Отчёт по лабораторной работе № 23 по курсу 1

студента группы М8О-108-19 Горохов Михаил Антонович, № по списку 7

Адреса www, e-mail, jabber, skype ma.gorokhoov@gmail.com

Работа выполнена: "16" апреля 2020г.

Преподаватель: Поповкин Александр Викторович каф.806

Входной контроль знаний с оценкой _____

Отчёт сдан "16" апреля 2020г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема**: Динамические структуры данных. Обработка Деревьев.
2. **Цель работы**: Научиться реализовывать динамические структуры напримере дерева и напримере задания
3. **Задание** (*вариант № 7*): Реализовать бинарное дерево. Реализовать проверку нахождения каждого листа дерева в заданном диапозоне
4. **Оборудование** (*лабораторное*):
   ЭВМ ---, процессор Intel(R) Pentium(R) CPU G2140 @ 3.30GHz, имя узла сети alice5 с ОП 8192 МБ НМД 350 ГБ. Терминал --- адрес 192.168.2.105. Принтер _____
   Другие устройства ---

   Оборудование ПЭВМ студента, если использовалось:
   Процессор Ryzen 3 2200U 2.5 GHz, ОП 4096 МБ, НМД 128 ГБ. Монитор ---
   Другие устройства ---
5. **Программное обеспечение** (*лабораторное*):
   Операционная система семейства Linux, наименование Ubuntu версия 18.04.3 LTS
   Интерпретатор команд Bash версия 4.4.19(1)-release
   Система программирования GCC версия ---
   Редактор текстов GNU Emacs версия 25.2.2
   Утилиты операционной системы _____
   Прикладные системы и программы _____
   Местонахождения и имена файлов программ и данных _____

   Программное обеспечение ЭВМ студента, если использовалось:
   Операционная система семейства Linux, наименование Xubuntu версия 18.04 bionic
   Интерпретатор команд Bash версия 5.0.9(1)-release
   Система программирования GCC версия ---
   Редактор текстов vim версия 8.1
   Утилиты операционной системы _____
   Прикладные системы и программы _____
   Местонахождения и имена файлов программ и данных _____
6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Идея проста:
Реализовать структуру данных дерево на динамических структурах (то есть на указателях),
Реализовать функции по заданию. Все особенности реализации приложены в коде.

C — команда создать дерево
a — команда добавить новый элемент
d — удалить последний элемент
s — узнать размер дерева
p — вывести содержимое дерева.

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

Код программы main.c:

```
#include <stdio.h>
#include <stdlib.h>

typedef float T;
typedef struct Tree {
   T value;
   int size;
   struct Tree* left;
   struct Tree* right;
} Tree;

void create(Tree* tree, T value) {
   tree->value = value;
   tree->left = NULL;
   tree->right = NULL;
   tree->size = 1;
   return;
}
void push(Tree* base_tree, Tree* new_tree) {
   base_tree->size += new_tree->size;
   // Если левое поддерево пустое - занять левое поддерево
   if(base_tree->left == NULL) {
      base_tree->left = new_tree;
   }
   // Тогда попробуем занять правое поддерево
   else if(base_tree->right == NULL) {
      base_tree->right = new_tree;
   }
   // Ага, деревья и слева, и справа уже есть. Тогда пристроимся к ним
   else if(base_tree->left->size <= base_tree->right->size) {
      push(base_tree->left, new_tree);
   }
   else {
      push(base_tree->right, new_tree);
   }
}
void delete_node(Tree* tree) {
   if((tree->left != NULL && tree->right == NULL) ||tree->left->size > tree->right->size){
      if(tree->left->size == 1) {
         free(tree->left);
         tree->left = NULL;
         tree->size--;
      }
      else {
         delete_node(tree->left);
         tree->size--;
      }
   }
   else {
      if(tree->right->size == 1) {
         free(tree->right);
         tree->right = NULL;
         tree->size--;
      }
```

```
        else {
            delete_node(tree->right);
            tree->size--;
        }
    }

    return;
}
void print(Tree* tree, int tabs) {
    for(int i=0; i < tabs; i++) {
        printf("\t");
    }
    printf("%f\n", tree->value);
    if(tree->left != NULL) print(tree->left, tabs+1);
    if(tree->right != NULL) print(tree->right, tabs+1);
    return;
}
short is_in_range(Tree* tree, T l, T r) {
    short result = 1;
    if(tree->left != NULL) {
        result = is_in_range(tree->left, l, r);
    }
    if(tree->right != NULL) {
        result = is_in_range(tree->right, l, r);
    }
    if (tree->size == 1 || (tree->value < l || r < tree->value)) {
        result = 0;
        return result;
    }
    return result;
}
int main() {
    Tree tree;
    char action;
    float value;

    while(scanf("%c", &action) != EOF) {
        if(action == 'c') {
            scanf("%f", &value);
            create(&tree, value);
            printf("The Tree has been created!\n");
            continue;
        }
        if(action == 'a') {
            scanf("%f", &value);
            Tree* new_tree = malloc(sizeof(Tree));
            create(new_tree, value);
            push(&tree, new_tree);
            continue;
        }
        if(action == 'p') {
            print(&tree, 0);
            continue;
        }
        if(action == 'd') {
            delete_node(&tree);
        }
        if(action == 's') {
            printf("size: %d\n", tree.size);
        }
        if(action == 'r') {
            T value2;
            scanf("%f%f", &value, &value2);
            if(is_in_range(&tree, value, value2)) {
```

```
            printf("Yes, values are in the range\n");
        }
        else {
            printf("No, values aren't in the range\n");
        }
    }
  }

  return 0;
}
```

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

Допущен к выполнению работы. Подпись преподавателя _____

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

```
[revammark@Raft laba23]$ ./main
c 1.0
The Tree has been created!
a 2.0
a 3.0
a 4.3
p
1.000000
      2.000000
            4.300000
      3.000000
d
p
1.000000
      2.000000
      3.000000
a 5.3
a 9.8
p
1.000000
      2.000000
            5.300000
      3.000000
            9.800000
a 4.5
a 3.673
p
1.000000
      2.000000
            5.300000
            4.500000
      3.000000
            9.800000
            3.673000
r 0 10
Yes, values are in the range
r 0 1
No, values aren't in the range
s
size: 7
d
d
d
d
```

```
p
1.000000
      2.000000
      3.000000
r 0.99 3.1
Yes, values are in the range
^C
[revammark@Raft laba23]$ date
Чт 16 апр 2020 13:43:44 MSK
[revammark@Raft laba23]$
```

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 Работа оказалась не сложной. Я научился реализовывать динамические структуры на примере бинарного дерева, а также обработке таких структур на примере задания проверки нахождения каждого листа в заданном диапозоне. _____
_____

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом_____
_____
_____

Подпись студента _____