

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

**КУРСОВАЯ РАБОТА**

Тема: «СОЗДАНИЕ ANDROID ПРИЛОЖЕНИЯ С ЗАКОДИРОВАННОЙ БАЗОЙ ДАННЫХ И ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ СЖАТИЯ ДАННЫХ»

Дисциплина: «Теория информации»

Специальность: 10.05.01 Компьютерная безопасность

Специализация: Математические методы защит информации

Обозначение курсового работы ТИ.890000.000 Группа ВКБ31

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.С. Пыдык

подпись, дата

Курсовая работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель, И.А. Алферова

подпись, дата

Ростов-на-Дону

2022



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ЗАДАНИЕ**

на выполнение курсовой работы

Тема «СОЗДАНИЕ ANDROID ПРИЛОЖЕНИЯ С ЗАКОДИРОВАННОЙ БАЗОЙ ДАННЫХ И ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ СЖАТИЯ ДАННЫХ»

Дисциплина: Теория информации

Обучающийся: Пыдык София Сергеевна

Обозначение курсовой работы ТИ.890000.000 Группа: ВКБ31

Срок представления работы к защите «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Исходные данные для курсовой работы:

1. [С. М. Рацеев](https://www.litres.ru/s-m-raceev/): Элементы высшей алгебры и теории кодирования. Учебное пособие для вузов

|  |  |  |  |
| --- | --- | --- | --- |
| **Содержание пояснительной записки** | | | |
| Введение:  Актуальность решения задачи. i | | | |
| Разделы основной части: | | | |
| 1. Алгоритмы кодирования данных. Изучение свёрточного кодирования. 2. Разработка программного средства. Обоснование выбора языка и среды разработки. Архитектура и основные методы, классы. 3. Тестирование программного средства. | | | |
| Заключение:  Выводы по результатам разработки | | | |
| Руководитель работы | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | И.А. Алферова |
|  |  |  |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | С.С. Пыдык |

**Содержание**

[Введение 5](#_Toc105409551)

[1. Сверточное кодирование 6](#_Toc105409552)

[1.1 Общие сведения 6](#_Toc105409553)

[1.2 Сверточное кодирование 6](#_Toc105409554)

[2. Разработка программного средства 8](#_Toc105409555)

[2.1 Обоснование выбора языка и среды разработки 8](#_Toc105409556)

[2.2 Архитектура программного средства 8](#_Toc105409557)

[2.3 Основные модули и классы 9](#_Toc105409558)

[2.4 Демонстрация программного средства 10](#_Toc105409559)

[3. Тестирование программного средства 14](#_Toc105409560)

[Заключение 18](#_Toc105409561)

[Перечень используемых информационных ресурсов 19](#_Toc105409562)

[Приложение А 20](#_Toc105409563)

[Приложение Б Листинг кода 24](#_Toc105409564)

# **Введение**

Двоичные сверточные коды представляют собой, пожалуй, наиболее популярный вид помехоустойчивого кодирования. Эти коды нашли многочисленные применения. Изначально эти коды были названы сверточными кодами с проверками на четность. Наиболее распространенным, на сегодняшний день, методом их декодирования является алгоритм Витерби. Он применяется также для решения таких, эквивалентных по сложности задач, как восстановление последовательностей по максимуму правдоподобия или прием сигналов в каналах с частичными откликом. С недавних пор часто используется сверточное кодирование в сочетании с алгоритмами сжатия данных для эффективности очень близкой к пределу Шеннона.

В данной работе будут рассмотрены сверточное кодирование, алгоритм декодирования Витерби с его программной реализацией.

Объект исследования: сверточное.

Предмет исследования: реализация программного сверточное кодирование и средства сжатия данных двумя алгоритмами.

Цель работы заключается в создании программного средства с использованием виртуальной базы закодированных данных.

Задачи исследования:

1. Изучение необходимой теории об алгоритмах кодирования данных.
2. Реализация алгоритма сверточного кода.
3. Реализация программного средства.

# **1. Сверточное кодирование**

## **1.1 Общие сведения**

Существуют множество видов кодирования, одним из которых является сверточное кодирование. Сверточные коды – это коды, исправляющие ошибки, которые используют непрерывную, или последовательную, обработку информации короткими фрагментами (блоками).

Помимо кодирования часто используются различные алгоритмы сжатия данных. Алгоритмы сжатия данных делятся на два метода – искажающие и неискажающие. Искажающие методы сжатия могут искажать исходные данные, после чего полное восстановление невозможно. Неискажающие методы сжатия гарантируют, что декодированные данные будут в точности совпадать с исходными.

## **1.2 Сверточное кодирование**

Сверточные коды являются частным случаем решетчатых кодов. Можно также полагать, что решетка является просто другим способом представления и обычных сверточных кодов, которые могут быть систематическими и несистематическими. В последовательности кодовых символов систематического кода без изменения содержится последовательность информационных символов. В несистематическом коде положение информационных символов в кодовой последовательности указать нельзя. При сверточном кодировании символы кода распределяются во времени на значительном интервале, что повышает его устойчивость к одиночным пакетам ошибок. Кодер для сверточного кода представляет собой устройство с памятью, и добавляемые избыточные символы зависят не только от текущего фрагмента сообщения, но и от предыдущих фрагментов сообщения.

