**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Генерация** **датасета»**

**Вариант – 1**

**Студент гр. 23Б15-пу**

**Сурин И.С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc177901232)
2. [Описание задачи (формализация задачи) 3](#_Toc177901233)
3. [Теоретическая часть 4](#_Toc177901234)
4. [Основные шаги программы 6](#_Toc177901235)
5. [Блок схема программы 7](#_Toc177901236)
6. [Описание программы 9](#_Toc177901237)
7. [Рекомендации пользователя 10](#_Toc177901238)
8. [Рекомендации программиста 10](#_Toc177901239)
9. [Исходный код программы 11](#_Toc177901240)
10. [Контрольный пример 11](#_Toc177901241)
11. [Вывод 13](#_Toc177901242)
12. [Источники 13](#_Toc177901243)

**Цель работы:**

Целью лабораторной работы является разработка генерации датасета для покупок в магазине с учетом различных требований и условий. Генератор данных должен уметь создавать большие наборы данных, которые включают информацию о магазинах, товарах, покупателях, транзакциях и их деталях. В частности, необходимо создать и настроить параметры генерации таких данных, как название магазина, координаты покупки (включая дату, время и географические координаты), категорию и бренд товара, номер платежной карты, количество товаров и стоимость покупки. Итоговый датасет должен соответствовать требованиям по объему данных и содержать реалистичные транзакции.

**Требования:**

1. **Название магазина**:
   * Пример названия — "М.Видео".
   * В дополнительных условиях магазины генерируются с названиями, состоящими минимум из 30 символов, в том числе с названием "Словарь".
2. **Координаты (широта и долгота)**:
   * Для каждого магазина нужно указать реальные географические координаты (широта и долгота).
   * Время покупки должно находиться в пределах рабочего времени магазина (с 10:00 до 22:00).
   * Координаты округляются до 10^-8 степени.
3. **Категория товара**:
   * Датасет должен поддерживать возможность настройки категорий товара в зависимости от типа магазина.
   * Пример категории — "ноутбук".
   * Дополнительно можно генерировать другие категории, с минимум 50 категориями на магазин.
4. **Бренд**:
   * Для каждой категории товаров можно настраивать бренды.
   * Пример бренда — "lenovo".
   * В дополнительных условиях должно быть минимум 500 брендов на магазин.
5. **Номер карточки**:
   * Генерация номеров карт с возможностью их многократного использования, но не более 5 раз на одну карту.
   * Возможность настройки вероятности, к какому банку (например, Сбербанк, Т-банк) и платежной системе (Visa, MasterCard) принадлежит карта.
6. **Количество товаров**:
   * Количество товаров в одной транзакции — минимум 5.
7. **Стоимость**:
   * В дополнительных условиях стоимость товаров зависит от категории и бренда, а также должна соответствовать реальным средним ценам товаров.
   * Важно учесть, что не должно быть пустых значений или покупок с нулевой стоимостью. В таких случаях должна выводиться ошибка.
8. **Объем данных**:
   * Датасет должен содержать минимум 50 000 строк.
   * Все строки в датасете должны быть уникальными.

**Ограничения:**

* Магазины и категории должны соответствовать тематике, то есть товары, бренды и категории должны быть логически связаны с магазинами.
* Время транзакций должно попадать в рабочие часы магазинов (10:00–22:00).

**Теоретическая часть**

Для создания датасета использованы несколько программных модулей, каждый из которых отвечает за генерацию определенных аспектов данных, необходимых для симуляции покупок в магазинах.

1. **Brands.py**: В этом модуле содержится три словаря, которые разделяют товары по уровню стоимости:
   * **Высокая стоимость**: Этот словарь включает товары премиум-класса. Ключами являются категории товаров, а значениями — списки брендов, которые производят эти товары.
   * **Средняя стоимость**: В этом словаре находятся товары средней ценовой категории, организованные аналогично — по категориям и брендам.
   * **Низкая стоимость**: Здесь представлены товары бюджетной категории, которые также классифицированы по типам товаров и брендам, которые их производят.

Это позволяет гибко выбирать товары и бренды для каждой покупки в зависимости от категории магазина и типа товара.

