IT@COOKBOOK

4판

Data Structures

이지영 지음

C로 배우는 쉬운 자료구조

연습문제 해답은 제공하지 않습니다.

한빛아카데미 Hanbit Academy, Inc.

4차 산업혁명 시대를 이끄는 문제 해결 능력

우리는 스마트한 지능정보기술 혁명의 4차 산업혁명 시대를 살고 있습니다. 하루가 다르게 새로운 서비스가 등장하고 '인공지능', '빅데이터', '스마트 서비스'를 주제로 한 뉴스가 가득합니다. 이러한 4차 산업혁명 시대를 올바르게 이끄는 것이 컴퓨터 전공자들의 역할입니다. 이 책에서 다루는 자료구조와 알고리즘은 컴퓨터 전공자들이 프로그램을 이용해 문제를 해결하는 능력을 기르는 데 기초가 됩니다.

컴퓨터 전공자는 컴퓨터 기반의 문제 해결 능력, 즉 컴퓨터 프로그래밍을 배웁니다. 초보 프로그래머는 자신이 만든 프로그램의 결괏값만 맞으면 완성된 것이라고 여깁니다. 하지만 '완성'을 넘어 '성공'적인 프로그램은 결괏값은 물론 프로그램이 제대로 구성되었는지, 데이터를 처리하는 데 올바른 기법을 사용하고 있는지, 데이터를 효율적으로 구성하고 사용하는지 등을 평가 받습니다. '완성을 만드는 개발자'가 아니라 이 시대가 요구하는 '성공을 만드는 능력자'가 되려면 컴퓨터 내부의 자료 표현 방법, 자료를 효율적으로 구성하는 방법, 알고리즘을 제대로 이해하고 응용하는 방법을 알아야 합니다. 이러한 여러 방법을 배우는 과목이 바로 자료구조입니다.

이 책에서는 자료구조와 알고리즘을 이론으로만 다루지 않고 독자들이 더 쉽게 이해할 수 있도록 다양한 그림을 활용해 설명합니다. 그리고 알고리즘을 구체화하여 C 프로그램으로 구현하면서 이론과 실습을 병행할 수 있기 때문에 자료구조와 알고리즘을 공부할 때 효과적입니다. 특히, 이번에 개정된 4판에서는 자료구조가 어떻게 활용되는지 학습할 수 있도록 코딩 테스트 스타일의 응용예제를 수록하였습니다. 부디 이 책이 컴퓨터 자료에 대한 다양한 구조화 방법을 이해하고 주어진 문제를 해결하는 최적의 방법을 선택 및 활용하는 데 도움이 되길 바랍니다.

어느새 4판을 출간하게 되었습니다. 이번 4판에서는 새로운 편집으로 외형을 단장하고 내용적으로도 많은 부분이 달라졌습니다. 4판에서 달라진 부분은 다음과 같습니다.

❶ **프로그래밍 심화 학습을 위해 분할 컴파일 방식으로 실습 예제 재구성**

자료구조는 일반적으로 프로그래밍 기본을 학습한 후에 심화 과정으로 배우는데 '프로그래밍고급' 교과목으로 수업을 하는 대학도 있습니다. 이러한 현실을 반영해 자료구조 구현을 통해 프로그래밍 수준을 향상시킬 수 있도록 분할 컴파일 방식으로 실습 예제를 재구성하였습니다. 자료형 및 함수 원형을 선언하는 헤더 파일, 함수를 정의하는 소스 파일, 그리고 함수를 호출하여 실행하는 main() 함수가 있는 ex0_0.c 파일로 분리하여 프로그래밍 심화 학습에 도움이 되도록 하였습니다.

❷ **자료구조를 활용한 문제 해결 능력 향상을 위해 코딩 테스트 스타일의 응용예제 추가**

이론과 실습 예제로 배운 자료구조가 어떻게 활용되는지 학습할 수 있도록 코딩 테스트 스타일의 응용예제를 장마다 추가하였습니다. 다양한 문제를 자료구조 관점에서 분석하고 해결 방법을 설계할 수 있도록 하였습니다.

❸ **최신 기출 문제로 연습문제 업데이트**

2020년에 개편된 정보처리기사를 비롯해 전자계산기조직응용기사, 전산직 공무원, 정보관리기술사 등의 최신 기출 문제를 활용하여 연습문제를 업데이트하였습니다. 연습문제를 통해 본문에서 학습한 내용을 확인하고, 주요 시험의 자료구조 문제를 준비할 수 있도록 하였습니다.

❹ **예제의 실습 편이성 향상**

예제 프로그램을 비주얼 스튜디오 2019 기반으로 버전업하고, 프로젝트 파일로 제공함으로써 예제 실습의 편이성을 향상시켰습니다. 또한 다른 버전의 컴파일러 환경에서 실습하는 독자를 위해 표준 C 소스 파일도 함께 제공합니다.

자료구조를 학생들의 눈높이에 맞춰 강의할 수 있도록 도와주고, 때로는 엉뚱한 질문으로 웃게해 준 내 소중한 학생들, 이 책이 나오도록 애써 준 한빛아카데미㈜, 책을 집필하는 동안 격려해주고 도움을 주신 모든 분에게 고마운 마음을 전합니다. 항상 힘이 되어 주시는 부모님과 가족들에게 감사합니다. 지칠 때마다 에너지를 충전해 주는 사랑하는 조카 상원이와 상범이에게도 고맙다고 말하고 싶습니다. 이 책을 읽고 공부할 독자 여러분, 모두 건강하고 행복하길 진심으로 바랍니다. 감사합니다.

저자 이지영

| 강의 보조 자료 | 한빛아카데미 홈페이지에서 '교수회원'으로 가입하신 분은 인증 후 교수용 강의 보조 자료를 제공받을 수 있습니다. 한빛아카데미 홈페이지 상단의 〈교수전용공간〉 메뉴를 클릭하세요.
http://www.hanbit.co.kr/academy

| 학습 보조 자료 | 본문의 예제 파일은 다음 주소에서 내려받을 수 있습니다. 비주얼 스튜디오 2019의 프로젝트 파일과 표준 C 소스 파일 두 가지 형태로 제공합니다.
http://www.hanbit.co.kr/src/4541

| 연습문제 해답 | 본 도서는 대학 강의용 교재로 개발되었으므로 연습문제 해답은 제공하지 않습니다.

| 본문 구성 |

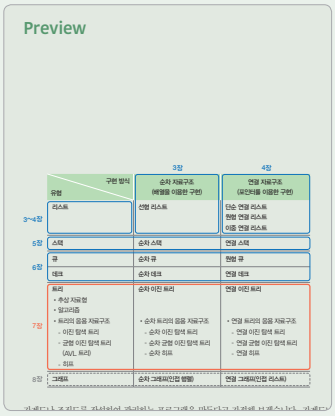
❶ **자료의 표현, 프로그래밍 기법(1~2장)** 자료구조와 알고리즘을 이해하기 위한 기본 지식을 학습합니다. 컴퓨터에서 자료를 표현하는 방법과 알고리즘을 이해하고, 프로그램을 작성할 때 필요한 프로그래밍 기법을 익힙니다.

❷ **순차 자료구조, 연결 자료구조(3~4장)** 자료의 논리적인 순서와 저장되는 물리적 순서를 일치시키는 순차 자료구조의 표현 방법을 알아봅니다. 물리적인 순서를 고려하지 않고 논리적인 순서대로 연결하여 자료를 구성하는 연결 자료구조의 표현 방법도 알아봅니다. 더불어 순차 자료구조와 연결 자료구조의 차이점을 살펴봅니다.

❸ **스택, 큐, 트리, 그래프(5~8장)** 순서가 있는 자료를 선형으로 구조화한 스택과 큐, 1:n의 계층형으로 구조화한 트리, m:n 관계를 구조화한 그래프에 대해 알아보고 그 특징과 연산 방법까지 살펴봅니다. 스택, 큐, 트리, 그래프를 배열을 이용한 순차 자료구조 표현 방법과 포인터를 이용한 연결 자료구조 표현 방법으로 구현하는 방법도 알아봅니다.

