**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе**

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

**на тему «Исследование хеш-функций с различными вводными условиями»**

**1 вариант**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б15 |  | Агишев А.Б. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г.**

**Содержание**

[1. Цель работы 2](#_Toc151158478)

[2. Задача 2](#_Toc151158479)

[3. Теоретическая часть 2](#_Toc151158480)

[4. Описание алгоритма 3](#_Toc151158481)

[5. Описание программы 5](#_Toc151158482)

[5.1 Описание функций 5](#_Toc151158483)

[6. Рекомендации пользователю 8](#_Toc151158484)

[7. Рекомендации программисту 8](#_Toc151158485)

[8. Контрольный пример 9](#_Toc151158486)

[9. Заключение 11](#_Toc151158487)

# **Цель работы**

Разработать программу, которая способна расшифровать набор данных, зашифрованный с помощью хеш-функции с использованием модификатора входа – соли, а также проанализировать решение аналогичной задачи при различных условиях.

1. **Задачи**

Приобрести и закрепить знания в области шифрования телефонных номеров. Написать программу для деобезличивания датасета. Протестировать программу на заданном варианте, а также с еще минимум 3-мя различными хеш-функциями, которыми зашифрован исходный деобезличенный набор. Определить от чего меняется скорость расшифровки.

# **Теоретическая часть**

*MD5 (Message Digest Algorithm 5)*:

* Описание: MD5 — это криптографический хэш-алгоритм, созданный для вычисления 128-битного хэша из произвольного объема данных. Он производит фиксированный 32-значный шестнадцатеричный хэш-код.
* Применение: использовался широко в прошлом для хэширования паролей и проверки целостности данных. Однако, MD5 уязвим к коллизиям (различным данным, создающим одинаковый хэш), и поэтому не рекомендуется для безопасных приложений.

*SHA-1 (Secure Hash Algorithm 1)*:

* Описание: SHA-1 — криптографический хэш-алгоритм, представляющий собой 160-битный хэш. Он является улучшением MD5 и производит более длинные хэши.
* Применение: ранее широко использовался, но с течением времени выявлены уязвимости, и он не рекомендуется для криптографических целей. SHA-1 подвержен коллизиям.

*SHA-256 (Secure Hash Algorithm 256-bit)*:

* Описание: SHA-256 — часть семейства SHA-2, созданного Национальным институтом стандартов и технологии (NIST). SHA-256 производит 256-битный хэш, что делает его более криптографически стойким по сравнению с MD5 и SHA-1.
* Применение: широко используется для безопасности данных, включая цифровые подписи, блокчейн и другие области, где требуется высокая стойкость к коллизиям.

*Брутфорс (brute force)* представляет собой метод взлома, при котором злоумышленник осуществляет попытки нахождения правильного пароля, путем перебора всех возможных комбинаций. Этот метод является прямолинейным и требовательным к ресурсам, поскольку включает в себя систематический перебор всех вариантов паролей. Применение брутфорса обосновано в ситуациях, когда отсутствуют альтернативные методы взлома, и может быть эффективен при работе с простыми паролями или недостаточными политиками безопасности.

*Соль (salt)* представляет собой случайную или уникальную строку, которая добавляется к паролю перед его хэшированием. Использование соли решает проблему коллизий (возможности различных данных создавать одинаковый хэш) и повышает стойкость к атакам, таким как использование таблиц радужных хэшей (rainbow tables). Применение соли имеет важное значение для предотвращения использования предварительно вычисленных таблиц, в которых содержатся хэши для всех возможных паролей. Она также усиливает уровень защиты от брутфорс-атак, даже при использовании слабых паролей. Однако стоит отметить, что для полноценной реализации соли требуется уникальная соль для каждого пользователя. Это также вносит дополнительные сложности в управление данными о соли. Несмотря на применение соли, она сама по себе не исключает возможность брутфорс-атак, но делает их более ресурсоемкими.

