를 이용한 st2값을 st1에 할당함.
13     st1.name = 'J';
14     st1.gender = 'M';
15     st1.age = 24;
16     struct list st2 = {'A', 'B', 240 };
17     st1 = st2;
18     printf("name gender age\n");
19     printf("%c %c %d\n", st1.name, st1.gender, st1.age);
20
21     return 0;
22 }
```

Microsoft Visual Studio 디버그 콘솔

name gender age
A B 240

구조체의 재정의

- typedef 을 이용해 구조체를 재정의 할 수 있다.
- 아래의 사진을 예로 들면 주석에 설명한대로 list 라는 구조체를 student 로 재정의했다.
- 구조체를 재정의 한 후에 객체를 선언할 땐 재정의한 자료형 + 객체로 선언한다.

```
#include <stdio.h>

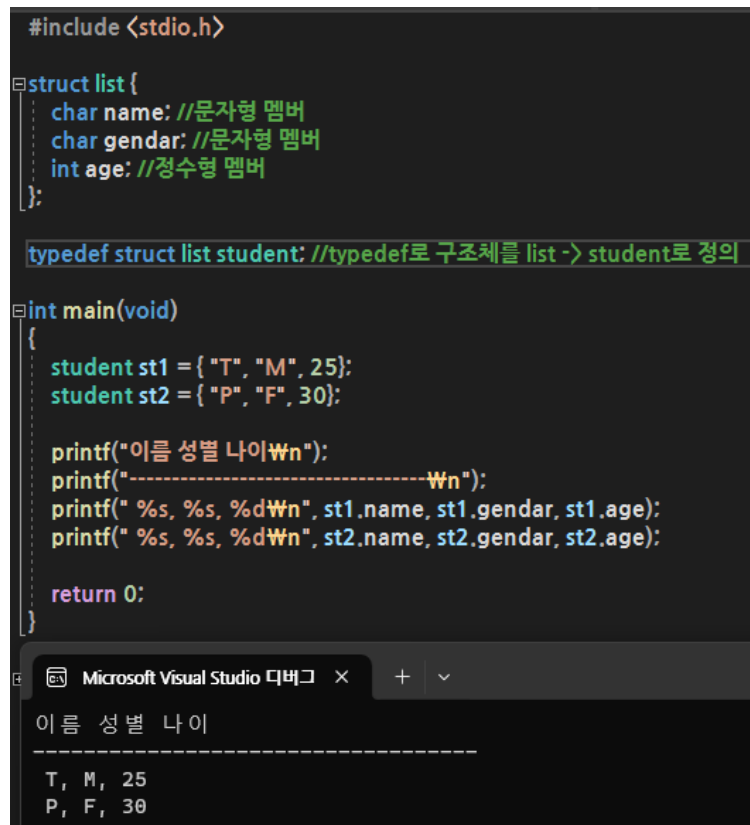
struct list {
    char name: //문자형 멤버
    char gendar: //문자형 멤버
    int age: //정수형 멤버
};

typedef struct list student: //typedef로 구조체를 list -> student로 정의

int main(void)
{
    student st1 = { "T", "M", 25};
    student st2 = { "P", "F", 30};

    printf("이름 성별 나이\n");
    printf("-----\n");
    printf(" %s, %s, %d\n", st1.name, st1.gendar, st1.age);
    printf(" %s, %s, %d\n", st2.name, st2.gendar, st2.age);

    return 0;
}
```



The screenshot shows a C program in Visual Studio. The code defines a struct named 'list' with three members: 'name' (char), 'gendar' (char), and 'age' (int). It then uses 'typedef' to create an alias 'student' for the 'list' struct. In the 'main' function, two 'student' objects, 'st1' and 'st2', are initialized with values. The program uses 'printf' to print the details of these objects, with a header line '이름 성별 나이' and a separator line '-----'. The output in the console shows the details for 'st1' (T, M, 25) and 'st2' (P, F, 30).

(구조체를 이용한 3 차원 정리?는 하실지 모르겠네요)

구조체의 사용

- 포인터를 이용해 구조체를 사용할 수 있다.

일반 변수 형태로 정의한 구조체의 멤버인 도트(.)연산자가 있다면

포인터 변수 형태로 정의하려면 화살표(->) 연산자로 정의한다.

