Пятое задание для студентов 1 курса кафедры ИУС ФТК.

Здесь приведены задания ,которые надо реализовать с применением двоичных деревьев и рекурсии.

1

Разработать процедуры:
1) P1 – итеративного формирования бинарного дерева поиска из string [10] (фамилии);
2) P2 – рекурсивного вывода дерева на экран (по вертикали);
3) P3 - выгрузки списка фамилий из дерева по алфавиту в линейный список.
Вывод линейного списка в выходной файл сделать в головной программе.

2

Разработать процедуры:
1.P1 - формирования дерева поиска из целых;
2) P2 – определения высоты дерева, используя итерацию;
3) P3 - обхода двоичного дерева по правилу TREE-LEFT-RIGHT и вывода его элементов на экран с показом подчиненности его элементов.
В головной программе проверить работоспособность этих процедур на тестах.

3

В файле дана последовательность целых чисел.
Построить из них двоичное дерево поиска .
Разработать процедуру перестройки двоичного дерева с целью сделать его более низким и симметричным (задача балансировки дерева).
Использовать ее для балансировки сформированного ранее дерева.
Применить рекурсию.

4

В файле дана корректная запись логического выражения, содержащего AND, NOT, OR, скобки, имена (односимвольные) логических переменных.
Разработать рекурсивную процедуру вычисления значения логического выражения, используя двоичные деревья , если после записи логического выражения в файле записаны значения используемых переменных в виде :
a=true c=false и т.д.

5

Во входном файле задана префиксная форма логического выражения , содержащая односимвольные имена (большие латинские буквы) логических операндов , символы операций (малые буквы n-вместо not, a-вместо and и o-вместо or) . Разработать процедуры:
1.P1-проверки корректности записи префиксной формы;
2.P2-преобразования префиксной формы в скобочную инфиксную ;
В случае ошибочности исходной префиксной формы выдать сообщение об ошибке.

6

Во входном файле задана инфиксная форма логического выражения , содержащая односимвольные имена (большие латинские буквы) логических операндов , символы операций (малые буквы n-вместо not, a-вместо and и o-вместо or) и круглые скобки Разработать процедуры:
1.P1-проверки корректности записи инфиксной формы;
2.P2-преобразования инфиксной формы в префиксную.
В случае ошибочности исходной инфиксной формы выдать сообщение об ошибке

7

Во входном файле задана постфиксная форма логического выражения , содержащая односимвольные имена (большие латинские буквы) логических операндов , символы операций (малые буквы n-вместо not, a-вместо and и o-вместо or) Разработать процедуры:
1.P1-проверки корректности записи постфиксной формы;
2.P2-преобразования постфиксной формы в инфиксную.
В случае ошибочности исходной постфиксной формы выдать сообщение об ошибке

8

Во входном файле задана постфиксная форма логического выражения , содержащая односимвольные имена (большие латинские буквы) логических операндов , символы операций (малые буквы n-вместо not, a-вместо and и o-вместо or) Разработать процедуры:
1.P1-проверки корректности записи постфиксной формы;
2.P2-преобразования постфиксной формы в префиксную.
В случае ошибочности исходной постфиксной формы выдать сообщение об ошибке.

.

9

Во входном файле задана инфиксная форма логического выражения , содержащая односимвольные имена (большие латинские буквы) логических операндов , символы операций (малые буквы n-вместо not, a-вместо and и o-вместо or) и круглые скобки. Разработать процедуры:
1.P1-проверки корректности записи инфиксной формы;
2.P2-преобразования инфиксной формы в префиксную.
В случае ошибочности исходной инфиксной формы выдать сообщение об ошибке.

10

Во входном файле задана префиксная форма арифметического выражения , содержащая односимвольные имена (большие латинские буквы) арифметических операндов , символы операций +,-,*,/ без унарных операций. Разработать процедуры:
1.P1-проверки корректности записи префиксной формы арифметического

выражения;
2.P2-преобразования префиксной формы в скобочную инфиксную.
В случае ошибочности исходной префиксной формы выдать сообщение об ошибке.

