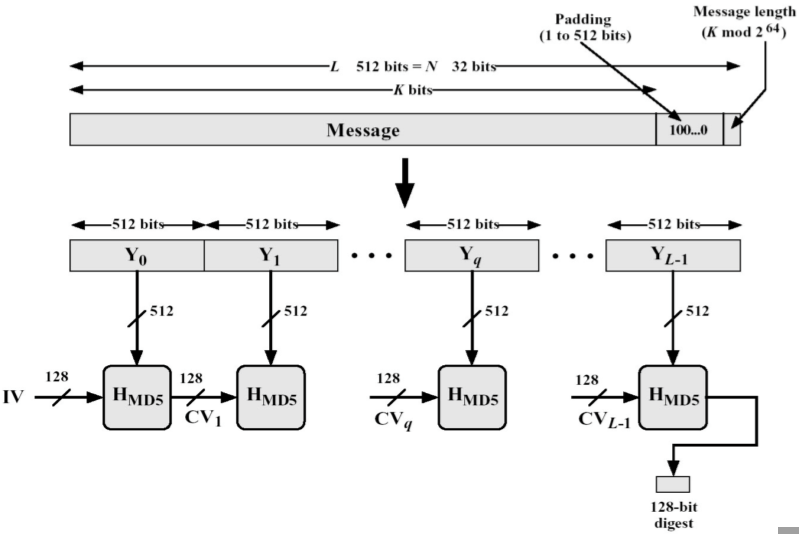
**Assignment2**

15331416 赵寒旭 数字媒体技术

1. **Design an executable MD5 program in C/C++ with some short samples.** 
   1. **MD5算法概述**

MD5信息摘要算法把输入的任意长度的信息计算生成128位长度输出结果。



1. **填充**

在原始消息数据尾部填充标识100…0，填充后的消息位数L448 (mod 512)。至少要填充1个位，所以标识长度1~512位。

 再向上述填充好的消息尾部附加原始消息的位数的低64位，最后得到一个长度L 是512位整数倍的消息。

即数据扩展至K\*512+448位，最后64位表示数据的原始长度b（添加填充位前的长度），若遇到b大于2^64（极少），只取低64位。

1. **分块**

把填充后的消息结果分割为L个512位的分组，结果表示为N个32位长的字。（N=L\*16）

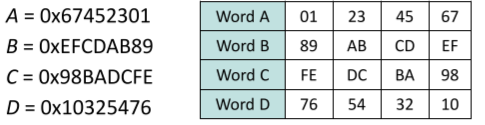
1. **初始化MD缓冲区**

初始化一个128位的MD缓冲区，也表示为4个32位长度的寄存器(A,B,C,D)。在MD缓冲区进行迭代，最后一步的128位输出即为算法结果。

ABCD分别是32位寄存器，初始化用十六进制数，采用小端存储。

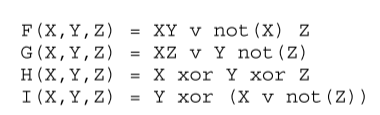
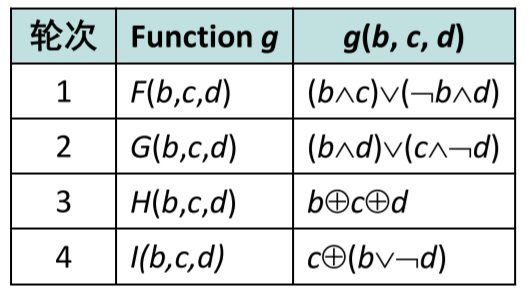
注意：此处小端意义：

