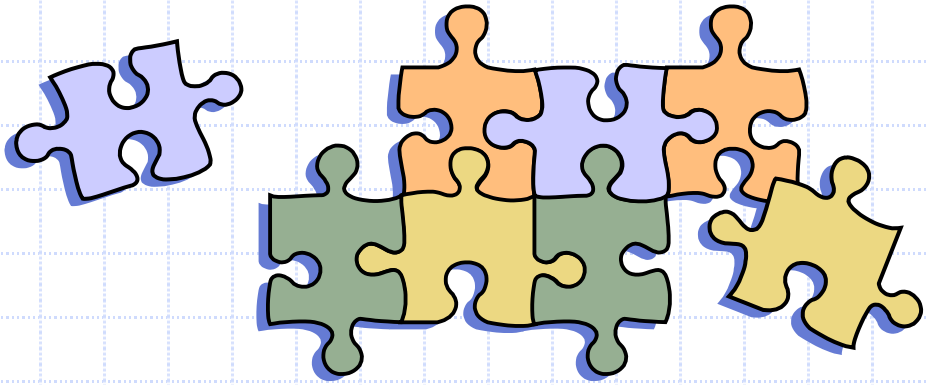
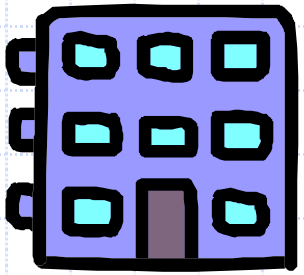
기초 데이터구조



Outline

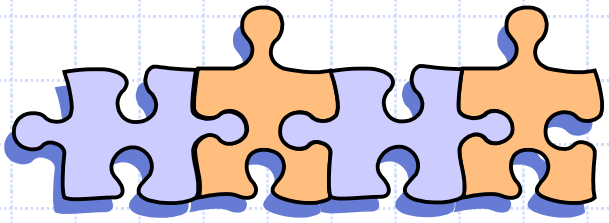
- 3.1 데이터구조의 기본 재료
- 3.2 배열
- 3.3 연결리스트

데이터구조의 기본 재료



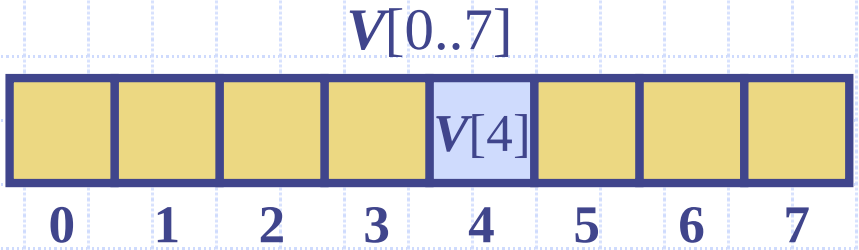
- 건물과 비유
  - **데이터구조:** 재료, 자재, 구조
  - **알고리즘:** 조명, 냉난방, 환기, 개폐 시스템
- 기본 재료
  - **배열**
  - **연결리스트**
- 차후의 고급 데이터구조를 구축하는 기본 재료
- 추상적이기 보다는 구체적(컴퓨터 구조에 가까움)

배열

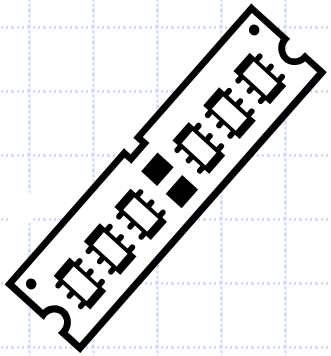


- **배열**(array): 순차 기억장소에 할당된 유한 개수의 동일 자료형 데이터원소들
  - **배열명**(array name), V : 배열 전체를 일컫는 기호
  - **배열크기**(array size), N : 원소를 저장하는 셀들의 개수
  - **배열첨자**(array index), i : 셀의 순위 (즉, 상대적 위치)
    - 시작: 0 또는, 일반적으로, LB(lower bound)
    - 끝: $N-1$ 또는, 일반적으로, UB(upper bound)
  - **배열원소**(array element), $V[i]$: 배열 V의 첨자 i에 저장된 원소
  - **배열표시**(array denotation): $V[LB..UB]$

