My Project

Generated by Doxygen 1.9.5

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 BSTNode Class Reference

BSTNode contain data,height and threepoints containing parent,leftchild and rightchild.

`#include <DSA.h>`

Public Member Functions

- BSTNode (ll val)

  Construct a new BSTNode object.

Public Attributes

- ll **info**
- ll **level**
- BSTNode * **left**
- BSTNode * **right**

3.1.1 Detailed Description

BSTNode contain data,height and threepoints containing parent,leftchild and rightchild.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BSTNode()

```
BSTNode::BSTNode (
            ll val )  [inline]
```

Construct a new BSTNode object.

Parameters

	val	Info in Head
out	*level*	Sets level as zero
out	*info*	Sets info as val
out	*left*	Sets left as nullptr
out	*right*	Sets right as nullptr

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.2 DoublyLinkedList Class Reference

This Class implements ADT (DoublyLinkedList) which is chain of DoublyLinkedListNodes which is specified by head and Last Node's next pointer is pointed to NULL and Head Node's previous pointer is pointed to NULL.

`#include <DSA.h>`

Public Member Functions

- DoublyLinkedList ()

  Construct a new Doubly Linked List object.
- void insert (ll data)

  Inserting a Node into the Chain by Adding after the Tail.
- void printer (string sep=", ")

  Function to Print DoublyLinked lists.
- void reverse ()

  Revers The linked List.

Public Attributes

- DoublyLinkedListNode * **head**
- DoublyLinkedListNode * **tail**

3.2.1 Detailed Description

This Class implements ADT (DoublyLinkedList) which is chain of DoublyLinkedListNodes which is specified by head and Last Node's next pointer is pointed to NULL and Head Node's previous pointer is pointed to NULL.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DoublyLinkedList()

`DoublyLinkedList::DoublyLinkedList ( )`

Construct a new Doubly Linked List object.

Parameters

out	*head*	Sets head to nullptr
out	*tail*	Sets tail to nullptr

3.2.3 Member Function Documentation

3.2.3.1 insert()

```
void DoublyLinkedList::insert (
            ll data )
```

Inserting a Node into the Chain by Adding after the Tail.

Parameters

in	*data*	Information of the new Node

Returns

Returns Nothing

3.2.3.2 printer()

```
void DoublyLinkedList::printer (
            string sep = ", " )
```

Function to Print DoublyLinked lists.

Parameters

out	*sep*	Prints the DoublyLinked list (as list in Python)

Returns

Returns Nothing

3.2.3.3 reverse()

```
void DoublyLinkedList::reverse ( )
```

Revers The linked List.

Returns

Returns Nothing

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.3 DoublyLinkedListNode Class Reference

This Class implements ADT (DoublyLinkedListNode) where everynode contains pointers to next Node and previous data, Info in them.

`#include <DSA.h>`

Public Member Functions

- DoublyLinkedListNode ()

  Construct a new Doubly Linked List Node object.
- DoublyLinkedListNode (ll val)

  Construct a new Doubly Linked List Node object.

Public Attributes

- ll **data**
- DoublyLinkedListNode * **next**
- DoublyLinkedListNode * **prev**

3.3.1 Detailed Description

This Class implements ADT (DoublyLinkedListNode) where everynode contains pointers to next Node and previous data, Info in them.

The list of member Functions:

1. insert
2. printer
3. reverse

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DoublyLinkedListNode() [1/2]

`DoublyLinkedListNode::DoublyLinkedListNode ( )`

Construct a new Doubly Linked List Node object.

Parameters

out	*data*	Data is set as -1
out	*next*	Next is set as NULL
out	*prev*	Previous is set as NUll

3.3.2.2 DoublyLinkedListNode() [2/2]

```
DoublyLinkedListNode::DoublyLinkedListNode (
            ll val )
```

Construct a new Doubly Linked List Node object.

Parameters

in	*val*	Sets value of head as Val
out	*next*	Next is set as NULL
out	*prev*	Previous is set as NUll

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.4 SinglyLinkedList Class Reference

This Class implements ADT (SinglyLinkedList) which is chain of SinglyLinkedListNodes which is specified by head and Last Node's pointer is pointed to NULL.

```
#include <DSA.h>
```

Public Member Functions

- SinglyLinkedList ()

  Construct a new Singly Linked List object.
- void insert (ll data)

  Inserting a Node into the Chain by Adding after the Tail.
- SinglyLinkedListNode * find (ll data)

  Searches for a Node with data in the Chain.
- bool deleteVal (ll data)

  Searches For Node with data and Deletes it from chain.
- void printer (string sep=", ")

  Function to Print SinglyLinked lists.
- void reverse ()

  Revers The Singlylinked List.

