МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования  
«Белорусский государственный технологический университет»  
  
  
  
  
  
  
Отчёт по лабораторной работе №7

ПЕРЕМЕЖЕНИЕ/ДЕПЕРЕМЕЖЕНИЕ ДАННЫХ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Выполнил: студент 3 курса специальности ИСиТ Калоша И.В.  
Проверила: Ржеутская Н. В.

Минск 2020

**Перемежение/деперемежение данных в информационно-вычислительных системах**

**Цель:** приобретение практических навыков использования методов перемежения/деперемежения двоичных данных в информационных системах.

**Теоретические сведения**

Проанализированные и исследованные нами коды, как и большинство других кодов, которые были разработаны для увеличения надежности каналов передачи и хранения информации, наиболее эффективны, когда возникающие ошибки статистически независимы, т. е. вероятность передачи (хранения в памяти) любого символа одинакова. Однако довольно часто распределение ошибок носит взаимозависимый характер. В таких случаях говорят о группах (или пакетах) ошибок. Такие ошибки характерны и для каналов передачи, и для устройств памяти (главным образом магнитной и полупроводниковой; cм., например, [9−10, 19−22]).

Существуют специальные коды, корректирующие пакетные ошибки, однако на практике чаще используют перемежение/деперемежение совместно с традиционными кодами

Идея перемежения/деперемежения состоит в следующем. Если биты каждого кодового слова Хn передаются не в обычной последовательности, а через интервалы, превышающие ожидаемую длину пакета ошибок (в промежутки между битами одного слова вставляются биты других кодовых слов), то при возникновении такого типа ошибки обратная перемежению операция – деперемежение – разнесет («размажет») группу ошибок по всей совокупности кодовых слов, составляющих данное сообщение.

Длина пакета в нашем случае – это число рядом расположенных ошибочных битов.

Рассмотренный метод блочного перемежения применяется в SM. К числу других используемых на практике относятся следующие методы перемежения/деперемежения: псевдослучайный, S-типа (применяется в турбо-кодировании: CDMA (Codе Division Multiple Access) – стандарт беспроводной связи множественного доступа с кодовым разделением каналов и др.); циклически-сдвиговый; сверточный; случайный; диагональный; многошаговый [5, 23, 24].

Перемежение (перестановка) символов также является основой некоторых классов криптографических методов [1], которые мы будем анализировать с практической точки зрения в другой части курса.

**Листинг кода**

def divide(EnterString,n):

StringWith16LenArray = [EnterString[i:i+n] for i in range(0,len(EnterString),n)]

#print(StringWith16LenArray)

return StringWith16LenArray

def ArrToHemming(Arr):

ArrInHemming = []

for elem in Arr:

ArrInHemming.append(to\_hamming(elem))

return ArrInHemming

def ArrTo16Bit(Arr):

NewArr = []

for elem in Arr:

NewArr.append(to\_16bit(elem))

return NewArr

def Random128String():

Arr = ""

for i in range(128):

Arr += str(random.randint(0,1))

return Arr

def StrEquals(str1, str2):

if err\_col<=5:

print("Строки идентичны")

else:

for i in range(len(str1)):

if str1[i]!=str2[i]:

print("Строки различаются")

return 0

def Peremezheniye(str):

StrInHemming = ArrToHemming(divide(str,16))

print(StrInHemming)

StrInHemming = [list(map(int,list(elem))) for elem in StrInHemming]

PeremezhMatrix = np.array(StrInHemming)

print()

print("Матрица перемежения")

print()

print(PeremezhMatrix)

PeremezhMatrix = np.transpose(PeremezhMatrix)

PeremezhString = ""

for i in PeremezhMatrix:

for j in i:

PeremezhString += np.str(j)

print()

print("Строка после перемежения")

print()

print(PeremezhString)

return PeremezhString

def GenerateError(str,n):

RandomPos = random.randint(0,len(str)-n)

print()

print("Позиция ошибки : ",RandomPos," кооличество ошибок : ", n)

print()

arr = list(str)

#print(arr)

for i in range(RandomPos,RandomPos+n,1):

if arr[i]=='1':

arr[i]='0'

else:

arr[i]='1'

string = ""

for i in arr:

string += i

#print(type(arr[5]))

print()

print("Строка после перемежения с ошибками")

print()

print(string)

return string

def PeremezhBack(str,s2):

PeremezhMatrixWithError =np.transpose(

np.array([list(map(int,list(elem))) for elem in divide(str,8)]))

print()

print("Матрица перемежения с ошибочными битами")

print()

print(PeremezhMatrixWithError)

