**Отчет о прохождении учебной практики**

# Задача 1

1. **МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ**

Задача: дана квадратная матрица A размерности n×n и номер столбца k. Вычислить ее определитель методом разложения по k столбцу.

Определитель (детерминант) матрицы является многочленом от элементов квадратной матрицы.

Для матрицы первого порядка детерминантом является единственный элемент этой матрицы (формула 1):

Для матрицы второго порядка детерминант определяется как (формула 2):

Для матрицы n-го порядка детерминант задается рекурсивно (формула 3):

Где — дополнительный минор к элементу a₁ⱼ. Эта один из видов формулы разложения по строке.

Дополнительный минор матрицы n-го порядка это определитель матрицы (n – 1)-го порядка, полученный из матрицы А вычеркиванием i-й строки и j-го столбца.

Алгебраическое дополнение Aᵢⱼ элемента aᵢⱼ определяется формулой:



Теорема о разложении определителя по элементам столбца. Определитель матрицы A равен сумме произведений элементов столбца на их алгебраические дополнения:

1. **ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ**

Алгоритм работы программы состоит из следующих шагов:

1. Программа запрашивает размерность матрицы, номер столбца, по которому надо вычислить определитель, и значение элементов матрицы.
2. Затем проверяет размерность матрицы и если она не больше двух, то сразу вычисляет определитель по готовым формулам 1 и 2.
3. Если размерность матрицы больше двух, то для каждого aᵢⱼ элемента заданного столбца находится дополнительный минор с помощью вычеркивания i-й строки и j-го столбца из исходной матрицы. В каждом дополнительном миноре данный алгоритм повторяется, до тех пор, пока не получится минор второго порядка. Тогда он вычисляется по формуле 2 и результат поступает в общую формулу 3.
4. Знак для каждого элемента столбца вычисляется по формуле (-1)ⁱ⁺ʲ.
5. Результат формулы 3 равен определителю заданной матрицы.
6. Вывод результата работы программы и окончание работы программы.
7. **ТЕХНИЧЕСКОЕ ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА**

Программа написана на языке программирования С++.

1. Данные поступают в программу из файла matr.txt, где указаны размерность матрицы n, номер столбца k, по которому надо найти определитель, и сама матрица \*\*a. Матрица из файла записывается в программу с помощью динамического двумерного массива.
2. Чтобы программа работала корректно надо ввести размерность матрицы равную количеству строк/столбцов и номер столбца в диапазоне от 1 до n. В качестве значений элементов матрицы могут выступать только элементы множества целых чисел.
3. Если данные были введены некорректно, программа сообщит об этом.
4. Далее матрица \*\*a(\*\*mas), ее размер n(m), номер столбца k, размер n1 и номер столбца j уменьшенные на единицу поступают в рекурсивную функцию Opred. В функции создается дополнительная двумерная матрица \*\*p для хранения миноров.
5. Если размер матрицы n равен 1, то функция сразу выводит определитель d равный единственному элементу этой матрицы.
6. Если размер матрицы n равен 2, то функция считает определитель с помощью выражения: d = mas[1][1] \* mas[0][0] - (mas[0][1] \* mas[1][0]).
7. Если размер матрицы n равен 3 и больше, то находится минор для каждого элемента столбца k, с помощью функции Minor. Тем самым уменьшается порядок матрицы до тех пор, пока ее размер не станет равным 2. После чего каждый определитель второго порядка можно вычислить по предыдущей формуле, а общий определитель с помощью выражения d = d (±) t \* mas[i][j] \* Opred(p, n, k, n1, j), где t переменная для чередования знаков.
8. Функция Minor предназначена для уменьшения i-ой строки и j-го столбца. На ее вход поступают текущая матрица mas и дополнительный минор p, который на порядок меньше mas, а также индекс строки i, номер столбца k и размер матрицы m. Переменные ki и kj служат индексами строк и столбцов минора p, а di и dj предназначены для изменения индексов строк и столбцов текущей матрицы mas. Полученная новая матрица p, индексы которой были скорректированы, снова поступает в функцию Opred, для дальнейшего вычисления определителя.
9. Функция PrintMatr выводит миноры p для матриц mas выше второго порядка m.
10. После выхода из функции Opred программа выводит полученный результат определителя d и записывает его в файл opred.txt.
11. **ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ**

