MD5算法原理

MD5消息摘要算法，属Hash算法一类。MD5算法对输入任意长度的消息进行运行，产生一个128位的消息摘要。

具体实现可参考博客 https://blog.csdn.net/sinat_27933301/article/details/79538169

和官方标准RFC1321 https://tools.ietf.org/html/rfc1321

算法实现

一些常量的定义

```
//定义循环右移函数
#define RPTATE_SHIFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

//定义F,G,H,I函数
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

//定义寄存器word A,B,C,D

#define A 0x67452301
#define B 0xefcdab89
#define C 0x98badcfe
#define D 0x10325476
//strBaye的长度
unsigned int strlength = 0;
//A,B,C,D的临时变量
int tempA = 0, tempB = 0, tempC = 0, tempD = 0;
//定义k数组，用于压缩函数
const unsigned int k[] = {
        0xd76aa478,0xe8c7b756,0x242070db,0xc1bdceee,0xf57c0faf,0x4787c62a,0xa83046
13,0xfd469501,
        0x698098d8,0x8b44f7af,0xffff5bb1,0x895cd7be,0x6b901122,0xfd987193,0xa67943
8e,0x49b40821,
        0xf61e2562,0xc040b340,0x265e5a51,0xe9b6c7aa,0xd62f105d,0x02441453,0xd8a1e6
81,0xe7d3fbc8,
        0x21e1cde6,0xc33707d6,0xf4d50d87,0x455a14ed,0xa9e3e905,0xfcefa3f8,0x676f02
d9,0x8d2a4c8a,
        0xfffa3942,0x8771f681,0x6d9d6122,0xfde5380c,0xa4beea44,0x4bdecfa9,0xf6bb4b
60,0xbebfbc70,
        0x289b7ec6,0xeaa127fa,0xd4ef3085,0x04881d05,0xd9d4d039,0xe6db99e5,0x1fa27c
f8,0xc4ac5665,
        0xf4292244,0x432aff97,0xab9423a7,0xfc93a039,0x655b59c3,0x8f0ccc92,0xffeff4
7d,0x85845dd1,
```

```
        0x6fa87e4f,0xfe2ce6e0,0xa3014314,0x4e0811a1,0xf7537e82,0xbd3af235,0x2ad7d2
bb,0xeb86d391 };

//用数组存储向左位移数，方便操作
//每一行 表示一轮的左位移数 ， 根据 RFC 1321的标准参数来做
const unsigned int s[] = { 7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22,
5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20,
4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23,
6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21 };

//用于转换16进制
const char str16[] = "0123456789abcdef";
```

这些定义包括标准里的 A,B,C,D word， F,G,H,I函数， 用于压缩参数的常量值， 这些先定义下来，方便后面写程序，防止出错

```
/* Round 1. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
[ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
[ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
[ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

/* Round 2. */
/* Let [abcd k s i] denote the operation
     a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
[ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
[ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
[ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

/* Round 3. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
[ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
[ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
[ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

/* Round 4. */
/* Let [abcd k s t] denote the operation
     a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
[ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
[ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
[ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

/* Then perform the following additions. (That is increment each
```


主要函数getMD5Code

此暗示通过传入一个字符串，返回对此字符串进行MD4处理得到的摘要字符串ik， 这是一个主要的流程。

首先初始化最终的变量A，B，C，D， 然后进行填充操作， 进行压缩操作， 最后转换为16进制哈希值。

```
string getMD5Code(string source) {
        //初始化

        tempA = A;
        tempB = B;
        tempC = C;
        tempD = D;

        //把string变成二进制， 同时附加填充位
        unsigned int *strByte = padding(source);

        //   对于i = 0到N / 16-1 将块i复制到X.
        //     对于j = 0到15做
        //       将X [j]设置为M [i * 16 + j]。
        //   进行压缩函数操作

        for (int i = 0; i<strlength / 16; i++) {

                unsigned int num[16];

                for (int j = 0; j<16; j++) {
                        num[j] = strByte[i * 16 + j];
                }

                MD5compress(num);
        }
        //把得到的摘要2进制变成16进制字符串输出
        return
changeToHex(tempA).append(changeToHex(tempB)).append(changeToHex(tempC)).append(ch
angeToHex(tempD));
}
```

