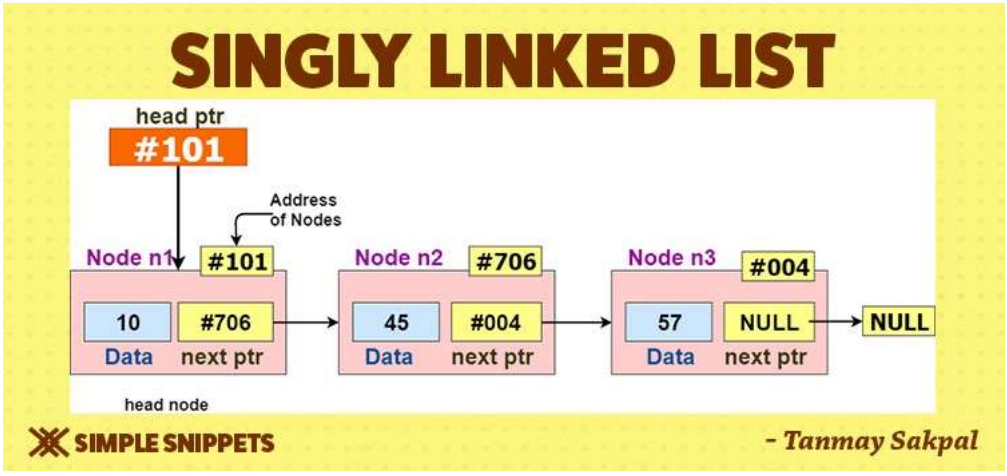


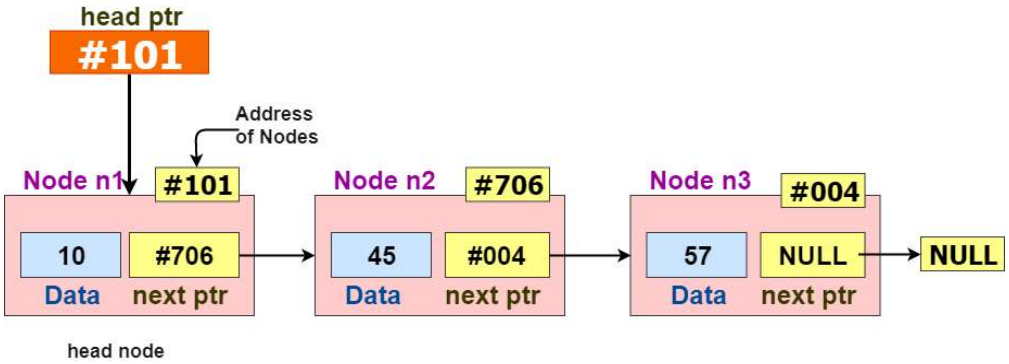

C++ Programming Tutorials Data Structures & Algorithms

# Singly Linked List Data Structure all Operations | C++ Program to Implement Singly Linked List

June 4, 2019 Tanmay Sakpal 0 Comments data structures, linked list, singly linked list

In this tutorial we will understand the working of Singly Linked List & see all operations of Singly Linked List. If you don't know what a Linked List Data Structure is please check this post.

Singly Linked list is a type of Linked List Data structure which behaves like a **one way list/chain**. The reason it is called a one way list or one way chain is because we can only **traverse this list in one direction**, start from the head node to the end.




As you can see from the diagram, each node object has 1 **data** field & 1 **pointer** field. The data field contains the actual data where as the pointer field(next pointer) points to the next node in the singly linked list. Since the nodes are not stored in contiguous memory locations, this extra pointer field assists in

## Related Links



## Popular Posts


C++ Program to Calculate Area of Triangle
75 views | by Tanmay Sakpal | posted on March 24, 2018


Binary Search Algorithm with C++ Code | Data Structures & Algorithms
53 views | by Tanmay Sakpal | posted on June 23, 2019


Linear Search Algorithm with C++ Code | Data Structures & Algorithms
50 views | by Tanmay Sakpal | posted on June 18, 2019


Creating Master Page in ASP.NET | Adding Navigation Menu & Footer to Master Page
39 views | by Tanmay Sakpal | posted on September 25, 2019


What is Binary SEARCH Tree (BST) Data structure ? | All BST operations with FULL CODE | DSA

locating the next node in memory. As we have only one pointer pointing to the next node, we can only traverse in one direction starting from the head node to the end.

