**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе №3**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Деревья

Студенты гр. 5307 Ахмадов Н.М, Самоуков Н.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колинько П.Г.

Санкт-Петербург

2016

Оглавление

[1. ЗАДАНИЕ 3](#_Toc462647558)

[2. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА С ИСПОЛЬЗОВАНИЕМ КЛАССОВ 3](#_Toc462647559)

[3. РЕЗУЛЬТАТ ЭКСПЕРИМЕНТА С ОТСЛЕЖИВАЕНИЕМ ВЫЗОВА ФУНКЦИЙ-ЧЛЕНОВ 3](#_Toc462647560)

[4. ВЫВОД 3](#_Toc462647561)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 3](#_Toc462647562)

[6. ПРИЛОЖЕНИЕ 3](#_Toc462647563)

1. ЗАДАНИЕ

Существует двоичное дерево. Требуется сделать прямую разметку и обойти данное дерево в ширину. Также надо вычислить общее количество вершин.

2. ОБОСНОВАНИЕ ВЫБОРА СПОСОБА ПРЕДСТАВЛЕНИЯ

Исходя из того, что дерево имеет всего лишь два сына мы считаем, что следует использовать представление в виде элемента и двух ссылок на левого и правого сына соответственно. Представление в виде таблицы смежности является излишним для деревьев, а в виде множества вершин и ребер даёт трудности при обработке.

3. ТЕСТОВЫЙ ПРИМЕР

Результат работы программы при вводе дерева с клавиатуры изображен на рисунке 1.

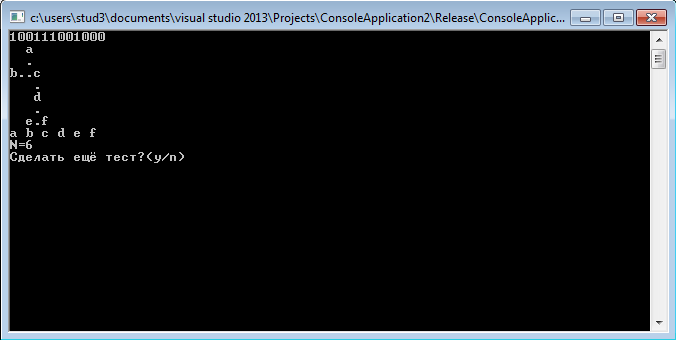


Рисунок 1. Результат работы программы

4. РЕЗУЛЬТАТ ПРОГРАММЫ

Результат работы программы изображен на рисунке 2.

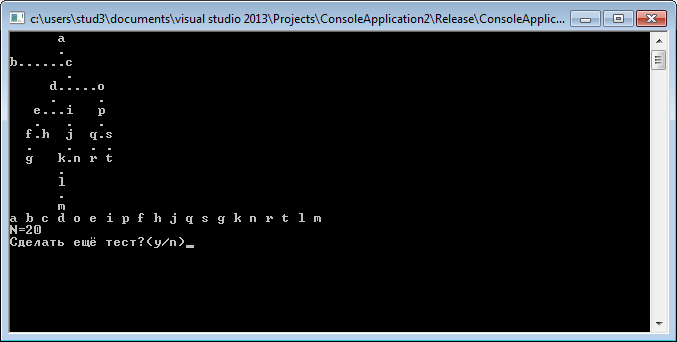


Рисунок 2. Результат работы программы

5. ОЦЕНКА ВРЕМЕННОЙ СЛОЖНОСТИ

Для функции обхода дерево сложность о(n).

Для функции создания сложность о(n).

Для функции обработки сложность о(n).

Для функции вывода сложность о(nlogn).

6. ВЫВОД

Для большинства алгоритмов работы с деревьями выгодно использовать рекурсивные функции.

7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Колинько П. Г. Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Ч. 1. – СПб.: Изд-во СПбГЭТУ <<ЛЭТИ>>, 2016. – 63 с.

8. ПРИЛОЖЕНИЕ

Файл: code.cpp – Исходный код.

#include <windows.h>

#include <conio.h>

#include <time.h>

#include <iostream>

#include <queue>

using namespace std;

class NODE {

char d; //тег узла

NODE \* pl; // левый сын

NODE \* pr; // правый сын

public:

NODE() : pl(nullptr), pr(nullptr), d(rand() % 20) {} // конструктор узла

~NODE()

{

if (pl) delete pl; // деструктор (уничтожает поддерево)

if (pr) delete pr;

}

friend class TREE;

friend int put\_node(NODE \* n, char \*\*s, int x, int y);

friend char mark\_pr(NODE \* n, char ch);

friend char obh\_sh(NODE \* n);

};

class TREE

{

int N; // узлов без сыновей

int depth; // максимальная глубина

NODE \* root; // указатель на вершину дерева

public:

void InTREE(char\*); // создание дерева

void GenTREE(int max\_depth); // создание дерева

void mark();

int proc();

void OutTREE(); // выдача на экран

};

char num\_to\_ch(char num)

{

if (num + 'a' <= 'z')

return num + 'a';

else

return num + 'a' - 'z' - 1 + 'A';

}

char obh\_sh(NODE \* n)