# **Описание алгоритма**

1. Деобезличивание датасета номеров телефонов с помощью инструмента hashcat.
2. Анализ номеров телефонов для выявления соли с использованием эталонных номеров.
3. Запись номеров телефонов без соли, полученных после деобезличивания, в файл для последующего использования.
4. Применение хеш-функций MD5, SHA1, SHA256 для хеширования номеров телефонов с различными значениями.
5. Проведение деобезличивания для каждого нового полученного датасета номеров телефонов с последующим анализом полученных результатов.

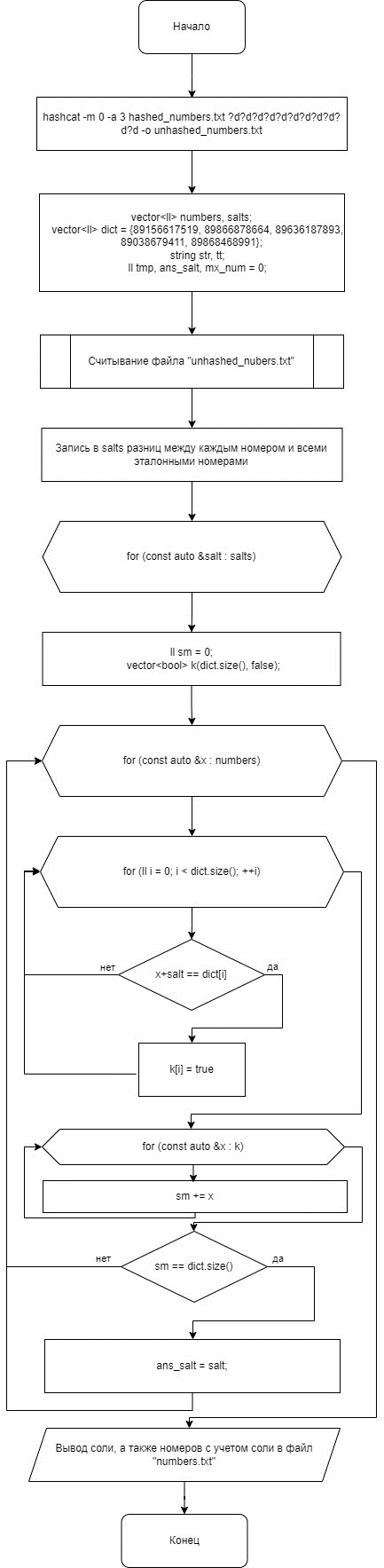


Рисунок 4.1. Блок-схема алгоритма деобезличивания датасета и нахождения соли.

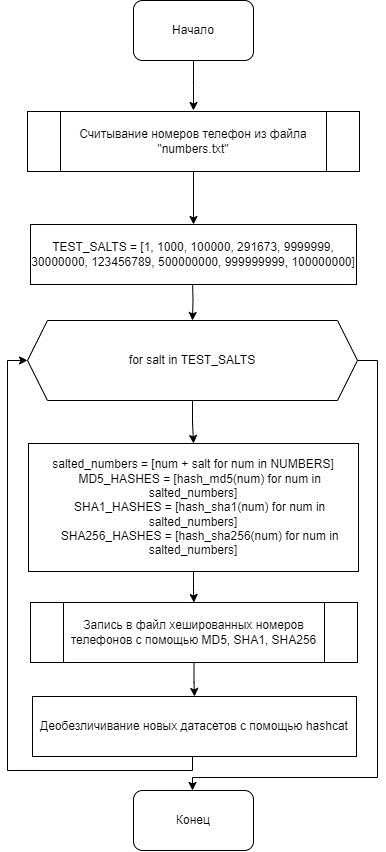


Рисунок 4.2. Блок-схема алгоритма деобезличивания датасета и нахождения соли.

# **Описание программы**

## **Описание функций**

В программе используется 6 функций: 2 связанных с интерфейсом и 3 – с генерацией данных. В таблице 5.1 представлено описание функций.