StrListWithErrInHemming = []

for x in PeremezhMatrixWithError:

string = ""

for i in x:

string += np.str(i)

StrListWithErrInHemming.append(string)

#print(StrListWithErrInHemming)

ReturnedArrWithoutErrors = ArrTo16Bit(StrListWithErrInHemming)

# print(ReturnedArrWithoutErrors)

resString = ""

for i in ReturnedArrWithoutErrors:

resString += i

print()

print("Строка после декодирования")

print()

print(s2)

return resString

**Ход работы**

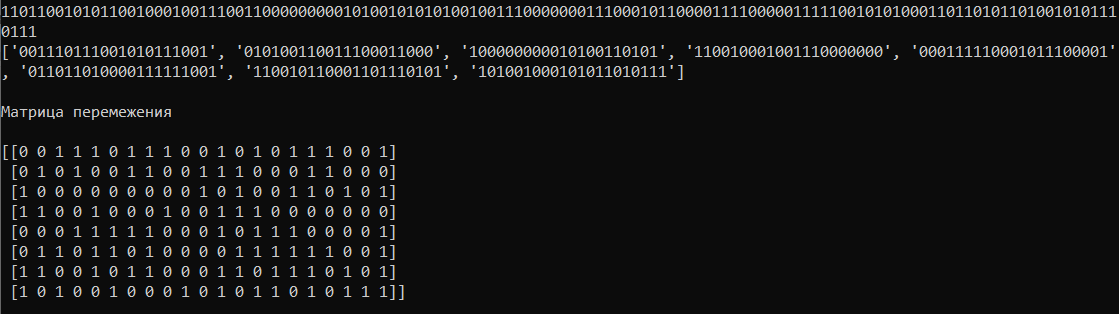
****

Рисунок 1 – начальная строка и матрица перемежения

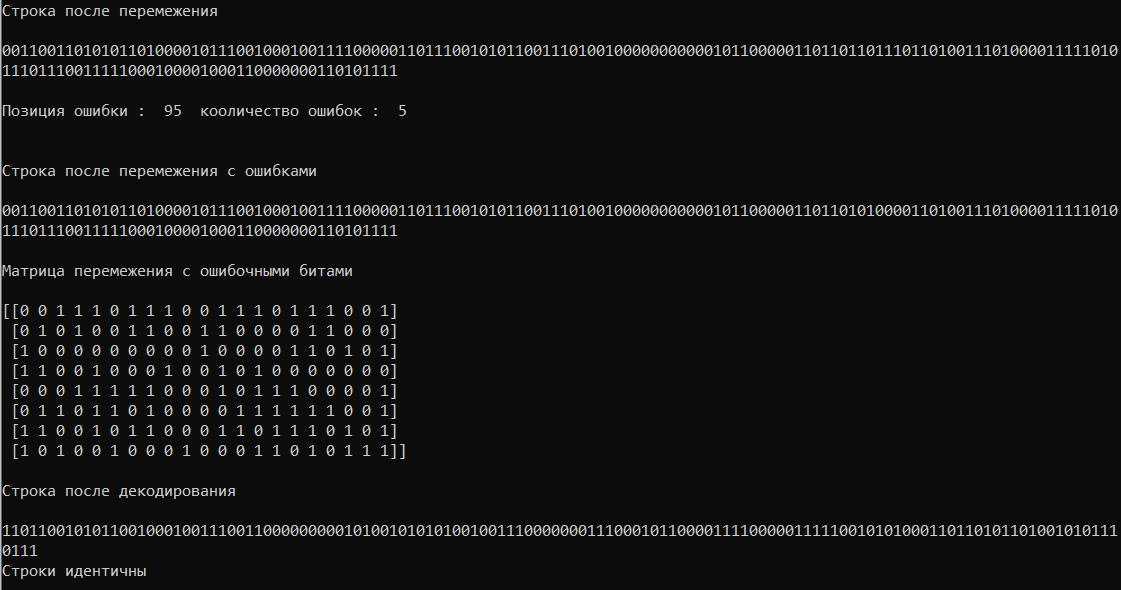
****

Рисунок 2 – Процесс деперемежения и исправления ошибок

**Вывод**: в данной работе было рассмотрено перемежение и деперемежение как варианты кодирования информации. Было установлено, что такой способ должен использоваться в совокупности с другими корректирующими кодами, например кодом Хемминга. Перемежение/деперемежение позволяет разбить исходную строку, на строки нужной длины корректирующего кода, так чтобы можно было исправить как можно больше ошибок.