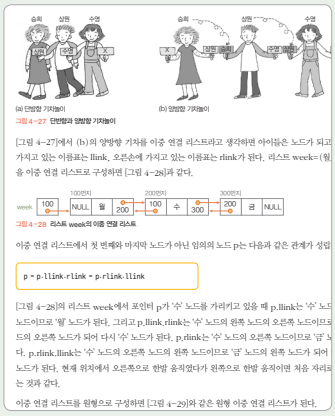
❹ **정렬과 검색(9~10장)** 자료를 응용하는 방법으로 정렬과 검색을 알아봅니다. 선택·버블·퀵·삽입·셸·병합·기수·히프·트리 정렬의 알고리즘을 살펴보고 프로그램으로 구현하는 방법까지 살펴봅니다. 순차·이진 검색, 이진 트리 검색과 해싱 등의 방법도 알아봅니다.

| 구성 요소 |



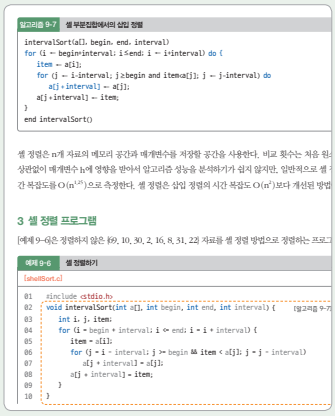
프리뷰

각 장의 내용을 본격적으로 접하기 전에 해당 장의 이론이 고안된 동기나 필요성 등을 예를 통해 간단히 소개하고 3장부터 8장까지는 로드맵을 통해 각 장에서 배울 내용을 보여 줍니다.



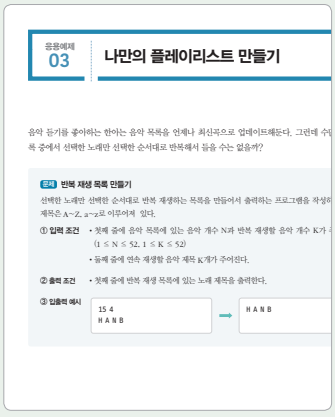
삽화와 도해

주요 개념을 삽화를 곁들여 소개하고 그 원리를 단계별 도해로 보여 줍니다. 개념 사이의 관계를 명확히 보여 주고, 핵심 개념을 일목요연하게 정리해 줍니다.



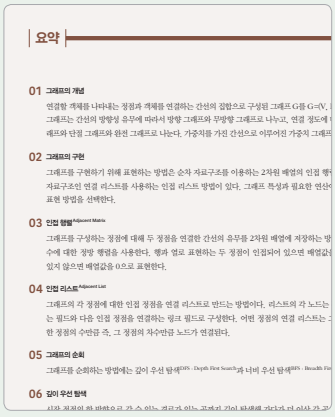
ADT, 알고리즘, 예제

추상 자료형과 핵심 연산에 대한 알고리즘을 정리해 줍니다. 그리고 이를 구체화한 C 프로그램을 실행 결과와 함께 보여 줍니다.



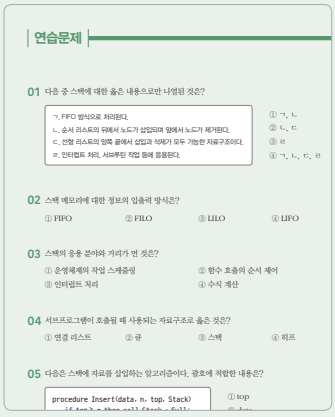
응용예제

자료구조를 어떻게 활용할 수 있는지 학습할 수 있는 코딩 테스트 스타일의 문제입니다. 자료구조 관점에서 문제를 분석하고 해결 방법을 설계하도록 알려 줍니다.



요약

본문의 핵심 내용을 요약해서 정리합니다. 본문에서 익힌 세분화된 지식을 다시 조립하여 전체적으로 완성해 볼 수 있습니다.



연습문제

본문에서 익힌 내용을 문제 형식으로 정리합니다. 개념 확인 문제부터 응용력을 기를 수 있는 프로그래밍 문제까지 다양한 유형의 문제를 경험할 수 있습니다.

이 책은 핵심 자료구조의 유형, 구현 방법, 응용 기법을 다룹니다. 2장에서는 자료구조를 C 언어로 구현하기 위해 필요한 C 프로그래밍 기법을 간단히 복습하고, 3~8장에서는 핵심 자료구조의 원리와 구현 방법을 알아봅니다. 마지막으로 9~10장에서는 자료에 대한 응용 기법으로 아홉 가지 자료 정렬 방법과 다섯 가지 자료 검색 방법을 살펴봅니다.

		3장	4장
	유형 \ 구현 방식	순차 자료구조 (배열을 이용한 구현)	연결 자료구조 (포인터를 이용한 구현)
3~4장	**리스트** • 추상 자료형 • 알고리즘 • 리스트의 응용	**선형 리스트**	**단순 연결 리스트** **원형 연결 리스트** **이중 연결 리스트**
5장	**스택** • 추상 자료형 • 알고리즘 • 스택의 응용	**순차 스택**	**연결 스택**
6장	**큐** • 추상 자료형 • 알고리즘 • 큐의 응용	**순차 큐** **연결 큐**	**원형 큐**
	데크 • 추상 자료형	**순차 데크**	**연결 데크**
7장	**트리** • 추상 자료형 • 알고리즘 • 트리의 응용 자료구조 - 이진 탐색 트리 - 균형 이진 탐색 트리 (AVL 트리) - 히프	**순차 이진 트리** • 순차 트리의 응용 자료구조 - 순차 이진 탐색 트리 - 순차 균형 이진 탐색 트리 - 순차 히프	**연결 이진 트리** • 연결 트리의 응용 자료구조 - 연결 이진 탐색 트리 - 연결 균형 이진 탐색 트리 - 연결 히프
8장	**그래프** • 추상 자료형 • 알고리즘 • 그래프의 응용 자료구조 - 신장 트리 - 최단 경로 그래프 (다익스트라 / 플로이드)	**순차 그래프(인접 행렬)** • 순차 그래프의 응용 자료구조 - 순차 신장 트리 - 순차 최단 경로 그래프	**연결 그래프(인접 리스트)** • 연결 그래프의 응용 자료구조 - 연결 신장 트리 - 연결 최단 경로 그래프

순차 자료구조와 선형 리스트

Chapter

03

학습목표

- 순차 자료구조의 개념과 특징을 알아본다.
- 선형 리스트의 개념과 연산을 알아본다.
- C 언어를 이용해 선형 리스트의 순차 자료구조를 구현한다.
- 선형 리스트의 응용과 순차 자료구조 구현 방법을 알아본다.

Chapter

03

Preview

		3장	4장
	구현 방식 / 유형	순차 자료구조 (배열을 이용한 구현)	연결 자료구조 (포인터를 이용한 구현)
3~4장	리스트 • 추상 자료형 • 알고리즘 • 리스트의 응용	선형 리스트	단순 연결 리스트 원형 연결 리스트 이중 연결 리스트
5장	스택	순차 스택	연결 스택
6장	큐	순차 큐	원형 큐
	데크	순차 데크	연결 데크
7장	트리	순차 이진 트리	연결 이진 트리
8장	그래프	순차 그래프(인접 행렬)	연결 그래프(인접 리스트)

문서를 정리하는 가장 단순한 방법은 무엇일까요? 문서를 순서대로 묶고, 페이지마다 번호를 붙여 놓으면 간단하고 쉽게 정리할 수 있습니다. 이런 방식으로 자료를 순서대로 구조화하는 기본적인 자료구조 유형이 리스트입니다. 그리고 리스트 자료구조에 들어 있는 자료(원소)를 메모리에 순차적으로 저장하도록 구현하는 방식을 순차 자료구조라고 합니다. 순차 자료구조 구현 방식은 프로그래밍의 배열을 이용한 구현 방식입니다.