Сверточные коды относят к рекуррентным. При рекуррентном кодировании, кодируемую последовательность, вычисляют непрерывно по мере поступления символов по определенным рекуррентным соотношениям, выбранным для данного кода. В случае использования линейных рекуррентных соотношений получается сверточный код. Для сверточных кодов понятие кодового слова не имеет смысла, так как кодовые символы вычисляются по определенному отрезку последних информационных символов.

Методы декодирования сверточных кодов. В настоящее время известны три важнейших метода декодирования сверточных кодов: пороговое, последовательное и декодирование по максимуму правдоподобия (алгоритм Витерби).

# **2. Разработка программного средства**

## **2.1 Обоснование выбора языка и среды разработки**

Для данной работы в качестве языка для разработки был выбран Python.

Python – это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым человеческим языком, на Python можно написать практически что угодно: веб-приложения, игры, скрипты по автоматизации, комплексные системы расчёта, системы управления жизнеобеспечением и многое другое без ощутимых проблем. Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нём не писал. За счёт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее по сравнению с Java или C++. Также Python обладает большим количеством библиотек, технической документации и учебных материалов.

В заключении можно сказать что Python – понятный, простой в изучении, универсальный, востребованный и перспективный язык. Именно поэтому Python является отличным выбором для программиста любого уровня.

## **2.2 Архитектура программного средства**

Архитектура программной системы – это ее организационная структура, включающая модули, их внешние характеристики, а также отношения между модулями.

Требования Платформы (ОС, языки, библиотеки) делится на: производительность, масштабируемость и распределение функциональности по физическим узлам. А также простота поддержки, расширения, повторного использования модулей.

Модульная архитектура программного обеспечения, в которой модули связаны посредством программных интерфейсов, обладает большей гибкостью. Отдельные модули программного компонента можно заменять новыми для адаптации к новым условиям использования, исправления ошибок или добавления новых функций.

При регистрации пользователь вводит свои персональные данные в модальное окно-интерфейс программного средства. После этого пользователь может отправить свои персональные данные путем нажатия кнопки «регистр». С помощью алгоритмов сверточного кодирования и LZ78 данные преобразуются и сохраняются в базе данных.

В случае хранения персональной информации в базе данных, пользователь может авторизироваться в своем личном кабинете. Информация из модальных окон вновь кодируется для того чтобы верифицировать пользователя в системе. Если пользователь найден в системе, то он автоматически переходит в личный кабинет. При неудачной проверке интерфейс остается статичным.

Таким образом получаем систему, использующие технологии и методы разработки, которые будут корректно функционировать в различных мобильных и десктопных операционных системах.

## **2.3 Основные модули и классы**

on\_press: root.manager.current = 'login' – метод переход из одного окна приложения в другое.

class LoginMenu(Screen) – класс с функциями, осуществляющий сверку данных, введенных пользователем, с данными из базы.

con = sqlite3.connect('data1.db') – Модуль sqlite3, метод connect() – подключение к базе данных.

u\_for\_c = self.ids.name\_for\_reg.text – модуль для получения данных из TextInput для осуществления входа в личный кабинет.

self.insert\_varible\_into\_table(username, password) – модуль, который вносит новые закодированные данные в базу.

username = s.encode(m) – модуль для кодирования имя пользователя сверточным кодом.

App().run() – модуль компиляции и запуска приложения.

## **2.4 Демонстрация программного средства**

Программное средство состоит из четырех взаимосвязанных окон: меню, окна регистрации и входа в личный кабинет, личный кабинет. Реализовано с помощью библиотеки kivy на языке python, база данных sqlite3.

Меню состоит из виджетов: двух Button (кнопка), трех Label (текстовое окошко), в которых хранится описание и назначение каждой из кнопок. Первая кнопка переносит пользователя в меню входа в личный кабинет, вторая – в окно регистрации. Результат можно увидеть на рисунке 1.

