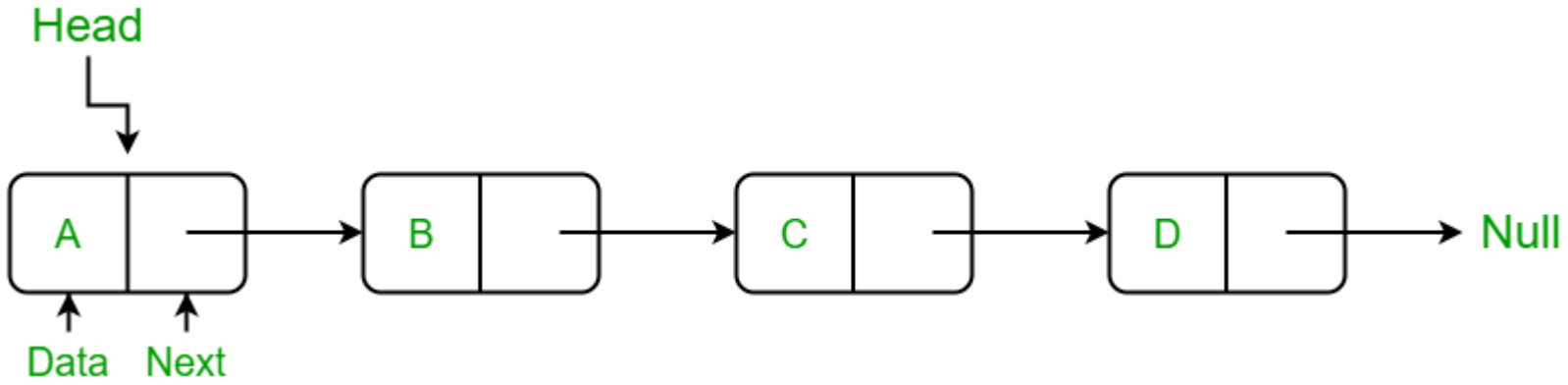
K-Digital Training 스마트 팩토리 3기

List

List ??

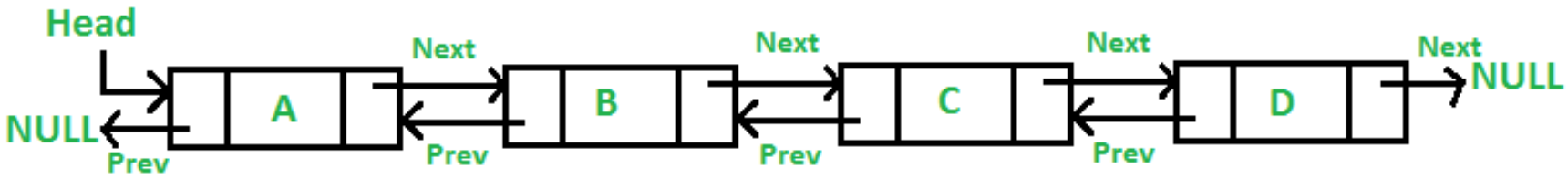
- 어떤 데이터를 저장할때 그 다음 순서의 자료가 있는 위치를 데이터에 포함시키는 방식으로 자료를 저장




```
class  Node {
public:
    int  data;
    // Pointer to next node
    Node*  next;
};
```

std::list

- Doubly linked list로 구현되어있음




```
class Node {
public:
    int data;
    // Pointer to next node
    Node* next;
    // Pointer to next node
    Node* prev;
};
```

std::list

- 멤버 함수에서 정렬(sort), 이어 붙이기(splice) 가능
- 임의 접근 반복자 at(), [] 불가능하고, 양방향 반복자 (++,--)이용해서 탐색
- push_front(), push_back(), pop_front(), pop_back() 이용해서 양 끝에서 삽입 삭제 가능
- Insert(), erase() 멤버 함수를 통해서 노드 중간에서도 삽입, 삭제 가능

list 사용하기

```
#include <list> // list 헤더파일을 추가해야 사용 가능



list<int> v = { 1,2,3,4,5 };
list <int> v(4); //int형 리스트 생성 후 크기를 4로 할당(모든 리스트요소 0으로 초기화)
list <int> v(5, 1); //int형 리스트 생성 후 크기를 5로 할당(모든 리스트요소 1로 초기화)

v.assign(5, 1);     //0~4인덱스의 값을 1로 초기화
```

list 반복문

```cpp
#include <iostream>
#include <list>

int main() {
    std::list<int> myList = {1, 2, 3, 4, 5};

    // for 루프를 사용하여 리스트의 요소를 출력
    std::cout << "리스트의 요소: ";
    for (const int& element : myList) {
        std::cout << element << " ";
    }
    std::cout << std::endl;

    for (std::list<int>::iterator it = myList.begin(); it != myList.end(); ++it) {
        std::cout << *it << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

List에만 존재하는 함수

- remove() – 특정 값 삭제
- remove_if(함수) – 특정 함수의 조건에 맞는 값 삭제
- splice() – 리스트간 요소 이동
- merge() – 정렬된 리스트 병합
- unique() – 중복 요소 제거
- sort() - 정렬

List에만 존재하는 함수

```cpp
#include <iostream>
#include <list>

int isOdd(int value) {
    return value % 2 == 0;
}

int main() {
    std::list<int> myList = {1, 2, 2, 3, 4, 4, 5};

    // 리스트에서 특정 값(예: 2)을 모두 제거
    myList.remove(2); // 1 3 4 4 5
    // 조건을 만족하는 요소(예: 짝수)를 모두 제거
    myList.remove_if(isOdd); //1 3 5

    return 0;
}
```

List에만 존재하는 함수

```cpp
#include <iostream>
#include <list>

int main() {
    std::list<int> myList1 = {1, 2, 3};
    std::list<int> myList2 = {4, 5};

    // myList1의 끝에 myList2의 모든 요소를 이동
    myList1.splice(myList1.end(), myList2);

    // myList1: 1 2 3 4 5
    // myList2:

    return 0;
}
```

List에만 존재하는 함수

```cpp
#include <iostream>
#include <list>

int main() {
    std::list<int> list1 = {1, 3, 5, 7};
    std::list<int> list2 = {2, 4, 6, 8};

    // 두 개의 정렬된 리스트를 병합
    list1.merge(list2);

    //병합된 리스트 (list1): 1 2 3 4 5 6 7 8
    //비어 있는 리스트 (list2):

    return 0;
}
```

List에만 존재하는 함수

```cpp
#include <iostream>
#include <list>

int main() {
    std::list<int> myList = {3, 1, 2, 2, 4, 3, 5, 4, 5};

    // 리스트 정렬
    myList.sort();
    //정렬된 리스트: 1 2 2 3 3 4 4 5 5

    // 중복 요소 제거
    myList.unique();
    //중복 제거된 리스트: 1 2 3 4 5

    return 0;
}
```

Array와 List의 비교

	Array	List
메모리	선언시 메모리 고정됨	메모리 재할당 용이
접근방식	Index기반 랜덤접근	Iterator 기반 접근
메모리 주소	연속적	불연속적
성능	빠름	느림
삽입, 삭제	느림	빠름