Оглавление

[Введение 6](#_Toc202509606)

[Анализ требований нормативно-правовых документов в области защиты информации 7](#_Toc202509607)

[Изучение алгоритма вычисления 8](#_Toc202509608)

[Реализация программы подсчета 9](#_Toc202509609)

[Заключение 13](#_Toc202509610)

[Список используемых источников 14](#_Toc202509611)

[Приложение А 15](#_Toc202509612)

[Приложение Б 16](#_Toc202509613)

[Приложение В 18](#_Toc202509614)

# Введение

Контрольные суммы и хеш-функции играют важную роль в обеспечении целостности данных, позволяя обнаруживать случайные и преднамеренные изменения информации. Одной из распространённых алгоритмических методик для проверки целостности данных является CRC32 (Cyclic Redundancy Check, 32-bit) — циклический избыточный код с длиной контрольной суммы 32 бита.

CRC32 представляет собой алгоритм вычисления контрольной суммы, основанный на делении входных данных на полином с последующим сохранением остатка от этого деления. Этот метод широко применяется в сетевых протоколах, файловых системах и архиваторах для обнаружения ошибок передачи или хранения данных. В отличие от криптографических хеш-функций, CRC32 не предназначен для защиты от преднамеренного изменения данных, так как не обладает стойкостью к коллизиям и обратимостью. Однако его вычислительная эффективность и простота реализации делают его популярным решением для быстрой проверки целостности.

Алгоритм CRC32 работает побитово или побайтово, используя предопределённый полином. Исходные данные обрабатываются как большое двоичное число, над которым выполняется операция деления по модулю 2. Результатом является 32-битная контрольная сумма, которая может быть использована для верификации данных.

# Анализ требований нормативно-правовых документов в области защиты информации

CRC32, как алгоритм контроля целостности данных должен рассматриваться в контексте современных стандартов. При выполнении практики были проанализированы международные и государственные стандарты, такие как:

RFC 1952 – определяет использование CRC32 для проверки контроля целостности данных в протоколе iSCSI.

RFC 3720 – регламентирует применение CRC32 для контроля целостности данных в протоколе iSCSI.

ГОСТ Р ИСО/МЭК 3309-98 – устанавливает стандарты использования CRC для контроля ошибок в телекоммуникационных протоколах.

ГОСТ Р 56939-2024 – включает требования к реализации механизмов контроля целостности при разработке ПО.

В результате анализа требований было установлено, что CRC32 широко применяется в различных областях информационных технологий благодаря следующим характеристикам:

1. высокой скорости вычисления;

2. эффективности обнаружения случайных ошибок;

3. простоте реализации как в программном, так и в аппаратном обеспечении.

# Изучение алгоритма вычисления

Алгоритм CRC32 использует принцип деления полиномов в конечном поле GF(2), где:

- Данные представляются как двоичный полином M(x)

- Генераторный полином G(x) степени 32 задается явно

- Контрольная сумма - остаток от деления x³²·M(x) на G(x)

Шаг 1: Подготовка полинома

Задается произвольный 32-битный полином G (старший бит всегда 1)

Шаг 2: Инициализация

Регистр CRC инициализируется начальным значением

Шаг 3: Обработка данных

Для каждого бита входного сообщения:

Выполняется XOR старшего бита регистра с входным битом

Регистр сдвигается на 1 бит вправо

Если результат XOR = 1:

Выполняется XOR регистра с полиномом G

Шаг 4: Реверсия

Регистр XOR с конечной маской (обычно 0xFFFFFFFF)

Полученное значение - итоговый CRC32

# Реализация программы подсчета

Была разработана программа функции подсчета CRC32, в которой пользователь вводит полином и выбирает файл. Результатом работы программы является вывод CRC32 для файла.

Для проверки корректности работы алгоритма, были использованы произвольные полиномы, и было произведено сравнение с эталонным значением. Результат работы программы представлен на рисунках 1-4. Код программы представлен в Приложении А-В.