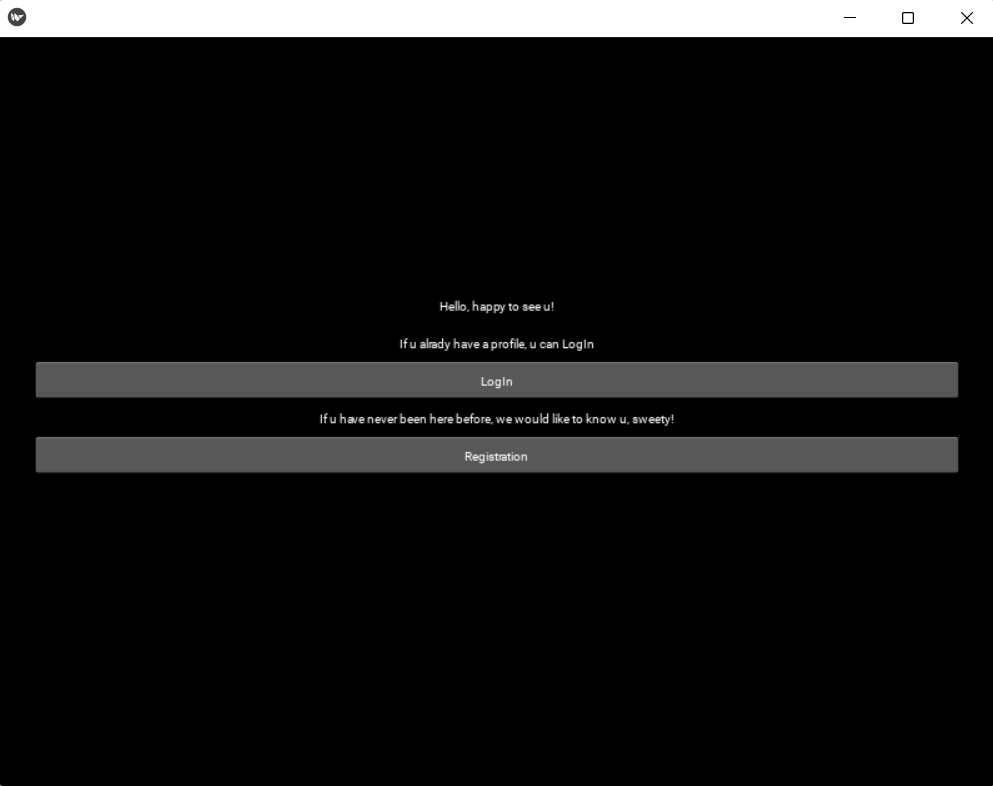


Рисунок 1 – Меню

В окне входа в личный кабинет, как и в окне регистрации, находятся 2 вида виджетов – Button (кнопка) и TextInput (окошко для ввода текста). В окошки для ввода текста пользователь записывает имя и пароль для регистрации или входа, если был зарегистрирован ранее. С помощью первой кнопки программа считывает введенные данные и, при совпадении с базой данных, переносит пользователя в личный кабинет (окно входа) или кодирует и добавляет в базу и переносит юзера назад в меню (окно регистрации). Вторая кнопка возвращает окно меню. Все это проиллюстрировано на рисунках 2 и 3.

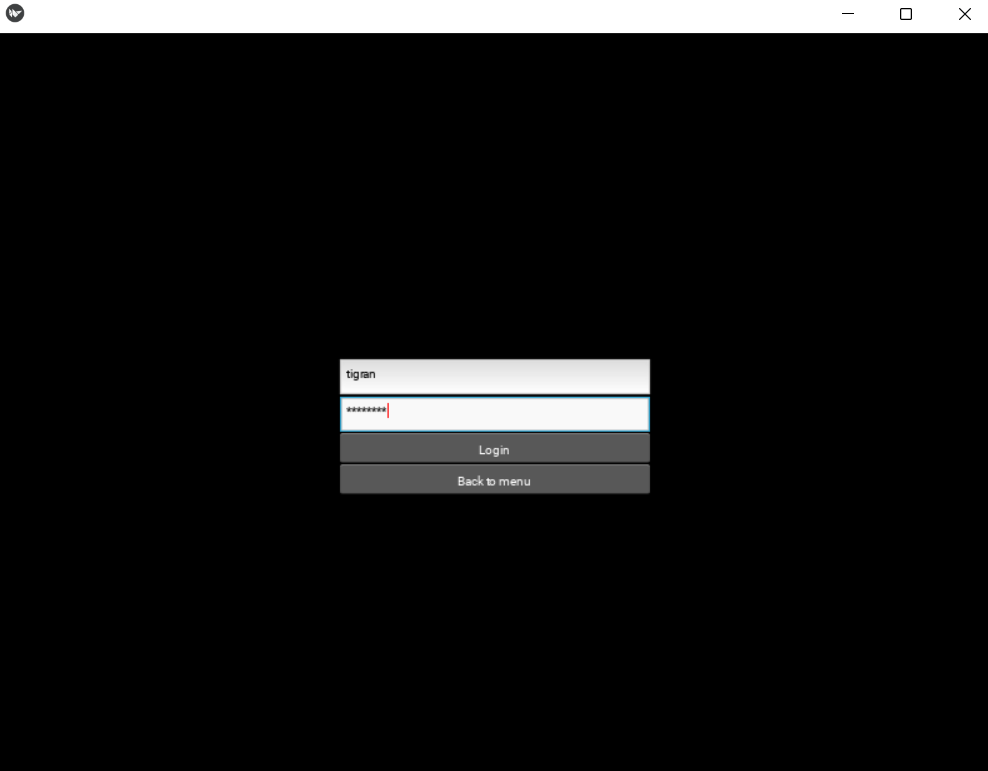


Рисунок 2 – Окно входа в личный кабинет.