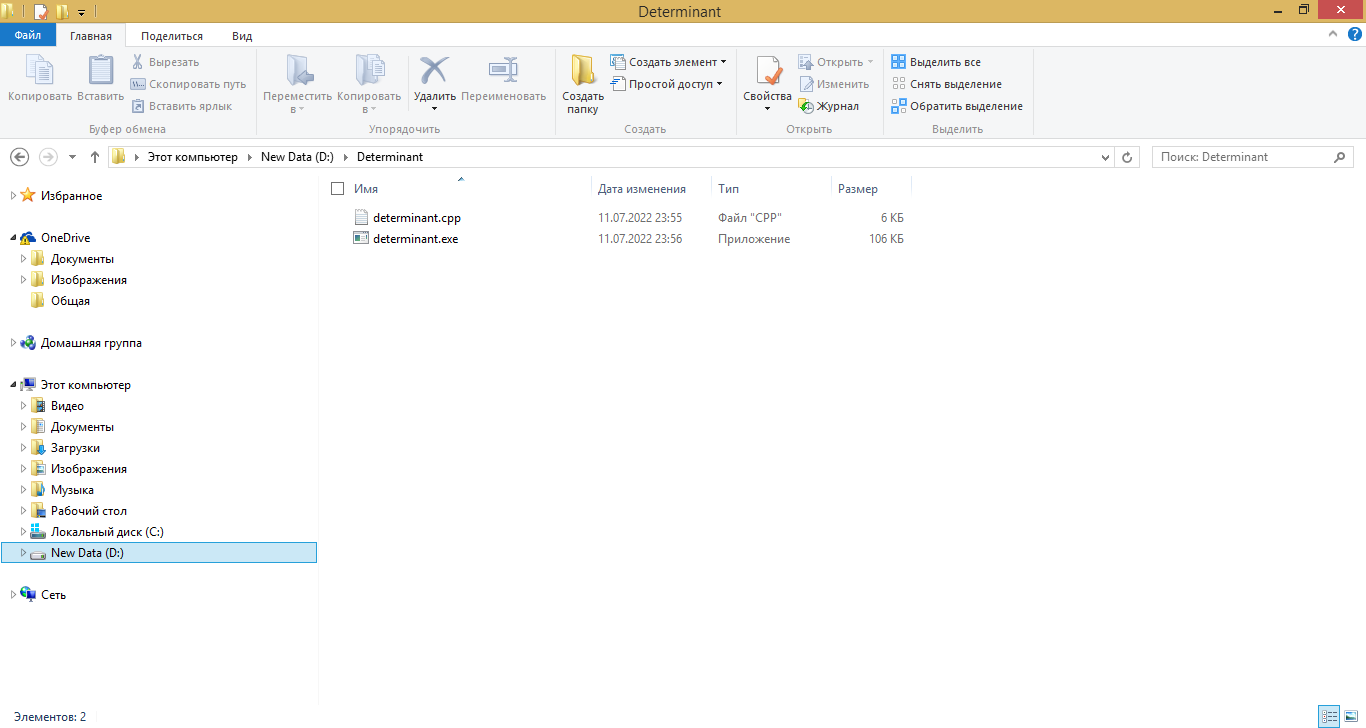
Для начала работы необходимо перейти на сайт <https://drive.google.com/file/d/1eVdRkd1cdnkX0X2z74DQPe-5ps8RYQi5/view?usp=sharing> и скачать папку Determinant.zip после чего распаковать ее в любую папку, путь к которой не содержит русских символов.

Рисунок 1 – Содержимое папки

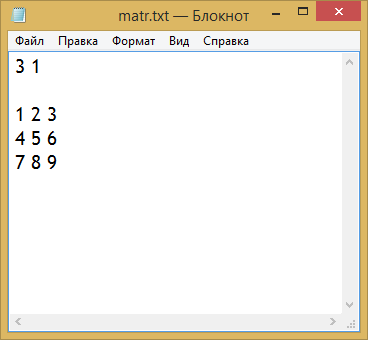
Далее необходимо в папке с программой создать файл с именем matr.txt и записать в него через пробел размер матрицы, номер столбца по которому необходимо вычислить определитель и саму матрицу.

