Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт кибербезопасности и защиты информации

ЛАБОРАТОРНАЯ РАБОТА № 9

«Методы контроля целостности»

1. по дисциплине «Основы информационной безопасности»
2. Выполнил
3. студент гр. 4851004/10001 Веремейчук Я.Ю.

<подпись>

1. Преподаватель
2. асс. преподавателя Панков И.Д.

<подпись>

1. Санкт-Петербург
2. 2022
3. Цель работы

Приобретение навыков в разработке и использовании программных средств контроля целостности информации, анализ свойств алгоритмов расчета контрольных сумм и циклических кодов.

1. Постановка задачи

Разработать программу, реализующую расчет восьмиразрядной контрольной суммы, разработать программу вычисления CRC с помощью правил полиномиального деления.

1. Теоретические исследования

Целостность – состояние информации, при котором отсутствует ее изменение либо изменение осуществляется только авторизованными субъектами. Наиболее простой способ проверки целостности – подсчет контрольных сумм.

Контрольная сумма – число, рассчитанное путем сложения всех кодов из потока входных данных. Если полученный результат превышает некоторое максимально допустимое значение, то контрольная сумма С – остаток от деления: С = Т % (М + 1), где Т – сумма входных данных, М – максимально допустимое значение контрольной суммы. Если сумма кодов всех символов сообщения отличается от эталонного, то целостность информации нарушена.

Недостаток метода контрольных сумм заключается в том, что, хотя несовпадение значений сумм служит доказательством изменения информации, их равенство не гарантирует, что информация осталась неизменной.

Более совершенным способом проверки достоверности данных является вычисление циклического избыточного кода (cyclic redundancy check, CRC).

Механизм CRC основан на полиномиальной арифметике. Делимое, делитель, частное и остаток рассматриваются не как целые числа, а как полиномы с двоичными коэффициентами. Число записывается в виде двоичной строки, биты которой служат коэффициентами полинома.

В контрольном суммировании важна полиномиальная арифметика по модулю 2, где все коэффициенты вычисляются по модулю 2. В качестве *х* берется 2 и коэффициенты приводятся к двоичным с учетом переноса единицы в старшие разряды: например, *х*7+*х*З+*х*2+*х*1+*х*0, что соответствует числу 143 (10001111b). Это обычная арифметика, в которой основание системы счисления записано явно. Суть в том, что если *х* неизвестно, то невозможно производить переносы. Если х=2, то перенос возможен: 3*х*3 = *х*4 +*х*З. В полиномиальной арифметике отношения между коэффициентами не определены, и коэффициенты при разных степенях полностью изолированы.

Некоторые типы ошибок, которые встречаются при передаче:

1. Ошибка в одном бите. *E* = 100...000. Такие ошибки можно отловить, если в *G* не менее двух битов установлено в 1. Как бы мы ни складывали сдвиги таких *G*, всегда получим строку, в которой, по крайней мере, два единичных бита, значит, Е не может быть кратно *G;*
2. Ошибка в двух битах. Надо подобрать *G*, для которых не являются кратными числа типа 1 1, 101, 1001, п т.д.;
3. Ошибка в нечетном количестве бит. Такие ошибки можно отловить, взяв *G*, в котором количество бит четно. Это следует из того, что CRC-умножение является результатом XOR константы в аккумулятор с разными сдвигами. XOR – поразрядная операция, инвертирующая те биты аккумулятора, напротив которых в прибавляемой строке стоит 1. Если сделать XOR кода с четным числом единиц, то в аккумуляторе инвертируется четное количество единиц, в результате чего четность количества единиц аккумулятора сохранится;
4. Ошибка пакета. *E* = 000...000111...111000...000. Ошибка локализована в непрерывном пакете внутри сообщения. *E* можно представить в виде произведения: *E* = (1000…00)(111111111), где первый множитель содержит *Z* нулей, а второй – *N*единиц. Если в *G* младший бит – 1, то левый множитель не может быть делителем G, тогда, если *G* длиннее правого множителя, то ошибка будет отловлена.

