2022113573 张宇杰

建立动态二叉树，并显示

文本

描述已自动生成

文本

描述已自动生成

将拓展先序遍历序列输出至文件fstream即为保存

将输入流stringstream改为fstream即为从文件中加载

建立静态二叉树，并显示

文本

描述已自动生成 文本

描述已自动生成

采用二叉树的上述二叉链表存储结构，编写程序实现二叉树的先序、中序和后序遍历的递归和非递归算法以及层序遍历算法，并以适当的形式显示和保存二叉树及其相应的遍历序列

1. #include "DynamicBinaryTree.hpp"
2. #include "StaticBinaryTree.hpp"
3. #include <sstream>
4. #include <iostream>
5. #include <string>
6. using namespace std;
7. using namespace BinaryTree;
8. int main(int argc, const char \*argv[])
9. {
10. system("chcp 65001");
11. stringstream ss("ABDH##I##E##CF#J##G##");
12. DynamicBinaryTree<char> tree;
13. tree.create(ss, '\0', '#');
14. tree.show(cout);
15. tree.for\_each(PRE\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
16. cout << endl;
17. tree.for\_each\_no\_rec(PRE\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
18. cout << '\n' << endl;
19. tree.for\_each(IN\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
20. cout << endl;
21. tree.for\_each\_no\_rec(IN\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
22. cout << '\n' << endl;
23. tree.for\_each(POST\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
24. cout << endl;
25. tree.for\_each\_no\_rec(POST\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
26. cout << '\n' << endl;
27. tree.for\_each(LEVER\_ORDER, [](char ch){cout << ch << ' ';}, [](){cout << "# ";});
28. cout << endl;
29. return 0;
30. }

文本

中度可信度描述已自动生成

将拓展先序遍历序列输出至文件fstream即为保存

将输入流stringstream改为fstream即为从文件中加载

注：for\_each中的形参使用的是lambda表达式

设计并实现判断任意一棵二叉树是否为完全二叉树的算法

1. #include "DynamicBinaryTree.hpp"
2. #include "StaticBinaryTree.hpp"
3. #include <sstream>
4. #include <iostream>
5. #include <string>
6. using namespace std;
7. using namespace BinaryTree;
8. int main(int argc, const char \*argv[])
9. {
10. system("chcp 65001");
11. stringstream ss("ABDH##I##E##CF#J##G##");
12. DynamicBinaryTree<char> tree1, tree2;
13. tree1.create(ss, '\0', '#');
14. ss.str("ABDH##I##E##CF##G##");
15. tree2.create(ss, '\0', '#');
16. tree1.show(cout);
17. cout << "tree1.complete() = " << tree1.complete() << endl;
18. tree2.show(cout);
19. cout << "tree2.complete() = " << tree2.complete() << endl;
20. return 0;
21. }

文本

描述已自动生成

设计并实现计算任意一棵二叉树的宽度的非递归算法

1. #include "DynamicBinaryTree.hpp"
2. #include "StaticBinaryTree.hpp"
3. #include <sstream>
4. #include <iostream>
5. #include <string>
6. using namespace std;
7. using namespace BinaryTree;
8. int main(int argc, const char \*argv[])
9. {
10. system("chcp 65001");
11. stringstream ss("ABDH##I##E##CF#J###");
12. DynamicBinaryTree<char> tree1, tree2;
13. tree1.create(ss, '\0', '#');
14. ss.str("ABDH##I##E##CF##G##");
15. tree2.create(ss, '\0', '#');
16. tree1.show(cout);
17. cout << "tree1.width() = " << tree1.width() << endl;
18. tree2.show(cout);
19. cout << "tree2.width() = " << tree2.width() << endl;
20. return 0;
21. }

文本

描述已自动生成