Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное

учреждения высшего образования

Университет науки и технологий МИСИС

«МИСиС»

Отчет по практической работе №2 на тему:

«Метод Хэмминга»

по дисциплине:

«Методы и средства защиты компьютерной информации»

Направление подготовки: 09.03.03

Семестр 5

|  |  |
| --- | --- |
| Выполнил:    Бобров Михаил Алексеевич  БПИ-20-2  (группа)  11.12.2022  (дата)  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Проверил:  доцент, к. т. н.  Костин Виталий Николаевич  \_\_\_\_\_\_\_\_\_\_  (дата)  \_\_\_\_\_\_\_\_\_\_  (оценка)  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2022

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc121603503)

[Определение входных и выходных данных 4](#_Toc121603504)

[1.1 Входные данные 4](#_Toc121603505)

[1.2 Выходные данные 4](#_Toc121603506)

[Алгоритм ХЭММИНГА И ВОССТАНОВЛЕНИя данных 5](#_Toc121603507)

[2.1 Алгоритм Хэмминга 5](#_Toc121603508)

[2.2 Алгоритм восстановления 5](#_Toc121603509)

[Блок-схемы алгоритма 7](#_Toc121603510)

[3.1 Блок-схема всего алгоритма 7](#_Toc121603511)

[3.2 Алгоритм Кодирования 8](#_Toc121603512)

[Листинг программы 9](#_Toc121603513)

[Тестирование программы 11](#_Toc121603514)

ВВЕДЕНИЕ

Отчёт составлен по результатам выполнения практической работы на тему «Метод Хэмминга» по дисциплине «Методы и средства защиты компьютерной информации».

Целью работы является кодирование данных с контрольными битами методом Хэмминга, восстановление искаженных данных, и его реализация на языке программирования Python.

Достижение указанной цели предполагает решение следующих задач:

1. Изучение метода Хэмминга;
2. Рассмотрение возможности его применения для данного случая;
3. Составление блок-схем, описывающих работу алгоритма;
4. Проверка корректности работы алгоритма путем его тестирования.

Определение входных и выходных данных

1.1 Входные данные

Требования: ввод одного двоичного числа длиной 8 бит, который на входе читается как строчный формат. Далее ввод разряда в целочисленном, в которой допустить ошибку для проверки работы алгоритма.

1.2 Выходные данные

Вывод выходных данных реализован посредством консоли.

Выходными данными, согласно требованиям, являются восстановленное число и значение корректора. Восстановленные данные выводятся строчным форматом. Корректор выводится целочисленным форматом.

Алгоритм ХЭММИНГА И ВОССТАНОВЛЕНИя данных

2.1 Алгоритм Хэмминга

Алгоритм Хэмминга кодирует данные следующим образом: контрольный бит с номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N. Номера контрольных битов равняются степеням двойки. В данной реализации программы используется 4 служебных бита на местах 1, 2, 4 и 8 на 8 информационных битов, что в итоге дает длину в 12 битов.

Изображение выглядит как седзи, кроссворд

Автоматически созданное описание

Рисунок 2.1 – пример работы кодирования алгоритмом Хэмминга, где красными нулями крестиками обозначены созданные контрольные биты до работы алгоритма и крестиками обозначены соответствующие им подконтрольные биты.

2.2 Алгоритм восстановления

Восстановление методом Хэмминга начинается с получения данных и нового отдельного вычисления контрольных битов при помощи информационных. Далее происходит сравнение входных и новых контрольных битов. Если новые контрольные биты отличаются от входных, то это означает, что информационные биты искажены. Данный алгоритм может восстановить только один искаженный разряд. Вычисление позиции искаженного разряда происходит сложением разрядов контрольных битов. Восстановление данных происходит инвертированием бита.

