DataStructure/2021

Linked List 1

- Basic -

안태진(taejin7824@gmail.com)
GitHub(github.com/taejin1221)
상명대학교 소프트웨어학과
201821002

Contents

- Array
  - 이론
  - 장/단점
- Linked List
  - 이론
  - 장/단점
- Implementation

Contents

Array

- 이론
  - 특정 자료형의 변수들을 모아둔 자료구조
  - 시작 위치부터 얼마나 떨어져 있는지를 계산하여 접근 => index
  - index로 접근하며 변수들을 변경 및 접근
    - 삽입
      - index
    - 탐색
      - index
    - 삭제
      - index

Contents

- Array
  - 이론
  - 장/단점
- Linked List
  - 이론
  - 장/단점
- Implementation

Array

- 장/단점
  - 장점
    1. 쉽다!
    2. 빠르게 특정 변수에 접근할 수 있다

  - 단점
    1. 중간 값 삭제가 어렵다
    2. 공간 낭비가 심하다
    3. 한번 정하면 크기를 변경하기 힘들다
       - 크기가 정해져 있다

Array

- 단점 (1/3)
  1. 중간 값 삭제가 어렵다
     - 그냥 0으로 채우면 되지 않을까?
       - 0을 허용하는 값들이면 값이 있는 것으로 착각
       - 그 위치가 비기 때문에 배열이 꽉 찬 것으로 생각

     - Time Complexity: $O(n)$

     - Array_Disadvantage1.cpp

```
int delete_idx;
cin >> delete_idx;

for ( int i = delete_idx; i < n - 1; i++ )
    arr[i] = arr[i + 1];
```

```
~/Gi/Taejin-Tutoring/Week4/Practice main ?1 g++ -o AD1 Array_Disadvantage1.cpp
~/Gi/Taejin-Tutoring/Week4/Practice main ?1 ./AD1
10
0 1 2 3 4 5 6 7 8 9
5
0 1 2 3 4 6 7 8 9
```

```
~/Gi/Taejin-Tutoring/Week4/Practice main ?1 ./AD1
10
0 1 2 3 4 5 6 7 8 9
0
1 2 3 4 5 6 7 8 9
```

Array

- 단점 (2/3)
  2. 공간 낭비가 심하다
     - 안쓰는 공간이라도 Memory 할당 받고 있음
       - Memory 낭비

     - Array_Disadvantage2.cpp

```cpp
int n;
cin >> n;

int* arr = new int[n];

int size;
cin >> size;

for ( int i = 0; i < size; i++ )
    cin >> arr[i];

PrintArray( arr, size );
```

```
~/Gi/Taejin-Tutoring/Week4/Practice main g++ -o AD2 Array_Disadvantage2.cpp

~/Gi/Taejin-Tutoring/Week4/Practice main ?1 ./AD2
1000
1
1
1
```

Array

- 단점 (3/3)
  3. 한번 정하면 크기를 변경하기 힘들다
     - 크기가 정해져 있음
     - 동적 할당이라도 한번 크기를 정하면 다시 크기를 늘리거나 줄이기 힘듬

     - Array_Disadvantage3.cpp

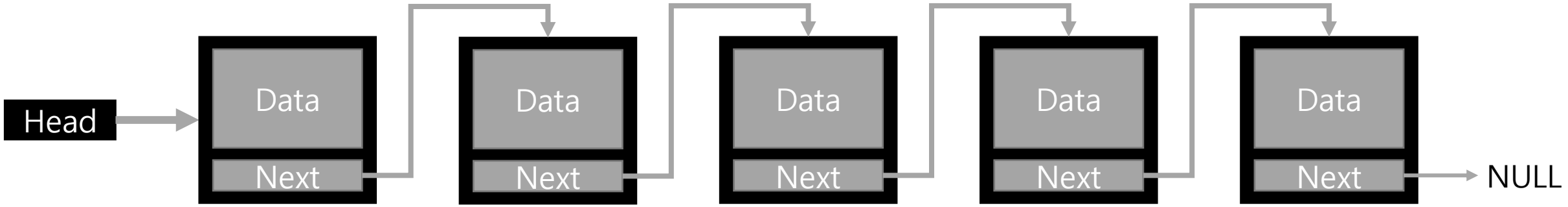
```cpp
int n;
cin >> n;

int* arr = new int[n];

for ( int i = 0; i < n; i++ )
    cin >> arr[i];

// do something

cout << "Oops I want more memory space!\n";
```

