

자료구조실습

연결 리스트

- 연결 리스트
 - 단일 연결 리스트
 - 이중 연결 리스트
 - 원형 연결 리스트

- C++ 의 연결 리스트
 - list

연결 리스트

☐ 연결 리스트 (Linked List)

- 각 노드가 데이터와 포인터를 가지고 한 줄로 연결되어 있는 방식으로 데이터를 저장하는 자료구조.
- 데이터를 담고 있는 노드들이 순서를 유지하여 연결되어 있음.
 - ☐ 노드의 포인터가 이전 또는 다음의 노드와의 연결을 담당함.

☐ 연결 리스트의 종류

- 단일 연결 리스트 (Singly Linked List)
- 이중 연결 리스트 (Doubly Linked List)
- 원형 연결 리스트 (Circular Linked List)

☐ 장점

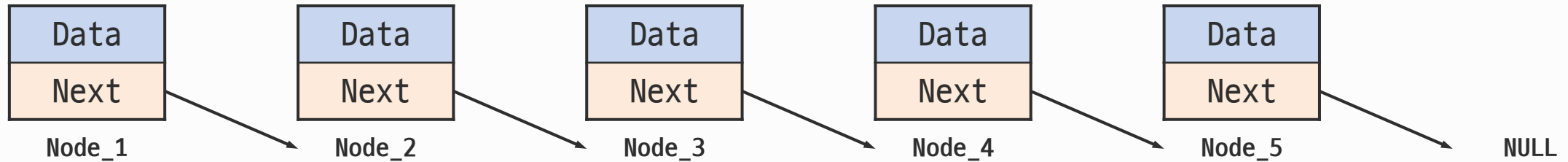
- 리스트의 중간 지점에서도 자료의 추가와 삭제하는 속도가 빠름.

☐ 단점

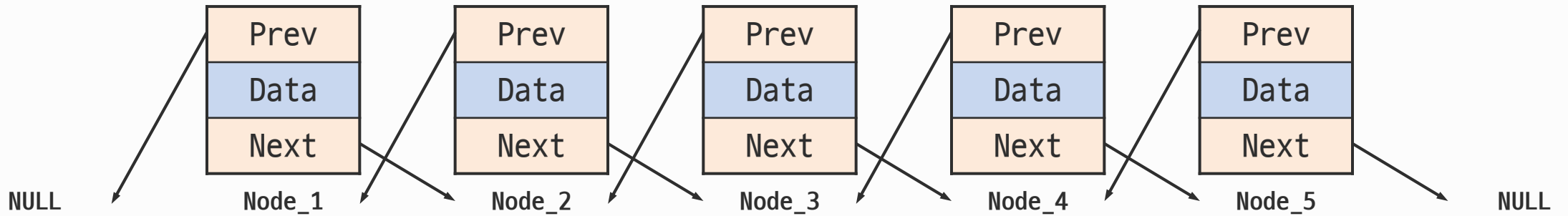
- 리스트의 특정 위치의 데이터를 검색하는 데에, 배열에 비해서 시간이 더 소요됨.

□ 단일 연결 리스트 (Singly Linked List)

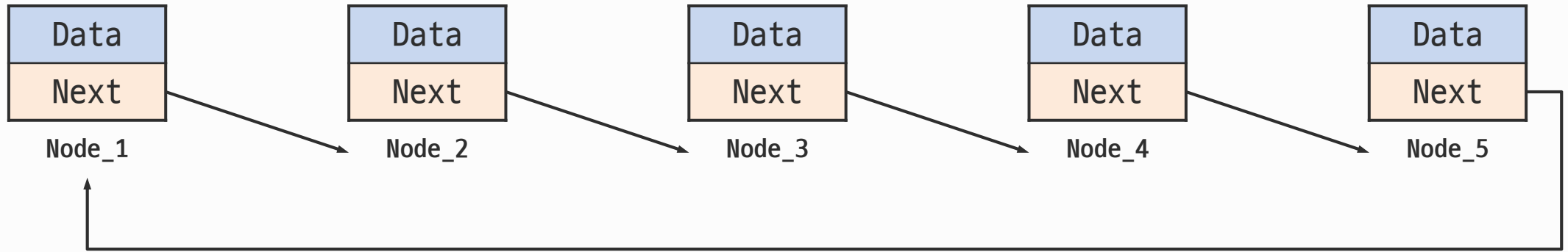
- 각 노드에 데이터와 한 개의 포인터가 있고, 각 노드의 포인터는 다음 노드를 가리키는 구조.



- 이중 연결 리스트 (Doubly Linked List)
 - 각 노드에 데이터와 두 개의 포인터가 있는 구조.
 - 한 개의 포인터는 이전 노드를 가리킴.
 - 다른 한 개의 포인터는 다음 노드를 가리킴.



