

GROUP ASSIGNMENT

Module code: AAPP001-4-2-DBDS

Module title: Database and Data Structures

Assignment title: Linked List (Question 1)

Lecturer name: Mr. Abdallah S.M Alnatsha

Group members:

|  |  |
| --- | --- |
| Name | TP Number |
| Ong Chang Chen | TP037265 |
| Isa Chong bin Abdullah Chong @ Chong Chee Keong | TP037515 |
| Lau Hong Yong | TP037423 |

Contents

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) Code snippet explanations 4

[2.1.](#_1fob9te) Implementation of structure 4

[2.2](#_3znysh7) Node insertion after node 4

[2.3](#_2et92p0) Node insertion between nodes 4

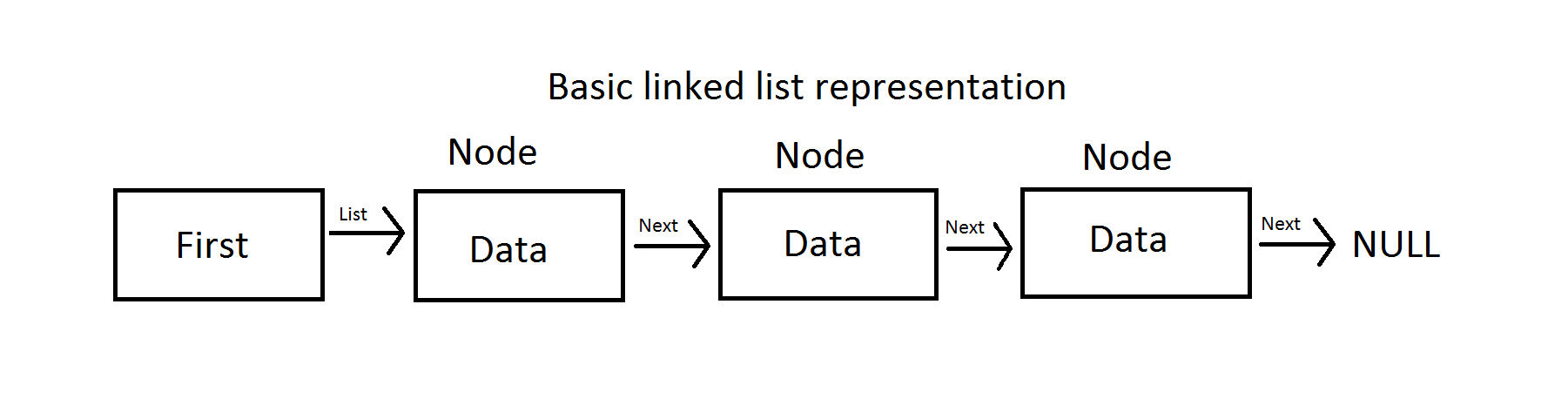
[2.4 Node insertion process explanation 6](#_tyjcwt)

[2.5 Node display explanation 8](#_3dy6vkm)

[2.6 Node deletion 9](#_1t3h5sf)

# Introduction

A linked list is a set of data structures which are connected using links. Each link contains a link to another link which contains a set of data, which contains what the user wants to enter. Linked lists is the second most used data structure after arrays in C programming. A diagram representation of a linked list is shown below.



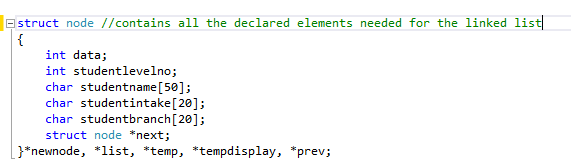
*Figure 1: Linked list representation using diagram*

In the above diagram, there are a few important terms in the linked list to be understood. The first one is List, which contains the first link from itself to the first node. The second term is Node, which contains a data field with user-input data and the last term is Next, which serves as a link to other nodes in the linked list. There are various types of linked lists such as singly linked list, doubly linked list and circular linked list. For this assignment, a singly linked list is used.