Рисунок 2 — Пример ввода данных в текстовом файле

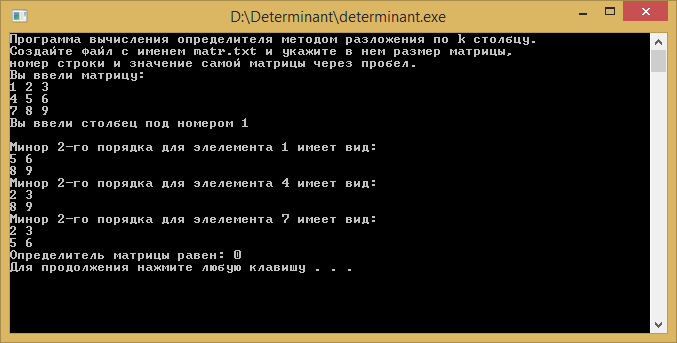
После необходимо запустить файл determinant.exe.

Рисунок 3 — Окно программы с полученными данными и результатом

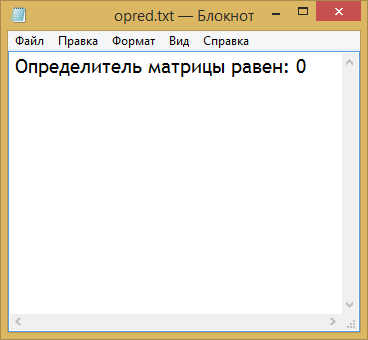
Результат работы программы будет выведен на экран, а также программа создаст файл с именем opred.txt, где будет храниться вычисленное значение определителя заданной матрицы.

Рисунок 4 — Полученный результат в текстовом файле

Если до запуска программы determinant.exe пользователь не создаст файла matr.txt, то программа сообщит об этом.

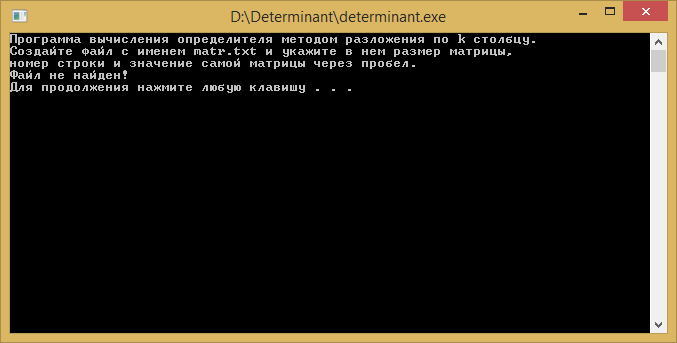


Рисунок 5 — Окно программы при отсутствии файла matr.txt

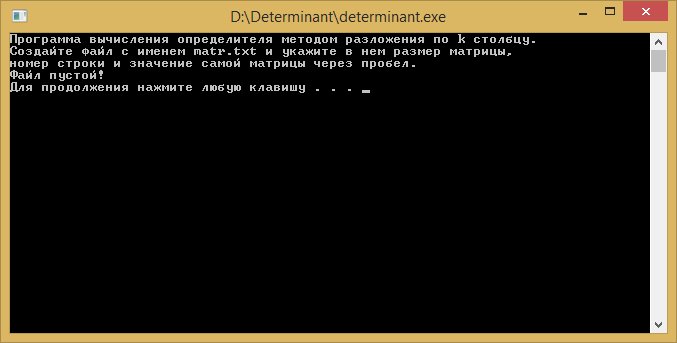
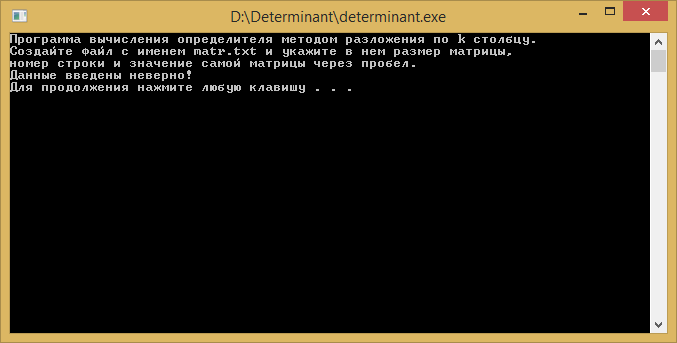
Если пользователь не сохранит данные в файл matr.txt и запустит программу determinant.exe, то она выдаст ошибку.