Качественным критерием оценки контрольной суммы является вероятность возникновения коллизии, то есть необнаружения искажения данных: подразумевается искажение данных таким образом, что подсчитанный CRC (от искаженных данных) совпадет с референсным CRC. Причём, чем длиннее сообщение, тем больше вероятность появления коллизии. Поэтому для расчета CRC применяют определенные полиномы.

Зная CRC, можно узнать о факте наличия ошибки при передаче текста, саму ошибку, возникшую при передаче текста, нельзя исправить, зная всего лишь CRC. Для того, чтобы локализовать ошибку и устранить ее, необходимо заново пересчитывать CRC для блоков текста.

1. Описание решения

Было выбрано число 11100101b и рассчитан LRC8 с помощью полиномиального деления (рисунок 1), LRC8 с помощью сдвиговых операций (рисунок 2), CRC для полинома (8, 7, 1, 0) с помощью полиномиального деления (рисунок 3).

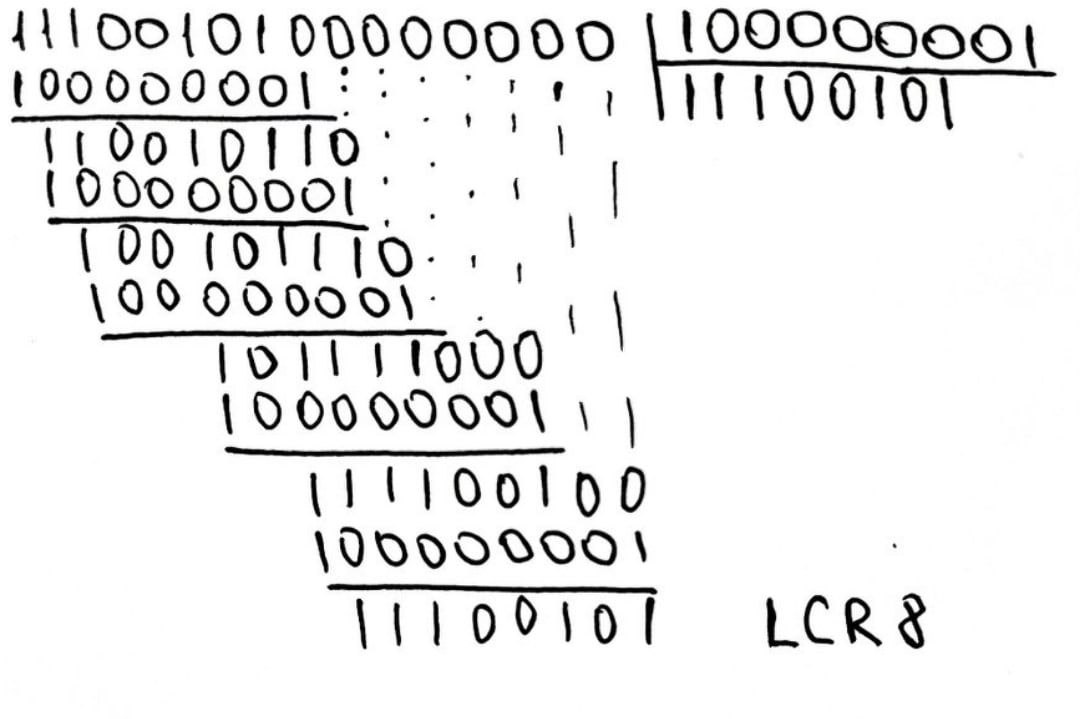


Рисунок 1 – LRC8 с помощью полиномиального деления