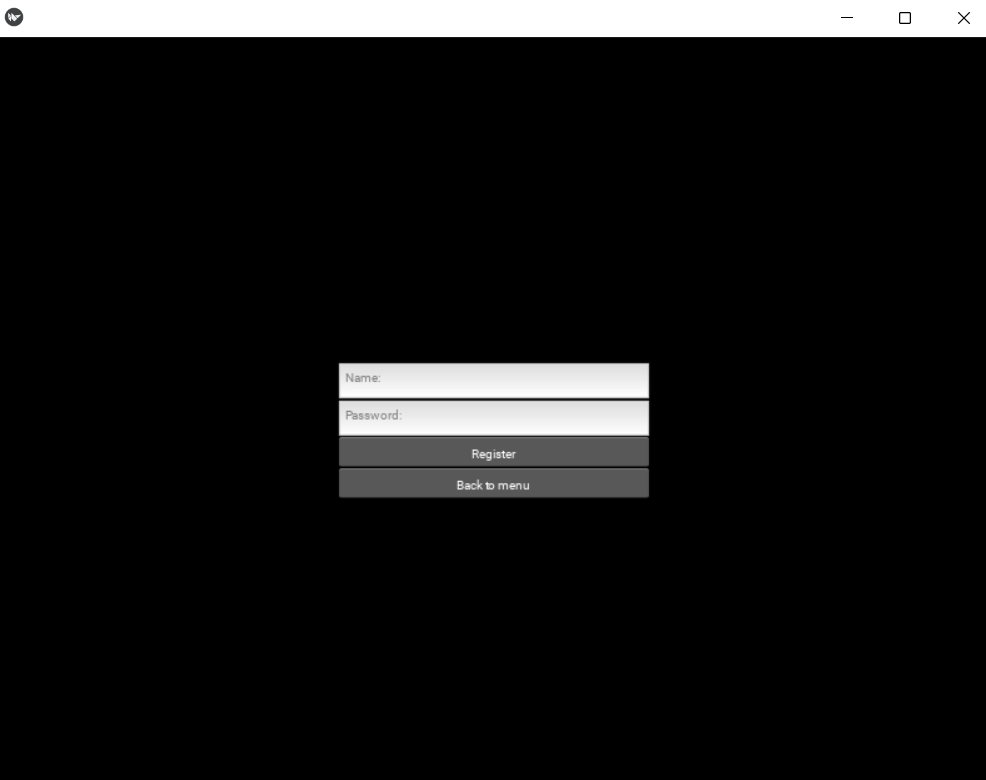


Рисунок 3 – Окно регистрации

Личный кабинет состоит из виджетов: одного Button (кнопка), одного Label (текстовое окошко). Кнопка выходит из личного кабинета и переносит пользователя назад в меню входа в личный кабинет. Результат работы можно увидеть на рисунке 4.

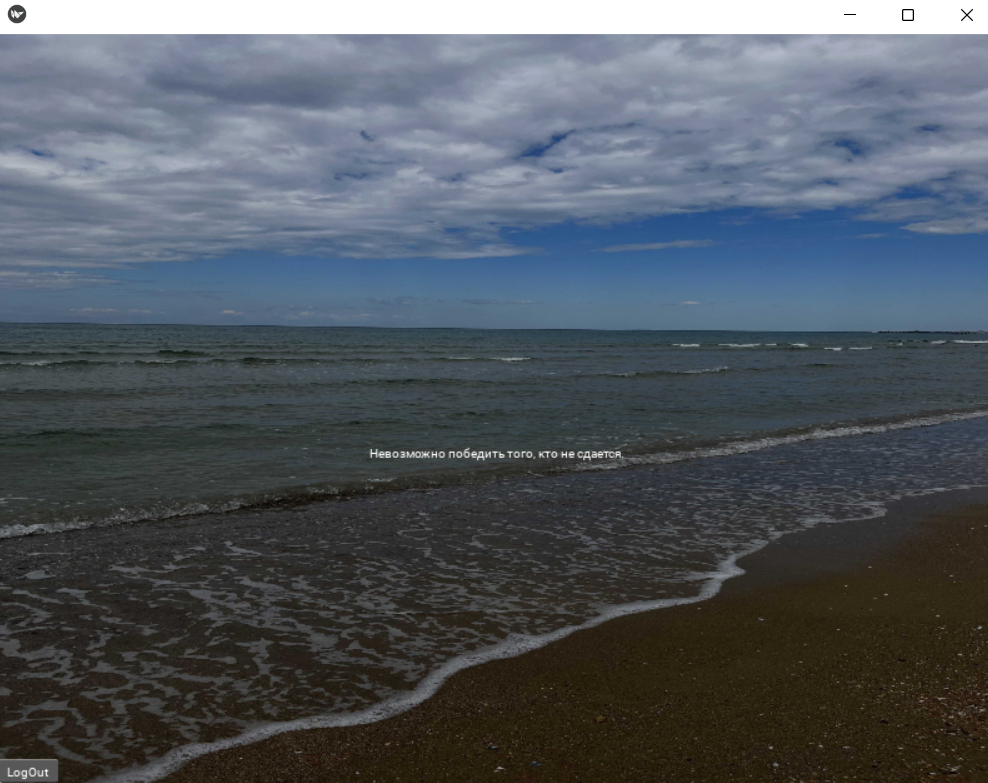


Рисунок 4 – Личный кабинет

Таким образом, мы получили графическое представление программного средства, а функции, связанные с работой описаны в других главах.