填充函数padding

填充函数 处理后应满足bits≡448(mod512), 填充方式为先加一个1,其它位补零,最后加上64位的原来长度

```
unsigned int* padding(string str) {
        //以512位,64个字节为一组， num表示组数， 利用整数相除直接得到组数

        unsigned int num = ((str.length() + 8) / 64) + 1;

        //对于一组需要16个整数来存储 16*4=64， strByte表示此字符串的2进制表示（这里用
int数组表示）
        unsigned int *strByte = new unsigned int[num * 16];

        //初始化字符串的长度（长度是16*组数）
        strlength = num * 16;

        //初始化 strByte数组
```

```
        for (unsigned int i = 0; i < num * 16; i++) {
                strByte[i] = 0;
        }
        //一个整数存储四个字节，一个unsigned int对应4个字节，保存4个字符信息

        for (unsigned int i = 0; i <str.length(); i++) {
                strByte[i / 4] |= (str[i]) << ((i % 4) * 8);
        }
        //尾部添加1 一个unsigned int保存4个字符信息,所以用128左移
        strByte[str.length() / 4] |= 0x80 << (((str.length() % 4)) * 8);
        /*
        *添加原长度，长度指位的长度，所以要乘8，然后是小端序，所以放在倒数第二个,这里长度
只用了32位
        */
        strByte[num * 16 - 2] = str.length() * 8;
        return strByte;
}
```

压缩函数

按照RFC 1321的标准 进行一共64次压缩运算， 因为每一次的参数是固定的， 所以可以提前输入参数， 这里我是k数组来存储这些参数

```
UINT4 state[4];
unsigned char block[64];
{
  UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

  Decode (x, block, 64);

  /* Round 1 */
  FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
  FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
  FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
  FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
  FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
  FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
  FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
  FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
  FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
  FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
  FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
  FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
  FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
  FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
  FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
  FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

 /* Round 2 */
  GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
  GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
  GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
```



```
//MD5 压缩函数 Hmd5
void MD5compress(unsigned int M[]) {
        int f = 0, g = 0;
        int a = tempA, b = tempB, c = tempC, d = tempD;

        /*第1轮迭代:
```

```
        *X[j] , j = 1..16.
        *第2轮迭代:
        *X[p2(j)], p2(j) = (1 + 5j) mod 16, j = 1..16.
        *第3轮迭代:
        *X[p3(j)], p3(j) = (5 + 3j) mod 16, j = 1..16.
        *第2轮迭代:
        *X[p4(j)], p4(j) = 7j mod 16, j = 1..16*/

        for (int i = 0; i < 64; i++) {

                if (i<16) {
                        f = F(b, c, d);
                        g = i;
                }
                else if (i<32) {
                        f = G(b, c, d);
                        g = (5 * i + 1) % 16;
                }
                else if (i<48) {
                        f = H(b, c, d);
                        g = (3 * i + 5) % 16;
                }
                else {
                        f = I(b, c, d);
                        g = (7 * i) % 16;
                }
                //每次循环使用相同的迭代逻辑和 4*16 次运算的预设参数表, 也就是前面的k表
                unsigned int tmp = d;
                d = c;
                c = b;
                b = b + RPTATE_SHIFT((a + f + k[i] + M[g]), s[i]);
                a = tmp;

        }
        tempA = a + tempA;
        tempB = b + tempB;
        tempC = c + tempC;
        tempD = d + tempD;
}
```

