**Частное учреждение образования**

**«Гродненский колледж бизнеса и права»**

**Лабораторная работа № 5**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Решение задач с помощью деревьев оптимального поиска

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 5**

Тема: Решение задач с помощью деревьев оптимального поиска.

Цель: Сформировать навыки построения бинарного дерева. Научиться решать задачи с помощью деревьев оптимального поиска.

Задачи: Ознакомление с примерами, выполнение индивидуальной работы.

**ЗАДАЧИ**

Условие: Создайте программу обхода дерева по принципу правый – корень – левый.

Алгоритм: Предоставлен в письменном виде преподавателю.

Решение:

**uses** crt;

**type**

PNode = ^Node;{Указатель на узел}

Node = **record** {Тип запись в котором будет храниться информация}

data: integer;

left, right: PNode; {Ссылки на левого и правого сыновей}

**end**;

**var**

Tree, p1: PNode; {Tree адрес корня дерева, p1-вспомогательная переменная}

n, x, i: integer;

ch: char;{для работы менюшки}

{Процедура добавления элемента }

**procedure** AddToTree(**var** Tree: PNode; x: integer);{Входные параметры - адрес корня дерева и добавь элемент }

**begin**

**if** Tree = nil **then** {Если дерево пустое то создаём его корень}

**begin**

**New**(Tree); {Выделяем память }

Tree^.data := x; {Добавляем данные }

Tree^.left := nil; {Зануляем указатели на левого }

Tree^.right := nil; {и правого сыновей }

**exit**;

**end**;

**if** x < Tree^.data **then** {Доб к левому или правому поддереву это завсит от вводимого элемента}

AddToTree(Tree^.left, x) {если меньше корня то в левое поддерево }

**else**

AddToTree(Tree^.right, x); {если больше то в правое}

**end**;

{Функция поиска по дереву}

**function** Search(Tree: PNode; x: integer): PNode;{Возвращает адрес искомого элемента, nil если не найден}

**var**

p: PNode;{вспомог переменнная }

**begin**

**if** Tree = nil **then** {если дерево пустое то}

**begin**

Search := nil; {присваеваем функции результат нил}

**exit**; {и выходим }

**end**;

**if** x = Tree^.data **then** {если искомый элемент равен корню дерева то }

p := Tree {запоминаем его адрес }

**else** {иначе}

**if** x < Tree^.data **then** {если вводимый элемент меньше корня то }

p := Search(Tree^.left, x) {то ищем его в левом поддереве}

**else** {иначе }

p := Search(Tree^.right, x); {ищем в правом поддереве }

Search := p; {присваиваем переменной с именем функции результат работы}

**end**;

{Обход дерева по принципу Левый-корень-правый }

**procedure** Lkp(Tree: PNode);

**begin**

**if** Tree = nil **then** {Если дерево пустое }

**exit**; {то выход }

Lkp(Tree^.right); {обходим правое поддерево}

write(' ', Tree^.data); {выводим данные }

Lkp(Tree^.left); {иначе начинаем обход с левого подднрева}

**end**;

{Процедура удаления дерева}

**procedure** DeleteTree(**var** Tree1: PNode );

**begin**

**if** Tree1 <> nil **then**

**begin**

DeleteTree(Tree1^.LEFT);

DeleteTree(Tree1^.RIGHT);

Dispose(Tree1);

**end**;

**end**;

{основная пограмма}

**begin**

Tree := nil;

**repeat**{цикл для нашего меню}

Writeln('Выберете действие ');

Textcolor(2);

Writeln('Доступны следующие пункты:');

Writeln('1) Создание дерева поиска');

Writeln('2) Поиск элемента в дереве');

Writeln('3) Вывод дерева');

Writeln('4) ‚Выход');

writeln;

readln(ch); {ожидаем нажатия клавиши}

**case** ch **of** {выбираем клавишу}

'1':

**begin**

writeln(' kolv elementov');

readln(n);

**for** i := 1 **to** n **do**

**begin**

writeln('Введите число');

readln(x);

AddToTree(Tree, x);

**end**;

**end**;

'2':

**begin**

writeln('Элемент для поиска');

readln(x);

p1 := Search(Tree, x);

**if** p1 <> nil **then**

writeln('Найден')

**else** writeln('Такого элемента нет!');

**end**;

'3':

**begin**

writeln('Само дерево');

Lkp(Tree);

writeln;

**end**;

**end**;

**until** ch = '4';

DeleteTree(Tree);

**end**.