```
#include <stdio.h>

struct ThreeDime {
    double x: //실수형 멤버
    double y: //실수형 멤버
    double z: //실수형 멤버
};

typedef struct ThreeDime ThreeDime; // typedef로 구조체 재정의

int main(void)
{
    ThreeDime A1 = { 3, 1, 8};

    ThreeDime* pA1 = &A1; //포인터 선언과 초기화

    printf("3차원 점 A1의 x: %.2lf, y: %.2lf, z: %.2lf이다.\n", pA1->x, pA1->y, pA1->z);
    //pA1에 x,y,z좌표의 주소 값을 대입 및 출력
    return 0;
}
```

Microsoft Visual Studio 디버그 × + ▾

3차원 점 A1의 x : 3.00, y:1.00, z:8.00이다.

↑ 위의 코드는 typedef 를 사용한 재정의 + 포인터 구조체를 사용한 예시이다.

(사소한 설명은 주석으로 설명)

열거형(enum)

- 순서가 있고 새로운 값(상수)를 가질 수 있는 사용자 정의 자료형

Ex)

```
#include <stdio.h>

int main(void)
{
    enum { yellow, red, blue, green } color; //enum {열거형 상수 리스트} 열거형 태그로 정의
    printf("원하는 색을 입력하세요.\n");
    printf("0번 : 노란색, 1번 : 빨간색\n");
    printf("2번 : 파란색, 3번 : 초록색\n");
    scanf_s("%d", &color);

    if (color == yellow) printf("노란색입니다.\n");
    else if (color == red) printf("빨간색입니다.\n");
    else if (color == blue) printf("파란색입니다.\n");
    else if (color == green) printf("초록색입니다.\n");

    return 0;
}
```

Microsoft Visual Studio 디버그 × + ▾

원하는 색을 입력하세요.
 0번 : 노란색, 1번 : 빨간색
 2번 : 파란색, 3번 : 초록색
 3
 초록색입니다.

- 예제에 보인 것처럼 enum {열거형 상수 리스트} 열거형 태그 순으로 정의할 수 있다.

(반대로 적어도 정의가 가능하다)

+ 만약 특정 열거 리스트 상수를 N으로 고정하고 싶다면 그 열거형에 특정 리스트의 상수 뒤에 = N을 써 선언할 수 있다.)

Ex)

```
#include <stdio.h>

int main(void)
{
    enum { yellow, red, blue=7, green } color; //enum {열거형 상수 리스트} 열거형 태그로 정의
    printf("원하는 색을 입력하세요.\n");
    printf("0번 : 노란색, 1번 : 빨간색\n");
    printf("7번 : 파란색, 8번 : 초록색\n"); //열거형 리스트 상수에서 7을 파란색으로 고정
    scanf_s("%d", &color);

    if (color == yellow) printf("노란색입니다.\n");
    else if (color == red) printf("빨간색입니다.\n");
    else if (color == blue) printf("파란색입니다.\n");
    else if (color == green) printf("초록색입니다.\n");

    return 0;
}
```

Microsoft Visual Studio 디버그 × + ▾

원하는 색을 입력하세요.
0번 : 노란색, 1번 : 빨간색
7번 : 파란색, 8번 : 초록색
3

- 하지만 그 사이에 있는 리스트들은 정의가 되지 않았기에 공백으로 나온다

그럼 그 다음 리스트를 출력하려면 어떻게 해야하나?

```
#include <stdio.h>

int main(void)
{
    enum { yellow, red, blue=7, green } color; //enum {열거형 상수 리스트} 열거형 태그로 정의
    printf("원하는 색을 입력하세요.\n");
    printf("0번 : 노란색, 1번 : 빨간색\n");
    printf("7번 : 파란색, 8번 : 초록색\n"); //열거형 리스트 상수에서 7을 파란색으로 고정
    scanf_s("%d", &color);

    if (color == yellow) printf("노란색입니다.\n");
    else if (color == red) printf("빨간색입니다.\n");
    else if (color == blue) printf("파란색입니다.\n");
    else if (color == green) printf("초록색입니다.\n");

    return 0;
}
```

Microsoft Visual Studio 디버그 × + ▾

원하는 색을 입력하세요.
0번 : 노란색, 1번 : 빨간색
7번 : 파란색, 8번 : 초록색
8
초록색입니다.

자기가 임의로 설정한 열거형 리스트 상수 + 1 의 값을 작성하면 그 리스트에 해당하는 값을
도출한다.