МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 НА ТЕМУ:**

**Избыточное кодирование данных в информационных системах. Код Хемминга**

Ф.И.О.

Божко Денис Владимирович

Преподаватель

асс. Берников Владислав Олегович

Минск 2021

**Цель:** приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

**Теоретические сведения**

Надежность системы – характеристика способности программного, аппаратного, аппаратно-программного средства выполнить при определенных условиях требуемые функции в течение определенного периода времени.

Достоверность работы системы (устройства) – свойство, характеризующее истинность конечного (выходного) результата работы (выполнения программы), определяемое способностью средств контроля фиксировать правильность или ошибочность работы.

Ошибка устройства – неправильное значение сигнала (бита – в цифровом устройстве) на внешних выходах устройства или отдельного его узла, вызванное технической неисправностью, или воздействующими на него помехами (преднамеренными либо непреднамеренными), или иным способом.

Ошибка программы – проявляется в не соответствующем реальному (требуемому) промежуточному или конечному значению (результату) вследствие неправильно запрограммированного алгоритма или неправильно составленной программы.

Как следует из вышеприведенного определения, надежность есть внутреннее свойство объекта, заложенное в него при изготовлении и проявляющееся во время эксплуатации. Вторая особенность надежности состоит в том, сто она проявляется во времени. И третья особенность – надежность проявляется по-разному при различных условиях эксплуатации и различных режимах применения объекта (информационной системы в целом, отдельного ее бока, канала передачи сообщения, оперативной или внешней памяти компьютера).

Надежность является комплексным свойством, включающим в себя единичные свойства: безотказность, ремонтопригодность, сохраняемость, долговечность.

Безотказность – это свойство технического объекта непрерывно сохранять работоспособное состояние в течение некоторого времени (или наработки). Наработка, как правило, измеряется в единицах времени.

Ремонтопригодность – это свойство технического объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания, ремонта (или с помощью дополнительны, избыточных технических средств, функционирующих параллельно с объектом). Большинство современных цифровых систем и устройств (в том числе, компьютеры и компьютерные системы, отдельные блоки и модули компьютеров - полупроводниковая, магнитная или оптическая память) содержат специальные средства, призванные автоматически восстанавливать работоспособность этих объектов при нарушении нормального функционирования.

**Ход работы**

1. На основе информационного сообщения, представленного символами русского/английского алфавитов, служебными символами и цифрами, содержащегося в некотором текстовом файле (согласовать с преподавателем), сформировать информационное сообщение в двоичном виде; длина сообщения в бинарном виде должна быть не менее 16 символов.



Рисунок 1 – Входные данные

1. Для полученного информационного слова построить проверочную матрицу Хемминга (значение минимального кодового расстояния согласовать с преподавателем).

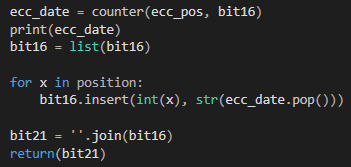


Рисунок 2 – Алгоритм построения проверочной матрицы

1. Используя построенную матрицу вычислить избыточные символы (слово Xr).



Рисунок 3 – Полученные слова без ошибок

**Листинг кода**

|  |
| --- |
| ecc\_pos =(  [0,1,3,4,6,8,10,11,13,15],  [0,2,3,5,6,9,10,12,13],  [1,2,3,7,8,9,10,14,15],  [4,5,6,7,8,9,10],  [11,12,13,14,15]  )  position16\_5 = (11,4,1,0,0)  ecc\_pos21 = (  [2,4,6,8,10,12,14,16,18,20],  [2,5,6,9,10,13,14,17,18],  [4,5,6,12,13,14,19,20],  [8,9,10,11,12,13,14],  [16,17,18,19,20]  )  position21 = (0,1,3,7,15)  def is\_bool(variable):  new\_bool = [x for x in variable if x in ["0", "1"]]  return len(variable) == len(new\_bool)    def counter(position, variable):  list\_bool = list(str(variable))  ecc\_value = []  for item\_on\_place in position:  y = 0  for x in item\_on\_place:  if list\_bool[x] == "1":  y += 1  if y % 2 == 0:  ecc\_value.append(0)  else:  ecc\_value.append(1)  return ecc\_value    def to\_hamming(value, position = position16\_5):  bit16 = str(value)  if len(bit16) != 16:  raise ValueError("Error: количество разрядов не равно 16")  elif is\_bool(bit16) != True:  raise ValueError("Число не двоичное")  ecc\_date = counter(ecc\_pos, bit16)  print(ecc\_date)  bit16 = list(bit16)  for x in position:  bit16.insert(int(x), str(ecc\_date.pop()))  bit21 = ''.join(bit16)  return(bit21)  def checker(variable, ecc21\_date):  list\_21 = list(str(variable))  ecc\_old = []  for x in position21:  a = list\_21[x]  ecc\_old.append(int(a))  error\_list = []  for i, x in enumerate(ecc\_old):  if ecc21\_date[i] != x:  error\_list.append(i)  return error\_list  def to\_16bit(variable):  bit21ecc = str(variable)    if len(bit21ecc) != 21:  raise ValueError("Error: число разрядов не равно 21")  elif is\_bool(bit21ecc) != True:  raise ValueError("Число не двоичное")  ecc21\_date = counter(ecc\_pos21, bit21ecc)  err\_list = checker(bit21ecc, ecc21\_date)  bit21ecc = list(bit21ecc)  if err\_list:  err\_pos = 0  for x in err\_list:  err\_pos += (position21[x] + 1)  err\_pos -= 1  if bit21ecc[err\_pos] == 0:  del bit21ecc[err\_pos]  bit21ecc.insert(err\_pos, '1')  else:  del bit21ecc[err\_pos]  bit21ecc.insert(err\_pos, '0')    for x in [15, 7, 3, 1, 0]:  del bit21ecc[x]  bit16\_len = ''.join(bit21ecc)  return bit16\_len  def test():  num1 = "1010111100011011"  num2 = "1011111011111010"  num3 = "0100010000111101"  print("\nCoder test")  ecc1 = to\_hamming(num1)  print(ecc1)  ecc2 = to\_hamming(num3)  print(ecc2)  ecc3 = to\_hamming(num2)  print(ecc3)  #ecc1 = "101011010010110100001"  #ecc2 = "101001101110111111110"  #ecc3 = "110110000100001011101"  print("\nDecoder test")  g = to\_16bit(ecc1)  print(g)  g = to\_16bit(ecc2)  print(g)  g = to\_16bit(ecc3)  print(g)  test() |

**Вывод**: в данной работе был рассмотрен проверочный код Хэмминга. Было выяснено, что данный вид кодирования позволяет представлять информацию таким образом, что даже при её ошибочной передаче, удастся восстановить одну ошибку. К сожалению, две ошибки исправить уже невозможно. Однако, можно разбить данные на меньшие промежутки, тогда можно будет исправить больше ошибок, но передача будет медленнее.