- 원형 연결 리스트 (Circular Linked List)
 - 각 노드에 데이터와 한 개의 포인터가 있는 구조.
 - 마지막 노드의 포인터는 처음 노드를 가리킴.



C++ 의 연결 리스트

☐ 헤더 파일 및 템플릿

```
#include <list>
```

```
template <typename T, class Alloc = allocator<T>> class list;
```

☐ 설명

- C++ STL에 포함되어 있는 연결 리스트(이중 연결 리스트)를 표현하는 컨테이너.

☐ 인자

- T: 데이터의 자료형

☐ 선언 및 초기화 예시

- `std::list<int> l;` : 1차원 정수형 연결 리스트 선언
- `std::list<int> l(5);` : 기본 크기가 5인 연결 리스트 선언
- `std::list<int> l(5, 2);` : 기본 크기가 5이고, 모든 노드의 데이터를 2로 초기화
- `std::list<int> l2(l1);` : l1와 동일한 연결 리스트 선언
- `std::list<std::list<int>> l;` : 2차원 정수형 연결 리스트 선언

□ 멤버 함수 (Iterators)

■ `iterator begin() noexcept;`

- 리스트의 첫번째 노드를 가리키는 반복자를 반환함.

■ `iterator end() noexcept;`

- 리스트의 마지막 노드의 다음을 가리키는 반복자를 반환함.

■ `reverse_iterator rbegin() noexcept;`

- 리스트를 역으로 했을 때, 그 첫번째 노드를 가리키는 역방향 반복자를 반환함.

■ `reverse_iterator rend() noexcept;`

- 리스트를 역으로 했을 때, 그 마지막 노드의 다음을 가리키는 역방향 반복자를 반환함.

list 컨테이너 (cont`d)

☐ 멤버 함수 (Capacity)

■ `size_type size() const noexcept;`

☐ 리스트의 크기를 반환함.

■ `bool empty() noexcept const;`

☐ 리스트가 비어있음의 여부를 반환함.

- ☐ 멤버 함수 (Element access)
 - reference `front()`;
 - ☐ 리스트의 첫번째 노드를 반환함.
 - reference `back()`;
 - ☐ 리스트의 마지막 노드를 반환함.

□ 멤버 함수 (Modifiers)

■ `void assign(size_type n, const value_type& val);`

□ 리스트를 n개 노드와 val 값으로 새로 할당함.

■ `void push_front(const value_type& val);`

□ 리스트의 첫 노드 이전에 val을 추가함.

■ `void push_back(const value_type& val);`

□ 리스트의 마지막 노드 다음에 val을 추가함.

■ `void pop_front();`

□ 리스트의 첫 노드를 삭제함.

■ `void pop_back();`

□ 리스트의 마지막 노드를 삭제함.

■ `iterator insert(const_iterator position, const value_type& val);`

□ 리스트의 지정한 위치의 노드 이전에 val를 추가하고, 그 위치를 가리키는 반복자를 반환함.

■ `iterator erase(const_iterator position);`

□ 리스트의 지정한 위치의 노드를 삭제하고, 그 위치를 가리키는 반복자를 반환함.

□ 멤버 함수 (Modifiers) (cont`d)

■ `void clear() noexcept;`

□ 리스트의 모든 노드를 삭제함.

■ `void swap(list& l);`

□ 리스트의 모든 노드를 리스트 l과 교환함.

■ `void resize(size_type n);`

□ 리스트의 크기를 n으로 변경함.

■ 크기가 증가하는 경우에, 추가된 노드를 0으로 초기화 함.

■ 크기가 감소하는 경우에, 기존의 노드는 유지함.

□ 멤버 함수 (Operations)

■ `void splice(const_iterator position, list& l);`

□ 리스트 l의 모든 노드를 지정한 위치에 삽입함.

■ `void remove(const value_type& val);`

□ 리스트에서 val와 동일한 모든 노드를 삭제함.

■ `void unique();`

□ 리스트에서 인접하고 중복되는 노드를 하나를 제외하고 삭제함.

■ `void reverse();`

□ 리스트의 노드가 연결된 순서를 역으로 변경함.

■ `void sort();`

□ 리스트의 모든 노드를 오름차순으로 정렬함.