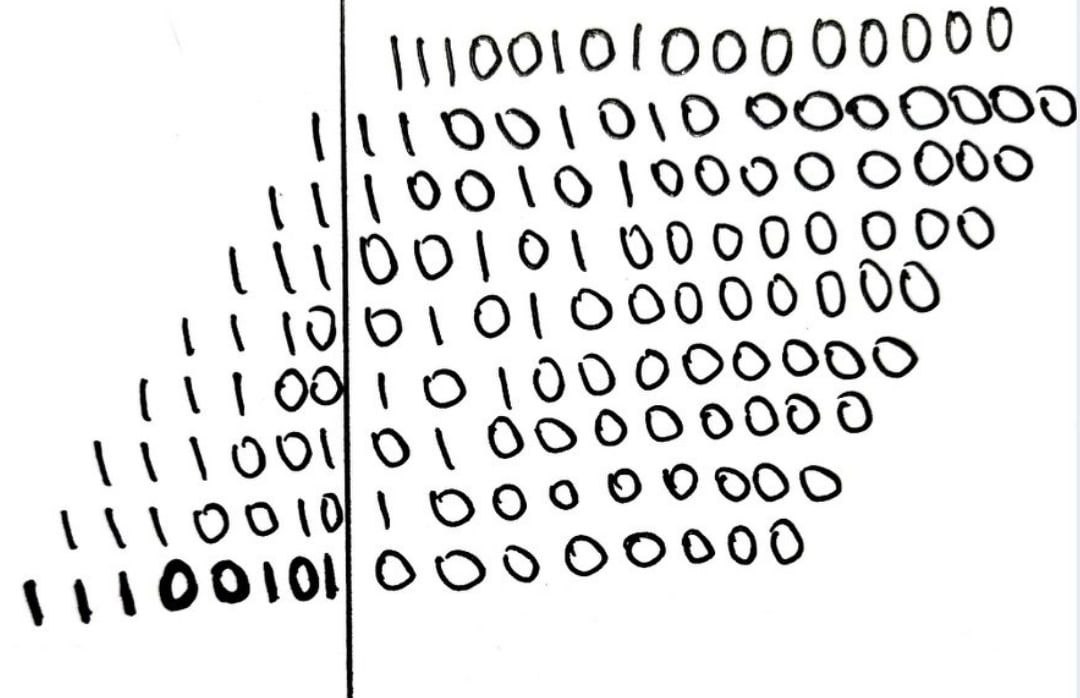


Рисунок 2 – LRC8 с помощью сдвигов

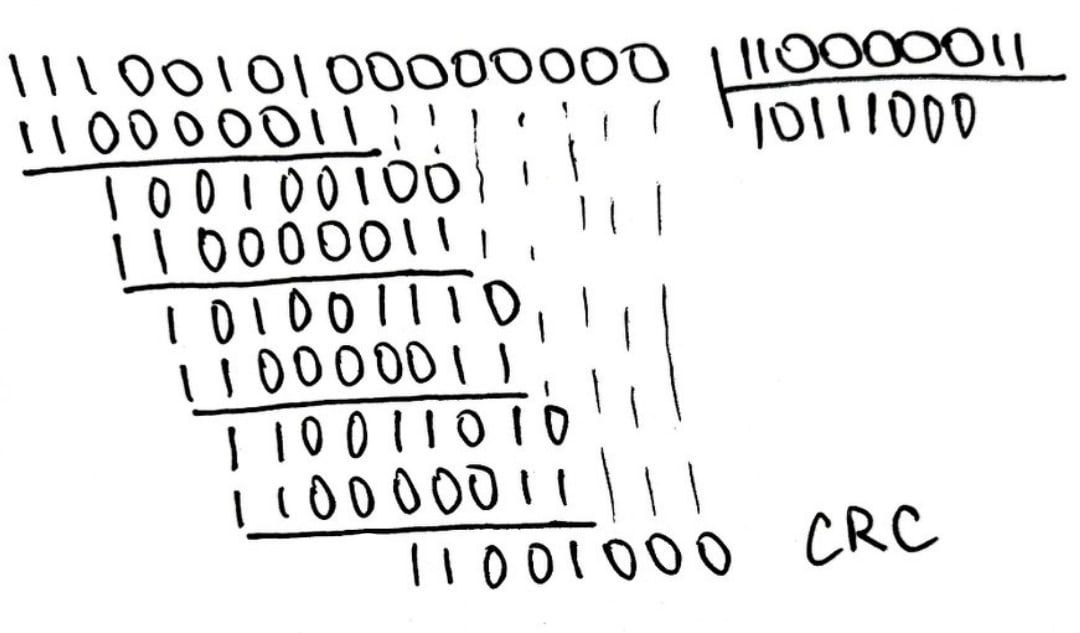


Рисунок 3 – CRC для полинома (8, 7, 1, 0)

Была разработана программа, реализующая расчет восьмиразрядной контрольной суммы по формуле С = Т % (М + 1). Была получена формула полинома-делителя CRC-32Q (x^32 + x^31 + x^24 + x^22 + x^16 + x^14 + x^8 + x^7 + x^5 + x^3 + x + 1) и разработана программа вычисления CRC с помощью правил полиномиального деления.

1. Тестирование и результаты работы программы

Был создан тестовый файл из 100 символов (рисунок 4) и подсчитано эталонное значение контрольной суммы для данных из файла (рисунок 5).

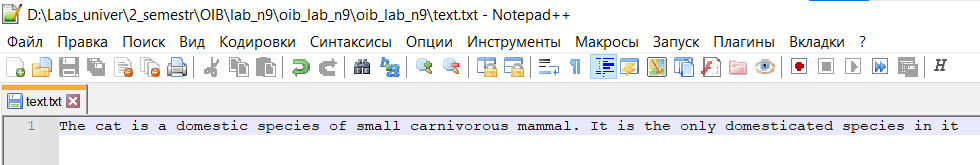


Рисунок 4 – Тестовый текстовый файл

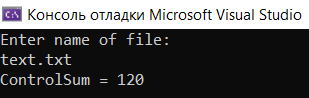


Рисунок 5 – Расчет контрольной суммы для текста из тестового файла

Далее один символ был заменен (рисунок 6). По таблице ASCII он больше предыдущего на единицу. Значит, контрольная сумма также должна увеличиться на единицу. Предположения оправдались (рисунок 7).

