C++
Linked Lists

CO658 Data Structures & Algorithms

Topics

- Big-O Notation
- Linked Lists

Static Memory

- Memory which is allocated at declaration time and is fixed at that time
- Single variables and static arrays are implemented with static memory
- The size of an existing array may not be increased or decreased
- Manipulation of data in an array may involve moving many elements

Dynamic Memory

- Works by allocating blocks of memory on an as-and-when-needed basis
- It is not allocated on the stack, so it remains allocated for the life of the allocating process, unless it is freed first
- Some languages operate automatic garbage collection, so that memory out of scope is returned to the system

Big-O Notation

- Describes the performance of algorithms applied to data structures (in the worse case scenario).
- The performance is measured in terms of the order (O)
- The loop below iterates through the elements of the array. The number of elements is referred to as n.
- The performance of this code is expressed as O(n)

```
const int SIZE = 9;
int data[] = {6,4,5,3,9,8,1,2,7};
for (int n = 0;n < SIZE;n++)
      cout << data[n];
```

Big-O Notation

- The code below sorts the array.
- The outer loop's performance is O(n) however for each element of the outer loop we roughly iterate through the array again O(n). This gives a performance of O(n^2)

```
for (int n = 0;n < SIZE;n++){
    int min = n;
    for(int i=n; i< SIZE;i++){
          if (data[i] < data[min])
                min = i;
          }
          int temp = data[n];
          data[n] = data[min];
          data[min] = temp;
}
```

- The actual efficiency of the inner loop averages out to n/2 however constants are ignored so we are left with n.

Big-O Notation

Big-O Notation	Example
O(1)	N has no affect on the performance of the algorithm
O(n)	Requires a single pass. Linear search over an array or linked list.
O(n^2)	Sorting algorithms
O(log n)	Searching binary trees. Log n represents the layers within the tree.

Log is the number of times the base is multiplied to get a number.

Example : 2 * 2 * 2 = 8 number is 8 , base is 2 log is 3

Assume O(log n) uses base 2. So log2 1024 = 10.

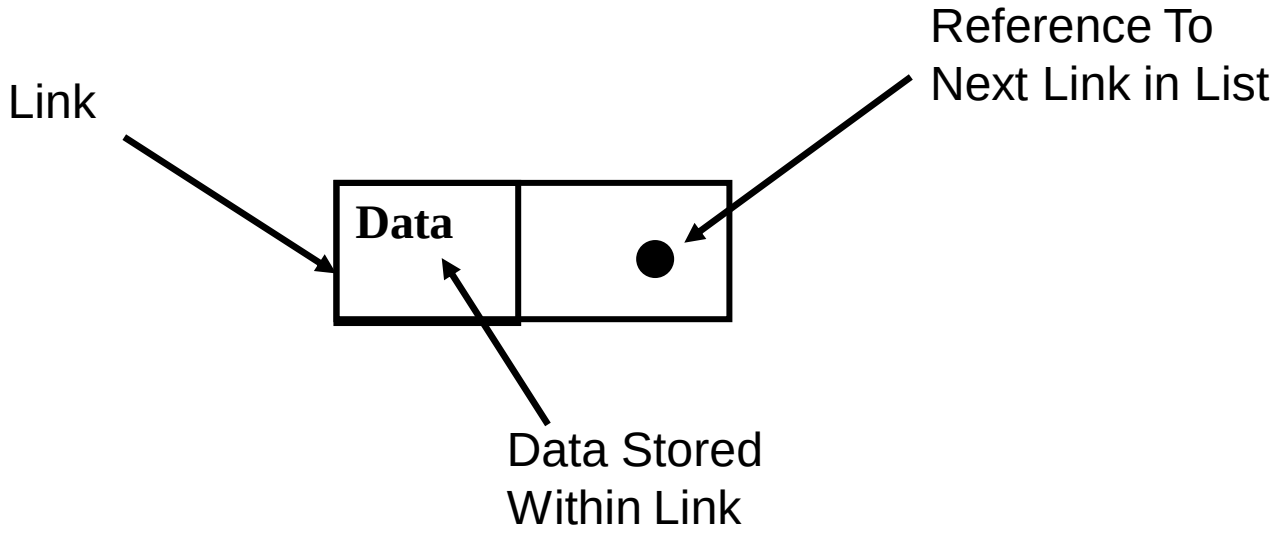
Applies to algorithms that are iteratively subdivided.

