Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по практической работе №4 «Определение параметров линейного кода»

Выполнил:

студент группы ИП-014

Бессонов А.О.

Работу проверил:

старший преподаватель

Дементьева К.И.

Новосибирск 2024 г.

# **Постановка задачи**

Цель работы: Изучение свойств линейного корректирующего кода

Язык программирования: С, С++, С#, Python

Результат: программа, тестовые примеры, отчет.

Задание:

1. Порождающая матрица записана в текстовом файле. Файл имеет следующий формат: в первой строке через пробел записаны два натуральных числа n (количество строк матрицы) и m (количество столбцов), в следующих n строках записаны через пробел по m нулей и единиц. Файл можно генерировать случайно.

Пример файла

3 5

1 0 1 1 1

0 1 0 1 0

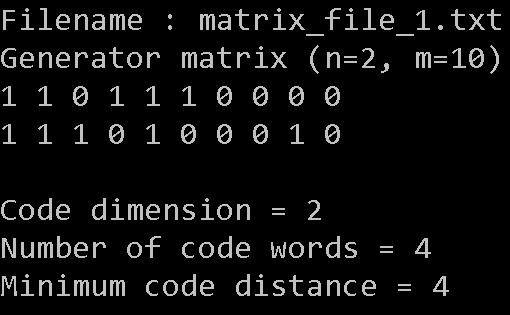
0 0 1 1 1

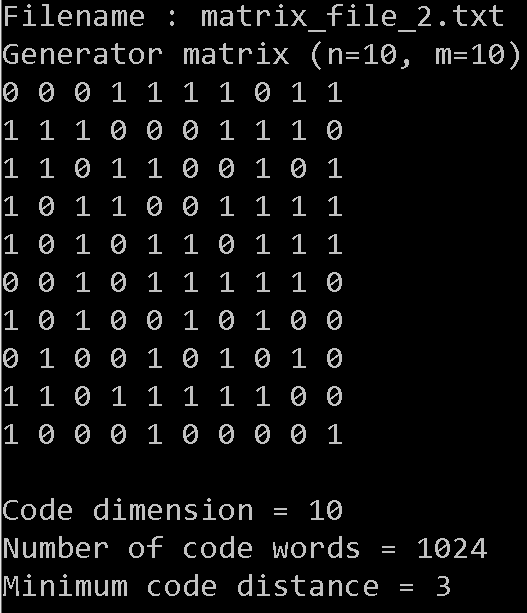
2. По заданной порождающей матрице определить характеристики линейного кода: размерность кода, количество кодовых слов, минимальное кодовое расстояние. Использовать 5 различных файлов.

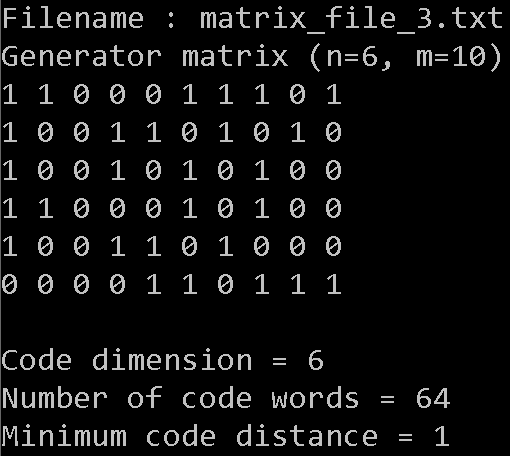
# **Ход работы**

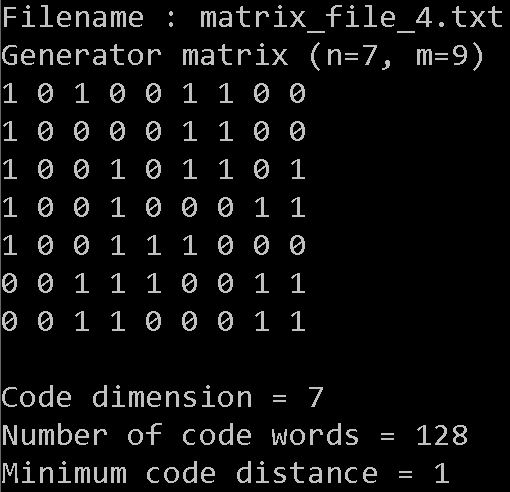
Файлы с порождающими матрицами генерируются случайным образом в программе, с условием того, что число столбцов не меньше числа строк, а так же с условием того, что строки в матрице не повторяются.