# **3. Тестирование программного средства**

Тестирование заключается в использовании разных данных для проверки работы созданных алгоритмов кодирования.

Тест 1: Текст, представленный на рисунке 5 – случайный набор иностранных букв. Результат выводится в базу данных, что демонстрируется на рисунке 6.

Пароль: abcdefghigklmnopqrstuvwxyz.

Имя пользователя: abcdefghigklmnopqrstuvwxyz.

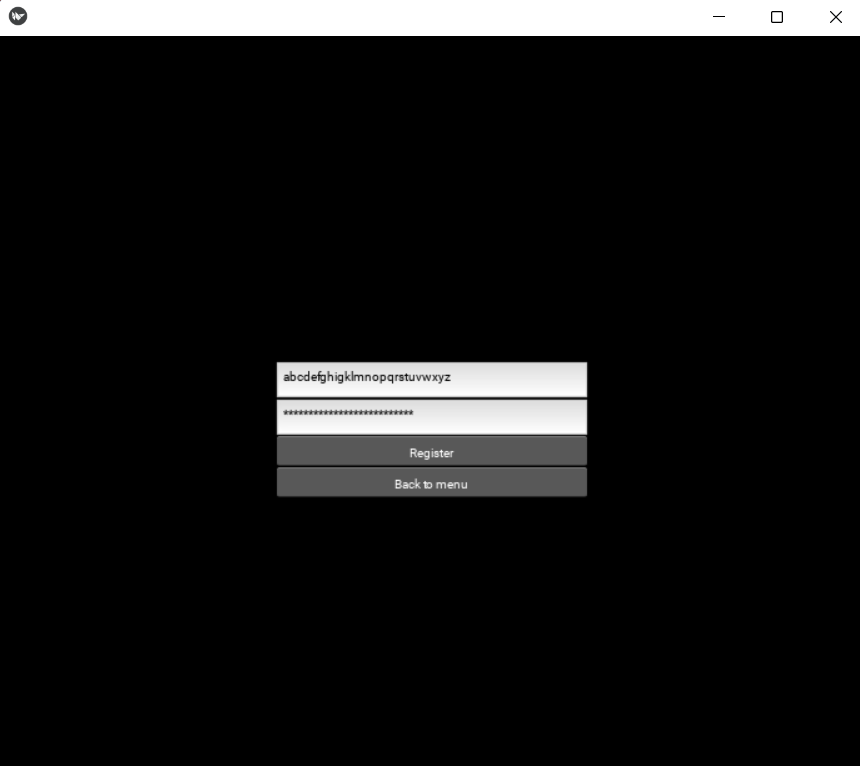


Рисунок 5 – Тест 1

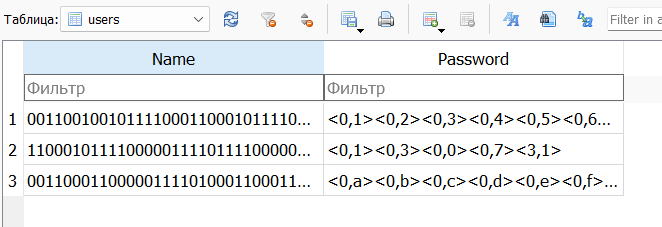


Рисунок 6 – Результат теста 1

Тест 2: Текст, представленный на рисунке 7 – случайный набор букв русского алфавита и цифр. Результат выводится в текстовые окна, как видно на рисунке 8.

Пароль: абвгдеёжзийклмнопрстуфхцчщъыьэюя1234567890.

Имя пользователя: абвгдеёжзийклмнопрстуфхцчщъыьэюя1234567890.

