# 자료구조 실습

**09/17**

김민서 (imkmsh@khu.ac.kr)

# 목차

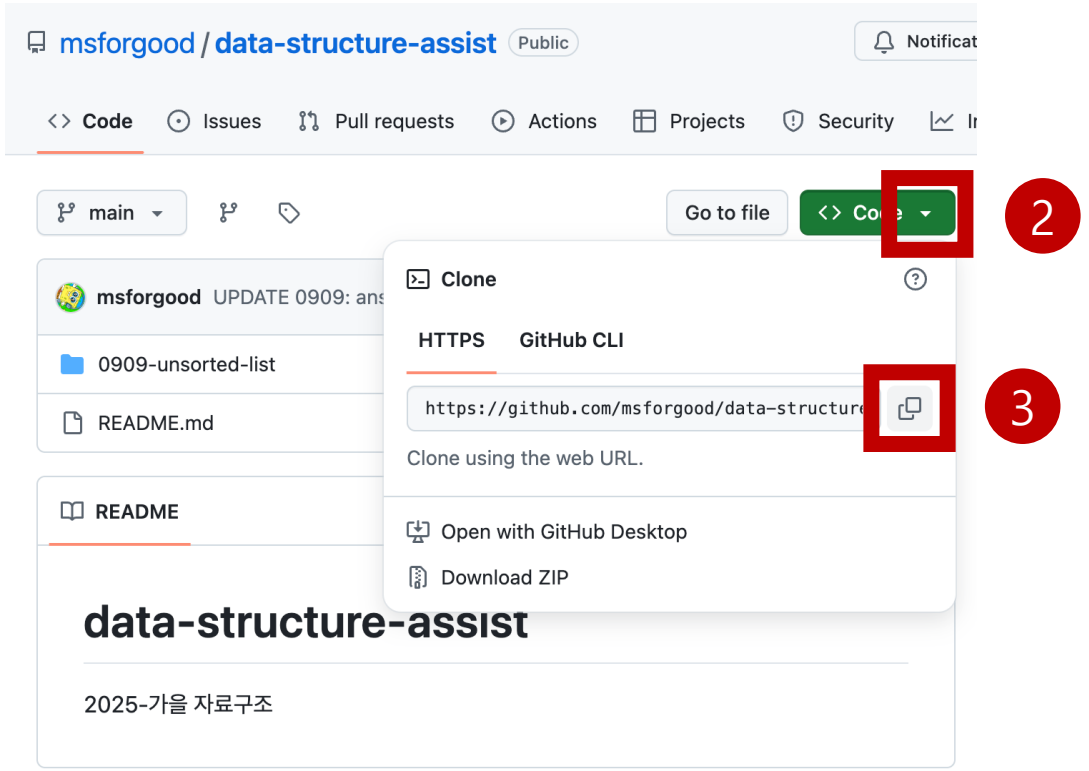
- 실습 코드 다운로드 및 실행 방법
- Sorted List
- Programming Tips

김민서 (imkmsh@khu.ac.kr)

# 실습 코드 다운로드 및 실행 방법

1

- 실습 코드 폴더 생성 원하는 곳에서
  $ git clone https://github.com/msforgood/data-structure-assist.git




김민서 (imkmsh@khu.ac.kr)

# 실습 코드 다운로드 및 실행 방법

- 저번 시간에 clone 했다면 main 브랜치 pull 만으로도 가능

  해당 폴더 터미널에서

  $ git pull origin main

# 실습 코드 다운로드 및 실행 방법

- 정답 코드는 **answer.py** 에 작성
- 코드를 잘 작성했는지 확인하려면 **main.py** 실행
  $ python {자신의 폴더 실행 경로}/main.py

- 추후 오피셜 정답 코드는 **answer_official.py** 로 push 될 예정

  main 브랜치를 pull 받은 뒤 ($ git pull origin main)
  **main.py** 내에서 < from ~~answer~~ answer_official import {…} >
  import 하는 클래스 모듈 위치를
  answer, answer_official 지정함에 따라 오피셜 코드 동작 확인 가능

김민서 (imkmsh@khu.ac.kr)

# Sorted List Class 정의

```
class SortedList:
    def __init__(self, max_size=100):
        self.data = [None] * max_size
        self.length = 0
        self.max_size = max_size
```

```
    # 보조 메서드
    def size(self):
        return self.length

    def isFull(self):
        return self.length == self.max_size

    def isEmpty(self):
        return self.length == 0

    def getItem(self, pos):
        if pos < 0 or pos >= self.length:
        raise IndexError("pos out of range")
        return self.data[pos]

    def clear(self):
        self.length = 0
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - insert #1

