

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. IstCreate
2. IstDispose

Checking number of elements

3. IstSize
4. IstIsFull
5. IstIsEmpty

Linked List Operations

- Peek Operations
 6. IstPeek
 7. IstPeekPrev
 8. IstPeekNext
- Retrieving values
 9. IstFirst
 10. IstLast
 11. IstNext
 12. IstPrev

List Operations

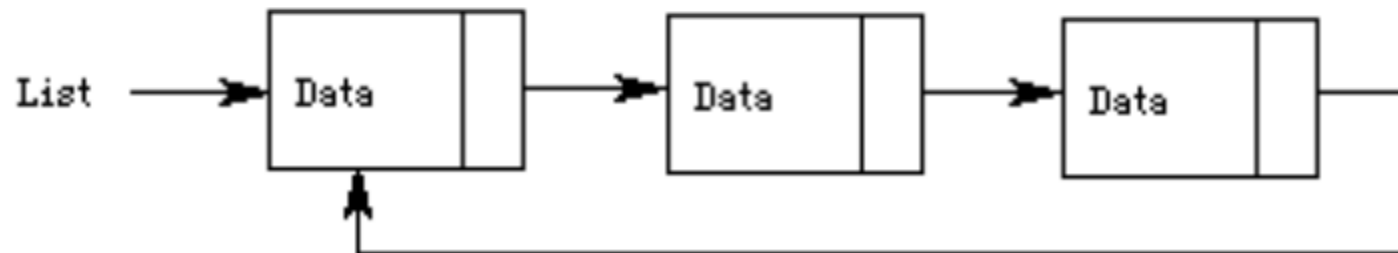
- Retrieving values
 - 13. IstDeleteCurrent
 - 14. IstInsertAfter
 - 15. IstInsertBefore
 - 16. IstUpdateCurrent
 - 17. IstHasNext
 - 18. IstHasPrev

Linked Lists

Singly Linked List

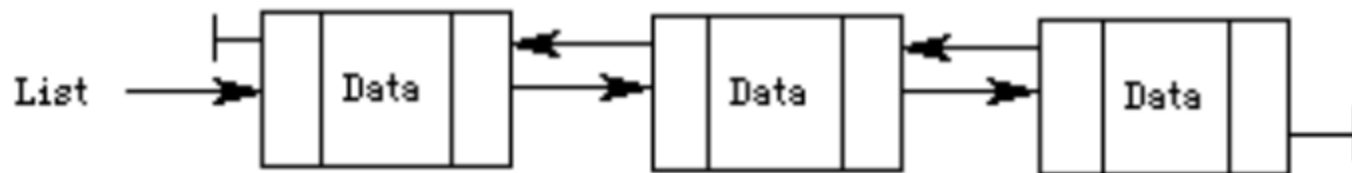


Singly Linked Circular List

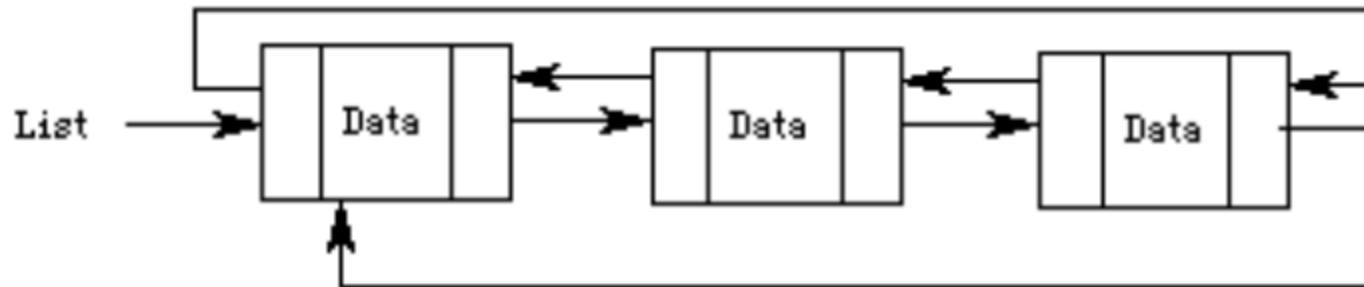


Linked Lists

Doubly Linked List



Doubly Linked Circular List



Linked List Implementation

- How might we implement the previously specified Linked List ADT?

Implementation

- There are any number of ways but let's begin with the following:

```
typedef char DATATYPE;

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

typedef struct List* ListPtr;
} typedef struct List
{
    ListElementPtr psHead;
    ListElementPtr psLast;
    ListElementPtr psCurrent;
    int numElements;
} List;
```