Рисунок 6 — Окно программы в случае если файл matr.txt пуст

Если пользователь сохранит неверные данные в файл matr.txt и запустит программу sinus.exe, то она выдаст ошибку.

Рисунок 7 — Окно программы при полученных некорректных данных 

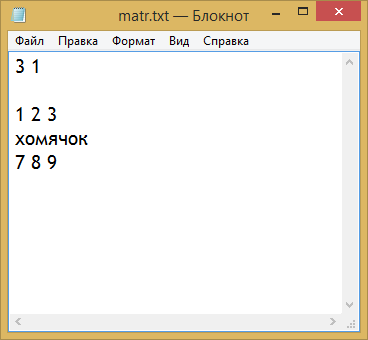


Рисунок 8 — Пример фала matr.txt с некорректными данными

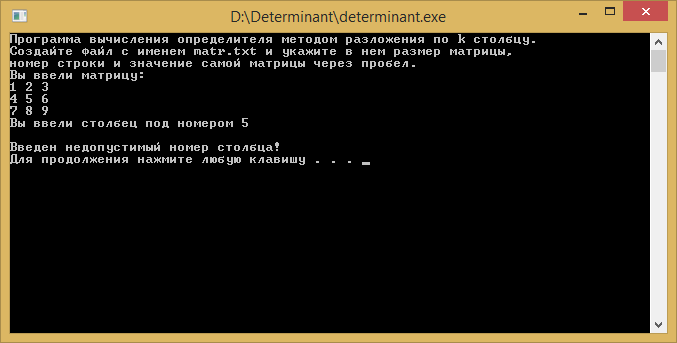
Если пользователь введет недопустимый номер столбца, то программа сообщит об этом.

Рисунок 9 — Окно программы при недопустимом номере столбца

Для окончания работы необходимо нажать любую клавишу или крестик в правом верхнем углу программы.

1. **НАБОР ИНФОРМАЦИИ ДЛЯ ТЕСТИРОВАНИЯ**

Таблица 1 содержит пример входных данных и выходные данные, которые выводит программа.

Таблица 1 — Набор данных для тестирования

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1 | 1 1  25 |  |
| 2 | 2 1  5 17  28 7 |  |
| 3 | 3 2  1 2 3  4 5 6  7 8 9 |  |
| 4 | 4 3  5 -7 1 4  6 2 -1 8  -9 5 7 -4  5 -6 -5 0 |  |
| 5 | 5 5  2 -1 3 4 -5  4 -2 7 8 -7  -6 4 -9 -2 3  3 -2 4 1 -2  -2 6 5 4 -3 |  |
| 6 | 6 5  1 2 3 4 5 6  2 3 4 5 6 1  3 4 5 6 1 2  4 5 6 1 2 3  5 6 1 2 3 4  6 1 2 3 4 5 |  |
| 7 | 4 4  1 2 3 4  3 6 8 11  7 14 20 26  31 23 55 42 |  |
| 8 | 3 2  1 2 -1  2 1 -1  1 -7 5 |  |
| 9 | 3 1  2 1 8  -3 -1 -2  4 5 -3 |  |
| 10 | 2 2  5 7  8 9 |  |

# Задача 2

* 1. **МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ**

Задача: реализовать программный продукт, позволяющий выполнить кодирование или декодирование вводимых двоичных последовательностей методом Хэмминга. Пользователь может ввести один блок с клавиатуры, тогда параметры выбираются в зависимости от его длины или указать файл с последовательностью, тогда кодирование происходит кодом (31,26,5).

