

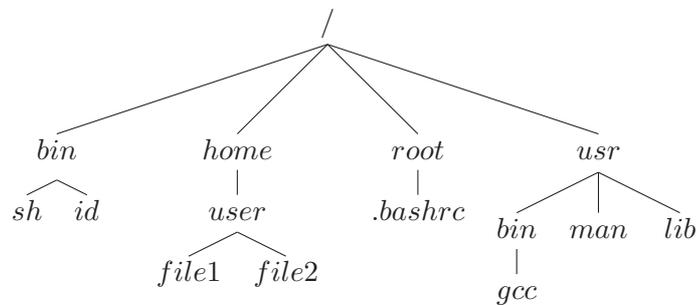
# 1 Задача

В программировании часто встречаются структуры, представимые в виде  $N$ -арного дерева, т.е. дерева, где каждый узел может содержать произвольное количество потомков.

Рассмотрим несколько примеров.

## 1.1 Дерево файловой системы

Практически любая файловая система в современных операционных системах представляет собой дерево, где узлами служат директории, а листьями – файлы. Например, корневой каталог Linux упрощённо можно представить так:



В стандартном пространстве .NET *System.IO* определён набор полезных классов для работы с файловой системой. Например, с помощью функции *GetDirectories* и *GetFiles*, можно “обходить” дерево каталогов и получать его содержимое.

## 1.2 Дерево XML-документа

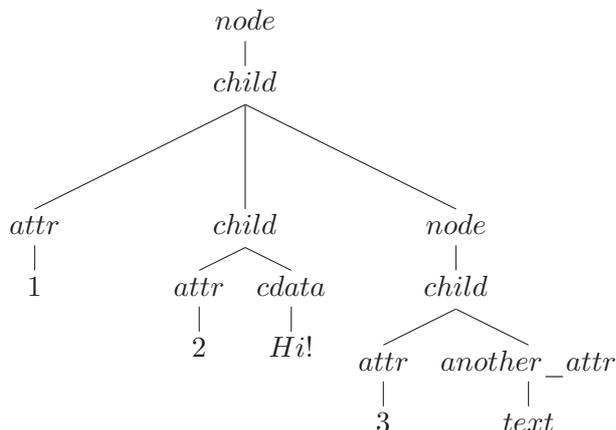
*XML* – язык разметки и текстовый формат, предназначенный для хранения структурированных данных. Синтаксически, XML-документ состоит из вложенных *элементов*, которые состоят из *тегов*, *аттрибутов* и *содержимого*.

Например:

```
<node>
  <child attr="1"/>
  <child attr="2">Hi!</child>
</node>
  <child attr="3" another_attr="text"/>
</node>
```

В документе из этого примера *node* и *child* – тэги элементов, *attr* и *another\_attr* – атрибуты, а строка “Hi!” – содержимое второго элемента *child*. Строчка “text” и цифры “1”, “2”, “3” – значения атрибутов.

Такой XML-документ легко представить в виде *N*-арного дерева:



Пространство имён *System.XML* .NET содержит набор классов и функций для работы с такими документами. Например, прочитать документ из файла можно следующим образом:

```

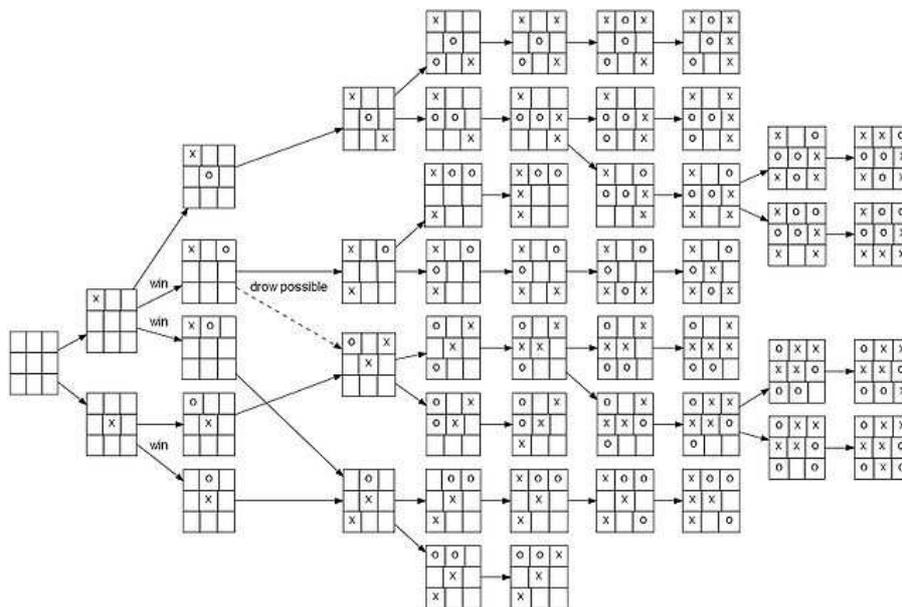
let doc = new XmlDocument() in
doc.LoadXml xml;
  