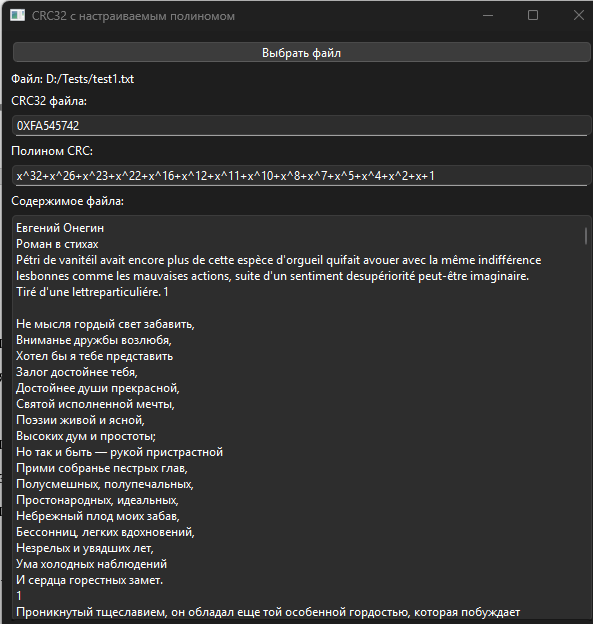


Рисунок 1 – Результат работы программы для 1 файла с 1 полиномом

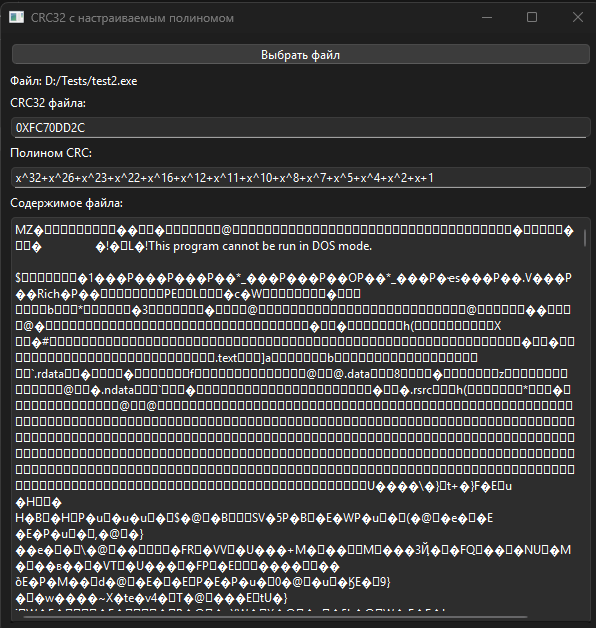


Рисунок 2 – Результат работы программы для 2 файла с 1 полиномом

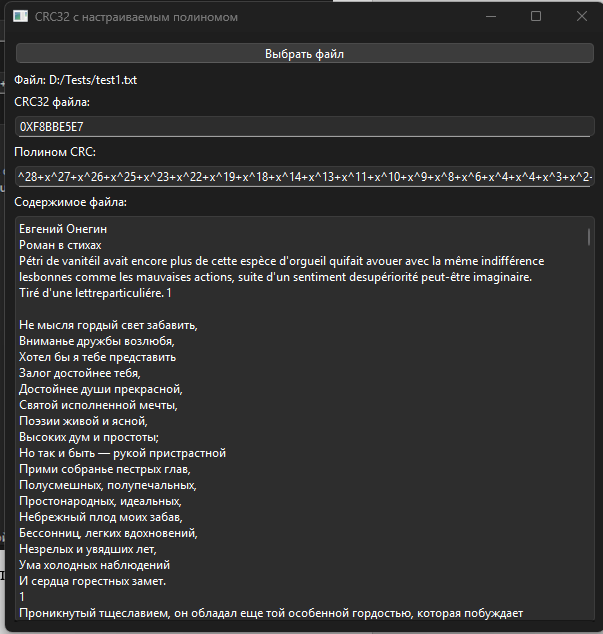


Рисунок 3 – Результат работы программы для 1 файла с 2 полиномом

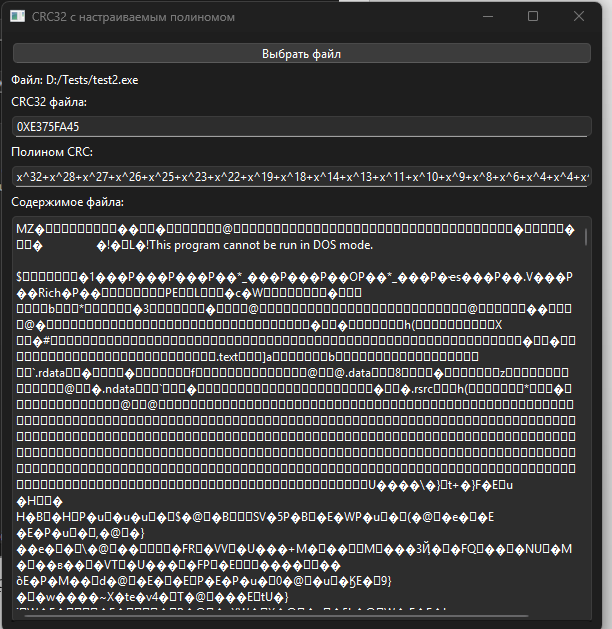


Рисунок 4 – Результат работы программы для 2 файла с 2 полиномом

# Заключение

В результате выполнения практики были проанализированы нормативно-правовые документы в области защиты информации, был изучен алгоритм вычисления CRC32, и была реализована программа функции подсчета CRC32, для файла, на основе вводимого полинома.