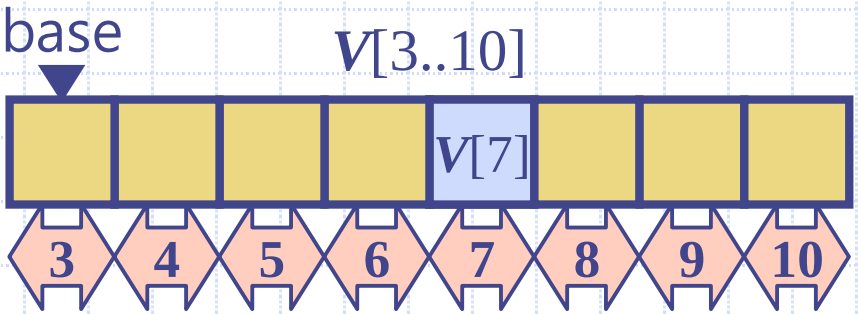
예:
한 학생의 8과목에서의 점수



배열의 메모리 할당



- 배열의 선언: $V[LB..UB]$
- 컴파일 시, 배열의 셀들은 **베이스**(base)라 불리는 배열의 첫째 셀 위치부터 시작하여 차례로 할당
- 각 셀은 베이스로부터 **오프셋**(offset) 만큼 떨어짐
  - $V[LB..UB]$의 base로부터 $V[k]$의 offset $= k - LB$
  - **예:** $V[3..10]$의 base로부터 $V[7]$의 offset $= 7 - 3 = 4$



다차원 배열

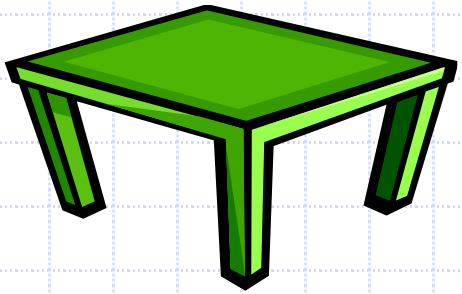
- 1차원 배열과 다른 점
  - 배열의 방들은 $n > 1$차원에 할당
  - **배열크기:** 각 차원 크기의 곱
  - **배열첨자**, $i_1, i_2, \ldots, i_n$: 각 차원에서의 방의 순위(즉, 상대적 위치)
    - 시작: $0, 0, \ldots, 0$, 또는 $LB_1, LB_2, \ldots, LB_n$
    - 끝: $N_1 - 1, N_2 - 1, \ldots, N_n - 1$, 또는 $UB_1, UB_2, \ldots, UB_n$
  - **배열원소**, $V[i_1, i_2, \ldots, i_n]$: 배열첨자들 $i_1, i_2, \ldots, i_n$에 저장된 원소
  - **배열표시**
    - $LB = 0$이면, $V[N_1 \times N_2 \ \ldots \times N_n]$
    - 일반적으로, $V[LB_1..UB_1, LB_2..UB_2, \ldots, LB_n..UB_n]$

메모리 할당

- 행우선 순서(또는 저차원우선 순서)
  - $V[LB_1..UB_1, LB_2..UB_2, \ldots, LB_n..Ub_n]$의 베이스로부터 $V[k_1, k_2, \ldots, k_n]$의 오프셋

$$= \sum_{j=1}^{n-1} \left[(k_j - LB_j) \times \prod_{m=j+1}^{n} (UB_m - LB_m + 1) \right] + (k_n - LB_n)$$

