Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчёт

по лабораторной работе № 2

на тему:

“Деревья”

по дисциплине “Алгоритмы и структуры данных”

Вариант 19

Выполнили студенты гр. 4306:

Табаков А. В.,

Сыромятников М. А.

Принял: Колинько П. Г.

Санкт-Петербург  
2015

**Цель**

Получить практические навыки работы с деревьями и узнать его характеристики.

**Задание**

Задание представлено в таблице 1*.*

Таблица. 1. Задание

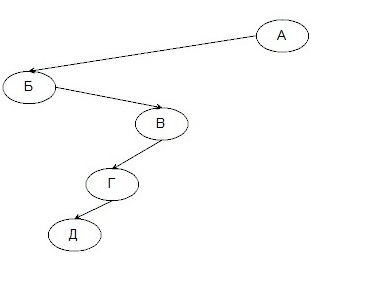
|  |  |  |  |
| --- | --- | --- | --- |
| **Вид дерева** | **Разметка** | **Способ обхода** | **Что надо вычислить** |
| Троичное | Глубинная | Внутренний | Количество вершин не на самом нижнем уровне |

**Способ представления списка в памяти ЭВМ**

Список с тремя указателями (левый узел, средний узел, правый узел).

**Контрольные примеры**

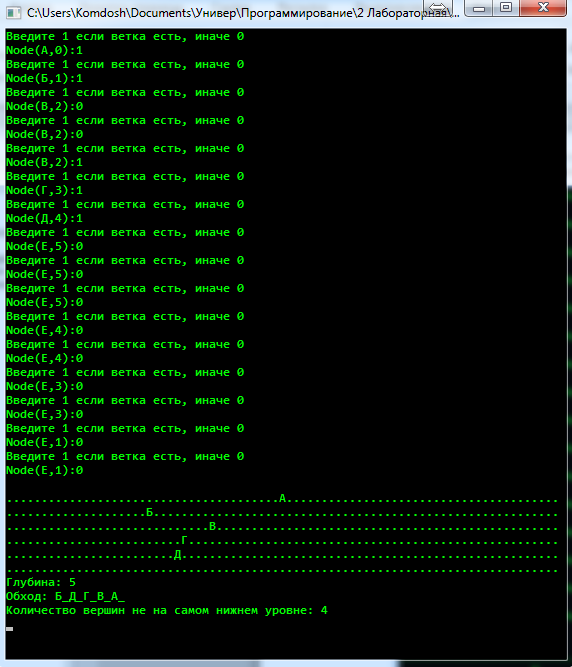
1. Ввод данных с клавиатуры
   1. Ожидаемый вывод дерева



* 1. Ожидаемый результат обхода: Б\_Д\_Г\_В\_А\_

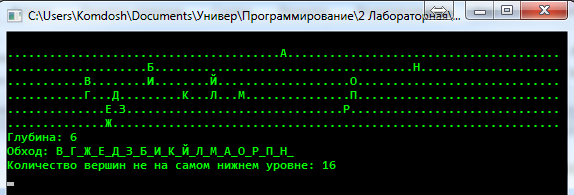
Ожидаемое количество вершин не на самом нижнем уровне:4

* 1. Результат выполнения программы:

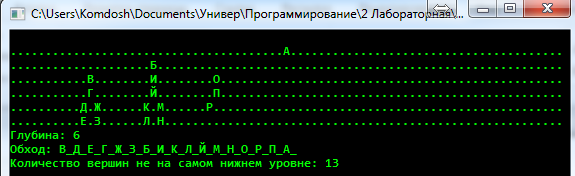


**Вывод**: Результаты совпали с ожидаемыми.

1. Генерация случайного дерева



1. Константное дерево



**Временная сложность**

Временная сложность представлена в таблице 2*.*

Таблица. 2. Временная сложность

|  |  |
| --- | --- |
| Функция | Ожидаемая |
| Создание дерева | O(n) |
| Обход | O(n) |
| Вывод | O(n) |

**Вывод**

При выполнении лабораторной работы были получены практические навыки работы c деревьями на языке программирования «С/C++».

**Список используемых источников**

* Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Федеральный образовательный стандарт / сост.: П.Г. Колинько. - СПб.: Изд-во СПБГЭТУ "ЛЭТИ", 2014. - 63 с.
* Освой С++ самостоятельно за 21 день. Сиддхартха Рао. 688 стр., с ил.; ISBN 978-5-8459-1825-3; 7 издание.
* http://stackoverflow.com – Сайт вопросов и ответов по программированию.
* http://cyberforum.ru – Форум программистов и сисадминов.

