**Н И Т У « М И С и С »**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ОТЧЁТ**

по

**ЛАБОРАТОРНОЙ РАБОТЕ**

**«Код Хемминга»**

Учебная дисциплина «Глобальные сети»

**Группа:** БПМ-16-2

**Студент:** Новицкий Дмитрий

**Вариант:** 12

**Преподаватель:** доц., к.т.н. Курочкин И. И.

**Отметка:**

**Дата защиты:**

**2019 г.**

Оглавление

[Задание 3](#_Toc24300268)

[Алгоритм работы программы 5](#_Toc24300269)

[Алгоритм передачи сообщения 5](#_Toc24300270)

[Алгоритм приёма сообщения 8](#_Toc24300271)

[Проверка работы программы 11](#_Toc24300272)

[Собственноручно написанный файлик 11](#_Toc24300273)

[Передача файла без ошибок 11](#_Toc24300274)

[Передача информации с не более, чем одной ошибкой в слове 12](#_Toc24300275)

[Передача файла с не более, чем двумя ошибками в слове 13](#_Toc24300276)

[Статья с Википедии об искусственных нейронных сетях 14](#_Toc24300277)

[Передача файла без ошибок 15](#_Toc24300278)

[Передача файла не более, чем с одной ошибкой в слове 16](#_Toc24300279)

[Передача файла с не более, чем двумя ошибками в слове 18](#_Toc24300280)

# Задание

1. С помощью сокетов реализовать передачу сообщений по локальной сети. Для определения хоста назначения необходимо использовать IP-адрес и порт (к примеру, 192.168.123.234:45102).
2. Закодировать передаваемое сообщение кодом Хемминга с определенной длиной слова. (длина слова – 53)
3. Иметь возможность добавлять в сообщение не более n ошибок на каждое слово (этот функционал должен быть реализован на стороне передачи сообщения).
4. Раскодировать сообщение, исправить одиночные ошибки и определить наличие множественных ошибок (>1) в словах.
5. Обратно передать сообщение с информацией о количестве исправленных ошибок, количество правильно и неправильно доставленных слов. Сравнить эту информацию с внесенными ошибками из п.3.

Сообщение должно быть длиной порядка 2000-3000 знаков и пробелов и содержать цифры, латинские и кириллические символы, знаки препинания. (Возьмите русскую статью с английскими терминами из Wikipedia, habr.com и т.д.)

Во время отладки Вы можете использовать специальный диапазон адресов 127.0.0.0/8

Предусмотреть возможность повторного запуска приложения с тем же портом.

Для проверки корректности работы нужно, как минимум, 3 раза отправить сообщение:

1. Без ошибок
2. С возможными ошибками (не более 1 на слово)
3. С множественными ошибками (более 1 на слово, но не обязательно во всех словах)

Загрузка в LMS в одном архиве:

1. Исходных кодов приложения/приложений;
2. Исполняемых файлов с необходимым окружением;
3. Мини-отчет в формате PDF с текстом сообщения, со снимками экрана, пояснениями, настройками и результатами.

Демонстрация лабораторной работы на семинаре на 2 разных ПК/ноутбуках, находящихся в одной локальной сети. Для демонстрации на семинаре могут быть использованы только материалы, предварительно загруженные в LMS.

# Алгоритм работы программы

Всего файлов для работы программы 4:

* Рабочий файл со стороны отправления сообщения
* Функции к рабочему файлу со стороны отправления сообщения
* Рабочий файл со стороны приёма сообщения
* Функции к рабочему файлу со стороны приёма сообщения

## Алгоритм передачи сообщения

Изначально задаётся путь к тексту, который необходимо отправить. Затем, с помощью метода get\_encoging(path) определяем кодировку текстового файла. После чего считываем файл с заданной кодировкой.

# Часть 1. Считывание текста с файла

print("Часть 1. Считывание текста с файла")

path\_1 = "texts/Задания.txt" # utf8

path\_2 = "texts/План работы.docx" # Неизвестная кодировка

path\_3 = "texts/Искусственная нейронная сеть.txt"

path\_4 = "texts/My\_text.txt"

# Определяем кодировку текста в файле

encoding = functions.get\_encoding(path\_4)

en = encoding['encoding']

if(print\_information):

print("Кодировка файла - ", en)

# Считываем текст с заданной кодировкой

file\_text = functions.read\_file(path\_4, en) # Исходный файл

if(print\_information):

print("file =")

print(file\_text)

print()

print("Считывание текста с файла окончено")

print()

Затем с помощью полученной ранее кодировки переводим каждый символ текста в список десятичных чисел, в котором каждое число соответствует заданному символу заданной кодировке.

