Отчёт по лабораторной работе№23 по курсу 1

студента группы М8О-108Б Попова Матвея, № по списку 18

Адреса www, e-mail, jabber, skype popov.m4tvei@yandex.ru

Работа выполнена: "1" апреля 2020г.

Преподаватель: Трубченко Никита Михайлович.

Входной контроль знаний с оценкой _____

Отчёт сдан "1" апреля 2021 г., итоговая оценка _____

Подпись преподавателя _____

- **Тема**: бинарные деревья поиска

- **Цель работы**: научиться реализовывать структуры данных
- **Задание (вариант 18)**: определить ширину дерева
- **Оборудование** (*лабораторное*):
  ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____
  МБ
  НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
  Другие устройства _____

  Оборудование ПЭВМ студента, если использовалось:
  Процессор AMD Ryzen 7 4800U, ОП 16 ГБ, НМД 1 ТБ. Монитор _____
  Другие устройства _____

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

  Программное обеспечение ЭВМ студента, если использовалось:
  Операционная система семейства _____, наименование _____ версия
  Интерпретатор команд _____ версия _____
  Система программирования _____ версия _____
  Редактор текстов _____ версия _____

Утилиты операционной системы _____

_____

Прикладные системы и программы _____

Местонахождения и имена файлов программ и данных _____

_____

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

```
#include <stdio.h> //#18 calculate width of the tree
#include <stdlib.h>

struct node
{
   int k;
   int d;
   struct node *l;
   struct node *r;
};

struct node *add(struct node *t, int n, int de)
{
   de ++;
   if (t == NULL)
   {
      struct node *t = malloc(sizeof(struct node));
      t->k = n;
      t->d = de;
      t->r = NULL;
      t->l = NULL;
      return t;
   }

   if (n < t->k)
      t->l = add(t->l, n, de);
   if (n > t->k)
      t->r = add(t->r, n, de);
   return t;
};

struct node *inf(struct node *t)
{
   if (t->l == NULL)
      return t;
   return inf(t->l);
}

struct node *delete(struct node *t, int k)
{
   if (t == NULL)
      return t;
   if (k < t->k)
      t->l = delete(t->l, k);
   else if (k > t->k)
```

```
        t->r = delete(t->r, k);
    else if (t->l != NULL && t->r != NULL)
    {
        t->k = inf(t->r)->k;
        t->r = delete(t->r, t->k);
    }
    else if (t->l != NULL)
    {
        struct node *l = t->l;
        free(t);
        t = l;
    }
    else if (t->r != NULL)
    {
        struct node *r = t->r;
        free(t);
        t = r;
    }
    else
    {
        free(t);
        t = NULL;
    }
    return t;
}

int search(struct node *t, int number)
{
    if (t == NULL)
        return 0;
    if (number == t->k)
        return 1;
    if (number < t->k && t->l != NULL)
        return search(t->l, number);
    if (number > t->k && t->r != NULL)
        return search(t->r, number);
    else
        return 0;
}

void preorder(struct node *t)
{
    if (t == NULL)
        return;
    printf("%d ", t->k);
    preorder(t->l);
    preorder(t->r);
}

void postorder(struct node *t)
{
    if (t == NULL)
        return;
    preorder(t->l);
    preorder(t->r);
```

```
    printf("%d ", t->k);
}

void inorder(struct node *t)
{
    if (t == NULL)
        return;
    inorder(t->l);
    printf("%d ", t->k);
    inorder(t->r);
}

int depth(struct node *t, int m)
{
    if (t == NULL)
        return m;
    if (t->d > m)
        m = t->d;
    depth(t->l, m);
    depth(t->r, m);
}

int width(struct node *t, int n, int *arr, int m)
{
    if (t == NULL)
    {
        for (int i = 0; i <= n; i++)
        {
            if (arr[i] > m)
                m = arr[i];
        }
        return m;
    }
    width(t->l, n, arr, m);
    arr[t->d] ++;
    width(t->r, n, arr, m);
}

struct node *rem(struct node *t)
{
    if (t == NULL)
        return t;
    if (t->l == NULL && t->r == NULL)
    {
        free(t);
        return NULL;
    }
    if (t->r != NULL)
        t->r = rem(t->r);
    if (t->l != NULL)
        t->l = rem(t->l);
    return rem(t);
}

int main()
```

```
{
    printf("Operations:\n");
    printf("an --- add element n to the tree\n");
    printf("$ --- end of adding elements\n");
    printf("rn --- remove element n from the tree\n");
    printf("sn --- search element n in the tree\n");
    printf("p1 --- output the tree by preorder\n");
    printf("p2 --- output the tree by postorder\n");
    printf("p3 --- outout the tree by inorder\n");
    printf("# --- finish\n");
    int e;
    char c;
    struct node *T = NULL;
    scanf("%c %d", &c, &e);
    T = add(T, e, 1);
    while (c != '$')
    {
        scanf("%c", &c);
        if (c == 'a')
        {
            scanf("%d", &e);
            T = add(T, e, 1);
        }
    }
    int ar[129];
    int dep = depth(T, 0);
    for (int i = 0; i < 129; i++)
        ar[i] = 0;
    printf("Width of the tree is %d\n", width(T, dep, ar, 0));
    scanf("%c", &c);
    while (c != '#')
    {
        if (c == 'r')
        {
            scanf("%d", &e);
            T = delete(T, e);
        }
        if (c == 's')
        {
            scanf("%d", &e);
            if (search(T, e) == 1)
                printf("Found\n");
            else
                printf("Not found\n");
        }
        if (c == 'p')
        {
            scanf("%d", &e);
            if (e == 1)
                preorder(T);
            if (e == 2)
                postorder(T);
            if (e == 3)
                inorder(T);
            printf("\n");
        }
```

```
    scanf("%c", &c);
  }
  T = rem(T);
  return 0;
}
```

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

Допущен к выполнению работы. Подпись преподавателя _____

- **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

- **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

- Замечание автора по существу работы

- Выводы : реализовал структуру данных
-

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

Подпись студента _____