

实验五 CRC 检验算法实践

一、实验目的

1. 了解循环冗余检验（CRC）用于差错检测的基本原理，以及国际常用的 CRC 参数模型；
2. 掌握不同多项式和不同原始数据下循环冗余码的计算；
3. 掌握国际常用 CRC 参数模型的 C 语言实现。

二、实验设备与环境

1. 具备 C 语言集成开发环境的计算机；

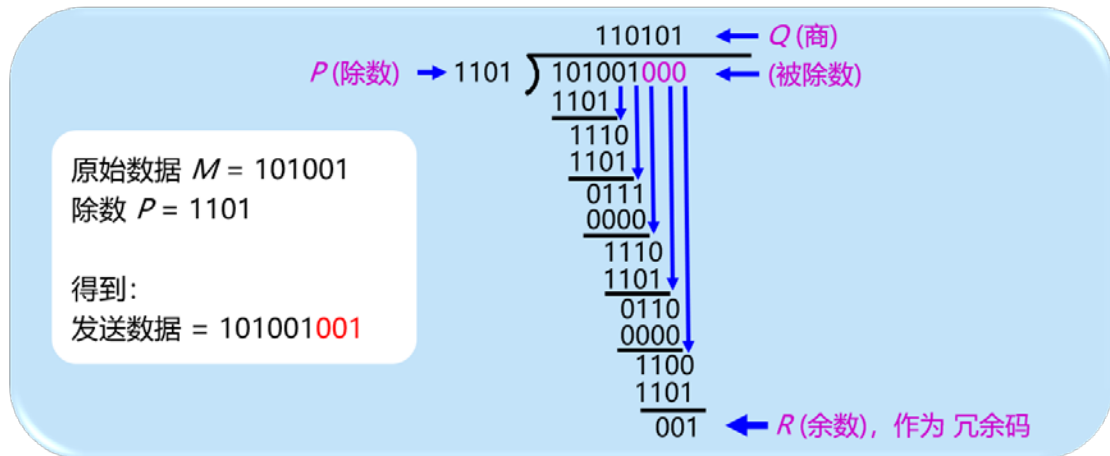
三、实验步骤

3.1 循环冗余检验

为了保证数据传输的可靠性，在计算机网络传输数据时，必须采用各种差错检测措施。目前在数据链路层广泛使用循环冗余检验（Cyclic Redundancy Check, CRC）的检错技术。其基本原理如下：

假定待发送的数据 M 有 k 个比特。CRC 运算就是在数据 M 的后面添加供差错检测的 n 位冗余码，然后构成一个 $(k+n)$ 位的数据发送。 n 位冗余码的计算方式如下：事先商定好长度为 $(n+1)$ 位的除数 P ，在 M 后面添加 n 个 0，得到的 $(k+n)$ 位的数除以除数 P （模二除法），得到余数是 R （ n 位，比 P 少一位）。这个余数 R 就作为冗余码拼接在数据 M 的后面发送出去。接收端把收到的数据除以同样的除数 P ，检查得到的余数 R ，如果余数 R 为 0 则判定无差错，如果余数不为 0 则判定有差错。

