**ИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Ухтинский государственный технический университет»**

**(УГТУ)**

Кафедра вычислительной техники, информационных систем и технологий

**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА № 2**

Дисциплина: «Алгоритмы и структуры данных»

Шифр 191407 Группа ИСТ-2-19 Курс 1

Морданов Егор Владимирович

Проверил:

старший преподаватель кафедры ВТИСиТ С. С. Сочко

Ухта

2020

Содержание

[ЗАДАНИЕ 1 3](#_Toc43427687)

[ЗАДАНИЕ 2 5](#_Toc43427688)

[ЗАДАНИЕ 3 10](#_Toc43427689)

[ПРИЛОЖЕНИЕ 12](#_Toc43427690)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 15](#_Toc43427691)

# ЗАДАНИЕ 1

*Задача № 1*

**Условие задачи**

Построить дерево арифметического выражения: .

Записать все три формы польской записи.

**Решение**

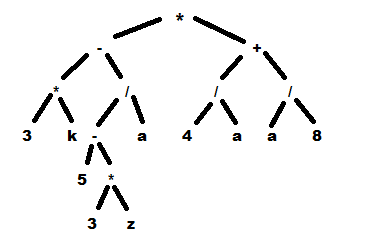


Рисунок - Дерево арифметического выражения

**Польская запись** — это форма записи логических, арифметических и алгебраических выражений.

Существует 3 формы польской записи:

* Инфиксная (обычная форма записи, предполагающая запись знака операции между операндами);
* Префиксная (запись, при которой обозначения операции записываются до операндов);
* Постфиксная (обратная запись).

**Infix:** 3 \* k – 5 – 3 \* z / a \* 4 / a + a / 8

**Prefix:** \* – \* 3 k / – 5 \* 3 z a + / 4 a / a 8

**Postfix:** 3 k \* 5 3 z \* – a / – 4 a / a 8 / + \*

*Задача № 2*

**Условие задачи**

По заданной постфиксной форме восстановить арифметическое выражение: x y \* z \* x / z y \* + 5 + .

**Решение**

Для того, чтобы было проще восстановить арифметическое выражение, по заданной постфиксной форме построим дерево.

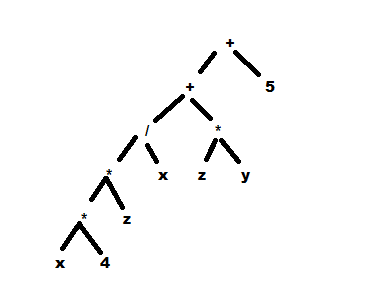


Рисунок - Дерево постфиксного выражения

**Арифметическое выражение:**

# ЗАДАНИЕ 2

**Условие задачи**

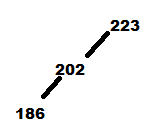
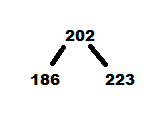
Для заданной последовательности ключей 223 ,202, 186, 123, 101, 256, 234, 289, 233, 245, 156, 170, 200, 201, 186, 213, 156, 101, 124, 115 построить:

* Построить АВЛ-дерево. Подписать все повороты и отметить вершины, в которых был нарушен баланс.
* Построить В-дерево 2-го порядка.
* Построить В+-дерево 2-го порядка. В листовую страницу помещается не более 3 элементов.

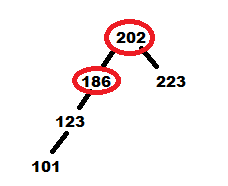
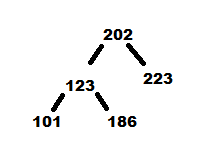
**Решение**

1. Построение АВЛ-дерева

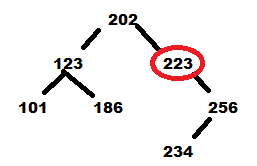
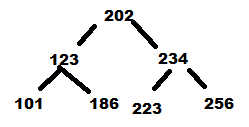
Корень дерева 223. Добавляем 202, 186. Баланс нарушился в корне 223. Делаем R-поворот.