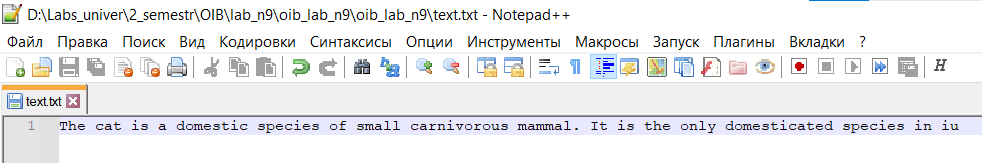


Рисунок 6 – Измененный файл

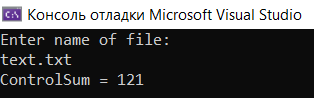


Рисунок 7 – Новая контрольная сумма

Далее текст файла был продублирован в его конец. Очевидно, контрольная сумма увеличилась вдвое (рисунок 8).

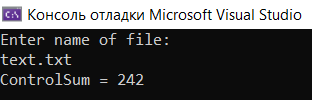


Рисунок 8 – Контрольная сумма продублированного текста

Далее в тексте были перемещены две буквы. Контрольная сумма не изменилась (рисунок 9).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Контрольная сумма текста, в котором две буквы поменяли местами

Была получена формула полинома-делителя CRC-32Q (x^32 + x^31 + x^24 + x^22 + x^16 + x^14 + x^8 + x^7 + x^5 + x^3 + x + 1) и разработана программа вычисления CRC с помощью правил полиномиального деления.

Был создан тестовый файл, содержащий текст «ABCD», и подсчитано эталонное значение CRC (рисунок 10).