Section 01

순차 자료구조와 선형 리스트의 이해

1 순차 자료구조의 개념

자료는 구조화하는 방법에 따라 리스트, 스택, 큐, 데크, 트리, 그래프 등으로 나뉜다. 이러한 자료구조 유형을 프로그램으로 구현하는 방식에는 순차 자료구조와 연결 자료구조(4장)가 있다. 순차 자료구조는 구현할 자료들을 논리적인 순서대로 메모리에 연속하여 저장하는 구현 방식이다. 따라서 순차 자료구조는 논리적인 순서와 물리적인 순서가 항상 일치해야 한다. C 프로그래밍에서 순차 자료구조의 구현 방식을 제공하는 프로그램 기법은 배열이다. 앞으로 순차 자료구조 방식으로 구현한다는 것은 배열을 이용하여 구현한다는 의미로 이해하면 된다.

표 3-1 순차 자료구조와 연결 자료구조의 비교

구분	순차 자료구조	연결 자료구조
메모리 저장 방식	메모리의 저장 시작 위치부터 빈자리 없이 자료를 순서대로 연속하여 저장한다. 논리적인 순서와 물리적인 순서가 일치하는 구현 방식이다.	메모리에 저장된 물리적 위치나 물리적 순서와 상관없이, 링크에 의해 논리적인 순서를 표현하는 구현 방식이다.
연산 특징	삽입·삭제 연산을 해도 빈자리 없이 자료가 순서대로 연속하여 저장된다. 변경된 논리적인 순서와 저장된 물리적인 순서가 일치한다.	삽입·삭제 연산을 하여 논리적인 순서가 변경되어도, 링크 정보만 변경되고 물리적 순서는 변경되지 않는다.
프로그램 기법	배열을 이용한 구현	포인터를 이용한 구현

2 선형 리스트의 표현

자료의 특징(추상 자료형)과 주로 사용할 연산(알고리즘)에 따라 최적의 형태로 자료를 구조화해야 하는데, 자료를 구조화하는 가장 기본적인 방법은 나열하는 것이다. [표 3-2]와 같이 동창 이름, 좋아하는 음식, 오늘 할 일 등을 하나씩 나열할 수 있는데, 이렇게 나열한 목록을 리스트List라고 한다.

표 3-2 리스트 예

동창 이름	좋아하는 음식	오늘 할 일
상원	김치찌개	운동
상범	닭볶음탕	자료구조 스터디
수영	된장찌개	과제 제출
현정	잡채	동아리 공연 연습
…	…	…

이때 원소들을 순서대로 나열한 리스트를 선형 리스트Linear List 또는 순서 리스트Ordered List라고 한다.

표 3-3 선형 리스트 예

동창 이름	
1	상원
2	상범
3	수영
4	현정
…	

좋아하는 음식	
1	김치찌개
2	닭볶음탕
3	된장찌개
4	잡채
…	…

오늘 할 일	
1	운동
2	자료구조 스터디
3	과제 제출
4	동아리 공연 연습
…	…

리스트를 표현하는 형식은 [그림 3-1]의 (a)와 같다. 원소가 순서대로 나열된 선형 리스트는 ❶에 나열된 순서가 원소들의 순서가 된다. [표 3-3]에 있는 동창 이름 선형 리스트를 리스트의 일반화 표현 형식으로 나타내면 (b)와 같다.

(a) 리스트 표현 형식 (b) 리스트 표현 예

그림 3-1 리스트 표현 형식과 예

원소가 하나도 없는 리스트는 공백 리스트라고 하고, [그림 3-2]와 같이 빈 괄호로 표현한다.

그림 3-2 공백 리스트 형식

선형 리스트는 메모리에 저장하는 구현 방식에 따라 순차 방식으로 구현하는 선형 순차 리스트Linear Sequential List와 연결 방식으로 구현하는 선형 연결 리스트Linear Linked List로 나뉜다. 일반적으로 선형 순차 리스트를 선형 리스트라고 하고, 선형 연결 리스트를 연결 리스트라고 한다. 3장에서는 순차 자료구조 방식으로 구현하는 선형 리스트를 살펴본다.

선형 리스트(선형 순차 리스트)는 원소들이 나열된 논리적인 순서와 메모리에 저장되는 물리적인 순서가 일치하는 순차 자료구조 방식으로 구현한다. 순차 자료구조는 원소를 논리적인 순서대로 메모리에 연속하여 저장한다. [표 3-3]에 있는 동창 이름 선형 리스트가 메모리에 저장되는 순서는 [그림 3-3]과 같이 논리 순서와 일치한다.

순차 자료구조는 원소들이 순서대로 연속하여 저장되기 때문에 시작 위치와 원소 크기를 알면 특정 원소의 위치를 쉽게 알 수 있다. 시작 위치가 α이고, 원소 크기가 ℓ인 선형 리스트에서 두 번째 원소의 위치는 [그림 3-4]와 같이 $\alpha+\ell$이고, 세 번째 원소의 위치는 $\alpha+2\ell$이다. 따라서 i번째 원소의 위치는 $\alpha+(i-1)\times\ell$이 된다.

그림 3-3 선형 리스트의 메모리 저장 구조 예

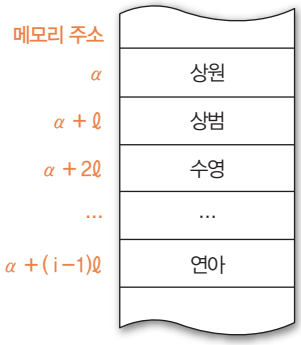


그림 3-4 선형 리스트에서 원소의 위치

3 선형 리스트의 배열 표현

선형 리스트는 배열을 사용해 순차 자료구조 방식을 구현한다. 배열은 〈인덱스, 원소〉 쌍으로 구성되어 메모리에 연속적으로 할당되는데, 이때 인덱스는 배열 원소의 순서를 나타낸다. 배열은 순서를 가진 배열 원소들을 메모리에 연속하여 순차적으로 구성하므로, 프로그래밍 언어에서 제공하는 배열을 사용하면 순차 자료구조 방식의 선형 리스트를 쉽게 구현할 수 있다.

1차원 배열을 이용한 선형 리스트 표현

1차원 배열은 인덱스를 하나만 사용하는 배열이다. 원소 순서를 한 개의 값으로 구별할 수 있는 간단한 선형 리스트는 1차원 배열을 사용하여 표현할 수 있다. [표 3-4]와 같이 분기별 노트북 판매량에 대한 리스트를 생각해 보자. 1/4~4/4 분기 판매량을 순서대로 관리해야 하므로, 분기 순서대로 값을 나열하는 선형 리스트로 표현할 수 있다.

표 3-4 분기별 노트북 판매량 리스트

분기	1/4 분기	2/4 분기	3/4 분기	4/4 분기
판매량	157	209	251	312

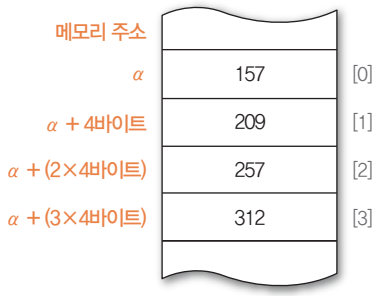
[표 3-4]를 1차원 배열을 사용하여 선형 리스트로 표현하면 [그림 3-5]와 같다. 1차원 배열 sale은 int 자료형으로 선언되었으므로 각 원소의 길이는 4바이트이다. C 프로그래밍에서 배열의 인덱스는 0부터 시작하므로 배열 sale에서 각 원소의 위치는 시작 주소가 α일 때 α+(원소의 배열 인덱스×4바이트)가 된다.

```
int sale[4] = { 157, 209, 251, 312 };
```

(a) 분기별 판매량 선형 리스트의 1차원 배열 선언

	[0]	[1]	[2]	[3]
sale	157	209	251	312

(b) 분기별 판매량 선형 리스트의 논리적 구조



(c) 분기별 판매량 선형 리스트의 물리적 구조

그림 3-5 분기별 판매량 선형 리스트 예

TIP_배열에서 첫 번째 원소의 인덱스는 0이고, 두 번째 원소의 인덱스는 1이고, i번째 원소의 인덱스는 (i−1)이다. 따라서 원소 크기가 ℓ일 때, i번째 원소의 주소는 'α+(원소의 배열 인덱스×ℓ) = α+((i−1)×ℓ)'이 된다.

