Lab09.
Heap

BeomSeok Kim
Department of Computer Engineering
KyungHEe University
passion0822@khu.ac.kr

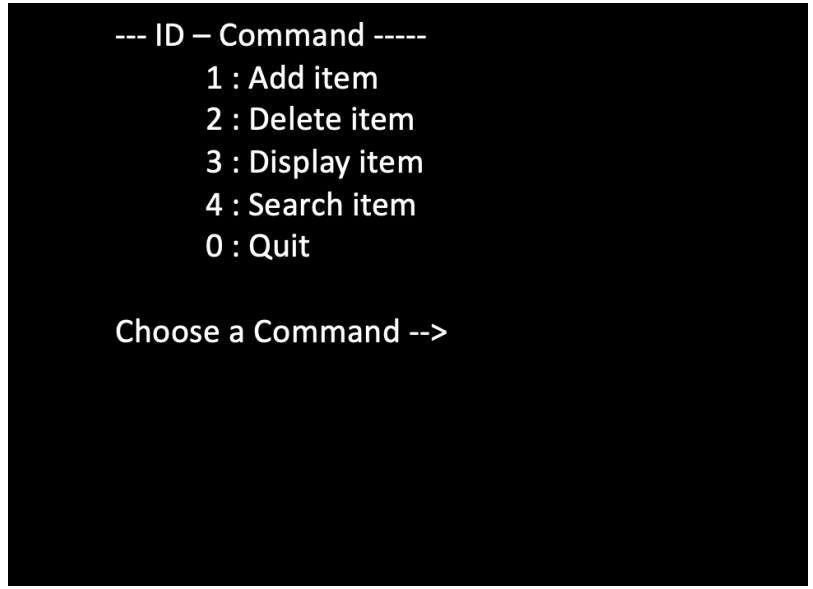
Intro.

- 목표
  - ✓ Array를 이용한 Heap 구현
  - ✓ 재귀(Recursive) 함수, 상속(Inheritance), Heap 이해
  - ✓ Heap을 이용한 운선순위 큐(Priority Queue) 구현

- 내용
  - ✓ Array 기반의 Min/Max Heap 구현
    - ➢ 재귀함수를 사용하는 Heap을 구현
  - ✓ ItemType은 list에서 사용하던 ItemType을 사용
    - ➢ ID를 기준으로 정렬하는 Heap을 구성

- 방법
  - ✓ Heap의 메커니즘을 분석하고 Max/Min Heap의 ADT를 바탕으로 구현
    - ➢ 특정 자료형과 무관하게(Generic) 정의
      - – ItemType에서는 비교연산자 (>, >=, <, <=, ==, !=)와 출력연산자 재정의 사용
    - ➢ Application에서는 MaxHeap 적용
    - ➢ 삭제할 때 해당 노드의 레벨을 고려하여 Heap의 형태를 유지할 수 있도록 구현
    - ➢ Root 노드가 삭제되는 priority queue의 기능을 구현

Example: Run() function



HeapBase Class ADT(1/2)

```
template <typename T>
class HeapBase
{
    public:
    HeapBase();                          // default constructor
    virtual ~HeapBase();                 // destructor

    // check the heap state whether empty or full.
    bool IsEmpty();                      // check heap is empty
    bool IsFull();                       // check heap is full

    int GetLength() const;               // get number of current node
    void MakeEmpty();                    // make empty
    virtual int Add(T item);             // add new node
    virtual int Delete(T item);          // delete node
    virtual T Pop();                     // pop a root node
```

HeapBase Class ADT(2/2)

```
    virtual void RetrieveItem(T &item, bool &found);        // retrieve item
    virtual void PrintHeap();                               // print nodes in Heap

    // pure virtual functions for recursive process.
    virtual void ReheapDown(int iparent, int ibottom) = 0;
    virtual void ReheapUp(int iroot, int ibottom) = 0;
    virtual void Delete(T item, bool &found, int iparent) = 0;
    virtual void Retrieve(T &item, bool &found, int iparent) = 0;

    protected:
    T *m_pHeap;                 // element array.
    int m_iLastNode;            // number of nodes in heap.
    int m_nMaxSize;             // maximum array size.
};
```

MaxHeap class ADT

```
template <typename T>
class MaxHeap : public HeapBase<T>
{
      public:
      MaxHeap();                    // default constructor
      MaxHeap(int size);            // constructor

      // recursive functions.
      virtual void ReheapDown(int iparent, int ibottom);          // reheap down
      virtual void ReheapUp(int iroot, int ibottom);              // reheap up
      virtual void Delete(T item, bool &found, int iparent);      // delete node
      virtual void Retrieve(T &item, bool& found, int iparent);   // search node
};
```