# Часть 2. Переводим текст в список цифр в десятичной СС

print("Часть 2. Переводим текст в список цифр в десятичной СС")

file\_code = list(file\_text.encode(en))

if(print\_information):

print("file\_text = ")

print(file\_code)

print()

print("Перевод текста в список цифр в десятичной СС окончен")

print()

Далее из полученного списка составляем список двоичных значений полученных ранее десятичных чисел.

# Часть 3. Переводим текст из списка цифр в десятичной СС в список цифр в двоичной СС

print("Часть 3. Переводим текст из списка цифр в десятичной СС в список цифр в двоичной СС")

# Поиск максимального числа

max\_length = 0

for i in range(len(file\_code)):

if(file\_code[i] > max\_length):

max\_length = file\_code[i]

if(print\_information):

print("max\_length = ")

print(max\_length)

print()

# Поиск длины слова в доичном виде

word\_length = math.ceil(math.log(max\_length, 2))

if(print\_information):

print("word\_length = ")

print(word\_length)

print()

# Преобразование в двоичную СС

bit\_mas = functions.get\_bit\_mas(file\_code, word\_length)

if(print\_information):

print("bit\_mas = ")

print(bit\_mas)

print("len(bit\_mas) = ", len(bit\_mas))

print("Перевод текста из списка цифр в десятичной СС в список цифр в двоичной СС окончен")

print()

Применяем к данному списку алгоритм кода Хемминга.  
# Часть 4. Применяем кодировку Хемминга

print("Часть 4. Применяем кодировку Хемминга")

start\_time = time.time()

# Длина слова Хемминга до вставки контрольных битов

length\_hamming\_word\_input = 53

# Длина слова Хемминга после вставки контрольных битов

length\_hamming\_word\_output = length\_hamming\_word\_input + math.floor(math.log(length\_hamming\_word\_input, 2)) + 1

Hamming\_mas = functions.Hamming\_code(bit\_mas, length\_hamming\_word\_input, 2)

if(print\_information):

print("Hamming\_mas = ")

print(Hamming\_mas)

print()

print("Длина Hamming\_mas = ")

print(len(Hamming\_mas))

print()

end\_time = time.time()

print("Применение кодировки Хемминга окончено")

print("Время, потраченное на эту часть равно ", end\_time - start\_time, "секунд")

print()

Затем отправляем данные на заданный ip-адрес и заданные порты компьютера (ноутбука).

# Отправляем исходный текст

sock\_1 = socket.socket()

#sock\_1.connect(('192.168.0.102', 9090))

sock\_1.connect(('127.0.0.1', 9090))

sock\_1.send(bytes(Hamming\_mas))

sock\_1.close()

# Отправляем значение длины слова для кода Хемминга

sock\_2 = socket.socket()

#sock\_2.connect(('192.168.0.102', 49100))

sock\_2.connect(('127.0.0.1', 49100))

sock\_2.send(bytes(functions.dec\_bin(length\_hamming\_word\_input)))

sock\_2.close()

# Отправляем значение кодировки исходного текста

# Преобразуем данные в необходимый для отправки вид

en\_code = list(en.encode('utf8'))

en\_bit\_mas = functions.get\_bit\_mas(en\_code, 16)

if(print\_information):

print("en = ", en)

print("type(en) = ", type(en))

print("en\_code = ", en\_code)

print("en\_bit\_mas = ", en\_bit\_mas)

sock\_3 = socket.socket()

#sock\_3.connect(('192.168.0.102', 49101))

sock\_3.connect(('127.0.0.1', 49101))

sock\_3.send(bytes(en\_bit\_mas))

sock\_3.close()

# Принимаем значение длины слова для кодировки

sock\_4 = socket.socket()

#sock\_4.connect(('192.168.0.102', 49102))

sock\_4.connect(('127.0.0.1', 49102))

sock\_4.send(bytes(functions.dec\_bin(word\_length)))

sock\_4.close()

print("Отправка завершена")

print()

После чего принимаем информацию о количестве верно и неверно принятых слов.  
# Часть 6. Приём ответной информации о переданном файле

print("Часть 6. Ожидаем приёма ответной информации о переданном файле")

start\_time = time.time()

# Принимаем информацию о количестве правильно принятых слов

sock\_1 = socket.socket()

#sock\_1.bind(('192.168.0.103', 49110))

sock\_1.bind(('127.0.0.1', 49110))

sock\_1.listen(1)

conn\_1, addr\_1 = sock\_1.accept()

count\_of\_true\_words = functions.bin\_dec(list(conn\_1.recv(1024)))

conn\_1.close()

# Принимаем информацию о количестве неправильно принятых слов

sock\_2 = socket.socket()

#sock\_2.bind(('192.168.0.103', 49111))

sock\_2.bind(('127.0.0.1', 49111))

sock\_2.listen(1)

conn\_2, addr\_2 = sock\_2.accept()

count\_of\_false\_words = functions.bin\_dec(list(conn\_2.recv(1024)))

conn\_2.close()

