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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «АиСД»

Тема: Деревья

Студент гр. 9382 Белоусова М.О.

Преподаватель Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться со структурой деревьев. Научиться создавать бинарные деревья, реализовывая их через динамическую (связанную) память (на базе указателей), и пользоваться ими. Закрепить их реализацию с помощью рекурсии на примере языка C++, освоить применение структур и классов, получить навыки работы с template c++. Выполнить работу в соответствии с заданием.

Основные теоретические положения.

Дерево – конечное множество T, состоящее из одного или более узлов, таких, что

а) имеется один специально обозначенный узел, называемый корнем данного дерева;

б) остальные узлы (исключая корень) содержатся в m >= 0 попарно не пересекающихся множествах T1, T2, ..., Tm, каждое из которых, в свою очередь, является деревом. Деревья T1, T2, ..., Tm называются поддеревьями данного дерева.

Лес – это множество (обычно упорядоченное), состоящее из некоторого (быть может, равного нулю) числа непересекающихся деревьев. Используя понятие леса, пункт *б* в определении дерева можно было бы сформулировать так: *узлы дерева, за исключением корня, образуют лес*.

КЛП - префиксная запись

ЛКП - инфиксная запись

ЛПК - постфиксная запись

Задание

Вариант №9в.

Рассматриваются бинарные деревья с элементами типа Elem (в качестве Elem использовать char). Заданы перечисления узлов некоторого дерева b в порядке КЛП и ЛКП. Требуется:

- восстановить дерево b и вывести его изображение;

- перечислить узлы дерева b в порядке ЛПК.

Ход работы.

1) Разработан алгоритм:

На вход подаются два выражения, являющиеся разными записями одного и того же бинарного дерева (ЛКП и КЛП). Используя теоретические знания о том, что во втором выражении первыми узлами записываются корни поддеревьев, находим эти корни в первом выражении, они будут в не с краю, потому что в ЛКП записи сначала записываются левые, потом корни, после правые поддеревья. Затем мы делим первое выражение на два, они будут образовывать два поддерева и будут являться левым и правым поддеревом соответственно нашего главного дерева. Затем таким образом рекурсивно составляем из выражений бинарное дерево.

2) Использованы функции:

1. main

   Сигнатура: int main().

   Назначение: является основной функцикй и телом программы.

   Описание аргументов: без аргументов.

   Возвращаемое значение: 0.

2. displayBT

   Сигнатура: void displayBT(binTree *tree, int b, int n).

   Назначение: отображение бинарного дерева.

   Описание аргументов: указатель на бинарное дерево, индекс корня, уровень корня.

   Возращаемое значение: ничего.

3. build

   Сигнатура: int build(binTree *tree, string &a, string &b, int x).

   Назначение: строит дерево по ЛКП и КЛП записи.

   Описание аргументов: указатель на бинарное дерево, строка с ЛКП записью, строка с КЛП записью, индекс корня в КЛП записи.

   Возвращаемое значение: индекс нового узла.

4. lpk

Сигнатура: void lpk(binTree *tree, int b, string& str).

Назначение: обход дерева ЛПК.

Описание аргументов: указатель на бинарное дерево, индекс корня в массиве, строка для результата.

Возвращаемое значение: ничего.

3) Использованы структуры:

1. node

Назначение: играет роль узла дерева, нужен для составления дерева.

Описание содержимого: char info – переменная типа char , которая хранит в себе содержимое узла; int lt и int lt – индексы в массиве узлов, являющиеся левым поддеревом и правым поддеревом соответственно; tode() – конструктор структуры, который инициализирует указатели lt и rt.

4) Использованы классы:

1. binTree

Назначение: в нем хранятся методы для создания дерева и его отображения.

Описание содержимого:

public: struct node, node* arr – указатель на массив, хранящий дерево, к которому можно обратиться.

Использованы методы класса binTree:

1. binTree

   Сигнатура:binTree(int len = 100).

   Назначение: конструктор.

   Описание аргументов: длинна масива для хранения дерева.

   Возвращаемое значение: ничего.

2. IsNull

   Сигнатура: bool isNull(int b).

   Назначение: проверка узла на пустой.

   Описание аргументов: индекс узла.

   Возвращаемое значение: булевое значение.

3. RootBT

Сигнатура: char RootBT(int b).

Назначение: для определения информации в узле.

Описание аргументов: индекс узла.

Возвращаемое значение: символ в узле.

4. Left

   Сигнатура: int Left(int b).

   Назначение: для нахождения левого сына.

   Описание аргументов: индекс узла.

   Возвращаемое значение: индекс левого сына.

5. Right

   Сигнатура: int Right(int b).

   Назначение: для нахождения правого сына.

   Описание аргументов: индекс узла.

   Возвращаемое значение: индекс правого сына.

6. ConsBT

   Сигнатура: int ConsBT(char b).

   Назначение: создает новый узел.

   Описание аргументов: символ, который будет лежать в узле.

   Возвращаемое значение: индекс нового узла.

7. ~binTree

   Сигнатура: ~binTree().

   Назначение: деструктор.

