Binary Search Tree

Generato da Doxygen 1.9.1

1 Binary Search Tree

1.1 27/05/2021

1.2 Analisi del problema

Il testo richiede la progettazione e l'implementazione di una o più opportune strutture dati astratte (ADT) che consentano di:

- inserire elementi nella struttura dati;
- cercare elementi nella struttura dati;
- cancellare elementi nella struttura dati.
- stampare gli elementi seguendo una vista PostOrder

- stampare gli elementi seguendo una vista PreOrder
- stampare gli elementi seguendo una vista InOrder
- cercare il massimo nella struttura dati
- cercare il minimo nella struttura dati
- cercare il successore di un elemento
- cercare il predecessore di un elemento

Tra i requisiti, si nota che:

- la ricerca e la cancellazione avvengono in base a dei valori passati come parametri alle rispettive funzioni;
- i dati da inserire sono forniti in un file di testo(non implementato, utilizzo un inserimento manuale per fare dei test);
- la stampa della struttura dati deve essere effettuata seguendo delle viste: inorder, postorder, preorder.

1.3 Considerazioni progettuali

1. La struttura di un Binary Search Tree suggerisce l'utilizzo di una struttura dati che sia "percorribile" in entrambi i versi;
2. Il fatto che la struttura dati venga popolata a partire da un file di testo suggerisce che è opportuno utilizzare un'implementazione dinamica, che consenta quindi di variare il numero di elementi nella struttura stessa;
3. L'inserimento viene effettuato seguendo le propietà dell'ordinamento parziale della struttura, poiche i nodi vanno inseriti a sinistra se la key del nodo da inserire è più piccola della key del nodo in cui siamo arrivati o a destra se più grande, cioè ogni nodo avra a sinistra i figli con la key più piccola e a destra i figli con la key più grande;

Si decide, pertanto, di utilizzare una **Lista** adattata all'esigenze della struttura dati, con un puntatore alla root; e dove gli elementi che la compongono hanno 3 puntatori:

- al parent
- al figlio sinistro
- al figlio destro .

1.4 Classi utilizzate

- **Node**<**T**>: classe template che rappresenta un nodo della lista con figlio sinistro(left) , figlio destro(right) e genitore(parent).
  - Qui vengono implementate le funzioni base di: creazione del figlio, settare figlio destro , figlio sinistro e parent, e restituire i puntatori a left, right e parent oltre all'overload del cout(operator<<) per poter effettuare la stampa dell'ogetto di tipo Node<T>.
- **BST**<**T**>: classe template che rappresenta una lista adattata alla struttura dati di un Binary Search Tree, con un puntatore alla root dell'albero.
  - Qui vengnono implementate le funzioni richieste: le varie viste per la stampa, le funzioni di ricerca e le funzioni di inserimento e cancellazione.

Trattandosi di un BST, le operazioni di inserimento, ricerca e cancellazione in generale avvengono in tempo **O(h)** dove h è l'altezza dell albero, e **O(n)** nel caso pessimo.

2 Indice dei tipi composti

2.1 Elenco dei tipi composti

Queste sono le classi, le struct, le union e le interfacce con una loro breve descrizione:

3 Indice dei file

3.1 Elenco dei file

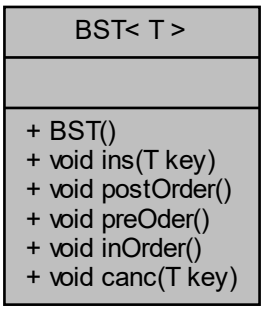
Questo è un elenco di tutti i file con una loro breve descrizione:

4 Documentazione delle classi

4.1 Template per la classe BST< T >

`#include <BST.h>`

Diagramma di collaborazione per BST< T >:



Membri pubblici

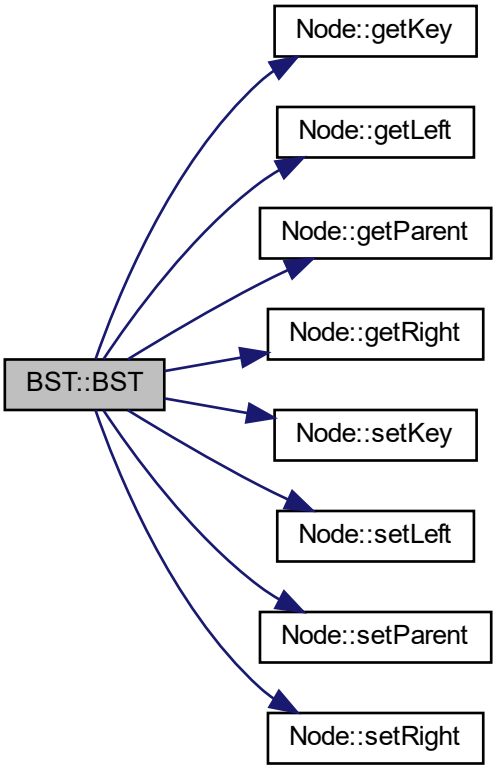
- BST ()
- void ins (T key)
- void postOrder ()
- void preOder ()
- void inOrder ()
- void canc (T key)

4.1.1 Documentazione dei costruttori e dei distruttori

4.1.1.1 BST() `template<class T >`

`BST< T >::BST ( ) [inline]`

Costruttore di defaul Questo è il grafo delle chiamate per questa funzione:

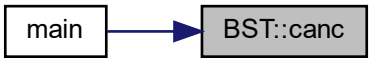


4.1.2 Documentazione delle funzioni membro

4.1.2.1 canc() template<class T >

```
void BST< T >::canc (
            T key )  [inline]
```

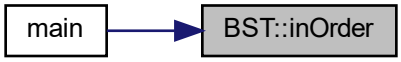
Questo è il grafo dei chiamanti di questa funzione:



4.1.2.2 inOrder() template<class T >

```
void BST< T >::inOrder ( )  [inline]
```

Questo è il grafo dei chiamanti di questa funzione:

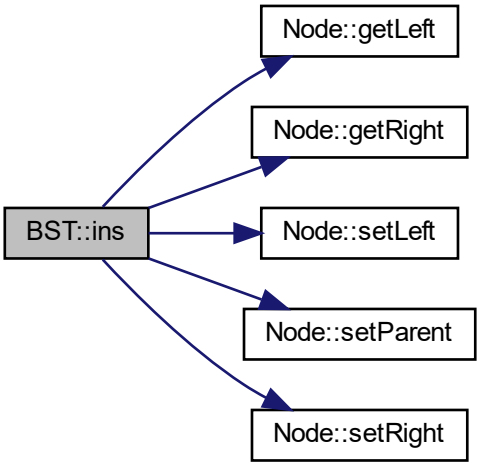


4.1.2.3 ins() template<class T >

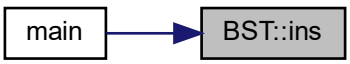
```
void BST< T >::ins (
            T key )  [inline]
```

Funzione per l'inserimento di un nodo tramite passaggio di una chiave di tipo generico T se l'albero oto allora il nodo creato diventer root, nel caso in cui l'albero non sia vuoto, allora si far confronto per vedere se il nuovo nodo

deve essere inserito nel sottoalbero destro o nel sottoalbero sinistro Questo è il grafo delle chiamate per questa funzione:



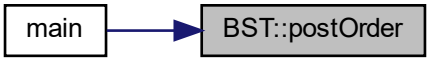
Questo è il grafo dei chiamanti di questa funzione:



4.1.2.4 postOrder() `template<class T >`

`void BST< T >::postOrder ( ) [inline]`

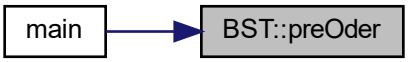
Questo è il grafo dei chiamanti di questa funzione:



4.1.2.5 preOder() template<class T >

void BST< T >::preOder ( ) [inline]

Questo è il grafo dei chiamanti di questa funzione:



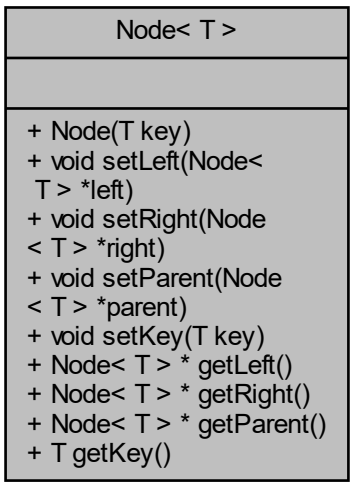
La documentazione per questa classe è stata generata a partire dal seguente file:

- include/BST.h

4.2 Template per la classe Node< T >

#include <Node.h>

Diagramma di collaborazione per Node< T >:



Membri pubblici

- Node (T key)
- void setLeft (Node< T > *left)
- void setRight (Node< T > *right)
- void setParent (Node< T > *parent)
- void setKey (T key)
- Node< T > * getLeft ()
- Node< T > * getRight ()
- Node< T > * getParent ()
- T getKey ()

Friend

- ostream & operator<< (ostream &out, Node< T > &n)

4.2.1 Documentazione dei costruttori e dei distruttori

4.2.1.1 Node() `template<class T >`

```
Node< T >::Node (
            T key ) [inline]
```

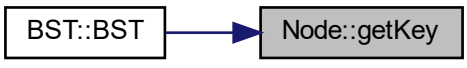
Costruttore che riceve in ingresso una key di tipo generico

4.2.2 Documentazione delle funzioni membro

4.2.2.1 getKey() `template<class T >`

```
T Node< T >::getKey ( ) [inline]
```

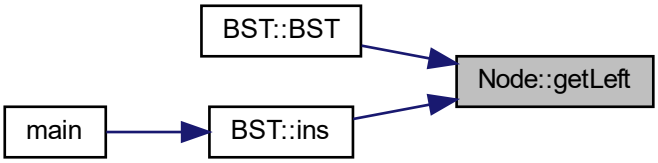
Metodo che ritorna la key Questo è il grafo dei chiamanti di questa funzione:



4.2.2.2 getLeft() template<class T >

Node<T>* Node< T >::getLeft ( ) [inline]

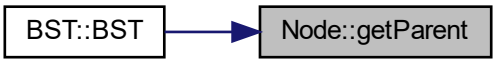
Metodo che ritorna il puntatore al figlio sinistro Questo è il grafo dei chiamanti di questa funzione:



4.2.2.3 getParent() template<class T >

Node<T>* Node< T >::getParent ( ) [inline]

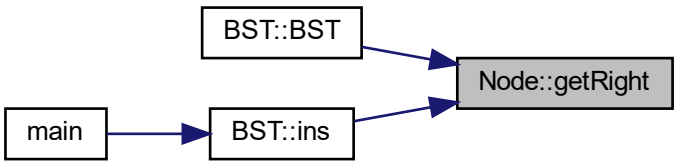
Metodo che ritorna il puntatore al parent Questo è il grafo dei chiamanti di questa funzione:



4.2.2.4 getRight() template<class T >

Node<T>* Node< T >::getRight ( ) [inline]

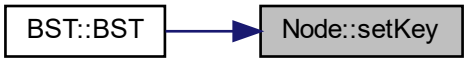
Metodo che ritorna il puntatore al figlio destro Questo è il grafo dei chiamanti di questa funzione:



4.2.2.5 setKey() template<class T >

```
void Node< T >::setKey (
            T key )  [inline]
```

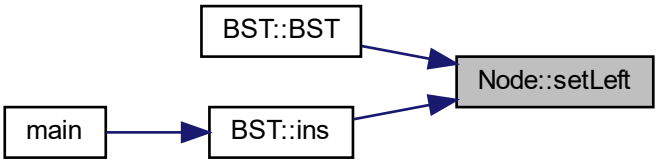
Metodo per impostare la key Questo è il grafo dei chiamanti di questa funzione:



4.2.2.6 setLeft() template<class T >

```
void Node< T >::setLeft (
            Node< T > * left )  [inline]
```

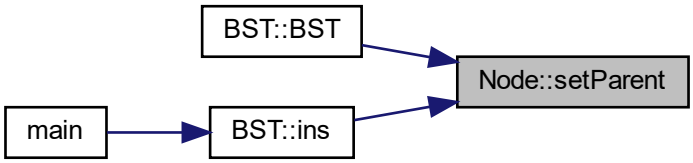
Metodo per impostare il figlio sinistro Questo è il grafo dei chiamanti di questa funzione:



4.2.2.7 setParent() template<class T >

```
void Node< T >::setParent (
            Node< T > * parent )  [inline]
```

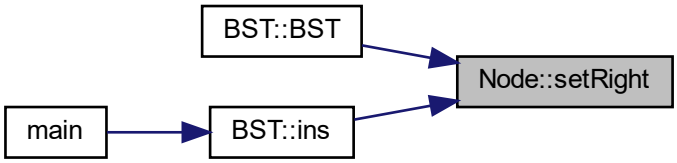
Metodo per impostare il parent Questo è il grafo dei chiamanti di questa funzione:



4.2.2.8 setRight() template<class T >

```
void Node< T >::setRight (
            Node< T > * right )  [inline]
```

Metodo per impostare il figlio detro Questo è il grafo dei chiamanti di questa funzione:



4.2.3 Documentazione dei friend e delle funzioni collegate

4.2.3.1 operator<< template<class T >

```
ostream& operator<< (
            ostream & out,
            Node< T > & n )  [friend]
```

Overriding dell operatore << del cout per stampare un ogetto di tipo Node<T> stampando la key del nodo

La documentazione per questa classe è stata generata a partire dal seguente file:

- include/Node.h

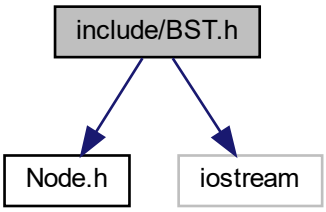
5 Documentazione dei file

5.1 Riferimenti per il file Considerazioni.md

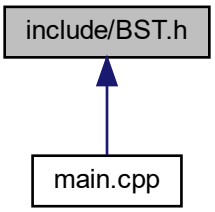
5.2 Riferimenti per il file include/BST.h

```
#include "Node.h"
#include <iostream>
```

Grafo delle dipendenze di inclusione per BST.h:



Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

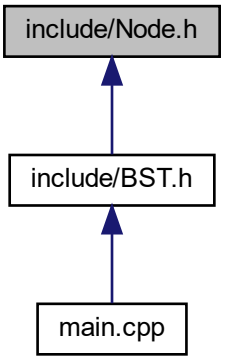


Composti

- class BST< T >

5.3 Riferimenti per il file include/Node.h

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



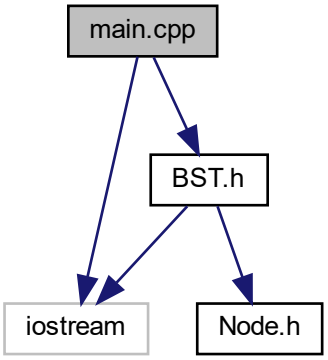
Composti

- class Node< T >

5.4 Riferimenti per il file main.cpp

```
#include <iostream>
#include "BST.h"
```

Grafo delle dipendenze di inclusione per main.cpp:



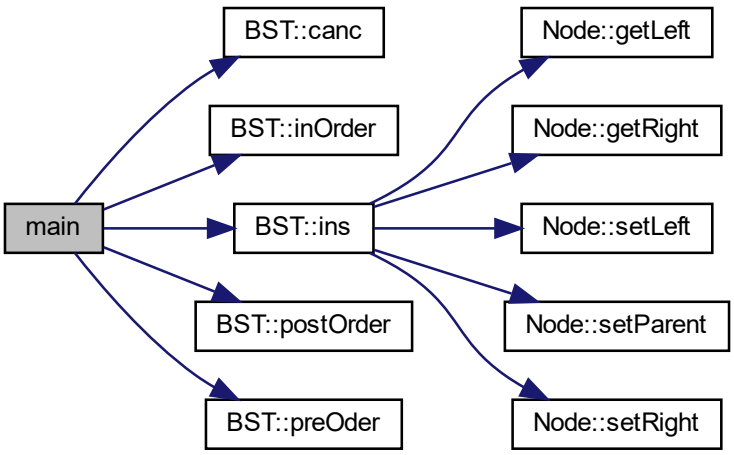
Funzioni

- int main ()

5.4.1 Documentazione delle funzioni

5.4.1.1 main() `int main ( )`

Questo è il grafo delle chiamate per questa funzione:



Indice analitico