[예제 3-1]은 분기별 판매량 선형 리스트를 1차원 배열로 구현하고, 논리적 순서와 물리적 순서가 일치하는 순차 구조인지 확인하는 프로그램이다.

예제 3-1 원소의 논리적·물리적 순서 확인하기

```
01  #include <stdio.h>
02
03  int main(void) {
04      int i, sale[4] = {157, 209, 251, 312};
05
06      for (i = 0; i<4; i++) {
07          printf("\n address : %u sale[%d]= %d", &sale[i], i, sale[i]);
08      }
09
10      getchar(); return 0;
11  }
```

```
address : 5241668 sale[0]= 157
address : 5241672 sale[1]= 209
address : 5241676 sale[2]= 251
address : 5241680 sale[3]= 312
```

07행 : 배열 원소의 메모리 주소(&sale[i])를 %u 형식으로 출력하고, 원소값(sale[i])을 %d 형식으로 출력한다. 실행 결과를 보면 배열 sale의 시작 주소가 5241668이다. 이 때 sale[2]의 위치를 계산해 보면 다음과 같다. 계산한 값과 실행 결과로 나온 주소가 일치하므로, 논리적인 순서대로 메모리에 연속하여 저장된 걸 확인할 수 있다.

```
시작 주소 + (인덱스×4바이트)
= 5241668 + (2×4바이트)
= 5241676
```

2차원 배열을 이용한 선형 리스트 표현

[표 3-5]와 같이 2021~2022년 분기별 노트북 판매량에 대한 리스트를 생각해 보자.

표 3-5 2021~2022년 분기별 노트북 판매량 리스트

연도 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
2021년	63	84	140	130
2022년	157	209	251	312

[표 3-5]는 분기와 연도를 모두 표현해야 하므로 순서가 두 종류 필요하다. 이 때는 [그림 3-6]과 같이 행 인덱스와 열 인덱스가 있는 2차원 배열을 사용한다.

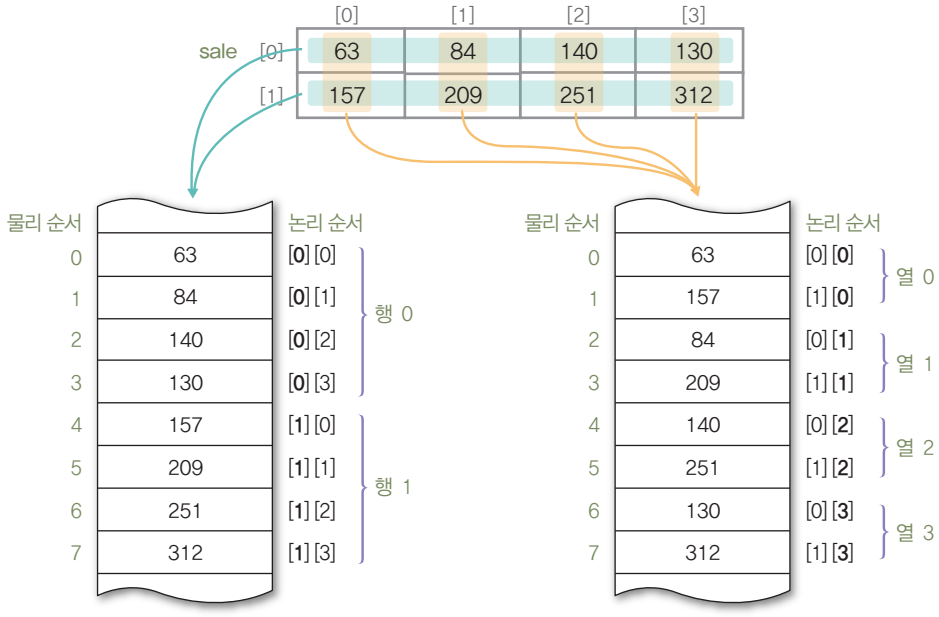
```
int sale[2][4] = { { 63, 84, 140, 130 },
                   { 157, 209, 251, 312 } };
```

sale	[0]	[1]	[2]	[3]
[0]	63	84	140	130
[1]	157	209	251	312

그림 3-6 2021~2022년 분기별 판매량 선형 리스트의 예

2차원 배열 구조를 논리적으로 표현할 때는 행과 열의 구조로 나타내지만, 실제로 메모리에는 1차원 구조로 저장된다. 2차원인 논리적 순서가 1차원인 물리적 순서로 변환되는 방법에는 인덱스를 기준으로 하는 행 우선 순서Row Major Order 방법과 마지막 인덱스를 기준으로 하는 열 우선 순서Column Major Order 방법이 있다.

행 우선 순서 방법은 행을 기준으로 같은 행 안에 있는 열을 먼저 저장하는 방법이다. [그림 3-6]의 2차원 배열을 행 우선 순서 방법으로 메모리에 저장하면 [그림 3-7]의 (a)와 같이 sale[0][0]=63, sale[0][1]=84, sale[0][2]=140, sale[0][3]=130, sale[1][0]=157, sale[1][1]=209, sale[1][2]=251, sale[1][3]=312 순서가 된다. 열 우선 순서 방법은 열을 기준으로 하여 같은 열 안에 있는 행을 먼저 저장하는 방법으로 (b)와 같이 sale[0][0]=63, sale[1][0]=157, sale[0][1]=84, sale[1][1]=209, sale[0][2]=140, sale[1][2]=251, sale[0][3]=130, sale[1][3]=312 순서가 된다.



(a) 행 우선 순서 방법 (b) 열 우선 순서 방법

그림 3-7 2차원 논리 순서를 1차원 물리 순서로 변환

행 우선 순서 방법이냐 열 우선 순서 방법이냐에 따라 물리적 저장 순서가 달라지므로 배열의 원소 위치를 계산하는 방법도 달라져야 한다. 행의 개수가 n_i이고 열의 개수가 n_j인 2차원 배열 A[n_i][n_j]의 시작 주소가 α고, 원소 길이가 ℓ이라면, i행 j열 원소, 즉 A[i][j]의 위치는 행 우선 순서 방법에서는 $\alpha+(i\times n_j+j)\times\ell$이 되고, 열 우선 순서 방법에서는 $\alpha+(j\times n_i+i)\times\ell$이 된다.

논리 순서를 물리 순서로 변환하는 방법은 프로그래밍 언어의 컴파일러에 따라 결정되는데, C 컴파일러는 행 우선 순서 방법을 사용한다.

SELF TEST_ C 프로그램에서 다음과 같이 배열 a를 선언하였다. 배열 a가 할당된 시작 주소를 10000이라고 가정했을 때, ① a[2][8] 주소와 ② a[2][8]이 몇 번째 원소인지 구하시오.

```
int a[4][10];
```

[예제 3-2]는 2021~2022년 분기별 판매량 선형 리스트를 2차원 배열로 구현하고, 논리적 순서와 물리적 순서가 일치하는 순차 구조인지 확인하는 프로그램이다.

예제 3-2 2차원 배열의 논리적 · 물리적 순서 확인하기

```
01  #include <stdio.h>
02
03  int main(void) {
04      int i, n = 0, *ptr;
05      int sale[2][4] = {{63, 84, 140, 130},
06                        {157, 209, 251, 312}}; // 2차원 배열의 초기화
07
08      ptr = &sale[0][0];
09      for (i = 0; i<8; i++) {
10          printf("\n address : %u sale %d = %d", ptr, i, *ptr);
11          ptr++;
12      }
13      getchar(); return 0;
14  }
```

```
address: 1374812 sale 0 = 63
address: 1374816 sale 1 = 84
address: 1374820 sale 2 = 140
address: 1374824 sale 3 = 130
address: 1374828 sale 4 = 157
address: 1374832 sale 5 = 209
address: 1374836 sale 6 = 251
address: 1374840 sale 7 = 312
```

08행은 배열 sale의 시작 주소인 &sale[0][0]을 포인터 ptr에 지정한다.

09~12행은 배열 sale의 첫 번째 원소부터 마지막 원소까지 메모리에 저장된 순서대로 출력한다.