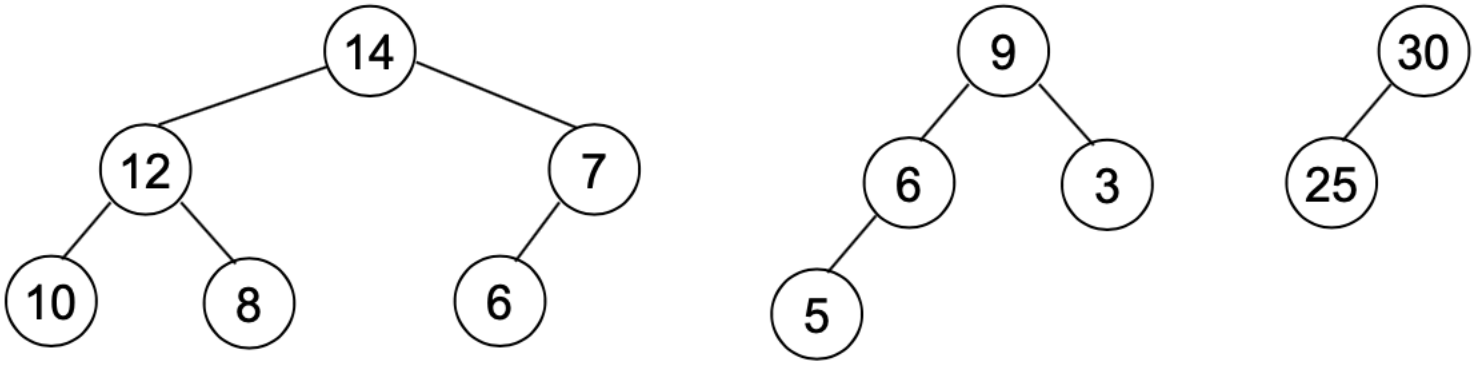
MinHeap class ADT

```
template <typename T>
class MinHeap : public HeapBase<T>
{
      public:
      MinHeap();                    // default constructor
      MinHeap(int size);            // constructor

      // recursive functions.
      virtual void ReheapDown(int iparent, int ibottom);          // reheap down
      virtual void ReheapUp(int iroot, int ibottom);              // reheap up
      virtual void Delete(T item, bool &found, int iparent);      // delete node
      virtual void Retrieve(T &item, bool& found, int iparent);   // search node
};
```

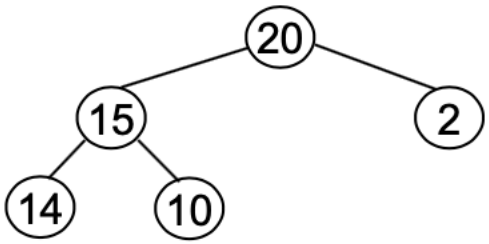
Max Heap

- Max tree는 각 노드의 키 값이 그 자식의 키 값보다 작지 않은 tree
- Max heap은 max tree이면서 complete binary tree

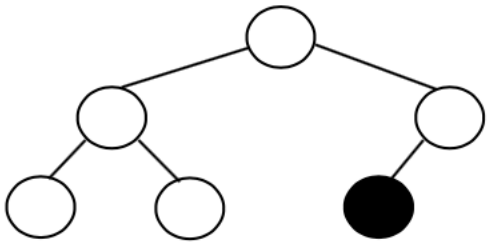


MaxHeap insertion

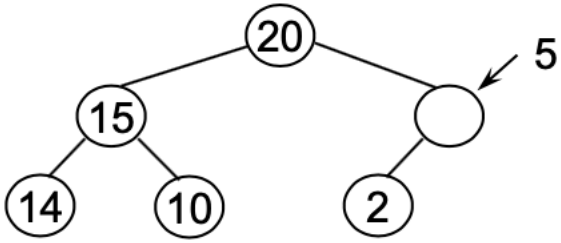
- 5, 12을 삽입할 때의 예



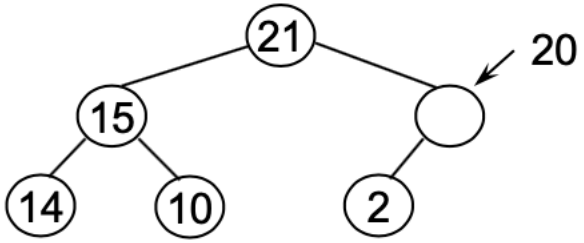
(a) 삽입 전 Heap
Before insert



(b) 새 노드의 초기 위치
Position of new node



(c) Heap (a)에 5를 삽입
insert 5-node into (a) heap



(d) Heap (a)에 21을 삽입
insert 21-node into (a) heap

Homework 5

- 목표
  - ✓ Doubly linked list와 iterator class의 구현 및 이를 이용한 사진 폴더 관리 시스템 프레임 생성
- 내용
  - ✓ List header와 trailer를 이용한 doubly linked list로 정의
  - ✓ Iterator class의 정의 및 이를 이용한 멤버함수 구현
  - ✓ 이전 과제에서 구현한 singly linked list를 template doubly linked list로 정의
  - ✓ 계층적 폴더 시스템을 구현 (이전과 동일

- 방법
  - ✓ Header와 trailer를 이용한 doubly linked list class 정의 및 멤버함수 구현
  - ✓ Iterator class 구현 및 이를 이용한 멤버 함수 구현
    - ➢ ResetList()와 GetNextItem() 함수를 이용하여 구현한 기존 멤버 함수나 application class 멤버 함수를 iterator class를 이용하여 구현
  - ✓ Doubly linked list를 template로 구현
    - ➢ 강의자료와 참조자료에 제공된 코드를 분석하여 자신만의 doubly linked list를 작성.
    - ➢ Template로 정의
  - ✓ Application class에 다음과 같은 기능을 doubly linked list에 맞게 추가 또는 수정한다.
    - ➢ 폴더 Navigation을 위한 기능 : 최 상위 폴더인 album폴더에서부터 시작하여 폴더를 오르내리면서 상하폴더를 자연스럽게 열어서 내용을 화면에 출력하는 함수를 작성
    - ➢ 범위 탐색 기능 추가 : 특정 이름의 폴더 및 사진을 현재 폴더에서만 탐색할지 album 전체에서 탐색할지 구분하여 탐색하는 기능
  - ✓ 생성된 기능들을 테스트하기 위한 Run()함수 수정

Thank You!
Q&A