Public Attributes

- SinglyLinkedListNode * **head**
- SinglyLinkedListNode * **tail**

3.4.1 Detailed Description

This Class implements ADT (SinglyLinkedList) which is chain of SinglyLinkedListNodes which is specified by head and Last Node's pointer is pointed to NULL.

The list of member Functions:

1. insert
2. find
3. deleteVal
4. printer
5. reverse

3.4.2 Constructor & Destructor Documentation

3.4.2.1 SinglyLinkedList()

```
SinglyLinkedList::SinglyLinkedList ( )
```

Construct a new Singly Linked List object.

Parameters

out	*head*	Sets head to Nullptr
out	*tail*	Sets tail to Nullptr

3.4.3 Member Function Documentation

3.4.3.1 deleteVal()

```
bool SinglyLinkedList::deleteVal (
            ll data )
```

Searches For Node with data and Deletes it from chain.

Parameters

in	*data*	Information of Node to be Deleted

Returns

true(if the Node is present in the Chain)

false (If the Node isnt Present in the Chain)

3.4.3.2 find()

```
SinglyLinkedListNode * SinglyLinkedList::find (
            ll data )
```

Searches for a Node with data in the Chain.

Parameters

in	*data*	Information of Node to be Searched
out	*prev*	Node containg data as Information

Returns

returns adress of the node contains the given information

3.4.3.3 insert()

```
void SinglyLinkedList::insert (
            ll data )
```

Inserting a Node into the Chain by Adding after the Tail.

Parameters

in	*data*	Information of the new Node

Returns

Returns Nothing

3.4.3.4 printer()

```
void SinglyLinkedList::printer (
            string sep = ", " )
```

Function to Print SinglyLinked lists.

Parameters

out	*sep*	Prints the SinglyLinked list (as list in Python)

Returns

Returns Nothing

3.4.3.5 reverse()

```
void SinglyLinkedList::reverse ( )
```

Revers The Singlylinked List.

Returns

Returns Nothing

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.5 SinglyLinkedListNode Class Reference

This Class implements ADT (SinglyLinkedListNode) where everynode contains pointer to next Node and data in them.

```
#include <DSA.h>
```

Public Member Functions

- SinglyLinkedListNode ()

  Construct a new Singly Linked List Node object.
- SinglyLinkedListNode (ll val)

  Construct a new Singly Linked List Node object.

Public Attributes

- ll **data**
- SinglyLinkedListNode * **next**

3.5.1 Detailed Description

This Class implements ADT (SinglyLinkedListNode) where everynode contains pointer to next Node and data in them.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 SinglyLinkedListNode() [1/2]

```
SinglyLinkedListNode::SinglyLinkedListNode ( )
```

Construct a new Singly Linked List Node object.

Parameters

out	*data*	Sets data value as -1
out	*next*	Sets next point to NULL

3.5.2.2 SinglyLinkedListNode() [2/2]

```
SinglyLinkedListNode::SinglyLinkedListNode (
            ll val )
```

Construct a new Singly Linked List Node object.

Parameters

in	*val*	Input given to set data
out	*data*	Sets value of data as val
out	*next*	Sets next point to NULL

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

Chapter 4

File Documentation

4.1 DSA.h File Reference

Implementation of Different types of Abstract Data Types.

```
#include <bits/stdc++.h>
```

Classes

- class SinglyLinkedListNode

  This Class implements ADT (SinglyLinkedListNode) where everynode contains pointer to next Node and data in them.
- class SinglyLinkedList

  This Class implements ADT (SinglyLinkedList) which is chain of SinglyLinkedListNodes which is specified by head and Last Node's pointer is pointed to NULL.
- class DoublyLinkedListNode

  This Class implements ADT (DoublyLinkedListNode) where everynode contains pointers to next Node and previous data, Info in them.
- class DoublyLinkedList

  This Class implements ADT (DoublyLinkedList) which is chain of DoublyLinkedListNodes which is specified by head and Last Node's next pointer is pointed to NULL and Head Node's previous pointer is pointed to NULL.
- class BSTNode

  BSTNode contain data,height and threepoints containing parent,leftchild and rightchild.

Macros

- #define **ll** long long int
- #define **vi** vector<int>
- #define **vll** vector<ll>

Functions

- ostream & **operator**<< (ostream &out, const SinglyLinkedListNode &node)
- class SinglyLinkedList **merge** (SinglyLinkedList list1, SinglyLinkedList list2)
- SinglyLinkedList ()

  Construct a new Singly Linked List object.
- void insert (ll data)

  Inserting a Node into the Chain by Adding after the Tail.
- SinglyLinkedListNode * find (ll data)

  Searches for a Node with data in the Chain.
- bool deleteVal (ll data)

  Searches For Node with data and Deletes it from chain.
- void printer (string sep=", ")

  Function to Print SinglyLinked lists.
- void reverse ()

  Revers The Singlylinked List.
- ostream & **operator**<< (ostream &out, const DoublyLinkedListNode &node)
- ostream & **operator**<< (ostream &out, const BSTNode &node)

Variables

- SinglyLinkedListNode * **head**
- SinglyLinkedListNode * **tail**
- class DoublyLinkedListNode **merge**

4.1.1 Detailed Description

Implementation of Different types of Abstract Data Types.

