|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Нелинейные структуры данных. Бинарное дерево.»** | |
|  | |
| Выполнил студент группы ИКБО-02-21 | Семянников Н.С. |
| Принял преподаватель | Сорокин А.В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# Постановка задачи

## Цель работы

Получение умений и навыков разработки и реализаций операций над структурой данных бинарное дерево.

## Постановка задачи

Создать идеально сбалансированное бинарное дерево из n узлов. Структура узла дерева включает: информационная часть узла, указатель на левое и указатель на правое поддерево. Отобразить дерево на экране, повернув его справа налево. Реализовать операции варианта №3.

Таблица 1 - Задание варианта

|  |  |  |
| --- | --- | --- |
| № Варианта | Значение информационной части | Операции варианта |
| 3 | Символьное значение | Вернуть самый левый узел дерева.  Определить длину пути (количество ребер) от корня до ближайшего узла с заданным значением.  Найти максимальное значение среди значений листьев дерева. |

## Вопросы

1. Что определяет степень дерева?
2. Какова степень сильноветвящегося дерева?
3. Что определяет путь в дереве?
4. Как рассчитать длину пути в дереве?
5. Какова степень бинарного дерева?
6. Может ли дерево быть пустым?
7. Дайте определение бинарного дерева?
8. Дайте определение алгоритму обхода?
9. Приведите рекуррентную зависимость для вычисления высоты дерева.
10. Изобразите бинарное дерево, корень которого имеет индекс 6, и которое представлено в памяти таблицей вида

|  |  |  |  |
| --- | --- | --- | --- |
| Индекс | key | left | right |
| 1 | 12 | 7 | 3 |
| 2 | 15 | 8 | NULL |
| 3 | 4 | 10 | NULL |
| 4 | 10 | 5 | 9 |
| 5 | 2 | NULL | NULL |
| 6 | 18 | 1 | 4 |
| 7 | 7 | NULL | NULL |
| 8 | 14 | 6 | 2 |
| 9 | 21 | NULL | NULL |
| 10 | 5 | NULL | NULL |

1. Укажите путь обхода дерева по алгоритму: прямой; обратный; симметричный



Рисунок 1 - Бинарное дерево к вопросу 11

1. Какая структура используется в алгоритме обхода дерева методом в «ширину»?
2. Выведите путь при обходе дерева в «ширину». Продемонстрируйте использование структуры при обходе дерева.



Рисунок 2 - Бинарное дерево к вопросу 13

1. Какая структура используется в не рекурсивном обходе дерева методом в «глубину»?
2. Выполните прямой, симметричный, обратный методы обхода дерева выражений.



Рисунок 3 - Дерево выражения к вопросу 15

1. Для каждого заданного арифметического выражения постройте бинарное дерево выражений:

1)a+b-c\*d+e

2)/a-b\*c d

3)a bc d / -\*

4)\* -/ + abcde

1. В каком порядке будет проходиться бинарное дерево, если алгоритм обхода в ширину будет запоминать узлы не в очереди, а в стеке?
2. Постройте бинарное дерево поиска, которое в результате симметричного обхода дало бы следующую последовательность узлов?

40 45 46 50 65 70 75

1. Приведённая ниже последовательность получена путём прямого обхода бинарного дерева поиска. Постройте это дерево.

50 45 35 15 40 46 65 75 70

1. Дано бинарное дерево поиска, представленное на рисунке 14. Выполните действия над исходным деревом и покажите дерево:

1) после включения узлов 1, 48, 75, 100

2) после удаления узлов 5, 35

3) после удаления узла 45

4) после удаления узла 50

5) после удаления узлов 5, 35

6) после удаления узла 65 и вставки его снова

7) после удаления узлов 5, 35



Рисунок 4 - Исходное бинарное дерево поиска к вопросу 20

## Ответы на вопросы

1. Степень дерева – это максимальная степень вершин, входящих в дерево
2. Степень сильноветвящегося дерева составляет степень больше 2
3. Путь в дереве — это список отдельных вершин, в котором последовательные вершины соединены рёбрами дерева.
4. Длина пути дерева определяется как сумма длин путей ко всем его вершинам
5. Степень бинарного дерева составляет степень не больше 2
6. Бинарное дерево может быть пустым
7. Бинарные деревья являются деревьями со степенью не более двух. Бинарное дерево – это динамическая структура данных, представляющее собой дерево, в котором каждая вершина имеет не более двух потомков
8. Обход дерева – это алгоритм, который позволяет осуществить доступ ко всем узлам дерева, только один раз
9. Рекуррентную зависимость для вычисления высоты дерева состоит в том, чтобы обойти дерево и вычислить высоту левого и правого поддерева.
10. Рисунок 5



Рисунок 5 - Ответ 10

Прямым:

A, B, D G, C, E, H, I, F

Обратный:

G, D, B, H, I, E, F, C, A

Симметричным:

G, D, B, A, H, E, I, C, F

1. Структура стек или очередь используется в алгоритме обхода дерева методом в «ширину»
2. A, B, C, D, E, F, G, H, I
3. Структура стек используется в алгоритме обхода дерева методом в «глубину»

Прямой: -+a/\*bcde

Симметричный: a+b\*c/d-e

Обратный: abc\*d/+e-

1. Рисунок 6-9



Рисунок 6 - Ответ 16.1

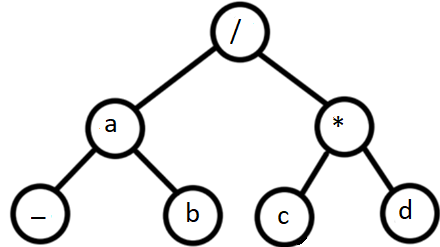


Рисунок 7 - Ответ 16.2



Рисунок 8 - Ответ 16.3



Рисунок 9 - Ответ 16.4

1. С левого края будет проходиться бинарное дерево, если алгоритм обхода в ширину будет запоминать узлы не в очереди, а в стеке
2. Рисунок 10



Рисунок 10 - Ответ 18

1. Рисунок 11



# Решение

## Постановка задачи

Дано: идеально сбалансированное дерево из n узлов.

Результаты: приложение, выполняющее операции отображение дерева на экране, возврата самого левого узла дерева, определения длины пути от корня до ближайшего узла с заданным значением и поиск максимального значения среди листьев дерева.

## Описание модели решения

Структура данных, хранящая данные узла, имеет следующие поля: data – данные хранящиеся в узле дерева, а также два указателя на левый и правый узел дерева. Для создания дерева используется случайный ввод данных. Вывод, поиск левого узла, определение длины пути от корня до ближайшего узла с заданным значением и поиск наибольшего значения происходит посредствам рекурсивного прохода по узлам дерева.

# Алгоритмы операций

## Функция создания узла

Принимает: значение узла

Возвращает: узел

Описание: создаёт новый узел и присваивает ему переданное значение.

Код:

BTNode\* createBTNode(char data)

{

BTNode\* p = new BTNode;

if (p != NULL)

{

p->data = data;

p->left = NULL;

p->right = NULL;

}

return p;

}

## Функция создания дерева

Принимает: ссылку на элемент root, отсортированный массив, начальный и конечный элементы

Описание: находит середину переданного отрезка и создаёт узел со значением середины, а за конец берёт середину текущего интервала. Затем создаёт узлы, которые находятся слева, по предыдущему алгоритму до момента, когда начало и конец отрезка совпадут, после этого переходит к созданию узлов, находящихся справа при этом сдвигая начало к середине предыдущего отрезка.

Код:

void createBTree(BTNode\*\* root, int arr[], int start, int end)

{

if (start <= end)

{

int mid = (start + end + 1) / 2;

\*root = createBTNode(arr[mid]);

createBTree(&((\*root)->left), arr, start, mid - 1);

createBTree(&((\*root)->right), arr, mid + 1, end);

}

}

## Функция возврата самого левого узла

Принимает: ссылку на узел

Возвращает: узел

Описание: проходит по всем элементам дерева, которые находятся слева и возвращает последний из них

Код:

BTNode\* getMostLeft(BTNode\* p) {

if (p->left)

{

getMostLeft(p->left);

}

else

{

return p;

}

}

## Нахождение пути до значения в дереве

Принимает: ссылку на узел, искомое значение, количество рёбер

Возвращает: количество рёбер

Описание: сравнивает искомое значение со значением в узле, если искомое больше, то переходим к правому дочернему элементу, если искомое меньше, то переходим к левому дочернему элементу, если искомое равно значению в узле, то возвращаем количество рёбер до этого узла.

Код:

int findValue(BTNode\* p, char needle, int steps) {

if (p != nullptr)

{

if (needle == p->data)

{

return steps;

}

if (needle > p->data)

{

findValue(p->right, needle, ++steps);

}

else {

findValue(p->left, needle, ++steps);

}

}

else {

return -1;

}

}

## Поиск наибольшего значения в дереве

Принимает: ссылка на узел

Возвращает: значение узла

Описание: проходит по всем элементам дерева, которые находятся справа и возвращает последний из них

Код:

char getMaxValue(BTNode\* p) {

if (p->right != nullptr)

{

getMaxValue(p->right);

}

else {

return p->data;

}

}

## Вывод дерева

Принимает: префикс, узел, правый ли это узел у родителя

Описание: выводит префикс перед каждым узлом, затем выводит узел, дальше идёт по правым дочерним узлам до конца, выводя их, после чего выводит узлы, находящиеся слева рекурсивно.

