

## Отчет по лабораторной работе №23

по курсу: языки и методы программирования

студент группы : М8О-105Б-21 Козлов Егор Сергеевич, № по списку: 9

Адреса www, e-mail, jabber, skype: iamaghoulzxc@gmail.com

Работа выполнена: "15 мая 2022 г."

Преподаватель: каф. 806 В.К.Титов

Входной контроль знаний с оценкой: \_\_\_\_\_

Отчет сдан " \_\_\_\_ " \_\_\_\_\_ 20\_\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_

**1. Тема:** Динамические структуры данных. Обработка деревьев

**2. Цель работы:** Составить программу на языке Си для построения и обработки двоичного дерева

**3. Задание (вариант 18):** Определить ширину двоичного дерева

**4. Оборудование(лабораторное):**

ЭВМ \_\_\_\_\_, процессор \_\_\_\_\_, имя узла сети \_\_\_\_\_ с ОП \_\_\_\_\_ ГБ

НМД \_\_\_\_\_ ГБ. Терминал \_\_\_\_\_ адрес \_\_\_\_\_. Принтер \_\_\_\_\_

Другие устройства \_\_\_\_\_

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Ryzen 7 5800 @ 8x 3.2 GHz , ОП 16384 МБ, НМД \_\_\_\_\_ ГБ. Монитор Встроенный

Другие устройства \_\_\_\_\_

**5. Программное обеспечение(лабораторное):**

Операционная система семейства UNIX, наименование \_\_\_\_\_ версия \_\_\_\_\_

Интерпретатор команд: \_\_\_\_\_ версия \_\_\_\_\_

Система программирования: \_\_\_\_\_ версия \_\_\_\_\_

Редактор текстов: \_\_\_\_\_ версия \_\_\_\_\_

Утилиты операционной системы: \_\_\_\_\_

Прикладные системы и программы: \_\_\_\_\_

Местонахождение и имена файлов и программ данных: \_\_\_\_\_

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства UNIX, наименование Ubuntu версия 22.04

Интерпретатор команд: bash версия \_\_\_\_\_

Система программирования: C версия \_\_\_\_\_

Редактор текстов: Emacs версия \_\_\_\_\_

Утилиты операционной системы: \_\_\_\_\_

Прикладные системы и программы: \_\_\_\_\_

Местонахождение и имена файлов и программ данных: /usr/bin , а также /bin

**6. Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Рекурсивно вычисляем глубину дерева. Создаем массив размером равным глубине дерева, в который будем складывать количество узлов на каждой глубине.

Начинаем поиск в ширину из корня дерева:

Рекурсивно проходимся по дереву, передавая каждый раз текущую вершину и её глубину. В массив для данной глубины прибавляется единица. Таким образом после окончания обхода мы получаем массив с количеством узлов на каждой глубине. Выводим максимальную глубину. Алгоритм поиска ширины дерева закончен.

**7. Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

23.cpp :

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<time.h>
4
5 struct Node;
6 typedef char treeData;
7 typedef Node *link;
8
9 struct Node {
10     treeData data;
11     link left, right;
12 } *tree;
13
14 int i, counter = 0;
15
16 void printTree(link tree) {
17     static int nodeDeep = 0;
18     ++nodeDeep;
19     if (tree) {
20         printTree(tree->right);
21         for (i = 0; i < nodeDeep; ++i) printf("    ");
22         printf("\_\\_\'%c\'\\_\\n", tree->data);
23         printTree(tree->left);
24     }
25     --nodeDeep;
26 }
27
28 void insertNode(link &t, treeData v) {
29     if (!t) {
30         t = new Node;
31         t->data = v;
32         t->right = 0;
33         t->left = 0;
34     } else {
35         if (int(v) <= int(t->data)) insertNode(t->left, v);
36         else if (int(v) > int(t->data)) insertNode(t->right, v);
37     }
38 }
39
40 link parent;
41
42 void deleteNode(link &tree) {
43     if (tree->right) deleteNode(tree->right);
44     else {
45         parent->data = tree->data;
46         parent = tree;
47         tree = tree->left;
48     }
49 }
50
51 void deleteTree(link &tree, treeData v) {
52     if (tree)
53         if (v < tree->data) deleteTree(tree->left, v);
54         else if (!(tree->right)) tree = tree->left;
55         else if (!(tree->left)) tree = tree->right;
56         else {
57             parent = tree;
58             deleteNode(parent->left);
59         }
60 }

```

