

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

**КУРСОВАЯ РАБОТА**

Тема: «РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА НА ОСНОВЕ ЦИКЛИЧЕСКИХ КОДОВ»

Дисциплина: «Теория информации»

Специальность: 10.05.01 Компьютерная безопасность

Специализация: Математические методы защит информации

Обозначение курсовой работы ТИ.190000.000 Группа ВКБ31

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Р. Теклюк

подпись, дата

Курсовая работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель, И.А. Алферова

подпись, дата

Ростов-на-Дону

2022



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ЗАДАНИЕ**

на выполнение курсовой работы

Тема «РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА НА ОСНОВЕ ЦИКЛИЧЕСКИХ КОДОВ»

Дисциплина: Теория информации

Обучающийся: Теклюк Артем Романович

Обозначение курсовой работы ТИ.190000.000 Группа: ВКБ31

Срок представления работы к защите «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Исходные данные для курсовой работы:

1. С.Б. Саломатин, П.Г. Семашко Методическое пособие “Исследование циклических кодов”

2. О.С. Когновицкий, В.М. Охорзин “Теория помехоустойчивого кодирования”

|  |  |  |  |
| --- | --- | --- | --- |
| **Содержание пояснительной записки** | | | |
| Введение:  Актуальность решения задачи. | | | |
| Разделы основной части: | | | |
| 1. Кодирование при помощи циклических кодов. Изучение помехоустойчивого кодирования, циклические коды. 2. Разработка программного средства. Обоснование выбора языка и среды разработки, описание основных методов и классов. 3. Тестирование программного средства. Демонстрация работы программы на примере. | | | |
| Заключение:  Выводы по результатам разработки | | | |
| Руководитель работы | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | И.А. Алферова |
|  |  |  |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | А.Р. Теклюк |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Содержание**  [Введение 5](#_bookmark0)   1. [Помехоустойчивое кодирование 7](#_bookmark1)    1. [Общая информация по кодированию 7](#_bookmark2)    2. [Циклические коды. Схема построения 9](#_bookmark3) 2. [Разработка программного средства 12](#_bookmark4)    1. [Обоснование выбора языка и среды разработки 12](#_bookmark5)    2. [Архитектура программного средства 13](#_bookmark6)    3. [Основные методы и классы 14](#_bookmark7)    4. [Демонстрация программного средства 15](#_bookmark8) 3. [Тестирование программного средства 18](#_bookmark9)   [Заключение 20](#_bookmark10)  [Перечень использованных информационных ресурсов 21](#_bookmark11)  [Приложение А 22](#_bookmark12)  [Приложение Б Листинг кода 26](#_bookmark13) | | | | | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | | | | | |
|  |  |  |  |  |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |
| *Разраб.* | | *Теклюк В.А.* |  |  | *Реализация программного средства на основе циклических кодов*  *Пояснительная записка* | *Лит.* | | | *Лист* | *Листов* |
| *Провер.* | | *Алферова И.А.* |  |  |  |  |  | *4* |  |
|  | |  |  |  | *ДГТУ*  *Кафедра КБИС* | | | | |
|  | |  |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Введение**  Сфера связи и информатизации является одной из важнейших инфраструктур и ей принадлежит особая роль во многих сферах деятельности общества.  Важной задачей является создание и развитие инфокоммуникационной инфраструктуры. Передаваемая в виде различных сигналов информация может представлять собой речь, данные, видеоизображение или любую их комбинацию, называемую мультимедийной. Наблюдается тенденция к увеличению количества передаваемой по сети телекоммуникаций мультимедийной информации неизбежно приводит к возрастанию информационных потоков.  Увеличение скоростей передачи данных в канале влечет за собой возрастание количества случайных помех, способных исказить хранимые, обрабатываемые и передаваемые данные. Это обуславливает актуальность разработки и использования новых алгоритмов и методов, позволяющих обнаруживать и корректировать подобные ошибки. В данных условиях особого внимания заслуживают помехоустойчивые коды, позволяющие эффективно бороться со случайными ошибками в канале.  Помехоустойчивое кодирование занимает важное место в современных системах связи. На сегодняшний день существует большое количество способов помехоустойчивого кодирования, которые активно используются в различных системах передачи данных и, следовательно, сохраняют актуальность исследования. Данная работа посвящена изучению одного из наиболее распространенных классов помехоустойчивых кодов, основанного на использовании двоичных кодовых комбинаций, а именно, двоичных циклических кодов. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *5* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Объект исследования: алгоритмы циклических кодов.  Предмет исследования: реализация программного средства на основе циклических кодов.  Цель работы заключается в изучении особенностей помехоустойчивого кодирования и реализации его на практике.  Задачи исследования:   1. Изучение научной литературы по теме исследования. 2. Анализ возможностей, достоинств и недостатков помехоустойчивого кодирования. 3. Анализ возможных вариантов использования данного варианта кодирования. 4. Реализация циклического кода. 5. Реализация программного средства кодирования и декодирования с помощью циклических кодов. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *6* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **Помехоустойчивое кодирование**    1. **Общая информация по кодированию**   Основной задачей помехоустойчивого кодирования является решение проблемы обеспечения высокой достоверности передаваемых данных за счет применения устройств кодирования/декодирования в составе системы передачи цифровой информации.  Рассмотрим основной принцип работы. Сначала источник данных порождает данные в виде двоичных символов. Обычно предполагают, что «нули» и «единицы» появляются независимо друг от друга и с одинаковыми вероятностями. Затем кодер канала вносит в принятую информационную последовательность некоторую избыточность (данный процесс называется кодированием), которую декодер сможет использовать для исправления возникающих при передаче данных по каналу связи ошибок.  Закодированные данные с выхода кодера поступают на модулятор, который с помощью какого-либо метода модуляции реализует их отображение в аналоговый сигнал S(t). В физическом канале сигнал S(t) подвергается воздействию шума n(t). Для количественной оценки степени влияния шума n(t) на сигнал S(t) обычно используют отношение сигнал-шум Es /N0.  Далее демодулятор преобразует принятый из канала сигнал R(t) в последовательность чисел, представляющих оценку переданных данных. После этого детектор квантует выход демодулятора на Q-уровней.  Затем квантованный выход детектора поступает на декодер канала, который, используя внесенную кодером избыточность, определяет переданное источником сообщение (данный процесс называется декодированием).  Сначала рассмотрим самые простые помехоустойчивые коды, данные алгоритмы отличают простота реализации, ограниченность обнаружения и исправления ошибок, а также тот факт, что данные коды в современном | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *7* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| помехоустойчивом кодировании используются в составе других более сложных кодов (например, в составе кодов Рида-Соломона, Турбокодов, Каскадных кодов и т.п.). К ним относятся следующие коды.  Код Хэмминга, как и коды с проверкой на четность, позволяет обнаруживать одиночные ошибки, но в отличии от них имеет более совершенную структуру, а именно: в данном коде для каждого числа проверенных символов используется специальная маркировка, которая состоит из двух составляющих, а именно, количества символов в сообщении и количества информационных символов в сообщении. Данный код также используется в составе более сложных кодов.  Код Боуза-Чаудхури-Хоквингхема или же, как его ещё называют, код БЧХ, данный код, в своё время, активно использовался в аппаратуре передачи данных, но в последнее время был вытеснен более совершенными кодовыми алгоритмами. Данный код отличается специальным выбором образующего циклический код полинома, что позволяет более просто декодировать передаваемую информацию, также данный код обладает нечётными значениями минимального кодового расстояния, всё это позволяет превосходно обнаруживать и исправлять ошибки с учётом группирования.  Блочные неравномерные коды, особенностью этих кодов является то, что в данных алгоритмах все кодовые комбинации содержат разное число разрядов с постоянной длительностью импульсов. Данный вид кодов не является корректирующим и способен только обнаруживать ошибки в передаваемой информации благодаря большой избыточности. Примерами таких кодов являются код Морзе, который в настоящее время используется как специализированный код передачи информации в военной сфере, и код Хаффмана, который применяется для компрессии информации и в настоящий момент используется в составе некоторых Каскадных кодов. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *8* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Циклический избыточный код (CRC код) это еще один код, который занимается обнаружением ошибок, данный код используется наравне с кодом Хэмминга и с кодами с проверкой на чётность. Отличительной особенностью данного кода является его способ кодирования информации, основанный на свойстве деления с остатком двоичных многочленов, что позволяет с хорошей точность определять ошибки в передаваемых сообщениях.  **1.2 Циклические коды. Схема построения.**  Широкое применение в различных системах находят циклические коды, используемые как для обнаружения, так и для исправления ошибок. В протоколах канального уровня, как правило, используются циклические коды для обнаружения ошибок, вследствие чего их реализация довольно проста. Требуемая эффективность, при этом, достигается за счет применения обратной связи. Однако во многих системах применение обратной связи неприемлемо, так как она вносит существенные задержки. Особенно это характерно для трафика реального времени комбинации.  Циклическим называется код, каждая комбинация которого может быть получена путем циклического сдвига комбинации, принадлежащей этому же коду. Если сдвиг осуществляется справа налево, крайний левый символ переносится в конец кодовой комбинации (таблица 1).   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | 0 | | 1 | 0 | 1 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | 0 | 1 |     Таблица 1 – Сдвиг комбинации  Описание циклических кодов удобно проводить с помощью многочленов. Для этого вводят фиктивную переменную x, степени которая соответствует номерам разрядов, начиная с 0. В качестве коэффициентов многочленов берут цифры 0 и 1. Например, первая строка из примера (таблица 1) описывается | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *9* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| многочленом:  0  *x*5  0  *x*4  1 *x*3  0  *x*2 1 *x*1  1 *x*0  *x*3  *x*  1.  Многочлен для каждой следующей строки образуется из предыдущего путем умножения на x. При этом, если крайний левый символ отличается от нуля для реализации операции переноса единицы в конец комбинации из результата необходимо вычесть (сложить по модулю 2) многочлен xn +1.  Все комбинации циклического кода могут быть построены на кольце многочленов путем задания на множестве n -разрядных кодовых комбинаций двух операций – сложения и умножения. Операция сложения многочленов в данном случае реализуется как сложение соответствующих коэффициентов по модулю 2.  Операция умножения реализуется в следующей последовательности. Многочлены перемножаются как обычно с последующим приведением коэффициентов по модулю 2. Если в результате умножения получается многочлен степе- ни n и выше, то осуществляется его деление на заданный многочлен степени n , а результатом умножения считают остаток от деления.  Ясно, что старшая степень этого остатка не будет превышать величины *n*-1, а полученный остаток будет соответствовать некоторой n -разрядной кодовой комбинации, т.е. обеспечивается замкнутость.  Для реализации циклического сдвига с использованием описанной операции умножения необходимо после умножения на x выполнить деление на двучлен *xn* 1. Эта операция называется *взятием остатка* или *приведением по модулю xn* 1, а сам остаток называют вычетом:  [https://i.gyazo.com/beb7ec5288175f159282cd557583ff1f.png](https://gyazo.com/beb7ec5288175f159282cd557583ff1f)  Рисунок 1 – операция взятия остатка  В данном случае остаток (вычет) формируется путем сложения по | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *10* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| модулю 2 двучлена *xn* 1 с результатом умножения на *x.*  Обнаружение ошибок при циклическом кодировании сводится к делению принятой кодовой комбинации на тот же образующий полином Р(х), который использовался при кодировании. Если ошибок в принятой кодовой комбинации нет, то деление на образующий полином производится без остатка. Если при делении получится остаток, то это свидетельствует о наличии ошибки. Остаток от деления в циклических кодах играет роль «синдрома».  Для определения местоположения ошибки в циклическом коде существует ряд методов, основанных на анализе «синдрома» R(x).  Основным функциональным узлом кодирующих и декодирующих устройств циклических кодов является схема деления. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *11* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