■ `void merge(list& l);`

□ 리스트에 리스트 l의 모든 노드를 merge sort를 수행하며 복사함.

list 컨테이너 예시

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

using namespace std;

int main(void) {
    double myDoubles[] = {12.15, 2.72, 73.0, 12.77, 3.14,
                          12.77, 73.35, 72.25, 15.3, 72.25};

    list<double> myList(myDoubles, myDoubles + 10);
    list<double>::iterator it;

    cout << ">> nodes of myList: ";
    for (it = myList.begin(); it != myList.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;

    cout << "<< myList.push_front(1.4);" << endl;
    myList.push_front(1.4);
```

```
    cout << "<< myList.push_back(1.4);" << endl;
    myList.push_back(1.4);

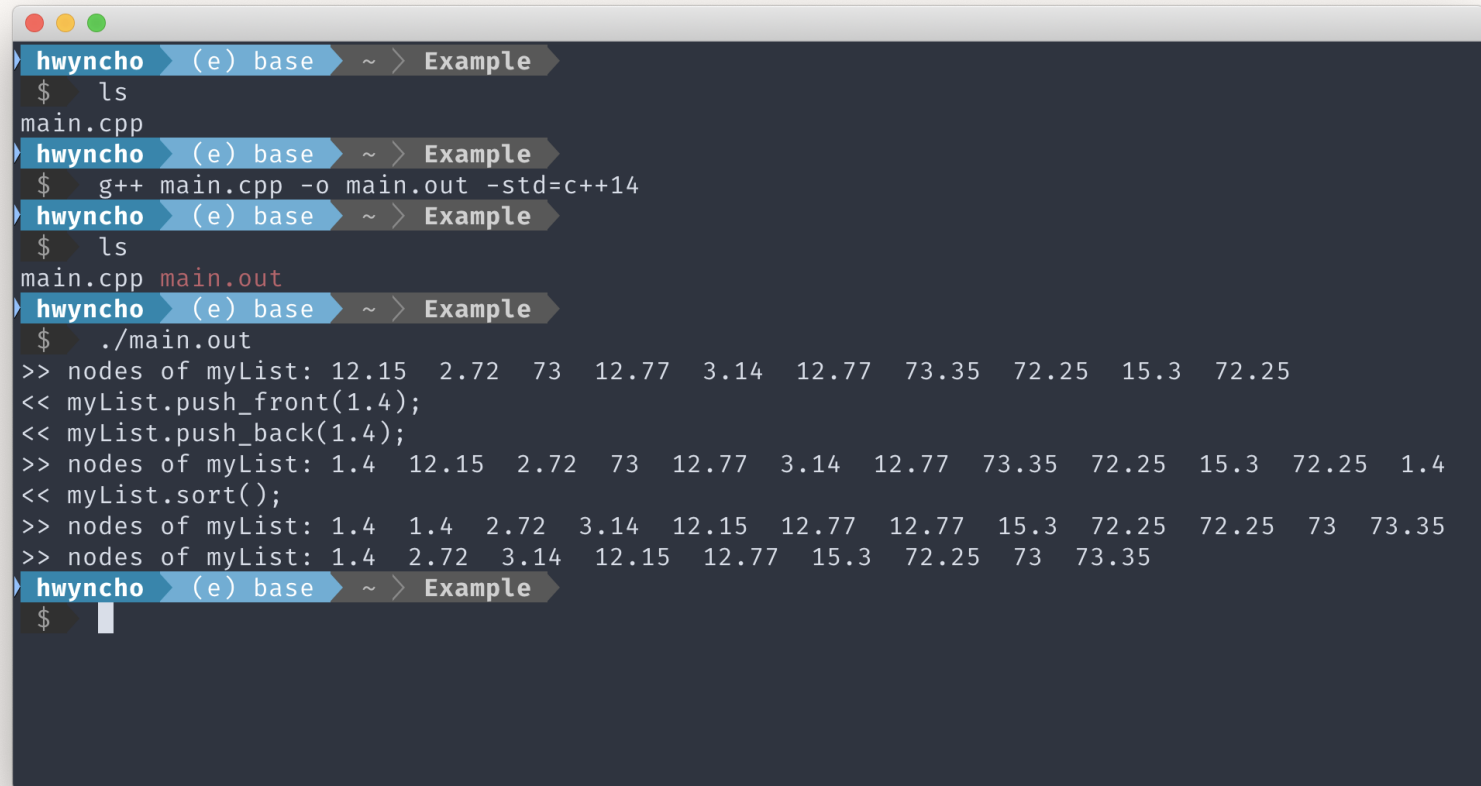
    cout << ">> nodes of myList: ";
    for (it = myList.begin(); it != myList.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;

    cout << "<< myList.sort();" << endl;
    myList.sort();

    cout << ">> nodes of myList: ";
    for (it = myList.begin(); it != myList.end(); ++it) {
        cout << *it << " ";
    }
    cout << endl;
```


list 컨테이너 예시 (cont`d)

