**Отчет по заданию №4**

**«СИСТЕМАТИЧЕСКИЙ КОД»**

по курсу «Теория информации и кодирования»

студента 3 курса группы ИВТ

Направления подготовки 09.03.01«Информатика и вычислительная техника»

, 2022

**Вариант 4.**

**Техническое задание:**

Источник информации вырабатывает сообщения, содержащие k информационных разрядов. Значения разрядов генерируются в двоичной системе счисления счетчиком случайных чисел. Необходимо:

1. разработать программное обеспечение для передатчика, которое будет строить систематический код с заданной исправляющей способностью;
2. разработать программное обеспечение на приемной стороне, позволяющее корректировать принятую ошибочную кодовую комбинацию;
3. провести комплекс численных экспериментов, в ходе которых на передающей стороне построить систематический код с заданной исправляющей способностью, сгенерировать ошибочный систематический код, на приемной стороне вычислить позицию ошибки и скорректировать принятую кодовую комбинацию.

**Описание работы программы:**

$$2^k \le \frac{2^n}{1+n}$$

Рис. 1 Условие, определяющее длину комбинации

Программа разработана на языке программирования Python 3 и позволяет проводить эксперименты с данными любой длины (не только 58 бит как по заданию в Варианте 4).

В начале программы мы вводим необходимое количество бит, генерируется случайное число в диапазоне чисел, которые можно задать при помощи этого количества бит. Далее данные передаются в модуль Transmitter, который осуществляет преобразование передаваемого сообщения и генерирует для него проверочную матрицу (H, для модуля Приемник).

```
113  k = input("Введите количество информационных бит: ")
114  try:    k = int(k)
115  except: k = 4
116  dat = Random_k_bits(k)
117  pass_block, H = Transmitter(dat, k)
118  Line_With_Errors(pass_block, 1.0)
119  Reciever(pass_block, H)
120  print("\n\nЛабораторная работа №4. Подготовил: Мазлов Иван, ИВТ-201\n")
```

Рис. 2 Инициализация с проверкой ввода количества информационных бит

Внутри модуля Transmitter вызывается модуль подбора полной длины информационной строки, включая проверочный код.

```python
def Transmitter(dat, k):
    info_block = bin2arr(dat, k)
    print("-"*70, "\nПередатчик. ", sep="", end="")
    print("Передается число:\n", dat, " (", "".join(map(str, info_block)), ")", sep="")
    H, H1 = NKpodbor(k, t = 1)
    print("Проверочная матрица, H:")
    [[prn_arr(i), print("")] for i in H]
    add_block = [Get_Sindrome_Code(info_block, i) for i in H1]
    print("Передаваемая информация:")
    prn_arr(info_block)
    print(" ", end="")
    prn_arr(add_block)
    pass_block = info_block + add_block
    # Информ блок + проверочный код, проверочная матрица для
    # приемника (Receiver)
    return pass_block, H
```

Рис. 3 Передатчик

В процедуре Nkpodbor рассчитывается минимальное расстояние – Dmin, по заданной длине информационных бит – k осуществляется подбор необходимого количества проверочных бит – p. Далее составляется массив из всех возможных комбинаций бит для p, список сортируется по убыванию количества бит, равных «1» в варианте. n=k+p.

Строится производящая матрица и отбираются такие варианты из проверочных, чтобы сумма «1» в строке была не меньше, чем Dmin. Полученная матрица транспонируется и каждый из её рядов дополняется соответствующим рядом единичной матрицы размера p*p. Таким образом на выходе функции мы получаем два массива:

- транспонированная дополнительная матрица (H1), которую будет использовать Передатчик для зашифровки передаваемого сообщения
- проверочная матрица (H), которую будет использовать Приёмник для поиска ошибок передачи при получении данных

```python
def NKpodbor(k, t = 1): # t = количество ошибок
    n = k + 1
    while (2**k) > ( (2**n) / (n+1) ): n += 1
    p, Dmin = n-k, 2 * t + 1
    print("Биты:: Информационные k=", k, ", проверочные p=", p, ", всего n=", \
    n, ". Dmin=", Dmin, sep="")
    # Формирование массива возможных вариантов проверочных кодов для дополнительной
    # матрицы ,сортируем по убыванию количества единиц в коде
    Hp = sorted( [bin2arr(j, p) for j in range(2**p)], \
                key = lambda stroka: sum(stroka), reverse=True)
    # Оставляем только те коды, сумма единиц в которых (вместе с единецей
    # в единичной матрице) >= Dmin. Обрезаем лишние коды в конце (до
    # размера k). Транспонируем дополнительную матрицу
    H1 = Transp([S for S in Hp if sum(S) >= Dmin - 1][:k])
    # Создаем единичную матрицу размером p (количество проверочных бит)
    Ed_p = GetEd(p)
    H = [H1[i] + Ed_p[i] for i in range(p)] # передается на приемник для декодирования
    return H, H1
```

