Циклический избыточный код (CRC-8)

Выполнили: студенты группы 2652

Азизов С.З., Новиков А.И., Микаелян А.К.

Определение

- **Циклический избыточный код** (англ. *Cyclic redundancy check, CRC*) — алгоритм нахождения контрольной суммы, предназначенный для проверки целостности данных. CRC является практическим приложением помехоустойчивого кодирования, основанном на определенных математических свойствах циклического кода.

Помехоустойчивое кодирование

Первые попытки создания кодов с избыточной информацией начались задолго до появления современных ПК. К примеру, ещё в шестидесятых годах прошлого века Ридом и Соломоном была разработана эффективная методика кодирования — Код Рида-Соломона. Использование её в те времена не представлялось возможным, так как произвести операцию декодирования за разумное время первыми алгоритмами не удавалось. Точку в этом вопросе поставила фундаментальная работа Берлекампа, опубликованная в 1968 году. Эта методика, на практическое применение которой указал через год Мэсси, и по сей день используется в цифровых устройствах, обеспечивающих прием RS-кодированных данных. Более того: данная система позволяет не только определять позиции, но и исправлять неверные кодовые символы (чаще всего октеты).

Но далеко не везде от кода требуется коррекция ошибок. Многие современные каналы связи обладают приемлемыми характеристиками, и зачастую достаточно лишь проверить, успешно ли прошла передача или возникли *какие-нибудь* сложности; структура же ошибок и конкретные позиции неверных символов совершенно не интересуют принимающую сторону. И в этих условиях очень удачным решением оказались алгоритмы, использующие контрольные суммы. CRC как нельзя лучше подходит для подобных задач: невысокие затраты ресурсов, простота реализации и уже сформированный математический аппарат из теории линейных циклических кодов обеспечили ей огромную популярность.

Контрольная сумма

В самом общем своем виде контрольная сумма представляет собой некоторое значение, построенное по определенной схеме на основе кодируемого сообщения. Проверочная информация при **систематическом кодировании** дописывается в конец сообщения — после полезных данных. На принимающей стороне абонент знает алгоритм вычисления контрольной суммы: соответственно, программа имеет возможность проверить корректность принятых данных.

При передаче пакетов по реальному каналу, разумеется, могут возникнуть искажения исходной информации вследствие разных внешних воздействий: электрических наводок, плохих погодных условий и многих других. Сущность методики в том, что *при хороших характеристиках хэш-функции* в подавляющем числе случаев ошибка в сообщении приведет к изменению вычисленного на приеме значения CRC. Если исходная и вычисленная суммы не равны между собой, принимается решение о недостоверности принятых данных, и можно запросить повторную передачу пакета.

Описание процедуры

Из файла берется первое слово (байтовый элемент). Если старший бит в слове «1», то слово сдвигается влево на один разряд с последующим выполнением операции XOR. Соответственно, если старший бит в слове «0», то после сдвига операция XOR не выполняется. После сдвига теряется старый старший бит, а младший бит освобождается — его значение устанавливается равным нулю. На место младшего бита загружается очередной бит из файла, и операция повторяется до тех пор, пока не загрузится последний бит файла. После прохождения всего файла, в слове остается остаток, который и является контрольной суммой.

Код программы

```
#include "stdafx.h"
#include <stdint.h>
#include <iostream>
#define POLYNOMIAL 0xD8 // Определяем полином для вычисления

uint8_t crc(uint8_t const message) // Объявляем функцию с 1 входным параметром — каким-либо числом
{
	uint8_t remainder;	// Объявляем переменную остатка от деления (в ней же потом будет результат)

	remainder = message; // Присваиваем переменной остатка наши входные данные (т.к. в самом начале остатка ещё нет)

	// Для каждой битовой позиции входных данных совершаем:
	for (uint8_t bit = 8; bit > 0; --bit)
	{
		// Если старший бит остатка == 1,
		if (remainder & 0x80)
		{
			// То складываем по модулю 2 (XOR) остаток и полином, получаем новый остаток.
			remainder ^= POLYNOMIAL;
		}
		// Сдвигаем остаток на 1 бит влево, таким образом, помещая в конец новый бит исходных данных.
		remainder = (remainder << 1);
	}  // Соответственно, если старший бит == 0, то просто сразу сдвигаем, не производя XOR.

	// Возвращаем значащие биты — это и есть CRC.
	return (remainder);
}
void main(void)
{
	const char test[] = "123456789";
	uint8_t n = atoi(test);
	printf("The CRC of %s is %i", test, crc(n));
	getchar();
}
```