Contents

- Array
  - 이론
  - 장/단점
- Linked List
  - 이론
  - 장/단점
- Implementation

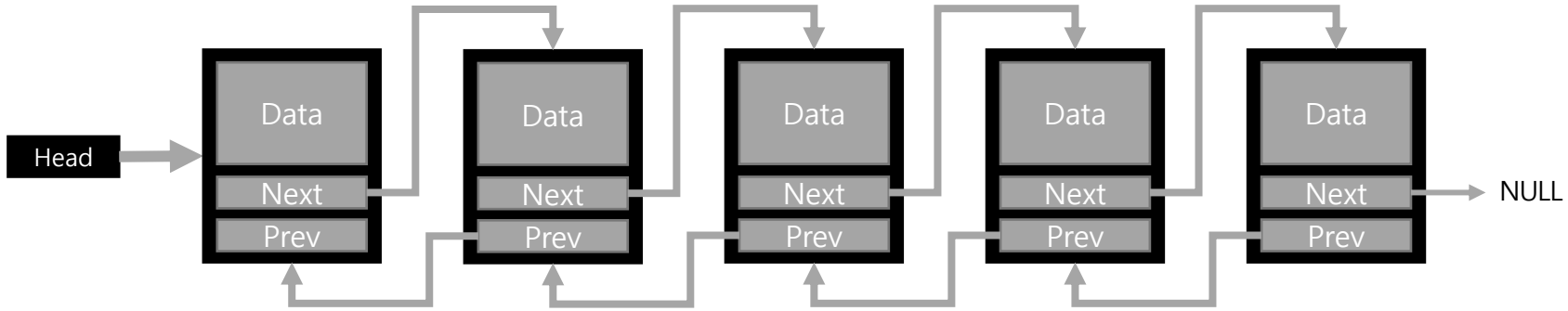
Linked List

- 이론 (1/2)
  - Array의 단점을 보완하기 위해 만들어진 Data Structure
  - 각각의 원소가 다음의 원소를 가리킴
    - Node가 value뿐만 아니라 다음 Node의 주소 값을 지니고 있음

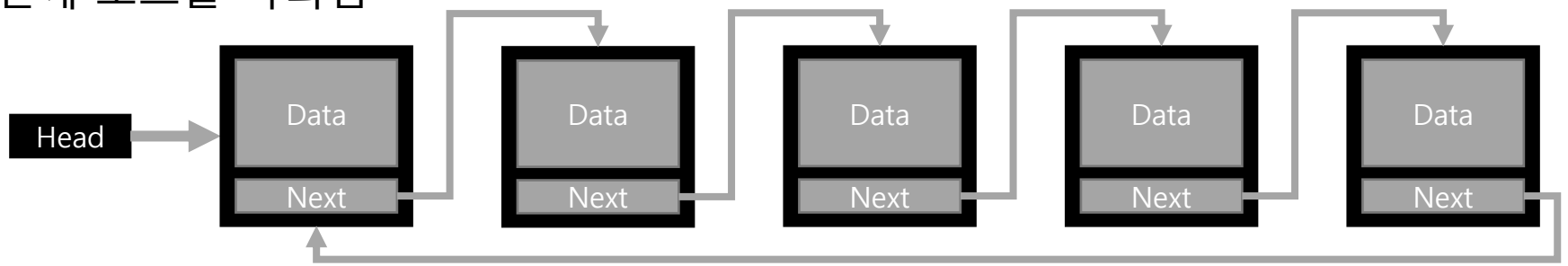


Linked List

- 이론 (2/2)
  - 종류
    - Singly
      - 이전 그림
    - Doubly
      - 이전 것도 가리킴



    - Circular
      - 마지막 노드가 첫번째 노드를 가리킴



Contents

- Array
  - 이론
  - 장/단점
- Linked List
  - 이론
  - 장/단점
- Implementation

Linked List

- 장/단점
  - 장점
    - 중간 값 삭제가 편하다!
    - 공간 낭비가 없다!
    - 크기는 언제든 변한다!

  - 단점
    - 어렵다!
    - index로 접근할 수가 없다

Contents

- Array
  - 이론
  - 장/단점
- Linked List
  - 이론
  - 장/단점
- Implementation

Implementation

- Structure

```
struct ListNode {
    int val;
    ListNode* next;
};
```

감사합니다!