Array와 List

배열(Array)과 리스트(List)

2023.06.08

|

Arrary (배열)

- 참고 생활코딩 - 배열
- 데이터가 많아지면 그룹 관리의 필요성이 생긴다. 이럴 때 프로그래밍에서 사용하는 것이 **배열**
- 여러 데이터를 하나의 이름으로 **그룹핑해서 관리** 하기 위한 자료구조
- 배열을 이용하면 하나의 변수에 여러 정보를 담을 수 있고, **반복문과 결합** 하면 많은 정보도 효율적으로 처리할 수 있다.
- 배열 인덱스는 값에 대한 **유일무이한 식별자** (참고로 리스트에서 인덱스는 몇 번째 데이터인가 정도의 의미를 가짐)

java array 예시

- 배열은 정의와 동시에 길이를 지정하며 길이를 바꿀 수 없다.

```
// array 정의 int [] numbers1 = new int [ 4 ]; // array에 값 저장
numbers1 [ 0 ]= 10 ; numbers1 [ 1 ]= 20 ; numbers1 [ 2 ]= 30 ; // array의 길이 확인
System . out . println ( numbers1 . length );   // 4
```

- 처음 정의시 크기를 4로 지정하였기 때문에 3개의 값이 설정되었어도 결과는 4이다.
- 바로 이런 이유에서 자바에서는 ArrayList나 LinkedList와 같은 리스트를 사용한다.
- 리스트는 자동으로 엘리먼트를 수용할 수 있는 크기가 조정되고 또 리스트 내의 엘리먼트의 실제 개수를 알려준다.

> 자바의 배열은 기능적으로 한계가 많습니다. 배열의 크기를 배열을 생성할 때 지정하는 것이나, 배열의 크기를 변경할 수 없는 것은 몹시 불편한 일입니다. 또 배열에서 설정된 엘리먼트의 개수를 알아낼 수 없는 것도 불편합니다. 그렇다고 배열이 쓸모가 없는 것은 아닙니다. 데이터의 크기가 확정적일 때 배열을 사용하는 것이 메모리나 처리속도 면에

> 서 좋습니다. 또한 배열은 다른 데이터 스트럭쳐의 부품이 되기도 합니다. 기능이 최소한일수록 좋은 부품이 될 수 있습니다. 기능이 많으면 사용하기는 좋아도 그것을 사용할 수 있는 용도가 제한되기 때문입니다. 그럼 배열의 이런 한계를 극복하려면 어떻게 해야할까요? 조금 더 개선된 기능의 데이터 스트럭쳐를 고안할 필요가 있습니다. 그런 데이터 스트럭쳐가 바로 list입니다 (출처 : 생활코딩)

배열의 특징

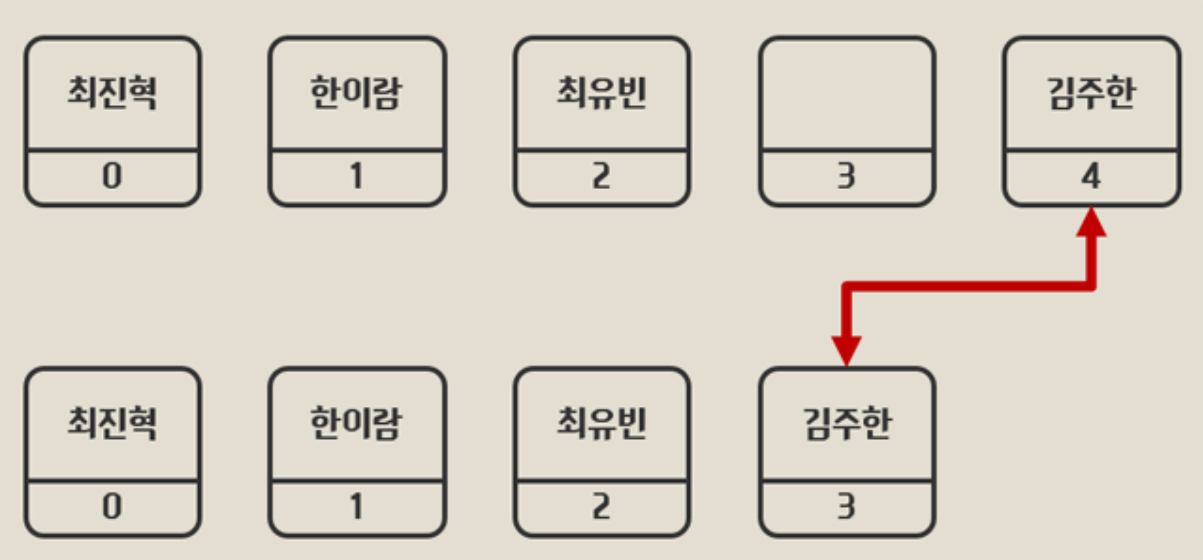
- **크기가 정해져 있다 / 기능이 없다**(이는 배열의 장점이자 단점으로 배열은 다른 자료구조의 좋은 부품으로 사용될 수 있다.)
- `인덱스` 를 가지며, 엘리먼트의 인덱스는 변경되지 않는다.(현실세계의 주민번호, 학급에서 학생에게 부여하는 번호)
- 유관 데이터를 메모리에 순차적으로 (인덱스) 나열할 수 있다.
- 인덱스를 활용하여 빠르게 조회가 가능하다.
- `cache hit` 의 가능성이 커져서 성능에 큰 도움이 된다.
- 인덱스를 이용하여 데이터를 가져오려면 데이터에 대한 인덱스 값이 고정되어야 한다. (삭제된 엘리먼트의 공간이 그대로 남는다. -> `메모리의 낭비`)

