

**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 – 2**

**Студент гр. 23Б15-пу  
Трофимов И.А.**

**Преподаватель  
Дик А.Г.**

**Санкт-Петербург**  
**2024 г.**

## Оглавление

1. Цель работы	4
2. Описание задачи (формализация задачи)	4
3. Теоретическая часть	5
4. Основные шаги программы	6
5. Блок схема программы	7
6. Описание программы	8
7. Рекомендации пользователя	10
8. Рекомендации программиста	10
9. Исходный код программы	11
10. Контрольный пример	11
11. Вывод	12
12. Источники	13

## Цель работы

Целью лабораторной работы является разработка системы генерации датасета для списка визитов к врачу с учетом определенных требований и условий. Датасет должен включать личные данные пациентов, информацию о симптомах, анализах и специалистах, а также данные о платежных картах для оплаты визитов.

## Описание задачи (формализация задачи)

Задача состоит в создании датасета для списка визитов к врачу со следующими требованиями:

1. **ФИО:** Славянские имена и фамилии.
2. **Паспортные данные:** Русские, Казахские и Белорусские паспортные данные с уникальными значениями.
3. **Дата посещения врача:** В рабочее время и дни недели. Повторное посещение врача возможно только спустя 24 часа после выдачи анализов.
4. **Дата отъезда и приезда:** Случайные даты в пределах года.
5. **Симптомы:** словарь минимум из 5000 симптомов.
6. **Анализы:** словарь минимум из 250 анализов.
7. **Стоимость:** только в рублях, всех анализов одного человека.
8. **Дата получения анализов:** в рабочие дни и время.
9. **Количество строк датасета:** не меньше 50000.

## Теоретическая часть

Для создания датасета использованы несколько программных модулей:

1. Firstnames.py: Хранение списка имён, как мужских так и женских.
2. Lastnames.py: Хранение списка фамилий, как мужских так и женских.
3. Patronymic.py: Хранение списка отчеств, как мужских так и женских.
4. doctors.py: Хранение словарей с анализами и симптомами, ключом является специалист, а значением анализ/симптом
5. main.py: Главный файл, отвечает за ввод данных и вызывает функции генерации и сохранения в XML из generic.py
6. generic.py: Содержит функции для генерации данных.

Ограничения:

- Количество строк в датасете ограничивается вводом пользователя, но минимальное количество сгенерированных строк будет 50000.
- ФИО пассажиров только славянские.
- Паспортные данные уникальные и могут быть только российские, белорусские и казахские.
- Уникальность СНИЛС.
- Веса банков и платежных систем определяются пользователем.
- Одной и той же банковской картой можно платить не более пяти раз .

## Основные шаги программы

- 1) Запуск программы (main.py):
- 2) Пользователь вводит количество людей, веса банков и платежных систем.
- 3) Запускается генерация датасета.
  - a) Создается класс человека, поля которого содержат персональные данные, информацию о визитах и карте оплаты.
    - i) Генерация карт и платёжных систем с заданным распределением.
    - ii) Генерация персональных данных: Имя, фамилия, отчество, серия и номер паспорта.
    - iii) Генерация визитов с учётом того, что дата получения анализов должна быть на несколько дней позже, а также следующий визит позже 24 часов.
  - b) Класс человека добавляется в список.
- 4) Список записываются в файл data\_set.xml.

