|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
| Практическое задание № 3 | | | |
| по дисциплине «Структуры данных и алгоритмы» | | | |
| **«Древовидные структуры данных»** | | | |
|  | | | |
|  | Группа | ПМ-21 | |
| Бригада 7 | Егупов иван | |
|  | Щербакова Мария | |
| Сагыдыев Данияр | |
|  | |
|  | |
| Преподаватель | ХИЦЕНКО ВАЛЕНТИНА ПАВЛОВНА | |
| Дата | 13.04. | |
| Новосибирск,2023 | | | |

1. **Условие задачи**

В заданном бинарном дереве найти первое вхождение заданного элемента и напечатать пройденные при поиске узлы дерева при обратном обходе дерева.

1. **Анализ задачи**

Входные данные: S ∈ Дерево

<Дерево>::= <элемент> | <Левое поддерево> | <Правое поддерево> | <Левое поддерево> <Правое поддерево>

<Левое поддерево>::= <элемент> | <Левое поддерево> <пусто> | <пусто> <правое поддерево> | <Левое поддерево> <Правое поддерево>|<пусто>|<Левое поддерево><элемент><Правое поддерево>

<Правое поддерево >::= <элемент> | <Левое поддерево> <пусто> | <пусто> <правое поддерево> | <Левое поддерево> <Правое поддерево>|<пусто> | <Левое поддерево><элемент><Правое поддерево>

<Элемент> ∈ G ; x ∈ G

G = {ABCDEFGIJKLMNOPRSTFHVQWXYZ}

Выходные данные: K = {ai| ai ∈ G, i≥1} или ∅

q ∈ {0;1}

Решение:

**При** p = 1-й элемент из S при обратном обходе, K = ∅, q = 0

**Повторять** записать p в K, p = очередной элемент из S при обратном обходе

Еслиp=x, то q=1

**Пока** p≠x И ∃p

Подзадачи:

*Buildtree* – размещение бинарного дерева в оперативной памяти

Входные данные: информация о файле, содержащем бинарное дерево, представленное скобочной записью

Выходные данные: бинарное дерево, размещенное в динамической памяти

*Inorder –* обработка дерева путем обратного обхода. Поиск заданного узла и запись пройденных при обходе узлов дерева

Входные данные: Бинарное дерево, искомый узел

1. **Структуры данных**

Внешнее представление   
Входные данные: Бинарное дерево представлено по форму <Дерево>

<Дерево>::=(<корень>,<левое поддерево>,<правое поддерево>)| (<корень>,<левое поддерево>,0)|(<корень>,0,<правое поддерево>)|(<корень>,0,0)

<Левое поддерево> и <Правое поддерево> заданы аналогично

<корень> ∈ { ABCDEFGIJKLMNOPRSTFHVQWXYZ }

Внутреннее представление: Бинарное дерево в динамической памяти

Внутреннее представление элемента дерева представлено в виде структуры

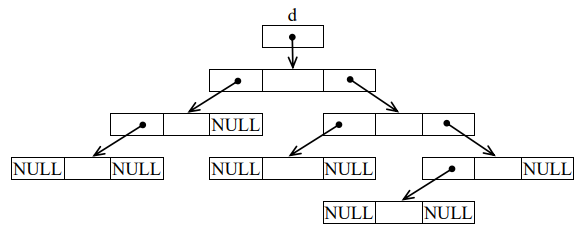
struct tree

{

tree\* left, \* right;

char elem;

}



1. **Алгоритм решения задачи**



Ввод x, S

g = 1-й элемент из S

K = ∅,q =0

P = ∅

q = 0 И g существует?

q = 1?

**Нет да**

g = достать из P

**нет нет**

да

Вывод K,q

Вывод q

g существует?

Стек P пуст?

да

нет

да

Положить g в P

g = левое поддерево

g = достать из P

Добавить g в K

q = 1

g = x ?

да

нет

Добавить g в K

Правое поддерево существует?

