МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: «бинарные деревья»

Студент гр. 9303 _____ Ефимов М. Ю.

Преподаватель _____ Филатов А. Ю.

Санкт-Петербург
2020

Цель работы.

Познакомиться с структурой данных “дерево”, и научиться использовать его.

Задание.

Вариан 3б.

Для заданного бинарного дерева b типа BT с произвольным типом элементов:

- напечатать элементы из всех листьев дерева b;
- подсчитать число узлов на заданном уровне n дерева b (корень считать узлом 1-го уровня).

Основные теоретические положения.

Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов (рис. 31.1). Каждый элемент дерева называется вершиной (узлом) дерева. Вершины дерева соединены направленными дугами, которые называют ветвями дерева. Начальный узел дерева называют корнем дерева, ему соответствует нулевой уровень. Листьями дерева называют вершины, в которые входит одна ветвь и не выходит ни одной ветви.

Каждое дерево обладает следующими свойствами:

1. существует узел, в который не входит ни одной дуги (корень);
2. в каждую вершину, кроме корня, входит одна дуга.

Деревья особенно часто используют на практике при изображении различных иерархий. Например, популярны генеалогические деревья.

Бинарные деревья являются деревьями со степенью не более двух.

Бинарное (двоичное) дерево – это динамическая структура данных, представляющее собой дерево, в котором каждая вершина имеет не более двух потомков . Таким образом, бинарное дерево состоит из элементов, каждый из которых содержит информационное поле и не более двух ссылок на различные бинарные поддеревья. На каждый элемент дерева имеется ровно одна ссылка.

Каждая вершина бинарного дерева является структурой, состоящей из четырех видов полей. Содержимым этих полей будут соответственно:

- информационное поле (ключ вершины);
- служебное поле (их может быть несколько или ни одного);
- указатель на левое поддерево;
- указатель на правое поддерево.

Выполнение работы.

Был реализован класс Tree. Каждый обьект класса является узлом дерева.Для этого класса были написаны соотвествующие методы,для внесения, получения данных . Написаны фукнци createTree и showTree для создания и выведения на консоль. Функция printSheet непосредственно выполняет данное задание,выводит “листья” на консоль. numberNodes считает колличество узлов на заданном уровне. Большенство функций реализованно рекурсивно,исходя из определения бинарного дерева.Данные вводятся и выводятся с файла. Промежуточные значения дублируются выводом на консоль. Код программы смотрите в приложении А. Результат тестирования в таблице В.

Выводы.

Были изучены теоретические материалы по бинарным деревьям, были разработаны алгоритмы по выводу и обработке бинарных деревьев. Был разработан класс Tree для реализации стуктуры и ее обработке.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <fstream>
void inputText(std::string text)
{
    std::string path = "result.txt";
    std::string test;
    std::ofstream fin;
    fin.open(path, std::ios_base::app);
    int depth;
    if( !fin.is_open() )
    {
        std::cout<<"Ошибка открытия файла";
    }
    else
    {
        fin << text<<std::endl;
    }
    fin.close();
}
class Tree
{
private:
    Tree* right;
    Tree* left;
    char data;
public:
    Tree() {
```

```
    data = '#';
    right = left = nullptr;
}
char getData()
{
    return data;
}
void satData(char x)
{
    data = x;
}
void makeRight()
{
    right = new Tree;
}
void makeLeft()
{
    left  = new Tree;
}
Tree* getRight()
{
    return right;
}
Tree* getLeft()
{
    return left;
}
bool isLeftFree()
{
    return left == nullptr;
```

```
    }
    bool isRightFree()
    {
        return right == nullptr;
    }
};

void createForest(Tree* a, std::string info, int& n) {

    if( n >= (int)info.length() || info[n]== '\0')
    {
        n++;
        return;
    }

    if(info[n]=='/')
    {
        a = new Tree;
        n++;
    }else {

        a->satData(info[n]);

        n++;
        a->makeLeft();
        createForest(a->getLeft(), info, n );

        a->makeRight();
        createForest(a->getRight(), info, n );
    }
```

```
}
void printTree(Tree* treePtr, int p)
{
   int i;
      if(treePtr != nullptr)
   {
      printTree(treePtr->getLeft(),p+3);
      for(i=0;i<p;i++)
      {
         printf(" ");
      }
      printf("%3c\n", treePtr->getData());
      printTree(treePtr->getRight(), p+3);
   }
}
void printSheet(Tree* head)
{
   if(!head->isRightFree() && !head->isLeftFree())
      if ( head->getLeft()->getData()=='#' && head->getRight()->getData()=='#')
      {
         std::string sheet = "Листья:";
         std::cout<< sheet << head->getData() << std::endl;
         sheet =+ head->getData();
         inputText(sheet);
         return;
      }
   if(!head->isRightFree())
      printSheet(head->getLeft());
   if(!head->isLeftFree())
```

```
        printSheet(head->getRight());
}
void numberNodes(Tree* head,int number,int current,int& sum)
{
    if( head->getData()=='#' ) return;

    if(current == number)
    {

        sum++;
        std::cout<<"Атом "<< head->getData() <<" Сумма ="<<sum<<std::endl;
        std::string  atom = "Атом ";
        atom+=head->getData();
        atom+=" Сумма =" + std::to_string(sum);
        inputText(atom);
    }
    if(!head->isRightFree())
        numberNodes(head->getLeft(),number,current+1,sum);
    if(!head->isLeftFree())
        numberNodes(head->getRight(),number,current+1,sum);
}
int main()
{
    std::string path = "test.txt";
    std::string test;
    std::ifstream fin;
    fin.open(path);
    int depth;
    if( !fin.is_open() )
    {
```

```
        std::cout<<"Ошибка открытия файла";
    }
    else
    {
        fin >> test;
        fin >> depth;
    }
    Tree* head = new Tree;
    int n = 0;
    createForest(head, test, n);
    printTree(head, 1);
    printSheet(head);
    int sum = 0;
    numberNodes(head, depth, 1, sum);
    inputText("Сумма узлов: " + std::to_string(sum));
    return 0;
}.
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.2 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	ab//c// 2	b c Атом b Сумма =1 Атом c Сумма =2 Сумма узлов: 2	Результат корректен
2.	Пустой фаил	Ошибка открытия файла	Результат корректен
3.	abc//d//egd//f/// 3	c d d f Атом c Сумма =1 Атом d Сумма =2 Атом g Сумма =3 Сумма узлов: 3	Результат корректен
4	abc////el//q// 3	Некорректный результат	По условию дерево заддано. Тут оно некорректно.