# Принимаем информацию о количестве исправленных слов

sock\_3 = socket.socket()

#sock\_3.bind(('192.168.0.103', 49112))

sock\_3.bind(('127.0.0.1', 49112))

sock\_3.listen(1)

conn\_3, addr\_3 = sock\_3.accept()

count\_of\_corrected\_words = functions.bin\_dec(list(conn\_3.recv(1024)))

conn\_3.close()

# Принимаем информацию о количестве неисправленных слов

sock\_4 = socket.socket()

#sock\_4.bind(('192.168.0.103', 49113))

sock\_4.bind(('127.0.0.1', 49113))

sock\_4.listen(1)

conn\_4, addr\_4 = sock\_4.accept()

count\_of\_uncorrected\_words = functions.bin\_dec(list(conn\_4.recv(1024)))

conn\_4.close()

end\_time = time.time()

print("Данные приянты")

print("Время ожидания отклика составило", end\_time - start\_time, "секунд")

print()

# Вывод на экран полученных данных

if(print\_information):

print("Количество правильно переданных слов -", count\_of\_true\_words)

print("Количество неправильно переданных слов -", count\_of\_false\_words)

print("Количество исправленных слов -", count\_of\_corrected\_words)

print("Количество неисправленных слов -", count\_of\_uncorrected\_words)

else:

file = open("texts/accepted\_text\_information.txt", "w")

s = "Количество правильно переданных слов - " + str(count\_of\_true\_words) + "\n"

s = s + "Количество неправильно переданных слов - " + str(count\_of\_false\_words) + "\n"

s = s + "Количество исправленных слов - " + str(count\_of\_corrected\_words) + "\n"

s = s + "Количество неисправленных слов - " + str(count\_of\_uncorrected\_words)

file.write(s)

file.close()

## Алгоритм приёма сообщения

Принимаем отправленное сообщение.  
# Часть 1. Работа с сокетами (приём данных)

print("Ожидаем приёма текста")

start\_time = time.time()

# Принимаем исходный текст

Hamming\_mas = []

sock\_1 = socket.socket()

#sock\_1.bind(('192.168.0.103', 9090))

sock\_1.bind(('127.0.0.1', 9090))

sock\_1.listen(1)

conn\_1, addr\_1 = sock\_1.accept()

while True:

data = list(conn\_1.recv(1024))

for i in range(len(data)):

Hamming\_mas.append(data[i])

if not data:

break

conn\_1.close()

# Принимаем значение длины слова для кода Хемминга

sock\_2 = socket.socket()

#sock\_2.bind(('192.168.0.103', 49100))

sock\_2.bind(('127.0.0.1', 49100))

sock\_2.listen(1)

conn\_2, addr\_2 = sock\_2.accept()

length\_hamming\_word\_input = functions.bin\_dec(list(conn\_2.recv(1024)))

conn\_2.close()

# Принимаем значение кодировки исходного текста

sock\_3 = socket.socket()

#sock\_3.bind(('192.168.0.103', 49101))

sock\_3.bind(('127.0.0.1', 49101))

sock\_3.listen(1)

conn\_3, adrr\_3 = sock\_3.accept()

encoding = functions.get\_ascii\_code((conn\_3.recv(1024)), 16)

encoding = functions.get\_text(encoding, 16, 'utf8')

conn\_3.close()

# Принимаем значение длины слова для кодировки

sock\_4 = socket.socket()

#sock\_4.bind(('192.168.0.103', 49102))

sock\_4.bind(('127.0.0.1', 49102))

sock\_4.listen(1)

conn\_4, adrr\_4 = sock\_4.accept()

word\_length = functions.bin\_dec(list(conn\_4.recv(1024)))

end\_time = time.time()

print("Данные приняты")

print("Время ожидания составило", end\_time - start\_time, "секунд")

print()

Затем проверяем и расшифровываем текст.  
# Часть 3. Проверка и расшифровка текста

print("Часть 3. Проверка и расшифровка текста")

start\_time = time.time()

# Обнаруживаем ошибки и исправляем, если они есть

bit\_mas\_new, count\_of\_true\_words, count\_of\_false\_words, count\_of\_corrected\_words, count\_of\_uncorrected\_words = functions.Repeat\_Hamming\_code(Hamming\_mas, length\_hamming\_word\_input)

# Переводим текст в список цифр в десятичной СС

file\_code\_2 = functions.get\_ascii\_code(bit\_mas\_new, word\_length)

if(print\_information):

print("file\_code\_2 = ")

print(file\_code\_2)

print()

max = 0

for i in range(len(file\_code\_2)):

if(file\_code\_2[i] > max):

max = file\_code\_2[i]

if(print\_information):

print("max = ", max)

print()

# Декодируем исходное сообщение

try:

text = bytes(file\_code\_2)

text = text.decode(encoding)

except UnicodeDecodeError:

print("Из-за проблем с кодировкой расшифровать сообщение, в котором хотя бы в одном слове более двух ошибок, невозможно.")