Код Хэмминга — самоконтролирующийся и самокорректирующийся код. Построен применительно к двоичной системе счисления. Позволяет исправлять одиночную ошибку (ошибка в одном бите слова) и находить двойную.

Самоконтролирующийся код позволяет автоматически обнаруживать ошибки при передаче данных. Для этого метод Хэмминга использует избыточное кодирование.

Избыточное кодирование — вид кодирования, использующий избыточное количество информации с целью последующего контроля целостности данных при записи/воспроизведении информации или при её передаче по линиям связи.

Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися.

* 1. **ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ**

Алгоритм работы программы состоит из следующих шагов:

1. Программа запрашивает номер команды и значение двоичной последовательности.
2. Если требуется закодировать последовательность программа добавляет к последовательности из 26 символов 5 контрольных бит, на места равные степеням двойки, то есть 1, 2, 4, 8, 16. И присваивает им значение 0.
3. Далее программа вычисляет значение каждого контрольного бита. Для этого вычисляется сумма битов, контролируемых информационным битом. Учитывая правило, что каждый контрольный бит с номером n контролирует все последующие n бит через каждые n бит, начиная с позиции n. Если полученная сумма четная контрольному биту присваивается значение 0, иначе 1. Можно и наоборот, однако алгоритм при декодировании должен быть точно таким же.
4. Полученный результат является закодированной двоичной последовательностью, после чего программа выводит ее и заканчивает свою работу.
5. Если требуется декодировать последовательность программа вычисляет значение каждого контрольного бита по алгоритму, описанному выше.
6. Полученная последовательность сравнивается с исходной. Если есть отличия, программа складывает значения позиций ошибочных контрольных битов, и заменяет значение бита на противоположное, на месте вычисленной суммы.
7. У исправленной и изначально верной двоичной последовательности удаляются контрольные биты.
8. Полученный результат является декодированной двоичной последовательностью, после чего программа выводит ее и заканчивает свою работу.
   1. **ТЕХНИЧЕСКОЕ ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА**

Программа написана на языке программирования С++.

1. Данные поступают в программу из файла posled.txt, где указаны номер команды r, и значение двоичной последовательности a.
2. Если r равен 1, программа закодирует введенную двоичную последовательность, а если равен 2 — декодирует.
3. Чтобы программа работала корректно надо ввести допустимый номер команды 1 или 2 и одну двоичную последовательность, состоящую только из 26 нулей и единиц для кодирования и 31 нуля и единицы для декодирования.
4. Если данные были введены некорректно, программа сообщит об этом.
5. Для кодирования к данной последовательности a[n] с помощью цикла for добавляется пять контрольных бит на позиции i с номерами, равными степеням step двойки. Им присваивается значение 0, и полученная последовательность записывается в массив b[n+5], где n количество элементов исходной последовательности, а i и t временные итераторы для массивов.
6. С помощью цикла for вычисляется для каждого информационного бита сумма sum контролируемых им битов. Если сумма четная, то контрольному биту присваивается значение 0, иначе 1. Переменные i и j являются временными итераторами для массива, а переменная p временный итератор для цикла.
7. Далее программа выводит закодированную двоичную последовательность b[n+5] и записывает ее в файл kod.txt.
8. Для декодирования данной последовательности a[n] с помощью цикла for вычисляется для каждого информационного бита сумма sum контролируемых им битов. Однако теперь в сумму не входят значения самих контрольных бит, которые находятся позициях i с номерами, равными степеням step двойки. Если сумма четная, то контрольному биту присваивается значение 0, иначе 1. Переменные i и j являются временными итераторами для массива, а переменная p временный итератор для цикла. Последовательность с пересчитанными контрольными битами записывается в массив b[n].
9. Если исходная последовательность a[n] равна новой b[n], значит ошибок нет, иначе вычисляется сумма kontrol значений позиций неправильных контрольных бит, с помощью цикла for, и значение бита на позиции равной kontrol заменяется на противоположное. Переменная iter служит временным итератором для цикла.
10. Далее удаляются все контрольные биты, и полученная последовательность записывается в массив c[n-5]. Переменная t служит временным итератором для массива.
11. После программа выводит декодированную двоичную последовательность c[n-5] и записывает ее в файл kod.txt.
    1. **ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ**

Для начала работы необходимо перейти на сайт <https://drive.google.com/file/d/1bC29n-VSoPzHlZLIwzFPGsb81JvYnPwB/view?usp=sharing> и скачать папку Hemming.zip после чего распаковать ее в любую папку, путь к которой не содержит русских символов.

