Project 5

Exercising a Doubly-Linked List Class

Full Name: Prashul Shrestha

Section #: 2

Project #: 5

Due Date: March 2, 2017.

**Design** **Document**

**Introduction**

A **doubly linked list** is a **linked** data structure that consists of a set of sequentially **linked** records called nodes. Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes. In this linked list, each node consists of two pointers associated with it, one pointing to the next node and other pointing to the back node. It is very compatible than singly-linked list since in double linked we can move to next node and previous node easily using two pointers associated with the nodes.

http://web.stcloudstate.edu/bajulstrom/cs301/projects/double.png

**Data** **Structures**

The program uses a class called Class **doblist()**, a char **array**[], and a doubly linked list to store each character into the node inputted by the user. First, the user enter the string sentence and each characters is read and stored into the char **array[]** which is then sent by reference to the implementation file in the **append()** function to save it in the node using the **get\_node()**  function. After all the characters are stored as a node in doubly-linked list, the **remove\_last()** functions gets called in order to check for the **‘#’** character and if found, the node previous to it gets deleted and also the node containing the **‘#’** character gets deleted.

**Functions**

The program uses **Six** functions to implement the Double-linked List. The functions are called from main() and some are member function to return the result within the function which called it. The list of the functions are given below:

* Doblist() : A constructor to initialize the list to be empty.
* ~DobList(): A destructor to delete nodes in the list and is called by default.
* **Void append**(**entry**) – This will insert characters in doubly-linked list.
* **Void remove\_last()**– This will remove the node containing the ‘#’ character and also the node previous to the node where the ‘#’ is found. Here, the linked list is looped in a reverse manner so that the efficiency is increased.
* Bool is\_empty(): checks if the linked list is empty or not.
* Output - Overloads the "<<" operator to write the invoking concordance to an output stream.
* **get**\_**node**(**entry**, node forward, node backward) – Returns or initializes new node to store the characters as double linked list.

The program uses **string**.**size**() from **string**.h library to get the length of the string. It also uses **assert()** function from the **assert** library to check the character is ‘#’.

**Menu**() is a the main function from where we first create an object or an instance of class **doblist** called **charc** and which helps invoke the functions such as **charc.append**(ch); and calls **remove\_last()** to remove the node.

**The Main Program**

**Menu**() is a the main function from where we first create an object or an instance of class **charc** and which helps invoke the functions such as **charc.append**(**entry**); and **charc**.**remove\_last** () to input characters into the node and remove the node. The program is quite simple first, the characters are inserted into the node and later the remove function deletes the node as per the necessary requirement.

User Document

A **doubly linked list** is a **linked** data structure that consists of a set of sequentially **linked** records called nodes. Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes. In this linked list, each node consists of two pointers associated with it, one pointing to the next node and other pointing to the back node. It is very compatible than singly-linked list since in double linked we can move to next node and previous node easily using two pointers associated with the nodes.

The main program named **DoubleList**.**cpp** can be compiled and run, using the code:

**g++ Doublelist**.**cpp**

**a.out**

**g++** function will compile the function and make it ready to be run using **a.out**. The function will prompt the following output:

**Ouptut:**

**Enter a line of characters.**

**# => delete the most recent character.**

**Input --> aa#bb#cc#dd#ee#f**

**The linked list is not empty.**

**Output:----> abcdef**

**Summary**

Completing this project, I learnt the use of double-linked list in the real-world experience. Declaring nodes pointer to delete nodes and rearrange linked list made me through of double linked list and implementing it on other programming problems can be very easy.

In this program, I used char **array** [] of the inputted string to store each character from the text. Each element from the array is sent to the function **append(entry)** to insert items into the linked list from **get\_node()** functionsusing objects of a class **doblist** called **charc.**

Other, thing I encountered while doing the project was that according to the question, we must delete the characters before the character ‘**#**’ and moving into the linked list in a forward direction created more complex issues. So, therefore, I decided to read the list in the reverse way and this made the code much simpler and much efficient. So, doing this, I am now so much familiarized in using pointers to create a linked list or in any case in future coding.