# Code snippet explanations

# Implementation of structure

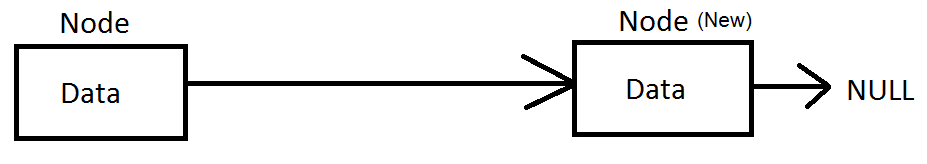
A structure is a user defined data type that allows users to input a set of data with different data types. For example, the user can input int, char, and float data types in the structure itself. The main difference between a structure and an array is that arrays can only store 1 kind of data type. To declare a structure, the variable struct () is used. The below figure shows the implementation of a structure in the program.



*Figure 2: A structure inside the program*

# Node insertion after node

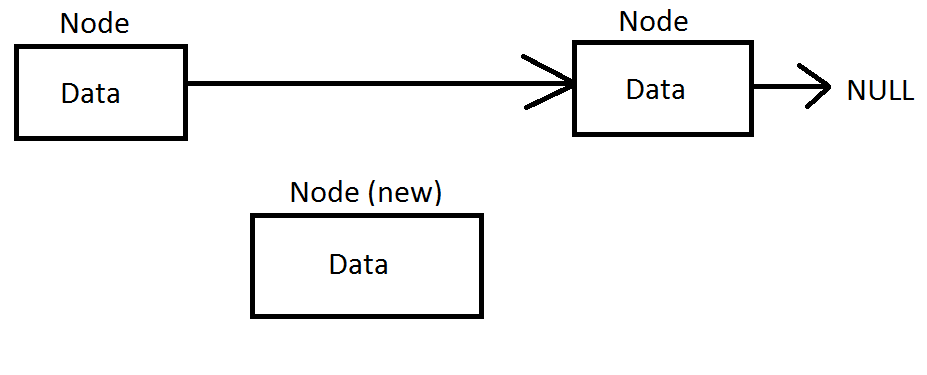
The concept of node insertion is a process of linking nodes to each other. To insert a node after a node, the new node is just linked back to the previous node. The diagram below shows the concept of node insertion after node.



*Figure 3: Node insertion after node*

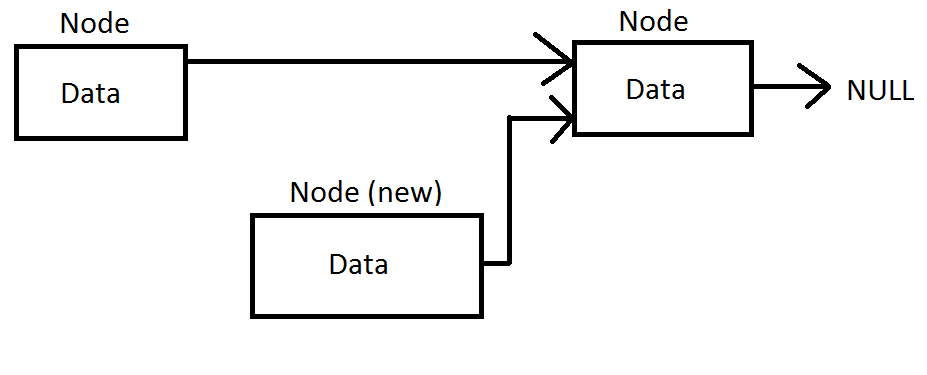
# Node insertion between nodes

The concept of node insertion is a process of linking nodes to each other. To insert a node between nodes, a new node is pointed to the next node in the list, then it is linked back to the previous node in the list. The diagrams below illustrate the concept of node insertion.