Код:

void printBT(const wstring& prefix, const BTNode\* node, bool isRight)

{

if (node != nullptr)

{

wcout << prefix;

wcout << (isRight ? L"├───" : L"└───");

wcout << (char)node->data << endl;

printBT(prefix + (isRight ? L"│ " : L" "), node->right, true);

printBT(prefix + (isRight ? L"│ " : L" "), node->left, false);

}

}

# Результат

## Тестирование

|  |  |  |
| --- | --- | --- |
| Вводимые значения | Полученные данные | Тест пройден/не пройден |
| n=10  arr = {b, h, h, i, j, k, m, r, u, x}  ch = i | Most left = b  path = 3  max = x | Пройден |

Доказательство работоспособности программы (Рисунок 11).

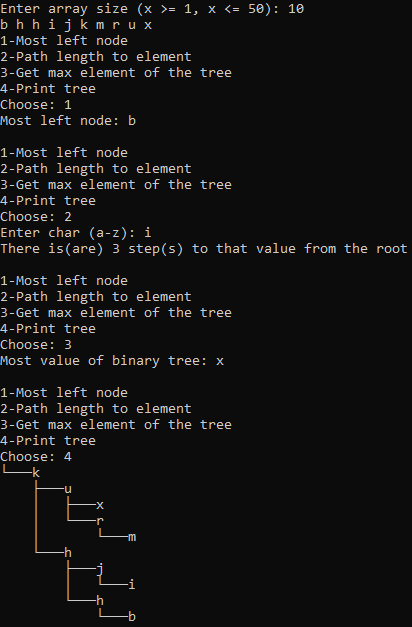


Рисунок 11 – Результат

## Код программы

#include <iostream>

#include <iomanip>

#include <io.h>

#include <fcntl.h>

using namespace std;

struct BTNode

{

int data;

struct BTNode\* left;

struct BTNode\* right;

};

BTNode\* createBTNode(char data)

{

BTNode\* p = new BTNode;

if (p != NULL)

{

p->data = data;

p->left = NULL;

p->right = NULL;

}

return p;

}

void createBTree(BTNode\*\* root, int arr[], int start, int end)

{

if (start <= end)

{

int mid = (start + end + 1) / 2;

\*root = createBTNode(arr[mid]);

createBTree(&((\*root)->left), arr, start, mid - 1);

createBTree(&((\*root)->right), arr, mid + 1, end);

}

}

void printBT(const wstring& prefix, const BTNode\* node, bool isRight)

{

if (node != nullptr)

{

wcout << prefix;

wcout << (isRight ? L"├───" : L"└───");

wcout << (char)node->data << endl;

printBT(prefix + (isRight ? L"│ " : L" "), node->right, true);

printBT(prefix + (isRight ? L"│ " : L" "), node->left, false);

}

}

void printBT(const BTNode\* node)

{

printBT(L"", node, false);

}

BTNode\* getMostLeft(BTNode\* p) {

if (p->left)

{

getMostLeft(p->left);

}

else

{

return p;

}

}

int findValue(BTNode\* p, char needle, int steps) {

if (p != nullptr)

{

if (needle == p->data)

{

return steps;

}

if (needle > p->data)

{

findValue(p->right, needle, ++steps);

}

else {

findValue(p->left, needle, ++steps);

}

}

else {

return -1;

}

}

char getMaxValue(BTNode\* p) {

if (p->right != nullptr)

{

getMaxValue(p->right);

}

else {

return p->data;

}

}

int compare(const void\* x1, const void\* x2)

{

return (\*(int\*)x1 - \*(int\*)x2);

}

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

{

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

char ch;

int res, n = 0, a;

wcout << "Enter array size (x >= 1, x <= 50): ";

cin >> n;

if (n < 0) {

n = 1;

}

else if (n > 50) {

n = 50;

}

int\* arr = new int[n];

srand(time(NULL));

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

{

arr[i] = rand() % 26 + 97;

}

qsort(arr, n, sizeof(int), compare);

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

{

wcout << (char)arr[i] << " ";

}

wcout << endl;

BTNode\* root = NULL;

createBTree(&root, arr, 0, n - 1);

while (true)

{

wcout << "1-Most left node\n2-Path length to element\n3-Get max element of the tree\n4-Print tree\nChoose: ";

cin >> a;

switch (a)

{

case 1: {

wcout << "Most left node: " << (char)getMostLeft(root)->data << endl;

break;

}

case 2: {

wcout << "Enter char (a-z): ";

cin >> ch;

res = findValue(root, ch, 0);

if (res == -1)

{

wcout << "No such value";

}

else

{

wcout << "There is(are) " << res << " step(s) to that value from the root";

}

wcout << endl;

break;

}

case 3: {

wcout << "Most value of binary tree: " << getMaxValue(root) << endl;

break;

}

case 4: {

printBT(root);

break;

}

default:

exit(0);

break;

}

wcout << endl;

}

return 0;

}

# Вывод

Входе выполнения практической работы была изучена работа с бинарным деревом. Была разработана и протестирована программа для задачи варианта.

# Список литературы

1. Система вопросов и ответов о программировании: <https://stackoverflow.com/>
2. Веб ресурс: <https://ru.wikipedia.org/wiki/Двоичное_дерево_поиска>