*Таблица 5.1. Описание функций*

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| Main | — | Создание интерфейса, вызов функции *«threaded\_run\_generator»* |
| threaded\_run\_generator | — | Запуск функции *«run\_generator»* в отдельном потоке |
| run\_generator | — | Проверка на корректность введенных данных, вызов функции *«generate\_dataset»* |
| generate\_dataset | Размерность, статус интерфейса, вероятностные коэффициенты выбора оператора связи, вероятностные коэффициенты выбора номеров операторов с домашним регионом Санкт-Петербург, вероятностные коэффициенты выбора работы на полставки | Генерация базы данных |
| generate\_random\_full\_name | Славянские мужские и женские ФИО | Генерация случайного ФИО |
| generate\_random\_phone\_number | Вероятностные коэффициенты выбора оператора связи, вероятностные коэффициенты выбора номеров операторов с домашним регионом Санкт-Петербург | Генерация случайного номера телефона |

Описание переменных функции *«generate\_dataset»* представлено в таблице 5.2.

*Таблица 5.2. Описание переменных функции «generate\_dataset»*

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| str[] | slavic\_male\_surnames | Список славянских мужских фамилий |
| str[] | slavic\_male\_names | Список славянских мужских имён |
| str[] | slavic\_male\_patronymics | Список славянских мужских отчеств |
| str[] | slavic\_female\_surnames | Список славянских женских фамилий |
| str[] | slavic\_female\_names | Список славянских женских имён |
| str[] | slavic\_female\_patronymics | Список славянских женских отчеств |
| str[] | saint\_petersburg\_addresses\_list | Содержимое файла *«saint\_petersburg\_addresses.txt»* |
| dict | saint\_petersburg\_addresses | Словарь всех улиц со всеми домами |
| str[][] | Dataset | Итоговая база данных |
| int | current\_amount | Текущее количество строк в базе |
| str | current\_address | Текущий адрес |
| int | number\_of\_employees | Количество работников, работающих на одном адресе |
| str | current\_job\_title | Текущая профессия |
| bool | is\_job\_half\_time | Работа на полставки |
| int | Salary | Зарплата |
| pd.DataFrame | Df | Итоговая таблица сгенерированных данных |
| dict | column\_widths | Название колонок в итоговой таблице и их ширина |
| worksheet | Worksheet | Название листа в Excel |

# **Рекомендации пользователю**

Поле *«Введите число N»* устанавливает количество записей в базе данных. По умолчанию начальное значение 50000.

Поля с заданием настроек выбора вероятностей случайных величин модифицируемые. По умолчанию указаны корректные величины. При неправильных настройках поля подсвечиваются красным цветом.

Кнопка *«Генерировать»* отвечает за запуск генерации. Программа выполнит генерацию синтетических данных на основе заданных настроек.

Данные сохраняются в файл *«dataset.xlsx»* в директории, хранящей в себе программу.