1. **Store.py**: Этот модуль содержит один основной словарь **shops**, где “подсловарем” является спецификация магазина. Ключем тут является название магазина, а значение — список координат (широта и долгота) магазинов. Это используется для того, чтобы при генерации данных каждому магазину присваивались реальные географические координаты, что делает датасет более правдоподобным.
2. **Tovar.py**: Здесь хранится словарь, где ключ — это спецификация магазина, а значение — список товаров, продаваемых в этом магазине. Каждый товар также связан с брендами, которые его производят. Этот модуль используется для того, чтобы корректно связывать магазины с категориями товаров и брендами при генерации датасета.
3. **Test\_uno.py**: Этот скрипт является основным и объединяет данные из предыдущих модулей для создания датасета. Вот ключевые функции и логика, использованные в нем:
   * **Генерация случайных дат и времени**: Покупки генерируются в пределах рабочего времени магазина (с 10:00 до 21:00), с случайными датами в пределах нескольких лет.
   * **Генерация номеров платежных карт**: Карты генерируются с учетом заданных весов, которые определяют вероятность принадлежности к определенному банку и платежной системе (например, Visa, MasterCard или МИР).
   * **Генерация стоимости товаров**: Стоимость товаров зависит от категории товара (высокая, средняя или низкая стоимость), и она корректируется на основе количества покупаемых товаров.
   * **Количество товаров и стоимости**: Для каждой покупки случайно генерируется количество товаров (от 5 до 50), а итоговая стоимость рассчитывается с учетом количества и типа товара.
   * **Запись данных в файл**: Генерированные данные сохраняются в формате XML, что удобно для последующего анализа и обработки.

**Ограничения:**

* **Количество строк**: Датасет должен содержать минимум 50 000 строк.
* **Реалистичность данных**: Все данные, включая номера карт, бренды и товары, должны быть реалистичными и соответствовать логике генерации.
* **Банки и платежные системы**: Для каждой карты устанавливается вероятность ее принадлежности к одному из трех банков и одной из трех платежных систем.
* **Товары и бренды**: Все товары должны соответствовать категории магазина, и для каждого товара есть ограниченное количество брендов.

**Основные шаги программы**

1. **Запуск программы**: Пользователь вводит вероятности для банков и платежных систем.
2. **Генерация даты и времени**: Случайно выбирается дата и время покупки в интервале с 2020 по 2024 год, с 10:00 до 21:00.
3. **Выбор магазина и координат**: Из файла store.py случайно выбираются магазин и его координаты.
4. **Выбор категории и бренда**: Из файла tovar.py выбирается категория товара, затем из файла brands.py — его бренд, в зависимости от уровня стоимости (высокая, средняя, низкая).
5. **Генерация номера карты**: Номер карты генерируется на основе введенных весов банков и платежных систем.
6. **Определение количества и стоимости товаров**: Случайным образом выбирается количество товаров и рассчитывается итоговая стоимость.
7. **Формирование строки данных**: Для каждой покупки собираются данные: магазин, координаты, дата, категория, бренд, номер карты, количество и стоимость.
8. **Запись данных**: Все данные сохраняются в файл dataset.xml в формате XML.
9. **Отчет о времени**: Выводится время выполнения генерации.



Рис 1. Блок-схема основной программы

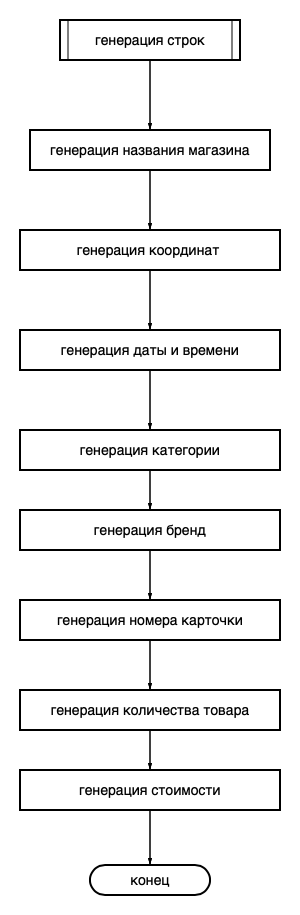
****

Рис 2. Блок-схема подпрограммы

**Описание программы**

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек: random, datetime, pandas. Программа организована через модульную структуру с акцентом на генерацию данных для покупок в магазинах. В процессе разработки программы использовались 4 основных модуля, каждый из которых отвечает за определённые аспекты генерации данных:

|  |  |  |
| --- | --- | --- |
| Структура данных | Описание | Возвращаемое значение |
| high\_price\_category | Содержит товары высокого ценового сегмента, разделенные по категориям и брендам | dict |
| medium\_price\_category | Содержит товары среднего ценового сегмента, разделенные по категориям и брендам | dict |
| low\_price\_category | Содержит товары низкого ценового сегмента, разделенные по категориям и брендам | dict |

Таблица 1. **brands.py**

|  |  |  |
| --- | --- | --- |
| Структура данных | Описание | Возвращаемое значение |
| shops | Содержит спецификацию магазинов с их координатами | dict |