计算举例：设待发送数据 $M=101001$ ，除数 $P=1101$ ，则被除数是 101001000 ，进行模二除法：



一种较方便的方法是用多项式来表示除数，比如用 $P(X)=x^3+x^2+1$ 表示上面的除数 $P=1101$ 。

3.2 国际常用 CRC 参数模型

CRC 可以定义任意多项式、数据长度等，这里列出了一些国际常用的 CRC 参数模型表：

CRC 算法名称	多项式公式	宽度	多项式	初始值	结果异或值	输入值反转	输出值反转
CRC-4/ITU	$x^4 + x + 1$	4	03	00	00	true	true
CRC-5/EPC	$x^4 + x^3 + 1$	5	09	09	00	false	false
CRC-5/ITU	$x^5 + x^4 + x^2 + 1$	5	15	00	00	true	true
CRC-5/USB	$x^5 + x^2 + 1$	5	05	1F	1F	true	true
CRC-6/ITU	$x^6 + x + 1$	6	03	00	00	true	true
CRC-7/MMC	$x^7 + x^3 + 1$	7	09	00	00	false	false
CRC-8	$x^8 + x^2 + x + 1$	8	07	00	00	false	false
CRC-8/ITU	$x^8 + x^2 + x + 1$	8	07	00	55	false	false
CRC-8/ROHC	$x^8 + x^2 + x + 1$	8	07	FF	00	true	true
CRC-8/MAXIM	$x^8 + x^5 + x^4 + 1$	8	31	00	00	true	true
CRC-16/IBM	$x^{16} + x^{15} + x^2 + 1$	16	8005	0000	0000	true	true
CRC-16/MAXIM	$x^{16} + x^{15} + x^2 + 1$	16	8005	0000	FFFF	true	true
CRC-16/USB	$x^{16} + x^{15} + x^2 + 1$	16	8005	FFFF	FFFF	true	true
CRC-16/MODBUS	$x^{16} + x^{15} + x^2 + 1$	16	8005	FFFF	0000	true	true

CRC-16/CCITT	$x^{16} + x^{12} + x^5 + 1$	16	1021	0000	0000	true	true
CRC-16/CCITT-FALSE	$x^{16} + x^{12} + x^5 + 1$	16	1021	FFFF	0000	false	false
CRC-16/x5	$x^{16} + x^{12} + x^5 + 1$	16	1021	FFFF	FFFF	true	true
CRC-16/XMODEM	$x^{16} + x^{12} + x^5 + 1$	16	1021	0000	0000	false	false
CRC-16/DNP	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$	16	3D65	0000	FFFF	true	true
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	32	04C11DB7	FFFFFFFF	FFFFFFFF	true	true
CRC-32/BZIP2	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	32	04C11DB7	FFFFFFFF	FFFFFFFF	false	false
CRC-32/MPEG-2	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	32	04C11DB7	FFFFFFFF	00000000	false	false

一个完整的 CRC 参数模型应该包含以下信息: WIDTH, POLY, INIT, REFIN, REFOUT, XOROUT。

- NAME: 参数模型名称。
- WIDTH: 宽度, 即生成的 CRC 数据位宽, 如 CRC-8, 生成的 CRC 为 8 位。
- POLY: 十六进制多项式, 省略最高位 1, 如 $x^8 + x^2 + x + 1$, 二进制为 1 0000 0111, 省略最高位 1, 转换为十六进制为 0x07。
- INIT: CRC 初始值, 和 WIDTH 位宽一致。
- REFIN: true 或 false, 在进行计算之前, 原始数据是否翻转, 如原始数据: 0x34 = 0011 0100, 如果 REFIN 为 true, 进行翻转之后为 0010 1100 = 0x2c; 注意: 针对的每个字节, 而不是整个数据。
- REFOUT: true 或 false, 运算完成之后, 得到的 CRC 值是否进行翻转, 如计算得到的 CRC 值: 0x97 = 1001 0111, 如果 REFOUT 为 true, 进行翻转之后为 11101001 = 0xE9。注意: 这里做的逆序和 REFIN 不同, 它不是按字节逆序, 而是整个逆序。
- XOROUT: 计算结果与此参数进行异或运算后得到最终的 CRC 值, 和 WIDTH 位宽一致。

3.3 算法实现思路

通过对 CRC 的基本了解我们知道多项式的首位必定为 1，而这个 1 的位置在下一步计算一定为 0，所以就把前面这个 1 省略例如多项式 $x^8 + x^2 + x + 1$ 记为 0x07。下面以单字节输入的 CRC-8 算法为例：

- ① 将 CRC 寄存器（8-bits，比生成多项式少 1bit）赋初值 0
- ② 将原始数据与 CRC 寄存器异或，结果保存在 CRC 寄存器中
- ③ for 循环（ 8-bits ）
- ④ if (CRC 寄存器首位是 1)
- ⑤ 左移 1 位再异或
- ⑥ else(CRC 寄存器首位是 0)
- ⑦ 左移 1 位
- ⑧ end
- ⑨ CRC 寄存器就是我们所要求的余数。