一个字32位，一个字节8位，4个字节一组转化，在字节单位上，此处小端将低位字节排放在低地址端，高位字节排放在高地址端，在字节内部，并不做转换，保持原顺序存储。



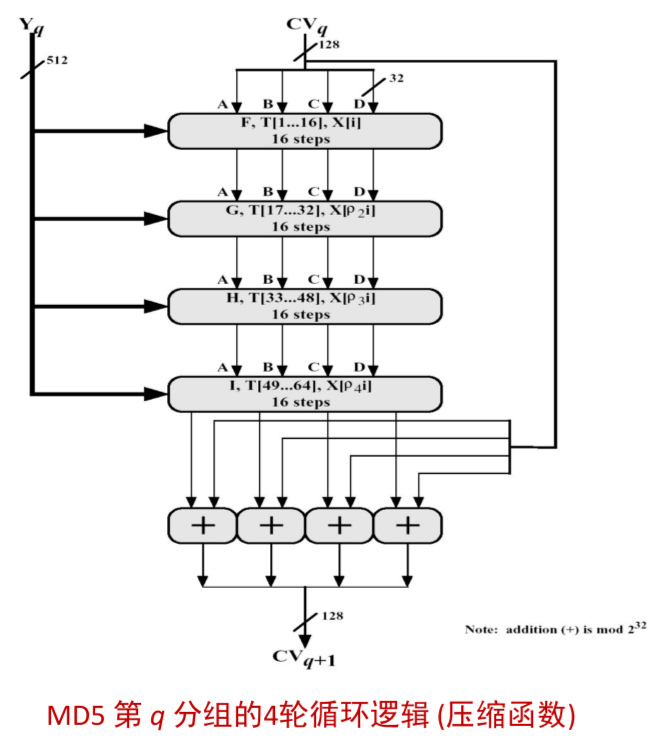
1. **处理16个32位字的数据**

以512位（16\*32）消息分组为单位，每一分组经过4个循环的压缩算法。

1. 定义4个辅助函数，每个函数输入三个32位长的字，输出一个32位长的字。

4轮循环中使用的生成函数 (轮函数) g 是一个 32位非线性逻辑函数，

在相应各轮的定义如右图。

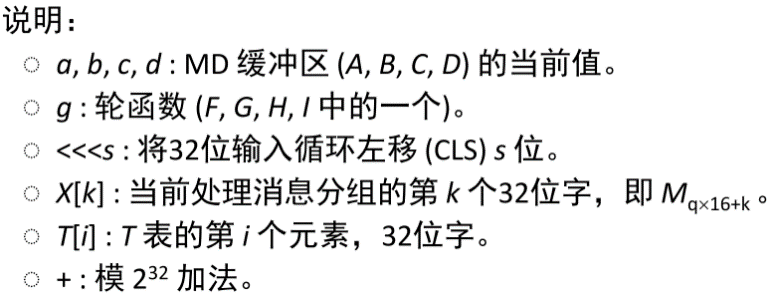
1. MD5压缩函数：循环迭代运算

MD5第q分组的4轮循环逻辑（压缩函数）如右图所示。

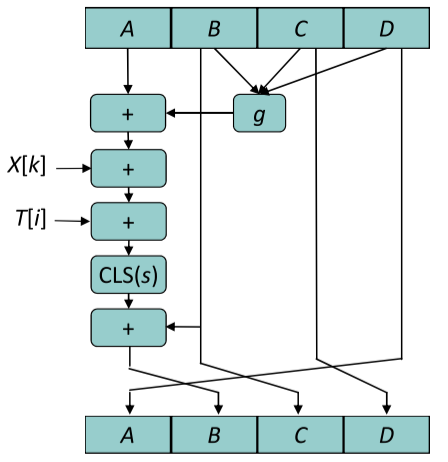
从CV输入128位，从 消息分组输入512位，完成 4轮循环后，输出128位， 用于下一轮输入的CV值。