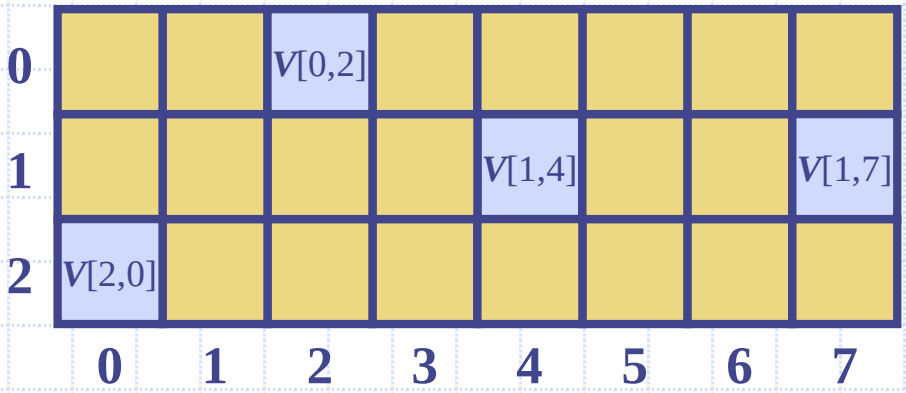
2차원 배열



- $n = 2$
- 배열표시
  - $LB = 0$이면, $V[N_1 \times N_2]$
  - 일반적으로, $V[LB_1..UB_1, LB_2..UB_2]$
- **테이블**(table)이라고도 불림
- 1차원과 2차원은 각각 **행**(row)과 **열**(column)로도 불림

예:
3학생의 8과목에서의 점수

$V[3 \times 8]$ or $V[0..2, 0..7]$



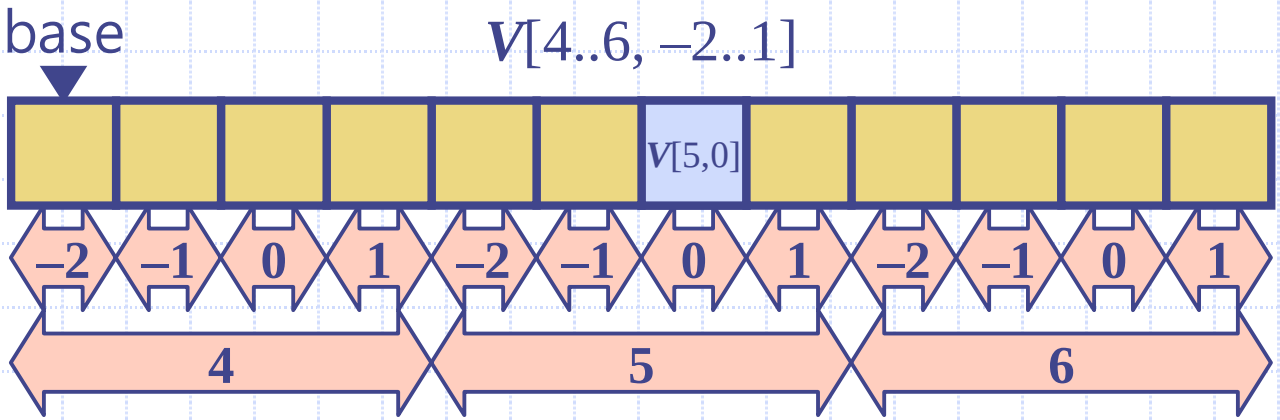
메모리 할당

- 행우선 순서(또는 저차원우선 순서)
  - $V[LB_1..UB_1, LB_2..UB_2]$의 베이스로부터 $V[k_1, k_2]$의 오프셋

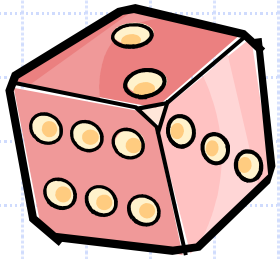
    $= [(k_1 - LB_1)(UB_2 - LB_2 + 1)] + (k_2 - LB_2)$
  - **예:** $V[4..6, -2..1]$의 베이스로부터 $V[5, 0]$의 오프셋