Рис. 4 Подбор длины строки бит

Для удобного рассчета суммы (по модулю 2) произведений перемножения строк сделан отдельный модуль.

```
40  # Находим значение бита синдрома ошибки
41  def Get_Sindrome_Code(k_dat, chk_line): #
42      return sum(k_dat[i]*chk_line[i] for i in range(min(len(k_dat), len(chk_line))))&1
```

Рис. 5 Вычисление бита синдрома ошибки

Полученное полное информационное сообщение (n=k+p) обрабатывается модулем «создания ошибки». Модуль выбирает один из битов сообщения случайным образом и, в соответствии коэфициентом вероятности инвертирует его значение. Таким образом мы симулирем происхождение ошибки при передаче в помехонезащищенной линии. Кроме того, для проверки модуль возвращает список бит (в тестах – только один бит), в которых была допущена ошибка.

```
70  def Line_With_Errors(pass_block, Er_prob = 0.2, possible_N_of_err_bits = 1):
71      # длина блока, и индексы битов с ошибками
72      block_len, error_bit_list = len(pass_block), []
73      # коррекция, если вероятность ошибки > 100%
74      if Er_prob > 1.0: Er_prob = 1.0
75      print("\n", "-"*70, "\nГенерации ошибки в линии с помехами (", Er_prob*100,\
76      "%). Получено для передачи:", sep="")
77      prn_arr(pass_block)
78      # коррекция, если ошибочных бит больше, чем передаваемых
79      if possible_N_of_err_bits > block_len: possible_N_of_err_bits = block_len
80      # Формируем список индексов битов, где может по вероятности Er_prob произойти
81      # ошибка (т.е. бит инвертируется)
82      err_bits_idxs = random.sample(list(range(block_len)), possible_N_of_err_bits)
83      # Проходим по списку выбранных для ошибки бит, применяем вероятность,
84      # инвертируем если надо, добавляем в итоговый список ошибочно переданных бит
85      for i in err_bits_idxs:
86          if random.uniform(0.0, 1.0) <= Er_prob:
87              pass_block[i] = 0 if pass_block[i] else 1
88              error_bit_list.append(i)
89      print("\nИндексы ошибочно переданных бит: ", error_bit_list)
90      print("Измененные данные:")
91      prn_arr(pass_block)
92      return sorted(error_bit_list)   # индексы бит, в которых была допущена ошибка
```

Рис. 6 Исправление ошибок в строке

Измененное значение (с ошибкой) передается в модуль Приемника. Кроме того, вторым аргументом туда передается проверочная матрица. Модуль делает проверку полученного сообщения и формирует массив синдрома ошибки. Если в этом массиве – все нули, то передача прошла без ошибки, иначе осуществляется поиск столбца в

проверочной матрице, который совпадает со значениями в массиве синдрома ошибки. Индекс этого столбца и будет указывать на индекс бита в информационном сообщении, где произошла ошибка в момент передачи данных. Модуль корректирует эту ошибку, инвертируя бит с этим индексом.

```
94  def Reciever(pass_block, H):
95      print("\n", "-"*70, "\nПриёмник. Получено из линии передачи:", sep="")
96      prn_arr(pass_block)
97
98      S = [Get_Sindrome_Code(pass_block, h) for h in H]
99      if not sum(S):
100         print("\nПередача прошла без ошибок. Код не изменен.")
101     else:
102         TH=Transp(H)
103         err_idx = -1
104         for idx in range(len(TH)):
105             if TH[idx]==S:
106                 err_idx=idx
107                 print("\nПроизошла ошибка в бите с индексом:", err_idx)
108                 pass_block[err_idx]= 0 if pass_block[err_idx] else 1
109                 print("Исправленное значение:")
110                 prn_arr(pass_block)
111                 break
```