```

С помощью функции *SelectNodes* можно выбрать из документа интересные элементы.

### 1.3 Дерево вариантов игры в крестики-нолики

Игру крестики-нолики можно описать в виде дерева, где корнем будет пустое поле с девятью потомками – по одному на каждую клетку, куда может походить первый игрок. У каждого из узлов этого уровня, в свою очередь, будет по восемь потомков – варианты хода для второго игрока.

Частичное дерево игровых ситуаций будет иметь следующий вид:



#### 1.4 Синтаксическое дерево регулярного выражения

*Регулярными выражениями* называют формальный язык, предназначенный для поиска и манипуляций с подстроками в тексте, основанный на использовании метасимволов. По сути, это строка-образец (“шаблон” или “маска”), состоящая из обычных символов и метасимволов, которую можно применить к другой строке.

В качестве метасимволов могут выступать:

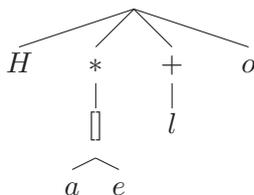
- \* – означает ноль или больше вхождений выражения, стоящего до метасимвола. Например, выражение  $a^*$  означает “ноль или несколько символов  $a$ ”, т.е. может быть применено к строкам “ $a$ ”, “ $aaaa$ ” и к пустой строке.
- + – означает *одно* или больше вхождений выражения. Т.е. выражение  $ab^+$  можно применить к “ $ab$ ” и “ $abbb$ ”.
- ? – означает ровно ноль или одно вхождение выражения.
- [] – означает любой из символов из находящихся в скобках. Например,  $[ab]$  применимо к  $a$  или к  $b$ .

Как и выражения на других формальных языках, регулярное выражение удобно записывать в виде синтаксического дерева, где в узлах дерева находятся операции (метасимволы), а в листьях – их аргументы.

Например, выражение

$$H[ae] * l + o \tag{1}$$

может быть представлено в виде дерева:



Его можно применить к строкам “Hello”, “Haaaallo” или, например, “Hlllllo”.

## 2 Задание

1. Из примеров, приведенных выше, возьмите файловую систему и ещё один из вариантов на выбор.
2. Для обоих вариантов реализуйте следующие функции-селекторы:
  - (a) *root* – получить вершину-корень дерева.
  - (b) *key t* – извлечь данные, хранящиеся в вершине *t*.
  - (c) *nchild t* – вернуть количество потомков *t*.
  - (d) *child t i* – вернуть вершину-*i*-го потомка *t*.

Не обязательно писать функции, работающие с реальными данными (с документом XML или с файловой системой). Достаточно реализовать функции, “эмулирующие” соответствующие структуры.

3. Напишите общую процедуру обхода произвольного дерева, которая принимает функции-селекторы и печатает на консоль все данные, которые встретит по пути.
4. Предполагая, что данные вершины – целое число, напишите функцию вычисления суммы элементов дерева.
5. Для дерева файловой системы реализуйте следующие функции:
  - (a) *filesize* – суммарный размер всех файлов в данной директории.
  - (b) *lastModified* – дата последнего изменения самого “свежего” файла в директории.