    $= [(5 - 4)(1 - (-2) + 1)] + (0 - (-2))$

    $= 6$



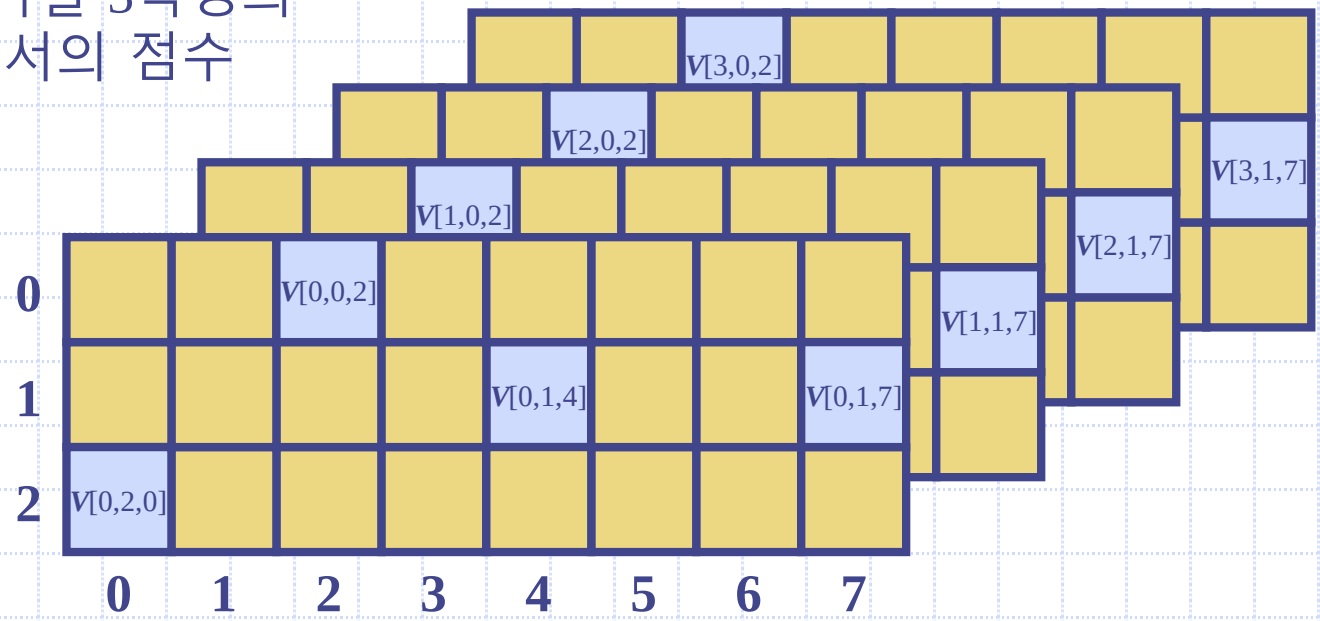
3차원 배열



- 배열표시
  - $LB = 0$이면, $V[N_1 \times N_2 \times N_3]$
  - 일반적으로, $V[LB_1..UB_1, LB_2..UB_2, LB_3..UB_3]$

예: 4분기별 3학생의 8과목에서의 점수

$V[4 \times 3 \times 8]$ or $V[0..3, 0..2, 0..7]$



메모리 할당

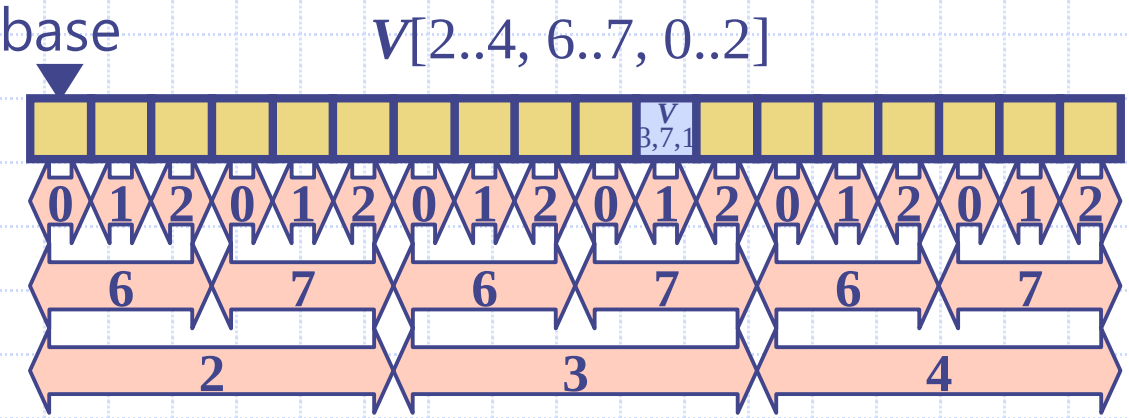
- 행우선 순서 (또는 저차원우선 순서)
  - $V[LB_1..UB_1, LB_2..UB_2, LB_3..UB_3]$의 베이스로부터 $V[k_1, k_2, k_3]$의 오프셋