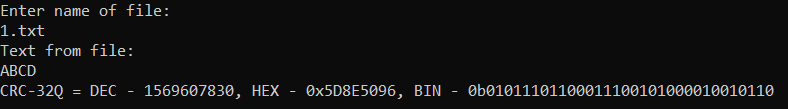


Рисунок 10 – CRC для тестового файла

Затем один символ в файле был заменен. Результат CRC можно увидеть на рисунке 11. Значение изменилось, значит, CRC позволяет обнаружить изменение текста.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – CRC для текста, в котором изменили одну букву

Затем текст был продублирован. Результат CRC можно увидеть на рисунке 12. Значение изменилось, значит, CRC позволяет обнаружить дублирование текста.

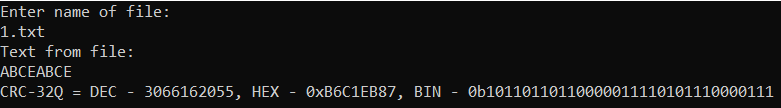


Рисунок 12 – CRC для продублированного текста

Затем были переставлены две буквы. Результат CRC можно увидеть на рисунке 13. Значение изменилось, значит, CRC позволяет обнаружить изменение текста.

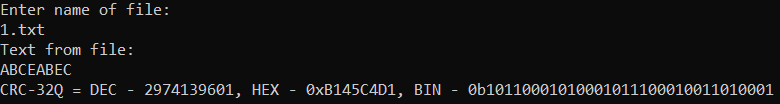


Рисунок 13 – CRC для текста, в котором поменяли местами две буквы

Был создан текстовый файл, содержащий текст «ABCD». С помощью плагина HEX-Editor в приложении Notepad++ файл был открыт в режиме HEX. В начало файла были записаны нули(рисунок 14). Затем был посчитан CRC для данного файла. Сравнивая результат с рисунком 10, можно заметить, что CRC не изменился.

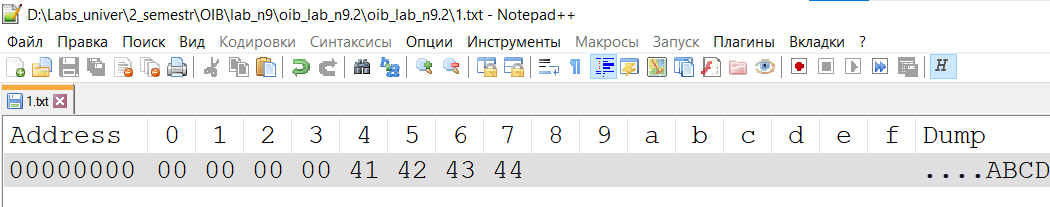


Рисунок 14 – Представление тестового файла в HEX

К концу контрольного текста в режиме HEX было дописано полученное значение CRC (рисунок 15) и снова подсчитано значение CRC (рисунок 16). Оно равно нулю, потому что остаток равен нулю. Значит, данные не были изменены.



Рисунок 15 – Содержание файла после добавления в конец значения CRC

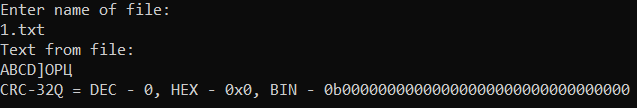


Рисунок 16 – Результат вычисления CRC для текста, в конец которого был дописан его CRC

Далее в конец полученного текста снова был дописан CRC (рисунок 17). В этот раз результат не равен нулю, ведь полученный текст уже совершенно другой (рисунок 18), не имеющий ничего общего с исходным файлом.