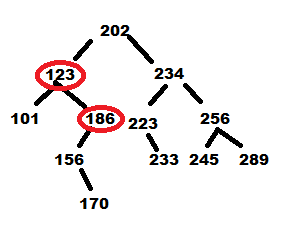
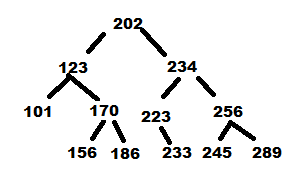
Добавляем 123, 101. Баланс нарушился в корне дерева и вершине 186. Делаем R-поворот.

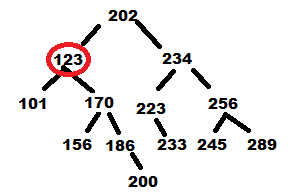
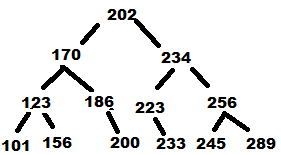
Добавляем 256,234. Баланс нарушился в вершине 223. Делаем RL-поворот.

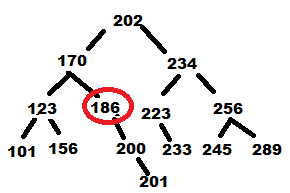
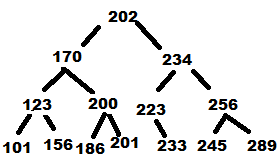
Добавляем 289, 233, 245,156,170. Баланс нарушился в вершине 186 и 123. Делаем LR-поворот.

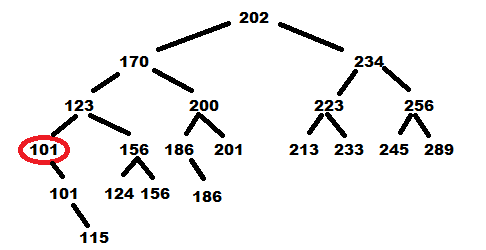
Добавляем 200. Баланс нарушился в вершине 123. Делаем LR-поворот.

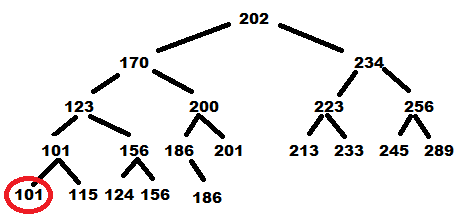
 

Добавляем 201. Баланс нарушился в вершине 186. Делаем L-поворот.

Добавляем 186,213,156,101,124,115. Баланс нарушился в вершине 101. Делаем L-поворот.





Отмеченный красным лист следуют удалить, т.к. он копирует предыдущую вершину по значению и будет иметь «практически» такой же код.