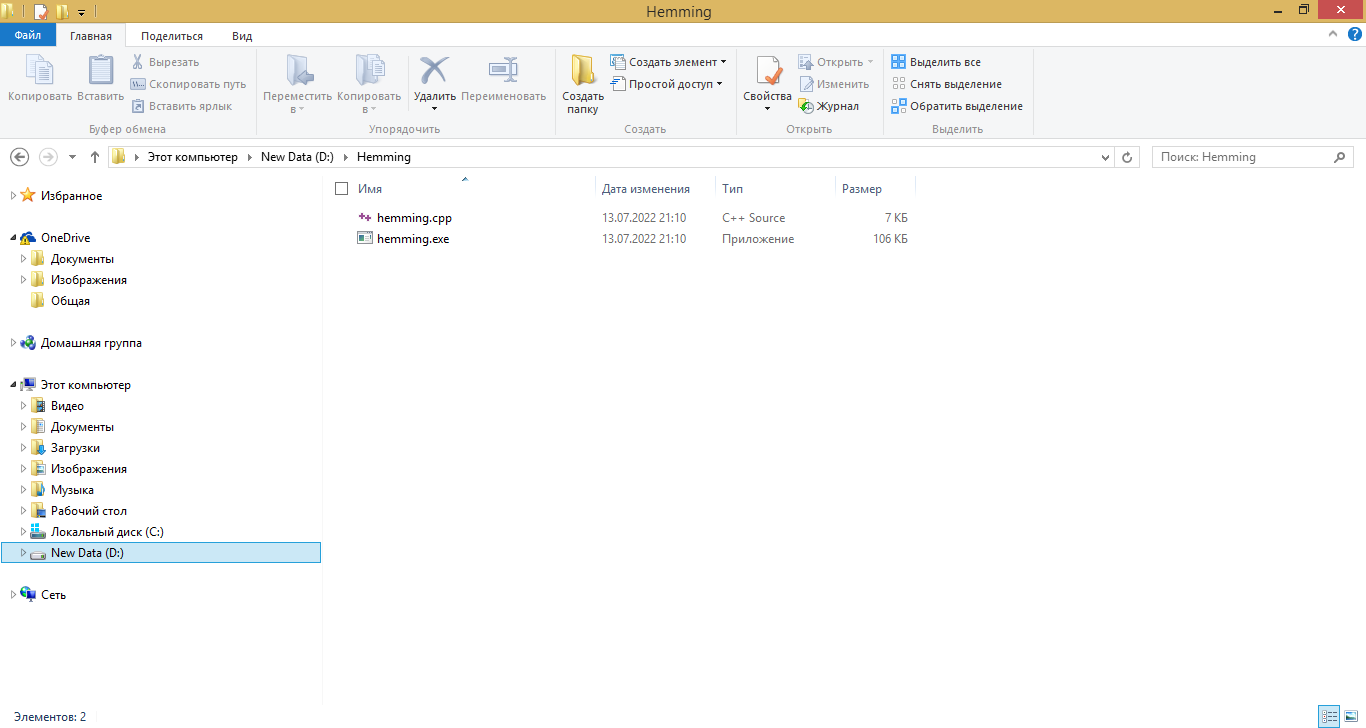


Рисунок 1 – Содержимое папки

Далее необходимо в папке с программой создать файл с именем posled.txt и записать в него номер команды (1 — для кодирования двоичной последовательности, а 2 — для декодирования) и значение двоичной последовательности через пробел.

