源码：flcode.h

```
/*
模块：分离码

功能：
分离码是一种信息编解码技术，主要利用数学的不同进制转换来形成，结合码表和数学的进制转
换，提出码位分离的编解码方法。
版权：
上海泥娃通信科技有限公司
email: zhangliuxue@126.com
*/
#pragma once
#include "math.h"
#include "malloc.h"
#include "string.h"
#include "iostream"
#include "sstream"
#include "string"
#include "map"
#include "vector"
using namespace std;

//注意：当字符串为空时，也会返回一个空字符串
void split(const std::wstring& s, std::wstring& delim, std::vector< std::wstring >*
ret)
{
    size_t last = 0;
    size_t index = s.find_first_of(delim, last);
    while (index != std::string::npos)
    {
        ret->push_back(s.substr(last, index - last));
        last = index + 1;
        index = s.find_first_of(delim, last);
    }
    if (index - last>0)
    {
        ret->push_back(s.substr(last, index - last));
    }
}
//进制转换结构描述：jz表示多少进制，letters表示进制的字符串    ，iCodeTable表示字符代表
的数
typedef struct JzStatus
{
    int jz;
    wstring letters;
    std::map<wchar_t, int> iCodeTable;
}JzStatus;
union i_c
{
    unsigned int iVal;
    unsigned char cVal[sizeof(unsigned int)];
};
#define uint32    unsigned int
#define BigtoLittle32(A) ((( (uint32)(A) & 0xff000000) >> 24) | \
    (((uint32)(A)& 0x00ff0000) >> 8) | \
    (((uint32)(A)& 0x0000ff00) << 8) | \
    (((uint32)(A)& 0x000000ff) << 24))
bool IsBigEndian()
{
    union NUM
    {
        int a;
        char b;
    }num;
    num.a = 0x1234;
    if (num.b == 0x12)
    {
        return true;
    }
    return false;
}
//初始化
void InitJz(JzStatus & st)
{
    if (st.letters.empty())
        st.letters =
L"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+-";

    st.jz = st.letters.size();
    for (int i = 0; i < st.jz; i++)
    {
        st.iCodeTable[st.letters.c_str()[i]] = i;
    }
}
//获取字符串表示的整数
unsigned int GetJzInt(JzStatus & st, const wchar_t * cValue, size_t iL)
{
    unsigned int i = 0;
    for (size_t k = 0; k<iL; k++)
    {
        i = i + st.iCodeTable[cValue[k]] * (unsigned int)pow((double)st.jz, (int)k);
    }
    return i;
}
//获取整数表示的字符串，并给出字符串的长度
wstring  GetJzChar(JzStatus & st, unsigned int iValue, int & k)
{
    wstringstream ws;
    unsigned int i = iValue;
    while (i != 0)
    {
        i = iValue / st.jz;
        ws << st.letters.c_str()[iValue % st.jz];
        if (i > 0)
        {
            k++;
        }
        iValue = i;
    }
    k++;
    return ws.str();
}
/*
编码:    输入消息，转换为字符串序列和位数序列
*/
wstring EncodeJz(JzStatus & st, const char* msg, size_t len, std::vector<unsigned
char>&iSer)
{
    int i = len / sizeof(unsigned int);
    int j = len % sizeof(unsigned int);
    wstring outstr;
    for (int k = 0; k < i; k++)
    {
        int p = 0;
        memcpy(&p, (void*)&msg[k * sizeof(int)], sizeof(int));
        int m = 0;
        wstring tmp = GetJzChar(st, p, m);
        outstr.append(tmp);
        iSer.emplace_back(m);
    }
    if (j>0)
    {
        int p = 0;
        memcpy(&p, (void*)&msg[i * sizeof(int)], j);
        int m = 0;
        wstring tmp = GetJzChar(st, p, m);
        outstr.append(tmp);
        iSer.emplace_back(m);
    }
    return std::move(outstr);
}
/*
解码
*/
vector<unsigned char> DecodeJz(JzStatus & st, const wstring &msg, const size_t& len,
const std::vector<unsigned char>&iSer)
{
    vector<unsigned char> p1;
    size_t offset = 0;
    const wchar_t * tp = msg.c_str();
    int po = 0;
    i_c ic;
    for (auto &x : iSer)
    {
        unsigned int m = GetJzInt(st, &tp[offset], x);
        ic.iVal = m;
        for (int h = 0; h < sizeof(int);h++)
            p1.emplace_back(ic.cVal[h]);
        offset += x;
    }
    return std::move(p1);
}
/*
编码：空格分割
*/
wstring EncodeJzByspace(JzStatus & st, const char* msg, size_t len)
{
    //cout << "Int size:" << sizeof(unsigned int) << endl;

    int i = len / sizeof(unsigned int);
    int j = len % sizeof(unsigned int);
    wstring outstr;
    for (int k = 0; k < i; k++)
    {
        int p = 0;
        memcpy(&p, (void*)&msg[k * sizeof(int)], sizeof(int));
        int m = 0;
        wstring tmp = GetJzChar(st, p, m);
        outstr.append(tmp);
        if (k != i - 1)
            outstr.append(L" ");
    }
    if (j>0)
    {
        int p = 0;
        outstr.append(L" ");
        memcpy(&p, (void*)&msg[i * sizeof(int)], j);
        int m = 0;
        wstring tmp = GetJzChar(st, p, m);
        outstr.append(tmp);
    }
    return std::move(outstr);
}
/*
解码：空格分割
*/
vector<unsigned char> DecodeJzByspace(JzStatus & st, const wstring &msg, const size_t&
len)
{
    vector<unsigned char> p1;
    const wchar_t * tp = msg.c_str();
    int po = 0;
    i_c ic;
    std::vector< std::wstring > strvec;
    wstring  delim = L" ";
    split(msg, delim, &strvec);
    for (auto &x : strvec)
    {
        unsigned int m = GetJzInt(st, x.c_str(), x.length());
        ic.iVal = m;
        for (int h = 0; h < sizeof(int); h++)
            p1.emplace_back(ic.cVal[h]);
    }
    return std::move(p1);
}
string DecodeJzByspace_s(JzStatus & st, const wstring &msg, const size_t& len)
{
    const wchar_t * tp = msg.c_str();
    int po = 0;
    i_c ic;
    std::vector< std::wstring > strvec;
    wstring  delim = L" ";
    split(msg, delim, &strvec);
    stringstream ss;
    for (auto &x : strvec)
    {
        unsigned int m = GetJzInt(st, x.c_str(), x.length());
        ic.iVal = m;
        for (int h = 0; h < sizeof(int); h++)
        {
            ss << ic.cVal[h];
        }
    }
    return std::move(ss.str());
}
```