```
myList.unique();  
cout << ">> nodes of myList: ";  
for (it = myList.begin(); it != myList.end(); ++it) {  
    cout << *it << " ";  
}  
cout << endl;  
  
return 0;  
}
```



```
hwyncho (e) base ~ > Example  
$ ls  
main.cpp  
hwyncho (e) base ~ > Example  
$ g++ main.cpp -o main.out -std=c++14  
hwyncho (e) base ~ > Example  
$ ls  
main.cpp main.out  
hwyncho (e) base ~ > Example  
$ ./main.out  
>> nodes of myList: 12.15 2.72 73 12.77 3.14 12.77 73.35 72.25 15.3 72.25  
<< myList.push_front(1.4);  
<< myList.push_back(1.4);  
>> nodes of myList: 1.4 12.15 2.72 73 12.77 3.14 12.77 73.35 72.25 15.3 72.25 1.4  
<< myList.sort();  
>> nodes of myList: 1.4 1.4 2.72 3.14 12.15 12.77 12.77 15.3 72.25 72.25 73 73.35  
>> nodes of myList: 1.4 2.72 3.14 12.15 12.77 15.3 72.25 73 73.35  
hwyncho (e) base ~ > Example  
$
```

list 컨테이너와 파일 접근 프리미티브 응용 - MyStudent.hpp

```
#ifndef __MYSTUDENT_H__
#define __MYSTUDENT_H__

#include <string>

#define MAX_NAME_LEN 32

class Student {
public:
    Student();
    Student(int id, std::string name, double score);

    void setId(int id);
    void setName(std::string name);
    void setScore(double score);

    int getId(void);
    std::string getName(void);
    double getScore(void);
```

```
private:
    int id;
    char name[MAX_NAME_LEN + 1];
    double score;
};

#endif
```

list 컨테이너와 파일 접근 프리미티브 응용 - MyStudent.cpp

```
#include "MyStudent.hpp"
#include <string.h>
#include <string>
```

```
Student::Student() {
    this->id = -1;
    memset(this->name, 0x00, MAX_NAME_LEN + 1);
    this->score = -1.0;
}
```

```
Student::Student(int id, std::string name, double score) {
    this->id = id;
    memcpy(this->name, name.c_str(), MAX_NAME_LEN);
    this->score = score;
}
```

```
void Student::setId(int id) { this->id = id; }
```

```
void Student::setName(std::string name) {
    memcpy(this->name, name.c_str(), MAX_NAME_LEN);
}
```

```
void Student::setScore(double score) { this->score = score; }
```

```
int Student::getId(void) { return this->id; }
```

```
std::string Student::getName(void) { return std::string(this->name); }
```

```
double Student::getScore(void) { return this->score; }
```

list 컨테이너와 파일 접근 프리미티브 응용 - main.cpp

```
#include "MyStudent.hpp"
#include <fcntl.h>
#include <iostream>
#include <list>
#include <string>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

using namespace std;

int main(void) {
    // 학생들의 정보를 입력받고, 연결 리스트에 저장
    list<Student> stuList;
    while (1) {
        string input;

        cout << "<< ID (input \'q\' to terminate): ";
        cin >> input;
```

```
        if (input.compare("q") == 0) {
            cout << ">> Terminate input." << endl;
            break;
        }
        int id = stoi(input);

        string name = "";
        cout << "<< Name: ";
        cin >> name;

        double score = -1.0;
        cout << "<< Score: ";
        cin >> score;

        Student stu(id, name, score);

        stuList.push_back(stu);
        cout << ">> Successfully added to list!" << endl;
    }
```

list 컨테이너와 파일 접근 프리미티브 응용 - main.cpp

```
// 입력된 학생들의 정보를 저장할 파일 열기
string filepath = "./StudentList.dat";
int fd = open(filepath.c_str(), O_CREAT | O_APPEND | O_WRONLY, 0644);
if (fd == -1) {
    perror("open() error");
    return 1;
}

// 학생들의 정보를 파일에 순차적으로 저장하기
list<Student>::iterator iter;
for (iter = stuList.begin(); iter != stuList.end(); ++iter) {
    if (write(fd, &(*iter), sizeof(Student)) == -1) {
        perror("write() error");
        return 2;
    }
}
close(fd);
```

```
cout << ">> " << stuList.size()
      << " students' info was successfully saved to the "
      << filepath << endl;

return 0;
}
```

list 컨테이너와 파일 접근 프리미티브 응용

```
bash
> hwncho (e) base ~ > Example
$ ls
MyStudent.cpp MyStudent.hpp main.cpp
> hwncho (e) base ~ > Example
$ g++ main.cpp MyStudent.cpp -o main.out -std=c++14
> hwncho (e) base ~ > Example
$ ./main.out
<< ID (input 'q' to terminate): 2017726001
<< Name: Gildong
<< Score: 100
>> Successfully added to list!
<< ID (input 'q' to terminate): 2018203001
<< Name: Hong
<< Score: 99.5
>> Successfully added to list!
<< ID (input 'q' to terminate): q
>> Terminate input.
>> 2 students' info was successfully saved to the ./StudentList.dat
> hwncho (e) base ~ > Example
$ ls
MyStudent.cpp MyStudent.hpp StudentList.dat main.cpp      main.out
> hwncho (e) base ~ > Example
$
```