LINKED LIST ADT

Linked List ADT

- A linked list is:
  1. a linear data structure
  2. a data structure where each node has a unique predecessor and a unique successor

- A data element can be inserted or removed anywhere in the list

Linked List ADT Specification

- **Elements**: List elements can be of any type, but we will assume ListElement

- **Structure**: Any mechanism for allowing the insertion, deletion, or modification of a ListElement anywhere in the list. Each ListElement has a unique predecessor and successor

Linked List ADT Continued

- **Domain**: The number of list elements is bounded. A list is considered full if the upper-bound is reached. A list with no elements is considered empty.

- Operations: There are 18 operations.

Linked List Operations

Allocation and Deallocation

1. lstCreate
2. lstDispose

Checking number of elements

1. lstSize
2. lstIsFull

Linked List Operations

- Peek Operations
  1. lstPeek
  2. lstPeekPrev
  3. lstPeekNext
- Retrieving values
  1. lstFirst
  2. lstLast
  3. lstNext
  4. lstPrev

List Operations

- Retrieving values
  5. lstFindKey
  6. lstDeleteCurrent
  7. lstInsertAfter
  8. lstInsertBefore
  9. lstUpdateCurrent
  10. lstHasNext
  11. lstHasPrev

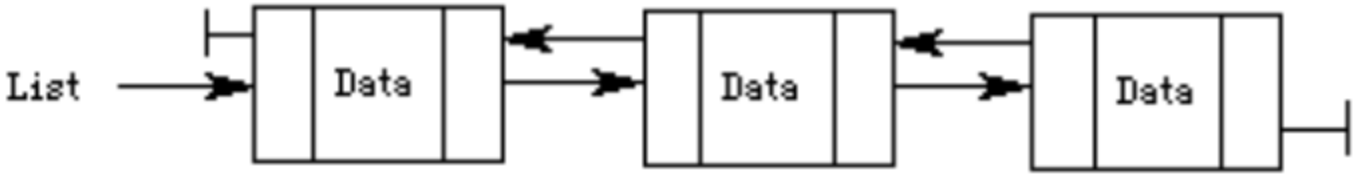
Linked Lists

Singly Linked List

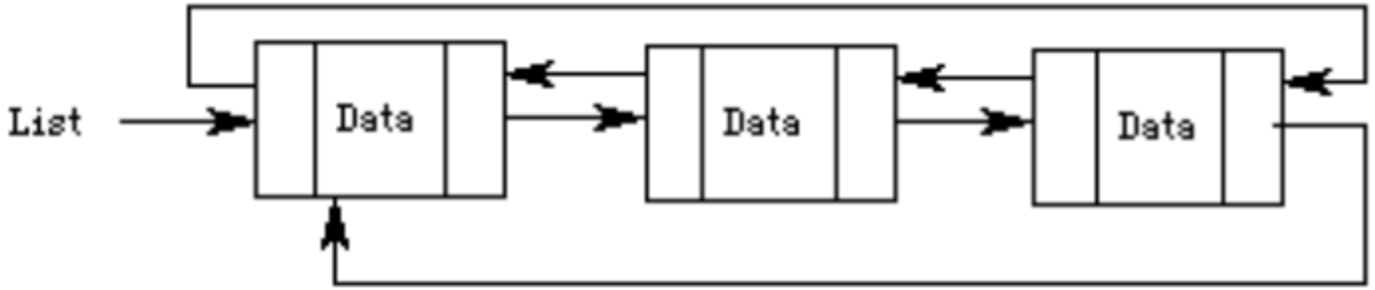
Singly Linked Circular List

Linked Lists

Doubly Linked List



Doubly Linked Circular List



Linked List Implementation

- How might we implement the previously specified Linked List ADT?
- There are any number of ways but let's begin with the following:

```
typedef char ListElementType;

typedef struct ListElement *ListElementPtr;
typedef struct ListElement
{
  ListElementType data;
  ListElementPtr psNext;
} ListElement;

int main(void)
{
  ListElementPtr psHead;
```

Problem

- Using the linked list implementation from the previous slide, write each of the following:

1. lstCreate (this is very hard to comprehend!!!)

2. lstSize