**Частное учреждение образования**

**«Гродненский колледж бизнеса и права»**

**Лабораторная работа № 4**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Построение бинарного дерева. Обход дерева

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 4**

Тема: Построение бинарного дерева. Обход дерева.

Цель: Сформировать навыки построения бинарного дерева. Обход дерева.

Задачи: Освоение теоретического материала и выполнение индивидуального задания.

**ЗАДАЧИ**

Условие:

Алгоритм: Предоставлен преподавателю в письменном виде.

Решение:

**type**

{Основные данные узла дерева (ключ).}

TData = String;

{Указатель на узел.}

TPNode = ^TNode;

{Узел дерева.}

TNode = **record**

Data : TData; {Основные данные узла дерева (ключ).}

PLeft, PRight : TPNode; {Указатели на левый и правый узел.}

**end**;

{Рекурсивная процедура для освобождения памяти, занятой деревом. (Удаление дерева).}

**procedure** TreeFree(**var** aPNode : TPNode);

**begin**

**if** aPNode = nil **then Exit**;

**if** aPNode^.PLeft <> nil **then** TreeFree(aPNode^.PLeft);

**if** aPNode^.PRight <> nil **then** TreeFree(aPNode^.PRight);

Dispose(aPNode);

aPNode := nil;

**end**;

{Рекурсивная функция поиска повтора значения, либо добавления узла, если

повтор не найден.

Функция возвращает значение:

True - узел со значением aData добавлен в дерево, повтор НЕ найден.

False - дерево осталось без изменений, повтор найден.}

**function** Add(**var** aPNode : TPNode; **const** aData : TData) : Boolean;

**begin**

**if** aPNode = nil **then begin**

**New**(aPNode);

aPNode^.Data := aData;

aPNode^.PLeft := nil;

aPNode^.PRight := nil;

Add := True;

**end else if** aData < aPNode^.Data **then**

Add := Add(aPNode^.PLeft, aData)

**else if** aData > aPNode^.Data **then**

Add := Add(aPNode^.PRight, aData)

**else**

Add := False;

**end**;

**procedure** Lkp(**var** aPNode : TPNode);

**begin**

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

**exit**; {то выход }

Lkp(aPNode^.Pright); {обходим правое поддерево}

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

Lkp(aPNode^.Pleft); {иначе начинаем обход с левого подднрева}

**end**;

**var**

PTree : TPNode;

Data : TData;

Cmd, Code : Integer;

S : String;

**begin**

{Начальная инициализация дерева.}

PTree := nil;

**repeat**

{Меню.}

Writeln('Выберите действие:');

Writeln('1: Построение дерева и поиск повторов.');

Writeln('2: Проверить дерево на пустоту.');

Writeln('3: Удалить дерево.');

Writeln('4: Вывод дерева.');

Writeln('5: Выход.');

Write('Введите команду: ');

Readln(Cmd);

**case** Cmd **of**

1:

**begin**

Writeln('Построение дерева и поиск повторов.');

**repeat**

Write('Задайте ключ (прервать ввод - пустая строка + Enter): ');

Readln(S);

**if** S <> '' **then**

**begin** {Если пользователь ввёл непустую строку.}

Data:=s;

**begin** {Если число задано правильно.}

**if** Add(PTree, Data) **then** {Если добавлен узел (значит повтор не найден).}

Writeln('Повтор не найден. Узел добавлен в дерево.')

**else** {Если узел не был добавлен (значит повтор найден).}

Writeln('!!! Найден повтор. Дерево осталось без изменений.');

Code := 1; {=1 - чтобы продолжить цикл работы с деревом.}

**end**;

**end**

**else**

**begin** {Если пользователь ввёл пустую строку.}

Writeln('Отмена.');

Code := 0; {Обнуляем код, чтобы выйти из цикла работы с деревом.}

**end**;

**until** Code = 0;

**end**;

4:

**begin**

writeln('Само дерево');

Lkp(PTree);

writeln;

**end**;

2:

**if** PTree <> nil **then**

Writeln('Дерево не пустое.')

**else**

Writeln('Дерево пустое.');

3, 5:

**begin**

TreeFree(PTree);

Writeln('Дерево удалено.');

**end**;

**else**

Writeln('Незарегистрированная команда. Повторите ввод.');

**end**;

**until** Cmd = 5;

Writeln('Работа программы завершена. Для выхода нажмите Enter.');

Readln;

**end**.

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Бинарное дерево-это конечное множество элементов, которое либо пусто, либо содержит один элемент, называемый корнем дерева, а остальные элементы множества делятся на два непересекающихся подмножества, каждое из которых само является бинарным деревом.
2. Каждый элемент бинарного дерева называется узлом дерева.
3. Существует 3 способа обхода бинарного дерева:  
   в прямом порядке;  
   в симметричном порядке;  
   в обратном порядке.
4. В прямом порядке:  
   Попасть в корень;  
   Пройти в прямом порядке левое поддерево;  
   Пройти в прямом порядке правое поддерево;  
   В симметричном порядке:  
   Пройти в симметричном порядке левое поддерево;  
   Попасть в корень;  
   Пройти в симметричном порядке правое поддерево;  
   В обратном порядке:  
   Пройти в обратном порядке левое поддерево;  
   Пройти в обратном порядке правое поддерево;  
   Попасть в корень.