|  |  |
| --- | --- |
| **Trie in C++** | |
| #include <iostream>  #include <string>  using namespace std;  class Trie {  private:  struct TrieNode {  char data;  bool isTerminating;  TrieNode\* children[26];  TrieNode(char data) {  this->data = data;  isTerminating = false;  for (int i = 0; i < 26; i++) {  children[i] = nullptr;  }  }  };  TrieNode\* root;  public:  Trie() {  root = new TrieNode('\0');  }  bool search(string word) {  return search(root, word);  }  void add(string word) {  add(root, word);  }  private:  bool search(TrieNode\* root, string word) {  if (word.length() == 0) {  return root->isTerminating;  }  int childIndex = word[0] - 'a';  TrieNode\* child = root->children[childIndex];  if (child == nullptr) {  return false;  }  return search(child, word.substr(1));  }  void add(TrieNode\* root, string word) {  if (word.length() == 0) {  root->isTerminating = true;  return;  }  int childIndex = word[0] - 'a';  if (root->children[childIndex] == nullptr) {  root->children[childIndex] = new TrieNode(word[0]);  }  add(root->children[childIndex], word.substr(1));  }  };  int main() {  Trie t;  t.add("this");  t.add("news");  cout << boolalpha; // Print bool values as "true" or "false"  cout << t.search("news") << endl; // Output: true  cout << t.search("test") << endl; // Output: false  return 0;  } | Dry Run (Step-by-Step)🌱 Step 1: Trie Initialization  * A root TrieNode is created with data = '\0', and all children set to nullptr.  📝 Step 2: Adding "this" Word: "this" Characters processed in order: 't' → 'h' → 'i' → 's'   | **Step** | **Char** | **Index** | **Action** | | --- | --- | --- | --- | | 1 | 't' | 19 | root->children[19] is nullptr, so create new TrieNode('t') | | 2 | 'h' | 7 | Create new TrieNode('h') as child of 't' | | 3 | 'i' | 8 | Create new TrieNode('i') as child of 'h' | | 4 | 's' | 18 | Create new TrieNode('s') as child of 'i', mark isTerminating = true |   ✅ "this" added to trie. 📝 Step 3: Adding "news" Word: "news" Characters: 'n' → 'e' → 'w' → 's'   | **Step** | **Char** | **Index** | **Action** | | --- | --- | --- | --- | | 1 | 'n' | 13 | root->children[13] is nullptr, create TrieNode('n') | | 2 | 'e' | 4 | Create TrieNode('e') under 'n' | | 3 | 'w' | 22 | Create TrieNode('w') under 'e' | | 4 | 's' | 18 | Create TrieNode('s') under 'w', mark isTerminating = true |   ✅ "news" added to trie. 🔍 Step 4: Searching "news" Traversal: 'n' → 'e' → 'w' → 's'   * All nodes exist and 's' has isTerminating = true.   ✅ Output: true 🔍 Step 5: Searching "test" Traversal: 't' → 'e'   * 't' exists (from "this") * 'e' does **not** exist under 't' → return false   ❌ Output: false ✅ Final Output true  false |
| true  false | |