|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | **Programming Fundamentals** |  | |  | **(CL214)** |  | |  | **LABORATORY MANUAL** |  | |  | **Spring 2021** |  | |  | **C:\Users\Aamer\Desktop\nu-new.png**  **LAB 13** |  | |  | **Linked List** |  | |  | **Engr. Ibrar Khan**  **Engr. Sana Saleh** |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | STUDENT NAME | | ROLL NO | | | SEC | |  | | | | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | | LAB ENGINEER SIGNATURE & DATE | | | | | | | **MARKS AWARDED: /10** | | | | | | |  | | | | | | | **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD** | | | | | | |  | | | | | | | Last Edited by: | Engr. Sana Saleh | | Version: | 4.01 | | | Prepared by: | Engr. Sana Saleh, Engr Sahar Liaqat | | Date: | 6 Dec, 2018 | | | Verified by: | Engr. Shahid Qureshi | | Date: | 29 April,2019 | | |

|  |  |
| --- | --- |
| **LAB13** | **Linked List** |

**Lab Objectives:**

1. To learn how to use Linked list.
2. To learn how to add new node at start, end and at specific position in a linked list.
3. Delete specific node enter by user.

**Software Required:**

* Dev C++

**Introduction:**

1. **Linked List:**

A linked list is a data structure that can store an indefinite number of items. These items are connected using pointers in a sequential manner.

The elements of a linked list are called the **nodes**. A node has two fields i.e. **data** and **next**. The data field contains the data being stored in that specific node. It cannot just be a single variable. There may be many variables presenting the **data** section of a node. The **next** field contains the address of the next node. So, this is the place where the link between nodes is established.

Creating a structure for node:

|  |
| --- |
| struct nodeType  {  float info;  nodeType \*link;  }; |

No matter how many nodes are present in the linked list, the very first node is called **head**.

* 1. **Creating new list:**

Now, we need a class which will contain the functions to handle the nodes. This class should have head pointers. The constructer will make them **NULL** to avoid any garbage value.

|  |
| --- |
| class FloatList  {  private:  nodeType \*head; // List head pointer  public:  FloatList(void) // Constructor  {  head = NULL; } |

* 1. **Creating New node:**

The process of creating node is very simple. We need a pointer of a node type (which we defined) and we will insert the value in its data field. The next field of node would be declared as NULL as it would be the last node of linked list.

Now, the function will have a very special case that we want to know what would happen if the linked list is still empty? We will have to check it. Do you remember that the head points to the first node? It means if the head is equal to NULL then we can conclude that the linked list is empty.

If there is just one node (which we are going to create) in linked lists, then it is called head.

The creation of a new node at the end of linked list requires, linking of the newly created node with last node. Means passing the address of a new node to the link pointer of a last node.

* 1. **Insertion:**

Inserting a new node in the linked list is called insertion. A new node is created and inserted in the linked list. There are three cases considered while inserting a node:

* Insertion at the start.
* Insertion at the end.
* Insertion at a particular position.

C++ code for insertion at start and end is given above.

* 1. **Deletion:**

Linked lists provide us the great feature of deleting a node. The process of deletion is also easy to implement. The basic structure is to declare a temporary pointer which points the node to be deleted. Then a little bit of working on links of nodes. There are also three cases in which a node can be deleted:

* + 1. Deletion at the start
    2. Deletion at the end
    3. Deletion at a particular position

**Deletion of node:**

The C++ code for deleting a node having particular value, entered by user, is given below:

|  |
| --- |
| void deleteNode(float num)  {  nodeType \*nodePtr, \*previousNode;  // If the list is empty, do nothing.  if (head == NULL)  {  return;  }  // Determine if the first node is the one to be deleted  if (head->info == num)  {  nodePtr = head->link;  delete head;  head = nodePtr;  }  else  { // Initialize nodePtr to head of list  nodePtr = head;  // Skip all nodes whose info member is not equal to num.  while(nodePtr!= NULL && nodePtr->info != num)  {  previousNode = nodePtr;  nodePtr = nodePtr->link;  }  //if end of linked list has been reached, and info has not been found  if(nodePtr == NULL)  {  cout<<num<<" not found\n\n";  }  // Else if the info has been found  else  { // Link the previous node to the node after  // nodePtr, then delete nodePtr.  previousNode->link = nodePtr->link;  delete nodePtr;  }  }  } |

**Practice Problems:**

1. Write a program in C++ to create a linked list that stores the age of the student i.e. integer data. Initially, create five nodes of linked list using append function and display them on the console. After that, display a menu on the console, through which the user can select the tasks that he/she wants to perform on the linked list. The tasks are mentioned below:

* Insertion of node at a particular position, entered by the user.
* Deletion of a node at a particular position, entered by the user.
* Find the total number of even numbers in the linked list.
* Find the position of all prime numbers in the linked list.
* Print the linked list.