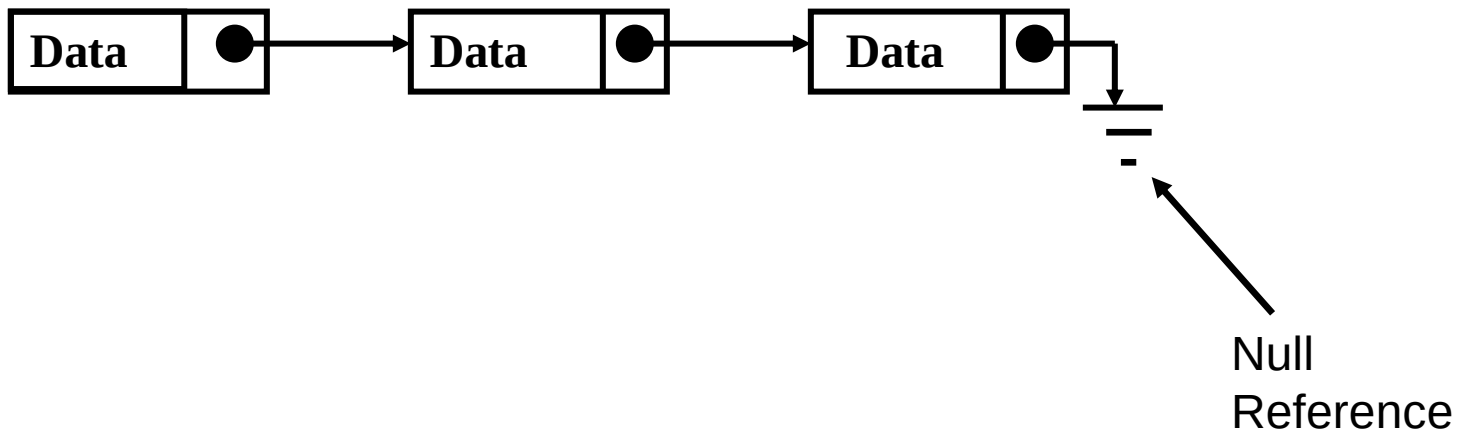
Linked Lists

Linked Lists

- Unlike arrays whose elements can be accessed directly using an index, the items in the linked list must be sequentially accessed