进制转换函数

把数字变为2进制 以字符串输出

```
string changeToHex(int num) {
        int b;
        string tmp;
        string str = "";

        for (int i = 0; i<4; i++) {
```

```
                tmp = "";
                b = ((num >> i * 8) % (1 << 8)) & 0xff;   //逆序处理每个字节
                for (int j = 0; j < 2; j++) {
                        tmp.insert(0, 1, str16[b % 16]);
                        b = b / 16;
                }
                str += tmp;
        }
        return str;
}
```

测试函数

验证函数，通过RFC 1321给的标准输入输出来判断

```
    The MD5 test suite (driver option "-x") should print the following
    results:

MD5 test suite:
MD5 ("") = d41d8cd98f00b204e9800998ecf8427e
MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661
MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72
MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") =
d174ab98d277d9f5a5611c2c9f419d9f
MD5 ("12345678901234567890123456789012345678901234567890123456789012345
78901234567890") = 57edf4a22be3c955ac49da2e2107b67a
```



```
int main() {

        //这是 RFC 1321的标准测试输入和输出，  用来验证此MD5算法的正确性
        string input[7] = {
                "",
                "a",
                "abc",
                "message digest",
                "abcdefghijklmnopqrstuvwxyz",
                "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",

"12345678901234567890123456789012345678901234567890123456789012345678901234567890"
};

    string expect[7] = {
        "d41d8cd98f00b204e9800998ecf8427e", "0cc175b9c0f1b6a831c399e269772661",
        "900150983cd24fb0d6963f7d28e17f72", "f96b697d7cb7938d525a2f31aaf161d0",
        "c3fcd3d76192e4007dfb496cca67e13b", "d174ab98d277d9f5a5611c2c9f419d9f",
        "57edf4a22be3c955ac49da2e2107b67a"};

    for (int i = 0; i < 7; i++) {
        cout << "--------------------------------" << endl;
        cout << "测试 " << i + 1 << ":" << endl;
        cout << "原消息:          " << input[i] << endl;
```

```
            cout << "MD5标准输出:      " << expect[i] << endl;
            string digest = getMD5Code(input[i]);
            cout << "MD5输出:          " << digest << endl;
        }
    }
```

测试结果

可看到和标准输出是相同的， 证明此MD5算法正确。

```
--------------------------------
测试 1:
原消息:
MD5标准输出:      d41d8cd98f00b204e9800998ecf8427e
MD5输出:          d41d8cd98f00b204e9800998ecf8427e
--------------------------------
测试 2:
原消息:           a
MD5标准输出:      0cc175b9c0f1b6a831c399e269772661
MD5输出:          0cc175b9c0f1b6a831c399e269772661
--------------------------------
测试 3:
原消息:           abc
MD5标准输出:      900150983cd24fb0d6963f7d28e17f72
MD5输出:          900150983cd24fb0d6963f7d28e17f72
--------------------------------
测试 4:
原消息:           message digest
MD5标准输出:      f96b697d7cb7938d525a2f31aaf161d0
MD5输出:          f96b697d7cb7938d525a2f31aaf161d0
--------------------------------
测试 5:
原消息:           abcdefghijklmnopqrstuvwxyz
MD5标准输出:      c3fcd3d76192e4007dfb496cca67e13b
MD5输出:          c3fcd3d76192e4007dfb496cca67e13b
--------------------------------
测试 6:
原消息:           ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
MD5标准输出:      d174ab98d277d9f5a5611c2c9f419d9f
MD5输出:          d174ab98d277d9f5a5611c2c9f419d9f
--------------------------------
测试 7:
原消息:           12345678901234567890123456789012345678901234567890123456789012345678901234567890
MD5标准输出:      57edf4a22be3c955ac49da2e2107b67a
MD5输出:          57edf4a22be3c955ac49da2e2107b67a
```


源码传送门

https://github.com/wangjiwu/implement-MD5-in-C-

参考文档

- MD5加密算法原理及实现 https://www.cnblogs.com/hjgods/p/3998570.html
- The MD5 Message-Digest Algorithm https://tools.ietf.org/html/rfc1321