10행 : 포인터 ptr에 저장된 배열 원소의 주소를 %u 형식으로 출력하고, 포인터 ptr이 가리키는 참조값(*ptr), 즉 배열 원소의 값을 %d 형식으로 출력한다.

11행 : 현재 포인터 ptr에 저장된 주소를 다음 자리, 즉 현재 위치에서 4바이트인 int만큼 이동하여 배열의 다음 원소 주소로 변경한다.

[예제 3-2]의 배열 sale에서 sale[1][2]의 위치를 행 우선 순서 방법으로 계산해 보면 시작 주소 α = 1374812, n_i=2, n_j=4, i=1, j=2, ℓ=4이므로 다음과 같이 계산할 수 있다. 값이 실행 결과와 일치하므로 C 컴파일러가 행 우선 순서 방법을 사용한다는 것을 확인할 수 있다.

$$\alpha+(i\times n_j+j)\times \ell$$
$$=1374812+(1\times 4+2)\times 4$$
$$=1374812+24$$
$$=1374836$$

3차원 배열을 이용한 선형 리스트 표현

[표 3-6]은 1팀과 2팀에 대한 2021~2022년 분기별 노트북 판매량에 대한 리스트이다.

표 3-6 1팀과 2팀의 분기별 노트북 판매량 리스트

팀	연도 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
1팀	2021년	63	84	140	130
	2022년	157	209	251	312
2팀	2021년	59	80	130	135
	2022년	149	187	239	310

[표 3-6]의 리스트는 팀 순서도 나타내야 하므로 세 종류의 순서를 표현해야 한다. 이럴 때는 면 인덱스, 행 인덱스, 열 인덱스가 있는 3차원 배열을 사용한다.

```
int sale[2][2][4] = { { { 63, 84, 140, 130 },
                        { 157, 209, 251, 312 } },
                      { { 59, 80, 130, 135 },
                        { 149, 187, 239, 310 } }
};
```

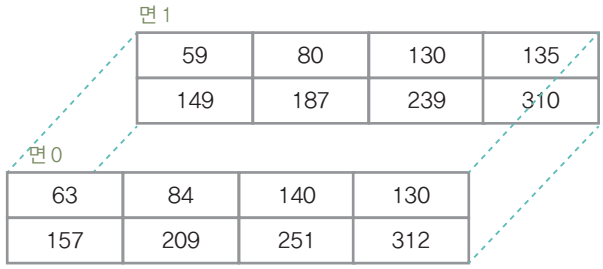
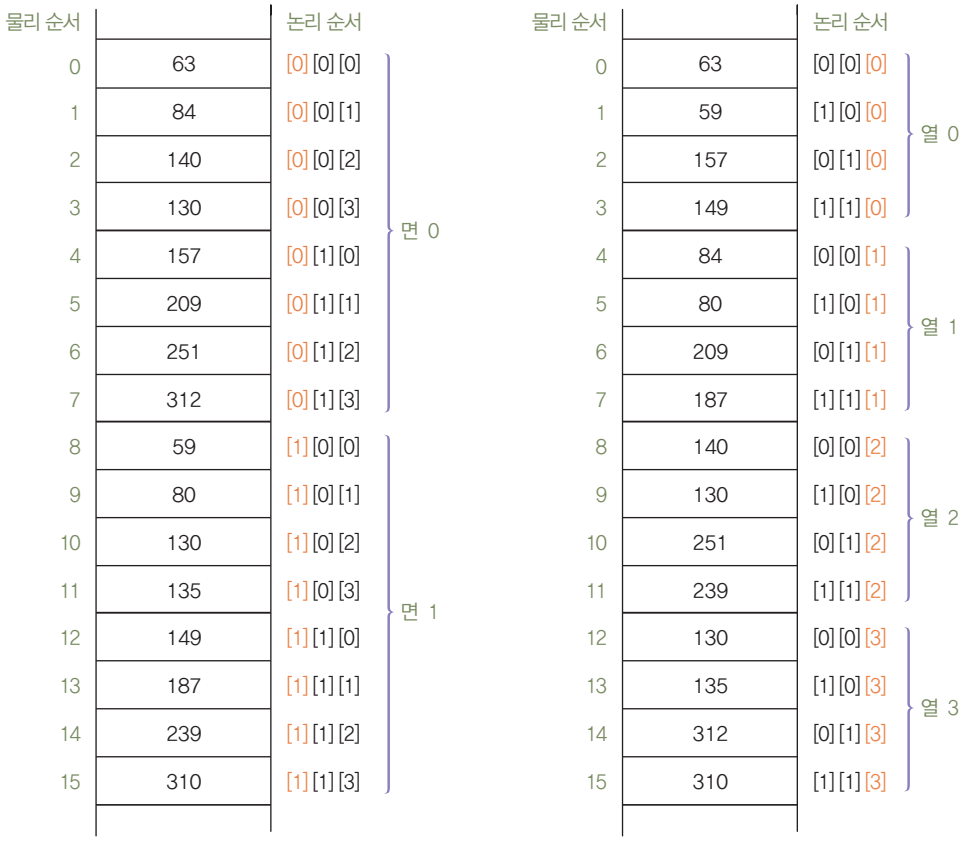


그림 3-8 선형 리스트의 3차원 배열 예

3차원 배열의 구조는 논리적인 구조일 뿐이고, 메모리에 저장되는 물리 구조는 2차원 배열처럼 1차원의 선형 구조가 된다. 3차원 논리 구조를 1차원의 물리 구조로 변환하는 방법 역시 첫 번째 인덱스인 면을 기준으로 하는 면 우선 순서 방법과 마지막 인덱스인 열을 기준으로 하는 열 우선 순서 방법이 있다.

3차원 논리 구조에 대한 면 우선 순서 방법은 면을 1차 기준으로 하여 같은 면 안에 있는 행을 먼저 저장하는데, 이때 다시 행을 2차 기준으로 하여 같은 행 안에 있는 열을 저장한다. [그림 3-8]의 3차원 배열 sale을 면 우선 순서로 저장하면 먼저 면 0을 저장해야 하므로 [그림 3-9]의 (a)와 같이 sale[0][0][0]=63, sale[0][0][1]=84, sale[0][0][2]=140, sale[0][0][3]=130, sale[0][1][0]=157, sale[0][1][1]=209, sale[0][1][2]=251, sale[0][1][3]=312 순서가 된다. 그리고 면 1을 저장하므로 다음 순서는 sale[1][0][0]=59, sale[1][0][1]=80, sale[1][0][2]=130, sale[1][0][3]=135, sale[1][1][0]=149, sale[1][1][1]=187, sale[1][1][2]=239, sale[1][1][3]=310이 된다.

다음으로 열 우선 순서 방법은 열을 1차 기준으로 하여 같은 열 안에 있는 행을 먼저 저장하는데, 이때 행을 2차 기준으로 하여 같은 행에 대한 면을 저장한다. 3차원 배열 sale을 열 우선 순서 방법으로 저장하면 [그림 3-9]의 (b)와 같이 sale[0][0][0]=63, sale[1][0][0]=59, sale[0][1][0]=157, sale[1][1][0]=149, sale[0][0][1]=84, sale[1][0][1]=80, sale[0][1][1]=209, sale[1][1][1]=187, sale[0][0][2]=140, sale[1][0][2]=130, sale[0][1][2]=251, sale[1][1][2]=239, sale[0][0][3]=130, sale[1][0][3]=135, sale[0][1][3]=312, sale[1][1][3] =310 순서가 된다.



(a) 면 우선 방법 (b) 열 우선 방법

그림 3-9 3차원의 논리 순서에 대한 1차원의 물리 순서 변환

면의 개수가 n_i이고 행의 개수가 n_j, 열의 개수가 n_k인 3차원 배열 A[n_i][n_j][n_k]의 시작 주소가 α이고 원소의 길이가 ℓ일 때, i면 j행 k열 원소, 즉 A[i][j][k]의 위치는 면 우선 순서 구조에서 $\alpha+\{(i\times n_j\times n_k)+(j\times n_k)+k\}\times\ell$이 되고, 열 우선 순서 구조에서는 $\alpha+\{(k\times n_i\times n_j)+(j\times n_i)+i\}\times\ell$이 된다.

