**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 – 3**

**Студент гр. 23Б15-пу**

**Абрахин Е.Д.**

**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 5](#_Toc177901235)
5. [Блок схема программы 6](#_Toc177901236)
6. [Описание программы 8](#_Toc177901237)
7. [Рекомендации пользователя 10](#_Toc177901238)
8. [Рекомендации программиста 10](#_Toc177901239)
9. [Исходный код программы 10](#_Toc177901240)
10. [Контрольный пример 11](#_Toc177901241)
11. [Вывод 13](#_Toc177901242)
12. [Источники 13](#_Toc177901243)

# Цель работы

Целью лабораторной работы является разработка системы генерации датасета для покупки железнодорожных билетов с учетом определенных требований и условий. Датасет должен включать личные данные пассажиров, информацию о поездах, вагонах и местах, а также данные о платежных картах для оплаты билетов.

# Описание задачи (формализация задачи)

Задача состоит в создании датасета для железнодорожных билетов со следующими требованиями:

1. **ФИО**: Славянские имена и фамилии.
2. **Паспортные данные**: Русские паспортные данные с уникальными значениями.
3. **Откуда и куда**: Города в РФ, с различием отправной и конечной точек.
4. **Дата отъезда и приезда**: Случайные даты в пределах года.
5. **Рейс**: Номера поездов в заданных диапазонах.
6. **Выбор вагона и места**: Учет классов вагонов и типов мест.
7. **Стоимость**: В зависимости от длины маршрута и типа вагона.
8. **Карта оплаты**: Генерация карт с возможностью многократного использования с повторением не больше пяти раз и возможностью настраивать вероятность к какому банку и платежной системе принадлежит карта.
9. **Количество строк датасета**: Не меньше 50000.

# Теоретическая часть

Для создания датасета использованы несколько программных модулей:

1. TimetableGeneration.py: Генерация расписания поездов с учетом различных типов поездов, временных интервалов и расстояний между городами. Учитывает специфику поездов (скоростные, высокоскоростные и т.д.) и создает файл timetable.csv.
2. PersonGeneration.py: Генерация личных данных пассажиров с использованием библиотеки Faker [[1]](#_Источники). Включает создание паспортных данных, ФИО и платежных карт с учетом введенных весов, которые влияют на частоту соответствующего банка и платежной системы.
3. SeatGeneration.py: Генерация информации о вагонах и местах, включая расчет стоимости билета в зависимости от типа вагона.
4. RailwayTicketsGeneration.py: Основной скрипт для генерации билетов, который объединяет данные из вышеуказанных модулей и записывает их в XML файл.

Ограничения:

* Количество строк в датасете ограничивается вводом пользователя, но минимальное количество сгенерированных строк будет 50000.
* ФИО пассажиров только славянские.
* Уникальность паспортных данных и их принадлежность к русским.
* Разные точки отправки и прибытия только по территории РФ.
* Рейсы уникальные из определенных номеров, которые бывают до полного возвращения на вокзал.
* Вагоны и места генерируются по заданным типам.
* Веса банков и платежных систем определяются пользователем и должны в сумме давать число больше нуля.
* Логика выбора карт оплаты с ограничением на 5 повторов.

# Основные шаги программы

1. Запуск программы (RailwayTicketsGeneration.py):
2. Пользователь вводит веса банков и платежных систем.
3. Загружается расписание поездов из файла timetable.csv через функцию loadTimetable. Если файл отсутствует, сначала генерируется расписание с помощью скрипта TimetableGeneration.py.
   1. Генерация расписания поездов, включая расчет расстояний, скорости, времени поездки и стоимости билетов на основе данных о городах и параметров поездов.
4. Пользователь вводит количество билетов для генерации.
   1. Билеты генерируются с использованием данных о расписании, персональных данных и месте в вагоне, а также информации о платежных картах.
      1. Генерация случайного места в поезде с учетом типа вагона и его стоимости.
      2. Генерация персональных данных: Имя, фамилия, отчество, серия и номер паспорта.
      3. Генерация платежной карты с использованием весов для выбора платежной системы и банка.
      4. Происходит расчет итоговой стоимости билета.
5. Данные собираются и записываются в файл tickets.xml.

