CS162: Introduction to Computer Science II

Week 4: Other Types Of Linked List

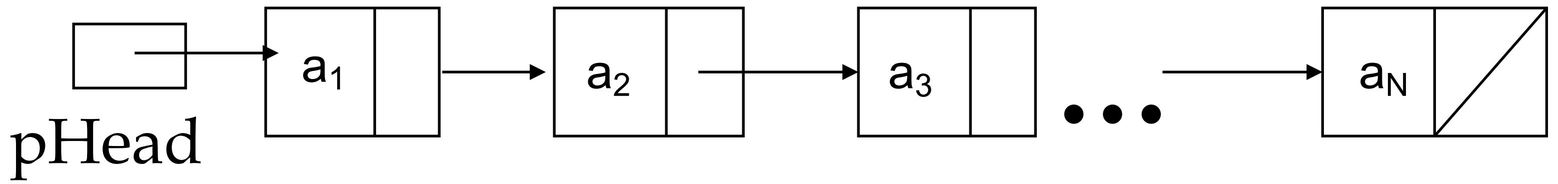
02/2024

CS162 – What is next?

- Ordered Linked List
- Doubly Linked List
- Circular Linked List

Ordered Linked List

- Nodes in the list are arranged in order



in which:

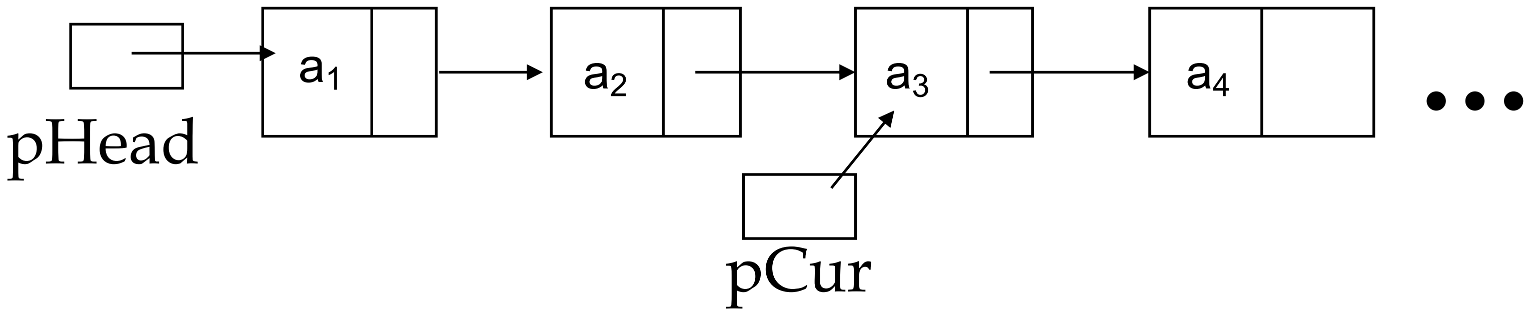
$$a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_N$$

Ordered Linked List

- Operations on an Ordered Link List
  - Create the ordered linked list
  - Find a node with value X
  - Insert a new node with value X
  - Remove a node with value X
  - Remove all nodes with value X
  - Merge the 2 ordered lists into one
  - …

Working on an Ordered Linked List

- When traversing on the ordered list to look for a value, e.g. X, since the values of the nodes are arranged in order, you do not need to traverse further if the current value is already bigger than X.
- For example: pCur stops at a_3 if it is already bigger than X



Example: Search for X in an ordered list

- With an ordered list, we can stop early when facing a node whose value is bigger than X

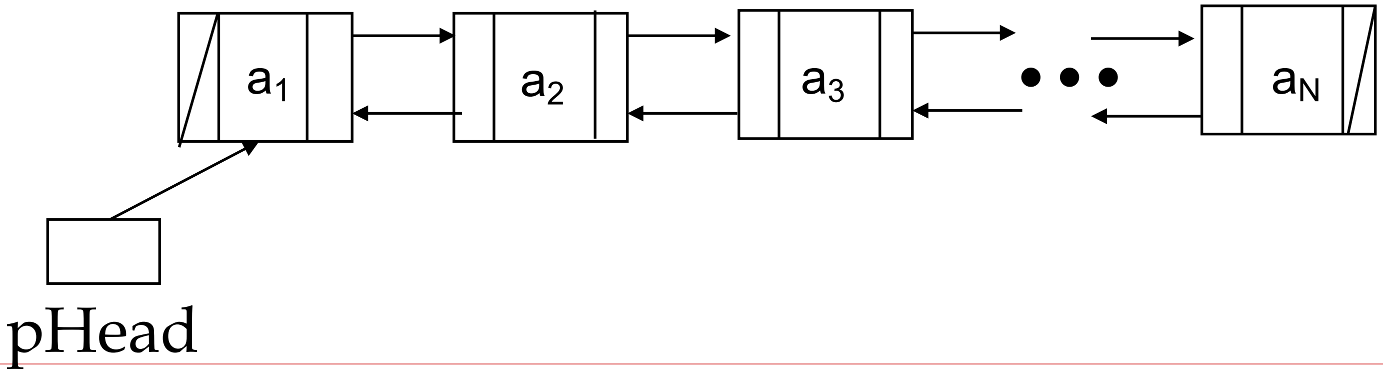
```
//Search for X, return the pointer to X
//or nullptr if not found
Node* search4X(Node* pHead, int x) {
    while (pHead && pHead->data<x)
        pHead = pHead->pNext;
    if (pHead && pHead->data==x) return pHead;
    else return nullptr;
}
```