每轮循环分别固定不同的 生成函数F, G, H, I，结合 指定的T表元素T[] 和消 息分组的不同部分X[] 做 16 次运算，生成下一轮 循环的输入。

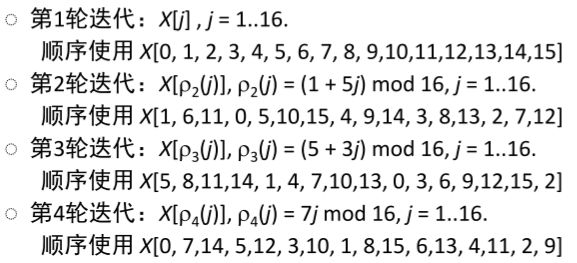
总共有64次迭代运算。

1. 每轮循环中的一步运算逻辑

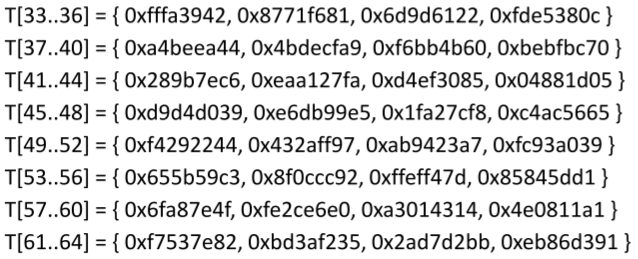
每轮循环中的一步运算逻辑：



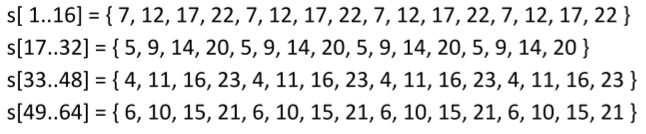
各轮迭代中X[k]之间的关系：



各次迭代运算采用的T值



各次迭代运算采用的左循环移位的s值



1. **输出结果**

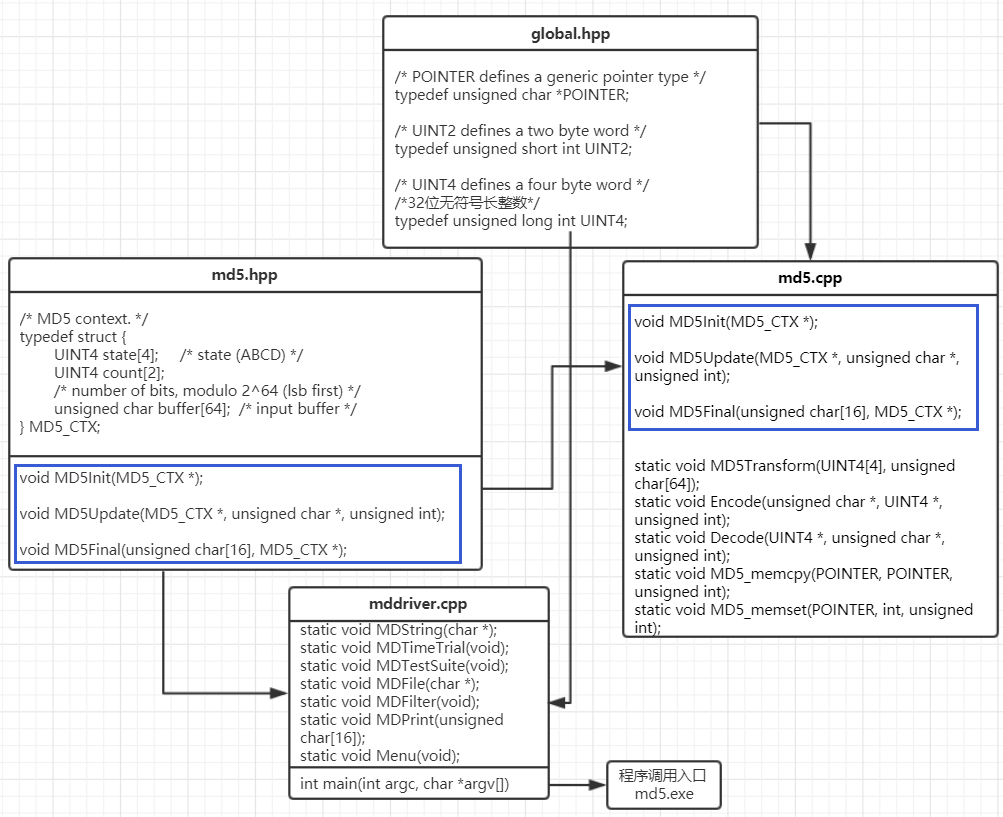
最终结果形式为：A，B，C，D，低位字节A开始，高位字节D结束。

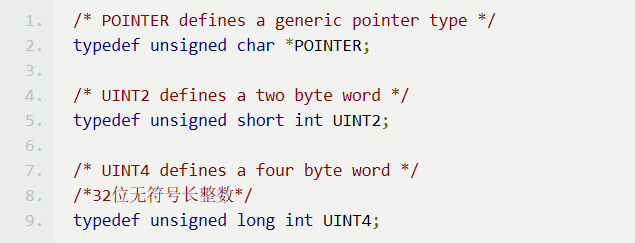
* 1. **C++算法实现**

此部分主要参考了<https://www.rfc-editor.org/pdfrfc/rfc1321.txt.pdf>资料

网址：ietf.org MD5(1992,RFC1321)