Author

BVSS Prabandh

Version

0.1

Date

2022-09-21

Copyright

Copyright (c) 2022

4.1.2 Function Documentation

4.1.2.1 deleteVal()

```
bool merge::deleteVal (
            ll data )
```

Searches For Node with data and Deletes it from chain.

Parameters

in	*data*	Information of Node to be Deleted

Returns

true(if the Node is present in the Chain)

false (If the Node isnt Present in the Chain)

4.1.2.2 find()

```
SinglyLinkedListNode * merge::find (
            ll data )
```

Searches for a Node with data in the Chain.

Parameters

in	*data*	Information of Node to be Searched
out	*prev*	Node containg data as Information

Returns

returns adress of the node contains the given information

4.1.2.3 insert()

```
void merge::insert (
            ll data )
```

Inserting a Node into the Chain by Adding after the Tail.

Parameters

in	*data*	Information of the new Node

Returns

Returns Nothing

4.1.2.4 printer()

```
void merge::printer (
            string sep = ", " )
```

Function to Print SinglyLinked lists.

Parameters

out	*sep*	Prints the SinglyLinked list (as list in Python)

Returns

Returns Nothing

4.1.2.5 reverse()

```
void merge::reverse ( )
```

Revers The Singlylinked List.

Returns

Returns Nothing

4.1.2.6 SinglyLinkedList()

```
merge::SinglyLinkedList ( )
```

Construct a new Singly Linked List object.

Parameters

out	*head*	Sets head to Nullptr
out	*tail*	Sets tail to Nullptr

4.2 DSA.h

Go to the documentation of this file.

```
1
11 #include <bits/stdc++.h>
12 #define ll long long int
13 #define vi vector<int>
```

```
14 #define vll vector<ll>
15 using namespace std;
16
17 /* ------------------------------ Data Structures ---------------------------------- */
18
19 // ------------------------------ Singly Linked List ----------------------------
25 class SinglyLinkedListNode {
26
27     public:
28
29         ll data;
30         SinglyLinkedListNode* next;
37         SinglyLinkedListNode ();
45         SinglyLinkedListNode (ll val);
46
47 };
48
49 ostream& operator«(ostream &out, const SinglyLinkedListNode &node) {
50     return out « node.data;
51 }
63 class SinglyLinkedList {
64
65     public:
66
67         SinglyLinkedListNode *head, *tail;
73         SinglyLinkedList ();
80         void insert (ll data);
88         SinglyLinkedListNode* find (ll data);
96         bool deleteVal (ll data);
103         void printer (string sep = ", ");
109         void reverse ();}
110 SinglyLinkedList merge (SinglyLinkedList list1, SinglyLinkedList list2);
111
112 // ------------------------------ Doubly Linked List ----------------------------
121 class DoublyLinkedListNode {
122
123     public:
124
125         ll data;
126         DoublyLinkedListNode *next, *prev;
133         DoublyLinkedListNode();
141         DoublyLinkedListNode (ll val);
142
143 };
144
145 ostream& operator«(ostream &out, const DoublyLinkedListNode &node) {
146     return out « node.data;
147 }
153 class DoublyLinkedList {
154
155     public:
156
157         DoublyLinkedListNode *head, *tail;
163         DoublyLinkedList ();
170         void insert (ll data);
177         void printer (string sep = ", ");
183         void reverse ();
184
185 };
186
187 // ------------------------------ Binary Search Tree ----------------------------
192 class BSTNode {
193
194     public:
195
196         ll info, level;
197         BSTNode *left, *right;
207         BSTNode (ll val) {
208             info = val;
209             level = 0;
210             left = NULL;
211             right = NULL;
212         }
213
214 };
215
216 ostream& operator«(ostream &out, const BSTNode &node) {
217     return out « node.info;
218 }
227 class BinarySearchTree {
228
229     public:
230
231         BSTNode *root;
232
233         enum order {PRE, IN, POST};
238         BinarySearchTree () {
```

```
239             root = NULL;
240         }
247         void insert(ll val);
255         void traverse (BSTNode* T, order tt);
262         ll height(BSTNode *T);
263
264 // ------------------------------ Suffix Trie ----------------------------
275 class Trie {
276
277     public:
278
279         ll count;
280         map<char,Trie*> nodes;
286         Trie () {
287             count = 0;
288             nodes = map<char,Trie*>();
289         }
298         bool find(Trie* T, char c);
305         void insert(string s);
313         bool checkPrefix(string s);
320         ll countPrefix(string s);
321
322 };
```

Index