07.11.2018

SEQUENCE

SINGLE LINKED LIST

WIKTOR BARA

TASK

Design a template of single linked list and implement external testing function “split”.

Int split(const Sequence<Key,Info>& source, int start, Sequence<Key,Info>& result1, int step1, Sequence<Key,Info>& result2, int step2) – Splits elements form one sequence to two another. Start at point ‘start’ and splits elements by step1 and step2.

Template <typename Key, typename Info>

class Sequence{

private:

struct element{

Key key;

Info info;

element* next;

};

element* head;

int nr;

public:

Sequence() – Constructor

Sequence(const Sequence &x) – Copy constructor

~Sequence() – Destructor

bool operator==(const Sequence &x)const – Compares to object by data and order (if equal returns true)

bool operator!=(const Sequence &x)const – Compares two object by data and order (if equal returns false)

Sequence operator+(const Sequence &x)const – It adds to the first sequence unique objects from the other

Sequence &operator=(const Sequence &x) – Copy operator

friend ostream& operator<<(ostream& os, const Sequence &x) – Displays every element ‘key:info’ in separated lines

bool isEmpty()const – Check if object possess any element, if yes returns false.

Bool keyExists(Key key)const – Check if in object is element with given key, if yes returns true

bool infoExists(Info info)const – Check if in object is element with given info, if yes returns true

bool addElement(Key key, Info info) – Adds at the beginning of list element with given data (Only unique key). True if added.

Bool addElement(const element &x – Adds at the beginning of list given element (Only unique key) . True if added.

Bool addAtTheEnd(Key key, Info info) – Adds at the end of list element with given data (Only unique key). True if added.

Bool addAtTheEnd(const element &x) – Adds at the end of list given element (Only unique key) . True if added.

Bool addAfterKey(Key k, Key key, Info info) – Adds after an element with given key new element with given data (Only unique key) . True if added.

Bool addAfterKey(Key k,const element &x) -Adds after an element with given key given element (Only unique key) . True if added.

Bool addVector(const vector<element> &x) – Adds elements contained in the vector at the beginning of the list (Only unique key) . True if added at least one.

Bool addVectorAtTheEnd(const vector<element> &x) – Adds elements contained in the ector at the end of the list (Only unique key) . True if added at least one.

Bool addVectorAfterKey(Key k, const vector<element> &x) – Adds elements contained in the vector after element with given key (Only unique key) . True if added at least one.

Bool getAddVectorAtTheEnd(const Sequence &x,int b,int n) – Adds elements taken from another object contained I the vector at the end of the list (Only unique key) . True if added at least one.

Bool deleteElement(Key key) – Deletes element with given key. True if deleted.

Bool deleteAllByInfo(Info info) – Deletes All elements with given info. True if deleted at least one.

Bool deleteFromKey(Key key, int n) – Deletes at most (if end of the list is closer) n elements beginning from the element with given key. True if deleted at least one.

Vector<Key> getAllKey()const - Returns all keys from the list

vector<Info> getAllInfo()const - Returns all info from the list

vector<Key> getByInfo(Info info)const – Returns all keys of elements that contains given info

vector<element> getAllElements()const – Returns all elements from the list

vector<element> getElementsFromKey(Key key, int n)const – Returns at most n elements in line beginning from key to n-th element after key or to the and of list

vector<element> getElementsFromPlace(int b, int n)const – Returns particular elements, b position of begin in counting from 0, n number of elements to copy

void setNr(int n) – Change the number of elements in the list

int getNr()const – Returns the number of the elements in the list

TESTING CONDUCTED

Split source into two sources when:

- source is not empty

-step1 and step2 are natural numbers lower than number of elements in source

Split source into two sources when:

-source is empty

-step1 and step2 are natural numbers

Split source into two sources when:

-source is not empty

-step1 and step2 are natural numbers lower than number of elements in source

-start is beyond the source

Split source into two sources when:

-source is not empty

-step1 and step2 are natural numbers when at least one is bigger than element number I source

Split source into two sources when:

-source is not empty

-step1 or step2 is lower or equal zero