Doubly Linked List

- Each Node has 2 links: **`pPrev`** and **`pNext`**

```
struct Node {
    int data;
    Node* pNext, *pPrev;
};
```

- The **`pPrev`** of the fist Node points to **`nullptr`**
- The **`pNext`** of the last Node points to **`nullptr`**



Doubly Linked List

- Some popular operations
  - Create the doubly linked list
  - Insert a new node at the beginning of the list
  - Insert a new node X after a node K
  - Insert a new node X before a node K
  - Remove a node with value X
  - Remove all nodes with value X
  - …

Example: remove all Node X's

```
void removeAllXs2(Node* &pHead, int x) {
    Node* cur = pHead;
    while (cur) {
        if (cur->data == x) {
            if (cur->pNext) cur->pNext->pPrev = cur->pPrev;
            if (cur->pPrev) cur->pPrev->pNext = cur->pNext;
            else pHead = cur->pNext;
            Node* tmp = cur;
            cur = cur->pNext;
            delete tmp;
        }
        else cur = cur->pNext;
    }
}
```

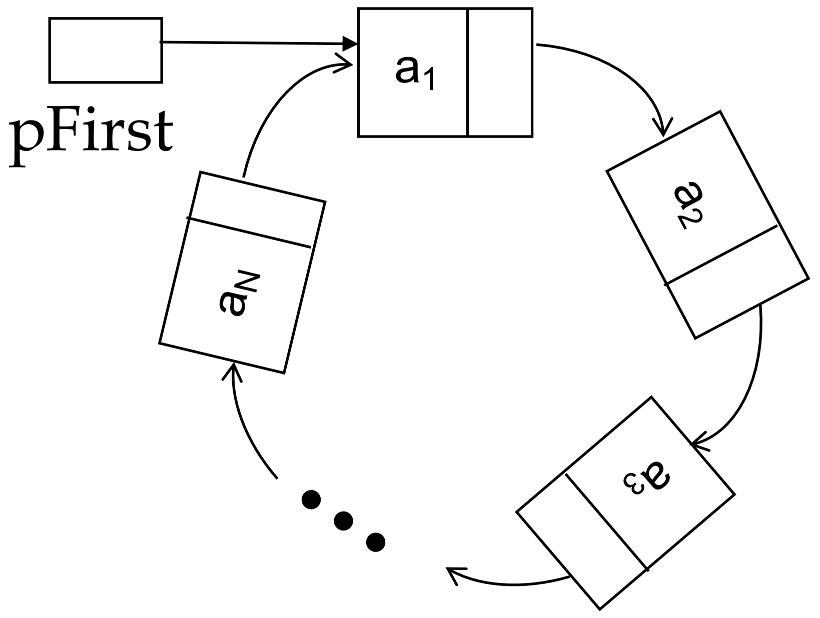
Make sure the Node is valid

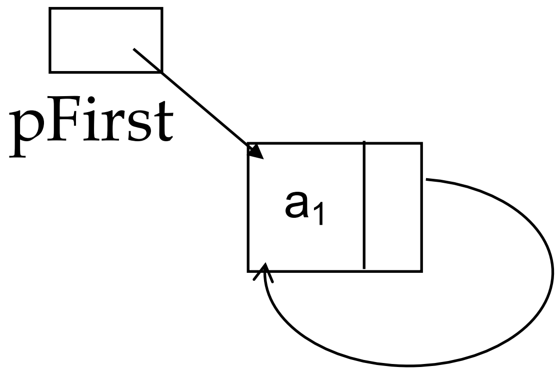
Thanks to the DOUBLE LINKS, just traversing without worrying

update the pHead when the old one is removed

Circular Linked List

- A singly linked list with the last node points to the first node





A circular linked list with only ONE node

Circular Linked List

- Some operations
  - Create the circular linked list
  - Find X from the list
  - Insert a new node X after a node K
  - Insert a new node X before a node K
  - Remove a node with value X
  - Remove all nodes with value X
  - …

Example: find X in a Circular Linked List

- Make sure to stop when going all for the whole list

```
Node* findX(Node* pFirst, int x) {
    if (!pFirst || pFirst->data==x) return pFirst;
    Node* pCur = pFirst->pNext;
    while (pCur!=pFirst && pCur->data!=x)
        pCur = pCur->pNext;
    if (pCur->data==x) return pCur;
    return nullptr;
}
```

Starting from the 2nd Node and it finishes a full circle when meeting **`pFirst`** again