SELF TEST_ C 프로그램에서 다음과 같이 배열 b를 선언하였다. 배열 b가 할당된 시작 주소를 10000이라고 가정했을 때, ① b[1][2][8] 주소와 ② b[1][2][8]이 몇 번째 원소인지 구하시오.

```
int b[3][4][10];
```

[예제 3-3]은 1팀과 2팀의 2021~2022년 분기별 노트북 판매량 선형 리스트를 3차원 배열로 구현하고 논리적 순서와 물리적 순서가 일치하는 순차 구조인지 확인하는 프로그램이다.

예제 3-3 3차원 배열의 논리적 · 물리적 순서 확인하기

```
01  #include <stdio.h>
02
03  int main(void) {
04     int i, n = 0, *ptr;
05     int sale[2][2][4] =  {{{63, 84, 140, 130}, // 3차원 배열의 초기화
06                         {157, 209, 251, 312}},
07                         {{59, 80, 130, 135},
08                         {149, 187, 239, 310}}};
09
10     ptr = &sale[0][0][0];
11     for (i = 0; i<16; i++) {
12        printf("\n address: %u sale %2d = %3d", ptr, i, *ptr);
13        ptr++;
14     }
15     getchar(); return 0;
16  }
```

```
address : 1374524 sale  0  = 63
address : 1374528 sale  1  = 84
address : 1374532 sale  2  = 140
address : 1374536 sale  3  = 130
address : 1374540 sale  4  = 157
address : 1374544 sale  5  = 209
address : 1374548 sale  6  = 251
address : 1374552 sale  7  = 312
address : 1374556 sale  8  = 59
address : 1374560 sale  9  = 80
address : 1374564 sale 10  = 130
address : 1374568 sale 11  = 135
address : 1374572 sale 12  = 149
address : 1374576 sale 13  = 187
address : 1374580 sale 14  = 239
address : 1374584 sale 15  = 310
```

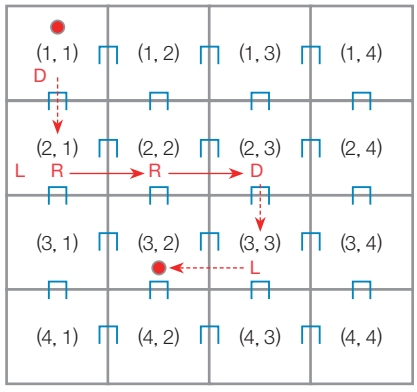
응용예제
01

게이트볼 공은 어디에?

한아는 마네의 작품 중 《크로켓 게임》을 좋아한다. 문득 우리나라에서 크로켓 게임을 할 수 없을까 알아보다가 그와 유사한 '게이트볼'이 있다는 것을 알게 되었다. 한아는 게이트볼 훈련장을 찾았다.

게이트볼 훈련장은 N×N 크기의 정사각형으로 되어있으며 1×1 크기의 정사각형으로 나눈 내부 공간의 경계선마다 게이트(⊓)가 설치되어 있다. 다음은 4×4 크기의 훈련장이다.

▶ 마네의 크로켓 게임(1873년 작)



▶ 4×4 크기의 게이트볼 훈련장

(1, 1) 공간을 보면 왼쪽과 위는 경기장 밖이므로 제외하고 오른쪽과 아래에 게이트가 설치되어 있다. (1, 2) 공간을 보면 위를 제외하고 왼쪽, 오른쪽, 아래에 게이트가 설치되어 있고 (2, 3) 공간은 왼쪽, 오른쪽, 위, 아래 네 방향으로 공을 칠 수 있으므로 네 개의 게이트가 있다. 시작 위치는 항상 (1, 1)이다.

훈련장에서는 연습생의 스윙 방향을 기록한다. 시작 위치 (1, 1)에서 스윙 기록을 따라가면 공을 찾을 수 있다. 스윙 기록은 현재 위치에서 스윙한 방향 즉, L^{Left}, R^{Right}, U^{Up}, D^{Down}로 표시된다. 스윙한 방향에 게이트가 없으면 벽에 헛스윙을 한 것이므로 공은 이동 없이 제자리에 있게 된다.

예를 들어, 4×4 크기의 훈련장에서 스윙 기록이 [D L R R D L]이라면 공은 (1, 1)에서 D이므로 (2, 1)로 이동, L이므로 헛스윙, 다시 (2, 1)에서 R이므로 (2, 2)로 이동하고 R이므로 (2, 3)으로 이동, D이므로 (3, 3)으로 이동, L이므로 (3, 2)로 이동하였다. 따라서 공의 위치는 (3, 2)가 된다.

문제 **공의 마지막 위치 구하기**

한아가 훈련을 시작하기 위해 공을 시작 위치인 (1, 1)에 둔다. 공은 어디로 갈까? 스윙 기록을 알고 있을 때 공의 마지막 위치를 출력하는 프로그램을 작성하라.

① 입력 조건
- 첫째 줄에 훈련장 크기를 나타내는 N이 주어진다. ($1 \le N \le 50$)
- 둘째 줄에 스윙 기록이 주어진다. ($1 \le$ 스윙 횟수 ≤ 50)

② 출력 조건
- 첫째 줄에 공의 마지막 위치를 (x,y) 형태로 출력한다.

③ 입출력 예시

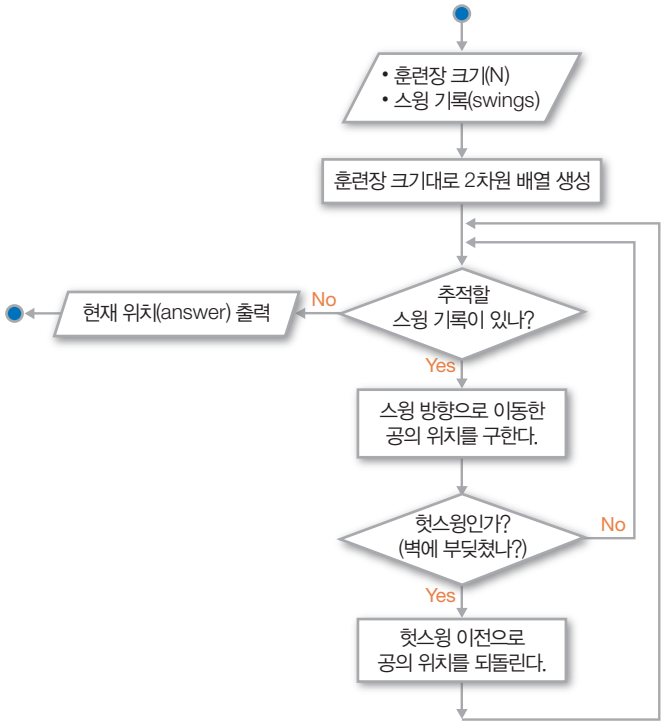
입력	출력
4 DLRRDL	(3, 2)

문제 해결

① 설계

- N×N 훈련장을 순차 자료구조인 2차원 배열로 생성한다.
- 내부 공간의 좌표를 2차원 배열의 행과 열 인덱스와 연관하여 처리한다.
- 헛스윙 여부를 판단하여 헛스윙인 경우에는 자리를 이동하지 않는다.

② 순서도



응용예제
02

TDM 축소하기

자연어 처리natural language processing 분야에서 임베딩embedding은 자연어를 컴퓨터가 이해할 수 있는 숫자인 벡터vector로 나타내는 것이다. 가장 간단한 임베딩은 문서document에 나타난 단어term의 빈도를 벡터로 사용하는 것이다. 아래와 같은 단어별 출현 빈도 행렬을 단어–문서 행렬TDM : Term-Document Matrix이라고 하는데 TDM에서 행은 단어에 대응하며 전체 행의 개수는 전체 문서에 포함되어 있는 단어의 총 개수가 된다. 열은 문서에 대응하며 열의 개수는 처리할 문서의 총 개수가 된다. 행렬의 원소값은 행에 대응하는 단어가 열에 대응하는 문서에 나타난 횟수(출현 빈도)가 된다. 아래 TDM에서 '순차'라는 단어는 문서1에 12번, 문서6에 7번 나타났고 나머지 문서에는 한 번도 나타나지 않았다.