Connection

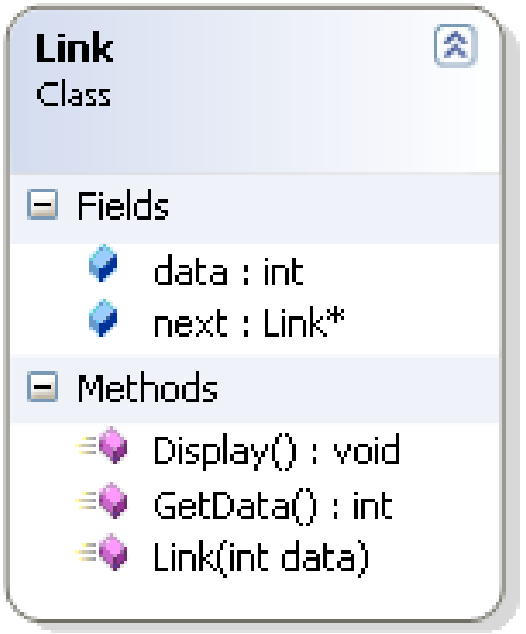

Link
Reference To
Next Link in List
Data
Data Stored
Within Link

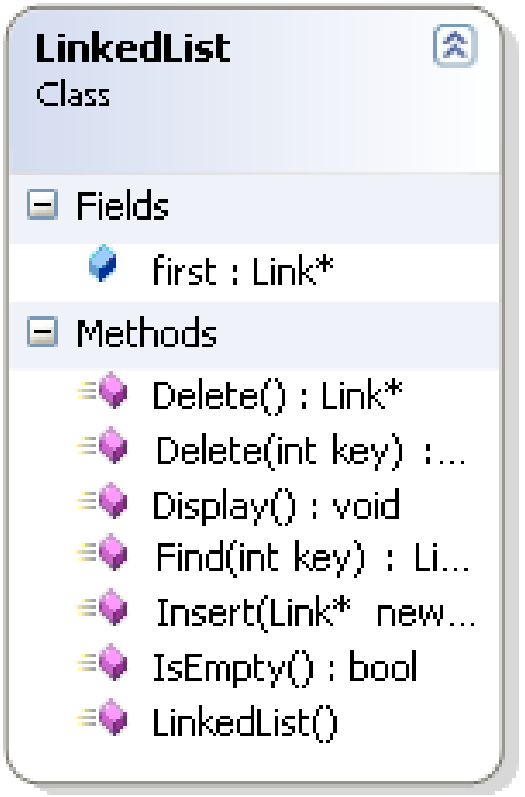

Data
Data
Data
Null
Reference

List Operations

- Initialise the list
- **Insert** – Add an item to the beginning of the list.
- **Delete** – Remove an item with a matching key value.
- **Find** - Retrieve an item with a matching key value.
- **IsEmpty** – Returns true if the list is empty

List Design


Link
Class
Fields
data : int
next : Link*
Methods
Display() : void
GetData() : int
Link(int data)


LinkedList
Class
Fields
first : Link*
Methods
Delete() : Link*
Delete(int key) :...
Display() : void
Find(int key) : Li...
Insert(Link* new...
IsEmpty() : bool
LinkedList()

List Rules

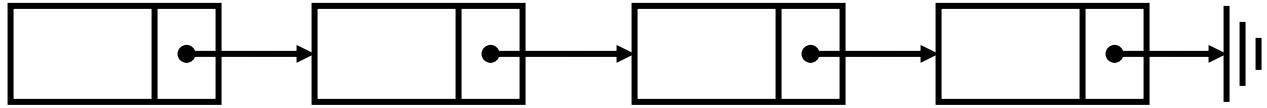
- The newly initialised list is empty
- Immediately after inserting an item into the list, the list is not empty
- Immediately after inserting and deleting the same item, the list is the same as it was before the insertion and deletion
- The previous position of the next position is the current position
- The next position of the previous position is the current position

Insertion and Deletion

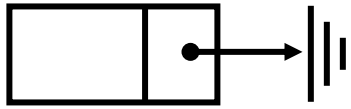
- Insertion and deletion are achieved by changing the links
- The data is not moved within the structure
- Care must be taken not to split the structure
- Structure may be maintained in order