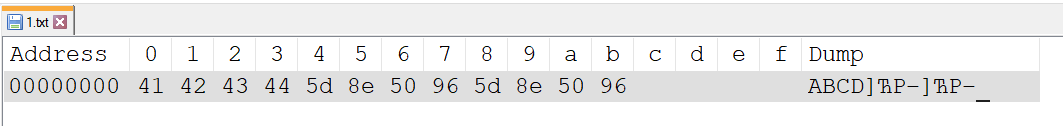


Рисунок 17 – Вид файла, в конец которого было продублировано значение CRC

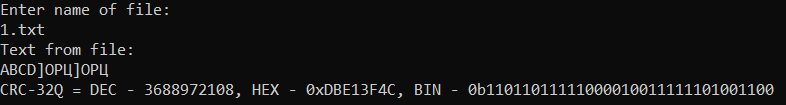


Рисунок 18 – Результат вычисления CRC при дублировании его в конец текста

В конец файла был записан байт 0x80 (рисунок 19). Значение CRC изменилось (рисунок 20).

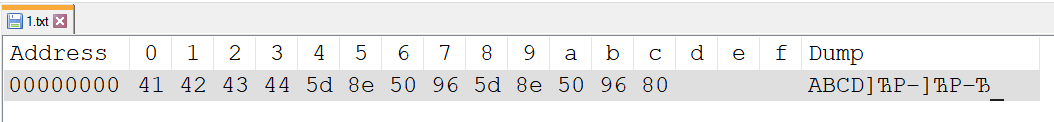


Рисунок 19 – Новый вид файла

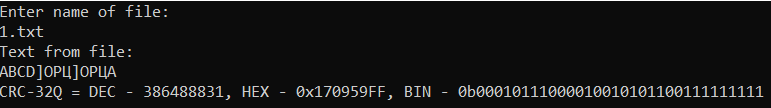


Рисунок 20 – Результат CRC

Для произвольного файла был посчитан CRC (рисунок 21). Затем в конец файла дважды было записано значение этого CRC. Получился новый CRC (рисунок 22).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 – CRC произвольного файла

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 – CRC произвольного файла, в конец которого дважды был записан CRC

Путем тестовых прогонов программы с использованием различных полиномов не были найдены полиномы, не показывающие достоверность информации.

1. Выводы

Была разработана программа, реализующая расчет восьмиразрядной контрольной суммы, разработана программа вычисления CRC с помощью правил полиномиального деления. В процессе работы были приобретены навыки разработки и использования программных средств контроля целостности информации с помощью методов контрольного суммирования, и были реализованы алгоритмы расчета контрольных сумм и циклических кодов. На практике было подтверждено, что метод CRC является более совершенным способом цифровой идентификации данных, чем метод контрольных сумм.

Список используемых источников

1. Зегжда П.Д. Основы информационной безопасности. Введение в профессиональную деятельность. Лабораторный практикум : учеб. пособие / П. Д. Зегжда, М. О. Калинин. – СПб. : ПОЛИТЕХ-ПРЕСС, 2019. – 112 с.
2. Циклический избыточный код [Электронный ресурс]. – Режим доступа:  https://ru.wikipedia.org/wiki/Циклический\_избыточный\_код, свободный.
3. Online CRC-32 Calculator [Электронный ресурс]. – Режим доступа:  https://crccalc.com/, свободный.
4. Вычисление проверок циклического резервирования [Электронный ресурс]. – Режим доступа:  https://translated.turbopages.org/proxy\_u/en-ru.ru.577e28be-62e99ba4-edccc830-74722d776562/https/en.wikipedia.org/wiki/Computation\_of\_crc, свободный.

ПРИЛОЖЕНИЕ А

Листинг программы «Расчет восьмиразрядной контрольной суммы»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CODE 255

void help(char\*);

char\* read\_file(char\*, char\*, int\*);

void file\_error();

int how\_many\_symbs\_in\_file(FILE\*);

int control\_sum(char\*, int);

int main()

{

char FileName[32] = {'\0'};

char\* textFromFile = { '\0' };

int count\_file\_symbs = 0;

int ControlSum = 0;

help(FileName);

textFromFile = read\_file(FileName, textFromFile, &count\_file\_symbs);

