Отчет по Лабораторной работе № 23

по курсу: 1 фундаментальная информатика

студент группы : М8О-105Б-21 Титеев Рамиль Маратович , № по списку: 23

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

Работа выполнена: "1 мая 2022г"

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 33*): Определить число вершин двоичного дерева, имеющих ровно два поддерева.
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 4600 @ 6x 3.0 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 20.04
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)

При запуске программы вызывается меню из 8 функций.

0: Программа останавливает работу.

1: Генерируется дерево из случайных чисел, кол-во которых вводится в консоли. Генерация происходит следующим образом: в цикле из n итераций создается случайное значение, затем спускаемся до листов, при этом сравнивая значения (если меньше, то спускаемся влево, если больше, то спускаемся вправо). После того, как мы дошли до листа дерева, создаем новый элемент со значением, которое мы сгенерировали ранее.

2: Рисует дерево;

3: Добавляет элемент в дерево; Действия анологичны действиям из генерации дерева.

4: Удаляет из дерева введенный элемент. Для этого производим по дереву поиск нужного элемента, затем нужно рассмотреть три возможные ситуации. Если у узла нет дочерних узлов, то у его родителя нужно просто заменить указатель на 0. Если у узла есть только один дочерний узел, то нужно создать новую связь между родителем удаляемого узла и его дочерним узлом. Если у узла два дочерних узла, то нужно найти следующий за ним элемент, его правого потомка подвесить на место найденного элемента, а удаляемый узел заменить найденным узлом.

5: Выводит кол-во вершин в дереве. Для этого рекурсивно проходимся по всему дереву и считаем кол-во вершин;

6: Удаляет дерево. Для этого дерево приравнивается к нулю.

7: Выводит кол-во вершин, имеющих два поддерева; Для этого рекурсивно проходимся по дереву, параллельно проводя проверку вершины. Если слева и справа есть дочерний узел, и при этом каждый из этих узлов имеет хотя бы один лист или узел, то увеличиваем кол-во вершин на единицу.

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

lab_23.cpp:

```
// определить число вершин двоичного дерева, имеющих ровно два поддерева

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

typedef int tdata;

int i, s, n_vert;

tdata m;

struct node;

typedef node * link;
link q;

struct node{
    tdata data;
    link left;
    link right;
} *tree;


void printtree(link t){
    static int l=0;
    l++;
    if(t){
        printtree(t->right);
```

```
        for(i=0;i<l;i++)printf("    ");
        printf("\\__%d\n",t->data);
        printtree(t->left);
    }
    l--;
}

void del(link &t){
    if(t->right) del(t->right);
    else { q->data=t->data; q=t; t=t->left;}
}

void deletetree(link &tree, tdata v){
    if(tree){
        if(v<tree->data) deletetree(tree->left,v);
        else if(v>tree->data) deletetree(tree->right,v);
            else if(!(tree->right)) tree=tree->left;
        else if(!(tree->left)) tree=tree->right;
        else{
            q=tree;
                    del(q->left);
                }
    }
}

void inserttree(link &t,tdata v){
    if(!t){
        t=new node;
        t->data=v;
            t->left=0; t->right=0;
           }
    else {
            if(v<t->data) inserttree(t->left,v);
            else if(v>t->data) inserttree(t->right,v);
        }
}

void addtree(link &t, int n){
    for(i=0;i<n;i++){
        int v=rand()%50+1;
                inserttree(tree,v);
        }
}

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

void action(link t){
    if(t){
```

```
        if(t->right && t->left){
            if((t->right->right || t->right->left) && (
                t->left->right || t->left->left)){
                n_vert++;
            }
        }
        action(t->right);
        action(t->left);
    }
}

int main(){
    time_t t; srand(time(&t)); int k=1, n;
    tree=0; tdata 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 - action\n");
        scanf("%d",&k);
        if(!k) break;
        if(k==1){
            printf("\nInput number of items: ==> ");
            scanf("%d",&n);
            addtree(tree,n);
        }
        if(k==2){
            if(tree) printtree(tree);
            else printf("\nTree is empty.\n");
        }
        if(k==3){
            printf("For insert Input v = ");
            scanf("%d",&v);
            inserttree(tree,v);
        }
        if(k==4){
            printf("For delete Input v = ");
            scanf("%d",&v);
            deletetree(tree,v);
        }
        if(k==5){
            s=0;
            count(tree);
            printf("\nNumber of nodes = %d\n",s);}
        if(k==6) tree=0;
        if(k==7){
            n_vert = 0;
            action(tree);
            printf("Number of nodes with two subtrees: %d\n", n_vert);
```

```
        }
    }
    return 0;
}
```

Пункты 1-7 отчѐта 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).

```
constantfear@constantfear:~/projects/laboratory/2_semester/lab_23$ cat header.txt
**********************************************************************
*                      Лабораторная работа №23                       *
*                   Динамические структуры данных.                   *
*                       Обработка деревьев.                          *
*                 Выполнил студент гр. М8О-105-Б                     *
*                    Титеев Рамиль Маратович                         *
**********************************************************************
constantfear@constantfear:~/projects/laboratory/2_semester/lab_23$ g++ lab_23.cpp
constantfear@constantfear:~/projects/laboratory/2_semester/lab_23$ ./a.out

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

Input number of items: ==> 40

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
2
                \__50
            \__49
        \__48
            \__46
    \__42
                \__41
            \__38
                        \__37
                    \__36
                \__34
                    \__33
        \__32
                    \__31
                \__30
                        \__27
                    \__26
```

```
            \__23
                        \__22
                    \__20
                        \__18
                \__17
                                    \__15
                                \__13
                            \__12
                        \__10
                    \__7
                            \__5
                                \__4
                        \__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 - action
3
For insert Input v = 1

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
2
                \__50
            \__49
        \__48
            \__46
    \__42
                \__41
            \__38
                        \__37
                    \__36
                \__34
                    \__33
        \__32
                    \__31
                \__30
                        \__27
                    \__26
            \__23
                        \__22
                    \__20
                        \__18
                \__17
                                    \__15
                                \__13
                            \__12
                        \__10
                    \__7
                            \__5
```

```
                                \__4
                        \__2
                            \__1

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
4
For delete Input v = 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 - action
2
                \__50
            \__49
        \__48
            \__46
    \__42
                \__41
            \__38
                        \__37
                    \__36
                \__34
                    \__33
        \__32
                    \__31
                \__30
                        \__27
                    \__26
            \__23
                        \__22
                    \__20
                        \__18
                \__17
                                \__13
                            \__12
                        \__10
                    \__7
                            \__5
                                \__4
                        \__2
                            \__1

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
```

```
 6 - clear tree
 7 - action
5

Number of nodes = 29

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

Tree is empty.

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

Input number of items: ==> 40

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
2
                \__50
            \__49
                \__48
                            \__46
                        \__45
                    \__44
        \__42
                \__36
            \__35
    \__29
```

```
            \__26
                    \__25
                \__24
                    \__23
        \__21
                \__19
            \__18
                        \__17
                            \__16
                    \__15
                \__14
                        \__11
                    \__10
                        \__8
                                    \__7
                                \__6
                            \__5
                        \__1

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
7
Number of nodes with two subtrees: 4

    MENU
 0 - exit
 1 - add random tree
 2 - print tree
 3 - insert item
 4 - delete item
 5 - number of nodes
 6 - clear tree
 7 - action
0
```

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 _____