为方便显示代码块，将部分源代码转为markdown高亮显示截图。

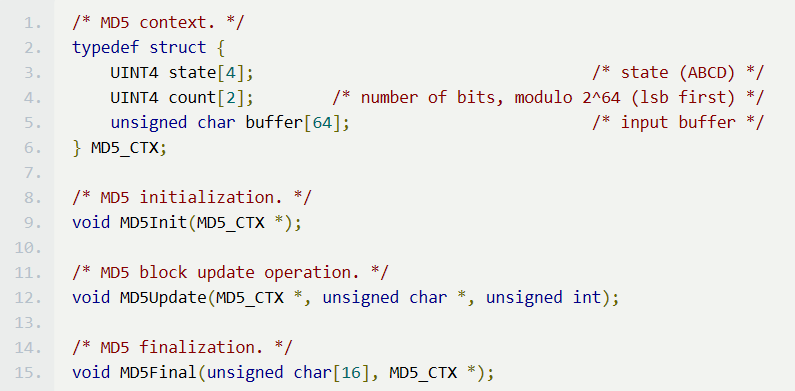
1. **程序文件结构**
2. **具体实现过程**
3. global.hpp 全局头文件



为复杂的声明定义在本程序中全局通用的较为简单的别名。

用无符号长整型变量存储寄存器中保存的值。

1. md5.hpp



state[4]对应4个32位寄存器A,B,C,D

count[2]是形成分组的累计位数（一边分组压缩，一遍累计长度），共64位

buffer输入缓冲区

1. md5.cpp

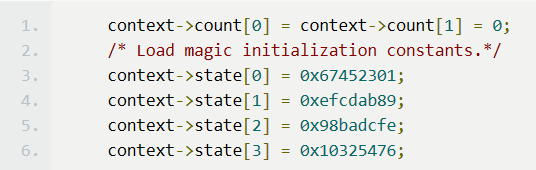
主要功能：实现md5.hpp中定义的三个函数

1. MD5Init:：MD5初始化

寄存器（A，B，C，D）置16进制初值作为初始向量，采用小端存储。

count置0。





1. MD5Update：MD5的主要计算过程

传入输入缓冲区和输入数据长度，进行MD5操作，更新context的值



1. MD5Final：结束MD5操作，得到MD5输出结果，context归零。



1. MD5Transform

MD5基本变换，用于在MD5Update时改变寄存器的值。



1. Encode

存储的无符号长整型数转为4个字节，以位处理对象数据



1. Decode

4个字节转一个无符号长整型数



1. MD5\_memcpy

按len长度把input中的值逐位赋值给output，用于MD5Update中context的更新等操作。



1. MD5\_memset

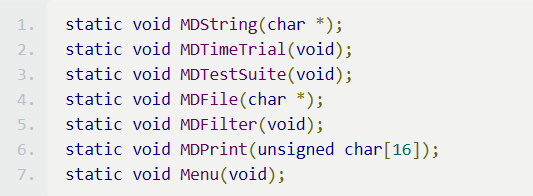
把output前len位置为value值。

用于在MD5Final中把cntext值置0。



1. mddriver.cpp

主要函数声明如下：

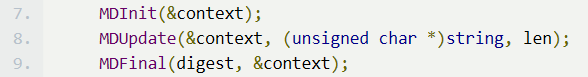


1. MDString

将一个字符串string经过MD5操作得到摘要digest的值，打印结果



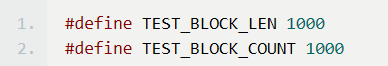
调用md5.hpp中声明的函数，得到digest结果



1. MDTimeTrial

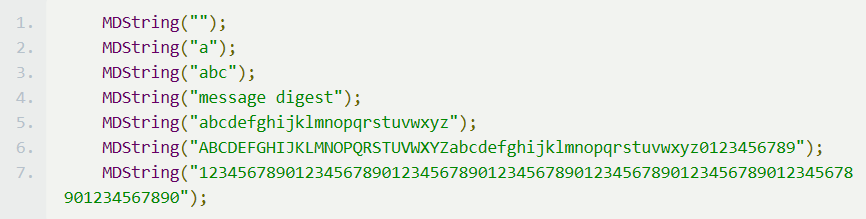
运行时间测量。

计算对测试长度数据做MD5需要的时间。