ControlSum = control\_sum(textFromFile, count\_file\_symbs);

printf("ControlSum = %d\n", ControlSum);

return 0;

}

void help(char\* FileName)

{

printf("Enter name of file:\n");

(void)scanf("%s", FileName);

}

void file\_error()

{

perror("Error: file not open");

exit(-1);

}

int how\_many\_symbs\_in\_file(FILE\* fin)

{

fseek(fin, 0x00, SEEK\_END);

int count = ftell(fin);

rewind(fin);

return count;

}

char\* read\_file(char\* FileName, char\* textFromFile, int\* count\_file\_symbs)

{

FILE\* fin = fopen(FileName, "rb");

if (fin == NULL)file\_error();

\*count\_file\_symbs = how\_many\_symbs\_in\_file(fin);

textFromFile = (char\*)calloc((\*count\_file\_symbs)+1, sizeof(char));

if(fin)fread(textFromFile, sizeof(char), \*count\_file\_symbs, fin);

if(fin)fclose(fin);

return textFromFile;

}

int control\_sum(char \*textFromFile, int count\_file\_symbs)

{

int C = 0, T = 0, M = MAX\_CODE;

for (int i = 0; i < count\_file\_symbs; i++)

{

T += textFromFile[i];

}

C = T % (M + 1);

return C;

}

ПРИЛОЖЕНИЕ B

Листинг программы «Вычисление CRC-32Q»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define POLYNOM 2168537515 // 1000 0001 0100 0001 0100 0001 1010 1011 - CRC-32Q

#define MAX\_CODE 256

#define BYTE 8

void help(char\*);

char\* read\_file(char\*, char\*, int\*);

void file\_error();

int how\_many\_symbs\_in\_file(FILE\*);

unsigned long crc32Q(char\*, int);

void bin(unsigned long, int);

int main()

{

char FileName[32] = { '\0' };

char\* textFromFile = { '\0' };

int count\_file\_symbs = 0;

help(FileName);

textFromFile = read\_file(FileName, textFromFile, &count\_file\_symbs);

//printf("Text from file:\n%s\n", textFromFile);

unsigned long CRC = crc32Q(textFromFile, count\_file\_symbs);

printf("CRC-32Q = DEC - %lu, HEX - 0x%X, ", CRC, CRC);

bin(CRC, sizeof(int));

return 0;

}

void help(char\* FileName)

{

printf("Enter name of file:\n");

(void)scanf("%s", FileName);

}

void file\_error()

{

perror("Error: file not open");

exit(-1);

}

int how\_many\_symbs\_in\_file(FILE\* fin)

{

fseek(fin, 0x00, SEEK\_END);

int count = ftell(fin);

rewind(fin);

return count;

}

char\* read\_file(char\* FileName, char\* textFromFile, int\* count\_file\_symbs)

{

FILE\* fin = fopen(FileName, "rb");

if (fin == NULL)file\_error();

\*count\_file\_symbs = how\_many\_symbs\_in\_file(fin);

textFromFile = (char\*)calloc((\*count\_file\_symbs) + 1, sizeof(char));

if(fin)fread(textFromFile, sizeof(char), \*count\_file\_symbs, fin);

if(fin)fclose(fin);

return textFromFile;

}

unsigned long crc32Q(char\* textFromFile, int count\_file\_symbs)

{

char\* buffer = textFromFile;

unsigned int CRC = 0;

while (count\_file\_symbs--)

{

CRC = CRC ^ (\*buffer << 24);

for (int i = 0; i < BYTE; i++)

{

CRC = CRC & 0x80000000 ? (CRC << 1) ^ POLYNOM : CRC << 1;

}

buffer++;

}

CRC = CRC ^ 0;

return (CRC);

}

void bin(unsigned long k, int size)

{

printf("BIN - 0b");

for (int i = size \* 8; i > 0; --i) {

printf("%d", (k & ((unsigned long)1 << (i - 1))) ? 1 : 0);

}

printf("\n");

}