да

g = правое поддерево

нет

5.Структура программы

tree\* BuildTree(FILE\* f) Назначение: размещение бинарного дерева в динамической оперативной памяти

FILE \*f – Указатель на файл, в котором записано бинарное дерево в скобочной записи

Выходные данные: возвращает указатель на корень дерева

int Inorder(tree\* root,char x) Назначение: Поиск заданного элемента в дереве, в случае нахождения, вывод пройденных узлов дерева при обратном обходе

tree \*root – указатель на корень дерева, char x – заданный элемент

Выходные данные: q ∈ {0;1}

6.Текст программы

Файл заголовка “Stack.h”

#pragma once

#include <stdio.h>

#define n 100

struct tree

{

tree\* left, \* right;

char elem;

};

struct stack

{

int stack\_size;

tree\* data[n];

};

void createStack(stack\* s)

{

s->stack\_size = 0;

}

void push(tree\* ch, stack\* stk)

{

stk->data[stk->stack\_size] = ch;

stk->stack\_size++;

}

tree\* pop(stack\* stk)

{

tree\* ch = stk->data[stk->stack\_size - 1];

stk->stack\_size--;

return ch;

}

int isempty(stack\* stk) {

if (stk->stack\_size == 0) return 1;

else return 0;

}

Основная программа Source.cpp

#include <stdio.h>

#include "Stack.h"

#define n 100

tree\* BuildTree(FILE\* f)

{

char c;

fscanf\_s(f, "%c", &c,1);

if (c == '(') {

tree\* t = new tree;

fscanf\_s(f, "%c", &c,1);

t->elem = c;

t->left = BuildTree(f);

t->right = BuildTree(f);

fscanf\_s(f,"%c",&c,1);

return t;

}

else if (c == '0') {

return nullptr;

}

else {

return BuildTree(f);

}

}

int Inorder(tree\* root, char x)

{

stack ps;

createStack(&ps);

char str[100];

int i = 0, k = 0;

FILE\* write;

int flag = 1;

while (flag == 1 && k == 0)

{

while (root != NULL)

{

push(root, &ps);

root = root->left;

}

if (isempty(&ps) == 0)

{

root = pop(&ps);

if (root->elem == x) { str[i] = x; k = 1; }

else

{

str[i] = root->elem;i++;

if (root->right != NULL) root = root->right;

else root = NULL;

}

}

else flag = 0;

}

if (k == 1)

{

str[i + 1] = '\0';

fopen\_s(&write, "output.txt", "w");

fprintf\_s(write, "%s", str);

fclose(write);

return 1;

}

else return 0;

}

int main()

{

FILE\* read;

fopen\_s(&read, "input.txt", "r");

tree\* root = BuildTree(read);

fclose(read);

char x;

scanf\_s("%c", &x);

int q = Inorder(root,x);

printf\_s("%d", q);

return 1;

}

7.Набор тестов

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Назначение |
| S = (A,(B,(D,0,0),0),(C,(E,0,(G,0,0)),(F,(H,0,0),(J,0,0))))  x = M | q = 0 | Бинарное дерево с правой и левой ветвью, заданный элемент не был найден |
| S = (A,(B,(D,0,0),0),(C,(E,0,(G,0,0)),(F,(H,0,0),(J,0,0))))  x = H | K = DBAEGCH  q = 1 | Это же бинарное дерево, в котором был найден элемент |
| S = (A,(B,(F,(G,0,0),0),0),(C,0,0))  x = G | K = G  q = 1 | Заданный элемент был первым при обходе |
| S = (A,0,(B,0,(C,0,(D,0,0))))  x = D | K = ABCD  q = 1 | Бинарное дерево, которое состоит только из правого ветвления |
| S = (A,(B,(C,(D,0,0),0),0),0)  x = C | K = DC  q = 1 | Бинарное дерево, состоящее только из левого ветвления |
| S = (A,(B,(F,(G,0,0),0),0),(C,0,0))  x = C | K = GFBAC  q = 1 | Заданный элемент был последним при обходе |