**Приложение**

**Листинг программы**

//Работу выполнили студенты 2 курса ФКТИ группы 4306 Табаков Андрей и Сыромятников Михаил

//Дерево троичное, разметка глубинная, способ обхода внутренний, надо вычислить количество вершин не на самом нижнем уровне

#include <iostream>

#include <cstdlib>

#include <cstring>

#include <algorithm>

#include <time.h>

#include <windows.h>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Классы дерева

class Node{

char d;

Node\* lNode;

Node\* mNode;

Node\* rNode;

int depth;

public:

Node():lNode(0), mNode(0), rNode(0){}

~Node(){

if(lNode) delete lNode;

if(mNode) delete mNode;

if(rNode) delete rNode;

}

friend class Tree;

};

class Tree{

Node\* root;

char letter, maxletter, startletter, startMaxletter;

int maxRow, offset, inp, maxDepth;

char\*\* screen;

void clrScr();

Node\* makeNode(int depth, int inp);

void outNodes(Node\* child, int row, int column);

Tree(const Tree&);

//Tree(Tree&&);

Tree operator = (const Tree&) const;

//Tree operator = (Tree&&) const;

public:

Tree(char letter, char maxletter, int maxRow);

~Tree();

void setInp(int inp);

Node\* getRoot()

{return root;}

int getDepth()

{return maxDepth;}

void makeTree()

{letter=startletter; maxletter=startMaxletter; maxDepth=0; root=makeNode(0, inp);}

bool exist()

{return root != NULL;}

int inside(int& count);

int inside(Node\* child, int& count);

void outTree();

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Конструкторы, деструктор

Tree::Tree(char inpletter, char inpMaxletter, int inpMaxRow):startletter(inpletter), startMaxletter(inpMaxletter), maxRow(inpMaxRow), offset(40), root(NULL), maxDepth(0)

{

screen = new char\* [maxRow];

for(int i=0; i<maxRow; ++i) screen[i]= new char[80];

}

Tree::~Tree()

{

for(int i=0; i<maxRow; ++i) delete [] screen[i];

delete [] screen;

delete root;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функции-члены

void Tree::setInp(int inpInp)

{inp=inpInp;}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Node\* Tree::makeNode(int depth, int inp)

{

Node\* tempChild = NULL;

int node=0;

switch(inp)

{

case 0:

do{

cin.clear();

cin.sync();

cout<<"Введите 1 если ветка есть, иначе 0"<<endl;

cout<<"Node("<<letter<<","<<depth<<"):"; cin>>node;

if(cin.fail())

cout<<"Что-то пошло не так, введите выражение повторно"<<endl;

}while(cin.fail());

break;

case 1:

node = (depth<rand()%maxRow+1) && (letter<=maxletter);

break;

case 2:

if(letter<=maxletter && depth<maxRow)

{

tempChild = new Node;

tempChild -> depth = depth;

tempChild -> d = letter++;

if(depth%2 || depth==0)

tempChild -> lNode = makeNode(depth+1, inp);

if(depth%3 ||depth==0)

tempChild -> mNode = makeNode(depth+1, inp);

if(depth%2 || depth==0)

tempChild -> rNode = makeNode(depth+1, inp);

}

break;

}

if(node)

{

tempChild = new Node;

tempChild -> depth = depth;

tempChild -> d = letter++;

tempChild -> lNode = makeNode(depth+1, inp);

tempChild -> mNode = makeNode(depth+1, inp);

tempChild -> rNode = makeNode(depth+1, inp);

}

if(depth>maxDepth) maxDepth=depth;

return tempChild;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Tree::outTree()

{

clrScr();

outNodes(root, 1, offset);

for(int i = 0; i<maxRow; ++i)

{

screen[i][79] = 0;

cout<<endl<<screen[i];

}

cout<<endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Tree::clrScr()

{

for(int i=0; i<maxRow; ++i)

memset(screen[i], '.', 80);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Tree::outNodes(Node\* child, int row, int column)

{

if(row&&(column<80)) screen[row-1][column-1] = child -> d;

if(row < maxRow)

{

if(child->lNode) outNodes(child -> lNode, row+1, column-(offset>>row)+1);

if(child->mNode) outNodes(child -> mNode, row+1, column);

if(child->rNode) outNodes(child -> rNode, row+1, column+(offset>>row)-1);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Рекурсивный обход

int Tree::inside(Node\* child, int& count)

{

if(child)

{

inside(child->lNode, count);

cout << child->d << '\_';

if(!(child->depth==(maxDepth-1))) {count++;}

inside(child->mNode, count);

inside(child->rNode, count);

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Прототипы функций

int menu();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Основная функция

int main(int argc, char\*\* argv)

{

srand(time(NULL));

setlocale(0, ".1251");

int iMenu, pause, count;

Tree tree('А', 'Р', 6);

do

{

system("cls");

switch(iMenu= menu())

{

case 1:

tree.setInp(0);

break;

case 2:

tree.setInp(1);

break;

case 3:

tree.setInp(2);

break;

case 0:

cout<<"До новых встреч!"<<endl;

break;

default:

cout<<"Такого пункта не существует, повторите ввод!"<<endl;

}

if(iMenu)

{

count = 0;

system("cls");

tree.makeTree();

if(tree.exist())

{

tree.outTree();

cout<<"Глубина: "<<tree.getDepth()<<endl;

cout<<"Обход: ";

tree.inside(tree.getRoot(), count); //рекурсивный обход

cout<<endl<<"Количество вершин не на самом нижнем уровне: "<<count<<endl;

cin.clear();

cin.sync();

cin.get();

}

}

}

while(iMenu);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функции

int menu()

{

int point;

do{

cin.clear();

cin.sync();

cout << "Выберите пункт меню" << endl;

cout << "1 - Ввести дерево" << endl;

cout << "2 - Сгенерировать дерево" << endl;

cout << "3 - Показать константное депрево" << endl;

cout << "0 - Выход" << endl;

cout << ">";

cin >> point;

if(cin.fail())

cout<<"Что-то пошло не так, выберите пункт меню повторно"<<endl;

}

while(cin.fail());

return point;

}