Following are the standard Singly Linked List Operations –

- **Traverse –** Iterate through the nodes in the linked list starting from the head node.
- **Append –** Attach a new node (to the end) of a list
- **Prepend –** Attach a new node (to the beginning) of the list
- **Insert –** attach a new node to a specific position on the list
- **Delete –** Remove/Delink a node from the list
- **Count –** Returns the no of nodes in linked list

## C++ Program to Implement Singly Linked List –

```
#include<iostream>

using namespace std;

class Node {
  public:
    int key;
  int data;
  Node * next;

  Node() {
    key = 0;
    data = 0;
    next = NULL;
  }
  Node(int k, int d) {
    key = k;
    data = d;
  }
};

class SinglyLinkedList {
  public:
    Node * head;

  SinglyLinkedList() {
    head = NULL;
  }
  SinglyLinkedList(Node * n) {
    head = n;
  }

  // 1. CHeck if node exists using key value
  Node * nodeExists(int k) {
    Node * temp = NULL;

    Node * ptr = head;
    while (ptr != NULL) {
      if (ptr -> key == k) {
        temp = ptr;
      }
      ptr = ptr -> next;

    }
    return temp;
  }

  // 2. Append a node to the list
  void appendNode(Node * n) {
    if (nodeExists(n -> key) != NULL) {
      cout << "Node Already exists with key value : " << n -> key << ". Append
another node with different Key value" << endl;
    } else {
      if (head == NULL) {
        head = n;
        cout << "Node Appended" << endl;
      } else {
        Node * ptr = head;
```

38 views | by Tanmay Sakpal | posted on October 8, 2020

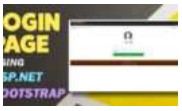





Infix to Postfix Conversion using Stack Data Structure (With C++ Program Code)
34 views | by Tanmay Sakpal | posted on March 2, 2020


Pointer to Class in C++
31 views | by Tanmay Sakpal | posted on March 19, 2018


What is AVL tree Data structure ? | Rotations in AVL tree | All AVL operations with FULL CODE | DSA
31 views | by Tanmay Sakpal | posted on January 21, 2021




Designing Login Page in ASP.NET with Bootstrap Styling | Admin & User Login Pages
28 views | by Tanmay Sakpal | posted on September 28, 2019