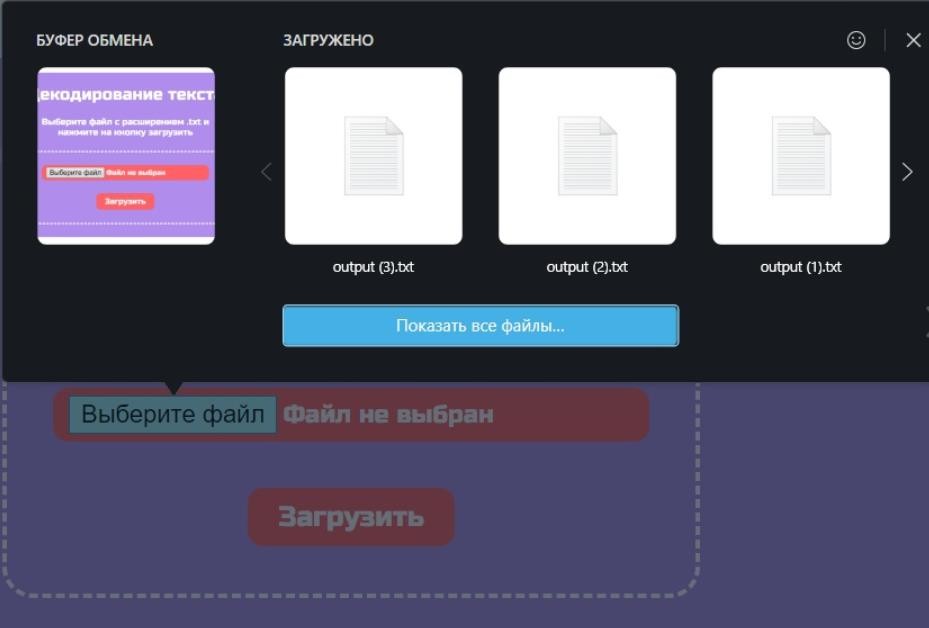
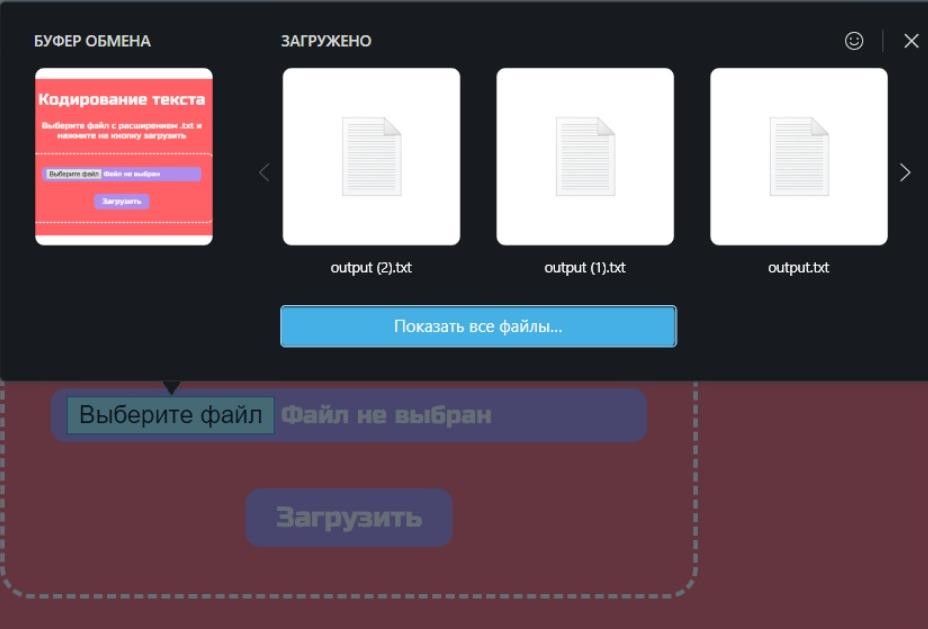
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **Разработка программного средства**    1. **Обоснование выбора языка и среды разработки**   Для данной работы в качестве языка для разработки был выбран Python.  Python — это язык программирования высокого уровня, но его легко изучить. Любой желающий может научиться программировать на Python всего за несколько часов или несколько дней. Освоение Python и всех его сложных концепций, пакетов и модулей может занять больше времени. Однако изучить базовый синтаксис очень легко по сравнению с другими популярными языками, такими как C, C ++ и Java.  Помимо стандартных библиотек Python, к нему можно подключить бесчисленное множество дополнительных модулей и библиотек, доступных для всех. Это означает, что программистам не нужно писать свой код для каждой отдельной вещи, в отличие от других языков программирования. Существуют библиотеки для работы с изображениями, базы данных, модульное тестирование и множество других функций.  Python — это язык программирования высокого уровня, поскольку программистам не нужно помнить архитектуру системы и управлять памятью. Это делает его удобным для программистов и является одной из ключевых особенностей Python.  Так же, используя Python удобно работать с массивами и циклами, что является основой в моей программе. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *12* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **2.2 Архитектура программного средства**  Данная программа относится к такому виду архитектур программных средств таких, как монолитная программа.  