GeekBrains

Структурная и процедурная парадигмы

Урок 2

Курс “Парадигмы программирования и языки парадигм”

Цели семинара

- 📌 Понять основные отличия между **структурной**, **процедурной** и **императивной** парадигмами
- 📌 Начать решать задачи в рамках одной выбранной парадигмы

План семинара

- 📌 Викторина
- 📌 Пишем код
- 📌 Решаем кейсы
- 📌 Подведение итогов

Викторина

Регламент

1. Прочитать код
2. Подумать в какой парадигме написана программа и поделиться своим ответом
3. Обсудить решение

Что за парадигма: четность числа

```python
if __name__ == '__main__':
    num = int(input('Enter a number: \n'))
    if num % 2 == 0:
        print(True)
    else:
        print(False)
```

Ответ: .. ?

Что за парадигма: четность числа

```python
if __name__ == '__main__':
    num = int(input('Enter a number: \n'))
    if num % 2 == 0:
        print(True)
    else:
        print(False)
```

Ответ: Это *структурная* парадигма!

Почему это так: Процедуры в данной программе не объявляются, но при этом все держится на структурных управляющих конструкциях: последовательностях и условиях (без циклов).

Что за парадигма: print_numbers

```python
def print_numbers(numbers):
    for num in numbers:
        print(num)
```

Ответ: .. ?

Что за парадигма: print_numbers

```python
def print_numbers(numbers):
    for num in numbers:
        print(num)
```

Ответ: Этот кусочек относится и к *структурной* и к *процедурной* парадигме!

Почему это так:

- Структурная, поскольку используется **for** и нет **goto**.
- Процедурная, поскольку код оформлен в виде **процедуры**.

Что за парадигма: quicksort

```python
def quicksort(arr):
    if len(arr) ⩽ 1:
        return arr
    else:
        base = arr[0]
        less = [x for x in arr[1:] if x ⩽ base]
        greater = [x for x in arr[1:] if x > base]
        return quicksort(less) + [base] + quicksort(greater)

numbers = [8, 3, 1, 5, 9, 2, 7, 4, 6]
sorted_numbers = quicksort(numbers)
print(sorted_numbers)
```

Ответ: .. ?

Что за парадигма: quicksort

```python
def quicksort(arr):
    if len(arr) ⩽ 1:
        return arr
    else:
        base = arr[0]
        less = [x for x in arr[1:] if x ⩽ base]
        greater = [x for x in arr[1:] if x > base]
        return quicksort(less) + [base] + quicksort(greater)

numbers = [8, 3, 1, 5, 9, 2, 7, 4, 6]
sorted_numbers = quicksort(numbers)
print(sorted_numbers)
```

Ответ: Здесь присутствует и *процедурная* и *структурная* парадигма!

Почему это так:

- Процедурная: есть процедура quicksort(arr)
- Структурная: *goto* тут нет, вся программа находится в рамках *трех управляющих конструкций*

Вопросы

Пишем код

Регламент

1. Вместе читаем условия задачи
2. Вы самостоятельно решаете задачу
3. Вместе обсуждаем решение

След матрицы

След матрицы: структурный

- **Контекст**
  След матрицы - это сумма чисел на её главной диагонали. *След* определён только для квадратных матриц (количество столбцов = количеству строк).

- **Задача**
  Реализовать чисто **структурную** реализацию вычисления следа для любой матрицы NxN.

- **Решение**.. ?

След матрицы: структурный

- **Контекст**
  След матрицы - это сумма чисел на её главной диагонали. *След* определён только для квадратных матриц (количество столбцов = количеству строк).

- **Задача**
  Реализовать чисто **структурную** реализацию вычисления следа для любой матрицы NxN.

- **Решение**

```python
matrix = [[1,2,3],
          [4,5,6],
          [7,8,9]]

trace = 0
for i, row in enumerate(matrix):
    for j, el in enumerate(row):
        if i == j:
            trace += el

print(trace)
```

След матрицы: процедурный

- **Контекст**
  След матрицы - это сумма чисел на её главной диагонали. *След* определён только для квадратных матриц (количество столбцов = количеству строк).

- **Задача**
  Добавить **процедурную** парадигму в имеющуюся реализацию вычисления следа.

- **Решение**.. ?

След матрицы: процедурный

- **Контекст**
  След матрицы - это сумма чисел на её главной диагонали. *След* определён только для квадратных матриц (количество столбцов = количеству строк).

- **Задача**
  Добавить **процедурную** парадигму в имеющуюся реализацию вычисления следа.

- **Решение**

```python
def get_trace(matrix):
    trace = 0
    for i, row in enumerate(matrix):
        for j, el in enumerate(row):
            if i == j:
                trace += el
    return trace

matrix = [[1,2,3],
          [4,5,6],
          [7,8,9]]
print(get_trace(matrix))
```

Десятичное в двоичное

Десятичное в двоичное

- **Условие**
  На вход подается число в десятичной системе счисления

- **Задача**
  Написать скрипт в любой парадигме, который возвращает полученное число переведенное в двоичную систему счисления. Обоснуйте сделанный выбор парадигм.

- **Решение** .. ?

Десятичное в двоичное

- **Условие**
  На вход подается число в десятичной системе счисления

- **Задача**
  Написать скрипт в любой парадигме, который возвращает полученное число переведенное в двоичную систему счисления. Обоснуйте сделанный выбор парадигм.

