


XML Parser

Да се напише програма, която разчита XML файлове и позволява правенето на прости XPath 2.0 заявки към тях.

Забележка: За решението на задачата не е позволено използването на готови библиотеки за работа с XML. Целта на задачата е да се построи подходящо дървовидно представяне на данните в паметта и да се реализират операции с него.

Внимание: Не се изисква осигуряване на всички условия в XML и XPath спецификациите! Достатъчно е файловете да “приличат на XML” (както файла в примера, който не е валиден XML), а завките да “приличат” на XPath.

Минимални XML елементи, които да се поддържат:

- идентификатор на елемента
- списък от атрибути и стойности
- списък от вложени елементи или текст

Да се поддържат уникални идентификатори на всички елементи по следния начин:

- Ако елементът има поле “id” във входния файл и стойността му е уникална за всички елементи от файла, да се ползва тази стойност.
- Ако елементът има поле “id” във входния файл, но стойността му не е уникална за всички елементи от файла, да се ползва тази стойност, но към нея да се конкатенира някакъв низ, който да допълни идентификатора до уникален низ. (например, ако два елемента имат поле id=”1”, то единият да получи id=”1_1”, а другият - id=”1_2”)
- Ако елементът няма поле “id” във входния файл, да му се присъедини уникален идентификатор, генериран от програмата.

Минимални изисквания за поддържаните XPath заявки

Примерите по-долу са върху следния прост XML низ:

```
<person id="0">
      <name>John Smith</name>
      <address type="home">Brooklyn, NY</address>
      <address type="work">Manhattan, NYC</address>
      <occupation>dentist</occupation>
</person>
<person id=”1”>
      <name>Ivan Petrov</name>
      <address>Bulgaria</address>
</person>
```


- да поддържат оператора `/`
  Пример: `person/address` връща списък с всички елементи `address`, които са child елементи на `person`, за всеки елемент `person` във файла. За горния пример, това е списъка [`"Brooklyn, NY", "Manhattan, NYC", "Bulgaria"`]
- да поддържат оператора `[]` (например `person/address[0]` връща първия елемент `address`, който е child на `person`, за всеки елемент `person` във файла)
- да поддържат оператора `@` (например `person[@id]` връща списък с `id` атрибутите на всички елементи `person` във файла)
- Оператори за сравнение = (например `person[address="Bulgaria"]/name` връща списък с всички имена (съдържание на child елемента `name`) на елементи `person` във файла, които имат child елемент `address` с текстово съдържание `Bulgaria`)
- да поддържа функция `count()`, която връща броя на елементите, върнати от заявката.

Забележка: XPath адресите по-горе (от типа `person/*`) избират елементи/атрибути спрямо кореновия (root) елемент.

След като приложението отвори даден XML файл, то трябва да може да извършва посочените по-долу операции, в допълнение на общите операции (`open, close, save, save as`, `help` и `exit`):

`print`	Извежда на екрана прочетената информация от XML файла (в рамките на посочените по-горе ограничения за поддържаната информация). Печатането да е XML коректно и да е форматирано визуално по подходящ начин (например, подчинените елементи да са по-навътре).
`select <id> <key>`	Извежда стойност на атрибут по даден идентификатор на елемента и ключ на атрибута.
`set <id> <key> <value>`	Присвояване на стойност на атрибут.
`children <id>`	Списък с атрибути на вложените елементи.
`child <id> <n>`	Достъп до n-тия наследник на елемент.
`text <id>`	Достъп до текста на елемент.
`delete <id> <key>`	Изтриване на атрибут на елемент по ключ.
`newchild <id>`	Добавяне на НОВ наследник на елемент. Новият елемент няма никакви атрибути, освен идентификатор.


`xpath <XPath>`	операции за изпълнение на прости XPath 2.0 заявки към кореновия елемент (описаните по-горе)

Бонуси:

- да се реализират XML namespaces
- да се реализират различните XPath оси (ancestor, child, parent, descendant,...)