## Блок схема программы

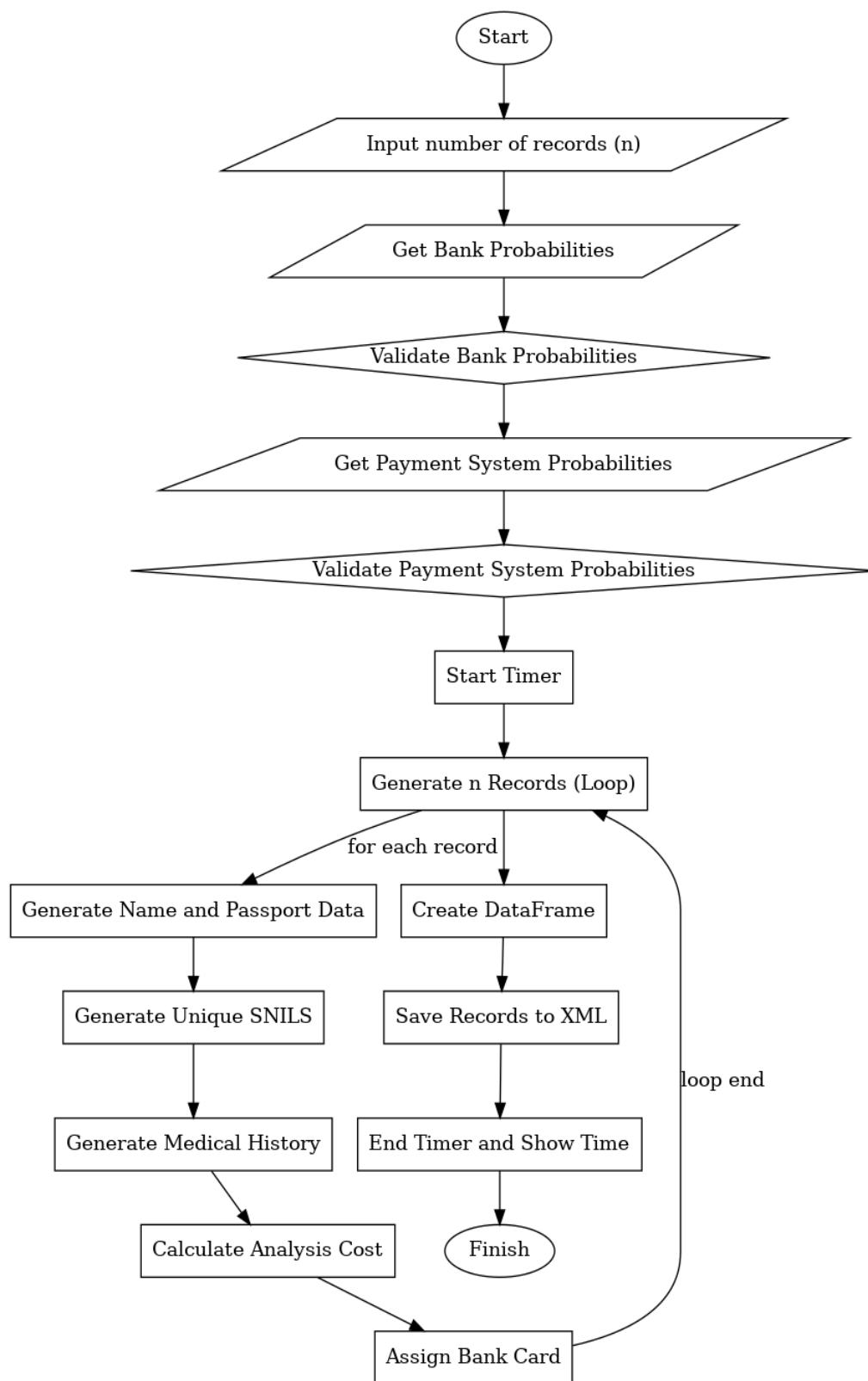


Рис 1. Блок-схема основной программы

## Описание программы

Программная реализация написана на языке Python 3.12.6 с использованием следующих библиотек: hashlib [\[1\]](#), random [\[2\]](#), datetime [\[3\]](#) xml [\[4\]](#) pandas [\[5\]](#), tqdm and time [\[6\]](#). Программа организована через парадигму ООП с акцентом на генерацию данных для списка визитов к врачу. В процессе разработки программы использовались 14 функций, каждая из которых имеет четко определенное назначение и 4 главных класса:

Функция/Класс	Описание	Возвращаемое значение
generate_passport	Генерация паспортных данных	tuple(int, int, str)
generate_snils	Генерация номера СНИЛС	string
get_bank	Генерация банков	list
get_pay_system	Генерация платежных систем	list
generate_visit_time	Генерация времени посещения врача и получения анализов	tuple(str, str)
generate_name	Генерация имен	str
generate_doctors	Генерация врачей	str
generate_symp	Генерация симптомов	str
generate_an	Генерация анализов	str

Таблица 1. generic.py



Функция	Описание	Возвращаемое значение
gen_history	Функция создающая историю посещения врача	list
ID.Card.generate_bank_card_number	Функция создающая уникальный номер карты	int
dict_to_xml	Функция для конвертации словаря в XML	xml.etree.ElementTree.Element
save_to_xml	Функция для сохранения записей из DataFrame в XML с отступами	None
show_time	Функция для вывода времени	None

Таблица 2. main.py

Класс	Описание	Возвращаемое значение
Класс ID	Класс, отвечающий за объект человека	None
Класс Passport	Класс, который является отражением паспорта, создан лишь для удобства	None

Класс Medicine	Класс, отвечающий медицинскую карту	None
Класс Card	Класс, отвечающий за характеристики оплаты	None

Таблица 3. main.py

### **Рекомендации пользователя**

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки. Код можно запустить в среде разработки. Запуск программы производится через файл main.py, который автоматически генерирует расписание в файл data\_set.xml. Важно периодически проверять корректность данных перед генерацией походов. Если вы хотите использовать собственные имена и/или симптомы и/или анализы, убедитесь, что вы редактируете соответствующие файлы Firstnames.py, Lastnames.py, Patronymic.py и doctors.py и их структура заголовков совпадает с требуемой: для имён это список, для симптомов и анализов это словарь. 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/hanglider/Labs/tree/11664007efebd35fde7eab8422df2aef393e75f0/1/1%20lite>