print("Мне очень жаль.")

count\_of\_true\_words = 0

count\_of\_false\_words = 0

count\_of\_corrected\_words = 0

count\_of\_uncorrected\_words = 0

text = ""

if(print\_information):

print("text = ")

print(text)

print()

else:

file = open("texts/accepted text.txt", "w", encoding = encoding)

file.write(text)

file.close()

print("Проверка и расшифровка текста окончена")

print("Время проверки и расщифровки составило", end\_time - start\_time, "секунд")

print()

После чего отправляем информацию о принятом текстовом файле.  
# Часть 4. Отправка информации о принятом файле

print("Часть 4. Отправка информации о принятом файле")

# Отправляем информацию о количестве правильно принятых слов

sock\_1 = socket.socket()

#sock\_1.connect(('192.168.0.102', 49110))

sock\_1.connect(('127.0.0.1', 49110))

sock\_1.send(bytes(functions.dec\_bin(count\_of\_true\_words)))

sock\_1.close()

# Отправляем информацию о количестве неправильно принятых слов

sock\_2 = socket.socket()

#sock\_2.connect(('192.168.0.102', 49111))

sock\_2.connect(('127.0.0.1', 49111))

sock\_2.send(bytes(functions.dec\_bin(count\_of\_false\_words)))

sock\_2.close()

# Отправляем информацию о количестве исправленных слов

sock\_3 = socket.socket()

#sock\_3.connect(('192.168.0.102', 49112))

sock\_3.connect(('127.0.0.1', 49112))

sock\_3.send(bytes(functions.dec\_bin(count\_of\_corrected\_words)))

sock\_3.close()

# Отправляем информацию о количестве неисправленных слов

sock\_4 = socket.socket()

#sock\_3.connect(('192.168.0.102', 49113))

sock\_4.connect(('127.0.0.1', 49113))

sock\_4.send(bytes(functions.dec\_bin(count\_of\_uncorrected\_words)))

sock\_4.close()

print("Отправка завершена")

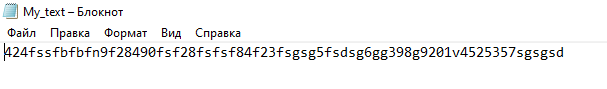
# Проверка работы программы

Проверим работу программы при передаче сообщения без ошибок, с не более, чем одной ошибкой и не более, чем с двумя ошибками.

Проверку будем осуществлять на двух файлах:

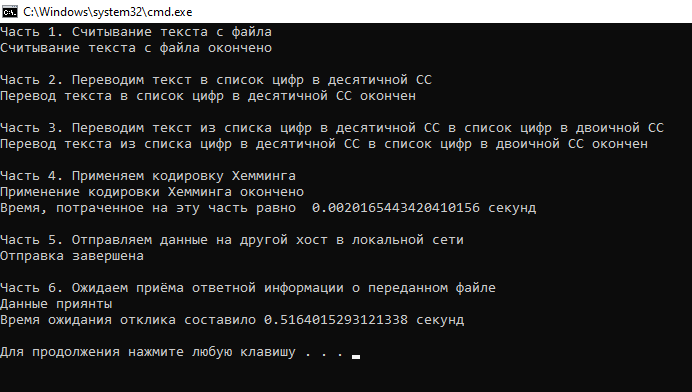
* Собственноручно написанный небольшой текстовый файлик
* Статья с Википедии об искусственных нейронных сетях

## Собственноручно написанный файлик

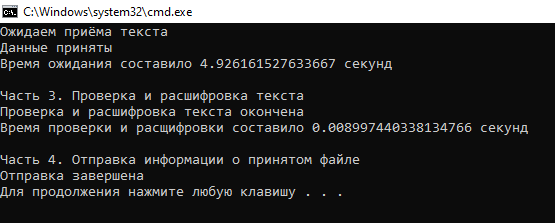


*Рис. 1. Мой текст.*

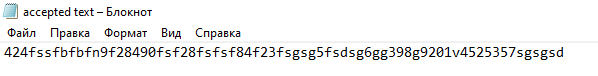
## Передача файла без ошибок



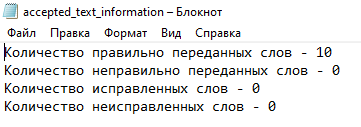
*Рис. 2. Результаты работы на стороне передачи.*