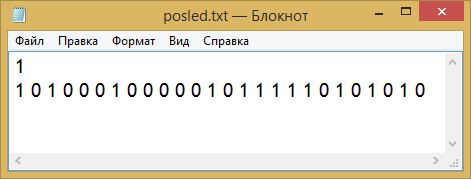


Рисунок 2 — Пример ввода данных в текстовом файле

После необходимо запустить файл hemming.exe.

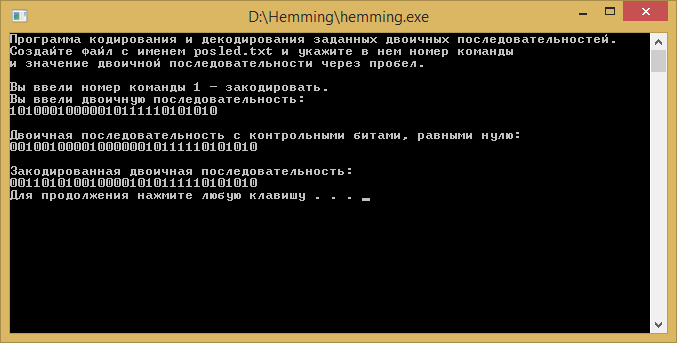


Рисунок 3 — Окно программы с полученными данными и результатом

Результат работы программы будет выведен на экран, а также программа создаст файл с именем kod.txt, где будет храниться закодированная или декодированная двоичная последовательность.

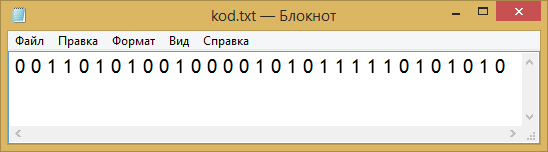


Рисунок 4 — Полученный результат в текстовом файле

Если до запуска программы hemming.exe пользователь не создаст файла posled.txt, то программа сообщит об этом.

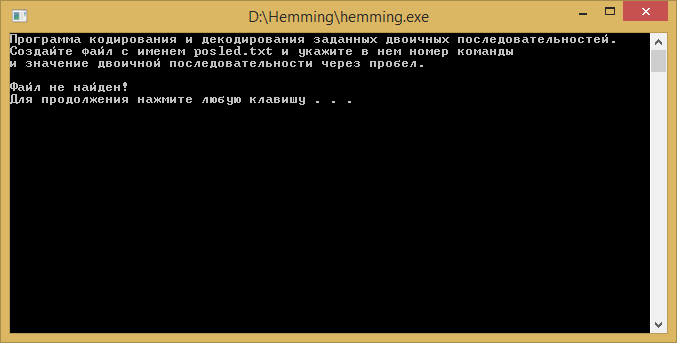


Рисунок 5 — Окно программы при отсутствии файла posled.txt

Если пользователь не сохранит данные в файл posled.txt и запустит программу hemming.exe, то она выдаст ошибку.

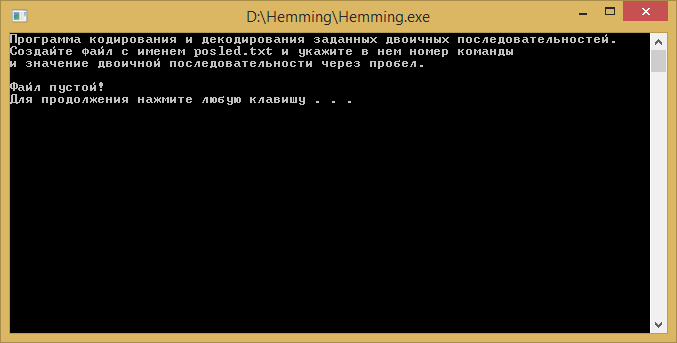


Рисунок 6 — Окно программы в случае если файл posled.txt пуст

Если пользователь сохранит неверные данные в файл posled.txt и запустит программу hemming.exe, то она выдаст ошибку.

