|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий |
| Кафедра вычислительной техники |
|  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2** | |
| **по дисциплине** | |
| *«Структуры и алгоритмы обработки данных»* | |
| Выполнил студент группы *ИКБО-02-17* | *Политов А.Ю* |
| Принял | *Асадова Ю.С* |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Подпись студента* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г.. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Подпись руководителя* |

Москва 2019

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт информационных технологий

Кафедра вычислительной техники

Выполнено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/А.Ю. Политов/

Зачтено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ю.С. Асадова/

**Задание на лабораторную работу**

**Дисциплина**: **«**Структуры и алгоритмы обработки данных**»**

**Студент** Политов А.Ю **Шифр** 17И0325 **Группа** ИКБО-02-17

**1. Тема**: «Древовидные структуры»

**2. Срок сдачи студентом законченной работы:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г.

**3. Исходные данные:** дерево **(**числа, разделенные пробелами)

**4. Задание:** Вычислить количество вершин, у которых равны или высоты поддеревьев, или количество потомков в правом и левом поддеревьях.

**5. Содержание отчета:**

* титульный лист;
* задание;
* оглавление;
* введение;
* основные разделы отчета;
* заключение;
* список использованных источников;

Руководитель работы Ю.С. Асадова \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2018г.

подпись

Задание принял к исполнению А.Ю. Политов \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018г.

подпись

*Оглавление*

Введение…………………………………………………………………………… 4

1. Теоретическое введение …………………………………………………… 5
2. Постановка задачи………………………………………………………….. 6
3. Определение входных и выходных данных………………………………. 6
4. Краткое описание программы …………………………………………….. 6
5. Разработка программы (код) ……………………………………………… 7
6. Разработка тестов …………………………………………………………. 8

Заключение ………………………………………………………………………..10

Список литературы ……………………………………………………………….11

*Введение*

Целью данной лабораторной работы является ознакомление с древовидными структурами данных, получение умений реализации типовых операций с ними, приобретение навыков применения древовидных структур при реализации алгоритмов поиска и сортировки. Лабораторная работа была выполнена в среде разработки MS Visual Studio на языке С++.

1. ***Теоретическое введение.***

Древовидная структура является одним из способов представления иерархической структуры в графическом виде.

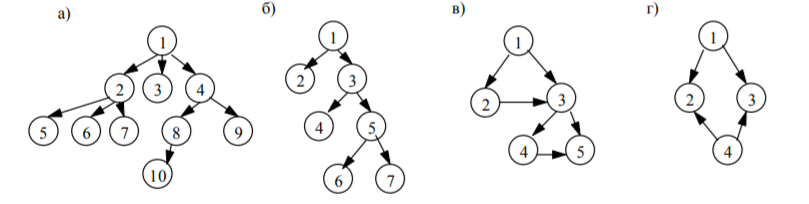
Древовидной структурой называется благодаря тому, что граф выглядит как перевернутое дерево. По этой же причине говорят, что корневой узел (корень) находится на самом верху, а листья — внизу.

В теории графов дерево — связанный ациклический граф (для не ориентированных графов) или связанный ацикличный граф, в котором не более одного узла не имеют входящих ребер, а остальные узлы имеют строго по одному входящему узлу (для ориентированных графов). Ациклический ориентированный граф без жесткого условия связывания называется сетью, Несвязанный граф из нескольких деревьев - лесом.

Из совокупности древовидных структур состоят неоднородные семантические сети.

Дерево – это совокупность узлов (вершин) и соединяющих их направленных ребер (дуг), причем в каждый узел (за исключением одного - корня) ведет ровно одна дуга. Корень – это начальный узел дерева, в который не ведет ни одной дуги. Линии, связывающие элементы называются «ветвями», а сами элементы называются узлами. Узлы без потомков называются «конечными узлами» или «листьями».

Например, на рисунке структуры, а) и б) являются деревьями, а в) и г) - нет.



1. ***Постановка задачи.***

Вычислить количество вершин, у которых равны или высоты поддеревьев, или количество потомков в правом и левом поддеревьях.

**Дано:** количество вершин;

**Ограничения на данные:** -

**Ограничения на решаемую задачу**: нет

1. ***Определение входных и выходных данных.***

**Входные данные** – числа, вводится с клавиатуры.

**Результат** – число.

1. ***Краткое описание программы.***

На вход программы подается количество вершин. После пользователь вводит данные, из которых строится дерево. Далее программа подсчитывает размер и глубину поддеревьев. После программа проходит по дереву и ищет все узлы с требуемыми характеристиками.

