```
'''Вариант - Е, вариант предметной области - 23
("Синтаксическая конструкция - язык программирования")'''


class SynCon:
    # Синтаксическая конструкция
    def __init__(self, id, name, len_con, lan_id):
        self.id = id
        self.name = name
        self.len_con = len_con  # кол-во символов в названии конструкции
        self.lan_id = lan_id


class LanProg:
    # Язык программирования
    def __init__(self, id, name):
        self.id = id
        self.name = name


class SynLan:
    # 'Синтаксические конструкция языка программирования' для реализации
    # связи многие-ко-многим

    def __init__(self, lan_id, syn_id):
        self.lan_id = lan_id
        self.syn_id = syn_id


# языки программирования
lans = [
    LanProg(1, 'C'),
    LanProg(2, 'C#'),
    LanProg(3, 'C++'),
    LanProg(4, 'Python'),
]

# синтаксические конструкции
syns = [
    SynCon(1, 'if', 2, 2),
    SynCon(2, 'else', 4, 2),
    SynCon(3, 'while', 5, 4),
    SynCon(4, 'for', 3, 4),
    SynCon(5, 'switch', 6, 3),
    SynCon(6, 'case', 4, 3),
    SynCon(6, 'elif', 4, 1)
]

syns_lans = [
    SynLan(1, 1),
    SynLan(1, 2),
    SynLan(1, 3),
    SynLan(1, 4),
    SynLan(1, 5),
    SynLan(1, 6),

    SynLan(2, 1),
    SynLan(2, 2),

    SynLan(3, 1),
    SynLan(3, 2),
    SynLan(3, 3),
    SynLan(3, 4),
    SynLan(3, 5),
```

```
    SynLan(3, 6),

    SynLan(4, 1),
    SynLan(4, 2),
    SynLan(4, 3),
    SynLan(4, 4),
]


def main():
    # Основная функция

    # Соединение данных один-ко-многим
    one_to_many = [(s.name, s.len_con, l.name)
                   for l in lans
                   for s in syns
                   if s.lan_id == l.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(l.name, sy.lan_id, sy.syn_id)
                         for l in lans
                         for sy in syns_lans
                         if l.id == sy.lan_id]

    many_to_many = [(s.name, s.len_con, lan_name)
                    for lan_name, lan_id, syn_id in many_to_many_temp
                    for s in syns if s.id == syn_id]
    print('Задание E1')
    # выбираем языки, в названии которых есть 'C'
    res_1 = list(filter(lambda x: 'C' in x[2], one_to_many))
    were = ''
    for elem in res_1:
        if(elem[2] == were):
            print(' ', elem[0], elem[1])
        else:
            were = elem[2]
            print(were, ":", sep='')
            print(' ', elem[0], elem[1])

    print('\nЗадание E2')
    avg_len = dict()
    for link in one_to_many:
        if (link[2] in avg_len):
            avg_len[link[2]].append(link[1])
        else:
            avg_len[link[2]] = [link[1]]
    for key, value in avg_len.items():
        print(key, round(sum(value) / len(value), 2))

    print('\nЗадание E3')
    # выбираем синтаксические единицы, которые начинаются с 'e'
    res_3 = list(filter(lambda x: x[0][0] == 'e', many_to_many))
    for i in range(len(res_3)):
        print(str(i + 1) + '.', res_3[i][0], res_3[i][2])


if __name__ == '__main__':
    main()
```

```
>>> [анализируем RK1 Platonov RT5-31B.py]
Задание E1
C:
  elif 4
C#:
  if 2
  else 4
C++:
  switch 6
  case 4

Задание E2
C 4.0
C# 3.0
C++ 5.0
Python 4.0

Задание E3
1. else C
2. elif C
3. else C#
4. else C++
5. elif C++
6. else Python
```