EADS LAB 1 DOCUMENTATION

Sai Koushik Neriaynuri (317100)

16th November 2022

This program is a complete implementation of a singly-linked list using templates. It consists of class Sequence (which contains structure Node along with other methods crucial for singly-linked list to work) and two external functions called split_by_pos, split_by_key, defined outside of class Sequence. Below, I've written down description of each method defined in the class.

Private Members in class sequence:

- Struct Node – A structure representing a single element of a singly-linked list. It contains pointers to elements key and info and a pointer to the next node of the singly-linked list.
- Constructor of Node – Struct Node contains the constructor of Node, which constructs elements of the node.
- Node*head – a pointer to which points to the first element of the Node in the list.
- Sequence_Length – A variable which stores the length of a sequence.
- printNode - A private method which displays all the information about the particular node which is passed to this method
- copySequence - this method is used for copying contents of one sequence into another

Public Methods in class sequence:

- Sequence() - constructor of the Sequence class which sets the head pointer to the nullptr.
- ~Sequence() - destructor of the Sequence class instance which uses the public method clearSequence() which deletes all the elements of singly-linked list and what is more important, free the memory taken by the nodes to prevent memory leaking after destruction.
- Sequence(const Sequence<Key, Info>&) - copy constructor sets its head pointer to nullptr, then uses 'this' operator to assign the passed Sequence to the newly created one.
- operator= - copies the contents of the one sequence to another and returns the newly created sequence.
- operator+ - adds two sequences and returns the newly created sequence
- operator+= - adds another sequence to itself and returns the newly created sequence

- operator[] - It traverses the sequence and returns the info of element of passed index. Where indexes of elements in the sequence are understood as elements of an array : a[0], a[1],....
- operator== - compares two sequences and returns true if two sequences have exactly the same elements
- operator!= - compares two sequences and returns true when given sequences are different are different
- operator> - compares two sequences and returns true if the length of seq1 > seq2
- operator>= - compares two sequences and returns true if length of seq1 >= seq2
- operator< - compares two sequences and returns true if length of seq1 < seq2
- operator<= - compares two sequences and returns true if length of seq1 <= seq2
- getInfobyKey() - method which returns the info of the elements based on the provided key and number of occurrences, throws an error if the key doesn't exist or if the sequence is empty
- getFirstElemInfo() - method returns info of the first element, throws an error if the sequence is empty
- getLastelemInfo() - Even though this method is inefficient as there is not tail pointer, I wrote it as an example. This method returns info of the last element, throws an error if the sequence is empty
- getKeyByInfo(), getLastElemKey(), getFirstElemKey() – These three methods work in the same manner as the above three
- insertAtBegining - method which inserts the element at the beginning of the singly-linked list by creating the new Node with the given Key and Info and setting head pointed to this node and setting the next pointer to the previously first element.
- InstertAtTheEnd - method which creates the element with the given Key and Info and inserts it at the end by setting the pointer 'next' of the last element (obtained by iterating) from nullptr to the newly created element
- insertAfter – This method creates an element with the given Key and Info and inserts it after the element with a given Key, Throws an error if the given key doesn't exist or if it's an empty list.
- clearSequence - method which iterates through the singly-linked list and deletes all of its elements and sets the head pointer to the nullptr and protects from memory leaking
- removeByKey, removeByInfo, removeLastElem, removeFirstelem, removeAtPosition – Methods which transverse the list and return true if the elements were deleted and false if they don't exist.
- findByInfo, findByKey, findElem - Boolean methods which return true if, with the specified key or/and info passed, the elements was found in the sequence.
- findIndexByKey - this method is used to find index of a given key in the sequence while also considering it's occurence. Returns index of the given key, It is also used in split_by_key function.
- sequenceLength() - returns length of the provided sequence method which iterates through the singly-linked list and adding up the occurrences of elements to the variable which is returned by this function this function as the rest of the similar methods uses the isEmpty() method to check whether the sequence is empty or not (if so then return 0)
- printsequencelength() - prints the length of the provided sequence
- printSequenceData, printSequenceKeys, printSequenceInfo - prints all the data, keys and infos that are in the provided sequence

- subSeqToEndFrom - method which returns the rest of the sequence from the given index
- subSeqFromByLength - method which returns the part of the sequence called subsequence depending on the startIndex and length parameter, this method is used in both split_by_key and split_by_pos functions.
- subSeqFromBeginTo - method which returns part of the sequence from the beginning to the given index.

External Functions defined outside of class sequence:

- split_by_pos – splits the given sequence into two sequences based on the provided position (index). It makes use of the public method subSeqFromByLength() defined in class sequence. It throws an error if the starting position is greater than the sequence length.
- split_by_key - splits the given sequence into two sequences based on the key provided and it's occurrence.it It makes use of both the public method findIndexByKey defined in class sequence as well as the external method split_by_pos.