Ее достоинствами являются:   1. Простая разработка и простой запуск программы. Из-за того, что вся разработка сконцентрирована в одном месте, легче интегрировать инструменты для облегченной разработки, те же каталоги или библиотеки. Плюс при необходимости изменить элементы программы не нужно вносить изменения по отдельности в разных местах — все делается в одном месте. 2. Сквозные проблемы практически отсутствуют. Большое количество приложений имеют зависимость от задач, которые совершаются между компонентами программы: логи, ограничения скорости, контрольные журналы и т. д. При монолитной архитектуре эти проблемы практически отсутствуют, так как все сконцентрировано в одном коде и все работает в одном приложении. 3. Улучшенная производительность. Если учитывать, что приложения были собраны правильно, то одно и то же приложение при монолитной архитектуре будет работать более производительно, чем при микросервисах. Это, опять же, обеспечивается единым кодом программы и работой из «одного» места.   Такую архитектуру я выбрал исходя из ее достоинств, но так же есть и недостатки:   1. Большой объем кода. Если разрабатываемый продукт довольно большой и постоянно масштабируется, то со временем его код разрастается до огромных размеров. Это утяжеляет его понимание и дальнейшее обслуживание. Плюс может наступить момент, когда код будет «перегружен» и потеряет от этого свое качество. 2. Сложно модернизируется. Иногда нужно добавить в приложение какую-то новую «фишку». При монолитной архитектуре можно столкнуться со | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *13* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