1. ***Разработка программы (код).***

**#include <iostream>**

**#include <Windows.h>**

**using namespace std;**

**struct Node {**

**int value;**

**Node\* left;**

**Node\* right;**

**};**

**void add\_node(int value, Node \*&MyTree) {**

**if (NULL == MyTree) {**

**MyTree = new Node;**

**MyTree->value = value;**

**MyTree->left = MyTree->right = NULL;**

**}**

**if (value == MyTree->value) return;**

**if (value < MyTree->value) {**

**if (MyTree->left != NULL)**

**add\_node(value, MyTree->left);**

**else {**

**MyTree->left = new Node;**

**MyTree->left->left = MyTree->left->right = NULL;**

**MyTree->left->value = value;**

**}**

**}**

**if (value > MyTree->value) {**

**if (MyTree->right != NULL)**

**add\_node(value, MyTree->right);**

**else {**

**MyTree->right = new Node;**

**MyTree->right->left = MyTree->right->right = NULL;**

**MyTree->right->value = value;**

**}**

**}**

**}**

**void del(Node \*&Tree) {**

**if (Tree != NULL) {**

**del(Tree->left);**

**del(Tree->right);**

**delete Tree;**

**Tree = NULL;**

**}**

**}**

**void show(Node \*Tree) {**

**if (Tree != NULL) {**

**cout << Tree->value << " ";**

**show(Tree->left);**

**show(Tree->right);**

**}**

**}**

**int count\_of\_items(int &count, Node\* tree) {**

**if (tree == NULL) return count;**

**if (count == 0) count++;**

**if (tree->left != NULL) {**

**count++;**

**count\_of\_items(count ,tree->left);**

**}**

**if (tree->right != NULL) {**

**count++;**

**count\_of\_items(count ,tree->right);**

**}**

**return count;**

**}**

**int height\_of\_tree(int &height, Node\* tree) {**

**if (height == 0 && tree != NULL) height++;**

**if (tree == NULL) return height--;**

**if (tree->right != NULL) {**

**height++;**

**height\_of\_tree(height, tree->right);**

**}**

**if (tree->left != NULL) {**

**height++;**

**height\_of\_tree(height, tree->left);**

**}**

**return height;**

**}**

**int count\_of\_nodes(int &count, Node\* tree) {**

**if (tree == NULL) return count;**

**int a, b ,c , d;**

**a = b = c = d = 0;**

**if (count\_of\_items(a, tree->right) == count\_of\_items(b, tree->left) ||**

**height\_of\_tree(c, tree->right) == height\_of\_tree(d, tree->left))**

**count++;**

**if(tree->left != NULL)**

**count\_of\_nodes(count, tree->left);**

**if (tree->right != NULL)**

**count\_of\_nodes(count, tree->right);**

**return count;**

**}**

**int main() {**

**SetConsoleCP(1251);**

**SetConsoleOutputCP(1251);**

**Node \*tree = NULL;**

**cout << "Введите количество элементов: ";**

**int count;**

**while (!(cin >> count)**

**) {**

**cout << "Ошибка чтения!\n";**

**cin.clear();**

**fflush(stdin);**

**}**

**cout << "Введите элементы: ";**

**for (int i = 0; i < count; i++) {**

**int value;**

**while (!(cin >> value)) {**

**cout << "Ошибка чтения!\n";**

**cin.clear();**

**fflush(stdin);**

**}**

**add\_node(value, tree);**

**}**

**count = 0;**

**cout << endl << "Количество узлов: " << count\_of\_nodes(count, tree) << endl;**

**del(tree);**

**system("pause");**

**}**

1. ***Разработка тестов***

В таблице 2 представленный сведения о тестировании программы.

Таблица 2 – Тестирование программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  теста | Входные данные | Эталон результата | Результат программы | Результат анализа  теста |
| word |  |  |  |
| 1 | 7  3  7  4  9  32  2  4 | 3 | 3 | + |

На рисунке 1 показана работа программы для данных, указанных в таблице 2.

Рисунок 1 - Результат выполнения теста 1

На рисунке 2 показана работа программы для данных, указанных в таблице 2.

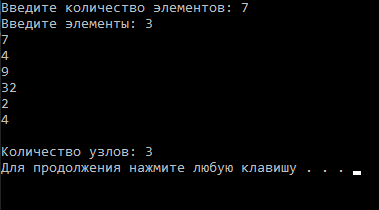


Рисунок 2 - Результат выполнения теста 2

***Заключение***

В ходе выполнения работы были получены практические навыки при работе древовидными структурами данных, получены умения реализации типовых операций с ними, приобретены навыки применения древовидных структур при реализации алгоритмов поиска и сортировки на языке С++.

***Список литературы***

1. «qaru.site» — информационный сервис для разработчиков [Электронный ресурс]. URL: <http://qaru.site> (дата обращения: 26.03.2019)
2. «cppstudio» — информационный сервис для разработчиков [Электронный ресурс]. URL: <http://cppstudio.com> (дата обращения: 27.03.2019)
3. [Кей Хорстманн, Гари Корнелл «С++. Библиотека профессионала. Том 1».10-е издание (2016](https://t.me/progbook/168)) (дата обращения: 27.03.2019)