Рис. 7 Приемник

Модуль формирования единичной матрицы заданного размера задан как лямбда-выражение.

```
16  # Формирование единичной (квадратной) матрицы заданного размера
17  GetEd  = lambda k: [[1 if i==j else 0 for i in range(k)] for j in range(k)]
```

Рис. 8 Построение единичной матрицы

Модуль транспонирования матрицы. Ещё одно лямбда-выражение.

```
19  # Транспонирование матрицы
20  Transp = lambda ar: [[ar[i][j]for i in range(len(ar))] for j in range(len(ar[0]))]
```

Рис. 9 Транспонирование матрицы

Было проведено 6 тестов с размером передаваемой информации 58 бит и вероятностью возникновения единичной ошибки 100%. Все тесты прошли успешно и ошибка передачи была выявлена и исправлена Приемником.

```
Введите количество информационных бит: 58
------------------------------------------------------------------------
Передатчик. Передается число:
17628618261365083 (0000111110101000010010001010001001010110011000000101011011)
Биты:: Информационные k=58, проверочные p=7, всего n=65. Dmin=3
Проверочная матрица, H:
10111111000000111111111111111100000000000000001111111111111111000000
11011111011111000001111111111000001111111111000000000001111010000
11101111101111011110000111111011110000111111000011111100000010000
11110111110111101111000011111101111000011111100001111110000100010
11111011111011110111011000111011101110001110111000111000100010100
11111101111101111011101100111011101101100110110110010110000010
11111110111110111101110101011101110101011011010101010000000010
11111110111110111101110100111101110100111011010011010000001
Передаваемая информация:
0000111110101000010010001010001001010110011000000101011011 0100000
------------------------------------------------------------------------
Генерации ошибки в линии с помехами (100.0%). Получено для передачи:
0000111110101000010010001010001001010110011000000101011011010000
Индексы ошибочно переданных бит:  [55]
Измененные данные:
0000111110101000010010001010001001010110011000000101011111010000
------------------------------------------------------------------------
Приёмник. Получено из линии передачи:
0000111110101000010010001010001001010110011000000101011111010000
Произошла ошибка в бите с индексом: 55
Исправленное значение:
0000111110101000010010001010001001010110011000000101011011010000

Лабораторная работа №4. Подготовил: Мазлов Иван, ИВТ-201
```

Рис. 10 Тест 1

```
Введите количество информационных бит: 58
------------------------------------------------------------------------
Передатчик. Передается число:
27450081709831937 (0001100001100001011011010000110001010111101011011100000001)
Биты:: Информационные k=58, проверочные p=7, всего n=65. Dmin=3
Проверочная матрица, H:
10111111000000111111111111111100000000000000001111111111111111000000
11011111011111000001111111111000001111111111000000000001111010000
11101111101111011110000111111011110000111111000011111100000010000
11110111110111101111000011111101111000011111100001111110000100010
11111011111011110111011000111011101110001110111000111000100010100
11111101111101111011101100111011101101100110110110010110000010
11111110111110111101110101011101110101011011010101010000000010
11111110111110111101110100111101110100111011010011010000001
Передаваемая информация:
0001100001100001011011010000110001010111101011011100000001 1101110
------------------------------------------------------------------------
Генерации ошибки в линии с помехами (100.0%). Получено для передачи:
0001100001100001011011010000110001010111101011011100000011101110
Индексы ошибочно переданных бит:  [60]
Измененные данные:
0001100001100001011011010000110001010111101011011100000011111110
------------------------------------------------------------------------
Приёмник. Получено из линии передачи:
0001100001100001011011010000110001010111101011011100000011111110
Произошла ошибка в бите с индексом: 60
Исправленное значение:
0001100001100001011011010000110001010111101011011100000011101110

Лабораторная работа №4. Подготовил: Мазлов Иван, ИВТ-201
```

