Отчет по лабораторной работе № по курсу
Алгоритмы и структуры данных

Студент группы М8О-103Б-22 Клименко Виталий Максимович, № по списку 11

Контакты www, e-mail, icq, skype vitalikklimenko96@gmail.com

Работа выполнена: 8 мая 2023 г.

Преподаватель: доцент Никулин С.П.

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 202 __ г., итоговая оценка ___

Подпись преподавателя _____

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

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

3. **Задание** (*вариант № 11*): Проверить монотонность убывания ширины уровня дерева

4. **Оборудование** (*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel 4x 3.5GHz___ с ОП 16 ГБ___ НМД HDD 200 ГБ___ . Монитор Встроенный 1920x1080___
Другие устройства Touchpad Synaptics

5. **Программное обеспечение** (*лабораторное*):
Операционная система семейства _____, наименование _____ версия _____
интерпретатор команд _____ версия _____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX_____, наименование Pop!_OS_____ версия 22.04 jammy_____
интерпретатор команд bash_____ версия 5.1.16_____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____

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

Реализовать структуру данных Node, в котором находятся следующие поля:

- prev – указатель на родительскую вершину
- children – список указателей на дочерние узлы
- value – значение узла

Реализовать функции над этой структурой данных – добавление, удаление узлов

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

Реализация функции выяснения монотонности убывания ширины дерева:

- Находим количество дочерних узлов у данного узла
- Сравниваем с максимальным количеством дочерних узлов всех дочерних узлов
- Если количество дочерних узлов у изначального меньше – возвращаем **false**, иначе – пробегаемся этой функцией по всем дочерним узлам применяя при этом логическую операцию **И**, чтобы, если хоть один дочерний узел не уменьшался в ширине, то и функция выдавала бы **false**, иначе – **true**

Пункты 1-7 отчета составляются строго до начала лабораторной работы.
Допущен к выполнению работы. **Подпись преподавателя** _____