1. Построение B-дерева второго порядка

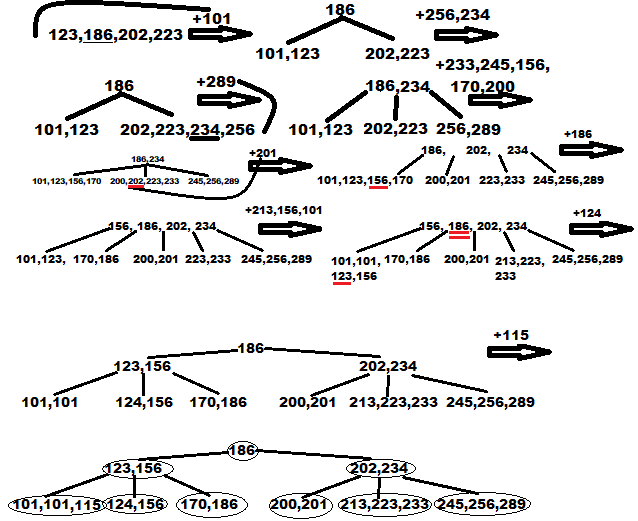


Рисунок – B-дерево

1. Построение B+-дерева второго порядка

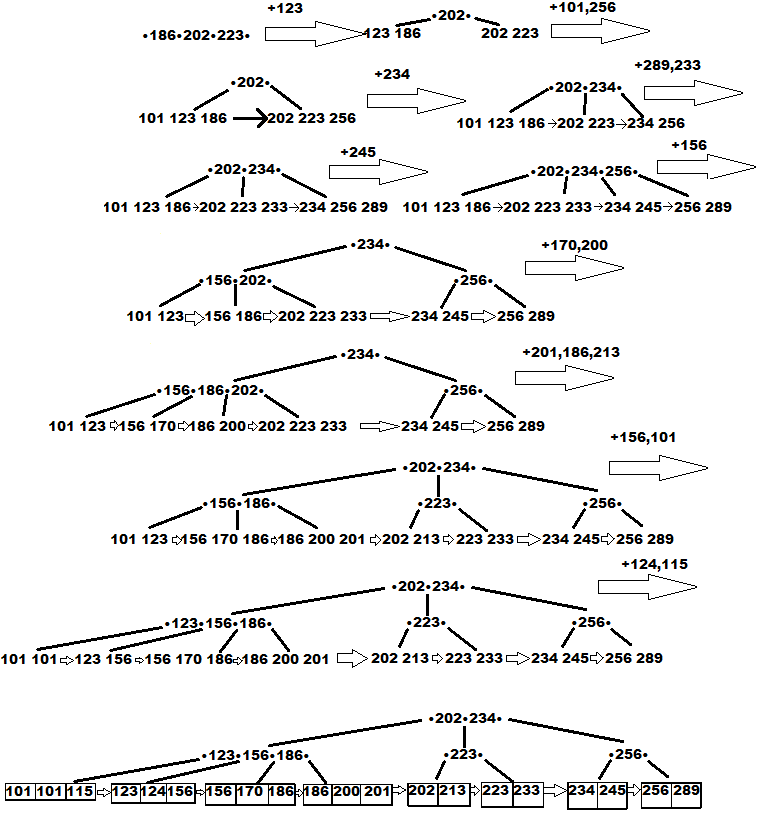


Рисунок - B+-дерево

# ЗАДАНИЕ 3

**Условие задачи**

Написать программу к задаче.

Дан текстовый файл, содержащий сведения о телефонных абонентах: ФИО, номер телефона, дата установки. Отсортировать данные файла с помощью сортированного дерева по ФИО.

**Решение**

Прежде всего определим интерфейс нашей программы. Предоставим пользователю следующие возможности:

* Чтение и сортировка данных;
* Вывод сортированных данных;
* выход из программы;

Когда изменения в одной процедуре не могут влиять на поведение другой, отладка и модификация программы значительно упрощаются. Поэтому при определении интерфейсов подпрограмм будем исходить из того, что подпрограммы должны быть независимы друг от друга. Для этого все, что подпрограмме необходимо получать извне, будем передавать через параметры, а величины, используемые только внутри подпрограммы, определим как локальные.

Список основных процедур и функций для работы программы

* Функция для определения является ли данный символ цифрой, используется для правильного добавления данных, так как наш текстовой файл не является типизированным:

function IsDigit(c:char):boolean;

* Процедура добавления данных в программу, процедура считывает данные и сортирует их в дерево по ФИО:

procedure add(var root:^tree;var f:text);

* Процедура инициализации дерева, которая определяет, что корень дерева равен nil:

procedure init(var root:^tree);

* Процедура вывода дерева, которое мы считали с текстового файла, используется рекурсивный метод вывода дерева:

procedure preorder(p:^tree;st:string);

* Процедура для открытия файла для чтения для последующего добавления данных в программу, после достижения конца файла закрывает его чтение:

procedure readfile();

Дерево — это совокупность элементов, называемых узлами, и отношений «предок-потомок», образующих иерархическую структуру узлов.

* Узлы, которые не имеют ни одного последующего узла, называются листьями.
* У каждого узла только один предок. Единственный узел, у которого нет предка, называется корнем дерева.
* Количество потомков внутренней вершины (не лист) определяет степень вершины. Максимальная степень всех вершин дерева является степенью дерева.

Для представления данных, среди которых идет поиск элементов по заданному ключу используют двоичное дерево, которое называется **двоичным деревом поиска** или **упорядоченным деревом**. Формируется это дерево по следующему правилу:

ключевое поле левого поддерева всегда должно содержать значение меньше, чем в корне, а ключевое поле правого поддерева – значение больше или равное значению в корне.