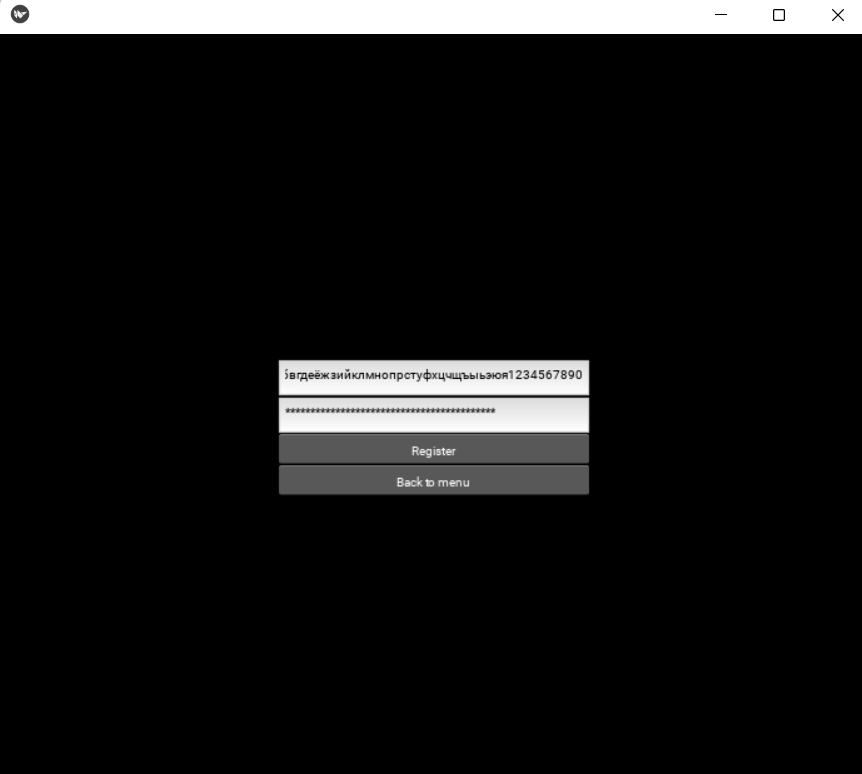


Рисунок 7 – Тест 2

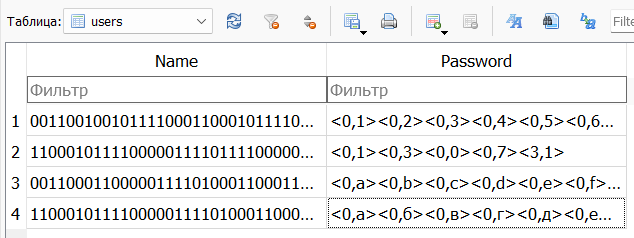


Рисунок 8 – Результат теста 2

Тест 3: Текст, представленный на рисунке 9 – случайный набор букв верхнего регистра русского и английского алфавита, а также символов. На рисунке 10 результат выводится в текстовые окна.

Пароль: ФмНкFqPo!@#$^.

Имя пользователя: ФмНкFqPo!@#$^.

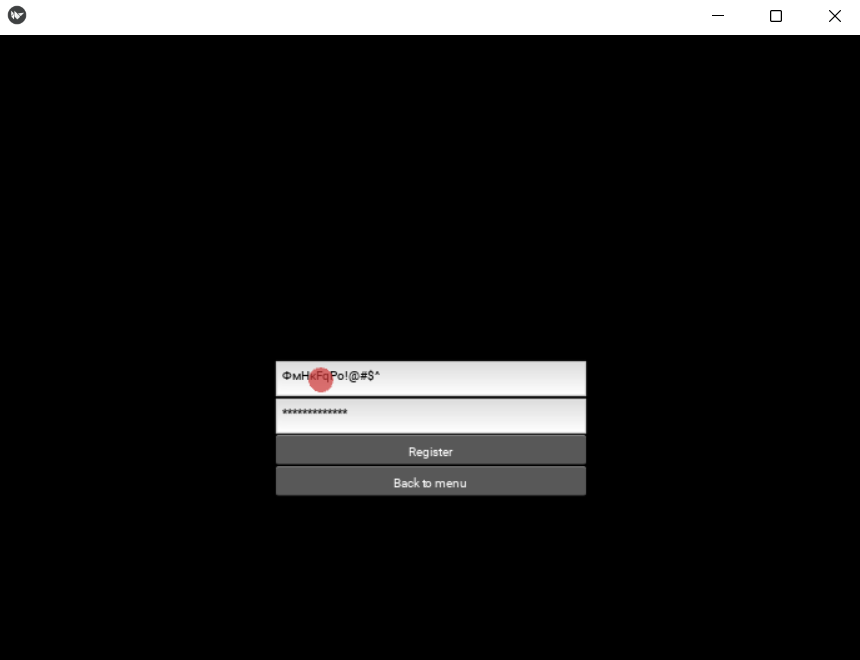


Рисунок 9 – Тест 3

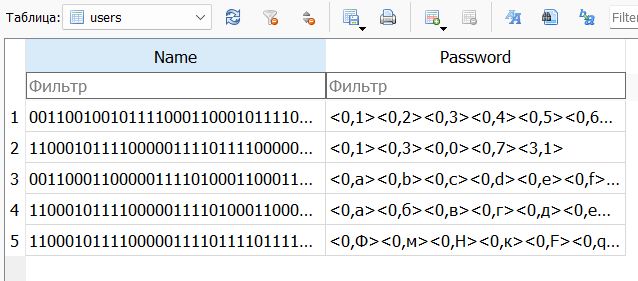


Рисунок 10 – Результат теста 3

Таким образом, можно сделать вывод, что алгоритмы работают корректно, как и все программное средство. Данные всех тестов были успешно внесены в базу и приложение работает с ними корректно.

# **Заключение**

В ходе курсовой работы были рассмотрен алгоритм сверточного кодирования.