    $= [(k_1 - LB_1)(UB_2 - LB_2 + 1)(UB_3 - LB_3 + 1)$

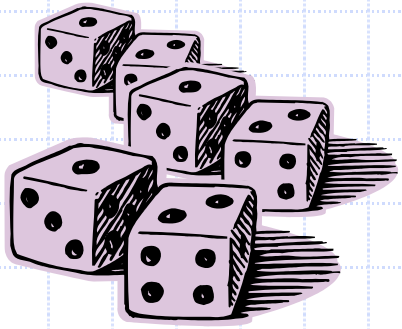
    $+ (k_2 - LB_2)(UB_3 - LB_3 + 1)] + (k_3 - LB_3)$
  - **예:** $V[2..4, 6..7, 0..2]$의 베이스로부터 $V[3, 7, 1]$의 오프셋

    $= [(3 - 2)(7 - 6 + 1)(2 - 0 + 1) + (7 - 6)(2 - 0 + 1)] + (1 - 0)$

    $= 10$



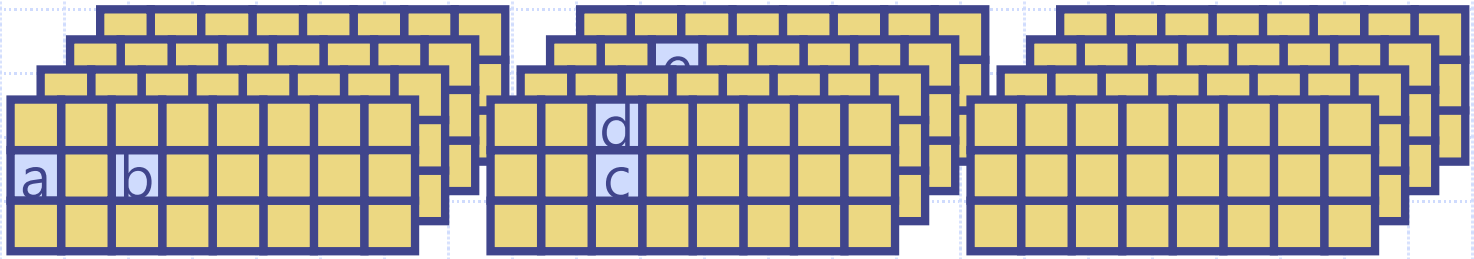
4차원 배열



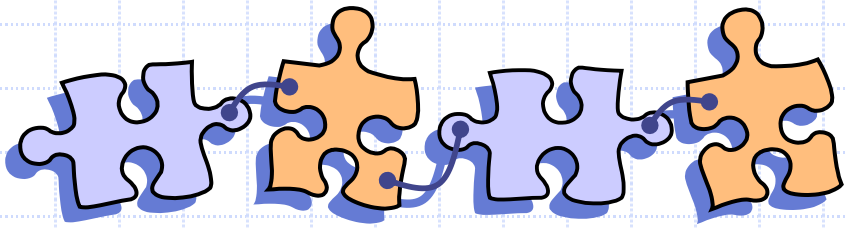
- 배열표시
  - $LB = 0$이면, $V[N_1 \times N_2 \times N_3 \times N_4]$
  - 일반적으로, $V[LB_1..UB_1, LB_2..UB_2, LB_3..UB_3, LB_4..UB_4]$

예: 3년간 4분기별 3학생의 8과목에서의 점수
(아래 그림에서 a, b, c, d, e는 각각 무엇인가?)