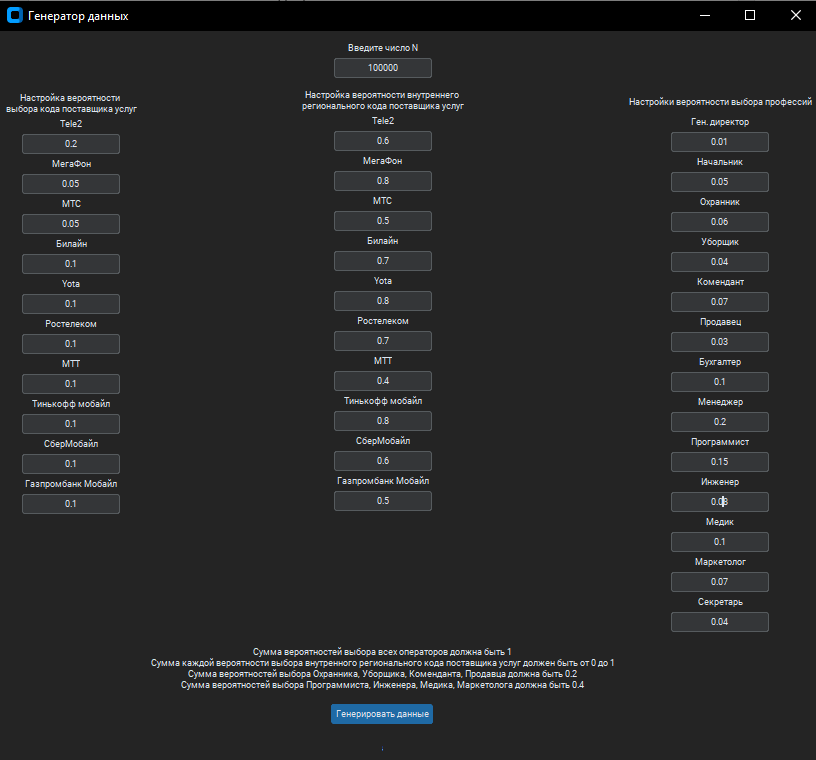
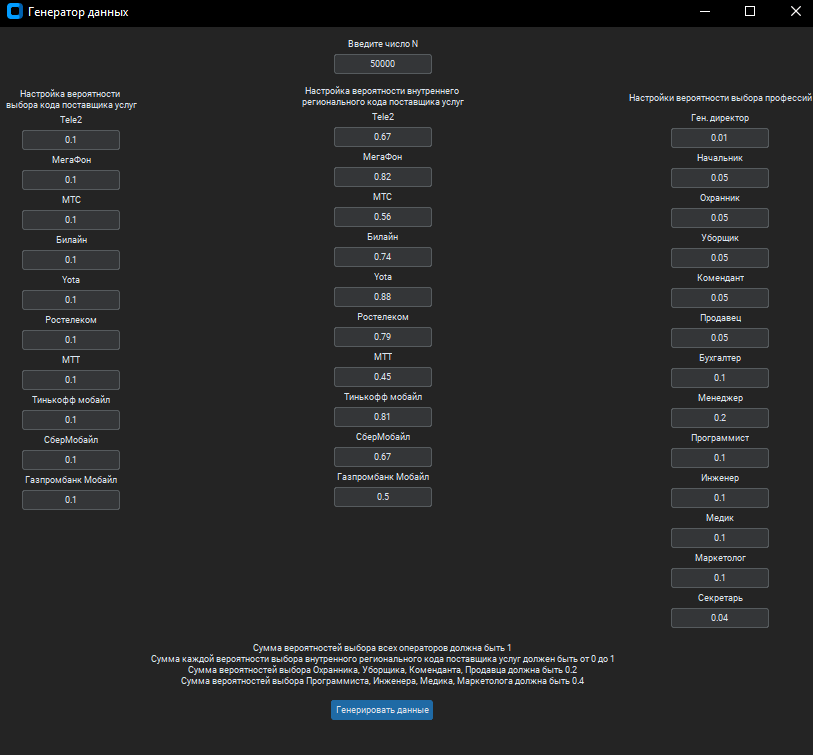
Для завершения работы нажмите на крестик в левом верхнем углу.