*Рис. 3. Результаты работы на стороне приёма.*

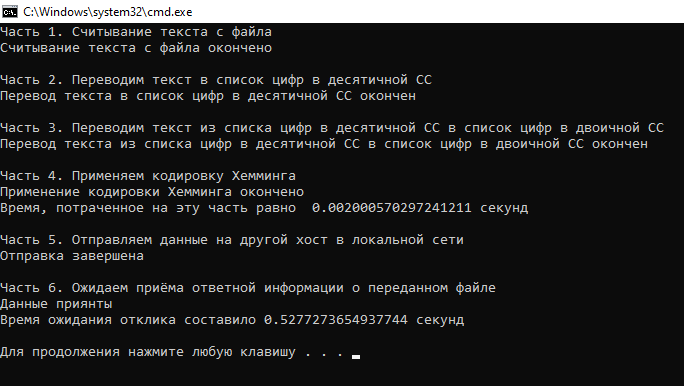


*Рис. 4. Принятый текст.*

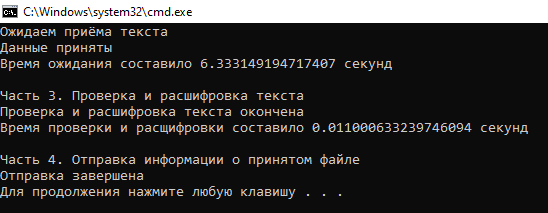


*Рис. 5. Информация о переданном тексте.*

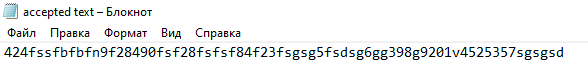
## Передача информации с не более, чем одной ошибкой в слове



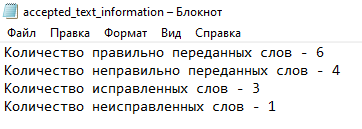
*Рис. 6. Результаты работы на стороне передачи.*



*Рис. 7. Результаты работы на стороне приёма.*

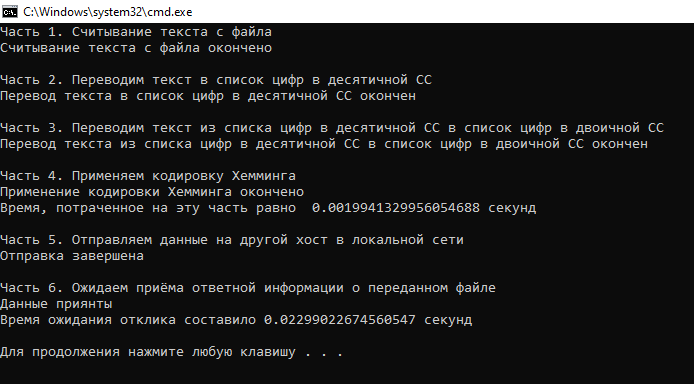


*Рис. 8. Принятый текст.*

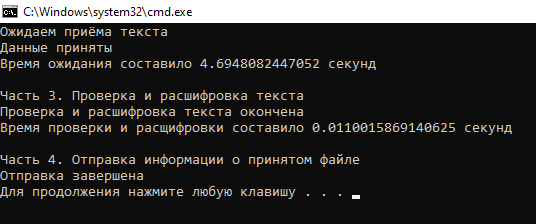


*Рис. 9. Информация о переданном тексте.*

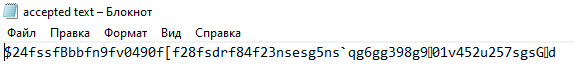
## Передача файла с не более, чем двумя ошибками в слове



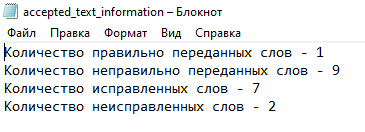
*Рис. 10. Результаты работы на стороне передачи.*