   Описание аргументов: нет.

   Возвращаемое значение: нет.

На вход подаются два выражения, являющимися ЛКП и КЛП записями. Затем строится дерево. Затем идет ЛПК обход. Выводится дерево и ЛПК запись

Пример работы программы.

Входные данные	Выходные данные
dbzeagfxc abdezcfgx	Считывается ЛКП запись... dbzeagfxc Считывается КЛП запись... abdezcfgx ЛПК: dzebgxfca В виде дерева: a c f x g b e z d

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	dbzeagfxc abdezcfgx	Считывается ЛКП запись... dbzeagfxc Считывается КЛП запись... abdezcfgx ЛПК: dzebgxfca В виде дерева: a c f x g b e z d	Проверка на корректность работы программы
2.	1+4*9-5 *+14-95	Считывается ЛКП запись... 1+4*9-5 Считывается КЛП запись... *+14-95 ЛПК: 14+95-* В виде дерева: * - 5 9 + 4 1	Проверка на корректность работы

3.	1+4*9-5	Считывается ЛКП запись... Считывается КЛП запись... 1+4*9-5 Одна из строк пустая	Проверка на корректность работы с пустыми строками
4.	1515151515 1+4*9-5	Считывается ЛКП запись... 1515151515 Считывается КЛП запись... 1+4*9-5 Некорректная запись	Проверка на корректность работы с неверными данными

Выводы.

В ходе работы была освоена реализация бинарного дерева на основе рекурсии и структур, отработано понимание его применения, и отработаны навыки письма в C++.

Код программы можно найти в приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <fstream>
#include <cstdlib>
#include "./src/binTree.h"
#include <string>

using namespace std;

void displayBT(binTree *tree, int b, int n)
{ // n — уровень узла
  if (b != 0) {
    cout << ' ' << tree->RootBT(b);
    if (!tree->isNull(tree->Right(b))) {
displayBT(tree, tree->Right(b), n + 1);
}
    else
cout << endl; // вниз

    if (!tree->isNull(tree->Left(b))) {
      for (int i = 1; i <= n; i++) cout << " "; // вправо
      displayBT(tree, tree->Left(b), n + 1);
    }
  }
  else {};
}

int build(binTree *tree, string &a, string &b, int x) {
string sub_a, sub_b;
  if (a == "") return 0;

  if (a.length() == 1) {
int p = tree->ConsBT(a[0]);
    return p;
  }
  if (x < 0) return 0;

  int flag = a.find(b[x]);
  if (flag < 0) return 0;

int r = tree->ConsBT(b[x]);

sub_a = a.substr(0, flag);
sub_b = b.substr(x+1, flag);
  tree->arr[r].lt = build(tree, sub_a, sub_b, x);

sub_a = a.substr(flag + 1);
```

```
sub_b = b.substr(b.size() - sub_a.size());
  tree->arr[r].rt = build(tree, sub_a, sub_b, x);
  return r;
}

void lpk(binTree *tree, int b, string &str) {
if(tree->isNull(b)) return;
lpk(tree, tree->arr[b].lt, str);
lpk(tree, tree->arr[b].rt, str);
str += tree->arr[b].info;
}

int main() {
binTree *MyTree = new binTree();
  ifstream fin("input.txt");
  if (!fin) { cout << "File not open for reading!\n"; return 1; }
  string str1, str2, str3;
  cout << "Считывается ЛКП запись..." << endl;
  getline(fin, str1, '\n');
cout << str1 << endl;
  cout << "Считывается КЛП запись..." << endl;
  getline(fin, str2, '\n');
cout << str2 << endl;
  build(MyTree, str1, str2, 0);
lpk(MyTree, 1, str3);
cout << "ЛПК: " << str3 << endl;
cout << "В виде дерева:" << endl;
displayBT(MyTree, 1, 1);
  return 0;
}
```

Название файла: binTree.h

```
#pragma once
#include <iostream>
#include <cstdlib>
using namespace std;

class binTree {
public:
struct node {
char info;
int lt;
int rt;
node(int inputlt = 0, int inputrt = 0) : lt(inputlt), rt(inputrt) {};
};
node *arr;
binTree(int len = 100) {
arr = new node[len];
for(int i = 0; i < len-1; i++){
arr[i].lt = i+1;
}
arr[len].lt = 0;
```

```
}
bool isNull(int b) {
return (b == 0);
}
char RootBT(int b) {
if (isNull(b)) {
cerr << "Error: RootBT(null) \n";
return 0;
}
else return arr[b].info;
}
int Left(int b) {
if (isNull(b)) {
cerr << "Error: Left(null) \n";
return 0;
}
else return arr[b].lt;
}
int Right(int b) {
if (isNull(b)) {
cerr << "Error: Right(null) \n";
return 0;
}
else return arr[b].rt;
}
int ConsBT(char b) {
int p = arr[0].lt;
arr[0].lt = arr[p].lt;
arr[p].info = b;
arr[p].lt = 0;
arr[p].rt = 0;
return p;
}
~binTree() {
delete[] arr;
}
};
```