```

61 void generateTree(link &t, int n) {
62     for (i = 0; i < n; ++i) {
63         treeData v = 33 + (rand() % 93);
64         insertNode(tree, v);
65     }
66 }
67
68 void count(link t) {
69     if (t) {
70         count(t->right);
71         counter++;
72         count(t->left);
73     }
74 }
75
76 int treeHeight(link t) {
77     if (!t) return 0;
78     int leftH = treeHeight(t->left);
79     int rightH = treeHeight(t->right);
80     if (leftH > rightH) return leftH + 1;
81     else return rightH + 1;
82 }
83
84 void dfs(link t, int treeHeight, int *Array) {
85     ++Array[treeHeight];
86     if (t->left) dfs(t->left, treeHeight + 1, Array);
87     if (t->right) dfs(t->right, treeHeight + 1, Array);
88 }
89
90 int treeWidth(link t) {
91     int max = 0;
92     int height = treeHeight(t);
93     int *Array = new int[height];
94     dfs(t, 0, Array);
95
96     for (int i = 0; i < height; ++i) {
97         if (Array[i] > max) max = Array[i];
98     }
99
100     return max;
101 }
102
103
104 void mainAction(link t) {
105 }
106
107
108 int main() {
109     time_t t;
110     srand(time(&t));
111     int k = 1, n;
112     tree = 0;
113     treeData v;
114     while (k) {
115         printf("\n      MENU\n "
116             "0 - exit\n"
117             "1 - add random tree"
118             "\n 2 - print tree"
119             "\n 3 - insert item"
120             "\n 4 - delete item"
121             "\n 5 - number of nodes"
122             "\n 6 - clear tree"
123             "\n 7 - levels"
124             "\n 8 - main action"
125             "\n =>");
126         scanf("%d", &k);
127         if (!k) break;
128         if (k == 1) {
129             printf("\nInput number of items -> ");
130             scanf("%d", &n);
131             generateTree(tree, n);
132         }
133         if (k == 2)
134             if (tree) printTree(tree);
135             else printf("\nTree is empty\n");
136         if (k == 3) {
137             printf("For insert input v -> ");
138             scanf(" %c", &v);
139

```

```

140         insertNode(tree, v);
141     }
142     if (k == 4) {
143         printf("For delete input v ->");
144         scanf(" %c", &v);
145         deleteTree(tree, v);
146     }
147     if (k == 5) {
148         counter = 0;
149         count(tree);
150         printf("\nNumber of nodes -> %d\n", counter);
151     }
152     if (k == 6) tree = 0;
153     if (k == 7) {
154         printf("Levels of tree -> %d\n", treeHeight(tree));
155     }
156     if (k == 8) {
157
158         printf("Breadth of tree %d\n", treeWidth(tree));
159     }
160 }
161 return 0;
162 }

```

Пункты 1-7 отчёта составляются **строго до** начала лабораторной работы.

Допущен к выполнению работы. Подпись преподавателя \_\_\_\_\_

## 5