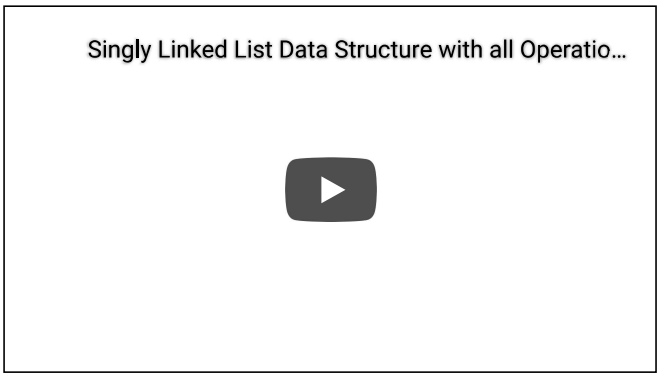

Singly Linked List Data Structure all Operations | C++ Program to Implement Singly Linked List
26 views | by Tanmay Sakpal | posted on June 4, 2019

```
        while (ptr -> next != NULL) {
          ptr = ptr -> next;
        }
        ptr -> next = n;
        cout << "Node Appended" << endl;
      }
    }

  }
  // 3. Prepend Node - Attach a node at the start
  void prependNode(Node * n) {
    if (nodeExists(n -> key) != NULL) {
      cout << "Node Already exists with key value : " << n -> key << ". Append
another node with different Key value" << endl;
    } else {
      n -> next = head;
      head = n;
      cout << "Node Prepended" << endl;
    }
  }

  // 4. Insert a Node after a particular node in the list
  void insertNodeAfter(int k, Node * n) {
    Node * ptr = nodeExists(k);
    if (ptr == NULL) {
      cout << "No node exists with key value: " << k << endl;
    } else {
      if (nodeExists(n -> key) != NULL) {
        cout << "Node Already exists with key value : " << n -> key << ". Append
another node with different Key value" << endl;
      } else {
        n -> next = ptr -> next;
        ptr -> next = n;
        cout << "Node Inserted" << endl;
      }
    }
  }

  // 5. Delete node by unique key
  void deleteNodeByKey(int k) {
    if (head == NULL) {
      cout << "Singly Linked List already Empty. Cant delete" << endl;
    } else if (head != NULL) {
      if (head -> key == k) {
        head = head -> next;
        cout << "Node UNLINKED with keys value : " << k << endl;
      } else {
        Node * temp = NULL;
        Node * prevptr = head;
        Node * currentptr = head -> next;
        while (currentptr != NULL) {
          if (currentptr -> key == k) {
            temp = currentptr;
            currentptr = NULL;
          } else {
            prevptr = prevptr -> next;
            currentptr = currentptr -> next;
          }
        }
        if (temp != NULL) {
          prevptr -> next = temp -> next;
          cout << "Node UNLINKED with keys value : " << k << endl;
        } else {
          cout << "Node Doesn't exist with key value : " << k << endl;
        }
      }
    }

  }
  // 6th update node
  void updateNodeByKey(int k, int d) {

    Node * ptr = nodeExists(k);
    if (ptr != NULL) {
      ptr -> data = d;
```

```
      cout << "Node Data Updated Successfully" << endl;
    } else {
      cout << "Node Doesn't exist with key value : " << k << endl;
    }

  }

  // 7th printing
  void printList() {
    if (head == NULL) {
      cout << "No Nodes in Singly Linked List";
    } else {
      cout << endl << "Singly Linked List Values : ";
      Node * temp = head;

      while (temp != NULL) {
        cout << "(" << temp -> key << "," << temp -> data << ") --> ";
        temp = temp -> next;
      }
    }

  }

};

int main() {

  SinglyLinkedList s;
  int option;
  int key1, k1, data1;
  do {
    cout << "\nWhat operation do you want to perform? Select Option number. Enter 0
to exit." << endl;
    cout << "1. appendNode()" << endl;
    cout << "2. prependNode()" << endl;
    cout << "3. insertNodeAfter()" << endl;
    cout << "4. deleteNodeByKey()" << endl;
    cout << "5. updateNodeByKey()" << endl;
    cout << "6. print()" << endl;
    cout << "7. Clear Screen" << endl << endl;

    cin >> option;
    Node * n1 = new Node();
    //Node n1;

    switch (option) {
    case 0:
      break;
    case 1:
      cout << "Append Node Operation \nEnter key & data of the Node to be Appended"
<< endl;
      cin >> key1;
      cin >> data1;
      n1 -> key = key1;
      n1 -> data = data1;
      s.appendNode(n1);
      //cout<<n1.key<<" = "<<n1.data<<endl;
      break;

    case 2:
      cout << "Prepend Node Operation \nEnter key & data of the Node to be
Prepended" << endl;
      cin >> key1;
      cin >> data1;
      n1 -> key = key1;
      n1 -> data = data1;
      s.prependNode(n1);
      break;

    case 3:
      cout << "Insert Node After Operation \nEnter key of existing Node after which
you want to Insert this New node: " << endl;
      cin >> k1;
      cout << "Enter key & data of the New Node first: " << endl;
      cin >> key1;
```

```
      cin >> data1;
      n1 -> key = key1;
      n1 -> data = data1;

      s.insertNodeAfter(k1, n1);
      break;

    case 4:

      cout << "Delete Node By Key Operation - \nEnter key of the Node to be deleted:
" << endl;
      cin >> k1;
      s.deleteNodeByKey(k1);

      break;
    case 5:
      cout << "Update Node By Key Operation - \nEnter key & NEW data to be updated"
<< endl;
      cin >> key1;
      cin >> data1;
      s.updateNodeByKey(key1, data1);

      break;
    case 6:
      s.printList();

      break;
    case 7:
      system("cls");
      break;
    default:
      cout << "Enter Proper Option number " << endl;
    }

  } while (option != 0);

  return 0;
}
```

## YouTube video tutorials –




← What is Linked List Data Structure ? | Arrays vs Linked List | Operations | Types | Applications