```
#pragma once
#include <iostream>
#include <bitset>
#include <sstream>
#include <string.h>
using namespace std;
namespace zy6{

#ifdef __cplusplus
        extern "C" {
#endif
                static const unsigned int zy6_BLOCK_SIZE = (384)/8;
                typedef struct zy3
                {
                        unsigned char c : 6;
                }zy3;

                typedef struct z4{
                        unsigned char c;
                }z4;
                typedef struct z2{
                        unsigned char l : 4;
                        unsigned char h : 4;
                }z2;
                union zy42{
                        z4 z4;
                        z2 z2;
                };
                typedef unsigned char uint8;
                typedef unsigned int uint32;

                typedef struct zy6_status
                {
                        unsigned char m_zy_c_64[zy6_BLOCK_SIZE*4/3];
                        unsigned int m_len;
                        unsigned char m_block[2 * zy6_BLOCK_SIZE];
                }zy6_status;
                void final(zy6_status& z_st);
                void Init_zy6(zy6_status& z_st);
                void transform(zy6_status& m_zy6_st,const unsigned char *message, unsigned int
block_nb);
                void update(zy6_status& z_st, const unsigned char *message, unsigned int len);
#ifdef __cplusplus
        }
#endif
        std::string Tostring64(zy6_status &m_zy6_st);
        std::string Tostring64HEX(zy6_status &m_zy6_st);
        std::wstring Tostring64BG(zy6_status &m_zy6_st);
        std::string Zy6String(const std::string &str);
        std::string Zy6StringHEX(const std::string &str);
};
/*
zy6消息摘要算法
作者:zhangliuxue
描述:
1、获取信息384位位基本块，换成64个6bit表示
2、定义运算的路径，64个6bit表示
3、定义结果的路径，64个6bit表示
```

```
4、两次散列。基于路径
5、结果最后一次调和，基于路径
*/
#include "zy6_block.h"
#include "utf8.h"
#ifdef UTF8
#include <string.h>
#endif
namespace zy6{

#ifdef __cplusplus
       extern "C" {
#endif
              //初始变换序列,路径，可以替换
              unsigned char m_zy64[] = { \
                     63, 0, 34, 17, 58, 23, 16, 2, \
                     59, 55, 56, 7, 47, 61, 8, 4, \
                     38, 25, 48, 3, 37, 41, 1, 32, \
                     39, 57, 33, 30, 18, 45, 14, 28, \
                     15, 60, 5, 40, 43, 53, 10, 20, \
                     49, 35, 62, 31, 6, 24, 22, 26, \
                     46, 29, 36, 9, 11, 52, 44, 13, \
                     27, 54, 19, 50, 51, 12, 42, 21 };

              unsigned char* To64(const unsigned char* c, unsigned char* d)
              {
                     for (char i = 0; i < zy6_BLOCK_SIZE/3; i++)
                     {
                            unsigned char t = std::move(i * 4);
                            unsigned char m = std::move(i * 3);
                            d[t] = c[m] >> 2;
                            d[t + 1] = std::move((c[m] & 0x3) << 4) + (c[m + 1] >> 4);
                            d[t + 2] = std::move((c[m + 1] & 0xf) << 2) + (c[m + 2] >> 6);
                            d[t + 3] = c[m + 2] & 0x3f;
                     }
                     return std::move(d);
              }
              void Init_zy6(zy6_status& z_st)
              {
                     z_st.m_len = 0;

                     for (char i = 0; i < 64; i++)
                     {
                            z_st.m_zy_c_64[i] = m_zy64[i];
                     }
                     memset(z_st.m_block, 0, sizeof(z_st.m_block));
              }
              void update(zy6_status& z_st, const unsigned char *message, unsigned int len)
              {
                     unsigned int block_nb;
                     unsigned int new_len, rem_len, tmp_len;
                     const unsigned char *shifted_message;
                     tmp_len = zy6_BLOCK_SIZE - z_st.m_len;
                     rem_len = len < tmp_len ? len : tmp_len;
                     memcpy(&z_st.m_block[z_st.m_len], message, rem_len);
                     if (z_st.m_len + len < zy6_BLOCK_SIZE) {
                            z_st.m_len += rem_len;
                            return;
```

```
			}
			new_len = len - rem_len;
			block_nb = (int)(new_len / zy6_BLOCK_SIZE);
			shifted_message = (const unsigned char *)( message + rem_len);
			transform(z_st, z_st.m_block, 1);
			transform(z_st, shifted_message, block_nb);
			rem_len = new_len % zy6_BLOCK_SIZE;
			memcpy(z_st.m_block, &shifted_message[block_nb * zy6_BLOCK_SIZE], rem_len);
			z_st.m_len = rem_len;
		}
		void final(zy6_status& z_st)
		{
			memset((void *)(z_st.m_block + z_st.m_len), 0, zy6_BLOCK_SIZE * 2 -
z_st.m_len);
			z_st.m_block[z_st.m_len] = 0x80;
			transform(z_st, z_st.m_block, 1);
			unsigned char pt = 0;
			for (unsigned char i = 0; i < 64; i++)
			{
					z_st.m_zy_c_64[pt] = std::move((z_st.m_zy_c_64[i] +
z_st.m_zy_c_64[pt]) % 64);
					pt = m_zy64[pt];
					z_st.m_zy_c_64[pt] = std::move((z_st.m_zy_c_64[i] +
z_st.m_zy_c_64[pt]) % 64);
					pt = m_zy64[pt];
					z_st.m_zy_c_64[pt] = std::move((z_st.m_zy_c_64[i] +
