Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

Учебная дисциплина «Структуры и алгоритмы обработки данных»

**Инструкция**

по выполнению лабораторной работы

«Реализация алгоритмов работы с бинарными деревьями»

Минск

2018

**Лабораторная работа № 13**

**Тема работы: «Реализация алгоритмов работы с бинарными деревьями»**

**1. Цель работы**

Формирование умений работы с нелинейной структурой данных «бинарное дерево».

**2. Задание**

Дана последовательность чисел. Построить бинарное дерево, содержащее эти числа.

**3. Оснащение работы**

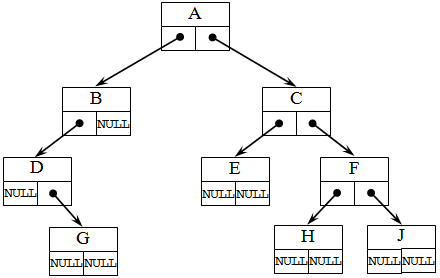
ПК, Pascal.

**4. Основные теоретические сведения**

**Бинарное (двоичное) дерево** – это динамическая структура данных, представляющая собой дерево, в котором каждая вершина имеет не более двух потомков.

Структура дерева отражается во входном потоке данных так: каждой вводимой пустой связи соответствует условный символ, например, **'\*'**(звездочка). При этом сначала описываются левые потомки, затем, правые.

Для структуры бинарного дерева, представленного на [рисунке ниже](https://www.intuit.ru/studies/courses/648/504/lecture/11458?page=1#image.31.6), входной поток имеет вид: **ABD\*G\*\*\*CE\*\*FH\*\*J\*\***.



Пример построения бинарного дерева:

**Program trees;**

**Type shablon=^zapis;**

**Zapic = record**

**A: integer;**

**Left: shablon;**

**Right: shablon;**

**End;**

**Var**

**Tree1, tree2: shablon;**

**Max: integer;**

**Procedure Create\_Tree (tree:shablon; n:integer);**

**var**

**New\_tree:shablon;**

**Begin**

**If n<=max then**

**begin**

**{переход на левый узел}**

**New (tree^.left);**

**New\_tree:= tree^.left;**

**New\_tree^.a:=random(10);**

**Create\_Tree (New\_tree, n+1);**

**{переход на правый узел}**

**New (tree^.right);**

**New\_tree:=tree^.right;**

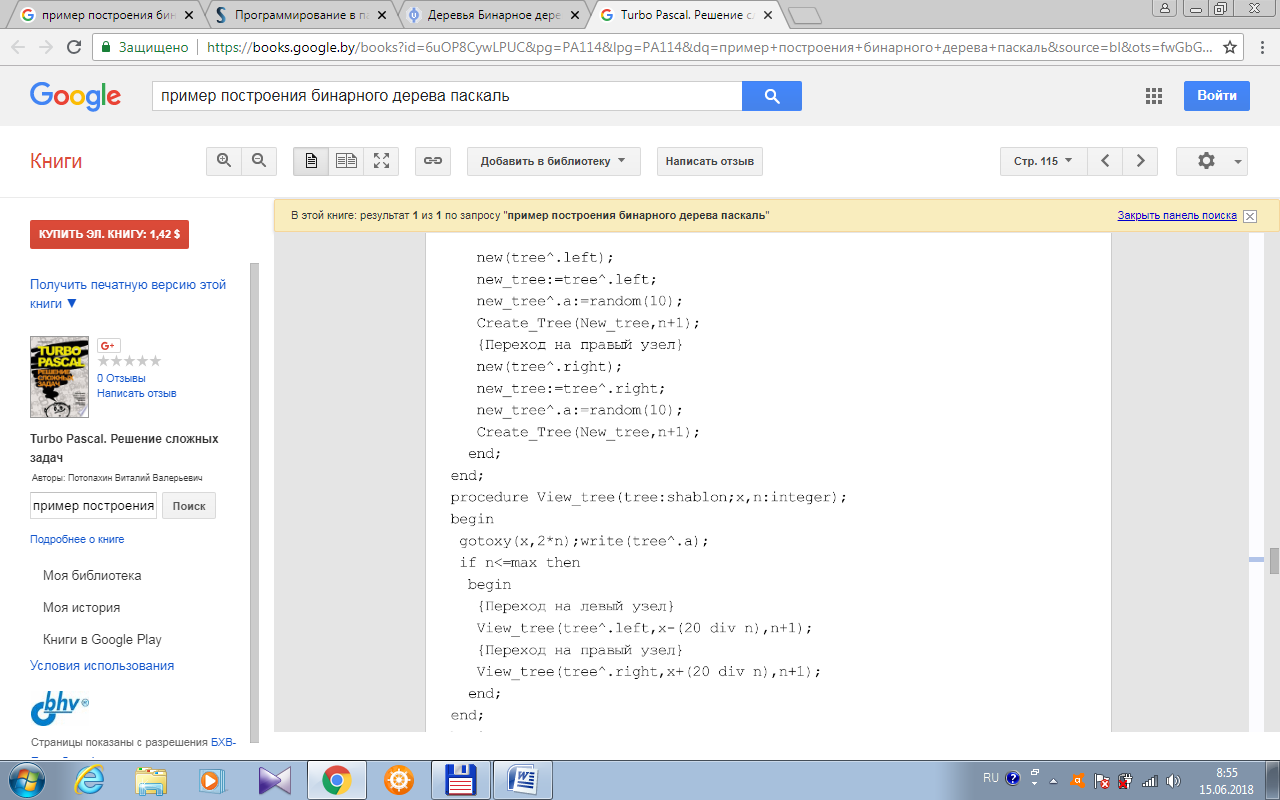
**New\_tree^.a:=random (10);**

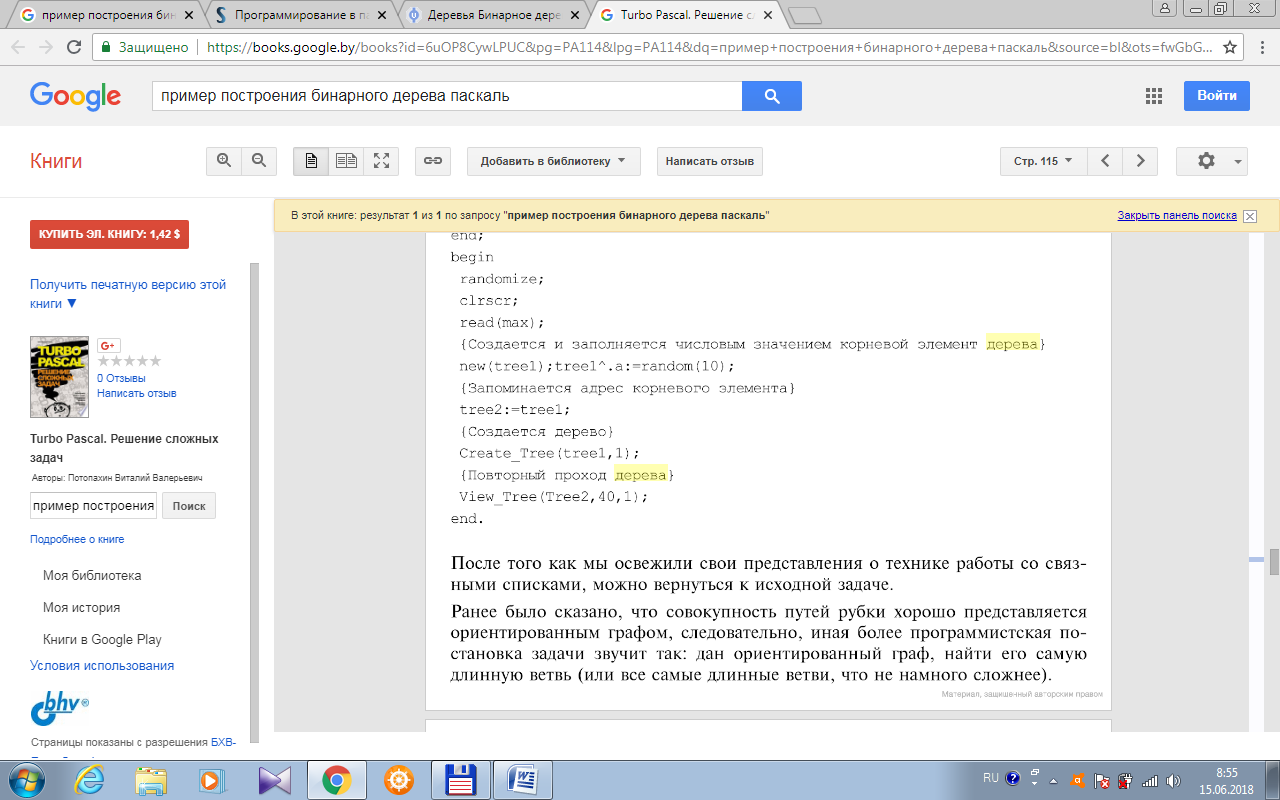
**Create\_Tree (New\_tree, n+1);**

**End;**

**End;**

**End.**



 Многие алгоритмы, использующие бинарные деревья, включают два этапа. На первом этапе строится бинарное дерево, а на втором выполняется систематический обход его вершин.

Наиболее часто используются следующие способы обхода вершин дерева:

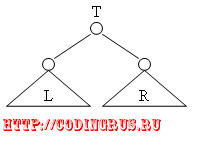
1) Обход в прямом порядке (просмотр в глубину).

2) Обход в обратном порядке.

