form now use temp instead of current

ChatGPT

Sure! Here's the updated algorithm for appending a node at the end of a singly linked list using the variable name temp instead of current:

Create a new node with the given data.

If the head is None, set the head to the new node and exit.

Initialize a pointer temp to the head.

Traverse the linked list until the next node of temp is None.

Update temp to its next node.

Set the next of the last node (temp) to the new node.

Exit.

Pseudocode:

python

Copy code

append(data):

new\_node = Node(data)

if head is None:

head = new\_node

exit

temp = head

while temp.next is not None:

temp = temp.next

temp.next = new\_node

exit

Using temp as the variable name doesn't affect the functionality of the algorithm. It's just a different variable name for clarity and readability.

User

algorithm for prepending

ChatGPT

Certainly! Here's the algorithm for prepending a node at the beginning of a singly linked list:

Create a new node with the given data.

Set the next of the new node to the current head.

Update the head to point to the new node.

Exit.

Pseudocode:

bash

Copy code

prepend(data):

new\_node = Node(data)

new\_node.next = head

head = new\_node

exit

This algorithm inserts a new node at the beginning of the linked list by updating the head pointer to the new node, making the new node the new head, and linking it to the previous head node.

User

algorithm for insert after

ChatGPT

Certainly! Here's the algorithm for inserting a node after a specific node in a singly linked list:

Create a new node with the given data.

Traverse the linked list until the node after which the new node needs to be inserted.

Set the next of the new node to the next of the current node.

Set the next of the current node to the new node.

Exit.

Pseudocode:

lua

Copy code

insert\_after(prev\_node, data):

new\_node = Node(data)

new\_node.next = prev\_node.next

prev\_node.next = new\_node

exit

This algorithm inserts a new node after the specified prev\_node by linking the new node to the node next to the prev\_node, and updating the next of the prev\_node to point to the new node.

User

algorihtm for delete

ChatGPT

Certainly! Here's the algorithm for deleting a node from a singly linked list:

Start at the head of the linked list.

Traverse the linked list until the node to be deleted is found or the end of the list is reached.

If the node to be deleted is found, update the next of the previous node to skip the node to be deleted.

Exit.

Pseudocode:

python

Copy code

delete(data):

if head is None:

print "Linked list is empty."

exit

if head.data == data:

head = head.next

exit

temp = head

prev = None

while temp and temp.data != data:

prev = temp

temp = temp.next

if temp is None:

print data, "is not found in the linked list."

exit

prev.next = temp.next

exit

This algorithm deletes the node with the specified data from the linked list by finding the node to be deleted, updating the next of the previous node to skip the node to be deleted, and effectively removing it from the linked list.

User

algorithm for search

ChatGPT

Here's the algorithm for searching for a node in a singly linked list:

Start at the head of the linked list.

Traverse the linked list until the node with the desired data is found or the end of the list is reached.

If the node with the desired data is found, return True to indicate that it exists in the linked list.

If the end of the list is reached without finding the desired data, return False to indicate that it doesn't exist in the linked list.

Exit.

Pseudocode:

kotlin

Copy code

search(data):

if head is None:

print "Linked list is empty."

return False

temp = head

while temp:

if temp.data == data:

return True

temp = temp.next

return False

This algorithm searches for a node with the specified data in the linked list by traversing the list and comparing the data of each node with the desired data. If a node with the desired data is found, it returns True, indicating that the data exists in the linked list. Otherwise, it returns False.

User

algorithm for display

ChatGPT

Here's the algorithm for displaying the elements of a singly linked list:

Start at the head of the linked list.

Traverse the linked list from the head to the end.

Print the data of each node.

Move to the next node.

Repeat steps 3-4 until the end of the linked list is reached.

Exit.

Pseudocode:

python

Copy code

display():

if head is None:

print "Linked list is empty."

return

temp = head

while temp:

print temp.data

temp = temp.next

This algorithm displays the elements of the linked list by traversing the list and printing the data of each node. It starts at the head and moves to the next node until the end of the list is reached.