Выполнены следующие поставленные задачи:

1. Изучена необходимая теория о сверточном кодировании.
2. Проанализирована работа сверточного кода.
3. Реализован алгоритм сверточного кодирования.
4. Реализовано программное средство с базой данных, в которой хранятся результаты работы сверхточного кода.

В результате выполнения курсовой работы была достигнута поставленная цель, заключающаяся в создании программного средства, обрабатывающего введенные данные с помощью алгоритма кодирования.

# **Перечень используемых информационных ресурсов**

1. Методы сжатия данных. [Электронный ресурс]. - URL:

<http://www.compression.ru/book/pdf/compression_methods_full_scanned.pdf> (Дата обращения: 07.03.2022)

1. Котенко В.В., Румянцев В.Е., Теория Информации // Издательство Южного федерального университета, 2018г.–241 c.
2. Сверточный кодирование. [Электронный ресурс]. - URL: https://kivy.org/doc/
3. Кудряшов, Б.Д. Теория информации: Учебник для вузов. // СПб.: Питер, 2009г. –320 с.
4. Алгоритмы сжатия данных без потерь [Электронный ресурс] URL: https://habr.com/ru (Дата обращения: 10.04.2022)
5. Учебник Сэломон. Д. Сжатие данных, изображения и звука. // Техносфера, 2006г – 368 с.
6. Ватодин Д., Ратушнях А., Смирнов М., Юкин В. Методы сжатия данных. // ДИАЛОГ-МИФИ, 2002г. - 384 c.

# **Приложение А**

Техническое задание

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СОГЛАСОВАНО | |  | |  | | «УТВЕРЖДЕНО» |
| Руководитель работы | |  | |  | | И.о. зав каф. «КБИС» |
| \_\_\_\_\_\_\_\_\_ И.А. Алферова |  | |  | |  | \_\_\_\_\_\_\_\_\_ к.т.н. Д.А. Короченцев |
| (подпись) |  | |  | |  | (подпись) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |  | |  | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**А.1 Введение**

**А.1.1 Наименование программы**

Наименование программы – «Приложение с кодированием персональных данных».

**А.1.2 Область применения**

Областью применения является любое приложение, в котором присутствует раздел для авторизации пользователя.

**А.2 Основания для разработки**

Разработка ведется в рамках выполнения курсовой работы по дисциплине «Теория информации». Тема разработки «Создание Android приложения с закодированной базой данных и использованием алгоритмов сжатия данных».

**А.3 Назначение разработки**

**А.3.1 Функциональное назначение**

Функциональным назначением программного средства является реализация приложения, содержащего алгоритмы кодирования обмена данных

в безопасном формате.

**А.3.2 Эксплуатационное назначение**

Эксплуатационное назначение: программное средство предназначено для эксплуатации в организациях, которым необходимо быстрая и качественная передача данных.

**А.4 Требования к программе**

**А.4.1 Требования к функциональным характеристикам**

Программное средство должно обрабатывать данные, вводимые пользователем с помощью двух алгоритмов кодирования и переносить их в базу данных.

1. convolotional – функция, реализующая алгоритм сверточного кодирования
2. insert\_varible\_into\_table – функция, реализующая перенос закодированных данных в базу данных

**А.4.2 Требования к надёжности**

Надёжность функционирования обеспечивается корректным функциони-рованием аппаратного обеспечения и правильным выбором входных данных.

**А.4.3 Условия эксплуатации**

Условия эксплуатации совпадают с условиями эксплуатации персональ-ных ЭВМ IBM PC и совместимых с ним персональных компьютеров. Требуется наличие одного оператора. Программа рассчитана на пользователя, владеющего базовыми навыками работы с персональным компьютером. Входные данные – текстовые сообщения. Результат работы не требует пояснений.

**А.4.4 Требования к составу и параметрам технических средств**

Для работы программы требуется наличие IBM PC – совместимого персонального компьютера с тактовой частотой процессора не менее 1,5 Гц.

**А.4.5 Требования к информационной и программной совместимости**

Базовый язык программирования Python 3.7.

**А.4.6 Требования к маркировке и упаковке**

Требований к маркировке и упаковке не предъявлялось.

**А.4.7 Требования к транспортированию и хранению**

Требования к транспортировке не предъявлялось.

**А.5 Требования к программной документации**

Программная документация должна включать документ «Техническое задание» (ГОСТ 19.201-78).

**А.6 Стадии и этапы разработки**

1. Системный анализ.

2. Общесистемное проектирование.

3. Подготовка технологических средств.

4. Программная реализация, рабочий проект.

5. Отладка программного средства в статике.

6. Тестовые испытания программного комплекса.