	문서1	문서2	문서3	문서4	문서5	문서6	문서7	문서8	문서9	문서10
오늘	0	1	0	0	0	0	1	0	2	3
데이터	9	0	0	1	0	4	0	0	0	0
날씨	0	2	0	0	0	0	1	0	1	5
순차	12	0	0	0	0	7	0	0	0	0
강아지	0	1	0	0	0	0	0	0	3	0

TDM에서 행의 개수는 분석할 말뭉치 전체의 어휘 수와 같기 때문에 몇만에서 몇십 만이 되는 경우가 보통이다. 그리고 문서 하나에서 모든 단어가 골고루 다 사용되는 경우는 거의 없기 때문에 TDM의 원소값은 대부분 0이다. 분석에 사용되지 않는 0의 값이 많은 TDM을 그대로 계산하면 메모리 사용량도 많아지고 계산 시간도 길어진다. 따라서 자연어 처리 알고리즘 성능 향상을 위해 TDM을 축소해야 한다.

문제 TDM 축소 행렬 구하기

0이 아닌 실제 빈도 값만 추출한 TDM 축소 행렬을 출력하는 프로그램을 작성하라.

① 입력 조건
- 첫째 줄에 단어 개수 N과 단어 목록이 주어진다. ($5 \le N \le 30$)
- 둘째 줄에 문서 개수 M이 주어진다. ($10 \le M \le 50$)
- 셋째 줄부터 N+2번째 줄까지 TDM의 원소값이 한 줄에 M개씩 주어진다.

② 출력 조건
- 첫째 줄부터 TDM 축소 행렬을 출력한다.

③ 입출력 예시

```
5 오늘 데이터 날씨 순차 강아지
10
0 1 0 0 0 0 1 0 2 3
9 0 0 1 0 4 0 0 0 0
0 2 0 0 0 0 1 0 1 5
12 0 0 0 0 7 0 0 0 0
0 1 0 0 0 0 0 0 3 0
```

→

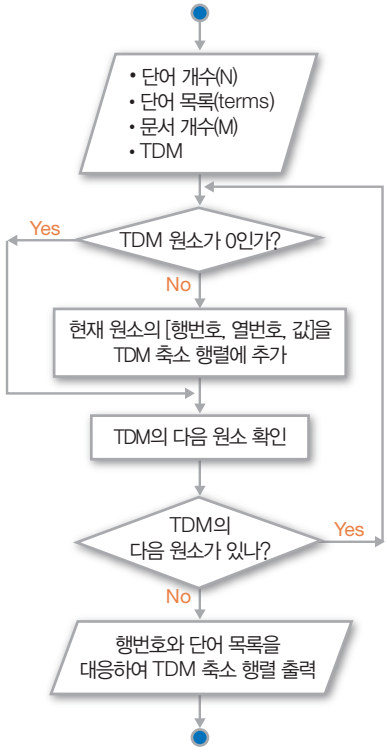
```
오늘 문서2 1
오늘 문서7 1
오늘 문서9 2
오늘 문서10 3
데이터 문서1 9
데이터 문서4 1
데이터 문서6 4
날씨 문서2 2
날씨 문서7 1
날씨 문서9 1
날씨 문서10 5
순차 문서1 12
순차 문서6 7
강아지 문서2 1
강아지 문서9 3
```

문제 해결

① 설계

TDM은 N×M 행렬로 대부분의 값이 0으로 채워진 희소행렬이다. TDM 축소 행렬은 희소행렬에 대한 정보 압축 행렬이 된다. 0이 아닌 값이 있는 원소를 찾아서 [행 번호, 열 번호, 값]의 형태로 정리하여 열의 크기가 3인 2차원 배열을 구성한다.

② 순서도



요약

01 선형 리스트 개념과 순차 자료구조의 관계

선형 리스트Linear List는 리스트에 나열한 원소들이 순서대로 나열된 리스트로 순서 리스트Ordered List라고도 한다. 메모리에 저장되는 방식에 따라 선형 순차 리스트와 선형 연결 리스트로 나뉘는데, 일반적으로 선형 순차 리스트를 선형 리스트라고 한다. 선형 리스트는 원소들이 나열된 논리적인 순서와 메모리에 저장되는 물리적인 순서가 같은 순차 자료구조이다. 순차 자료구조에서는 원소들이 순서대로 연속하여 저장된다. 시작 위치가 α이고, 원소 길이가 ℓ인 리스트에서 i번째 원소의 위치는 $\alpha+(i-1)\times\ell$이 된다.

02 선형 리스트의 연산

선형 리스트에서 삽입 또는 삭제 연산을 하면 원소들의 논리적 순서가 바뀌기 때문에 물리적 순서도 바뀌어야 한다. 말 그대로 원소 위치를 물리적으로 옮기는 작업을 해야 한다.

- 삽입 연산에 따른 원소의 이동 횟수 = 마지막 원소의 인덱스 – 삽입할 자리의 인덱스 + 1
- 삭제 연산에 따른 원소의 이동 횟수 = 마지막 원소의 인덱스 – 삭제한 자리의 인덱스

03 선형 리스트의 구현

선형 리스트는 C 프로그래밍의 배열을 사용해 구현한다. 배열은 〈인덱스, 원소〉의 쌍으로 구성되어 메모리에 연속적으로 할당되는데, 이때 인덱스는 배열 원소의 순서를 나타내며 배열 원소들이 순서대로 메모리에 연속하여 순차 저장된다.

04 희소 다항식의 선형 리스트 표현

희소 다항식은 메모리 사용 효율성을 높이기 위해 지수에 따라 (지수+1) 크기의 배열을 생성하는 방법 대신 항의 개수에 따라 배열 크기를 결정하는 방법을 사용하여 〈지수, 계수〉 쌍을 2차원 배열에 저장한다. 다항식을 표현한 2차원 배열에서 행의 개수는 희소 다항식의 항의 개수가 된다.

05 행렬의 선형 리스트 표현

희소행렬을 처리하기 위해 선형 리스트로 표현하고, 행과 열 인덱스가 있는 2차원 배열을 사용하여 구현한다.

06 다항식의 선형 리스트 표현

다항식의 표현과 연산을 처리하기 위해 선형 리스트를 사용할 수 있다. n차 다항식 $P(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x^1 + a_0x^0$을 (n+1)개 원소를 가지는 배열을 사용하여 표현할 수 있다. 배열의 인덱스는 다항식 항의 지수를 표현하는 데 이용되고, 배열 원소에는 다항식 항의 계수가 저장된다.

연습문제

01 순차 자료구조와 관련된 것은?

① 디스크 조각 모으기 ② 재귀호출
③ 포인터 ④ 백신 프로그램

02 선형 리스트에 대한 설명으로 틀린 것은?

① 선형 리스트는 메모리에 연속된 공간을 사용한다.
② 삽입 연산에 효율적이다.
③ 배열을 사용해 구현하는 순차 자료구조이다.
④ 선형 리스트의 전체 원소를 순서대로 액세스하는 속도가 빠르다.

03 선형 리스트를 L[m][n]의 2차원 배열로 구현할 때, 선형 리스트에 저장할 수 있는 원소의 최대 개수는?

① (m + 1) × (n + 1)개 ② (m × n)개
③ (m − 1) × (n − 2)개 ④ (m × n) + 1개

04 선형 리스트를 구현한 2차원 배열 L[5][7]에서 L[3][2]의 물리적 주소는? (단, 원소 한 개의 크기는 4바이트이고, 행 우선 순서를 사용한다.)

① 0 + (3 × 7 + 2) 4 ② 0 + (3 × 5 + 2) 4
③ L + (3 × 7 + 2) 4 ④ L + (2 × 5 + 3) 4

05 배열 int array[10][200]을 행 우선 순서로 저장하는 경우에 원소 array[7][12]의 시작 주소는 몇 번지인가? (단, 배열 array의 시작 주소는 10840h로, int의 크기는 4바이트로 가정한다. 배열 첨자는 0부터 시작하며 숫자에 붙은 h는 16진수 표기를 의미한다.)