```
# 핵심 1: 삽입 (정렬 유지)
# 1) 들어갈 위치 탐색 (선형)
# 2) 뒤에서부터 한 칸씩 밀기 (shift)
# 3) 값 넣기 + length 증가
def insertItem(self, value):
     if self.isFull():
          raise OverflowError("List is full")

     # 1) 위치 찾기: 첫 번째로 value <= data[i]가 되는 위치 i
     location = 0
     while NEED_TO_SOLVE:
          if NEED_TO_SOLVE:
                    location += 1
          else:
                    break
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - insert #1

```
# 핵심 1: 삽입 (정렬 유지)
# 1) 들어갈 위치 탐색 (선형)
# 2) 뒤에서부터 한 칸씩 밀기 (shift)
# 3) 값 넣기 + length 증가
def insertItem(self, value):
    if self.isFull():
        raise OverflowError("List is full")

    # 1) 위치 찾기: 첫 번째로 value <= data[i]가
    location = 0
    while NEED_TO_SOLVE:
        if NEED_TO_SOLVE:
            location += 1
        else:
            break
```

```
# 핵심 1: 삽입 (정렬 유지)
# 1) 들어갈 위치 탐색 (선형)
# 2) 뒤에서부터 한 칸씩 밀기 (shift)
# 3) 값 넣기 + length 증가
def insertItem(self, value):
    if self.isFull():
        raise OverflowError("List is full")

    # 1) 위치 찾기: 첫 번째로 value <= data[i]가 되는 위치 i
    location = 0
    while location < self.size():
        if self.data[location] < value:
            location += 1
        else:
            break
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - insert #2

```
    # 2) 뒤에서부터 오른쪽으로 한 칸씩 밀기
    # [location..length-1] 구간을 한 칸씩 뒤로
    i = self.size()
    while NEED_TO_SOLVE:
        NEED_TO_SOLVE
        i -= 1

    # 3) 삽입
    NEED_TO_SOLVE = value
    self.length += 1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - insert #2

```
    # 2) 뒤에서부터 오른쪽으로 한 칸씩 밀기
    # [location..length-1] 구간을 한 칸씩 뒤로
    i = self.size()
    while NEED_TO_SOLVE:
        NEED_TO_SOLVE
        i -= 1

    # 3) 삽입
    NEED_TO_SOLVE = value
    self.length += 1
```

```
# 2) 뒤에서부터 오른쪽으로 한 칸씩 밀기
#    [location..length-1] 구간을 한 칸씩 뒤로
i = self.size()
while i > location:
    self.data[i] = self.data[i - 1]
    i -= 1

# 3) 삽입
self.data[location] = value
self.length += 1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - insert 확인

- $ python {자신의 폴더 실행 경로}/main.py

```
(base) minseo@minseo-MacBookAir-2 data-structure-assist % python 0917-1-sorted-list/main.py
삽입 ================
3 삽입 시도
결과 : [3]

5 삽입 시도
결과 : [3, 5]

1 삽입 시도
결과 : [1, 3, 5]

4 삽입 시도
결과 : [1, 3, 4, 5]

2 삽입 시도
결과 : [1, 2, 3, 4, 5]

====================
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #1

```
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #1

```python
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

```python
def findItem(self, item):
    first, last = 0, self.size() - 1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #2

```python
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #2

```
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