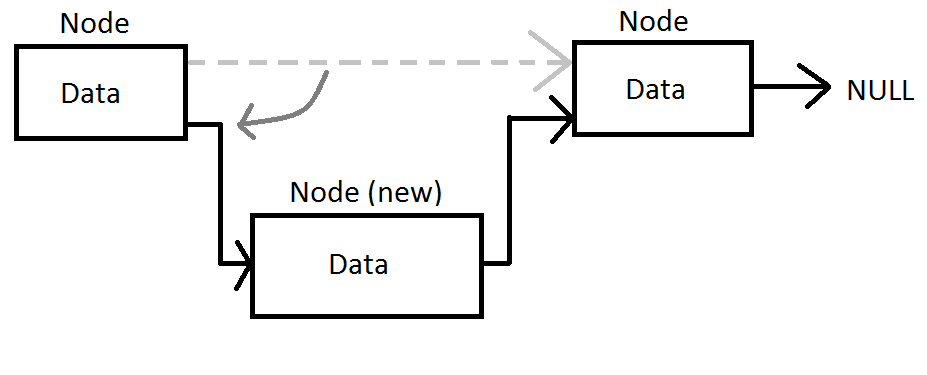
*Figure 4: First step for node insertion*

As shown in Figure 3, a new node is going to be inserted into the list. To do that, it must be linked to the previous node and the next node.



*Figure 5: Step 2 for node insertion*

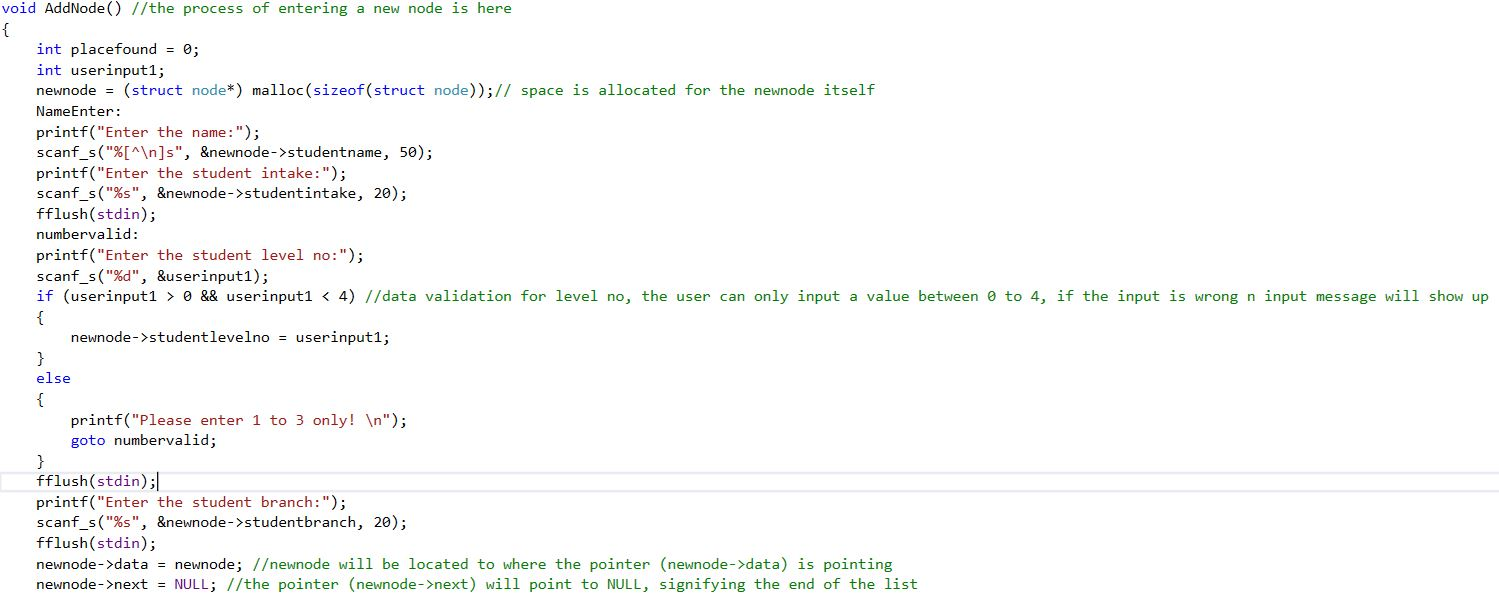
Figure 4 shows the second step for node insertion. As shown in the figure above, the new node is pointing to the next node inside the list.