# Блок схема программы

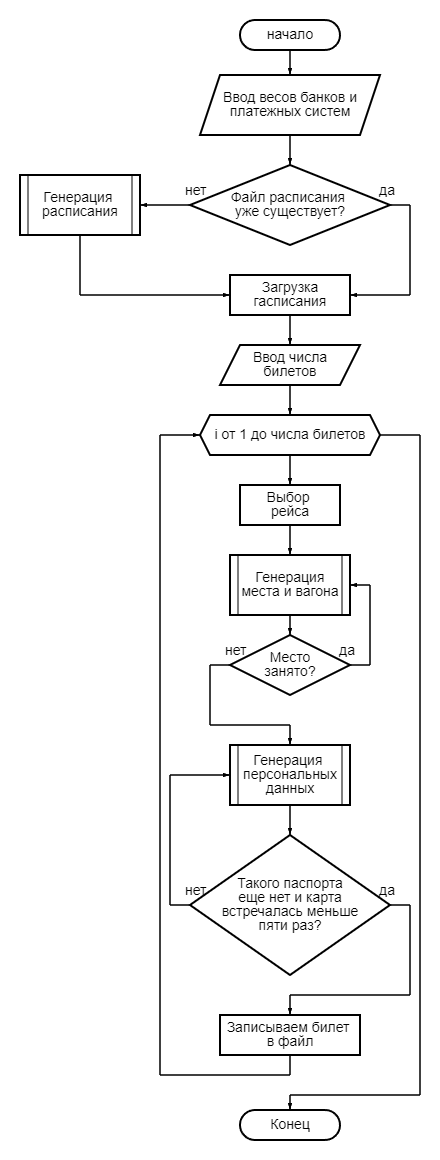


Рис 1. Блок-схема основной программы

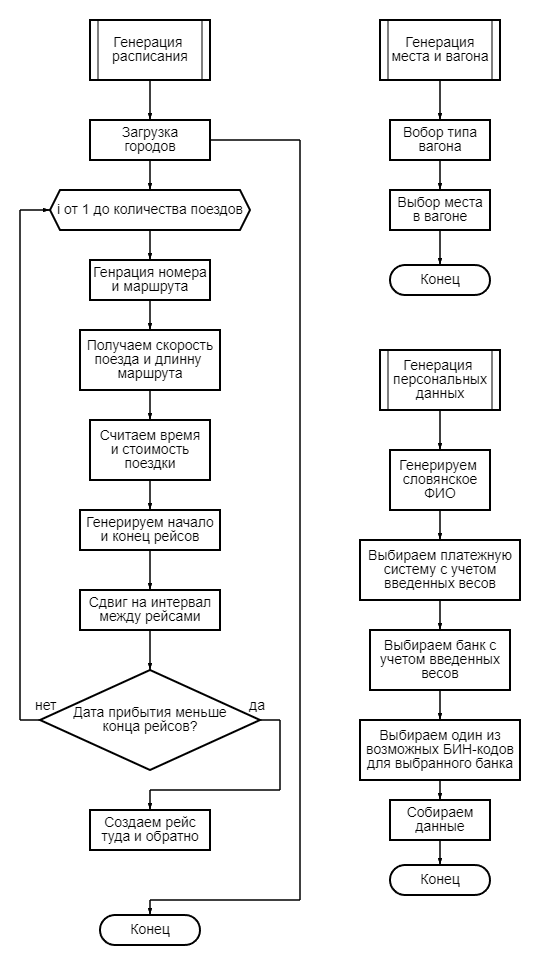


Рис 2. Блок-схема подпрограмм

# Описание программы

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек: faker [[1]](#_Источники), random [[2]](#_Источники), csv[[3]](#_Источники) и xml [[4]](#_Источники). Программа организована через модульную структуру с акцентом на генерацию данных для железнодорожных билетов. В процессе разработки программы использовались 11 функций, каждая из которых имеет чётко определённое назначение и 4 основных модуля:

Таблица 1. RailwayTicketsGeneration.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| generateTimetable | Запуск генерации расписания поездов. | None |
| loadTimetable | Загрузка расписания из файла или генерация нового, если файл отсутствует. | list |
| generateTicketsXml | Генерация билетов и запись в XML [[4]](#_Источники) файл. | None |

Таблица 2. PersonGeneration.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| generatePersonalData | Генерация основных личных данных пассажира (имя, фамилия, паспорт). | tuple |
| generatePerson | Генерация полного набора данных для пассажира, включая платежные карты. | list |

Таблица 3. SeatGeneration.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| generateRandomSeat | Генерация информации о вагоне и месте пассажира (тип вагона, место и цена). | tuple |

Таблица 4. TimetableGeneration.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| calculateDistance | Расчет расстояния между городами на основе формулы гаверсинусов [[5]](#_Источники). | float |
| determineDirection | Определение направления движения поезда. | str |
| determineTrainSpeed | Определение скорости поезда в зависимости от его номера. | int |
| generateOddNumbers | Генерация нечетных чисел. | list |