11

Во входном файле задана префиксная форма арифметического выражения , содержащая односимвольные имена (большие латинские буквы) арифметических операндов , символы операций +,-,*,/ без унарных операций.
Разработать процедуры:
1.P1-проверки корректности записи префиксной формы арифметического выражения, используя рекурсию;
2.P2-преобразования постфиксную формы в префиксной.
В случае ошибочности исходной префиксной формы выдать сообщение об ошибке.

12

Во входном файле задана постфиксная форма арифметического выражения , содержащая односимвольные имена (большие латинские буквы) арифметических операндов , символы операций +,-,*,/ без унарных операций.
Разработать процедуры:
1.P1-проверки корректности записи постфиксной формы арифметического выражения, используя итерацию;
2.P2-преобразования постфиксной формы в префиксную, используя стеки.
В случае ошибочности исходной постфиксной формы выдать сообщение об ошибке.

13

Во входном файле задана инфиксная форма арифметического выражения , содержащая односимвольные имена (большие латинские буквы) арифметических операндов , символы операций +,-,*,/ без унарных операций.
Разработать процедуры:

1.P1-проверки корректности записи инфиксной формы арифметического выражения, используя итерацию;
2.P2-преобразования инфиксной формы в префиксную, используя деревья.
В случае ошибочности исходной инфиксной формы выдать сообщение об ошибке.

14

Во входном файле задана инфиксная форма арифметического выражения , содержащая односимвольные имена (большие латинские буквы) арифметических операндов , символы операций +,-,*,/ без унарных операций.
Разработать процедуры:
1.P1-проверки корректности записи инфиксной формы арифметического выражения, используя рекурсию;
2.P2-преобразования инфиксной формы в постфиксную, используя деревья.
В случае ошибочности исходной инфиксной формы выдать сообщение об ошибке.

15

Во входном файле задана корректная инфиксная форма арифметического выражения , содержащая односимвольные операнды в виде десятичных цифр , символы операций +,-,*,/ и круглые скобки. Унарные операции допускаются.
Разработать процедуры:
1.P1-преобразования инфиксной формы в постфиксную, используя деревья.
2.P2- вычисления корректной инфиксной формы арифметического выражения, используя рекурсию;
Применить эти процедуры к содержимому заданного файла.

16

Во входном файле задан список имен по одному на строке.
Разработать процедуры:
1.P1- итеративной записи этого списка в бинарное дерево поиска;
2.P2- рекурсивного вычисления длин всех ветвей этого дерева (расстояний от

корня до каждого из листьев) и вывода этой информации в текстовый файл.
Применить эти процедуры к содержимому заданного файла.

17

Во входном файле задана строка символов .
Разработать процедуры:
1.P1- итеративной записи этого списка в бинарное дерево поиска;
2.P2- итеративного вычисления длин всех ветвей этого дерева (расстояний от корня до каждого из листьев) и вывода этой информации в текстовый файл.
Применить эти процедуры к содержимому заданного файла.

18

Во входном файле задана строка символов .
Разработать процедуры:
1.P1- итеративной записи этого списка в бинарное дерево поиска;
2.P2- итеративного вычисления расстояния от корня до заданного элемента дерева и вывода этой информации в текстовый файл. Если заданного элемента нет в дереве , выдать соответствующую диагностику.
Применить эти процедуры к содержимому заданного файла.

19

Во входном файле задана строка из десятичных чисел .
Разработать процедуры:
1.P1- записи этих чисел в бинарное дерево поиска;
2.P2- итеративного вычисления числа элементов на каждом уровне дерева и

вывода этой информации в текстовый файл.
Применить эти процедуры к содержимому заданного файла.

20

Во входном файле заданы две строки из десятичных чисел .
Разработать процедуры:
1.P1- записи этих чисел из одной строки текстового файла в бинарное дерево поиска;
2.P2- итеративного вычисления числа элементов дерева и вывода этой информации в текстовый файл;
3.P3- сравнения числа элементов двух заданных деревьев.
В головной программе запустить дважды процедуру P1, сформировав два дерева из двух заданных во входном файле строк чисел. А затем обратиться к процедуре P3 , которая с помощью процедуры P2 произведет подсчет элементов этих деревьев и сравнит результаты.