배열의 한계

- 배열은 길이를 바꿀 수 없다. 가변 배열과 같이 길이가 변경 가능한 배열의 경우,
  - 기존의 배열은 그대로 두고,
  - 새로운 길이로 지정된 배열을 따로 할당 후 (메모리가 있는지 탐색 필요)
  - 데이터의 복사를 진행하고,
  - 기존의 배열을 삭제한다.
  - `총 3번의 작업` + `메모리 탐색` 이 필요하기 때문에 리소스 낭비가 크다.
- 위와 같은 한계를 해결하기 위해서 `linked list 자료형`을 활용할 수 있다. (탐색, 할당, 복사, 삭제 등의 리소스 낭비가 없다.)
- 배열은 인덱스에 따라서 값을 유지하기 때문에, 엘리먼트가 삭제되어도 빈자리(null)가 남게 된다. (불필요한 메모리 차지)

- 조건문을 통해서 빈 엘리먼트를 제외할 수 있으나, 조건문을 많이 사용하는 것은 좋지 않다.
- 삭제한 데이터를 뒤에 위치한 엘리먼트로 메꾸면, 데이터가 순서에 따라서 빈틈없이 연속적으로 위치하며 이를 list(리스트) 라고 한다.
- 인덱스가 중요한 경우는 배열을 사용, 인덱스가 중요하지 않은 경우에는 리스트를 사용한다.

배열은 삭제된 데이터의 빈공간을 그대로 남겨둔다, 리스트는 빈공간을 채운다. (출처: 생활코딩)



---

list (리스트)

- 참고 생활코딩 - 리스트
- 리스트는 배열이 가지고 있는 인덱스라는 장점을 버리고 대신 **빈틈없는 데이터의 적재** 라는 장점을 취한 데이터 스트럭쳐
- 리스트 자료구조의 핵심은 엘리먼트들 간의 `순서`. 따라서 리스트를 다른 말로는 시퀀스(sequence) 라고도 부른다. 즉 **순서가 있는 데이터의 모임이** 리스트이다.
- 리스트에서 인덱스는 **몇 번째 데이터인가** 정도의 의미를 가진다. (배열-Array에서의 인덱스는 값에 대한 **유일무이한 식별자**)
- 빈 엘리먼트는 허용하지 않는다.
- 순차성을 보장하지 못하기 때문에 spacial locality 보장이 되지 않아서 cash hit가 어렵다.

- 데이터 갯수가 확실하게 정해져 있고, 자주 사용된다면 array가 더 효율적이다.
- 리스트에 대한 효용은 어떤 언어를 사용하느냐에 따라서 달라진다. 특히 많은 언어가 리스트를 기본적으로 지원하기 때문에 리스트를 직접 구현하는 경우는 줄고 있다.

list 자료구조의 대표 기능 (operation)

> 자료구조에서 가장 중요한 점은, 해당 자료구조가 어떠한 기능을 가지고 있느냐는 것이다.

- 처음, 끝, 중간에 **엘리먼트를 추가/삭제** 하는 기능
- 리스트에 데이터가 있는지를 체크하는 기능
- 리스트의 모든 데이터에 접근할 수 있는 기능

언어별 list 지원 비교

- data-structure를 직접 구현할 수도 있지만, 대부분의 경우 이미 누군가가 구현한 라이브러리를 사용하는 경우가 더 많다.
- 최근의 언어는 리스트를 기본적으로 지원한다.
- data-structure는 언어마다 다른다.