*Рис. 11. Результаты работы на стороне приёма.*

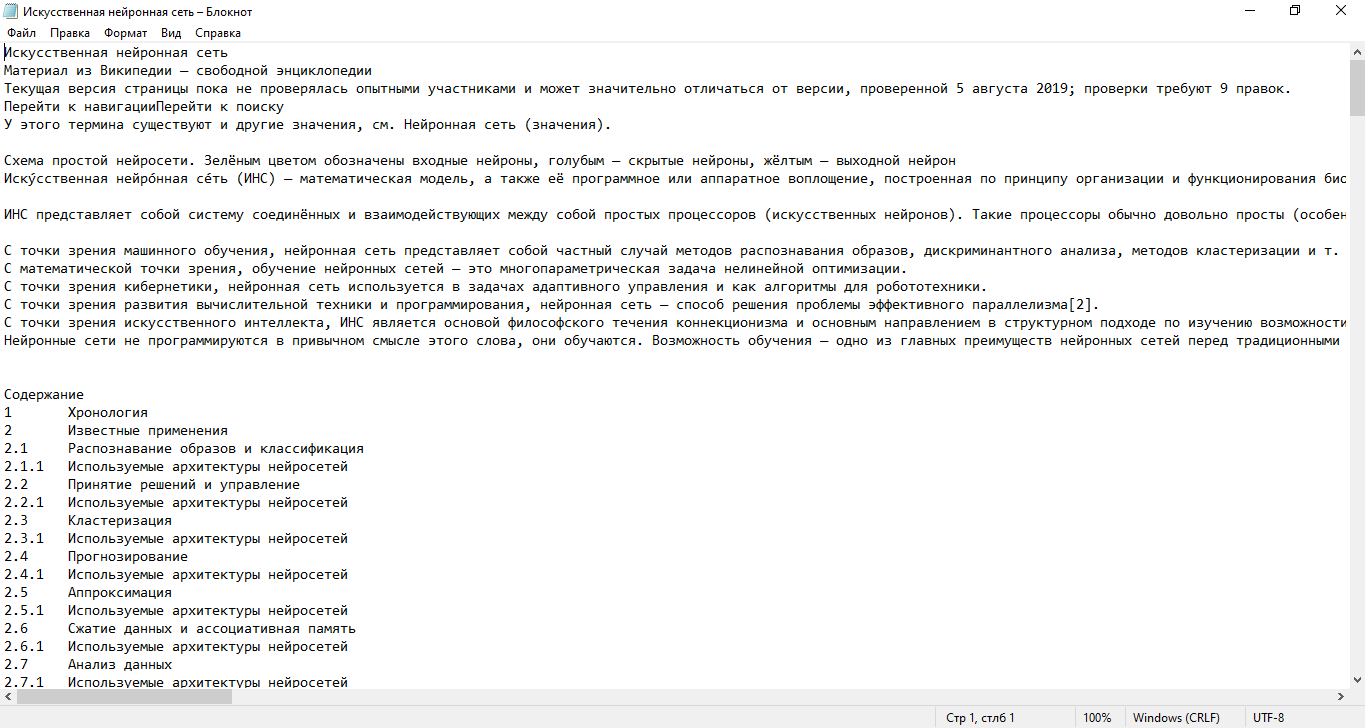


*Рис 12. Принятый текст.*



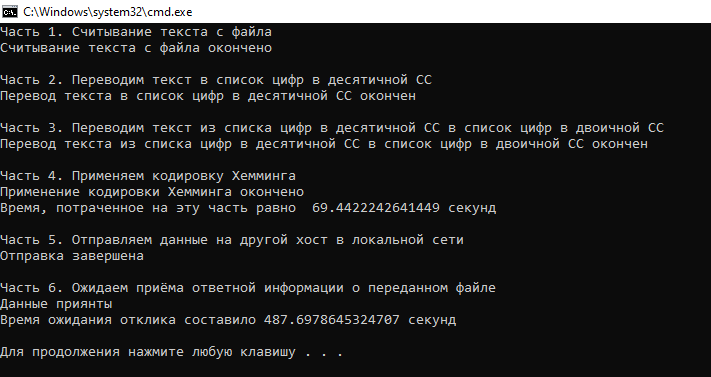
*Рис. 13. Информация о переданном тексте.*