$V[3 \times 4 \times 3 \times 8]$ or $V[0..2, 0..3, 0..2, 0..7]$



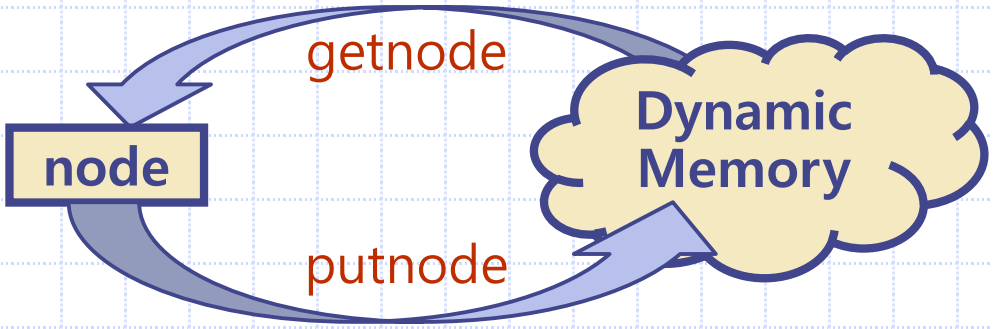
연결리스트



- **연결리스트**(linked list): 동적메모리에 할당된, 링크에 의해 연결된 유한 개수의 데이터원소 노드들
  - **연결리스트 명**(linked list name), L : 연결리스트의 시작 위치, 즉, 첫 노드의 주소
  - **연결리스트 크기**(linked list size), n : 연결리스트내 노드 수
- 연결리스트의 종류
  - 단일연결리스트(singly linked lists)
  - 이중연결리스트(doubly linked lists)
  - 원형연결리스트(circularly linked lists)
  - 헤더 및 트레일러 연결리스트(linked lists with header and trailer)
  - 이들의 복합

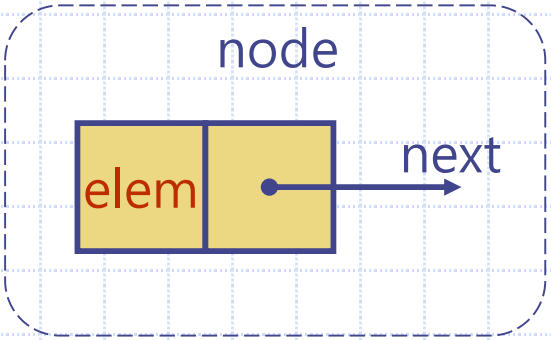
노드에 대한 메모리 할당

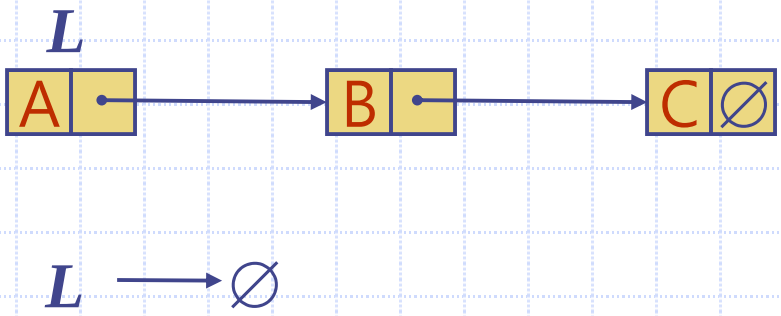
- **노드**(node): 한 개의 데이터원소를 저장하기 위해 동적 메모리(dynamic memory)에 할당된 메모리
- 노드를 위한 메모리의 동적 할당(allocation)과 해제(deallocation)는 실행시간에 system call에 의해 처리
  - getnode(): 노드를 할당하고 그 노드의 주소를 반환 (동적 메모리가 고갈된 시점이면 널포인터를 반환)
  - putnode(i): 주소 ***i***의 노드에 할당되었던 메모리의 사용을 해제하고 이를 동적 메모리에 반환 (메모리 재활용을 위함)



단일연결리스트

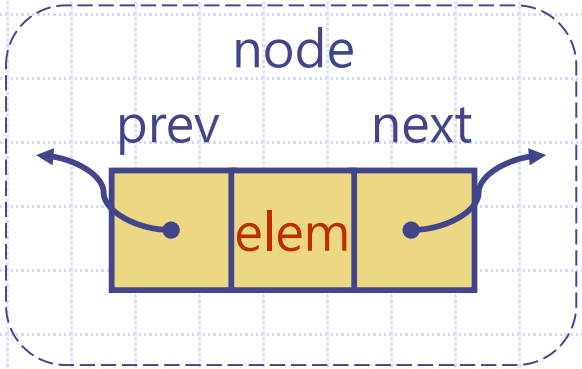
- 연속 노드로 구성된, 가장 단순한 연결 데이터구조
- 노드 저장내용
  - **원소**(element): 데이터원소
  - **링크**(link): 다음 노드의 주소 – 다음 노드가 없는 경우 널링크(∅)를 저장
- 접근점
  - 첫 노드, 즉, **헤드노드**(head node)의 주소