*Figure 6: Step 3 for node insertion*

In the final step, the new node is linked back to the previous node in order to complete the insertion steps. Figure 5 shows the previous node of the list pointing to the new node, thus completing the process.

# 2.4 Node insertion process explanation



*Figure 7: Add node part 1*

Figure 7 shows the first part of the process of adding a node. First, the new node is allocated a space in the memory and it is shown in the diagram below.



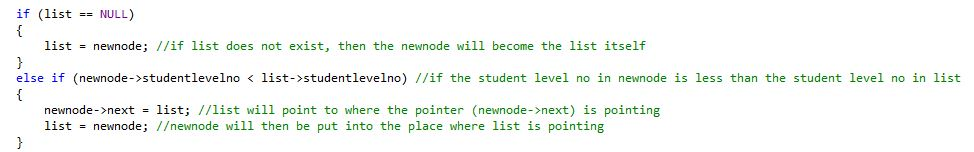
*Figure 8: Space allocation for new node*

After that, the program will accept the user input. Once the user has finished entering the data they want, the program will start the insertion process. The figure below shows the code for the insertion of node in the linked list.



*Figure 9: Insertion of node in a linked list (Part 1)*

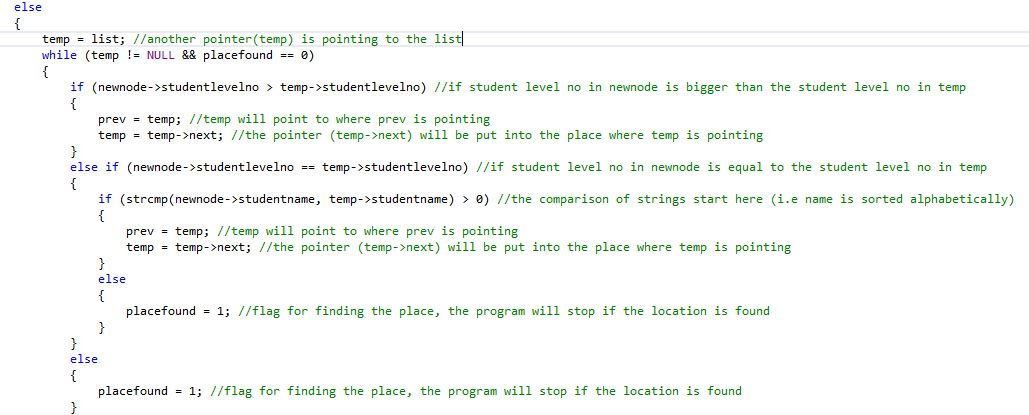
As shown in the figure above, newnode is the data that the user has just entered, newnode->data is a pointer itself. Since the above code is executed when there is no data in the linked list, newnode->data will point to newnode, then it will be pointed to NULL, which is the end of the list itself.



*Figure 10: Insertion of node in a linked list (Part 2)*

The above figure continues the insertion process for an empty linked list. In the figure above, list is another pointer. If list is pointing to NULL, then the newnode will become the list itself. That condition is only eligible if the linked list has nothing inside.

The else if statement below states that if the student level number entered by the user is smaller than the one in the list, sorting of data will occur. The list will point to where the pointer (newnode->next) is pointing, then newnode will be pointing to the same place as list.