## Статья с Википедии об искусственных нейронных сетях

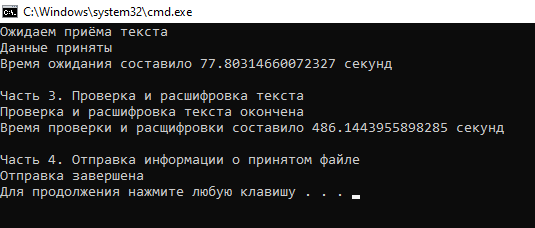


*Рис. 14. Статья с Википедии об искусственных нейронных сетях*

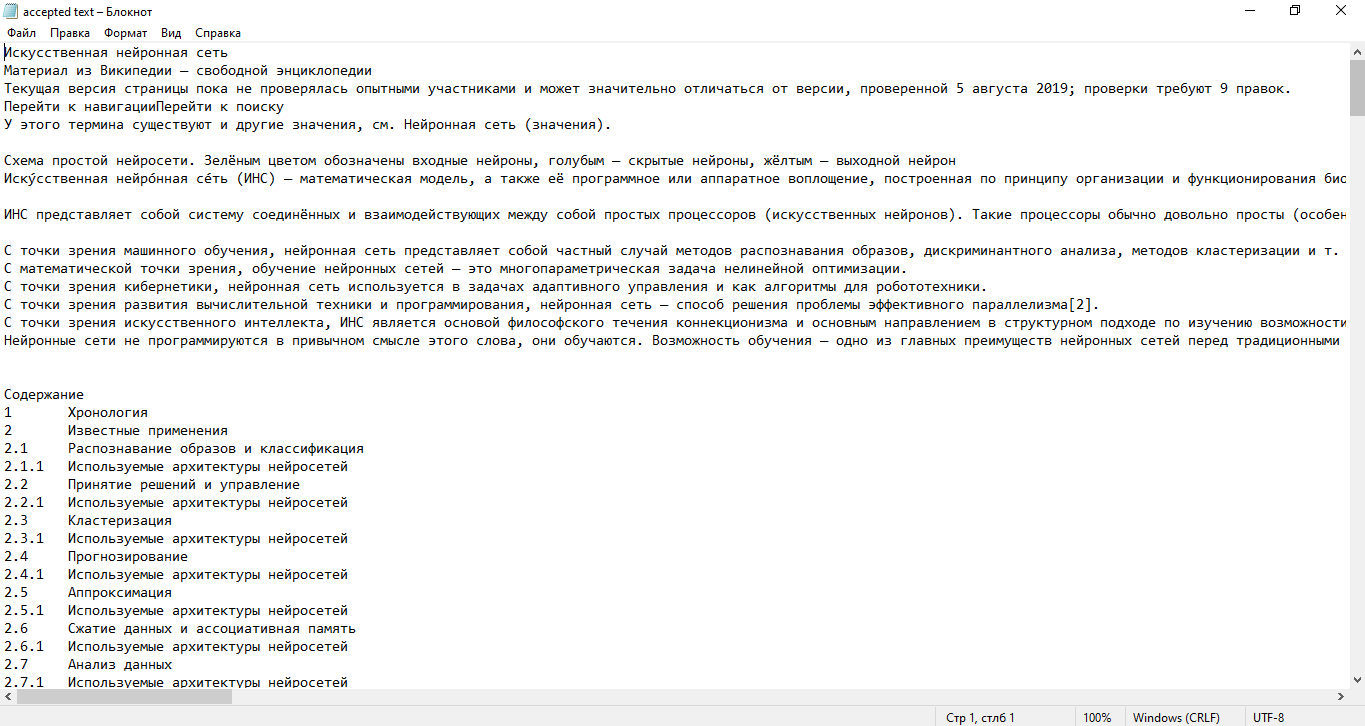
## Передача файла без ошибок



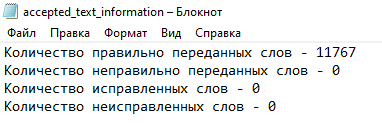
*Рис. 15. Результаты работы на стороне передачи.*