# Список используемых источников

1 ПФ АО «НТЦ «Атлас» [Электронный ресурс] – Электрон. Дан. – Режим доступа: <http://www.atlas.sura.ru/> Дата доступа: 26.08.2025

2 RFC 1952 – Internet-Draft Author Resources [Электронный ресурс] – Электрон. Дан. – Режим доступа https://datatracker.ietf.org/doc/html/rfc1952 Дата доступа: 26.08.2025

3 RFC 3720 – Internet Small Computer Systems Interface [Электронный ресурс] – Электрон. Дан. – Режим доступа <https://datatracker.ietf.org/doc/html/rfc3720> Дата доступа: 26.08.2025

4 ГОСТ Р ИСО/МЭК 3309-98 [Электронный ресурс] – Электрон. Дан. – Режим доступа https://files.stroyinf.ru/Data/90/9059.pdf Дата доступа: 26.08.2025

5 ГОСТ Р 56939-2024 [Электронный ресурс] – Электрон. Дан. – Режим доступа <https://meganorm.ru/Data/837/83777.pdf> Дата доступа: 26.08.2025

6 CRC – это просто (деление столбиком) [Электронный ресурс] //Habr – Электрон. Дан. – Режим доступа https://habr.com/ru/articles/770014/ Дата доступа: 26.08.2025

# Приложение А

Содержимое файла main.cpp

#include "mainwindow.h"

#include <QApplication>

// Точка входа в приложение

int main(int argc, char \*argv[]) {

QApplication a(argc, argv); // Создание объекта приложения

MainWindow w; // Создание главного окна

w.show(); // Отображение главного окна

return a.exec(); // Запуск основного цикла обработки событий

}

# Приложение Б

Содержимое файла mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QPushButton>

#include <QLineEdit>

#include <QVBoxLayout>

#include <QFileDialog>

#include <QFile>

#include <QByteArray>

#include <QLabel>

#include <QDropEvent>

#include <QDragEnterEvent>

#include <QTextEdit>

// Главное окно приложения для расчета CRC32

class MainWindow : public QMainWindow {

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

protected:

// Обработчики событий drag-and-drop

void dragEnterEvent(QDragEnterEvent \*event) override;

void dropEvent(QDropEvent \*event) override;

private slots:

// Слоты для обработки действий пользователя

void selectFile(); // Выбор файла через диалог

void calculateCRC32(const QString& filePath); // Расчет CRC32 для файла

private:

// Виджеты интерфейса

QWidget \*centralWidget;

QPushButton \*selectFileButton;

QLineEdit \*resultLineEdit; // Поле для вывода результата

QLineEdit \*polynomialLineEdit; // Поле для ввода полинома

QLabel \*filePathLabel; // Метка для отображения пути к файлу

QTextEdit \*fileContentTextEdit; // Поле для отображения содержимого файла

// Вспомогательные методы

uint32\_t computeCRC32(const QByteArray &data, uint32\_t poly); // Вычисление CRC32

uint32\_t parsePolynomialString(const QString &polyString); // Парсинг строки полинома

};

#endif // MAINWINDOW\_H

# Приложение В

Содержимое файла mainwindow.cpp

#include "mainwindow.h"

#include <QFile>

#include <QFileDialog>

#include <QHBoxLayout>

#include <QVBoxLayout>

#include <QMessageBox>

#include <QMimeData>

#include <QRegularExpression>

// Конструктор главного окна

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent),

centralWidget(new QWidget(this)),

selectFileButton(new QPushButton("Выбрать файл", this)),

resultLineEdit(new QLineEdit(this)),

polynomialLineEdit(new QLineEdit(this)),

filePathLabel(new QLabel("Путь к файлу:", this)),

fileContentTextEdit(new QTextEdit(this))