```
def findItem(self, item):
    first, last = 0, self.size() - 1
    while first <= last:
        mid = (first + last) // 2
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #3

```
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find #3

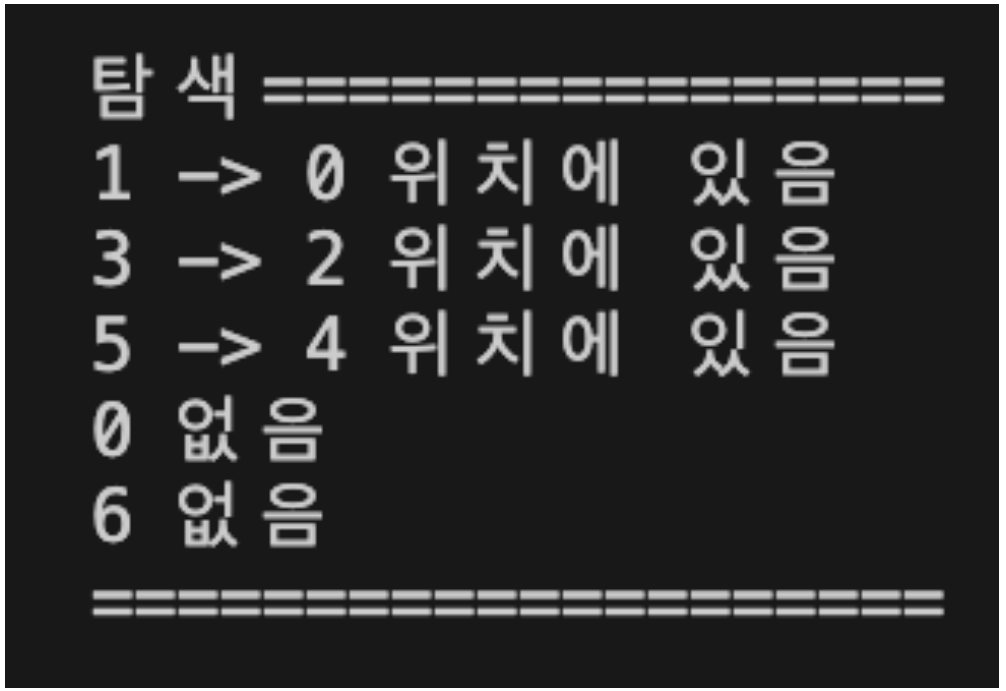
```python
# 핵심 2: 이진 탐색 (O(log N))
# 찾으면 인덱스, 없으면 -1
def findItem(self, item):
    first, last = NEED_TO_SOLVE, NEED_TO_SOLVE
    while NEED_TO_SOLVE:
        mid = NEED_TO_SOLVE

        if NEED_TO_SOLVE:
            NEED_TO_SOLVE
        elif NEED_TO_SOLVE:
            NEED_TO_SOLVE
        else:
            return mid

    return -1
```

```python
def findItem(self, item):
    first, last = 0, self.size() - 1
    while first <= last:
        mid = (first + last) // 2

        if item < self.data[mid]:
            last = mid - 1
        elif item > self.data[mid]:
            first = mid + 1
        else:
            return mid

    return -1
```

김민서 (imkmsh@khu.ac.kr)

# Sorted List - find 확인

- $ python {자신의 폴더 실행 경로}/main.py



# Sorted List - remove (참고용)

```python
# 핵심 3: 삭제 (검색 후 shift로 당기기)
def removeItem(self, value):
    if self.isEmpty():
        return False

    idx = self.findItem(value)
    if idx == -1:
        return False

    # idx 이후를 한 칸씩 왼쪽으로 당김
    i = idx
    while i < self.size() - 1:
        self.data[i] = self.data[i + 1]
        i += 1

    # 꼬리 정리
    self.data[self.size() - 1] = None
    self.length -= 1
    return True
```

```
삭제 ================
1 삭제 시도
결과 : [2, 3, 4, 5]

2 삭제 시도
결과 : [3, 4, 5]

3 삭제 시도
결과 : [4, 5]

4 삭제 시도
결과 : [5]

5 삭제 시도
결과 : []

====================
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips 실행 방법

GNU C++ 컴파일러 컴파일할 소스 파일 목록 output 실행 파일 이름을 main으로

- $ g++ main.cpp answer_official.cpp -o main # 정답 버전 컴파일
- $ g++ main.cpp answer.cpp -o main # 실습 버전 컴파일
- $ ./main # 실행

- g++이 없다면?
  - Linux: $ sudo apt install build-essential
  - macOS: 보통 clang++이 심볼릭 링크 되어 있음
    하지만 안된다면 $ xcode-select -install
  - Windows: vsCode로 간편 설치 권장

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - 포인터 기초

## 0. 포인터 기초

- `&x` : 변수 `x`의 주소 (메모리 위치)
- `int *p` : `int` 타입 주소를 담는 포인터 변수
- `*p` : 포인터 `p`가 가리키는 주소의 실제 값 (역참조)

Tip: 원본 값을 직접 바꾸고 싶으면, 주소를 넘겨서 `*p`를 조작해야 합니다.