**А.7 Порядок контроля и приемки**

Контроль и приемка разработки осуществляется на основе испытаний контрольно-отладочных примеров. Примеры должны демонстрировать правильность работы используемых в программе структур данных и алгоритмов в различных ситуациях, которые могут возникнуть при выполнении программы. При этом проверяются выполнение всех функций программы и полнота документации.

|  |  |
| --- | --- |
| **Разработчик:** | Пыдык София Сергеевна |
| **Дата начала разработки:** | «10» февраля 2022 г. |

# **Приложение Б Листинг кода**

project.py:

import sqlite3

from kivy.app import App

from kivy.lang import Builder

from kivy.uix.screenmanager import ScreenManager, Screen

import sqlite3 as sql

from convolutional1 import \*

<…>

class MenuScreen(Screen):

pass

class LoginMenu(Screen):

def login(self):

s = convolotional()

from turbo import compression

username = s.encode(text\_bin(self.ids.login.text))

password = compression(self.ids.password.text)

con = sql.connect("data1.db")

cur = con.cursor()

statement = f"SELECT Name from users WHERE Name='{username}' AND Password = '{password}';"

cur.execute(statement)

if not cur.fetchone(): # An empty result evaluates to False.

print("Login failed")

else:

sm.current = 'ps'

return

class RegistrationMenu(Screen):

def insert\_varible\_into\_table(self, username, password):

try:

con = sqlite3.connect('data1.db')

cur = con.cursor()

print("Подключен к SQLite")

sqlite\_insert\_with\_param = """INSERT INTO users

(Name, Password)

VALUES (?,?);"""

data\_tuple = (username, password)

cur.execute(sqlite\_insert\_with\_param, data\_tuple)

con.commit()

print("Переменные Python успешно вставлены в таблицу")

cur.close()

except sqlite3.Error as error:

print("Ошибка при работе с SQLite", error)

finally:

if con:

con.close()

print("Соединение с SQLite закрыто")

sm.current = 'menu'

def registration(self):

s = convolotional()

from turbo import compression

u\_for\_c = self.ids.name\_for\_reg.text

p\_for\_c = self.ids.pass\_for\_reg.text

m = text\_bin(u\_for\_c)

username = s.encode(m)

password = compression(p\_for\_c)

con = sql.connect("data1.db")

cur = con.cursor()

self.insert\_varible\_into\_table(username, password)

class PS(Screen):

pass

sm = ScreenManager()

sm.add\_widget(MenuScreen(name='menu'))

sm.add\_widget(LoginMenu(name='login'))

sm.add\_widget(RegistrationMenu(name='registration'))

sm.add\_widget(PS(name='ps'))

class App(App):

def build(self):

return sm

if \_\_name\_\_ == '\_\_main\_\_':

App().run()

convolutional1.py:

import queue

class convolotional():

def \_\_init\_\_(self):

self.state = 00

def encode(self, string):

output = ''

self.state = 00

for i in string:

if i == '0' and self.state == 00:

output += '00'

elif i == '1' and self.state == 00:

output += '11'

self.state = 10

elif i == '0' and self.state == 10:

output += '11'

self.state = 1

elif i == '1' and self.state == 10:

output += '00'

self.state = 11

elif i == '0' and self.state == 1:

output += '10'

self.state = 00

elif i == '1' and self.state == 1:

output += '01'

self.state = 10

elif i == '0' and self.state == 11:

output += '01'

self.state = 1

elif i == '1' and self.state == 11:

output += '10'

self.state = 11

return output

def drived\_from(self, state):

if state == 0:

return [(0,'00', '0'),(1,'10', '0')]

if state == 1:

return [(3,'01', '0'),(2,'11', '0')]

if state == 2:

return [(0,'11','1'),(1,'01', '1')]

if state == 3:

return [(3,'10', '1'),(2,'00', '1')]

def calc\_diff(self, st1, st2):

r = 0

if st1[0] != st2[0]:

r += 1

if st1[1] != st2[1]:

r += 1

return r

def decode(self, string):

res = ['','','','']

pre\_res = ['','','','']

matrix = [[0 for x in range(int(len(string)/2) + 1)] for y in range(4)]

matrix[1][0] = 10000

matrix[2][0] = 10000

matrix[3][0] = 10000

p = [string[i] + string[i+1] for i in range(0,int(len(string)),2)]

for i in range(1, len(p) + 1):

for j in range(4):

status = self.drived\_from(j)

l = self.calc\_diff(p[i - 1], status[0][1]) + matrix[status[0][0]][i-1]

r = self.calc\_diff(p[i - 1], status[1][1]) + matrix[status[1][0]][i-1]

if l < r:

matrix[j][i] = l

res[j] = pre\_res[status[0][0]] + str(status[0][2])

else:

matrix[j][i] = r

res[j] = pre\_res[status[1][0]] + str(status[1][2])

for j in range(4):

pre\_res[j] = res[j]

# for x in range(4):

# print(matrix[x])

indx = 0

for i in range(4):

# print(res[i] , matrix[i][-1])

if matrix[i][-1] < matrix[indx][-1]:

indx = i

return res[indx]

def text\_bin(text,encoding="utf-8",errors = "surrogatepass"):

bits = bin(int.from\_bytes(text.encode(encoding,errors),"big"))[2:]

return bits.zfill(8\*((len(bits)+7)//8))

def textf(bits, encoding='utf-8', errors='surrogatepass'):

n = int(bits, 2)

return n.to\_bytes((n.bit\_length() + 7) // 8, 'big').decode(encoding, errors) or '\0'