| множеством препятствий, чтобы это реализовать. Потому что в некоторых случаях добавить какую-то «фишку» означает полностью переписать приложение. А это долго и дорого.   1. Зависимость между компонентами. С одной стороны, это является достоинством, так как увеличивает производительность. Но с другой стороны, если в каком-то компоненте программы будет ошибка, то это замедлит или вообще остановит работу всего приложения, а не одного компонента.   **2.3 Основные методы и классы**  Tkinter – библиотека для реализации графического интерфейса программы.  Numpy – библиотека для работы с матрицами.  Def perevod – функция перевода последовательности “0” и ”1” в многочлен.  Def mistake – эта функция случайно вносит в каждую закодированную последовательность 1 ошибку.  Def conventor – функция перевода многочлена в последовательность “0” и ”1”.  Def coding – принимает 2 значения: delimoe и delitel и возвращает результат деления и остаток.  Def calculate – с помощью этой функции программа реагирует на ввод данных пользователем.  Def OK\_polinom – вывод введенного пользователем полинома.  Def OK\_inf\_slovo\_1 – вывод введенного пользователем 1-го информационного слова.  Def OK\_inf\_slovo\_2 – вывод введенного пользователем 2-го | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *14* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| информационного слова.  Def OK\_inf\_slovo\_3 – вывод введенного пользователем 3-го информационного слова.  Def Matrix – вывод матрицы со сдвигом.  Def X\_umnojenie – посимвольное умножение строк матрицы с использованием информационных слов.  Def decoding\_search\_miss – поиск ошибки, декодирование и вывод результата.  **2.4 Демонстрация программного средства**  При открытии приложения можно увидеть сразу весь интерфейс программы. Ее функция заключается в том, чтобы наглядно и быстро производить кодирование, зашумление и декодирование многочленов (Рисунок 2).  [https://i.gyazo.com/4c59c73d4c423bea4dd11dca75f599aa.png](https://gyazo.com/4c59c73d4c423bea4dd11dca75f599aa)  Рисунок 2 – Окно программы  Ввод полинома и информационных слов осуществляется в виде многочленов с помощью панели кнопок (Рисунок 3). | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *15* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 3 – Ввод полинома  При нажатии на кнопку, при нажатии на кнопку ОК полином записывается в программу.  Далее вводим информационные слова, чтобы их записать необходимо нажать на соответствующую кнопку, после записи последнего слова выводится n, k, m(Рисунок 4).  [https://i.gyazo.com/d6df20985822aec33f195a191f1aa5e4.png](https://gyazo.com/d6df20985822aec33f195a191f1aa5e4)  Рисунок 4 – Ввод инф. слов | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *16* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Далее нажатием на соответствующие кнопки выводятся матрица и закодированные слова по 1-му способу кодирования (в двоичной системе) и по 2-му (в виде полиномов) (Рисунок 4).  [https://i.gyazo.com/5642d2459cde47ab3c6f05b8900d9ef8.png](https://gyazo.com/5642d2459cde47ab3c6f05b8900d9ef8)[https://i.gyazo.com/d4bbb422a282cce9103ea565791c7b50.png](https://gyazo.com/d4bbb422a282cce9103ea565791c7b50)  Рисунок 4 – 1 и 2 способы кодирования  Далее представлен способ декодирования Мэдита. При нажатии на кнопку “Внести ошибки” вносятся ошибки в закодированные последовательности и происходит декодирование с последующим выводом результата в виде полиномов и промежуточных значений декодирования, таких как ex, e’x, Sx, S’x, i’x. (Рисунок 5).  [https://i.gyazo.com/819674dade3d1782a0c4b238e487babf.png](https://gyazo.com/819674dade3d1782a0c4b238e487babf)  Рисунок 5 – Способ декодирования Мэдита | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *17* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **3 Тестирование программного средства**  Проведу тестирование со следующими значениями:  Полином: *x^4+x+1*  1-е информационное слово: x^2+x+1  2-е информационное слово: x^2+1  3-е информационное слово: x^2  [https://i.gyazo.com/08c0645cdd21cb3552a4d401b3be74e5.png](https://gyazo.com/08c0645cdd21cb3552a4d401b3be74e5)  Рисунок 5 – Результат тестирования  Теперь возьмем более сложные значения:  Полином: x^8+x^5+x^4+x+1  1-е информационное слово: x^6+x^3+x  2-е информационное слово: x^5+x+1  3-е информационное слово: x^3+x^2+x+1 | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *18* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [https://i.gyazo.com/fa6446ede1dc84df50140648f78a6ee2.png](https://gyazo.com/fa6446ede1dc84df50140648f78a6ee2)  Рисунок 5 – Результат тестирования №2  Сравнивая начальные значения информационных слов с конечными, можно сделать вывод, что программа работает корректно. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *19* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Заключение**  Задача кодирования является одним из главных понятий информатики, так как кодирование предшествует передаче и хранению информации, и, соответственно, является основой их успешного осуществления.  При передаче сообщений по каналам связи могут возникать помехи, способные привести к искажению принимаемых знаков. Эта проблема решается с помощью помехоустойчивого кодирования. Помехоустойчивое кодирование передаваемой информации позволяет в приемной части системы обнаруживать и исправлять ошибки. Коды, применяемые при помехоустойчивом кодировании, называются корректирующими кодами.  В ходе курсовой работы были рассмотрены такие алгоритмы кодирования, как Помехоустойчивое кодирование, кодирование при помощи циклических кодов и способ декодирования Мэдита.  Выполнены следующие поставленные задачи:   1. Изучена научная литература по теме исследования. 2. Проанализированы возможности, достоинства и недостатки помехоустойчивого кодирования информации. 3. Проанализированы возможные варианты использования данного варианта кодирования. 4. Приведен пример циклического кода. 5. Реализовано программное средство кодирования и декодирования с помощью циклических кодов.   В результате выполнения курсовой работы была достигнута поставленная цель, заключающаяся в анализе особенностей помехоустойчивого кодирования и реализации его на практике. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *20* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Перечень использованных информационных ресурсов**   1. Помехоустойчивое кодирование [Электронный ресурс]. – URL: <https://www.referat911.ru/Informatika/pomehoustojchivoe-kodirovanie/296261-2647172-place1.html> (Дата обращения: 29.05.2022.) 2. Помехоустойчивое кодирование [Электронный ресурс]. – [URL: https://www.opds.spbsut.ru/data/\_uploaded/mu/teor\_kod/kogn13up\_tpk-1\_cikl\_kody.pdf](URL:%20https://www.opds.spbsut.ru/data/_uploaded/mu/teor_kod/kogn13up_tpk-1_cikl_kody.pdf) **(**Дата обращения: 29.05.2022) 3. Циклические коды [Электронный ресурс]. – URL: https://www.bibliofond.ru/view.aspx?id=806418 (Дата обращения: 30.05.2022.) 4. Золотарёв В.В., Овечкин Г.В. Справочное издание “Помехоустойчивое кодирование. Методы и алгоритмы” [Текст]. – Горячая линия-Телеком, 2004г. – 5,6 с. 5. Монолитная архитектура. Традиционный метод разработки приложений [Электронный ресурс]. – URL: <https://codernet.ru/articles/drugoe/monolitnaya_arxitektura_tradiczionnyij_metod_razrabotki_prilozhenij/> (Дата обращения: 31.05.2022.) 6. Блейхут Р. Теория и практика кодов, контролирующих ошибки = Theory and Practice of Error Control Codes. — М.: Мир, 1986. — 576 с. 7. Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. Теория кодов, исправляющих ошибки. М.: Радио и связь, 1979. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *21* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Приложение А**  Техническое задание  СОГЛАСОВАНО «УТВЕРЖДЕНО»  Руководитель проекта И.о. зав каф. «КБИС»  И.А. Алферова к.т.н. Д.А. Короченцев  (подпись) (подпись)  « \_» 2022 г. « \_» 2022 г.  **А.1 Введение**  **А.1.1 Наименование программы**  Наименование программы – «Программное средство кодирования и декодирования с помощью циклических кодов».  **А.1.2 Область применения**  Областями применения данного программного средства являются учебные заведения, которые нуждаются в обучающей программе для наглядного сопровождения этапов кодирования и декодирования с помощью циклических кодов.  **А.2 Основания для разработки**  Разработка ведется в рамках выполнения курсовой работы по дисциплине  «Теория информации». Тема разработки «Разработка и реализация программного средства кодирования и декодирования с помощью циклических кодов». | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *22* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **А.3 Назначение разработк**и  **А.3.1 Функциональное назначение**  Функциональным назначением программного средства является кодирование полиномов с помощью циклических кодов, внесение ошибок с последующим декодированием.  **А.3.2 Эксплуатационное назначение**  Эксплуатационное назначение: программное средство предназначено для эксплуатации в учебных заведениях, которые нуждаются в обучающей программе для наглядного сопровождения этапов кодирования и декодирования с помощью циклических кодов.  **А.4 Требования к программе**  **А.4.1 Требования к функциональным характеристикам**  Программное средство должно производить кодирование и декодирование данных с использованием алгоритмов циклического кодирования. В программе используются следующие функции:   1. Def conventor – функция перевода многочлена в последовательность “0” и ”1”; 2. Def perevod – функция перевода последовательности “0” и ”1” в многочлен; 3. Def mistake – эта функция случайно вносит в каждую закодированную последовательность 1 ошибку; 4. Def coding – принимает 2 значения: delimoe и delitel и возвращает результат деления и остаток; | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *23* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **А.4.2 Требования к надёжности**  Надёжность функционирования обеспечивается корректным функционированием аппаратного обеспечения и правильным выбором входных данных.  **А.4.3 Условия эксплуатации**  Условия эксплуатации совпадают с условиями эксплуатации персональных. Программа рассчитана на пользователя, владеющего базовыми навыками работы с персональным компьютером. Входные данные – текстовые сообщения. Результат работы не требует пояснений.  **А.4.4 Требования к составу и параметрам технических средств**  Для работы программы требуется наличие ПК – совместимого персонального компьютера с тактовой частотой процессора не менее 1,5 Гц.  **А.4.5 Требования к информационной и программной совместимости**  Базовый язык программирования Python 3.10  **А.4.6 Требования к маркировке и упаковке**  Требований к маркировке и упаковке не предъявлялось. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *24* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **А.4.7 Требования к транспортированию и хранению**  Требования к транспортировке не предъявлялось.  **А.5 Требования к программной документации**  Программная документация должна включать документ «Техническое задание» (ГОСТ 19.201-78)  **А.6 Стадии и этапы разработки**   1. Системный анализ. 2. Общесистемное проектирование. 3. Подготовка технологических средств. 4. Программная реализация, рабочий проект. 5. Отладка программного средства в статике. 6. Тестовые испытания программного комплекса.   **А.7 Порядок контроля и приемки**  Контроль и приемка разработки осуществляется на основе испытаний контрольно-отладочных примеров. Примеры должны демонстрировать правильность работы используемых в программе структур данных и алгоритмов в различных ситуациях, которые могут возникнуть при выполнении программы. При этом проверяются выполнение всех функций программы и полнота документации.  **Разработчик:** Теклюк Артем Романович  **Дата начала разработки:** «24» февраля 2022 г. | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *25* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Приложение Б Листинг кода**  buttons = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "del", "C", "x", "\*\*", "+"] x = 9 y = 70 for button in buttons:  get\_lbl = lambda x=button: calculate(x)  Button(frame1, text=button, bg="silver", font=("Roboto", 10), command=get\_lbl).place(x=x, y=y, width=57, height=39)  x+= 59  if x > 200:  x = 9  y+=45  formula = "0"  def calculate(operation):  global formula  if operation == "C":  formula = ""  elif operation == "del":  formula = formula[0:-1]  else:  if formula == "0":  formula = ""  formula += operation  label\_text.configure(text=formula)  print(formula)  tk = Tk() tk.protocol("WM\_DELETE\_WINDOW", on\_closing) tk.title("Мое приложение") tk.configure(bg="white") tk.resizable(0, 0) tk.geometry("1200x800") tk.wm\_attributes("-topmost", 1)  frame1 = Frame(width=250, height=340, bg = "white") frame1.grid(column = 0, row = 0, sticky="w") frame2 = Frame(width=250, height=340, bg = "white") frame2.grid(column = 0, row = 1, sticky="n") frame3 = Frame(width=400, height=340, bg ="white") frame3.grid(column=1, row=0, sticky="n") frame3.grid\_propagate(False) frame4 = Frame(width=400, height=250, bg ="white") frame4.grid(column=1, row=1, sticky="n") frame4.grid\_propagate(False) frame5 = Frame(width=400, height=340, bg ="white") frame5.grid(column=2, row=0, sticky="n") frame5.grid\_propagate(False) | | | | | | |
|  |  |  |  |  | *ТИ.190000.000 ПЗ* | *Лист* |
|  |  |  |  |  | *26* |
| *Изм.* | *Лист* | *№ докум.* | *Подпись* | *Дата* |