*Рис. 16. Результаты работы на стороне приёма.*

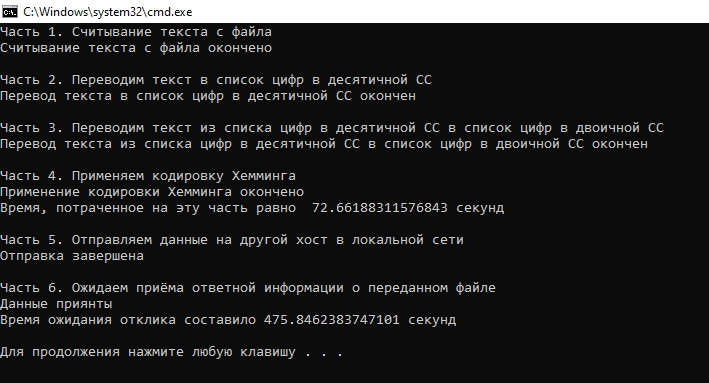


*Рис 17. Принятый текст.*

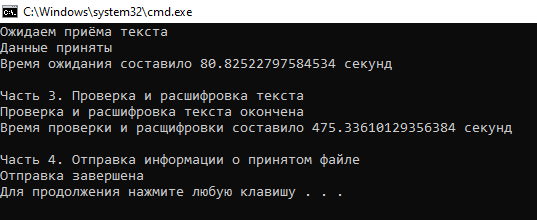


*Рис. 18. Информация о переданном тексте.*

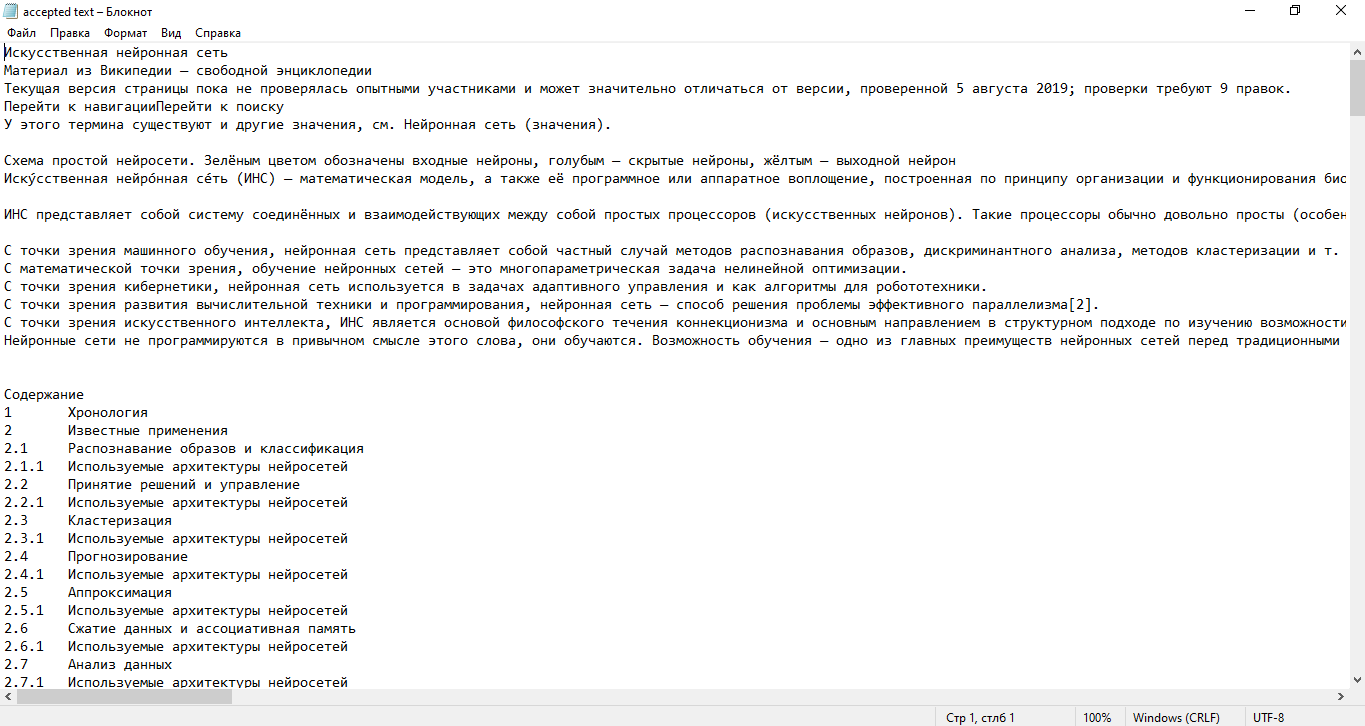
## Передача файла не более, чем с одной ошибкой в слове



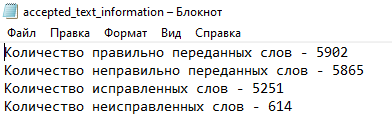
*Рис. 19. Результаты работы на стороне передачи.*



*Рис. 20. Результаты работы на стороне приёма.*

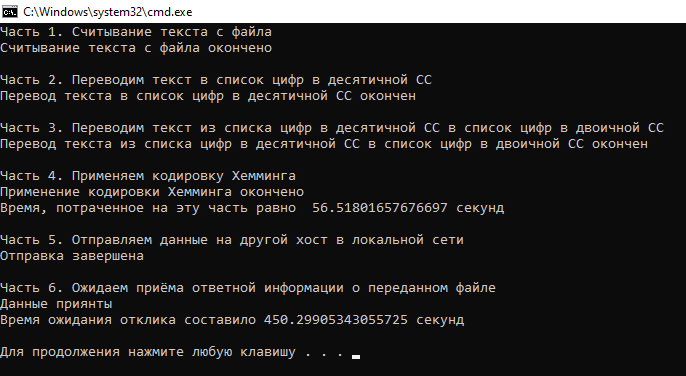


*Рис 21. Принятый текст.*

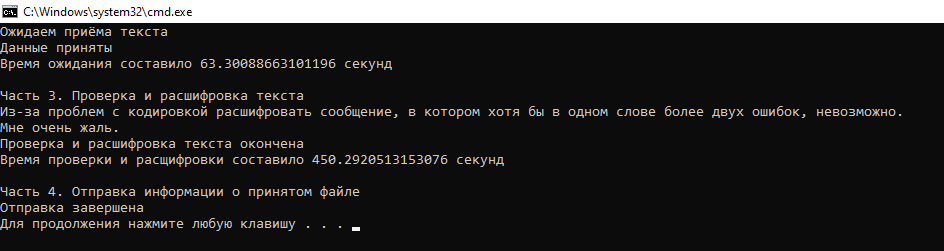


*Рис. 22. Информация о переданном тексте.*

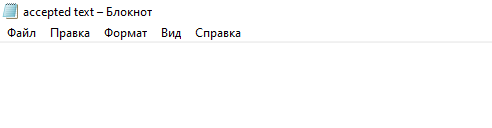
## Передача файла с не более, чем двумя ошибками в слове



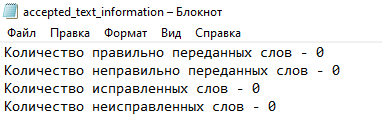
*Рис. 23. Результаты работы на стороне передачи.*



*Рис. 24. Результаты работы на стороне приёма.*



*Рис 25. Принятый текст.*



*Рис. 26. Информация о переданном тексте.*