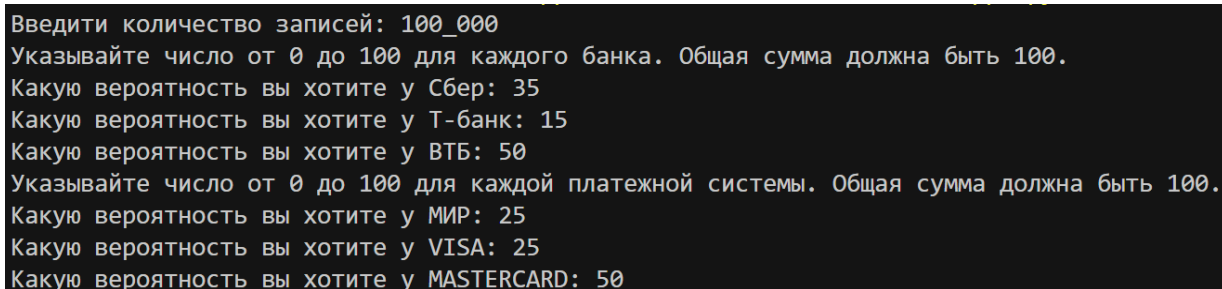
## Контрольный пример

### 1. Запуск программы

Для запуска программы используйте файл main.py. Программа будет отвечать за генерацию визитов к врачам на основе заданных данных о платежных системах и банках.

### 2. Ввод количества людей и весов платежных систем и банков

После запуска программы пользователю предложено ввести количество людей и веса для платежных систем и веса для банков (рис. 2). Веса определяют вероятность выбора той или иной платежной системы или банка при генерации билета.



```
Ввести количество записей: 100_000
Указывайте число от 0 до 100 для каждого банка. Общая сумма должна быть 100.
Какую вероятность вы хотите у Сбер: 35
Какую вероятность вы хотите у Т-банк: 15
Какую вероятность вы хотите у ВТБ: 50
Указывайте число от 0 до 100 для каждой платежной системы. Общая сумма должна быть 100.
Какую вероятность вы хотите у МИР: 25
Какую вероятность вы хотите у VISA: 25
Какую вероятность вы хотите у MASTERCARD: 50
```

Рис 2. пример ввода данных

### 3. Запуск генерации полей

После ввода количества билетов программа приступает к их генерации, используя введенные параметры, затем сгенерированный датасет сохраняется в dataset.xml. (рис. 3)

```

1  <?xml version="1.0" ?>
2  <record>
3      <Firstname>Вячеслав</Firstname>
4      <Lastname>Фролов</Lastname>
5      <Patronymic>Иванович</Patronymic>
6      <Passport_data>
7          <country>Казахстан</country>
8          <series>SM</series>
9          <number>1790679</number>
10     </Passport_data>
11     <snils>629-365-715 47</snils>
12     <med_card>
13         <med_card>
14             <doctor>Маммолог</doctor>
15             <symptoms>
16                 <item>Выделения из сосков</item>
17                 <item>Опухоли</item>
18             </symptoms>
19             <date>2023-04-03 08:04:00</date>
20             <date_offset>2023-04-05 12:04:00</date_offset>
21             <analyzes>
22                 <item>('Анализ на тест на аллергию на латекс', 1200)</item>
23             </analyzes>
24             <total_analysis_cost>1200</total_analysis_cost>
25             <bank_card>
26                 <pay_system>MASTERCARD</pay_system>
27                 <bank>Т-банк</bank>
28                 <bank_card_number>99</bank_card_number>
29             </bank_card>
30         </med_card>
31     </med_card>
32 </record>

```

Рис 3. пример одного визита

## Вывод

В рамках данной работы были исследованы принципы генерации синтетических данных, применительно к моделированию визитов людей к врачам. Было реализовано программное обеспечение для автоматической генерации датасета, включающего такие данные, как личные данные пациентов, информация о симптомах, анализах и платежных системах. Программа позволяет настраивать параметры генерации банковских карт оплаты, обеспечивая соответствие заданным требованиям и реалистичность получаемого датасета.

## Источники

1. Hashlibs's documentation // Hashlib URL: [hashlib — Secure hashes and message digests — Python 3.12.6 documentation](#)  
(дата обращения: 29.09.2024).
2. Random — Generate pseudo-random numbers // Python URL: [Generate pseudo-random numbers — Python 3.12.6 documentation](#)  
(дата обращения: 29.09.2024).
3. Datetime — The datetime module supplies classes for manipulating dates and times. // Python URL: [datetime — Basic date and time types — Python 3.12.6 documentation](#)  
(дата обращения: 29.09.2024).
4. XML Processing Modules // Python URL: [XML Processing Modules — Python 3.12.6 documentation](#)  
(дата обращения: 29.09.2024).
5. Pandas // Python URL: [pandas documentation — pandas 2.2.3 documentation \(pydata.org\)](#)  
(дата обращения: 29.09.2024)
6. Tqdm and Time // Tqdm URL: [tqdm documentationn](#)  
(дата обращения: 29.09.2024)  
// Time URL: [time — Time access and conversions — Python 3.12.6 documentation](#)  
(дата обращения: 29.09.2024)