[Skip to content](https://gist.github.com/toboqus/def6a6915e4abd66e922%22%20%5Cl%20%22start-of-content)

Top of Form



Bottom of Form

[**All gists**](https://gist.github.com/discover)[**Back to GitHub**](https://github.com/)

[Sign in](https://gist.github.com/auth/github?return_to=https%3A%2F%2Fgist.github.com%2Ftoboqus%2Fdef6a6915e4abd66e922)[Sign up](https://gist.github.com/join?return_to=https%3A%2F%2Fgist.github.com%2Ftoboqus%2Fdef6a6915e4abd66e922&source=header-gist)

Instantly share code, notes, and snippets.

 [toboqus](https://gist.github.com/toboqus)/[**btree.cpp**](https://gist.github.com/toboqus/def6a6915e4abd66e922)

Created 5 years ago

* [**Star**](https://gist.github.com/login?return_to=https%3A%2F%2Fgist.github.com%2Ftoboqus%2Fdef6a6915e4abd66e922)[**43**](https://gist.github.com/toboqus/def6a6915e4abd66e922/stargazers)
* [**Fork**](https://gist.github.com/login?return_to=https%3A%2F%2Fgist.github.com%2Ftoboqus%2Fdef6a6915e4abd66e922)[**20**](https://gist.github.com/toboqus/def6a6915e4abd66e922/forks)

[Code](https://gist.github.com/toboqus/def6a6915e4abd66e922)[Revisions **1**](https://gist.github.com/toboqus/def6a6915e4abd66e922/revisions)[Stars **43**](https://gist.github.com/toboqus/def6a6915e4abd66e922/stargazers)[Forks **20**](https://gist.github.com/toboqus/def6a6915e4abd66e922/forks)

Embed



[**Download ZIP**](https://gist.github.com/toboqus/def6a6915e4abd66e922/archive/73ccc51f5482a1aa07e68310807dddcd32258c3e.zip)

Binary tree implementation in c++

[**Raw**](https://gist.github.com/toboqus/def6a6915e4abd66e922/raw/73ccc51f5482a1aa07e68310807dddcd32258c3e/btree.cpp)

 [**btree.cpp**](https://gist.github.com/toboqus/def6a6915e4abd66e922#file-btree-cpp)

|  |  |
| --- | --- |
|  | #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  |  |
|  | struct node{ |
|  |  int value; |
|  |  node \*left; |
|  |  node \*right; |
|  | }; |
|  |  |
|  | class btree{ |
|  | public: |
|  |  btree(); |
|  |  ~btree(); |
|  |  |
|  |  void insert(int key); |
|  |  node \*search(int key); |
|  |  void destroy\_tree(); |
|  |  void inorder\_print(); |
|  |  void postorder\_print(); |
|  |  void preorder\_print(); |
|  |  |
|  | private: |
|  |  void destroy\_tree(node \*leaf); |
|  |  void insert(int key, node \*leaf); |
|  |  node \*search(int key, node \*leaf); |
|  |  void inorder\_print(node \*leaf); |
|  |  void postorder\_print(node \*leaf); |
|  |  void preorder\_print(node \*leaf); |
|  |  |
|  |  node \*root; |
|  | }; |
|  |  |
|  |  |
|  | btree::btree(){ |
|  |  root = NULL; |
|  | } |
|  |  |
|  | btree::~btree(){ |
|  |  destroy\_tree(); |
|  | } |
|  |  |
|  | void btree::destroy\_tree(node \*leaf){ |
|  |  if(leaf != NULL){ |
|  |  destroy\_tree(leaf->left); |
|  |  destroy\_tree(leaf->right); |
|  |  delete leaf; |
|  |  } |
|  | } |
|  |  |
|  | void btree::insert(int key, node \*leaf){ |
|  |  |
|  |  if(key < leaf->value){ |
|  |  if(leaf->left != NULL){ |
|  |  insert(key, leaf->left); |
|  |  }else{ |
|  |  leaf->left = new node; |
|  |  leaf->left->value = key; |
|  |  leaf->left->left = NULL; |
|  |  leaf->left->right = NULL; |
|  |  } |
|  |  }else if(key >= leaf->value){ |
|  |  if(leaf->right != NULL){ |
|  |  insert(key, leaf->right); |
|  |  }else{ |
|  |  leaf->right = new node; |
|  |  leaf->right->value = key; |
|  |  leaf->right->right = NULL; |
|  |  leaf->right->left = NULL; |
|  |  } |
|  |  } |
|  |  |
|  | } |
|  |  |
|  | void btree::insert(int key){ |
|  |  if(root != NULL){ |
|  |  insert(key, root); |
|  |  }else{ |
|  |  root = new node; |
|  |  root->value = key; |
|  |  root->left = NULL; |
|  |  root->right = NULL; |
|  |  } |
|  | } |
|  |  |
|  | node \*btree::search(int key, node \*leaf){ |
|  |  if(leaf != NULL){ |
|  |  if(key == leaf->value){ |
|  |  return leaf; |
|  |  } |
|  |  if(key < leaf->value){ |
|  |  return search(key, leaf->left); |
|  |  }else{ |
|  |  return search(key, leaf->right); |
|  |  } |
|  |  }else{ |
|  |  return NULL; |
|  |  } |
|  | } |
|  |  |
|  | node \*btree::search(int key){ |
|  |  return search(key, root); |
|  | } |
|  |  |
|  | void btree::destroy\_tree(){ |
|  |  destroy\_tree(root); |
|  | } |
|  |  |
|  | void btree::inorder\_print(){ |
|  |  inorder\_print(root); |
|  |  cout << "\n"; |
|  | } |
|  |  |
|  | void btree::inorder\_print(node \*leaf){ |
|  |  if(leaf != NULL){ |
|  |  inorder\_print(leaf->left); |
|  |  cout << leaf->value << ","; |
|  |  inorder\_print(leaf->right); |
|  |  } |
|  | } |
|  |  |
|  | void btree::postorder\_print(){ |
|  |  postorder\_print(root); |
|  |  cout << "\n"; |
|  | } |
|  |  |
|  | void btree::postorder\_print(node \*leaf){ |
|  |  if(leaf != NULL){ |
|  |  inorder\_print(leaf->left); |
|  |  inorder\_print(leaf->right); |
|  |  cout << leaf->value << ","; |
|  |  } |
|  | } |
|  |  |
|  | void btree::preorder\_print(){ |
|  |  preorder\_print(root); |
|  |  cout << "\n"; |
|  | } |
|  |  |
|  | void btree::preorder\_print(node \*leaf){ |
|  |  if(leaf != NULL){ |
|  |  cout << leaf->value << ","; |
|  |  inorder\_print(leaf->left); |
|  |  inorder\_print(leaf->right); |
|  |  } |
|  | } |
|  |  |
|  | int main(){ |
|  |  |
|  |  //btree tree; |
|  |  btree \*tree = new btree(); |
|  |  |
|  |  tree->insert(10); |
|  |  tree->insert(6); |
|  |  tree->insert(14); |
|  |  tree->insert(5); |
|  |  tree->insert(8); |
|  |  tree->insert(11); |
|  |  tree->insert(18); |
|  |  |
|  |  tree->preorder\_print(); |
|  |  tree->inorder\_print(); |
|  |  tree->postorder\_print(); |
|  |  |
|  |  delete tree; |
|  |  |
|  | } |

|  |
| --- |
|  |