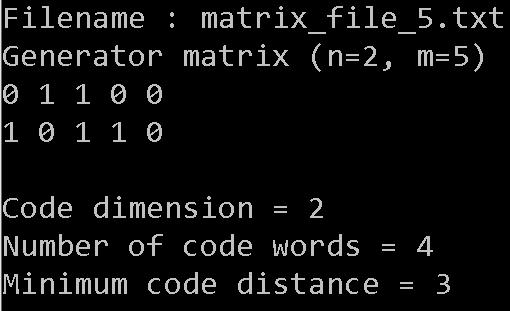
*Результаты работы программы:*











# **Выводы**

В практической работе было реализовано нахождение свойств линейного корректирующего кода по файлу с порождающей матрицей.

# **Код программы**

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <random>  #include <string>  #include <vector>  **using** **namespace** std;  **void** **generate\_file**(string filename) {  ofstream output(filename);    random\_device rd;  mt19937 gen(rd());  uniform\_int\_distribution<**int**> n\_distribution(**1**, **10**);  **int** n = n\_distribution(gen);  uniform\_int\_distribution<**int**> m\_distribution(n, **10**);  **int** m = m\_distribution(gen);  output << n << " " << m << "**\n**";  uniform\_int\_distribution<**int**> bool\_distribution(**0**, **1**);  vector<vector<**int**>> generator\_matrix;  **for** (**int** i = **0**; i < n; i++) {  vector<**int**> new\_generator\_row;  **while** (true) {  new\_generator\_row.clear();  **for** (**int** j = **0**; j < m; j++)  new\_generator\_row.push\_back(bool\_distribution(gen));  **bool** found = false;  **for** (vector<**int**> generator\_row : generator\_matrix) {  **if** (new\_generator\_row == generator\_row) {  found = true;  **break**;  }  }  **if** (!found)  **break**;  }  generator\_matrix.push\_back(new\_generator\_row);  **for** (**int** value : new\_generator\_row)  output << value << " ";  output << "**\n**";  }  output.close();  }  **void** **determine\_characteristics**(string filename) {  ifstream input(filename);    **int** n, m;  input >> n >> m;  vector<vector<**int**>> generator\_matrix;  **for** (**int** i = **0**; i < n; i++) {  vector<**int**> generator\_row;  **for** (**int** j = **0**; j < m; j++) {  **int** value;  input >> value;  generator\_row.push\_back(value);  }  generator\_matrix.push\_back(generator\_row);  }  input.close();  cout << "Filename : " << filename << "**\n**";  cout << "Generator matrix (n=" << n << ", m=" << m << ")**\n**";  **for** (vector<**int**> generator\_row : generator\_matrix) {  **for** (**int** value : generator\_row)  cout << value << " ";  cout << "**\n**";  }  cout << "**\n**";  **int** code\_dimension = n;  cout << "Code dimension = " << code\_dimension << "**\n**";  //int code\_length = m;  //cout << "Code length = " << code\_length << "\n";  **int** number\_of\_code\_words = **1** << n;  cout << "Number of code words = " << number\_of\_code\_words << "**\n**";  //double block\_code\_redundancy = (double) m / n;  //cout << "Block code redundancy = " << block\_code\_redundancy << "\n";  //double code\_speed = (double) n / m;  //cout << "Code speed = " << code\_speed << "\n";  **int** minimum\_code\_distance = m;    **for** (**int** i = **0**; i < n; i++) {  **for** (**int** j = i + **1**; j < n; j++) {  **int** code\_distance = **0**;  **for** (**int** k = **0**; k < m; k++)  **if** (generator\_matrix[i][k] != generator\_matrix[j][k])  code\_distance++;    **if** (code\_distance < minimum\_code\_distance)  minimum\_code\_distance = code\_distance;  }  }  cout << "Minimum code distance = " << minimum\_code\_distance << "**\n**";  cout << "**\n\n**";  }  **int** **main**() {  **for** (**int** i = **1**; i <= **5**; i++) {  string filename = "matrix\_file\_" + to\_string(i) + ".txt";  generate\_file(filename);  determine\_characteristics(filename);  }  **return** **0**;  } |