|  |  |
| --- | --- |
| 姓名 | 单禹嘉 |
| 学号 | 2023215177 |

|  |  |
| --- | --- |
| 实验成绩 |  |

华中师范大学计算机学院

实验报告书

课程名称：数据结构

主讲教师：沈显君

课程编号：

班级：2306

1. 实验目的

目的：

利用二叉树和树的生成及遍历模拟因特网域名查询设计一个能够反映域名结构的树，定义并实现因特网域名的存储。通过树的遍历操作实现因特网域名查询。要求从文件中读入因特网域名信息，创建域名树；对系统中所有的域名可以通过遍历进行显示

1. 实验要求

1）正确实现域名树；

2）实现树的遍历、显示、添加、修改等操

1. 作；实验环境

VSCode

1. 附录（源代码）与说明

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <string>

#include <algorithm>

#include <limits>

using namespace std;

// 二叉树节点

struct DomainNode {

string domain;

DomainNode\* left;

DomainNode\* right;

DomainNode(string domain) : domain(domain), left(nullptr), right(nullptr) {}

};

// 域名树类

class DomainTree {

private:

DomainNode\* root;

// 递归插入节点

DomainNode\* insert(DomainNode\* node, const string& domain) {

if (!node) return new DomainNode(domain);

if (domain < node->domain)

node->left = insert(node->left, domain);

else if (domain > node->domain)

node->right = insert(node->right, domain);

return node;

}

// 递归查询节点

DomainNode\* search(DomainNode\* node, const string& domain) {

if (!node || node->domain == domain) return node;

if (domain < node->domain)

return search(node->left, domain);

else

return search(node->right, domain);

}

// 递归中序遍历

void inorderTraversal(DomainNode\* node, vector<string>& result) {

if (!node) return;

inorderTraversal(node->left, result);

result.push\_back(node->domain);

inorderTraversal(node->right, result);

}

// 递归删除节点

DomainNode\* remove(DomainNode\* node, const string& domain) {

if (!node) return nullptr;

if (domain < node->domain) {

node->left = remove(node->left, domain);

} else if (domain > node->domain) {

node->right = remove(node->right, domain);

} else {

if (!node->left) {

DomainNode\* temp = node->right;

delete node;

return temp;

} else if (!node->right) {

DomainNode\* temp = node->left;

delete node;

return temp;

} else {

DomainNode\* temp = findMin(node->right);

node->domain = temp->domain;

node->right = remove(node->right, temp->domain);

}

}

return node;

}

// 找到最小值节点

DomainNode\* findMin(DomainNode\* node) {

while (node && node->left) node = node->left;

return node;

}

// 清空树

void clear(DomainNode\* node) {

if (!node) return;

clear(node->left);

clear(node->right);

delete node;

}

public:

DomainTree() : root(nullptr) {}

~DomainTree() { clear(root); }

// 插入域名

void insert(const string& domain) { root = insert(root, domain); }

// 查询域名

bool search(const string& domain) { return search(root, domain) != nullptr; }

// 删除域名

void remove(const string& domain) { root = remove(root, domain); }

// 获取所有域名（中序遍历）

vector<string> getAllDomains() {

vector<string> result;

inorderTraversal(root, result);

return result;

}

};

// 文件操作函数

void loadDomainsFromFile(const string& filename, DomainTree& tree) {

ifstream file(filename);

if (!file.is\_open()) {

cout << "无法打开文件 " << filename << "。\n";

return;

}

string line;

while (getline(file, line)) {

tree.insert(line);

}

file.close();

cout << "域名数据加载完成。\n";

}

void saveDomainsToFile(const string& filename, DomainTree& tree) {

ofstream file(filename);

if (!file.is\_open()) {

cout << "无法保存到文件 " << filename << "。\n";

return;

}

vector<string> domains = tree.getAllDomains();

for (const string& domain : domains) {

file << domain << endl;

}

file.close();

cout << "域名数据已保存到 " << filename << "。\n";

}

// 获取用户选择的有效数字

int getValidInput(int min, int max) {

int choice;

while (true) {

cin >> choice;

if (cin.fail() || choice < min || choice > max) {

cout << "输入无效，请输入一个介于 " << min << " 和 " << max << " 之间的数字: ";

cin.clear(); // 清除错误状态

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // 丢弃无效输入

} else {

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // 丢弃多余输入

return choice;

}

}

}

// 主程序

int main() {

DomainTree tree;

string filename = "domains.txt";

// 加载域名数据

loadDomainsFromFile(filename, tree);

int choice;

do {

cout << "\n=== 域名管理系统 ===\n";

cout << "1. 显示所有域名\n";

cout << "2. 查询域名\n";

cout << "3. 添加域名\n";

cout << "4. 删除域名\n";

cout << "5. 保存域名到文件\n";

cout << "6. 退出\n";

cout << "请输入您的选择: ";

// 获取有效的用户输入

choice = getValidInput(1, 6);

switch (choice) {

case 1: {

vector<string> domains = tree.getAllDomains();

cout << "\n所有域名:\n";

for (const string& domain : domains) {

cout << domain << endl;

}

break;

}

case 2: {

string domain;

cout << "请输入要查询的域名: ";

cin >> domain;

if (tree.search(domain)) {

cout << "域名 " << domain << " 存在。\n";

} else {

cout << "域名 " << domain << " 不存在。\n";

}

break;

}

case 3: {

string domain;

cout << "请输入要添加的域名: ";

cin >> domain;

tree.insert(domain);

cout << "域名 " << domain << " 已添加。\n";

break;

}

case 4: {

string domain;

cout << "请输入要删除的域名: ";

cin >> domain;

tree.remove(domain);

cout << "域名 " << domain << " 已删除。\n";

break;

}

case 5: {

saveDomainsToFile(filename, tree);

break;

}

case 6:

cout << "退出系统。\n";

break;

}

} while (choice != 6);

return 0;

}