Блок-схемы алгоритма

3.1 Блок-схема всего алгоритма

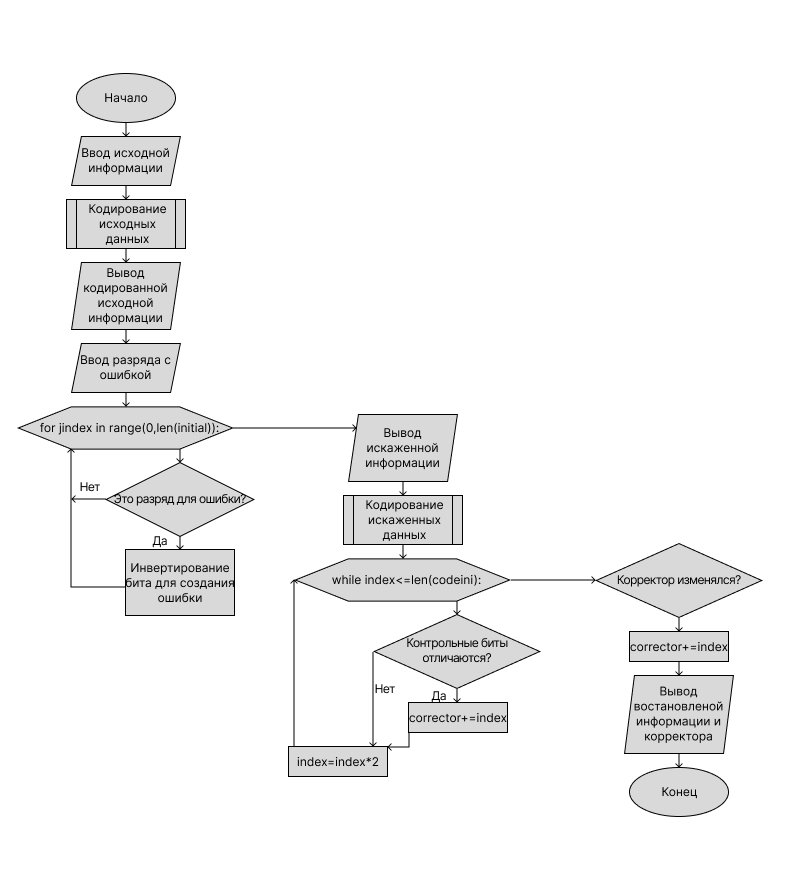


Рисунок 3.1 – блок-схема всего алгоритма

3.2 Алгоритм Кодирования

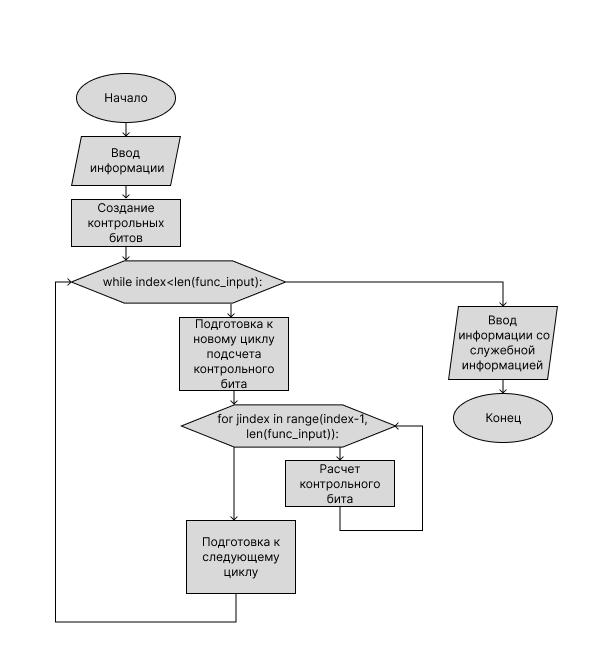


Рисунок 3.2 – алгоритм кодирования

Листинг программы

Ниже приведен листинг программы на языке программирования Python.

#8->4->12

def code(func\_input):

    func\_input=str(func\_input)

    func\_input="00"+func\_input[0]+"0"+func\_input[1:4]+"0"+func\_input[4:]

    func\_input=[int(x) for x in func\_input]

    index=1

    while index<len(func\_input):

        corrector=0

        counter=index

        decounter=0

        for jindex in range(index-1, len(func\_input)):

            if decounter>0:

                decounter-=1

                continue

            corrector+=func\_input[jindex]

            counter-=1

            if counter == 0:

                decounter=index

                counter=index

        func\_input[index-1]=corrector%2

        index\*=2

    return func\_input

initial=input("\nВведите исходную информацию (8 битов)")

codeini=code(initial)

res=""

res=[str(x)+res for x in codeini]

out\_str=""

print(f"\nИсходная веденная информация c контрольными битами 1, 2, 4 и 8: {out\_str.join(res)}")

werror=int(input("\nВведите искаженный разряд: "))-1

initial=str(initial)

errinit=""

for jindex in range(0,len(initial)):

    if jindex==werror:

        temp=str(abs(int(initial[jindex])-1))

        errinit+=temp

        continue

    errinit+=initial[jindex]

out\_str=""

print(f"\nИскаженная в разряде {werror+1} информация: {out\_str.join(errinit)}")

errinit=out\_str.join(errinit)

codeerr=code(errinit)

#Начало восстановления

index=1

corrector=-1

while index<=len(codeini):

    if codeini[index-1]!=codeerr[index-1]:

        corrector+=index

    index=index\*2

if corrector>-1:

    codeerr[corrector]=abs(codeerr[corrector]-1)

index=1

result=""

for jindex in range(0, len(codeerr)):

    if jindex==index-1:

        index\*=2

        continue

    result=result+str(codeerr[jindex])

print(f"\nВосстановленная информация: {result}, корректор: {corrector+1}.")

Тестирование программы

Входные данные: 11010010. Позиция с ошибкой: 3. Заранее программой предопределенна длина информационных битов в 8, служебных в 4, и итоговая в 12.

Выходные данные: получаем восстановленные данные и корректор со значением 6.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1 – тестирование программы.