C

- 리스트를 지원하지 않는다. 대신 배열을 지원한다.
- 리스트를 사용하려면 직접 만들거나 라이브러리를 사용해야한다. (따라서 리스트에 대한 깊은 이해와 안목이 필요함)

JavaScript

- C 패밀리 랭귀지
- 자바스크립트에서는 배열에 리스트의 기능이 포함되어 있다.

```
numbers = [10, 20, 30, 40, 50];

numbers.splice(3, 1);  // 인덱스 3에서부터 1개의 값을 삭제한다. for (i = 0; i < numbers.length; i ++ ){
  console.log(numbers[i]);
   // 10, 20, 30, 50 }
```

Python

- 기본적으로 리스트를 제공하며, 배열은 제공하지 않는다.
- 파이썬에서는 리스트가 배열이다.
- 파이썬의 리스트는 크기가 가변적이고, 어떤 원소 타입이던 저장할 수 있다는 장점을 가진다. 대신 C의 array 보다 메모리를 더 많이 필요로 한다는 단점이 있다.

```
numbers = [10, 20, 30, 40, 50]
numbers . pop(3)  # 40 / 3번째 인덱스 값 리턴 후 삭제 for number in numbers:
    print (number)  # 10, 20, 30, 50
```

Java

- 자바는 배열과 리스트를 모두 지원하고, 두 가지가 완전히 분리되어 있다.
- 배열은 배열의 장점이, 리스트는 리스트의 장점이 있기 때문에 개발자가 원하는대로 직접 선택가능하다.
- java는 javascript, python에 비해서 자료구조에 대해 더 알아야 할 필요성이 있지만,그만큼 개발자에게 더 큰 자유도가 주어진다.
- 자바는 2가지 형태의 리스트를 지원한다.
    - `LinkedList / ArrayList`
    - 똑같은 기능(메소드)를 가진 리스트가 2가지 존재한다.
    - 각각의 장단점이 분명하다.

```
// 배열 - 추가, 삭제가 어렵다. 직접 구현해야한다. int [] numbers =
{ 10 , 20 , 30 , 40 , 50 }; // 리스트 (ArrayList) ArrayList numbers = new
ArrayList (); numbers . add ( 10 );  // 추가 numbers . remove ( 0 );  // 삭제 // 리스
트 (LinkedList) LinkedList numbers = new LinkedList (); numbers . add ( 10 );  //
추가 numbers . remove ( 0 );  // 삭제
```

Java - ArrayList / LinkedList

- Java에서는 배열 (array) 이외에 ArrayList와 LinkedList 2종류의 리스트를 제공한다.
- 인덱스를 이용해서 데이터를 가져오는 것이 빈번하다면 내부적으로 배열을 이용하는 ArrayList가 훨씬 빠르다. 하지만 데이터의 추가/삭제가 빈번하다면 LinkedList가 훨씬 효과적이다.
- 이와같이 처리하고자 하는 **데이터에 따라서 어떤 데이터 스트럭쳐를 선택할지를 잘 판단하는 것은 대규모 시스템을 구축하는데는 필수적인 능력이다.**
- 이러한 판단을 하기 위해서는 직접 데이터 스트럭쳐를 구현해서 사용하지 않더라도 내부적인 메커니즘을 이해할 필요가 있다.

java의 ArrayList, LinkedList 비교 (출처: 생활코딩)

Array List

- Array List는 배열을 이용해서 리스트를 구현한 것을 의미한다.
- 장점 : 내부적으로 배열을 사용하기 때문에 인덱스를 이용해서 접근하는 것이 빠르다.
- 단점 : 데이터의 추가와 삭제가 느리다.

데이터의 추가

- Array List는 내부적으로 데이터를 배열에 저장한다. 배열의 특성상 데이터를 리스트의 처음이나 중간에 저장하면, 이후의 데이터들이 한칸씩 뒤로 물러나야한다.

데이터의 삭제

- 삭제도 추가와 유사하게 빈자리가 생기면 빈자리를 채우기 위해서 순차적으로 한칸씩 땡겨야 한다.

데이터 조회(가져오기)

- 인덱스를 이용하여 데이터를 가져오고 싶을 때 Array로 구현한 리스트는 속도가 매우 빠르다. (메모리 상의 주소를 정확하게 참고해서 가져오기 때문이다.)