4.9 구조체와 캡슐화

개요

프로그래밍을 할 때, 일반적인 자료형들을 사용하기에 적합하지 않을 수 있습니다. 이런 상황에서 우리는 데이터를 **캡슐화**하여 **어떤 개체에 연관되는 정보들을 한 덩어리로 묶을 수 있게 됩니다.** 예를 들어 학생은 이름(string형), 나이(int형), 평점(float형)와 같은 정보들을 갖고 있는데, 이 정보들은 단독으로는 큰 의미를 갖고 있지 않습니다. 이 정보들이 모여 학생이라는 개체를 이룰 때 의미를 갖게 되고 C에서는 **구조체(structure)**라는 방법을 사용합니다.

핵심개념

* 구조체
* 캡슐화
* 추상화
* 멤버

배열과 구조체

데이터를 묶어 효율적으로 활용하기 위하여 배열을 사용하였습니다. 배열의 장점은 각 학생의 인덱스를 알 때, 모든 학생들에게 임의 접근할 수 있으며, 각 배열 원소를 순환하면서 확인할 수 있다는 것입니다. 하지만 배열은 같은 데이터형의 변수들을 하나로 묶을 수 있지만, 서로 다른 데이터형의 변수를 묶어서 사용할 수 없습니다. 그리고 사용 전에 배열의 크기를 선언해야만 합니다.

```
1 typedef struct
2 {
3   string name;
4   int year;
5   float gpa;
6 }
7 student;
```

▲ 〈코드 1〉

데이터를 묶어주는 또 다른 방법은 **구조체**입니다. 구조체를 사용함으로써 **서로 다른 자료형의 변수를 하나로 묶어 새로운 자료형을 만들 수 있습니다.** 〈코드 1〉은 student라는 구조체를 만든 예시입니다. student 구조체에는 학생과 연관된 정보들이 string형, int형, float형으로 구성되어 있습니다. 이러한 정보들을 각각 **멤버**라고 부릅니다. 구조체 자료형의 특정 멤버에 접근하고 싶다면 **구조체명.멤버명** (student.name)으로 하면 됩니다. 데이터를 구조체로 저장할 때의 장점은 서로 다른 자료형의 데이터들을 하나로 묶을 수 있다는 것입니다. 구조체를 사용했을 때의 또 다른 장점은 더 이상 학생이 몇 명이 있는지 선언할 필요가 없다는 것입니다. 구조체에서는 학생 수를 정의하지 않고 원하는 수만큼 만들 수 있습니다. 하지만 배열이 인덱스를 사용하여 각 멤버들을 순환하는 것과 달리 구조체는 멤버를 순환할 수는 없습니다.

구조체 구현하기

```
1 student s1 = {'Zamyla', 2014, 4.0};
2 s1.gpa = 3.5;
```

▲ 〈코드 2〉

위의 코드에서는 student라는 새로운 자료형을 정의했습니다. int가 자료형인 것처럼 student는 우리가 정의한 자료형인 겁니다. student 자료형으로 된 새로운 변수를 만들기 위해서는 int형 변수를 선언하듯이 〈코드 2〉의 1행 같은 방법으로 사용하면 됩니다. s1의 gpa에 접근하고 싶다면 2행처럼 s1.gpa라고 하면 됩니다.