Рис. 11 Тест 2

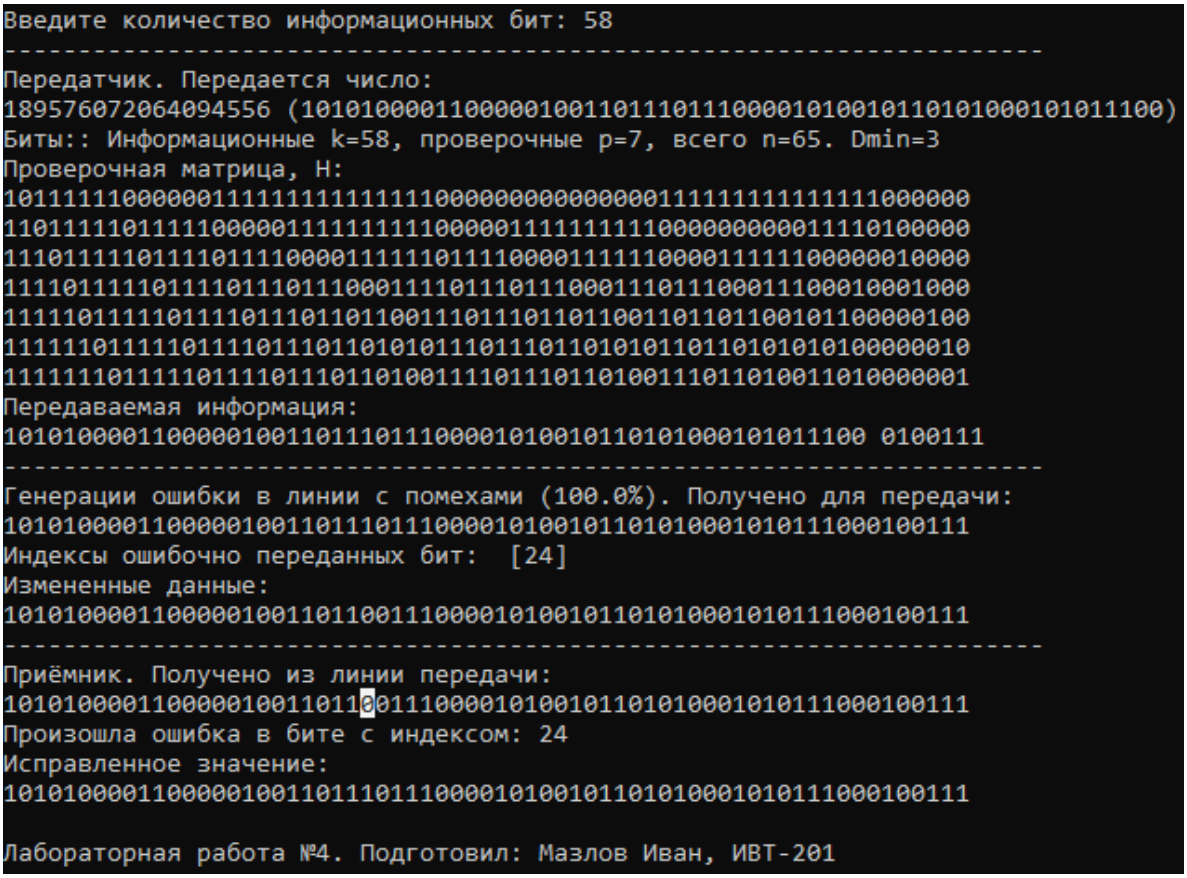


Рис. 12 Тест 3

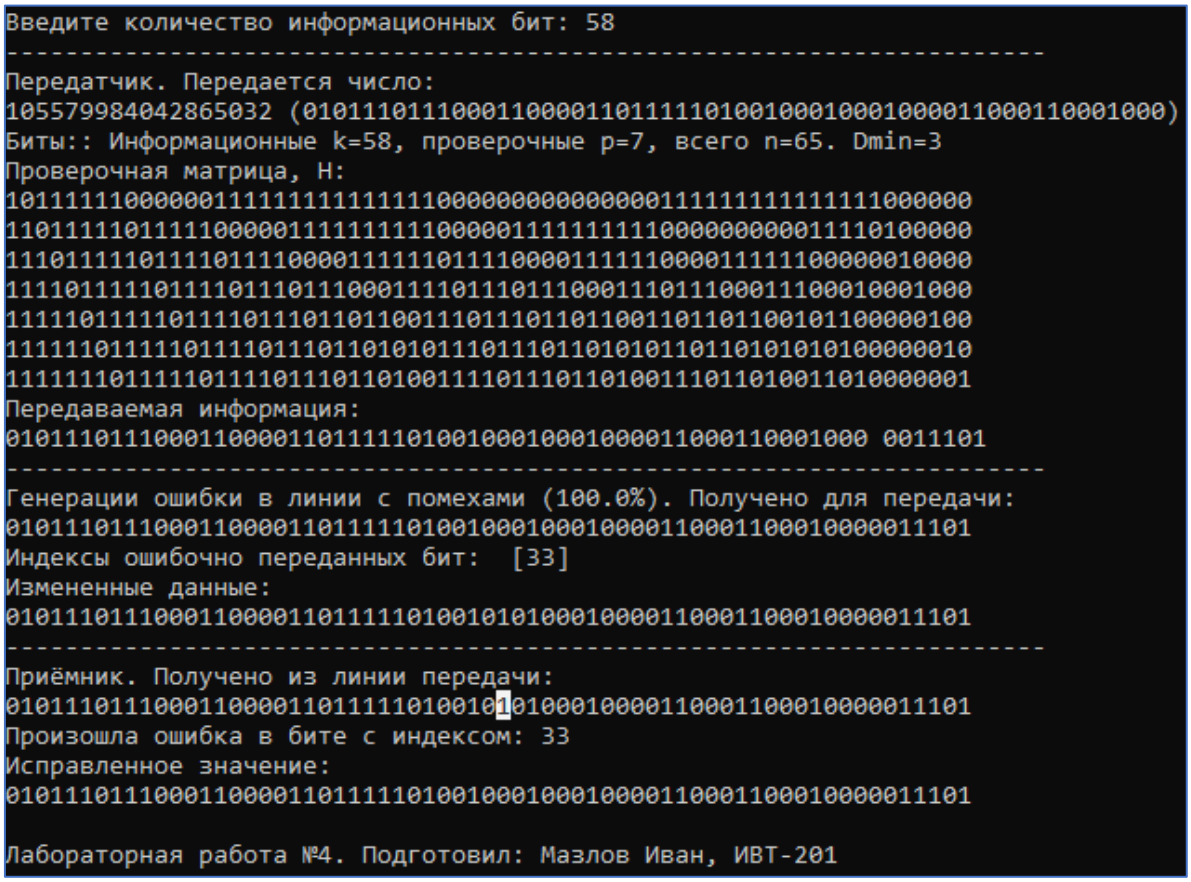


Рис. 13 Тест 4

```
Введите количество информационных бит: 58
------------------------------------------------------------------------
Передатчик. Передается число:
279780114023689089 (1111100001111110101000100000000111100110011010101110000001)
Биты:: Информационные k=58, проверочные p=7, всего n=65. Dmin=3
Проверочная матрица, H:
10111111000000111111111111111000000000000000011111111111111111000000
11011111011111000001111111111000001111111111000000000001111010000
11101111101111011110000111111101111000011111100001111110000010000
11110111110111101110111000111101110111000111011100011100010001000
11111011111011110111011001110111011011001101101100101100000100
11111101111101111011101101010111011101101010110110101010100000010
11111110111110111101110110100111101110110100111011010011010000001
Передаваемая информация:
11111000011111101010001000000001111001100110101011100000010010001
------------------------------------------------------------------------
Генерации ошибки в линии с помехами (100.0%). Получено для передачи:
11111000011111101010001000000001111001100110101011100000010010001
Индексы ошибочно переданных бит:  [42]
Измененные данные:
11111000011111101010001000000001111001100100101011100000010010001
------------------------------------------------------------------------
Приёмник. Получено из линии передачи:
11111000011111101010001000000001111001100100101011100000010010001
Произошла ошибка в бите с индексом: 42
Исправленное значение:
11111000011111101010001000000001111001100110101011100000010010001

Лабораторная работа №4. Подготовил: Мазлов Иван, ИВТ-201
```