z_st.m_zy_c_64[pt]) % 64);
					pt = m_zy64[pt];
					z_st.m_zy_c_64[pt] = std::move((z_st.m_zy_c_64[i] +
z_st.m_zy_c_64[pt]) % 64);
					pt = m_zy64[pt];
			}
		}
		void transform(zy6_status& z_st, const unsigned char *message, unsigned int block_nb)
		{
			const unsigned char *sub_block;
			unsigned int i;
			unsigned char pt = 0;
			unsigned char d[(zy6_BLOCK_SIZE * 4) / 3];
			for (i = 0; i < block_nb; i++)
			{
				sub_block = (const unsigned char *)(message + (i * zy6_BLOCK_SIZE));
				//组成64卦
				To64(sub_block, d);
				unsigned char pt = 0;
				for (unsigned int ti = 0; ti < (zy6_BLOCK_SIZE * 4) / 3; ti++)
				{
					if (d[ti]>0)
					{
						pt = ti;
						{
							z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + d[ti]) % 64);
							pt = m_zy64[pt];
							z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + d[ti]) % 64);
							pt = m_zy64[pt];
							z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + d[ti]) % 64);
                                                                pt = m_zy64[pt];
                                                                z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + d[ti]) % 64);

                                                        }
                                                        pt = d[ti];
                                                        {
                                                                z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64);
                                                                pt = m_zy64[pt];
                                                                z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64);
                                                                pt = m_zy64[pt];
                                                                z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64);
                                                                pt = m_zy64[pt];
                                                                z_st.m_zy_c_64[pt] =
std::move((z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64);

                                                        }
                                                }
                                        }
                                }
                        }
#ifdef __cplusplus
        }
#endif
        std::string Tostring64(zy6_status &m_zy6_st)
        {
                static const char * letters =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ=-";
                char ss[65];
                ss[64] = 0;
                for (char i = 0; i < 64; i++){
                        ss[i] = letters[m_zy6_st.m_zy_c_64[i]];
                }
                return std::move(string(ss));

        }
        std::string Tostring64HEX(zy6_status &m_zy6_st)
        {
                static const char * letters = "0123456789abcdef";
                zy42 zy42;
                char buf[97];
                buf[96]=0;
                //4个6bit转换为3个8bit
                for (char i = 0; i < 16; i = i + 1)
                {
                        char k = std::move(i * 4);
                        char k1 = std::move(i * 6);
                        zy42.z4.c = std::move((m_zy6_st.m_zy_c_64[k] << 2) + (m_zy6_st.m_zy_c_64[k +
1] >> 4));
                        buf[k1] = letters[zy42.z2.h];
                        buf[k1 + 1] = letters[zy42.z2.l];
                        zy42.z4.c = std::move(((m_zy6_st.m_zy_c_64[k + 1] & 0xf) << 4) +
(m_zy6_st.m_zy_c_64[k + 2] >> 2));
                        buf[k1 + 2] = letters[zy42.z2.h];
                        buf[k1 + 3] = letters[zy42.z2.l];
                        zy42.z4.c = std::move(((m_zy6_st.m_zy_c_64[k + 2] & 0x3) << 6) +
m_zy6_st.m_zy_c_64[k + 3]);
                        buf[k1 + 4] = letters[zy42.z2.h];
                        buf[k1 + 5] = letters[zy42.z2.l];
                }
                return std::move(string(buf));
        };
        std::wstring Tostring64BG(zy6_status &m_zy6_st)
        {
                static const wchar_t * letters =
L"䷀䷁䷂䷃䷄䷅䷆䷇䷈䷉䷊䷋䷌䷍䷎䷏䷐䷑䷒䷓䷔䷕䷖䷗䷘䷙䷚䷛䷜䷝䷞䷟䷠䷡䷢䷣䷤䷥䷦䷧䷨䷩䷪䷫䷬䷭䷮䷯䷰䷱䷲䷳䷴䷵䷶䷷䷸䷹䷺䷻䷼䷽䷾䷿";
                wstringstream ss;
                for (int i = 0; i < 64; i++){
                        ss << letters[m_zy6_st.m_zy_c_64[i]];
                }
                return ss.str();
        };

        std::string Zy6String(const std::string &str)
        {
                zy6::zy6_status z_st;
                Init_zy6(z_st);
                update(z_st, (unsigned char *)str.c_str(), str.length());
                final(z_st);
                return std::move(Tostring64(z_st));
        }

        std::string Zy6StringHEX(const std::string &str)
        {
                zy6::zy6_status z_st;
                Init_zy6(z_st);
                update(z_st, (unsigned char *)str.c_str(), str.length());
                final(z_st);
                return std::move(Tostring64HEX(z_st));
        }
};
```