书面作业 1 参考答案

习题 8

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    char str[100] = "BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD"; //密文
    int len = strlen(str);
    for (int i=0; i<26; ++i){
        for (int j=0; j<len; ++j)
            putchar( (str[j]-'A'+i) %26 + 'A' );//每个字母移位
        putchar('\n');
    }
}
```

Python

```python
ciphertext = "BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD" # 密文字符串
for i in range(1, 26): # 对 1-25 穷举破译
    tem_ans = "" # 存放转换后的字符串
    for ch in ciphertext:
        tem_ans += chr(((ord(ch)-ord('a')+i)%26) + ord('a')) # 字符移位
print(tem_ans) # 输出转换后的字符串
```

习题 9

2 阶 Hill 密码

C++

```
#include <bits/stdc++.h>
using namespace std;

int mat[2][2],nymat[2][2],tmp[100],res[100],det;
char str[100];

int find_ny(int x){
    for (int i=1; i<26; ++i)
        if ((x*i)%26 == 1)
            return i;
}

int transform_ch_to_num(char c){
    if (c>='a') return c-'a';
    return c-'A';
}

char transform_num_to_ch(int c){
    return c+'A';
}

int main(){
    cout<<"请输入 2*2 的在 Z26 上可逆的矩阵"<<endl;
    cin >> mat[0][0] >> mat[0][1] >> mat[1][0] >> mat[1][1];

    //计算逆矩阵 nymat
    ( det = (mat[0][0]*mat[1][1] - mat[0][1]*mat[1][0]) % 26 + 26 ) %=
26;
    det = find_ny(det);
    nymat[0][0] = (mat[1][1]*det)%26;
    nymat[0][1] = (26 - (mat[0][1]*det)%26)%26;
    nymat[1][0] = (26 - (mat[1][0]*det)%26)%26;
    nymat[1][1] = (mat[0][0]*det)%26;
    int op;
    while(1){
        cout<<"输入操作模式，加密请输入 0，解密请输入 1，退出输入 2"<<endl;
        cin>>op;
        if (op == 0){
            cout<<"输入明文"<<endl;
            scanf("%s",str);
            int len = strlen(str), flag = 0;
            if (len&1){
                str[len] = 'x';
                ++len;
                flag = 1;
            }
            for (int i=0; i<len; ++i){
                tmp[i] = transform_ch_to_num(str[i]);
                tmp[i+1] = transform_ch_to_num(str[i+1]);
                res[i] = (tmp[i]*mat[0][0] + tmp[i+1]*mat[1][0])%26;
                res[i+1] = (tmp[i]*mat[0][1] + tmp[i+1]*mat[1][1])%26;
            }
            len -= flag;
            cout<<"加密后密文为："<<endl;
            for (int i=0; i<len; ++i)
                putchar(transform_num_to_ch(res[i]));
            putchar('\n');
        }
        else if(op == 1){
            cout<<"输入密文"<<endl;
            scanf("%s",str);
            int len = strlen(str), flag = 0;
            if (len&1){
                str[len] = 'x';
                ++len;
                flag = 1;
            }
            for (int i=0; i<len; ++i){
                tmp[i] = transform_ch_to_num(str[i]);
                tmp[i+1] = transform_ch_to_num(str[i+1]);
                res[i] = (tmp[i]*nymat[0][0] + tmp[i+1]*nymat[1][0])%26;
                res[i+1] = (tmp[i]*nymat[0][1] +
tmp[i+1]*nymat[1][1])%26;
            }
            len -= flag;
            cout<<"解密后明文为："<<endl;
            for (int i=0; i<len; ++i)
```

```
                putchar(transform_num_to_ch(res[i]));
            putchar('\n');

        }
        else break;
    }
}
```

Python

```
import numpy as np

class Hill_encode:
    '''
    编码
    '''
    def __init__(self) -> None:
        self.__K = np.array([[11,8],[3,7]]) # Z26 上的可逆矩阵 K

    def __call__(self, plaintext : str) -> str:
        '''
        编码函数调用
        '''
        assert len(plaintext)%2 == 0

        packets = [plaintext[i:i+2] for i in range(0, len(plaintext)-1,
2)] # 将原始明文中的字母两两一组分组
        ciphertext = ""
        for packet in packets: # 分别对每一组进行线性变换
            X = []
            X.append(ord(packet[0]) - ord('a'))
            X.append(ord(packet[1]) - ord('a'))
            X = np.array(X)

            # 线性变换
            tem = np.matmul(X, self.__K)
            tem = tem%26

            Y = []
            Y.append(chr(tem[0]+ord('a')))
            Y.append(chr(tem[1]+ord('a')))
            ciphertext += ''.join(Y)
```

```python
        return ciphertext

class Hill_decode:
    '''
    解码
    '''
    def __init__(self) -> None:
        self.__rK = np.array([[7,18],[23,11]]) # 对应的逆矩阵

    def __call__(self, ciphertext : str) -> str:
        '''
        解码函数调用
        '''
        assert len(ciphertext)%2 == 0

        packets = [ciphertext[i:i+2] for i in range(0, len(ciphertext)-
1, 2)] # 将密文中的字母两两一组分组
        plaintext = ""
        for packet in packets: # 分别对每一组进行线性变换
            Y = []
            Y.append(ord(packet[0]) - ord('a'))
            Y.append(ord(packet[1]) - ord('a'))
            Y = np.array(Y)

            # 线性变换
            tem = np.matmul(Y, self.__rK)
            tem = tem%26

            X = []
            X.append(chr(tem[0]+ord('a')))
            X.append(chr(tem[1]+ord('a')))
            plaintext += ''.join(X)
        return plaintext


# 测试数据，每个字符串的长度保证为 2 的倍数
test_plaintext = ["hill", "dude", "what", "is", "love", "over"]

H_E = Hill_encode()
H_D = Hill_decode()

# 测试
print("原始数据：")
print(str(test_plaintext))
```

```

list_ciphertext = []
for text in test_plaintext:
    list_ciphertext.append(H_E(text))
print("加密后的密文数据：")
print(str(list_ciphertext))

list_plaintext = []
for text in list_ciphertext:
    list_plaintext.append(H_D(text))
print("对密文进行解密后的数据：")
print(str(list_plaintext))
```