# Рекомендации пользователя

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как faker [[1]](#_Источники) и xml [[4]](#_Источники). Код можно запустить в среде разработки или через командную строку, используя консоль для настройки параметров и генерации данных. Также убедитесь, что все модули программы находятся в одной директории для корректного выполнения. Запуск программы производится через файл RailwayTicketsGeneration.py, который автоматически генерирует расписание в файл timetable.csv, если его нет. Важно периодически проверять корректность данных перед генерацией билетов. Отдельно сгенерировать файл timetable.csv можно, запустив TimetableGeneration.py. Если вы хотите использовать собственный файл расписания, убедитесь, что он называется timetable.csv и его структура заголовков совпадает с требуемой: TrainNumber, DepartureCity, DestinationCity, DepartureTime, ArrivalTime, Cost. Также настройте веса для платежных систем и банков согласно вашим требованиям, убедившись, что веса больше нуля.

# Рекомендации программиста

Поддерживайте актуальные версии используемых библиотек и Python для обеспечения работоспособности программы на современных системах. Следуйте передовым практикам разработки (best practices), уделяйте внимание четкому именованию переменных и функций. Регулярно проводите тестирование программы на различных входных данных, чтобы убедиться в её надежности и корректности.

# Исходный код программы

[**https://github.com/FasterXaos/Algorithms\_and\_Data\_Structures**](https://github.com/FasterXaos/Algorithms_and_Data_Structures)

# Контрольный пример

1. Запуск программы

Для запуска программы используйте файл **RailwayTicketsGeneration.py.** Программа будет отвечать за генерацию билетов на основе заданных данных о расписании поездов, платежных системах и банках.

2. Ввод весов платежных систем и банков

После запуска программы пользователю предложено ввести **веса для платежных систем** (Рис. 3) и **веса для банков** (Рис. 4). Веса определяют вероятность выбора той или иной платежной системы или банка при генерации билета.

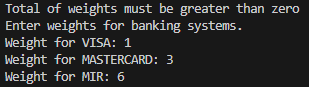


Рис 3. пример ввода весов платежных систем

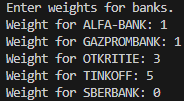


Рис 4. пример ввода весов банков

3. Ввод количества билетов

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



Рис 5. пример ввода количества билетов



Рис 6. пример ввода количества билетов меньшего 50000

4. Генерация билетов

После ввода количества билетов программа приступает к их генерации, используя данные о расписании поездов и введенные параметры, затем сгенерированный датасет сохраняется в tickets.xml.

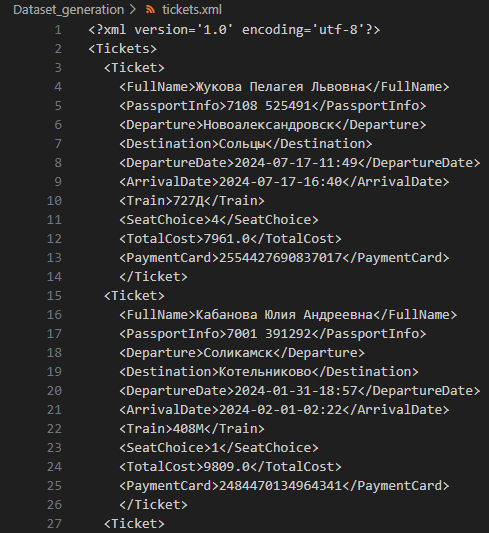


Рис 7. пример датасета билетов

# Вывод

В рамках данной работы были исследованы принципы генерации синтетических данных, применительно к моделированию покупок железнодорожных билетов. Разработан алгоритм, который учитывает особенности пассажирских данных, маршрутов, типов поездов и вагонов, а также ценообразования билетов. Было реализовано программное обеспечение для автоматической генерации датасета, включающего такие данные, как личные данные пассажиров, информация о поездах, билетах и платежных системах. Программа позволяет настраивать параметры генерации банковских карт оплаты, обеспечивая соответствие заданным требованиям и реалистичность получаемого датасета.

# Источники

1. Faker’s documentation // Faker URL: <https://faker.readthedocs.io/en/master/> (дата обращения: 18.09.2024).
2. random — Generate pseudo-random numbers // Python URL: <https://docs.python.org/3/library/random.html> (дата обращения: 18.09.2024).
3. csv — CSV File Reading and Writing // Python URL: <https://docs.python.org/3/library/csv.html> (дата обращения: 18.09.2024).
4. XML Processing Modules // Python URL: <https://docs.python.org/3/library/xml.html> (дата обращения: 18.09.2024).
5. Haversine formula // Wikipedia URL: https://en.wikipedia.org/wiki/Haversine\_formula (дата обращения: 18.09.2024).