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap1

## 1. swap_value(int a, int b)

(Call by Value – 값 전달)

- 정의: 인자로 값이 복사되어 들어옴 → 원본 `x, y` 는 안 바뀜
- 역할: Call by Value의 동작 확인
- input: 정수 두 개 (복사본)
- 주의: 원본 바꾸려면 참조/포인터 필요

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap1

```cpp
// 1) 값 전달
void swap_value(int a, int b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_value 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap1

```cpp
// 1) 값 전달
void swap_value(int a, int b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_value 내부] a=" << a << ", b=" << b << endl;
}
```

```cpp
// 1) 값 전달
void swap_value(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    cout << "[swap_value 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap2

## 2. swap_ref(int &a, int &b)

(Call by Reference – 참조 전달, C++ 전용)

- 정의: a, b는 원본 x, y와 같은 메모리 → 바꾸면 원본도 바뀜
- 역할: Call by Reference 예제
- input: 참조 두 개 (원본 변수)
- 주의: C에서는 불가능. 포인터로 같은 효과를 낼 수 있음.

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap2

```
// 2) 참조 전달
void swap_ref(int &a, int &b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_ref 내부] a=" << a << ", b=" << b << endl;
}
```

# Programming Tips - swap2

```
// 2) 참조 전달
void swap_ref(int &a, int &b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_ref 내부] a=" << a << ", b=" << b << endl;
}
```

```
// 2) 참조 전달
void swap_ref(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
    cout << "[swap_ref 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap3

## 3. swap_pointer_deref(int *a, int *b)

(Call by Address – 주소 전달 후 역참조)

- 정의: a, b는 주소를 가짐. `*a`, `*b`로 원본 값 수정 가능
- 역할: 포인터 역참조로 Call by Address 구현
- input: int* 두 개 (`&x`, `&y` 형태로 호출)
- 주의: 널 포인터는 역참조하면 크래시

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap3

```
void swap_pointer_deref(int *a, int *b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_pointer_deref 내부] a=" << a << ", b=" << b << endl;
}
```

# Programming Tips - swap3

```
void swap_pointer_deref(int *a, int *b) {
    int temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_pointer_deref 내부] a=" << a << ", b=" << b << endl;
}
```

```
void swap_pointer_deref(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
    cout << "[swap_pointer_deref 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap4

## 4. swap_pointer_var(int *a, int *b)

(포인터 변수 자체만 교환)

- 정의: a, b는 “포인터 변수 복사본”. 둘만 바꾸면 원본 값(x,y)은 안 바뀜
- 역할: 포인터 스왑 예시 (pdf의 swap4)
- input: int* 두 개
- 주의: 원본 값을 바꾸려면 3번처럼 `*a`, `*b` 사용해야 함

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap4

```
void swap_pointer_var(int *a, int *b) {
    int *temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_pointer_var 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - swap4

```
void swap_pointer_var(int *a, int *b) {
    int *temp = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    NEED_TO_SOLVE = NEED_TO_SOLVE;
    cout << "[swap_pointer_var 내부] a=" << a << ", b=" << b << endl;
}
```

```
void swap_pointer_var(int *a, int *b) {
    int *temp = a;
    a = b;
    b = temp;
    cout << "[swap_pointer_var 내부] a=" << a << ", b=" << b << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - increment

## 5. increment_pointer(int *ptr)

(포인터로 값 1 증가)

- 정의: ptr이 가리키는 정수 값을 1 증가
- 역할: 포인터 역참조로 원본 값 수정
- input: int* (예: &x)
- 주의:
  - `(*ptr)++` : 가리키는 값 증가 (원하는 동작)
  - `*ptr++` : 포인터 자체가 다음 주소로 이동 (원치 않는 동작)

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - increment

```
void increment_pointer(int *ptr) {
    NEED_TO_SOLVE;
    cout << "[increment_pointer 내부] *ptr=" << NEED_TO_SOLVE << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# Programming Tips - increment

```
void increment_pointer(int *ptr) {
    NEED_TO_SOLVE;
    cout << "[increment_pointer 내부] *ptr=" << NEED_TO_SOLVE << endl;
}
```

```
void increment_pointer(int *ptr) {
    (*ptr)++;
    cout << "[increment_pointer 내부] *ptr=" << *ptr << endl;
}
```

김민서 (imkmsh@khu.ac.kr)

# 수고하셨습니다

**자료구조 실습 09/17**

**EOF**

김민서 (imkmsh@khu.ac.kr)