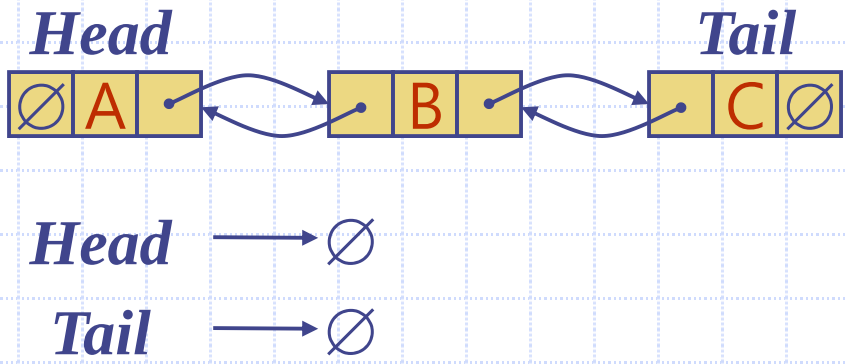




이중연결리스트

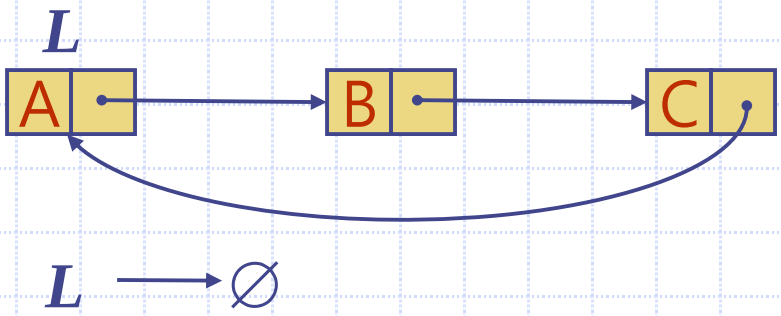
- 추가 링크를 사용하여 역방향 순회도 가능
- 노드 저장내용
  - **원소**(element)
  - **링크**(link): 다음 노드의 주소
  - **링크**(link): 이전 노드의 주소
- 접근점
  - 헤드노드(head node)의 주소
  - 테일노드(tail node)의 주소