1. MDTestSuite

固定样本测试函数



1. MDFile

对指定文件进行MD5操作得到其digest值，打印结果



1. MDFilter

计算标准输入的digest，打印结果



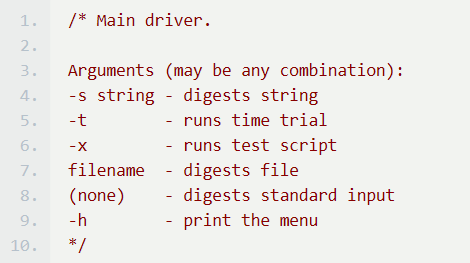
1. MDPrint

以16进制打印digest



1. Menu

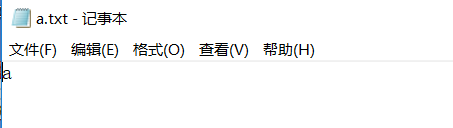
在命令行显示功能菜单



* 1. **程序运行截图**

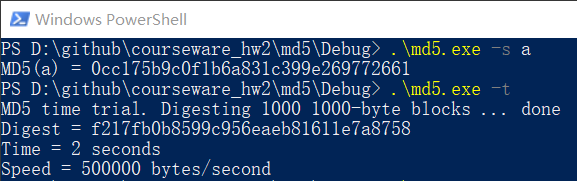
按照menu功能顺序依次测试

注：文件a.txt事先放入Debug文件夹，内容为a。

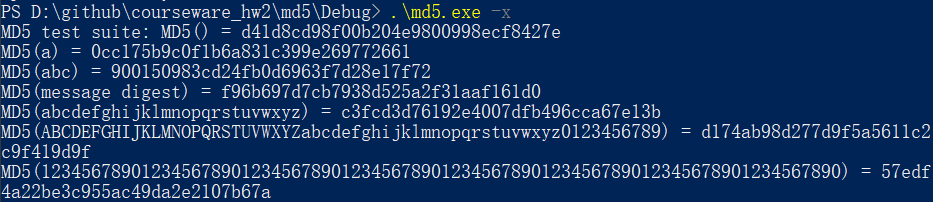


命令行字符串输入进行MD5转换：-s a

时间测试：-t



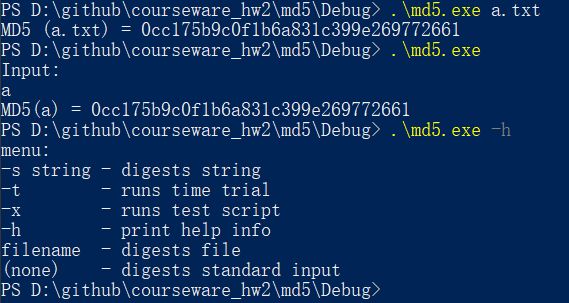
固定样例测试结果：-x



文件MD5转换：a.txt

空命令行后续参数直接调用MD5Filter()

显示帮助：-h



1. **How MD5 works for password protection**

**2.1 MD5算法实现**

MD5算法的具体过程可以参考1.1中的算法描述部分，已经做出详细解释。

**2.2 MD5实现密码保护的工作原理**

(1) 不可逆性

MD5是一个安全的散列算法，输入两个不同的明文不会得到相同的输出值，根据输出值，不能得到原始的明文，即其过程不可逆。

MD5在计算过程中丢失了原文的信息，且理论上有小概率使一个MD5的结果对应多个明文。（发生碰撞）

1. 安全性

MD5是输入不定长度信息，输出固定长度128-bits的算法。经过程序流程，生成四个32位数据，最后联合起来成为一个128-bits散列。基本方式为，求余、取余、调整长度、与链接变量进行循环运算。得出结果。

应用于密码保护时：用户的密码不能使用明文存入到数据库中，中间要经过一定的算法，对密码进行转换成暗文，MD5是很好的一种选择，MD5是一种不可逆的字符串变换算法，我们在使用的时候，将用户密码转换成MD5值存入到数据库中，当用户登录是同样对密码进行运算得到MD5值，再将该值与数据库中得MD5值进行比较，若相同则登录成功，否则失败。这样系统管理员不能从数据库中的MD5值获得用户的原密码值，保障了密码的安全性。