{

// Настройка основного окна

setWindowTitle("CRC32 с настраиваемым полиномом");

resize(600, 600);

setAcceptDrops(true); // Разрешаем drag-and-drop

// Настройка виджетов

resultLineEdit->setReadOnly(true);

polynomialLineEdit->setPlaceholderText("Введите полином (например: x^32 + x^26 + ... + 1)");

fileContentTextEdit->setReadOnly(true);

fileContentTextEdit->setPlaceholderText("Содержимое файла будет отображено здесь...");

// Создание и настройка layout

QVBoxLayout \*layout = new QVBoxLayout();

layout->addWidget(selectFileButton);

layout->addWidget(filePathLabel);

layout->addWidget(new QLabel("CRC32 файла:", this));

layout->addWidget(resultLineEdit);

layout->addWidget(new QLabel("Полином CRC:", this));

layout->addWidget(polynomialLineEdit);

layout->addWidget(new QLabel("Содержимое файла:", this));

layout->addWidget(fileContentTextEdit);

centralWidget->setLayout(layout);

setCentralWidget(centralWidget);

// Подключение сигналов к слотам

connect(selectFileButton, &QPushButton::clicked, this, &MainWindow::selectFile);

}

MainWindow::~MainWindow() {}

// Обработчик события входа объекта в область окна при drag-and-drop

void MainWindow::dragEnterEvent(QDragEnterEvent \*event) {

if (event->mimeData()->hasUrls())

event->acceptProposedAction();

}

// Обработчик события отпускания объекта при drag-and-drop

void MainWindow::dropEvent(QDropEvent \*event) {

QList<QUrl> urls = event->mimeData()->urls();

if (!urls.isEmpty()) {

QString filePath = urls.first().toLocalFile();

filePathLabel->setText("Файл: " + filePath);

calculateCRC32(filePath);

}

}

// Слот для выбора файла через диалог

void MainWindow::selectFile() {

QString fileName = QFileDialog::getOpenFileName(this, "Выбрать файл");

if (!fileName.isEmpty()) {

filePathLabel->setText("Файл: " + fileName);

calculateCRC32(fileName);

}

}

// Расчет CRC32 для указанного файла

void MainWindow::calculateCRC32(const QString &filePath) {

QFile file(filePath);

if (!file.open(QIODevice::ReadOnly)) {

QMessageBox::critical(this, "Ошибка", "Не удалось открыть файл.");

return;

}

// Чтение содержимого файла

QByteArray data = file.readAll();

fileContentTextEdit->setPlainText(QString::fromUtf8(data));

// Парсинг полинома из строки

QString polyStr = polynomialLineEdit->text();

uint32\_t poly = parsePolynomialString(polyStr);

if (poly == 0) {

QMessageBox::warning(this, "Ошибка", "Не удалось распознать полином.");

return;

}

// Вычисление и вывод CRC32

uint32\_t crc = computeCRC32(data, poly);

QString crcStr = QString("0x%1").arg(crc, 8, 16, QLatin1Char('0')).toUpper();

resultLineEdit->setText(crcStr);

}

// Парсинг строки с полиномом в числовое представление

uint32\_t MainWindow::parsePolynomialString(const QString &polyString) {

QString input = polyString;

input.remove(" ");

if (!input.contains("x")) return 0;

uint32\_t result = 0;

// Регулярное выражение для поиска членов полинома

QRegularExpression regex("x\\^?(\\d\*)");

QRegularExpressionMatchIterator i = regex.globalMatch(input);

while (i.hasNext()) {

QRegularExpressionMatch match = i.next();

QString degreeStr = match.captured(1);

int degree = 1;

if (!degreeStr.isEmpty())

degree = degreeStr.toInt();

if (degree < 0 || degree > 32) continue;

result |= (1u << degree);

}

// Проверка на наличие свободного члена (+1)

if (input.contains("+1") || input.endsWith("1"))

result |= 1u;

return result;

}

// Вычисление CRC32 для данных с заданным полиномом

uint32\_t MainWindow::computeCRC32(const QByteArray &data, uint32\_t polynomial) {

uint32\_t table[256]; // Предварительно вычисленная таблица

// Заполнение таблицы

for (uint32\_t i = 0; i < 256; ++i) {

uint32\_t crc = i;

for (int j = 0; j < 8; ++j)

crc = (crc >> 1) ^ ((crc & 1) ? polynomial : 0);

table[i] = crc;

}

// Вычисление CRC32 с использованием таблицы

uint32\_t crc = 0xFFFFFFFF;

for (unsigned char byte : data)

crc = (crc >> 8) ^ table[(crc ^ byte) & 0xFF];

return crc ^ 0xFFFFFFFF; // Финальная инверсия битов

}