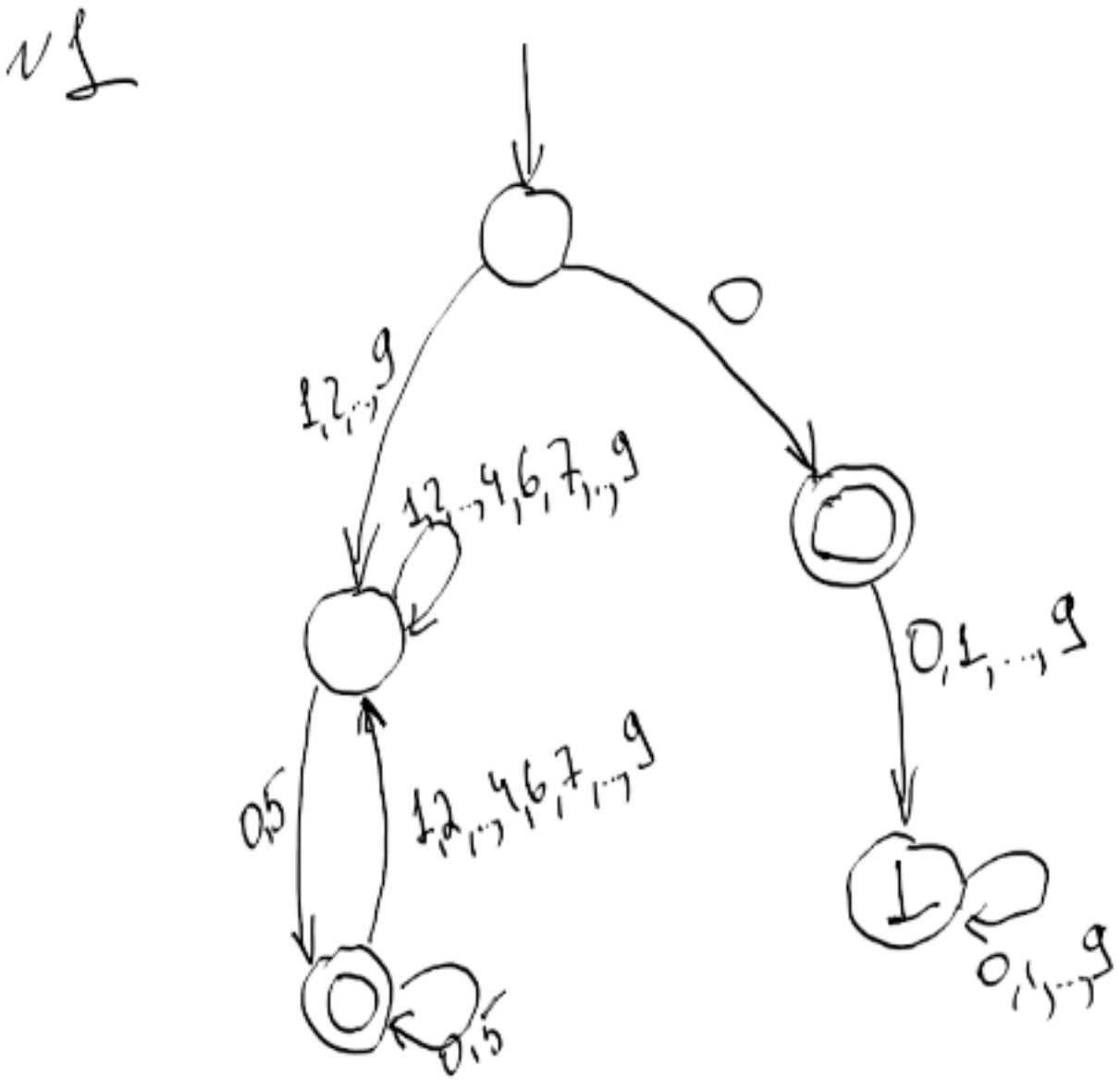
ФЯ, дз на 13.09.2021

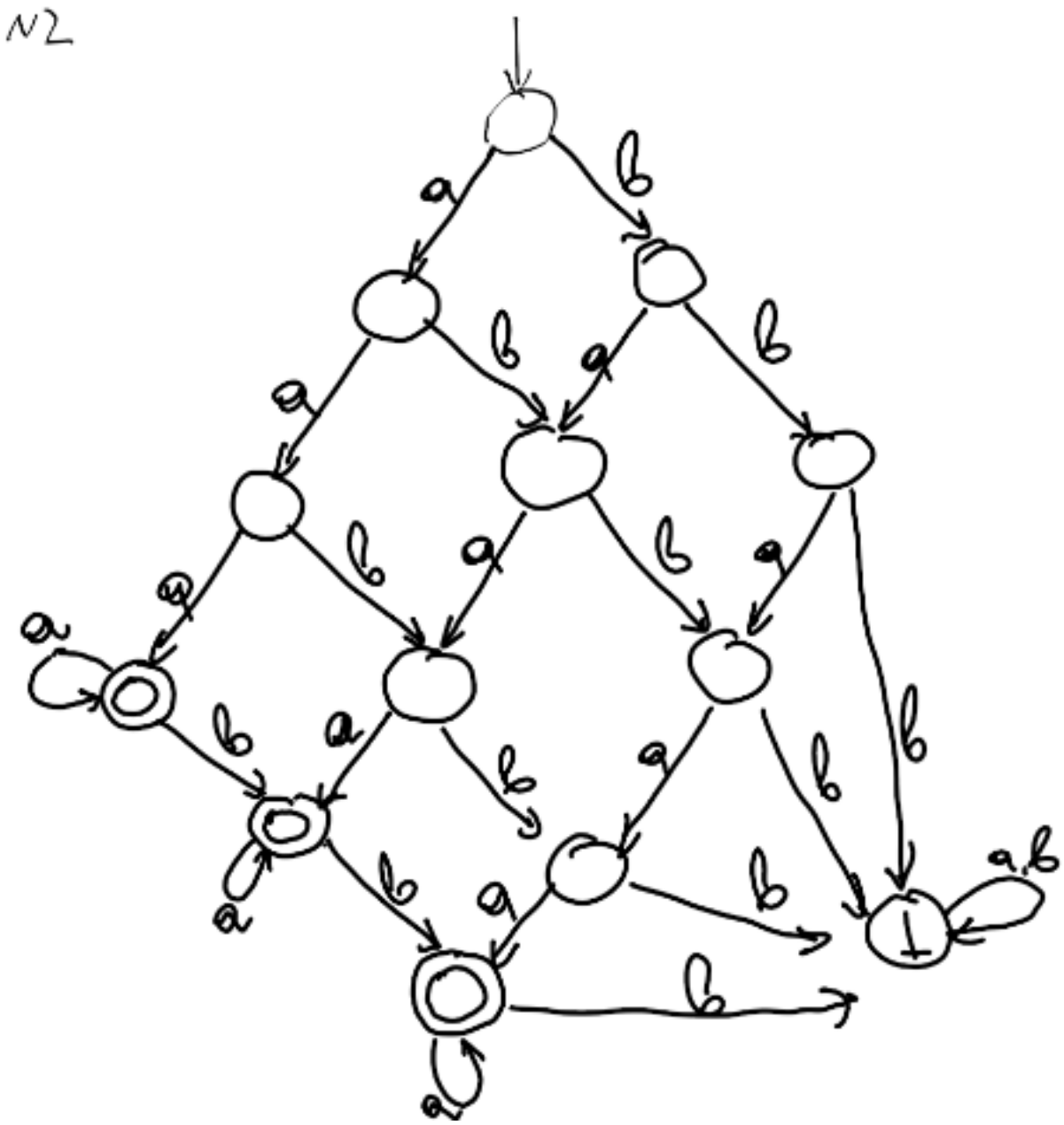
Малофеев Михаил

12 сентября 2021 г.

Задание №1



Задание №2



Задание №3

Я бы хотел показать пару особенностей языка python, о которых я раньше не знал.

- Синтаксис лямбда-функций в языке python

  Мой основной язык программирования — C++, и в нём относительно сложный синтаксис объявления лямбда-функций

```
auto sum = [](int a, int b){
    return a + b;
}
```

  В языке python объявление выглядит красивее и лаконичнее:

```
f = lambda x, y: x + y
```

https://docs.python.org/3.10/reference/expressions.htmllambda

- Конструкция with ... as

  Эта конструкция гарантирует то, что «критические» функции выполнятся в любом случае. Это аналог конструкции try ... except ... finally. Конструкция with более лаконичная и удобная. В частности:

```
manager = (EXPRESSION)
enter = type(manager).__enter__
exit = type(manager).__exit__
value = enter(manager)
hit_except = False

try:
    TARGET = value
    SUITE
except:
    hit_except = True
    if not exit(manager, *sys.exc_info()):
        raise
finally:
    if not hit_except:
        exit(manager, None, None, None)
```

  эквивалентно коду:

```
with EXPRESSION as TARGET:
    SUITE
```

  Такой код выглядит гораздо более лаконично и читабельно. Это крайне удобно использовать для открытия файлов:

```
with open('newfile.txt', 'w', encoding='utf-8') as g:
    d = int(input())
    print('1 / {} = {}'.format(d, 1 / d), file=g)
```