Таблица 2. **shops.py**

|  |  |  |
| --- | --- | --- |
| Структура данных | Описание | Возвращаемое значение |
| tovars | Содержит товары, их категории и бренды по спецификациям магазинов | dict |

Таблица 3. **tovar.py**

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| |  | | --- | | weighted\_choice |  |  | | --- | |  | | Выбор элемента из списка на основе весов | string |
| |  | | --- | | generate\_random\_date |  |  | | --- | |  | | Генерация случайной даты и времени покупки | datetime |
| generate\_random\_cost | Генерация случайной стоимости товара на основе его категории | int |
| generate\_random\_card\_number | Генерация случайного номера карты на основе банка и платежной системы | string |
| generate\_row | Генерация строки с данными о покупке: магазин, товар, категория, стоимость | dict |

Таблица 4. **test\_uno.py**

**Рекомендации пользователя**

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как random, pandas, и datetime. Все модули программы должны находиться в одной директории для корректной работы. Запуск программы осуществляется через файл test\_uno.py, который генерирует данные о покупках в магазинах. Перед запуском программы настройте веса для банков и платежных систем, вводя их в консоль. Обратите внимание, что введенные веса должны быть больше нуля. Программа автоматически генерирует покупки, используя данные из модулей brands.py, store.py, и tovar.py. Если у вас есть собственные данные о магазинах, товарах или брендах, убедитесь, что они структурированы аналогично файлам brands.py, store.py, и tovar.py.

**Рекомендации программиста**

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек и Python. Следите за правильной структурой данных в модулях brands.py, store.py, и tovar.py, чтобы избежать ошибок при генерации данных. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода. Регулярно тестируйте программу на различных наборах данных и проверяйте корректность сгенерированных данных, таких как номера карт, стоимость товаров и выбор магазинов.

# **Исходный код программы:**

[**https://github.com/Ignatio27/spbu-fundamentals-of-algorithms**](https://github.com/Ignatio27/spbu-fundamentals-of-algorithms)

**Контрольный пример**

1. **Запуск программы**  
   Для запуска программы используйте файл test\_uno.py. Этот скрипт отвечает за генерацию данных о покупках на основе информации о магазинах, товарах и брендах, используя данные из файлов brands.py, store.py, и tovar.py.
2. **Ввод весов для банков и платежных систем**  
   После запуска программы пользователю будет предложено ввести веса для банков (Рис. 3) и платежных систем (Рис. 4). Веса определяют вероятность того, какая платежная система или банк будут выбраны для каждой покупки.

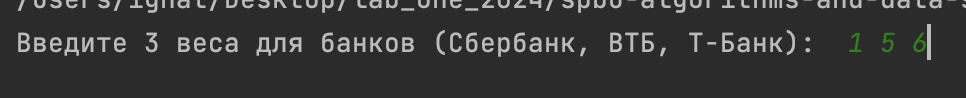


Рис 3. пример ввода весов банков

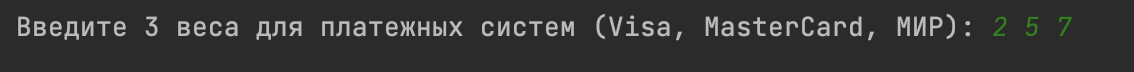


Рис 4. пример ввода весов платежных систем

1. **Ввод количества строк**

После успешной загрузки пользователю предлагается ввести количество строк для генерации (Рис. 5). Минимальное количество строк, которое можно сгенерировать, составляет **50 000** (Рис. 6).

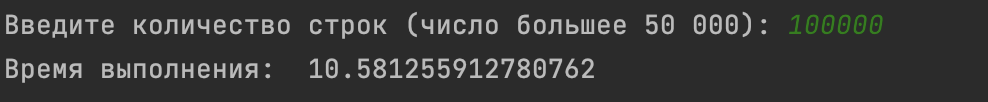


Рис 5. пример ввода количества строк

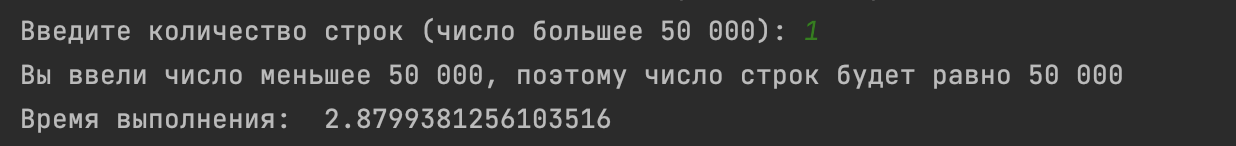


Рис 6. пример ввода количества строк меньшего 50000

4. Генерация строк

После ввода количества строк программа приступает к их генерации, используя данные о магазинах и введенные параметры, затем сгенерированный датасет сохраняется в dataset.xml.