```

void deleteTree(link &tree, treeData v) {
    if (tree)
        if (v < tree->data) deleteTree(tree->left, v);
        else if (!(tree->right)) tree = tree->left;
        else if (!(tree->left)) tree = tree->right;
        else {
            parent = tree;
            deleteNode(parent->left);
        }
}

void generateTree(link &t, int n) {
    for (i = 0; i < n; ++i) {
        treeData v = 33 + (rand() % 93);
        insertNode(tree, v);
    }
}

void count(link t) {
    if (t) {
        count(t->right);
        counter++;
        count(t->left);
    }
}

int treeHeight(link t) {
    if (!t) return 0;
    int leftH = treeHeight(t->left);
    int rightH = treeHeight(t->right);
    if (leftH > rightH) return leftH + 1;
    else return rightH + 1;
}

void dfs(link t, int treeHeight, int *Array) {
    ++Array[treeHeight];
    if (t->left) dfs(t->left, treeHeight + 1, Array);
    if (t->right) dfs(t->right, treeHeight + 1, Array);
}

int treeWidth(link t) {
    int max = 0;
    int height = treeHeight(t);
    int *Array = new int[height];
    dfs(t, 0, Array);

    for (int i = 0; i < height; ++i) {
        if (Array[i] > max) max = Array[i];
    }

    return max;
}

void mainAction(link t) {
}

int main() {
    time_t t;
    srand(time(&t));
    int k = 1, n;
    tree = 0;
}

```

```

treeData v;
while (k) {
    printf("\n    MENU\n "
        "0 - exit\n"
        "1 - add random tree"
        "\n 2 - print tree"
        "\n 3 - insert item"
        "\n 4 - delete item"
        "\n 5 - number of nodes"
        "\n 6 - clear tree"
        "\n 7 - levels"
        "\n 8 - main action"
        "\n =>");
    scanf("%d", &k);
    if (!k) break;
    if (k == 1) {
        printf("\nInput number of items -> ");
        scanf("%d", &n);
        generateTree(tree, n);
    }
    if (k == 2)
        if (tree) printTree(tree);
        else printf("\nTree is empty\n");
    if (k == 3) {
        printf("For insert input v -> ");
        scanf(" %c", &v);
        insertNode(tree, v);
    }
    if (k == 4) {
        printf("For delete input v ->");
        scanf(" %c", &v);
        deleteTree(tree, v);
    }
    if (k == 5) {
        counter = 0;
        count(tree);
        printf("\nNumber of nodes -> %d\n", counter);
    }
    if (k == 6) tree = 0;
    if (k == 7) {
        printf("Levels of tree -> %d\n", treeHeight(tree));
    }
    if (k == 8) {
        printf("Breadth of tree %d\n", treeWidth(tree));
    }
}
return 0;
}
isitmuse@isitmuse:~/lab/secondSem/23$ g++ 23.cpp -o 23
isitmuse@isitmuse:~/lab/secondSem/23$ ./23

```

```

MENU
0 - exit
1 - add random tree
2 - print tree
3 - insert item
4 - delete item
5 - number of nodes
6 - clear tree
7 - levels
8 - main action
=>1

```

Input number of items -> 15

MENU

0 - exit  
1 - add random tree  
2 - print tree  
3 - insert item  
4 - delete item  
5 - number of nodes  
6 - clear tree  
7 - levels  
8 - main action  
=>2

```
\_ 'm'
      \_ 'h'
            \_ 'c'
      \_ 'b'
            \_ 'Y'
                  \_ 'R'
                        \_ 'G'
                              \_ '>'
      \_ '='
            \_ '6'
                  \_ '2'
                        \_ '-'
                              \_ ')'
\_ '('
      \_ ''
```

MENU

0 - exit  
1 - add random tree  
2 - print tree  
3 - insert item  
4 - delete item  
5 - number of nodes  
6 - clear tree  
7 - levels  
8 - main action  
=>3

For insert input v -> S

MENU

0 - exit  
1 - add random tree  
2 - print tree  
3 - insert item  
4 - delete item  
5 - number of nodes  
6 - clear tree  
7 - levels  
8 - main action  
=>2

