**Вариант запросов B. Предметная область 7.**

# «Микросхема» и «Компьютер» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название производителя начинается с буквы «A», и микросхемы, которые в них установлены.

# «Микросхема» и «Компьютер» связаны соотношением один-ко-многим. Выведите список компьютеров с наименьшем количеством ядер, отсортированный по минимальному числу ядер.

# «Микросхема» и «Компьютер» связаны соотношением многие-ко-многим. Выведите список всех связанных компьютеров и микросхем, отсортированный по компьютерам, сортировка по микросхемам произвольная.

**Код программы.**

class MicroScheme:

def \_\_init\_\_(self, id: int, company: str, m\_id: int, core: int):

self.\_id = id

self.\_company = company

self.\_m\_id = m\_id

self.\_core = core

@property

def id(self) -> int:

return self.\_id

@property

def m\_id(self) -> int:

return self.\_m\_id

@property

def core(self) -> int:

return self.\_core

class Computer:

def \_\_init\_\_(self, id: int, name: str):

self.\_id = id

self.\_name = name

@property

def id(self) -> int:

return self.\_id

@property

def name(self) -> str:

return self.\_name

class MicroSchemeComputer:

def \_\_init\_\_(self, MicroScheme\_id: int, m\_id: int):

self.\_MicroScheme\_id = MicroScheme\_id

self.\_m\_id = m\_id

@property

def m\_id(self) -> int:

return self.\_m\_id

@property

def MicroScheme\_id(self) -> int:

return self.\_MicroScheme\_id

def task1(Computers: list[Computer], MicroSchemes: list[MicroScheme]):

print("Запрос 1")

data = [(a, b) for a in MicroSchemes for b in Computers if a.m\_id == b.id and a.\_company.startswith("A")]

for (a, b) in data:

print(a.\_company, b.name)

print()

def task2(Computers: list[Computer], MicroSchemes: list[MicroScheme]):

print("Запрос 2")

data = {}

for Computer in Computers:

Computer\_core = [a.core for a in MicroSchemes for b in Computers if a.m\_id == b.id and b.id == Computer.id]

data[Computer.name] = min(Computer\_core)

data\_items = list(data.items())

data\_items.sort(key=lambda x: x[1])

for (Computer, min\_core) in data\_items:

print(Computer, min\_core)

print()

def task3(Computers: list[Computer], MicroSchemes: list[MicroScheme], MicroSchemes\_Computers: list[MicroSchemeComputer]):

print("Запрос 3")

data = [(a, c) for ab in MicroSchemes\_Computers for a in MicroSchemes for b in Computers if ab.MicroScheme\_id == a.id and ab.m\_id == b.id]

data.sort(key=lambda x: x[0].\_company)

for (MicroScheme, Computer) in data:

print(MicroScheme.\_company, Computer.name)

print()

def main():

Computers = [

Computer(1, "Enigma"),

Computer(2, "Altair-8800"),

Computer(3, "Agat"),

Computer(4, "Macintosh"),

Computer(5, "Datapoint-2200")

]

MicroSchemes = [

MicroScheme(1, "BAIKAL", 1, 24),

MicroScheme(2, "BAIKAL", 1, 20),

MicroScheme(3, "AMD", 2, 20),

MicroScheme(4, "BAIKAL", 2, 16),

MicroScheme(5, "BAIKAL", 3, 8),

MicroScheme(6, "BAIKAL", 3, 24),

MicroScheme(7, "AMD", 4, 24),

MicroScheme(8, "BAIKAL", 4, 8),

MicroScheme(9, "AMD", 5, 16)

]

MicroSchemes\_Computers = [

MicroSchemeComputer(1, 1),

MicroSchemeComputer(1, 2),

MicroSchemeComputer(1, 4),

MicroSchemeComputer(2, 1),

MicroSchemeComputer(3, 2),

MicroSchemeComputer(4, 4),

MicroSchemeComputer(5, 5),

MicroSchemeComputer(9, 3)

]

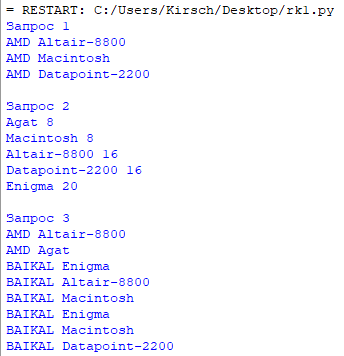
task1(Computers, MicroSchemes)

task2(Computers, MicroSchemes)

task3(Computers, MicroSchemes, MicroSchemes\_Computers)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат выполнения.  
**