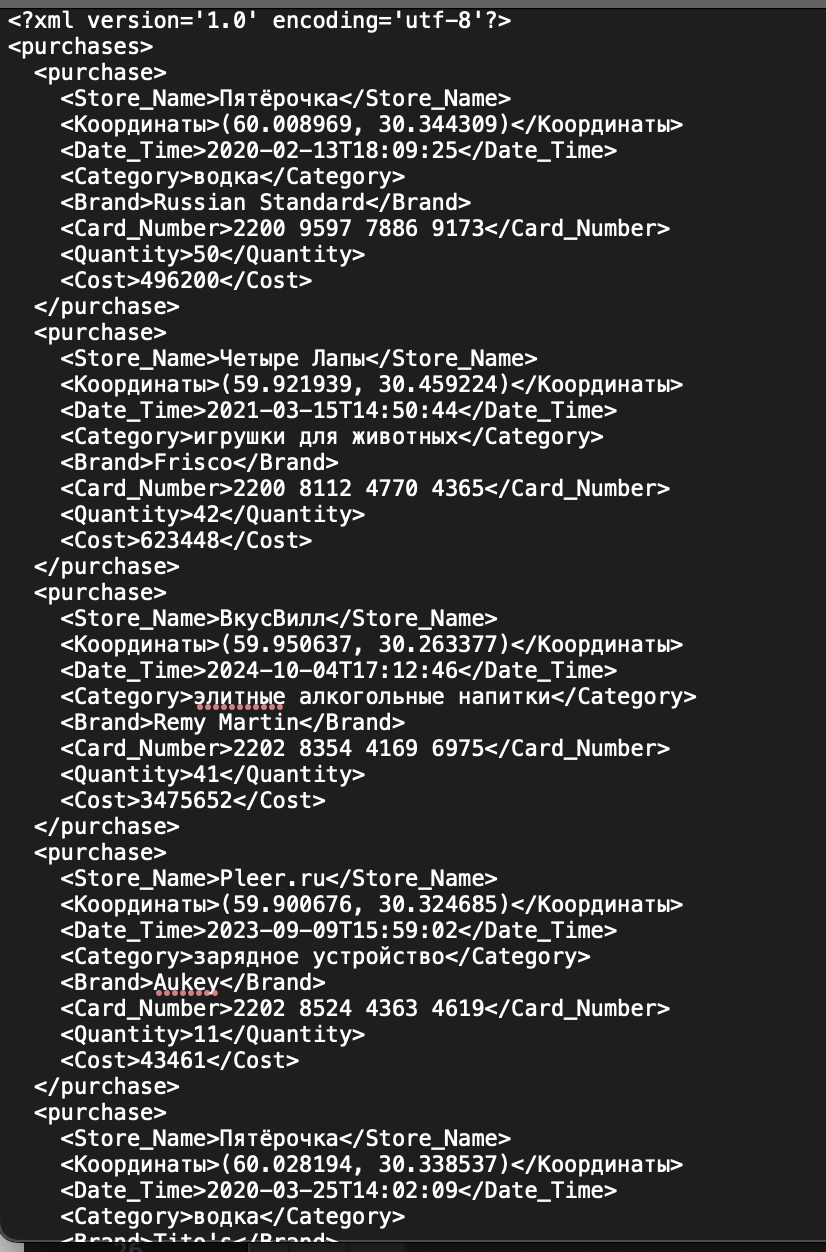


Рис 7. пример датасета товаров

**Вывод**  
В рамках данной работы были исследованы принципы генерации синтетических данных, применительно к моделированию покупок в различных магазинах. Разработан алгоритм, который учитывает информацию о магазинах, категориях товаров, брендах, а также спецификации товаров и их стоимости. Было реализовано программное обеспечение для автоматической генерации датасета, включающего такие данные, как координаты магазинов, категории товаров, бренды, количество и стоимость покупок, а также данные о платежных картах. Программа позволяет настраивать параметры генерации данных для банков и платежных систем, что обеспечивает реалистичность и соответствие получаемого датасета заданным требованиям.

**Источники**

1. random — Generate pseudo-random numbers // Python URL: <https://docs.python.org/3/library/random.html>
2. pandas — Python Data Analysis Library // pandas URL: <https://pandas.pydata.org/docs/> .
3. XML Processing Modules // Python URL: <https://docs.python.org/3/library/xml.html>
4. Python datetime module // Python URL: <https://docs.python.org/3/library/datetime.html>