{

if (n == nullptr)

return 0;

queue < NODE\* > q;

int N = 1;

q.push(n);

while (!q.empty())

{

n = q.front();

if (n->pl)

{

N++;

q.push(n->pl);

}

if (n->pr)

{

N++;

q.push(n->pr);

}

cout << num\_to\_ch(n->d) << ' ';

q.pop();

}

return N;

}

char mark\_pr(NODE \* n, char ch = 0)

{

if (n == nullptr)

return ch;

n->d = ch;

return mark\_pr(n->pr, mark\_pr(n->pl, ch + 1));

}

void TREE::mark()

{

mark\_pr(root);

}

int TREE::proc()

{

return obh\_sh(root);

}

int put\_node(NODE \* n, char \*\*s, int x = 0, int y = 0)

{

if (n == nullptr)

return 1;

int ras, ras1, ras2;

ras1 = put\_node(n->pl, s, x, y + 2);

ras = ras1;

if (n->pl == nullptr || n->pr == nullptr)

ras -= 2;

ras2 = put\_node(n->pr, s, x + ras + 1, y + 2);

ras = ras + 1 + ras2;

s[y][x + ras / 2] = num\_to\_ch(n->d);

if (n->pl != nullptr || n->pr != nullptr)

s[y + 1][x + ras / 2] = '.';

if (n->pl != nullptr && n->pr != nullptr)

{

for (int i = x + ras1 / 2 + 1; i <= x + ras1 + 1 + ras2 / 2 - 1; i++)

s[y + 2][i] = '.';

}

return ras;

}

void TREE::OutTREE()

{

char \*s;

char \*\*ss;

s = new char[N \* 2 \* (depth \* 2 + 1) + 1];

ss = new char\*[depth \* 2 + 3];

for (int i = 0; i < N \* 2 \* (depth \* 2 + 1); i++)

s[i] = ' ';

for (int i = 0; i < depth \* 2 + 1; i++)

{

s[(i + 1)\*N \* 2 - 1] = '\n';

ss[i] = s + i\*N \* 2;

}

s[N \* 2 \* (depth \* 2 + 1)] = 0;

put\_node(root, ss);

cout << s;

delete[]ss;

delete[]s;

}

void TREE::InTREE(char \*s)

{

N = 0;

depth = 0;

NODE \*\* arr\_n = new NODE\*[100 + 1];

int \* arr\_s = new int[100 + 1];//0-left 1-right

int l = 0;

bool b = 0;

root = new NODE();

arr\_n[0] = root;

arr\_s[0] = 0;

for (; b == 0;)

{

if (\*s == 0)

return;

if (\*(s++) == '1')

{

l++;

if (l>depth)depth = l;

if (arr\_s[l - 1] == 0)

{

arr\_n[l] = new NODE();

arr\_n[l - 1]->pl = arr\_n[l];

arr\_s[l] = 0;

}

else

{

arr\_n[l] = new NODE();

arr\_n[l - 1]->pr = arr\_n[l];

arr\_s[l] = 0;

}

}

else

{

if (arr\_s[l] == 0)

{

arr\_s[l] = 1;

}

else

{

if (arr\_n[l]->pl == nullptr)

{

N++;

}

while (l >= 0 && arr\_s[l] == 1)

l--;

if (l < 0)

{

b = 1;

}

else

{

arr\_s[l] = 1;

}

}

}

}

delete[]arr\_n;

delete[]arr\_s;

}

void TREE::GenTREE(int n)

{

N = 0;

depth = 0;

if (n < 0)return;

if (n == 0)

root = new NODE();

NODE \*\* arr\_n = new NODE\*[n + 1];

int \* arr\_s = new int[n + 1];//0-left 1-right

int l = 0;

bool b = 0;

root = new NODE();

arr\_n[0] = root;

arr\_s[0] = 0;

for (; b == 0;)

{

if (rand() % n < n - l)

{

l++;

if (l>depth)depth = l;

if (arr\_s[l - 1] == 0)

{

arr\_n[l] = new NODE();

arr\_n[l - 1]->pl = arr\_n[l];

arr\_s[l] = 0;

}

else

{

arr\_n[l] = new NODE();

arr\_n[l - 1]->pr = arr\_n[l];

arr\_s[l] = 0;

}

}

else

{

if (arr\_s[l] == 0)

{

arr\_s[l] = 1;

}

else

{

if (arr\_n[l]->pl == nullptr)

{

N++;

}

while (l >= 0 && arr\_s[l] == 1)

l--;

if (l < 0)

{

b = 1;

}

else

{

arr\_s[l] = 1;

}

}

}

}

delete[]arr\_n;

delete[]arr\_s;

}

int main()

{

setlocale(0, "RU");

srand(time(nullptr));

TREE l;

char ch;

char buf[70];

do{

system("cls");

l.GenTREE(7);

//gets\_s(buf, 70);

//l.InTREE(buf);

l.mark();

l.OutTREE();

int n = l.proc();

cout << "\nN=" << n << "\n";

cout << "Сделать ещё тест?(y/n)";

do

{

ch = getch();

} while (ch != 'y' && ch != 'Y' &&ch != 'n' &&ch != 'N');

} while (ch == 'y' || ch == 'Y');

}