3) Симметричный обход (обход во внутреннем порядке).

Все три способа описываются рекурсивными алгоритмами.

Бинарное дерево схематически можно представить следующим образом:



T – выделенная вершина – корень, L – левое поддерево, R – правое поддерево.

Тогда для прямого обхода используется алгоритм TLR. Он включает следующие шаги:

1) просмотр корня T;

2) просмотр с помощью алгоритма TLR левого поддерева;

3) просмотр с помощью алгоритма TLR правого поддерева.

Обход в обратном порядке реализуется с помощью алгоритма LRT:

1) обход левого поддерева с помощью алгоритма LRT;

2) обход правого поддерева с помощью алгоритма LRT;

3) просмотр корня T.

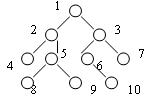
Симметричный обход реализуется с помощью алгоритма LTR:

1) просмотр левого поддерева с помощью алгоритма LTR;

2) просмотр корня T;

3) просмотр правого поддерева с помощью алгоритма LTR.

Пример:



Прямой порядок: TLR: 1,2,4,5,8,9,3,6,10,7.

Обратный порядок: LRT: 4,8,9,5,2,10,6,7,3,1.

Симметричный порядок: LTR: 4,2,8,5,9,1,6,10,3,7.

Рассмотрим реализацию этих алгоритмов в программах. Пусть соответствующая процедура обхода вершин некоторого дерева T печатает метки этих вершин.

INFO(i,T)

p – указатель на корень дерева

q – указатель на другую вершину

LEFT(i,T) – определение левого потомка вершины i в дереве T.

RIGHT(i,T) – определение правого потомка вершины i в дереве T.

Функции возвращают номер вершины.

INFO(i,T) – возвращает информацию, приписанную вершине i дерева T.

**Procedure TLR (p:position);**

**Var q:position;**

**Begin**

**If p<>NIL then**

**Begin**

**write(INFO(p,T));**

**q:=LEFT(p,T); TLR(q);**

**q:=RIGHT(p,T); TLR(q);**

**end;**

**end;**

**Procedure LRT (p:position);**

**Var q:position;**

**Begin**

**If p<>NIL then**

**Begin**

**q:=LEFT(p,T); LRT(q); q:=RIGHT(p,T); LRT(q);**

**write(INFO(p,T));**

**end;**

**end;**

**Procedure LTR (p:position);**

**Var q:position;**

**Begin**

**If p<>NIL then**

**Begin**

**q:=LEFT(p,T); LTR(q); write(INFO(p,T));**

**q:=RIGHT(p,T); LTR(q); end;**

**end;**

**5. Порядок выполнения работы**

1. Построить бинарное дерево, содержащее заданные числа.

2\*. Произвести обход дерева.

3. После выполнения программы очистить память, занятую древовидной структурой.

**6. Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7. Контрольные вопросы и задания**

1. С чем связана популярность использования деревьев в программировании?

2. Можно ли список отнести к деревьям? Ответ обоснуйте.

3. Какие данные содержат адресные поля элемента бинарного дерева?

4. Может ли бинарное дерево быть строгим и неполным? Ответ обоснуйте.

5. Может ли бинарное дерево быть нестрогим и полным? Ответ обоснуйте.

6. Каким может быть почти сбалансированное бинарное дерево: полным, неполным, строгим, нестрогим? Ответ обоснуйте.

7. Чем отличаются, с точки зрения реализации алгоритма, прямой, симметричный и обратный обходы бинарного дерева?

**8. Рекомендуемая литература**

1.Ахо, А.В.Структуры данных и алгоритмы / А.В. Ахо, Дж.Хопкрофт, Дж. Д.Ульман. – пер. с англ. – М.: Издательский дом "Вильямс", 2007.-400 с.

2. Вирт,Н. Алгоритмы и структуры данных / Н. Вирт. –СПб.:Невский диалект, 2008. – 352с.

3. Вирт, Н. Алгоритмы и структуры данных с примерами на Паскале / Н. Вирт[перевод с английского Д. Б. Подшивалова]; – 2-е изд., испр., – СПб.: Невский Диалект, 2005. – 352с.

4. Гагарина, Л.Г. Алгоритмы и структуры данных / Л.Г. Гагарина, В.Д. Колдаев. – учеб.пособие – М: Финансы и статистика, 2009. – 304с.

5. Котов, В.М. Алгоритмы и структуры данных: учеб.пособие / В.М. Котов, Е.П. Соболевская, А.А. Толстиков – Минск: БГУ, 2011. – 267с.

6. Макконнелл, Дж. Основы современных алгоритмов / Дж. Макконнелл – 2-е дополненное издание – М.:Техносфера, 2006. – 368с.

7. Окулов, С.М. Программирование в алгоритмах / С.М. Окулов. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2007. – 383с.