**Algorithm to insert node at the end of Singly linked list**

**Algorithm to insert node at the end of a Singly Linked List**

**Begin:**

**createSinglyLinkedList** (*head*)

**alloc** (*newNode*)

**If** (*newNode* == **NULL**) then

write ('Unable to allocate memory')

**End if**

**Else** then

read (*data*)

*newNode.data* ← *data*

*newNode.next* ← **NULL**

*temp* ← *head*

**While** (*temp.next* != **NULL**) do

*temp* ← *temp.next*

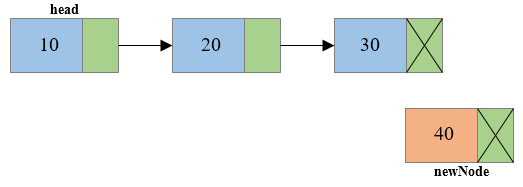
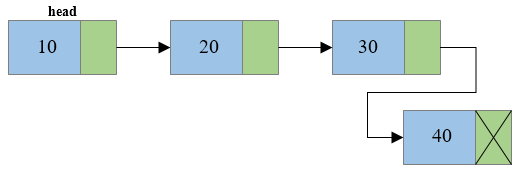
**End while**

*temp.next* ← *newNode*

**End else**

**End**

## Steps to insert node at the end of Singly linked list

1. Create a new node and make sure that the address part of the new node points to NULL i.e. newNode->next=NULL
2. Traverse to the last node of the linked list and connect the last node of the list with the new node, i.e. last node will now point to new node. (lastNode->next = newNode).

## Program to insert node at the end of Singly linked list

/\*\*

\* C program to insert new node at the end of a Singly Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

/\* Structure of a node \*/

struct node {

int data; // Data

struct node \*next; // Address

}\*head;

void createList(int n);

void insertNodeAtEnd(int data);

void displayList();

int main()

{

int n, data;

/\*

\* Create a singly linked list of n nodes

\*/

printf("Enter the total number of nodes: ");

scanf("%d", &n);

createList(n);

printf("\nData in the list \n");

displayList();

/\*

\* Insert data at the end of the singly linked list

\*/

printf("\nEnter data to insert at end of the list: ");

scanf("%d", &data);

insertNodeAtEnd(data);

printf("\nData in the list \n");

displayList();

return 0;

}

/\*

\* Create a list of n nodes

\*/

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

head = (struct node \*)malloc(sizeof(struct node));

/\*

\* If unable to allocate memory for head node

\*/

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

/\*

\* Reads data of node from the user

\*/

printf("Enter the data of node 1: ");

scanf("%d", &data);

head->data = data; // Link the data field with data

head->next = NULL; // Link the address field to NULL

temp = head;

/\*

\* Create n nodes and adds to linked list

\*/

for(i=2; i<=n; i++)

{

newNode = (struct node \*)malloc(sizeof(struct node));

/\* If memory is not allocated for newNode \*/

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link the data field of newNode with data

newNode->next = NULL; // Link the address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

/\*

\* Create a new node and inserts at the end of the linked list.

\*/

void insertNodeAtEnd(int data)

{

struct node \*newNode, \*temp;

newNode = (struct node\*)malloc(sizeof(struct node));

if(newNode == NULL)

{

printf("Unable to allocate memory.");

}

else

{

newNode->data = data; // Link the data part

newNode->next = NULL;

temp = head;

// Traverse to the last node

while(temp != NULL && temp->next != NULL)

temp = temp->next;

temp->next = newNode; // Link address part

printf("DATA INSERTED SUCCESSFULLY\n");

}

}

/\*

\* Display entire list

\*/

void displayList()

{

struct node \*temp;

/\*

\* If the list is empty i.e. head = NULL

\*/

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print data of current node

temp = temp->next; // Move to next node

}

}

}

OUPUT:

Output

Enter the total number of nodes: 3

Enter the data of node 1: 10

Enter the data of node 2: 20

Enter the data of node 3: 30

SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list

Data = 10

Data = 20

Data = 30

Enter data to insert at end of the list: 40

DATA INSERTED SUCCESSFULLY

Data in the list

Data = 10

Data = 20

Data = 30

Data = 40