구조체 (Structures)

구조

> Structures (또는 structs)는 여러 관련 변수를 한 곳에 그룹화하는 방법입니다. 구조체의 각 변수는 구조체의 멤버 로 알려져 있습니다.

> 배열 과 달리 구조에는 다양한 데이터 유형(int, float, char 등)이 포함될 수 있습니다.

구조 만들기

struct예약어를 사용하여 구조를 만들고 중괄호{} 안에 각 멤버를 선언할 수 있습니다.

```
struct 구조체이름 {
  자료형 멤버변수1;
  자료형 멤버변수2;
};
```

```
struct MyStructure {   // 구조체 선언
  int myNum;           // 멤버변수 (int variable)
  char myLetter;       // 멤버변수 (char variable)
};
```

- 구조에 액세스하려면 해당 구조의 변수를 만들어야 합니다.
- main()메서드 내에서 struct키워드를 사용하고 그 뒤에 구조체 이름, 구조체 변수 이름을 사용하세요.

```
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  struct myStructure s1;
  return 0;
}
```

구조체 멤버 접근

구조체의 멤버에 액세스하려면 점 구문(.)을 사용하세요.

```
// 구조체 생성
struct myStructure {
  int myNum;
```

```
  char myLetter;
};

int main() {
  // s1이라는 myStructure의 구조 변수를 만듭니다
  struct myStructure s1;

  // s1의 구성원에게 값 할당
  s1.myNum = 13;
  s1.myLetter = 'B';

  // Print values
  printf("My number: %d\n", s1.myNum);
  printf("My letter: %c\n", s1.myLetter);

  return 0;
}
값 :
My number: 13
My letter: B
```

> 이제 하나의 구조를 사용하여 서로 다른 값을 가진 여러 구조 변수를 쉽게 만들 수 있습니다.

```
#include <stdio.h>

struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  struct myStructure s1;
  struct myStructure s2;

  // Assign values to different struct variables
  s1.myNum = 13;
  s1.myLetter = 'B';

  s2.myNum = 20;
  s2.myLetter = 'C';

  printf("s1 number: %d\n", s1.myNum);
  printf("s1 letter: %c\n", s1.myLetter);

  printf("s2 number: %d\n", s2.myNum);
  printf("s2 letter: %c\n", s2.myLetter);

  return 0;
};
결과:
s1 number: 13
s1 letter: B
```

```
s2 number: 20
s2 letter: C
```

구조체의 문자열

> C의 문자열은 실제로 문자의 배열

```
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30]; // String
};

int main() {
  struct myStructure s1;

  // s1.myString = "Some text"; 오류 발생
  // 배열에 값을 할당할 수 없다
  strcpy(s1.myString, "Some text");

  // Print the value
  printf("My string: %s", s1.myString);

  return 0;
}
결과 :
My string: Some text
```

> strcpy (string copy)함수는 문자열을 다른 문자열에 복사하는 함수 문자열을 복사한 곳의 문자열 포인터를 반환합니다. strcpy(대상문자열, 원본문자열);

```
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일

int main()
{
    char s1[10] = "Hello";    // 크기가 10인 char형 배열을 선언하고 문자열 할당
    char s2[10];              // 크기가 10인 char형 배열을 선언

    strcpy(s2, s1);         // s1의 문자열을 s2로 복사
  //strcpy(대상문자열, 원본문자열);
    printf("%s\n", s2);    // Hello

    return 0;
}
실행결과 :
Hello
```

char s1[10]	H	e	l	l	o	NULL
strcpy(s2, s1)						
char s2[10]	H	e	l	l	o	NULL

더 간단한 구문

- 선언 시 구조 변수의 멤버에 값을 한 줄로 할당할 수도 있습니다.
- {}중괄호 안에 쉼표로 구분된 목록에 값을 삽입하면 됩니다
- 이 기술을 사용하면 문자열 값에 대한 strcpy()함수를 사용할 필요가 없습니다 .

```
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  struct myStructure s1 = {13, 'B', "Some text"};

  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);
  return 0;
}
결과
13 B Some text
```

> 참고: 삽입된 값의 순서는 구조에 선언된 변수 유형의 순서와 일치해야 합니다(int의 경우 13, char의 경우 'B' 등).

구조 복사

한 구조를 다른 구조에 할당할 수도 있습니다.

다음 예에서는 s1의 값이 s2에 복사됩니다.

```
struct myStructure s1 = {13, 'B', "Some text"};
struct myStructure s2;

s2 = s1;
```

값 수정

> 값을 변경/수정하려면 점 구문(.)을 사용하면 됩니다.

```
struct myStructure {
  int myNum;
  char myLetter;
```

```
  char myString[30];
};

int main() {
  struct myStructure s1 = {13, 'B', "Some text"};

  // 값 수정
  s1.myNum = 30;
  s1.myLetter = 'C';
  strcpy(s1.myString, "Something else");

  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);
  return 0;
}
```

> 값 수정은 구조 값을 복사할 때 특히 유용합니다.

```
// 구조체 변수를 생성하고 값을 할당합니다.
struct myStructure s1 = {13, 'B', "Some text"};

// 다른 구조체 변수를 생성합니다.
struct myStructure s2;

// s2에 s1값을 복사
s2 = s1;

// s2값을 수정
s2.myNum = 30;
s2.myLetter = 'C';
strcpy(s2.myString, "Something else");

printf("%d %c %s\n", s1.myNum, s1.myLetter, s1.myString);
printf("%d %c %s\n", s2.myNum, s2.myLetter, s2.myString);
```

30 C Something else

구조는 어떻게 유용할까요?

브랜드, 모델, 연도 등 자동차에 대한 다양한 정보를 저장하는 프로그램을 작성해야 한다고 상상해 보세요. 구조의 가장 큰 장점은 단일 "자동차 템플릿"을 만들어서 만드는 모든 자동차에 사용할 수 있다는 것입니다. 실제 사례는 아래를 참조하세요.

```
struct Car {
  char brand[50];
  char model[50];
  int year;
};

int main() {
  struct Car car1 = {"BMW", "X5", 1999};
  struct Car car2 = {"Ford", "Mustang", 1969};
```

```
    struct Car car3 = {"Toyota", "Corolla", 2011};

    printf("%s %s %d\n", car1.brand, car1.model, car1.year);
    printf("%s %s %d\n", car2.brand, car2.model, car2.year);
    printf("%s %s %d\n", car3.brand, car3.model, car3.year);

    return 0;
}
결과
BMW X5 1999
Ford Mustang 1969
Toyota Corolla 2011
```