- **Решение**

```python
def decimal_to_binary(decimal):
    binary = ""
    while decimal > 0:
        binary = str(decimal % 2) + binary
        decimal = decimal // 2
    return binary
```

Решаем кейсы

Регламент

1. Вместе читаем кейс
2. Вы думаете как можно его решить
3. Вместе обсуждаем решение

Рекомендация музыки

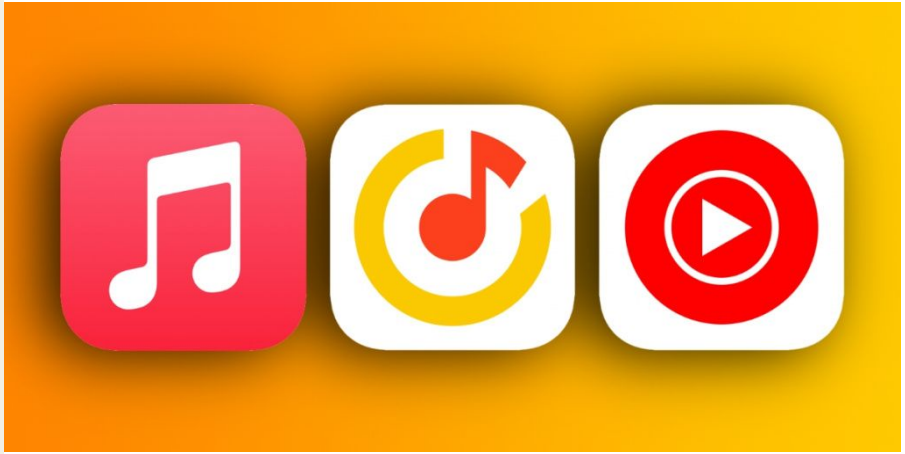
Кейс. Рекомендация музыки

Контекст:
Есть большой и известный музыкальный стриминговый сервис. Вас наняли на проект обновления рекомендательной системы.

Рекомендательная система - это программа, которая сортирует объекты (в нашем случае - музыкальные треки) в порядке убывания “интереса” пользователя. Простыми словами, мы пытаемся предсказать что пользователю будет наиболее интересно и сделать так, чтобы оно было вначале, а все неинтересное - в конце.

ТЗ: Ваша программа - это machine learning сервис, который будет смотреть на профиль пользователя и его историю прослушанных треков, а выдавать - список новых треков в качестве рекомендации.

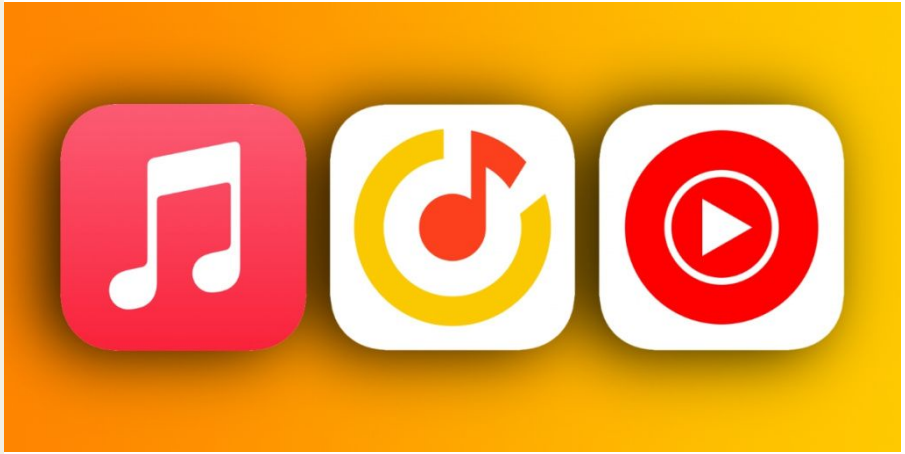
Задача: Какие парадигмы вы будете использовать для разработки сервиса и почему именно её?



Кейс. Рекомендация музыки

Обсуждение:
В этом проекте можно использовать не только структурную и процедурную парадигму, но и другие, про которые пока что речи не было. Если вы можете их назвать и обосновать свой ответ - то это здорово. А пока что, рассмотрим один из возможных вариантов ответа в рамках трёх известных нам парадигм:

- Структурная - **скорее всего да**. Вы конечно можете захотеть ускорить алгоритмы с помощью *goto*, но в рамках данного проекта с большой вероятностью обнаружите, что эта задача алгоритмической оптимизации уже решена за вас в различных библиотеках на разных языках. Поэтому можно структурный код.
- Процедурная - **точно да**, поскольку проект такого объёма крайне неудобно будет реализовывать без единой процедуры.
- Декларативная - **возможно, но не обязательно**. В данном проекте декларативной парадигме может быть отдана большая часть обработки данных, но может и не быть. Это зависит от объёма данных и технологического стека вашей компании.



Итоги семинара

Итоги семинара

- 📌 Викторина “Что за парадигма”
  - Решили 3 задачи на классификацию парадигм
- 📌 Пишем код
  - Решили задачу “След матрицы” в двух парадигмах
  - Решили задачу “Десятичное в двоичное”
- 📌 Решаем кейсы
  - Решили и обсудили кейс “Рекомендация музыки”
- 📌 Подвели итоги

Домашнее задание

Таблица умножения

- **Условие**
  На вход подается число *n*.

- **Задача**
  Написать скрипт в любой парадигме, который выводит на экран таблицу умножения всех чисел от 1 до *n*.
  Обоснуйте выбор парадигм.

- **Пример вывода:**

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
..
1 * 9 = 9
```

Конец семинара
Спасибо за внимание!