```
\_ 'm'
      \_ 'h'
            \_ 'c'
      \_ 'b'
            \_ 'Y'
                  \_ 'S'
                        \_ 'R'
                              \_ 'G'
                                    \_ '>'
```



```

      \_ '='
        \_ '6'
          \_ '2'
            \_ '-'
              \_ ')'
\_('('
  \_ ''

```

MENU

- 0 - exit
  - 1 - add random tree
  - 2 - print tree
  - 3 - insert item
  - 4 - delete item
  - 5 - number of nodes
  - 6 - clear tree
  - 7 - levels
  - 8 - main action
- =>4

For delete input v ->6

MENU

- 0 - exit
  - 1 - add random tree
  - 2 - print tree
  - 3 - insert item
  - 4 - delete item
  - 5 - number of nodes
  - 6 - clear tree
  - 7 - levels
  - 8 - main action
- =>2

```

      \_ 'm'
        \_ 'h'
          \_ 'c'
            \_ 'b'
              \_ 'Y'
                \_ 'S'
                  \_ 'R'
                    \_ 'G'
                      \_ '>'
                        \_ '='
                          \_ '6'
                            \_ '2'
                              \_ '-'
                                \_ ')'
\_('('
  \_ ''

```

MENU

- 0 - exit
  - 1 - add random tree
  - 2 - print tree
  - 3 - insert item
  - 4 - delete item
  - 5 - number of nodes
  - 6 - clear tree
  - 7 - levels
  - 8 - main action
- =>5

Number of nodes -> 15

MENU

```

0 - exit
1 - add random tree
2 - print tree
3 - insert item
4 - delete item
5 - number of nodes
6 - clear tree
7 - levels
8 - main action
=>7

```

Levels of tree -> 9

#### MENU

```

0 - exit
1 - add random tree
2 - print tree
3 - insert item
4 - delete item
5 - number of nodes
6 - clear tree
7 - levels
8 - main action
=>8

```

Breadth of tree 3

#### MENU

```

0 - exit
1 - add random tree
2 - print tree
3 - insert item
4 - delete item
5 - number of nodes
6 - clear tree
7 - levels
8 - main action
=>1

```

Input number of items -> 15

#### MENU

```

0 - exit
1 - add random tree
2 - print tree
3 - insert item
4 - delete item
5 - number of nodes
6 - clear tree
7 - levels
8 - main action
=>2

```

```

      \_ 'w'
        \_ 'w'
          \_ 'u'
            \_ 'q'
              \_ 'm'
                \_ 'h'
                  \_ 'c'
                    \_ 'b'
                      \_ 'Y'
                        \_ 'S'
                          \_ 'R'
                            \_ 'P'
                              \_ 'M'

```

```

      \_ 'J'
      \_ 'G'
      \_ 'D'
      \_ 'B'
      \_ '>'
      \_ '='
      \_ ':'
      \_ '6'
      \_ '5'
      \_ '2'
      \_ '/'
      \_ '.'
      \_ '-'
      \_ ')'
      \_ ')'
      \_ ''
      \_ '!'

```

MENU

- 0 - exit
  - 1 - add random tree
  - 2 - print tree
  - 3 - insert item
  - 4 - delete item
  - 5 - number of nodes
  - 6 - clear tree
  - 7 - levels
  - 8 - main action
- =>8

Breadth of tree 5

MENU

- 0 - exit
  - 1 - add random tree
  - 2 - print tree
  - 3 - insert item
  - 4 - delete item
  - 5 - number of nodes
  - 6 - clear tree
  - 7 - levels
  - 8 - main action
- =>0

isitmuse@isitmuse:~/lab/secondSem/23\$

**9. Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

**10.** Замечание автора по существу работы \_\_\_\_\_

**11.** Выводы \_\_\_\_\_ Я научился работать с двоичными деревьями, используя динамические структуры.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом \_\_\_\_\_

Подпись студента \_\_\_\_\_