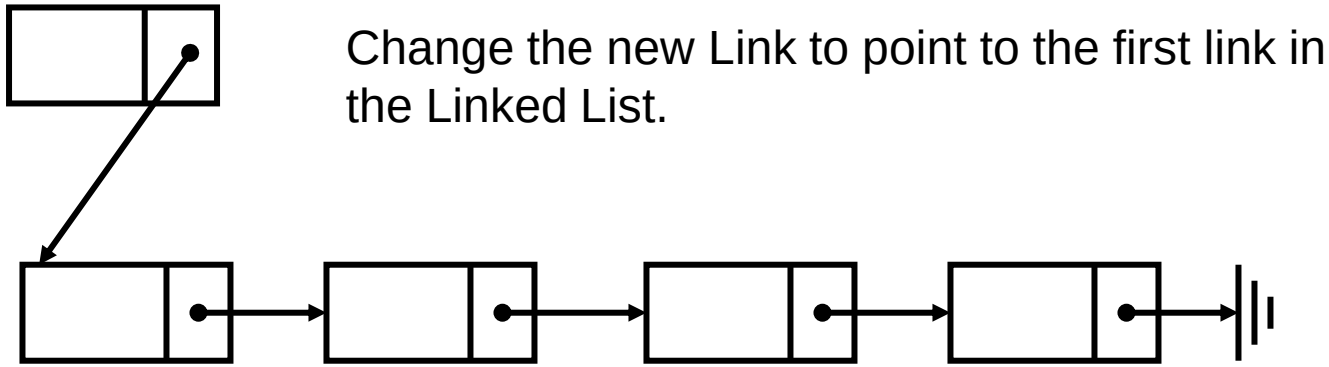
Linked List Insertion

Linked List



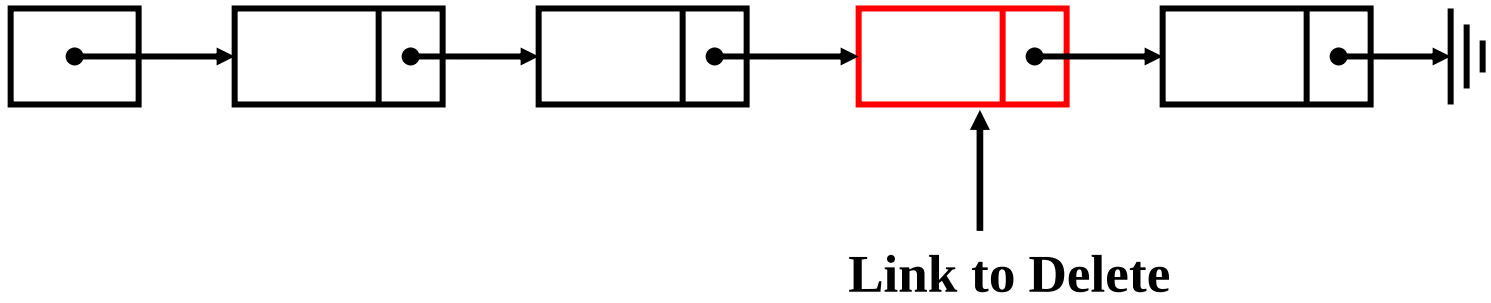
New Link



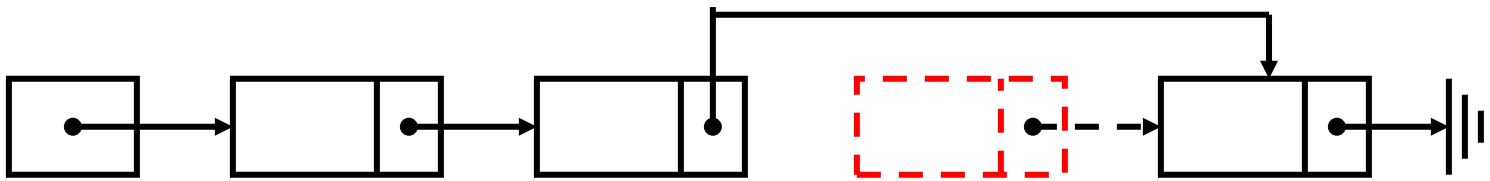


Change the new Link to point to the first link in the Linked List.

Linked List Deletion



Locate Link to be deleted.
Set Previous Link to point to the Link the deleted link pointed to.



Summary

- Both stacks and lists are Abstract Data Types.
- The ADT publishes an interface. The implementation however is hidden.
- Selecting the correct data structure is critical to the performance of the program.