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем)

```
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ ls
./  ../  l23-2012.djvu  main.c  makefile  tree.c  tree.h

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ cat main.c
#include <stdio.h>
#include <stdlib.h>

#include "tree.h"


void handle_add_delete(Node* n, int delete) {
    int command = 0;
    uint64_t node_index = 0;

    do {
        printf(
            "\n"
            "Do you want to go deeper into tree, stay, or go up? (1/0/-1)\n"
            "Current tree:\n"
        );
        node_print_tree(n);

        scanf("%d", &command);

        switch (command) {
            case -1:
                if (n->prev != NULL) {
                    n = n->prev;
                } else {
                    printf("Node has no parents!\n");
                }
                break;
            case 1:
                if (n->children[0] != NULL) {
                    uint64_t child_count = node_get_children_count(n);
                    node_index = 0;

                    if (child_count == 0) {
                        printf("Node has no children!\n");
                    } else {
                        if (child_count == 1) {
                            node_index = 0;
                        } else {
                            printf("Pick a node from 0 to %I64d in current node:\n", child_count - 1);
                            scanf("%I64d", &node_index);
                        }
                    }

                    if (node_index > child_count - 1) {
                        printf("No such child!\n");
                    } else {
                        n = n->children[node_index];
                    }
                } else {
                    printf("Node has no children!\n");
                }
                break;
            default:
                break;
        }
    } while (command != 0);


    if (delete) {
        if (n->prev == NULL) {
            printf("Current node is root! Can't delete it.\n");
            return;
        }

        node_delete_child(n->prev, node_index);
        return;
    }

    int64_t value = 0;
    printf("Input child value: ");
    scanf("%I64d", &value);
```

```
    Node** children = (Node**) calloc(2, sizeof(Node*));

    children[0] = (Node*) calloc(1, sizeof(Node));
    node_empty(children[0]);
    children[0]->value = value;

    children[1] = NULL;

    node_add_children(n, children, 1);
}

void print_menu() {
    printf(
        "Pick a command by typing a corresponding number\n\n"
        "1 - Add new node\n"
        "2 - Visualize tree\n"
        "3 - Delete a node\n"
        "4 - Check if tree's width is descending\n"
        "To exit press Ctrl + C\n\n"
    );
}


int main() {
    Node* root = (Node*) calloc(1, sizeof(Node));
    node_empty(root);

    while (1) {
        print_menu();

        int command = 0;
        scanf("%d", &command);

        printf("_________________________\n");

        switch (command) {
        case 1:
            printf("Adding mode...\n");
            handle_add_delete(root, 0);
            printf("Current tree:\n");
            node_print_tree(root);
            break;
        case 2:
            printf("Tree visualizer...\n");
            node_print_tree(root);
            break;
        case 3:
            printf("Deleting mode...\n");
            handle_add_delete(root, 1);
            printf("Current tree:\n");
            node_print_tree(root);
            break;
        case 4:
            printf("Checking if tree's width is descending...\n");
            bool descends = node_is_width_descending(root);
            if (descends) {
                printf("The tree's width is descending!\n");
            } else {
                printf("The tree's width is not descending!\n");
            }
            break;
        default:
            printf("Unknown command...\n");
            break;
        }

        printf("\n");
    }

    return 0;
}

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ cat tree.h
#ifndef TREE_H
#define TREE_H

#include <inttypes.h>
#include <stdbool.h>

typedef struct Node {
    struct Node* prev;
    struct Node** children;
    int64_t value;
} Node;
```

```
void node_empty(Node* n);
void node_set_children(Node* n, Node** children, uint64_t count);
void node_add_children(Node* n, Node** children, uint64_t count);
void node_delete_child(Node* n, uint64_t index);

uint64_t node_get_children_count(Node* n);

void node_print_tree(Node* n);

bool node_is_width_descending(Node* n);


#endif
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ cat tree.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "tree.h"

#define SEPARATOR " "
#define TABS "\t\t\t\t\t\t\t\t\t\t"


uint64_t max(uint64_t n1, uint64_t n2) {
    if (n1 > n2) return n1;
    return n2;
}

void node_empty(Node* n) {
    n->children = (Node**) calloc(1, sizeof(Node*));
    n->children[0] = NULL;
    n->prev = NULL;
    n->value = 0;
}

void node_set_children(Node* n, Node** children, uint64_t count) {
    n->children = (Node**) calloc(count + 1, sizeof(Node*));

    for (uint64_t i = 0; i < count; ++i) {
        n->children[i] = children[i];
        n->children[i]->prev = n;
        n->children[i]->value = children[i]->value;
    }

    n->children[count] = NULL;
}

void node_add_children(Node* n, Node** children, uint64_t count) {
    uint64_t old_count = node_get_children_count(n);

    n->children = (Node**) realloc(n->children, (old_count + count + 1) * sizeof(Node*));

    for (uint64_t i = old_count; i < old_count + count; ++i) {
        n->children[i] = children[i - old_count];
        n->children[i]->prev = n;
        n->children[i]->value = children[i - old_count]->value;