ПРИЛОЖЕНИЕ

Листинг 1 – Описание типа

|  |
| --- |
| **type** tree=**record**  info:**record**  fio:string;  number:string;  date:string;  **end**;  left:^tree;  right:^tree;  **end**;  **var** root:^tree; f:text; nummenu:integer; st:string; |

Листинг 2 – Основная программа

|  |
| --- |
| **function** IsDigit(c:char):boolean;  **begin**  IsDigit:=c **in** ['0'..'9']  **end**;  **procedure** init(**var** root:^tree);  **begin**  root:=nil;  **end**;  **procedure** add(**var** root:^tree;**var** f:text);  **var** t,pp,p:^tree; c:char; buff:string;  **begin**  **new**(t);  **repeat**  buff:=buff+c;  read(f,c);  **until** IsDigit(c);  t^.info.fio:=buff;  buff:='';  **repeat**  buff:=buff+c;  read(f,c);  **until** c =' ';  t^.info.number:=buff;  read(f,t^.info.date);  t^.left:=nil;  t^.right:=nil;  **if** root=nil **then**  root:=t  **else begin**  p:=root;  **while** p<>nil **do begin**  pp:=p;  **if** t^.info.fio<p^.info.fio **then**  p:=p^.left  **else** p:=p^.right;  **end**;  **if** t^.info.fio<pp^.info.fio **then**  pp^.left:=t |

Продолжение листинга 2

|  |
| --- |
| **else**  pp^.right:=t;  **end**;  **end**;  **procedure** preorder(p:^tree;st:string);  **begin**  **if** p<>nil **then begin**  writeln(st,p^.info.fio, ' | ', p^.info.number, ' | ', p^.info.date);  st:=st+' ';  preorder(p^.left,st);  preorder(p^.right,st);  **end**;  **end**;  **procedure** menu;  **begin**  writeln('1-Чтение');  writeln('2-Вывод');  writeln('3-Выход');  **end**;  **procedure** readfile();  **begin**  reset(f);  **while not** eof(f) **do**  **while not** eoln(f) **do begin**  add(root,f);  readln(f)  **end**;  writeln('Чтение завершено');  close(f);  **end**;  **begin**  assign(f,'text.txt');  init(root);  **repeat**  menu;  readln(nummenu);  clrscr;  **case** nummenu **of**  1:readfile();  2:preorder(root,st);  **end**;  **until** nummenu=3;  **end**. |

Протокол выполнения программы



Рисунок 5 – Меню программы

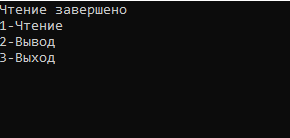


Рисунок 6 – Выполнение чтения

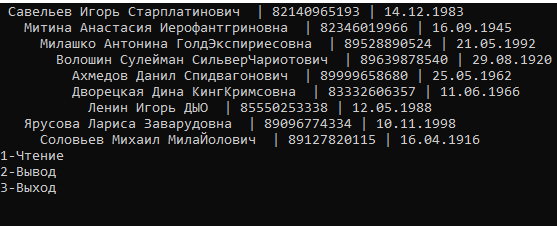


Рисунок 7 – Вывод отсортированного списка данных по ФИО

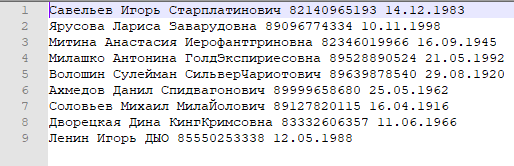


Рисунок 8 – Исходные данные в текстовом файле

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Кормен , Х., Лейзерсон, Ч. И. и др «Алгоритмы:построение и анализ»;
2. Сочко, С. С. Абстрактные структуры данных. Практикум [Текст] : учеб. пособие / С. С. Сочко. ‒ Ухта : УГТУ, 2019 – 104 с.;
3. Андреева, Т. А. Программирование на языке Pascal [Электронный ресурс] / Т. А. Андреева. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 277 c.;
4. Павловская, Т. А. Паскаль. Программирование на языке высокого уровня: Практикум. – СПб.: Питер, 2006. – 317 с.