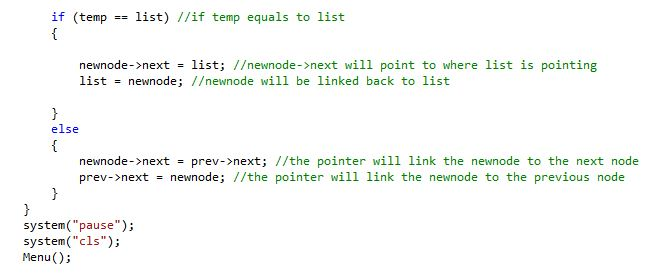


*Figure 11: Insertion of node in a linked list (Part 4)*

The above else statement is executed when the list is not empty and the level numbers of the student is bigger than that in the list. In this code block, the sorting of names alphabetically will occur. First, a new pointer (temp) is used here, and list will be pointed to it. Then the navigation of list will occur in the while loop, in the while loop, as long as temp is not pointing to NULL and placefound is equal to 0, it will continue to loop.

In the while loop, when the student level number is bigger than that in the list, another 3 pointers called temp, temp->next and prev are used. To insert the node, temp will point to where prev is pointing, then temp->next will point to where temp is pointing. This completes the sorting process for level numbers.

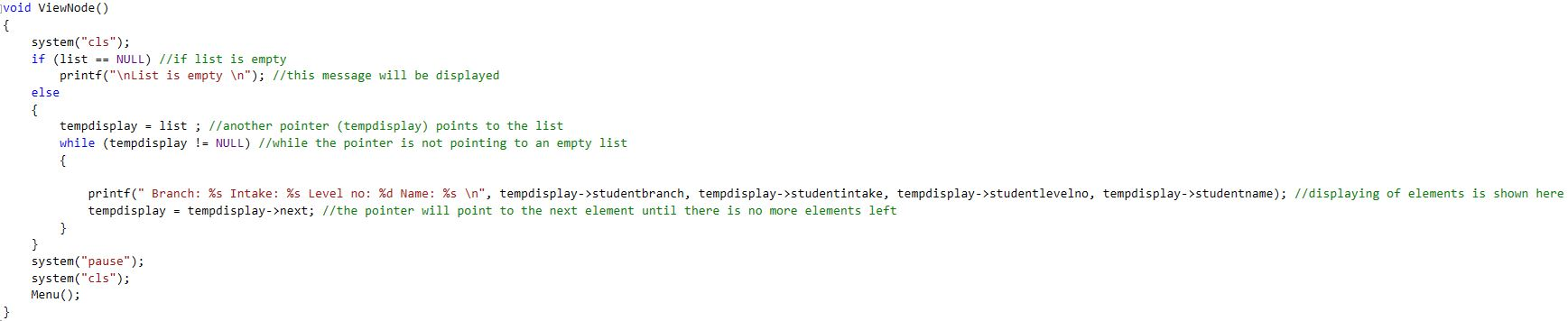
If the student number inputted by the user is the same as the one in the list, sorting of names will occur. The sorting of names is done using the ASCII values of the alphabets to be compared with each other. If the ASCII value of the user inputted value is bigger than the one in the temp, temp will point to where prev is pointing, then temp->next will point to where temp is pointing. This completes the insertion of node and also completes the sorting process for alphabets. The program will stop looping when placefound equals to 1.



*Figure 12: Insertion of node in a linked list (Part 5)*

The above figure shows the last step of inserting a node in the linked list. If temp is equal to list, then the pointer list will be pointing to where the pointer newnode->next is pointing. If not, then the pointer will link the newnode to the next node and to the previous node as well. Lastly, the program will go back to the main menu.