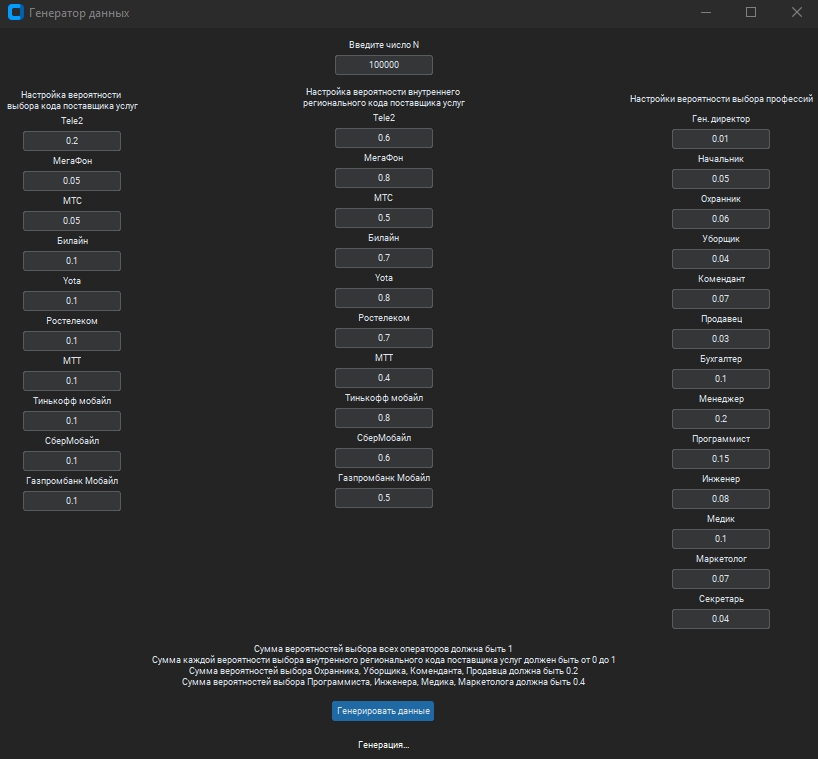
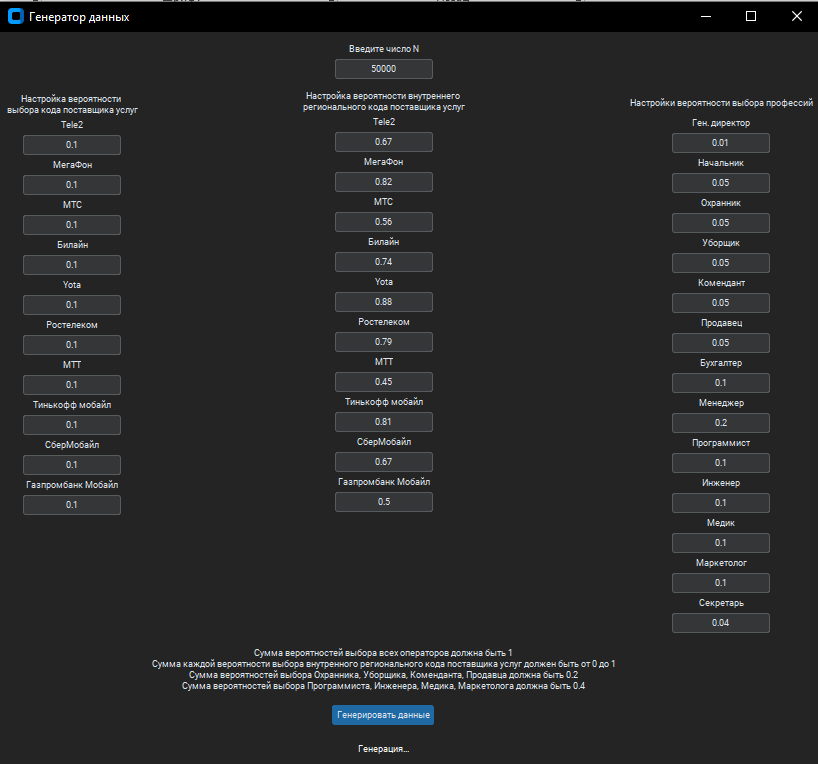
# **Рекомендации программисту**

Для запуска программы необходим Python версии не ниже 3.10.6, а также 64-битная операционная система Windows/Linux/macOS. Предварительно необходимо установить библиотеки: customtkinter версии не ниже 5.2.0, pandas версии не ниже 2.1.0. Для работы с кодом необходим PyCharm версии не ниже 2022.2.1.

