分离编解码 C 算法使用手册

上海泥娃通信科技有限公司

目录

第一章 分离码算法介绍

利用数学不同进制之间的转换结合变换的码表，实现信息编解码，包括：信息的码表单元；信息的编码单元；信息的解码单元。

实现文档分解成码表、变换序列和位数系列三个部分，或者采用默认码表的变换序列和位数序列两部分；本发明还实现通过码表，变换序列和位数序列还原文档的方法。

基于上述目的不同进制之间转换形成码和位分离编解码的方法包括：

制定码表：确定处理信息的单元位数，确定转换的进制，定义码表；

编码：根据要求读取 64 位（或者 128 位，或者其它）赋值给整数，然后根据要求转换成相应的进制（对应的数字用码表表示的字符表示），转换结果记录到变换序列，转换后的位数记录到位数序列，一直持续到转换完毕，最后形成两个部分。变换序列的字符一定是码表的字符，位数序列主要记载转换单元对应在变换记录中的长度。

解码：读取位数信息，按位数读取相关的字符，查找码表变换成相应的数字，结合原有的进制定义，转换为整数，存入到文件中，一直到转换完毕，得到相关的文件。

有益效果在于：

用于信息的多路存储和传输，不再借助于密钥来保证信息的安全，信息的存储分为三个部分：进制定义和码表、十进制到给定进制之间的转换结果、转换后结果的位数。并且可以针对不同的要求，实现给定进制和码表的信息存储和传输，用于网络之间的多路通信和多路存储；实现自定义进制和码表的信息存储，用于特定场合的传输和保存。

功能描述：

1、信息安全算法，码表对应加密的密钥，利用数学运算的特性实现信息的加密；

2、密文全文搜索：利用加密信息和原文信息的前缀一致性，结合语义树全文索引算法实现密文全文检索。

本算法获得国家发明专利：**《一种分离编解码的方法》,专利号:ZL 2016 1 0238974.2**

第二章 分离码的实现和应用

算法采用 C 语言实现，主要分为初始化，编码和解码，以及分离码表示组成。

1.1. 源代码

```
/*
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());
    }
```

1.2. 函数和结构说明

1.2.1.JzStatus 结构

```
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;
```

1.2.2.InitJz：初始化

```
输入：JzStatus & st，码表结构体
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;
    }
}
```

1.2.3.GetJzInt：获取字符串表示的整数

输入：st，码表结构体；cValue，字符串转换；il 字符串的长度
输出：返回整数值

```
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;
}
```

1.2.4.GetJzChar：获取整数表示字符串，并给出字符串长度

输入：st，码表结构体；iValue，字符串转换，k 返回字符串长度

```
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();
}
```

1.2.5.EncodeJz：编码输入消息，转换为字符串序列和位数序列

输入：st，码表结构体；msg，需要编码的信息；len，信息长度；iSer，编码位序列
输出：编码后字符序列

```
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);
}
```

1.2.6.DecodeJz：解码

输入：st，码表结构体；msg，编码的字符串序列；len 字符串长度；iSer 编码后位序列
输出：返回解码后信息

```
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);
}
```

1.2.7.EncodeJzByspace：编码，空格分割

输入：st，码表；msg，需要编码的信息；len，需要编码信息的长度
输出：编码后的结果，用空格分隔

```
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);
}
```

1.2.8.DecodeJzByspace：解码，空格分割

输入：st，码表结构；msg 以空格分割编码的字符串；len，保留
输出：字符串向量

```
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);
}
```

1.2.9.DecodeJzByspace_s：解码空格编码的字符串

输入：st，码表结构；msg 以空格分割编码的字符串；len，保留
输出：字符串

```
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());
}
```