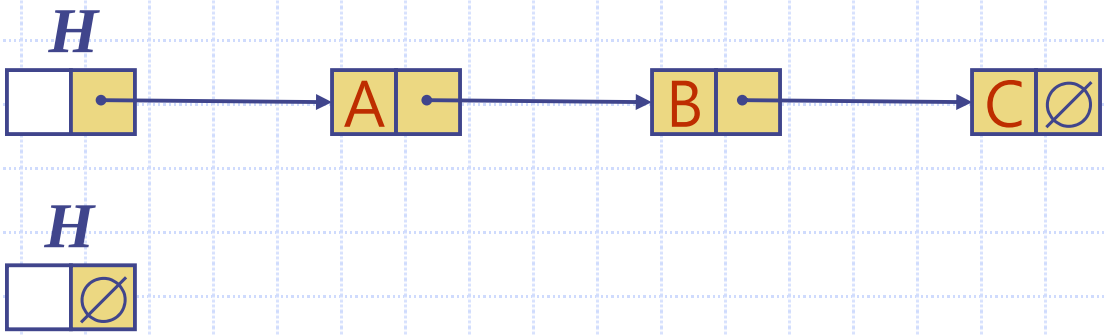
원형연결리스트

- 마지막 노드의 링크가 헤드노드의 주소
- 접근점
  - 헤드노드의 주소



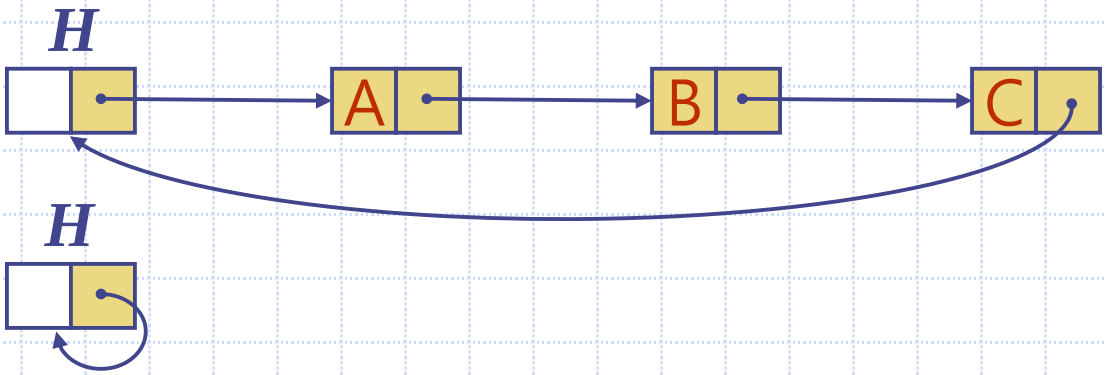
헤더와 트레일러

- 헤드노드 바로 앞에 특별한 **헤더**(header) 노드를 추가하여 작업 편의성을 증진
- 같은 목적으로 테일노드 바로 뒤에 **트레일러**(trailer) 노드 추가 가능
- 특별노드 저장내용
  - 모조 원소(dummy element)
- 접근점
  - 헤더노드(또는 트레일러노드)의 주소

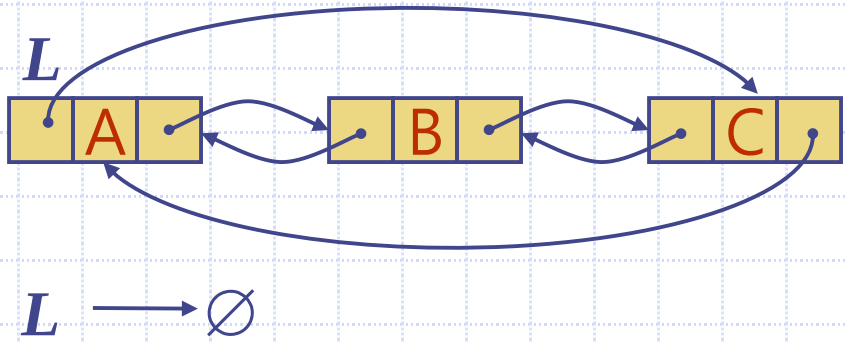


기타 연결리스트

- 원형 헤더연결리스트(circular header linked lists)

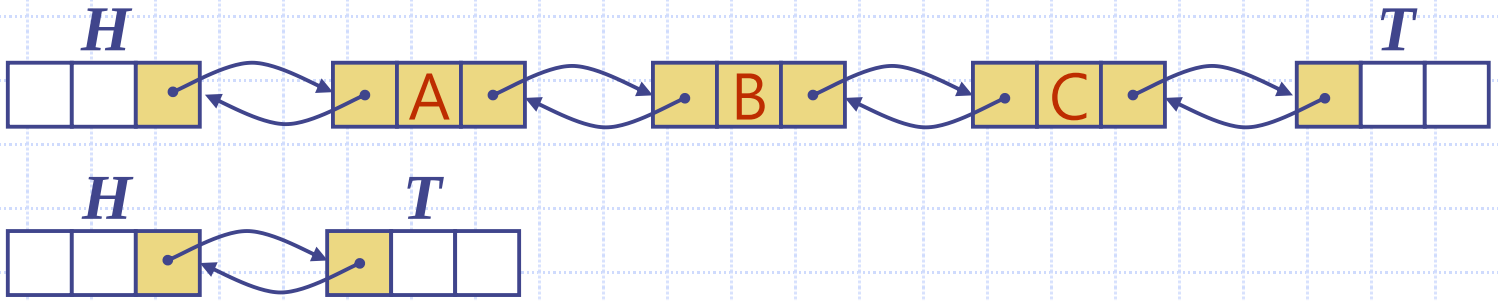


- 원형 이중연결리스트(circular doubly linked lists)



기타 연결리스트 (conti.)

- 헤더 및 트레일러 이중연결리스트



- 헤더 및 트레일러 원형 이중연결리스트