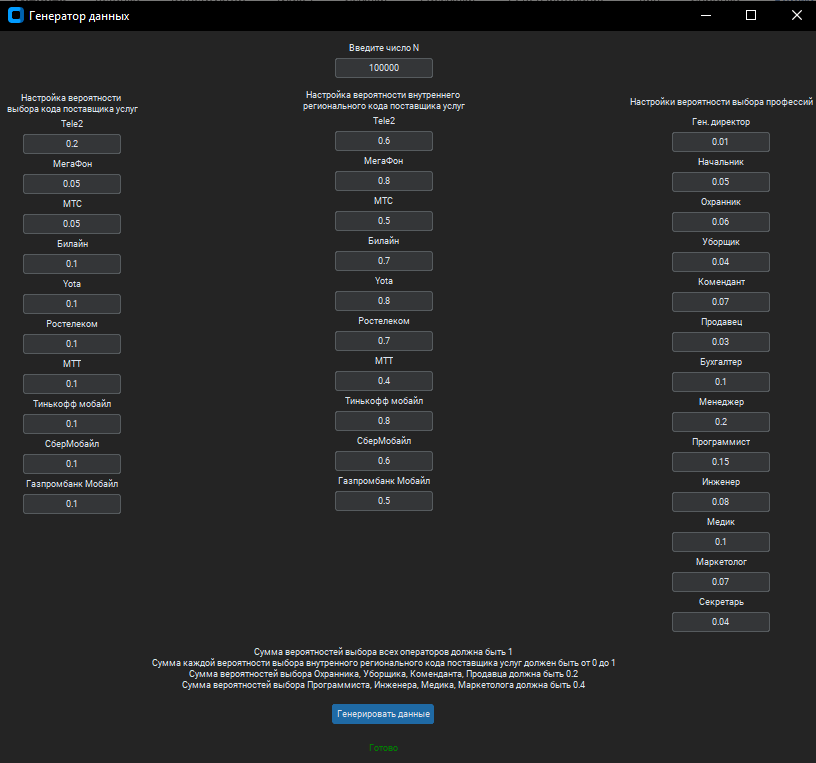
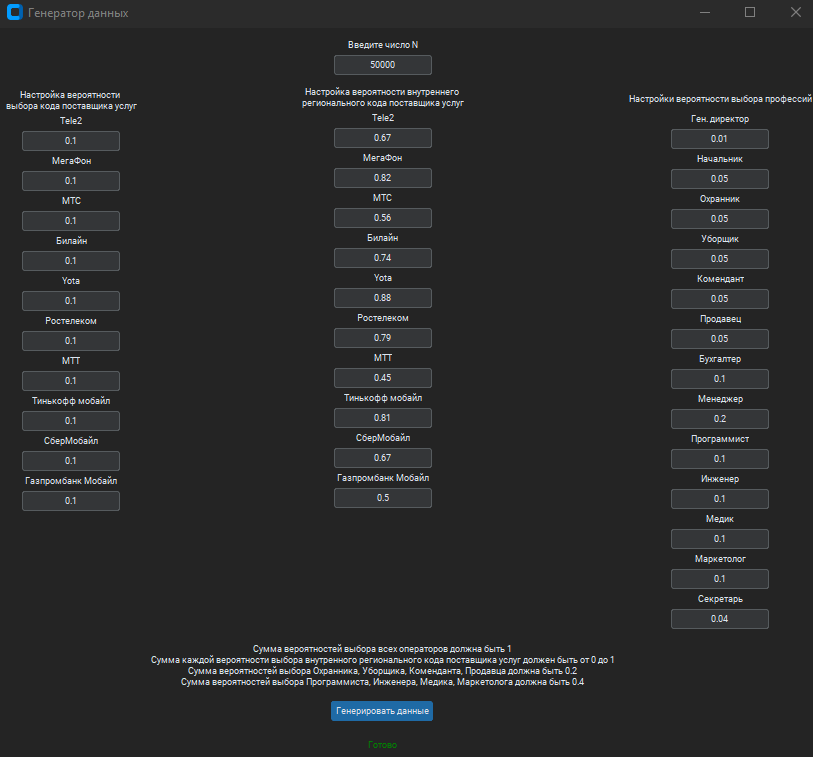
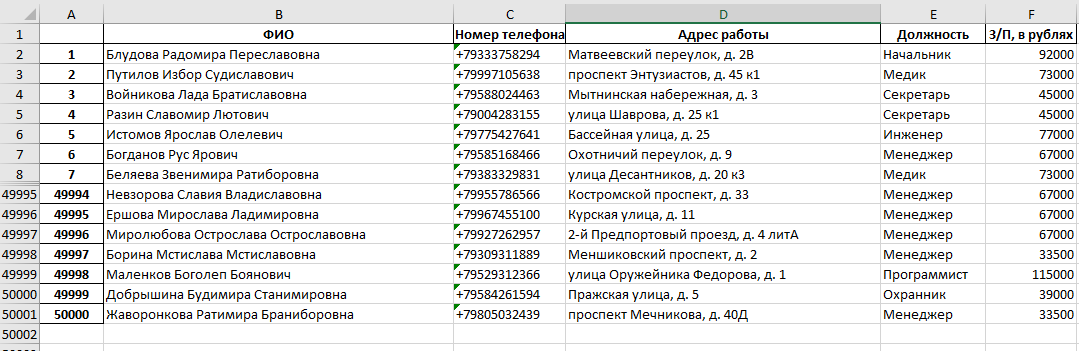
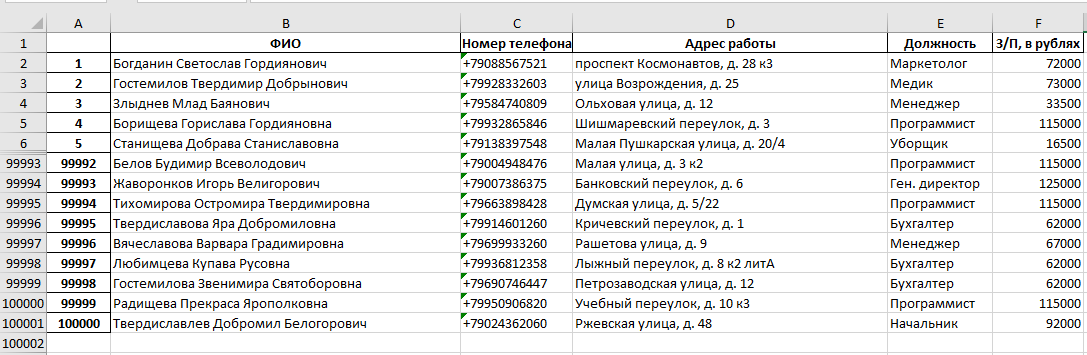
Минимальное необходимое место на диске: 3.5 МБ. Минимальное необходимое количество оперативной памяти: 400 МБ.