Рис. 14 Тест 5

```
Введите количество информационных бит: 58
------------------------------------------------------------------------
Передатчик. Передается число:
192344985719299070 (1010101011010110001011111011010100001000000101101111111110)
Биты:: Информационные k=58, проверочные p=7, всего n=65. Dmin=3
Проверочная матрица, H:
10111111000000111111111111111000000000000000011111111111111111000000
11011111011111000001111111111000001111111111000000000001111010000
11101111101111011110000111111101111000011111100001111110000010000
11110111110111101110111000111101110111000111011100011100010001000
11111011111011110111011001110111011011001101101100101100000100
11111101111101111011101101010111011101101010110110101010100000010
11111110111110111101110110100111101110110100111011010011010000001
Передаваемая информация:
10101010110101100010111110110101000010000001011011111111100001111
------------------------------------------------------------------------
Генерации ошибки в линии с помехами (100.0%). Получено для передачи:
10101010110101100010111110110101000010000001011011111111100001111
Индексы ошибочно переданных бит:  [32]
Измененные данные:
10101010110101100010111110110101100010000001011011111111100001111
------------------------------------------------------------------------
Приёмник. Получено из линии передачи:
10101010110101100010111110110101100010000001011011111111100001111
Произошла ошибка в бите с индексом: 32
Исправленное значение:
10101010110101100010111110110101000010000001011011111111100001111

Лабораторная работа №4. Подготовил: Мазлов Иван, ИВТ-201
```

Рис. 15 Тест 6

**Вывод:**

В ходе выполнения лабораторной работы мы познакомились на практике с алгоритмом создания и применения систематического кода для передачи информации по помехонеустойчивым линиям.

На языке программирования Python 3 был выполнен программный модуль, рассчитывающий производящую и проверочную матрицы, добавляющий к информационному сообщению проверочный код нужной длины. Программа имитировала ошибку в передаче одного из битов сообщения, а приемная часть программы обнаруживала и исправляла её, используя проверочную матрицу.

Было проведено 6 тестов при исходном количестве информационных бит – 58. Все тесты прошли успешно, что позволяет говорить, об эффективности данного алгоритма защиты от помех при передаче сообщений. При небольшом увеличении длины передаваемого сообщения существенно повышается помехоустойчивость передачи. Для поиска и коррекции ошибки на приемнике должна быть проверочная матрица, сгенерированная Передатчиком. Такая матрица генерируется один раз и далее все сообщения по линии передаются с добавленными проверочными кодами, которые позволяют Приемнику проверять правильность передачи.

Чем больше длина информационного сообщения в битах, тем меньший процент дополнительного трафика (в сравнении с длиной сообщения) приходится передавать дополнительно в виде проверочного кода. При малых длинах сообщений такая технология сильно теряет свой смысл, так как количество добавляемых бит проверочного кода сопоставимо по длине с передаваемым информационным сообщением и проще передать сообщение повторно.