3.4 CRC 算法 C 语言实践

利用 C 语言实现部分 CRC 算法，对于输入的数据进行 CRC 冗余码的计算。

四、实验要求

用 C 语言实现 CRC-8, CRC-16/XMODEM, CRC-16/CCITT 算法，根据个人能力由易到难实现，先实现单字节输入的 CRC-8 算法，再实现多字节输入的 CRC-8 算法，最后实现多字节输入的 CRC-16/XMODEM 算法和 CRC-16/CCITT 算法。

输入：十六进制单字节（简单）；十六进制多字节（中等）

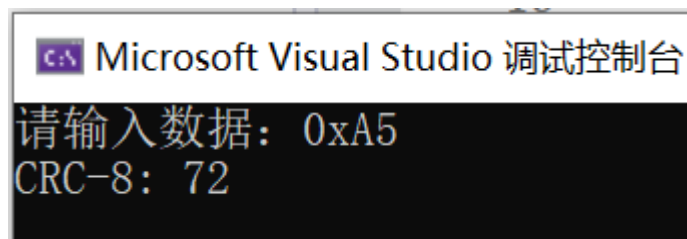
处理：1）CRC-8（中等）； 2）CRC-16/XMODEM（进阶）； 3）CRC-16/CCITT（进阶）

三种算法实现

输出：十六进制 CRC 冗余码

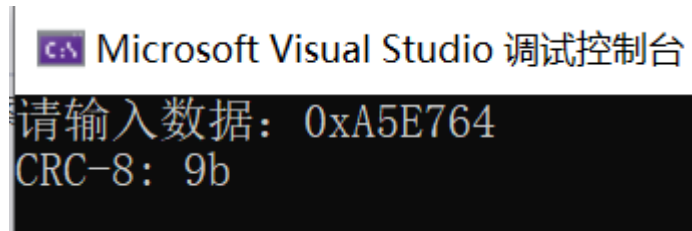
示例：

1) CRC-8 算法 C 语言实践（单字节输入）



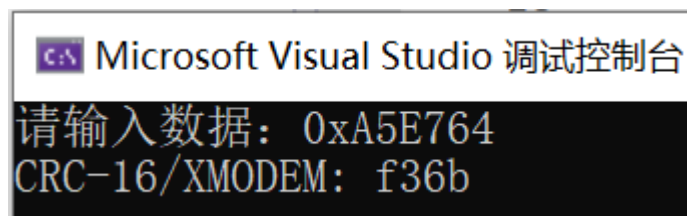
```
C:\> Microsoft Visual Studio 调试控制台
请输入数据: 0xA5
CRC-8: 72
```

2) CRC-8 算法 C 语言实践 (多字节输入)



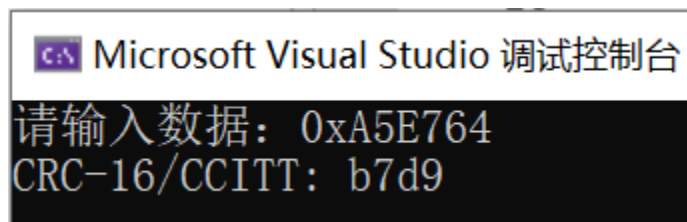
```
C:\> Microsoft Visual Studio 调试控制台
请输入数据: 0xA5E764
CRC-8: 9b
```

3) CRC-16/XMODEM 算法 C 语言实践 (多字节输入)



```
C:\> Microsoft Visual Studio 调试控制台
请输入数据: 0xA5E764
CRC-16/XMODEM: f36b
```

4) CRC-16/CCITT 算法 C 语言实践 (多字节输入)



```
C:\> Microsoft Visual Studio 调试控制台
请输入数据: 0xA5E764
CRC-16/CCITT: b7d9
```

实验报告要求:

1. 包含相关程序的完整代码;
2. 包含代码的详细注释, 解释说明相关语句的作用;
3. 包含多个示例数据的输入和输出结果截图。