# **Контрольный пример**

 В данном разделе представлены два контрольных примера (описываются они параллельно), демонстрирующие способность программы генерировать данные, основываясь на ранее заданных параметрах.

*Рисунок 8.1–8.2. Интерфейс программы*

*Рисунок 8.3–8.4. Окно программы в процессе генерации*

*Рисунок 8.5–8.6. Окно программы после окончания генерации*

*Рисунок 8.7–8.8. Результат работы программы*

# **Заключение**

В рамках данной работы была успешно достигнута поставленная цель - разработка алгоритма и программы для генерации синтетических данных. Данная лабораторная работа позволила не только приобрести, но и закрепить знания в области генерации данных. Был создан набор синтетических данных, отвечающих определенным требованиям и характеристикам, что имеет важное значение для последующего использования в различных учебных целях.

|  |  |  |  |
| --- | --- | --- | --- |
| Значение соли | MD5, мин:сек | SHA1, мин:сек | SHA256, мин:сек |
| 1 | 9:43 | 19:21 | 38:29 |
| 1000 | 10:08 | 19:32 | 38:40 |
| 100000 | 10:54 | 20:06 | 39:01 |
| 291673 | 11:36 | 20:59 | 39:40 |
| 9999999 | 11:57 | 21:32 | 40:21 |
| 30000000 | 11:19 | 20:21 | 39:53 |
| 100000000 | 11:01 | 20:07 | 39:22 |
| 123456789 | 11:51 | 21:15 | 40:02 |
| 500000000 | 11:14 | 20:22 | 39:49 |
| 999999999 | 12:10 | 22:07 | 41:44 |