① 10804h ② 11E50h ③ 16488h
④ 108BFh ⑤ 10A3Ch

06 원소가 100개 있는 선형 리스트에서 열 번째 원소를 삭제하는 연산을 수행하였다. 원소의 이동 횟수는?

① 90 ② 91 ③ 92 ④ 93

07 3차원 배열 A(1:5, 2:4, 1:3)를 1차원 배열 B(1:45)에 행 우선으로 저장하는 경우 A(2, 3, 3)이 저장되는 B의 인덱스는?

① 12 ② 13 ③ 14 ④ 15

08 행 우선으로 저장되는 3차원 배열 a[4][5][3]이 있을 때, 이 배열의 첫 번째 원소 a[0][0][0]의 주소를 α라고 하면, a[2][3][1]의 주소를 α+i로 표현했을 때 i의 값은?

① 6 ② 40 ③ 56 ④ 168

09 K[1: 2, 1: 3, 1: 2]로 선언된 3차원 배열을 행 우선으로 1차원 배열에 저장했을 때, 아홉 번째에 저장되는 요소는?

① K(1, 2, 2) ② K(1, 3, 2) ③ K(2, 2, 1) ④ K(2, 3, 2)

10 행 우선 배열 A[3:6][2:7][8:12]에서 A[4][5][10]은 배열 A의 몇 번째 원소인가? (단, 첫 번째 원소는 A[3][2][8]이고 마지막 원소는 A[6][7][12]이다.)

① 45 ② 46 ③ 47 ④ 48

11 행 우선으로 배열값을 저장하는 C 언어에서 3차원 배열 A[4][2][3]을 선언하였다. A[0][0][0]부터 A[3][1][2]에 정수값 1~24를 행 우선 순서에 따라 차례대로 저장할 때, A[2][1][2]에 저장되는 값은?

① 17 ② 18 ③ 19 ④ 20

12 희소행렬에 대한 설명으로 옳지 않은 것은?

① 원소값이 대부분 0으로 구성되어 있다.
② 2차원 배열로 표현하면 특정 항목의 접근이 용이하다.
③ 연결 리스트 구조로 표현하더라도 행렬의 덧셈 연산을 할 수 있다.
④ 연결 리스트 구조로 표현하면 기억 공간이 낭비된다.

13 m×n 크기의 정수값 희소행렬을 정수형 배열에 저장하려고 한다. 가장 효과적인 저장 방법을 사용할 때, 필요한 배열 크기는? (단, t는 0이 아닌 희소행렬 원소의 개수이며, 배열 크기는 배열 원소의 수를 의미한다.)

① m × n ② t ③ m+n+t ④ 3(t+1)

14 값이 0인 원소들의 비율이 90%인 1000×1000 희소행렬 표현에 관한 설명 중 옳은 것은?

① 2차원 배열로 모든 원소들을 표현하는 것이 저장 관점에서 효율적이다.
② 2차원 배열로 0이 아닌 원소들만 표현하기 위해서 저장 공간이 1000개 필요하다.
③ 0이 아닌 원소들만 연결 리스트로 표현하는 것이 배열로 표현하는 것보다 전치행렬 연산에 효율적이다.
④ 0이 아닌 원소들만 3원소 쌍(행의 인덱스, 열의 인덱스, 값)으로 배열에 저장하는 것이 저장 관점에서 효율적이다.

15 X , Y , Z는 m×n 희소행렬에서 원소값이 0이 아닌 k개의 각 원소를 표현하기 위해 〈행, 열, 값〉 3원소 쌍을 사용하는 2차원 배열이며, 변환 규칙은 (가)와 같다. 행렬 P와 Q가 (나)와 같이 X, Y로 각각 표현되었을 때, P와 Q를 곱한 행렬 R을 Z로 표현할 경우, 이에 대한 설명으로 옳지 않은 것은? (단, 행렬의 행 번호와 열 번호는 0부터 시작한다.)

(가)

- 행 우선 배열이며, 크기는 (k+1)×3이다.
- 첫 번째 행의 세 개 열에는 희소행렬의 행의 크기(m), 열의 크기(n), 0이 아닌 원소의 수(k)를 순서대로 저장한다.
- 나머지 k개의 각 행에는 0이 아닌 원소들에 대한 정보가 저장되며, 해당 행의 각 열에는 원소들의 행 번호, 열 번호, 저장된 값이 각각 저장된다.
- 원소들에 대한 정보를 저장할 때는 원소들의 행 번호와 열 번호가 차례대로 오름차순이 되도록 저장한다.

(나)

X

행 \ 열	0	1	2
0	3	4	3
1	0	3	4
2	1	1	7
3	2	3	1

Y

행 \ 열	0	1	2
0	4	2	3
1	0	1	1
2	3	0	3
3	3	1	2

① Z[0][1] = 2이다.
② Z[0][2] = 4이다.
③ Z[1][0] = 0이다.
④ Z[3][2] = 3이다.

16 다항식을 표현하기 위하여 다음의 구조체를 정의하였다. 다항식 $A(x)=\sum_{i=0}^{n} a_i x^i$에 대해 poly.degree = n, poly.coef[i] = a_{n-i}로 표현되며, 구조체 멤버 중 degree는 다항식의 최고 차수를 저장하고 배열 coef[MAX_DEGREE]는 다항식의 최고 차수 항부터 최저 차수 항까지의 계수를 차례로 저장한다. 이때, 다항식 A(x)를 저장하기 위한 구조체 변수 poly의 선언 과정에서 다항식 $100x^5+60x$를 초기화하기 위한 ㉠의 내용으로 옳은 것은?

```
#define MAX_DEGREE 6

struct polynomial {
   int degree;
   int coef[MAX_DEGREE];
} poly = ㉠;
```

① {5, {100, 1, 60}}
② {5, {100, 60, 0, 0, 0, 1}}
③ {5, {100, 0, 0, 0, 60, 0}}
④ {5, {100, 0, 60}}

IT@COOKBOOK

단계별 그림과 삽화로 이론을 다지고 C 언어로 구현해 보는 자료구조 입문서

자료를 구조화하는 다양한 방법을 단계별 그림과 삽화를 곁들여 쉽게 설명하고, 자료구조의 핵심 알고리즘을 C 프로그램으로 차근차근 구현해 보도록 구성하였습니다. 이론과 실습을 병행할 수 있어 학습 효과를 극대화할 수 있고, 주어진 문제에 대한 최적의 해결 방법을 선택하고 활용할 수 있도록 도와주어 자료구조를 처음 배우는 학생들에게 안성맞춤입니다. 특히 4판에 새롭게 추가된 코딩 테스트 유형의 응용예제를 통해 문제 해결 과정을 단계별로 학습할 수 있습니다.

1-2장 — 자료구조의 개념 및 구현을 위한 C 프로그래밍 기법

- 자료구조의 개념과 종류
- 알고리즘의 개념과 성능 분석
- 자료의 표현과 추상화
- 자료구조 구현을 위한 C 프로그래밍 기법

3-8장 — 핵심 자료구조의 단계별 구현 : 추상 자료형 정의 → 알고리즘 설계 → C 프로그램 작성

- 리스트 : 선형 리스트, 단순 연결 리스트, 원형 연결 리스트, 이중 연결 리스트
- 스택 : 순차 스택, 연결 스택
- 큐 : 순차 큐, 연결 큐, 원형 큐
- 데크 : 순차 데크, 연결 데크
- 트리 : 순차 이진 트리, 연결 이진 트리
- 그래프 : 순차 그래프, 연결 그래프

9-10장 — 자료구조의 응용

- 정렬 : 선택 · 버블 · 퀵 · 삽입 · 셸 · 병합 · 기수 · 히프 · 트리 정렬
- 검색 : 순차 · 이진 검색, 이진 트리 검색, 해싱

응용예제

컴퓨터공학 / 자료구조

9 791156 645412 93000

ISBN 979-11-5664-541-2 정가 29,000원