**Приложение1.**

**Полный листинг программы**

```
import random
# Побитный перевод полученного двоичного числа в массив (бинарную матрицу)
def bin2arr(i, k): #  11 3 => [0,1,1]
    # Получает число i и возвращает list длиной k-элементов, где каждый элемент
    # 0\1 в зависимости от бита в числе
    i = bin(i)[2:]
    return [int(j) for j in ("0"*(k-len(i)) + i) ]                  # fastest
    #return [int(j) for j in (i[:k] if L>=k else (("0"*(k-L)) + i)) ] # fast (проверка)
    #return [1 if (i&(2**b)) else 0  for b in range(k-1,-1,-1)]       # less slow
    #return [int(bool(i&(2**b))) for b in range(k)][::-1]            # slow

def prn_arr(arr):
    #print("[", end="")
    [print(i, sep="", end="") for i in arr]
```

```
    #print("]")
# Формирование единичной (квадратной) матрицы заданного размера
GetEd  = lambda k: [[1 if i==j else 0 for i in range(k)] for j in range(k)]

# Транспонирование матрицы
Transp = lambda ar: [[ar[i][j]for i in range(len(ar))] for j in range(len(ar[0]))]

def NKpodbor(k, t = 1): # t = количество ошибок
    n = k + 1
    while (2**k) > ( (2**n) / (n+1) ): n += 1
    p, Dmin = n-k, 2 * t + 1
    print("Биты:: Информационные k=", k, ", проверочные p=", p, ", всего n=", \
    n, ". Dmin=", Dmin, sep="")
    # Формирование массива возможных вариантов проверочных кодов для
дополнительной
    # матрицы ,сортируем по убыванию количества единиц в коде
    Hp = sorted( [bin2arr(j, p) for j in range(2**p)], \
              key = lambda stroka: sum(stroka), reverse=True)
    # Оставляем только те коды, сумма единиц в которых (вместе с единецей
    # в единичной матрице) >= Dmin. Обрезаем лишние коды в конце (до
    # размера k). Транспонируем дополнительную матрицу
    H1 = Transp([S for S in Hp if sum(S) >= Dmin - 1][:k])
    # Создаем единичную матрицу размером p (количество проверочных бит)
    Ed_p = GetEd(p)
    H = [H1[i] + Ed_p[i] for i in range(p)] # передается на приемник для декодирования
    return H, H1
# Находим значение бита синдрома ошибки
def Get_Sindrome_Code(k_dat, chk_line): #
    return sum(k_dat[i]*chk_line[i] for i in range(min(len(k_dat), len(chk_line))))&1

def Random_k_bits(k): # ==> int
    return random.randint(0, (2**k) - 1)

# Генерирует
def Transmitter(dat, k):
    info_block = bin2arr(dat, k)
    print("-"*70, "\nПередатчик. ", sep="", end="")
    print("Передается число:\n", dat, " (", "".join(map(str, info_block)), ")", sep="")
    H, H1 = NKpodbor(k, t = 1)
    print("Проверочная матрица, H:")
    [[prn_arr(i), print("")] for i in H]
    add_block = [Get_Sindrome_Code(info_block, i) for i in H1]
    print("Передаваемая информация:")
    prn_arr(info_block)
```

```
    print(" ", end="")
    prn_arr(add_block)
    pass_block = info_block + add_block
    # Информ блок + проверочный код, проверочная матрица для
    # приемника (Receiver)
    return pass_block, H

# Линия передач с помехами
#   pass_block - передаваемые данные (информ биты + проверочные - list 0\1)
#   possible_N_of_err_bits - количество бит, в которых может произойти ошибка
#   Er_prob - вероятность наступления ошибки в выбранном
#
def Line_With_Errors(pass_block, Er_prob = 0.2, possible_N_of_err_bits = 1):
    # длина блока, и индексы битов с ошибками
    block_len, error_bit_list = len(pass_block), []
    # коррекция, если вероятность ошибки > 100%
    if Er_prob > 1.0: Er_prob = 1.0
    print("\n", "-"*70, "\nГенерации ошибки в линии с помехами (", Er_prob*100,\
    "%). Получено для передачи:", sep="")
    prn_arr(pass_block)
    # коррекция, если ошибочных бит больше, чем передаваемых
    if possible_N_of_err_bits > block_len: possible_N_of_err_bits = block_len
    # Формируем список индексов битов, где может по вероятности Er_prob
произойти
    # ошибка (т.е. бит инвертируется)
    err_bits_idxs = random.sample(list(range(block_len)), possible_N_of_err_bits)
    # Проходим по списку выбранных для ошибки бит, применяем вероятность,
    # инвертируем если надо, добавляем в итоговый список ошибочно переданных
бит
    for i in err_bits_idxs:
        if random.uniform(0.0, 1.0) <= Er_prob:
            pass_block[i] = 0 if pass_block[i] else 1
            error_bit_list.append(i)
    print("\nИндексы ошибочно переданных бит: ", error_bit_list)
    print("Измененные данные:")
    prn_arr(pass_block)
    return sorted(error_bit_list)   # индексы бит, в которых была допущена ошибка

def Reciever(pass_block, H):
    print("\n", "-"*70, "\nПриёмник. Получено из линии передачи:", sep="")
    prn_arr(pass_block)

    S = [Get_Sindrome_Code(pass_block, h) for h in H]
    if not sum(S):
```

```
        print("\nПередача прошла без ошибок. Код не изменен.")
    else:
        TH=Transp(H)
        err_idx = -1
        for idx in range(len(TH)):
            if TH[idx]==S:
                err_idx=idx
                print("\nПроизошла ошибка в бите с индексом:", err_idx)
                pass_block[err_idx]= 0 if pass_block[err_idx] else 1
                print("Исправленное значение:")
                prn_arr(pass_block)
                break

k = input("Введите количество информационных бит: ")
try:    k = int(k)
except: k = 4
dat = Random_k_bits(k)
pass_block, H = Transmitter(dat, k)
Line_With_Errors(pass_block, 1.0)
Reciever(pass_block, H)
print("\n\nЛабораторная работа №4. Подготовил: xxx, ИВТ\n")
```