    }

    n->children[old_count + count] = NULL;
}

void node_delete_child(Node* n, uint64_t index) {
    uint64_t count = node_get_children_count(n);

    for (uint64_t i = index; i < count + 1; ++i) {
        n->children[i] = n->children[i + 1];
    }

    n->children = (Node**) realloc(n->children, (count + 1) * sizeof(Node*));
    n->children[count] = NULL;
}


uint64_t node_get_children_count(Node* n) {
    uint64_t count = 0;
    while (n->children[count] != NULL) ++count;
    return count;
}


void _node_print_tree(Node* n, int lvl) {
```

```
    if (n == NULL) return;

    lvl += 1;
    int seps = (int) (strlen(SEPARATOR) * lvl);

    printf("%*s%I64d", (int) (seps - strlen(SEPARATOR)), SEPARATOR, n->value);
    printf("\n");

    if (n->children[0] == NULL) return;

    printf("%*s\\\n", (int) (seps - strlen(SEPARATOR) + 1), SEPARATOR);
    for (uint64_t i = 0; n->children[i] != NULL; ++i) {
        _node_print_tree(n->children[i], lvl);
    }
}

void node_print_tree(Node* n) {
    _node_print_tree(n, 1);
}


bool node_is_width_descending(Node* n) {
    if (n == NULL) return true;
    uint64_t node_count = node_get_children_count(n), count = 0;
    for (uint64_t i = 0; n->children[i] != NULL; ++i) {
        count = max(count, node_get_children_count(n->children[i]));
    }

    if (count > node_count) {
        return false;
    } else {
        bool is_descending = true;
        for (uint64_t i = 0; n->children[i] != NULL || !is_descending; ++i) {
            is_descending = is_descending && node_is_width_descending(n->children[i]);
        }
        return is_descending;
    }
}
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ make
gcc -std=c99 -Wall -Wextra   -c -o main.o main.c
gcc -std=c99 -Wall -Wextra   -c -o tree.o tree.c
gcc -std=c99 -Wall -Wextra -o main.out main.o tree.o

vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ cat makefile
CC = gcc
CFLAGS = -std=c99 -Wall -Wextra

main: main.o tree.o
        $(CC) $(CFLAGS) -o main.out main.o tree.o
main_o:
        $(CC) $(CFLAGS) -c main.c
tree_o:
        $(CC) $(CFLAGS) -c tree.c
clean:
        rm -f *.o main.out
vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$ ./main.out
Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

2
------------------------
Tree visualizer...
 0

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
```

```
Current tree:
 0
0
Input child value: 1
Current tree:
 0
  \
  1

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
-------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
0
Input child value: 2
Current tree:
 0
  \
  1
  2

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
-------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
  2
1
Pick a node from 0 to 1 in current node:
0

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 1
0
Input child value: 4
Current tree:
 0
  \
  1
   \
   4
  2

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
-------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
```

```
  1
   \
    4
  2
0
Input child value: 7
Current tree:
 0
  \
  1
   \
    4
  2
  7

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
   \
    4
  2
  7
1
Pick a node from 0 to 2 in current node:
2

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 7
-1

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
   \
    4
  2
  7
1
Pick a node from 0 to 2 in current node:
1

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 2
0
Input child value: 25
Current tree:
 0
  \
  1
   \
    4
  2
   \
    25
  7

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

4
------------------------
Checking if tree's width is descending...
```

```
The tree's width is descending!

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
   \
   4
  2
   \
   25
  7
1
Pick a node from 0 to 2 in current node:
0

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 1
  \
  4
0
Input child value: 8
Current tree:
 0
  \
  1
   \
   4
   8
  2
   \
   25
  7

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

4
------------------------
Checking if tree's width is descending...
The tree's width is descending!

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
   \
   4
   8
  2
   \
   25
  7
```

```
1
Pick a node from 0 to 2 in current node:
0

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 1
  \
  4
  8
0
Input child value: 9
Current tree:
 0
  \
  1
   \
   4
   8
   9
  2
   \
   25
  7

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

1
-------------------------
Adding mode...

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 0
  \
  1
   \
   4
   8
   9
  2
   \
   25
  7
1
Pick a node from 0 to 2 in current node:
0

Do you want to go deeper into tree, stay, or go up? (1/0/-1)
Current tree:
 1
  \
  4
  8
  9
0
Input child value: 123
Current tree:
 0
  \
  1
   \
   4
   8
   9
   123
  2
   \
   25
  7

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C
```

```
4
------------------------
Checking if tree's width is descending...
The tree's width is not descending!

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C

------------------------
Unknown command...

Pick a command by typing a corresponding number

1 - Add new node
2 - Visualize tree
3 - Delete a node
4 - Check if tree's width is descending
To exit press Ctrl + C



vital@vitos-hp16 MINGW64 /c/important/docs/mai/labs/l23 (master)
$
```

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

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

10. **Замечания автора** по существу работы: _____

11. **Выводы:** Я научился реализовывать деревья на Си, обрабатывать их, реализовывать на них алгоритмы

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____