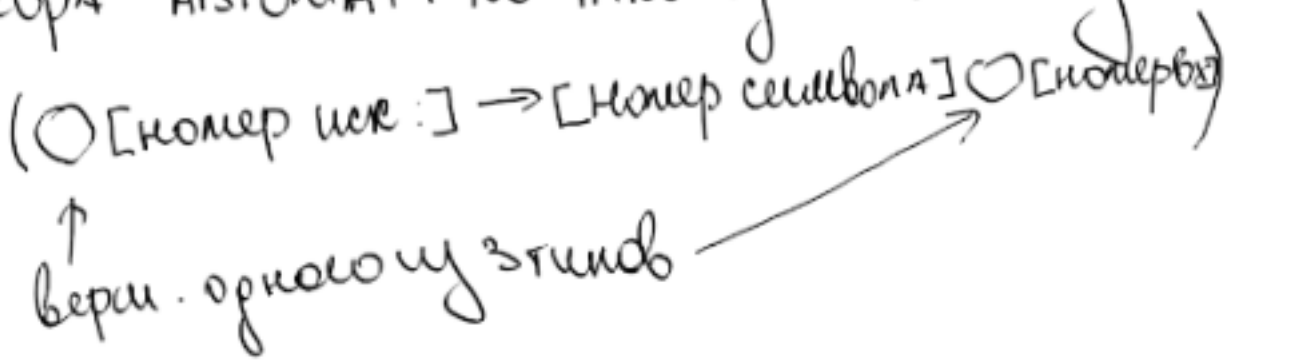
  Этот код гарантирует, что файл будет закрыт вне зависимости от того, что ввёл пользователь.

Задание №4

№4

Заметим, что для того, чтобы описывать автоматы, нам достаточно описывать ориентированные графы, соответствующие этим автоматам. Наш алфавит для описания конечных автоматов будет состоять из символов $\Sigma = \{($, $)$, $[$, $]$, $\circ$,  , $\rightarrow$,  , $0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Словом в нашем языке будет автомат. Мы будем кодировать каждое ребро графа, кодируя вершину, из которой оно исходит, вершину, в которую она входит, а так же символ алфавита, по которому осуществляется переход от одного состо-

ный к другому. Заметим, что по та-
кой кодировке можно однозначно вос-
становить автомат, и по автомату можно
однозначно восстановить его кодировку.
Сперва пронумеруем все вершины
автомата, а затем пронумеруем все
символы алфавита языка, по которому
строится автомат. Далее кодируем
рёбра автомата по такому шаблону:



Номер исх. - номер вершины, откуда исходит ребро.

Номер символа - номер символа, по которому осуществляется переход

Номер вх. - номер вершины, куда входит ребро.

Каждое ребро будем заключать в круглые скобки.

Закодируем автоматы:

1) Автомат для проверки, является ли число натуральным

P.S. это 10 ребер в сокр. записи

Σавт. Σ языка

1 - 1
2 - 2
3 - 3
4 - 4
5 - 5
6 - 6
7 - 7
8 - 8
9 - 9
0 - 0

Кор автомата:

(δ[0] → [1] ◎ [1])(δ[0] → [2] ◎ [1])
(δ[0] → [3] ◎ [1])(δ[0] → [4] ◎ [1])

([0]→[1]◎[1])([0]→[6]◎[1])
([0]→[7]◎[1])([6]→[8]◎[1])
([0]→[9]◎[1])(◎[1]→[0]◎[1])
(◎[2]→[2]◎[1])(◎[3]→[2]◎[1])
(◎[3]→[3]◎[3])(◎[3]→[4]◎[3])
(◎[3]→[5]◎[1])(◎[1]→[6]◎[1])
(◎[1]→[7]◎[1])(◎[1]→[8]◎[1])
(◎[1]→[9]◎[1])

2) Автомат над $\Sigma = \{a, b\}$ для проверки, начинается ли слово с буквы a.

Σавт. Σслова

a – 1

b – 2

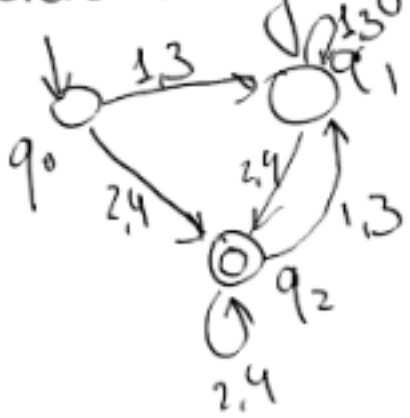
Код автомата:

(0[0] → [2] 0[1]) (0[1] → [1] 0[1])

(0[1] → [2] 0[1]) (0[0] → [1] @[2])

(@[2] → [1] @[2]) (@[2] → [2] @[2])

3) Автомат, проверяющий, что число, составленное из цифр {1,2,3,4} : 2



Σ авт	Σ языка
1	1
2	2
3	3
4	4

Код автомата:

(0[0] → [1] 0[1]) (0[1] → [3] 0[1])

(0[0] → [2] @[2]) (0[0] → [4] @[2])

(0[1] → [1] 0[1]) (0[1] → [3] 0[1])

(0[1] → [2] @[2]) (0[1] → [4] @[2])

(@[2] → [1] 0[1]) (@[2] → [3] 0[1])

(@[2] → [2] @[2]) (@[2] → [4] @[2])

Задание №5

Я сделал частичную подсветку лексического синтаксиса для языка C++. В нём подсвечиваются некоторые ключевые слова, такие как for, while и прочие. Так же подсвечиваются строковые литералы, числа, некоторые базовые типы, такие как int, char и др., имена функций, аргументы шаблонов и название библиотек, а так же однострочные комментарии.