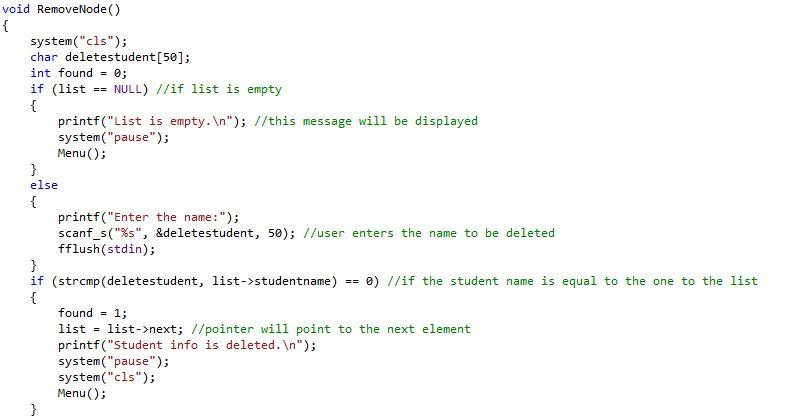
# 2.5 Node display explanation



*Figure 13: Displaying of nodes*

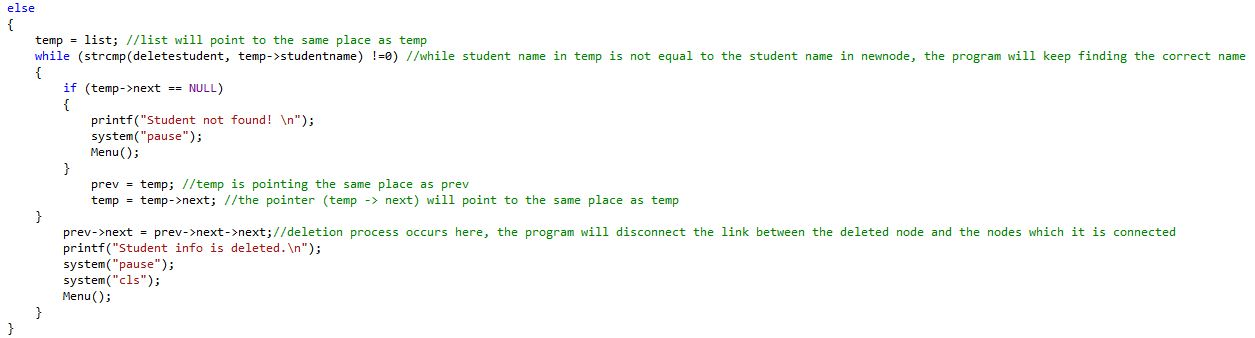
The above figure shows the display of nodes in the program. First, the program will check if the list itself has data or empty, if the list is empty then the program will display a message then go back to the main menu. If the list has data in it, a pointer called tempdisplay is used. Now, the program will loop. While tempdisplay is not pointing towards NULL, the program will display all the items in the linked list, navigation in the list occurs when tempdisplay->next is pointing towards tempdisplay, making the next pointer the first pointer. Then the user can press any key to go back to the main menu.

# 2.6 Node deletion



*Figure 14: Node deletion part 1*

The above figure shows the process to delete a node. First the program will check if the list is empty or full, if it is empty then a message is shown and the user will be redirected back to the main menu. If not, the user will be prompted to enter the name of the student. After that, comparison of strings will occur. If the deleted student is the first student in the list, the next pointed element (list->next) will be the first element in the list (list). A message will pop out showing that the student is deleted, then the user will be redirected back to the main menu.



*Figure 15: Node deletion part 2*

If the deleted student is not the first one, then the list will be pointed to temp, which is another pointer. While the inputted name is not equal to the one in the list, it will keep looping until it is pointing to NULL. When it is pointing to NULL, a message will show stating that the student is not found, then the user will be redirected to the main menu. If the student is found, the deletion process will start.

Deleting a node in a linked list is a process of disconnecting a node from other nodes. In this deletion process, the pointer prev->next->next will be pointing to prev->next, which disconnects the node in the linked list. After deleting a node, the user will be taken back to the main menu.