자료 구조와 알고리즘

03. 자료 구조에 대한 이해

학습목표

- 자료 구조가 무엇인지 알 수 있다.
- 자료 구조의 필요성에 대해 알 수 있다.
- 추상 자료형이 무엇인지 알 수 있다.

개요

컴퓨터 프로그램은 데이터를 입력받아 조작하고 반환하는 것이 전부다. 데이터를 어떻게 조직하냐에 따라 프로그램은 수십 수백 배 더 빠르거나 느리게 실행될 수 있다. 이러한 데이터를 조직하는 방법을 **자료 구조**(Data Structure)라고 하는데, 더 나은 프로그램을 작성하기 위해서는 다양한 자료 구조를 알고, 각각의 자료 구조가 성능에 어떤 영향을 미치는지 알고 있어야 한다. 또한 자료 구조에 적절한 알고리즘을 선택하는 것도 중요하다. 이번 시간에는 자료구조에 대해 이해해보자.

자료 구조

자료 구조는 크게 **선형 구조**(Linear Structure)와 **비선형 구조**(Non-linear Structure)로 나눌 수 있으며, 선형 구조에는 리스트, 스택, 큐가 있고, 비선형 구조에는 그래프와 트리가 있다. 또, 대부분의 자료 구조는 네 가지 기본 방법을 사용하며 이를 **연산**(Operation)이라고 한다.

- 읽기 : 자료 구조 내 특정 위치를 찾아보는 것이다.
- 검색 : 자료 구조 내 특정 값을 찾는 것이다.
- 삽입 : 자료 구조에 새로운 값을 추가하는 것이다.

- 삭제 : 자료 구조 내 특정 값을 삭제하는 것이다.

자료 구조를 구현하는 방법에는 크게 **순차 자료 구조**(contiguous data structure)와 **연결 자료 구조**(linked data structure)가 있는데, 순차 자료 구조는 구현할 자료들을 논리적인 순서대로 메모리에 **연속**하여 저장하는 구현 방식이고, 연결 자료 구조는 노드라는 여러 개의 메모리 청크에 데이터를 저장하며, 이를 **연결**하여 구현하는 방식이다. 즉, 순차 자료구조는 배열을 이용하고, 연결 자료구조는 참조를 이용한다. 둘을 비교하면 아래와 같다.

순차 자료구조	연결 자료구조
모든 데이터가 메모리에 연속적으로 저장된다.	데이터는 노드에 저장되고, 노드는 메모리 곳곳에 흩어져 있을 수 있다.
임의 원소에 즉각적으로 접근할 수 있다.	임의 원소에 접근하는 것은 선형 시간 복잡도를 가지며 느린 편이다.
캐시 지역성* 효과로 인해 모든 데이터를 순회하는 것이 매우 빠르다.	캐시 지역성* 효과가 없어 모든 데이터를 순회하는 것이 느리다.
데이터 저장을 위해 정확하게 데이터 크기만큼 메모리를 사용한다.	각 노드에서 참조 저장을 위해 여분의 메모리를 사용한다.

*지역성(locality) : CPU가 기억장치의 특정 부분에 위치한 데이터나 프로그램 코드를 집중적으로 액세스하는 현상이다.

프로그래머는 둘의 특징을 명확히 알고 구현할 작업의 요구 조건 및 사용 빈도에 따라 둘 중 하나를 선택하거나 또는 두 개를 조합하여 응용 프로그램을 개발해야 한다.

추상 자료형

자료 구조는 상기했듯 같은 기능을 한다고 할지라도 서로 다르게 구현할 수 있다. 순차적으로 구현할 수도 있고, 연결적으로 구현할 수도 있다. 그렇기 때문에 구체적으로 어떻게 구현하라는 내용이 없이 기능만 정의한 **추상 자료형**(ADT; Abstract Data Type) 형태로 나타낸다.

컬렉션

대부분의 언어는 프로그래머의 편의성을 위해 여러 자료 구조와 알고리즘을 제공하고 있다. 이를 **컬렉션**(Collection) 혹은 **컨테이너**(Container)라 부른다.* 이런 라이브러리는 자료 구조 구현체와 자료에 하나씩 접근할 수 있는 반복자(Iterator)**를 제공하고 있다.***

* C#의 컬렉션은 여기서 확인할 수 있다.

** 이를 반복자 패턴(Iterator Pattern)이라고 한다.

*** 추상 자료형을 나타낸 인터페이스도 존재한다.

foreach문

foreach문을 이용하면 반복자를 사용해 컬렉션의 전체를 순회할 수 있다.

```
// foreach의 문법은 아래와 같다.
// <element-variable> 은 각 자료를 담을 변수다.
// <enumerable-type>은 순회가 가능한 타입이다.
// 순회가 가능하려면 IEnumerable 인터페이스를 구현해야 한다.
foreach (<element-variable> in <enumerable-type>)
{
  // body
}

// Array는 IEnumerable 인터페이스를 구현하고 있어 foreach 문에 사용할 수 있다.
int[] arr = { 1, 2, 3, 4, 5 };

// 첫 번째 원소부터 마지막 원소까지 출력한다.
foreach (int elem in arr)
{
  Console.WriteLine(elem);
}
```

더 나아가기

1. **자료 구조**가 무엇이며, 프로그램 작성 시 자료 구조에 대한 이해가 왜 필수적인지 그 **필요성**을 설명하세요.

2. 자료 구조의 종류는 어떻게 되나요?
3. 자료 구조의 기본 연산(Operation) 4가지와 각각의 의미를 쓰세요.
4. 자료 구조를 구현하는 **순차 자료 구조**와 **연결 자료 구조**의 메모리 저장 방식 및 임의 원소 접근 속도 측면에서의 가장 큰 차이점을 서술하세요.
5. **추상 자료형 (ADT)**의 정의를 설명하고, 자료구조를 ADT 형태로 나타내는 이유를 구현 방식의 관점에서 설명해 보세요.
6. **순차 자료 구조**가 **연결 자료 구조**보다 데이터를 순회하는 것이 매우 빠른 이유를 **'캐시 지역성'** 개념을 사용하여 설명해 보세요.

참고자료

- C로 배우는 쉬운 자료구조
- 누구나 자료 구조와 알고리즘
- [무료] C로 배우는 자료구조 및 여러가지 예제 실습
- Collections (C#) | Microsoft Learn