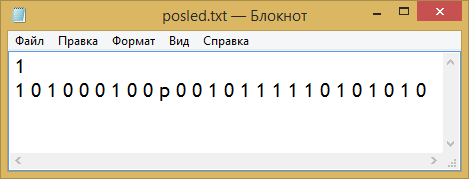


Рисунок 7 — Окно программы при полученных некорректных данных

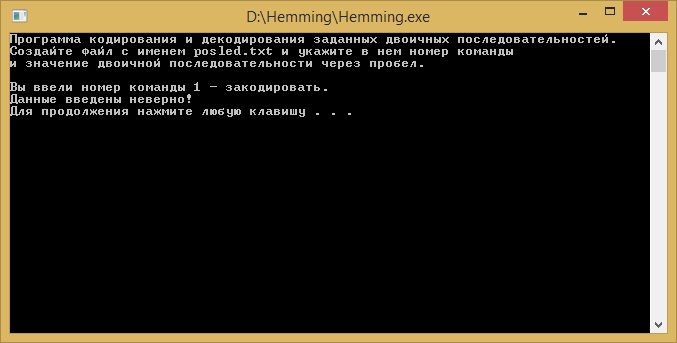


Рисунок 8 — Пример фала posled.txt с некорректными данными

Если пользователь введет недопустимый номер команды, то программа сообщит об этом.

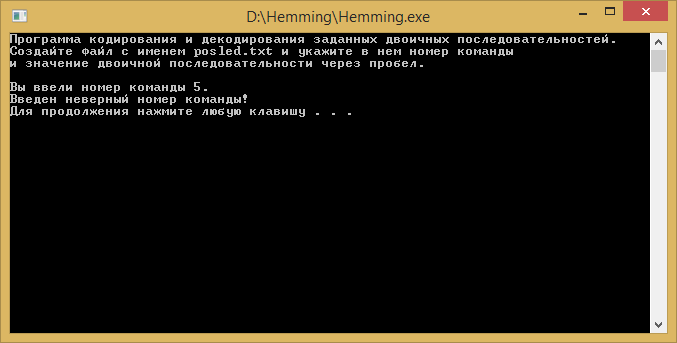


Рисунок 9 — Окно программы при недопустимом номере команды

Для окончания работы необходимо нажать любую клавишу или крестик в правом верхнем углу программы.

* 1. **НАБОР ИНФОРМАЦИИ ДЛЯ ТЕСТИРОВАНИЯ**

Таблица 1 содержит пример входных данных и выходные данные, которые выводит программа.

Таблица 1 – Набор данных для тестирования

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1 | 1  1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 |  |
| 2 | 2  0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 |  |
| 3 | 1  1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 |  |
| 4 | 2  0 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 |  |
| 5 | 1  0 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 |  |
| 6 | 2  1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 0 0  1 1 1 0 |  |
| 7 | 1  1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 |  |
| 8 | 2  0 0 1 0 0 1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1  0 1 0 1 |  |
| 9 | 1  1 0 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 |  |
| 10 | 2  0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 |  |

# ЗАКЛЮЧЕНИЕ

Приобретен опыт в реализации программных продуктов, основанных на знаниях, полученных за время обучения на первом курсе: разработана программа вычисляющая определитель рекурсивным методом, а также разработан программный продукт для кодирования и декодирования двоичных последовательностей методом Хэмминга.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дискретная математика для программистов; Новиков Ф.А.; Учебник для вузов. 3-е изд. — СПб.: Питер, 2009. — 384 с.: ил. — (Серия «Учебник для вузов»).
2. 'Основы математического анализа'; В 2 томах. Ильин, В.А.; Позняк, Э.Г.; Изд-во: Наука, 1982 г.
3. Курс алгебры. — 5-е изд., Винберг Э. Б.; стереотип. — М.: МЦНМО, 2021. — 592 с.: ил.
4. Методы программирования: учеб.-метод. пособие/ авт. В. В. Подколзин, А. Н. Полетайкин, Е. П. Лукащик, О. В. Гаркуша, С. Г. Синица, А. А. Полупанов, А. В. Харченко, А. В. Уварова, А. А. Михайличенко. — Краснодар: Кубанский гос. ун-т, 2020. — 174 с. — 500 экз.