上海理工大学光电信息与计算机工程学院

《信息安全》实验报告

专　　业	计算机科学与技术
学生姓名	许耀永
学　　号	1712480131
年　　级	2017 级
指导教师	刘亚
成　　绩：	
教师签字：	

报告格式要求

1、 正文字体中文为宋体，五号，行距为固定值 18 磅，西文为 Times New Rome, 五号，行距为固定值 18 磅。

2、 章节标题为加粗宋体，小四号，段前段后各 0.5 行，行距为固定值 18 磅。

3、 打印时需双面打印。

实验一　凯撒密码

一、 实验目的

实现凯撒密码的加解密
其中加密的明文使用小写字母，解密的密文用大写字母

二、程序的数据结构设计和算法流程图或算法描述

(1)在数学形式上，将字母映射成数字
(2)选择一个整数密钥 K(K 在 1 到 25 之间)
(3)加密时，依据“加 K 模 26”，将原字母由 L 转换成(L + K) % 26
(4)解密时，依据“减 K 模 26”，将密文字母由 L 转换成(L - K) % 26

三、程序代码

//caesarCipher.h

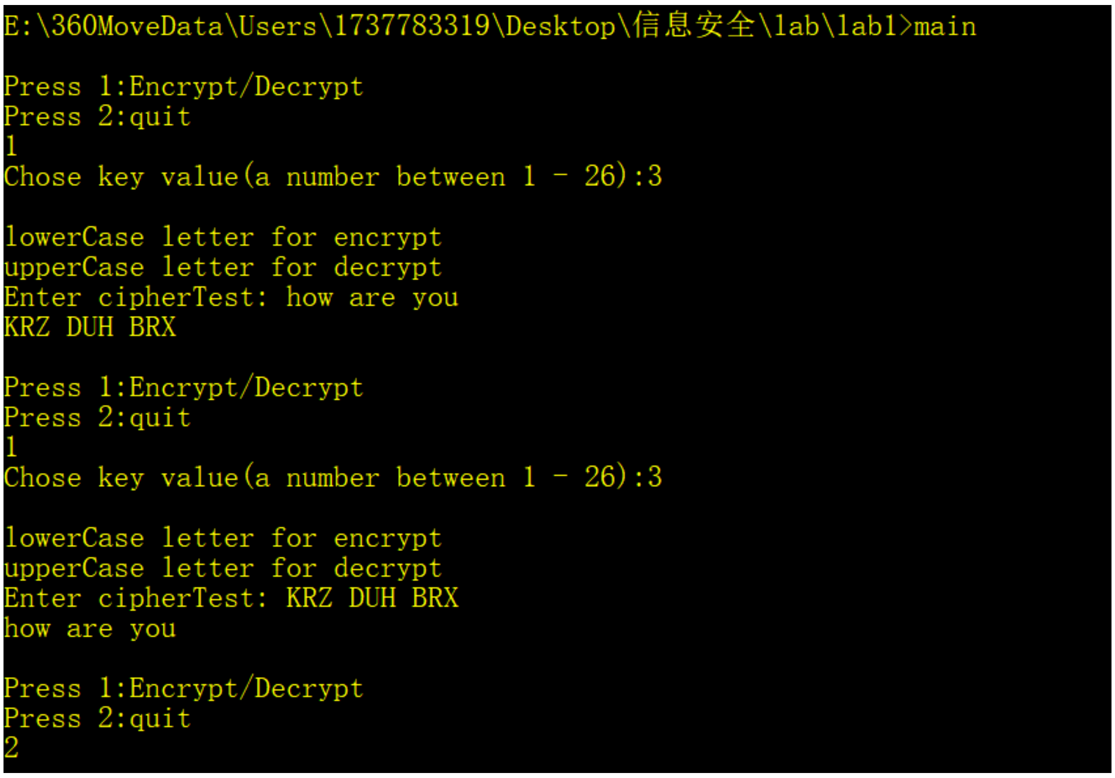
```
#include<stdio.h>
#include<ctype.h>

char caesar(char c,int k){
	if(isalpha(c) && c!= toupper(c) ){
		c = toupper(c);

		c = ( ( (c - 65) + k ) % 26 ) + 65; //encrypt
	}else if(isalpha(c) && c == toupper(c) ){
		c = ( ( (c - 65) - k + 26) % 26 ) + 65; //decrypt

		c = tolower(c);
	}
	return c;
}
```

//testCaser.cpp

```
#include"caesarCipher.h"
#include<stdlib.h>
#include<iostream>
#include<string>

using namespace std;

int main(){
	string input,output;
	int choice = 0;
	while(choice != 2){
		cout << endl << "Press 1:Encrypt/Decrypt" << endl << "Press 2:quit "<< endl;

		try{
			cin >> choice;

			if(choice != 1 && choice != 2){
				throw "Incorrect Choice";
			}
		}catch(const char* chc){
			cerr << "Incorrect Choice "<<endl;
			return 1;
		}

		if(choice == 1){
			int key;

			try{
				cout << "Chose key value(a number between 1 - 26):";

				cin >> key;

				cin.ignore();

				if(key < 1 || key > 26){
					throw "Incorrect key";
				}
			}catch(const char* K){
				cerr << "Incorrect key value chosen "<<endl;
				return 1;
			}



			try{
				cout << endl << "lowerCase letter for encrypt"<<endl <<"upperCase letter for
decrypt" << endl;

				cout << "Enter cipherTest: ";
				getline(cin,input);

				for(int i = 0;i < input.size(); i++){
```

```
                    //if(!(input[i] >= 'a' && input[i] <= 'z') && !(input[i] >= 'A' && input[i] <= 'Z')) throw "Incorrecr String";

                }
            }catch(const char* str){
                cerr << "may have some digit or special symbols "<<endl;
                cerr << "put only alphabets "<<endl;
                return 1;
            }

            for(unsigned int x = 0;x < input.length();x++){
                output += caesar(input[x],key);
            }

            cout << output << endl;
            output.clear();

        }

    }

}
```

4.运行截图



```
E:\360MoveData\Users\1737783319\Desktop\信息安全\lab\lab1>main

Press 1:Encrypt/Decrypt
Press 2:quit
1
Chose key value(a number between 1 - 26):3

lowerCase letter for encrypt
upperCase letter for decrypt
Enter cipherTest: how are you
KRZ DUH BRX

Press 1:Encrypt/Decrypt
Press 2:quit
1
Chose key value(a number between 1 - 26):3

lowerCase letter for encrypt
upperCase letter for decrypt
Enter cipherTest: KRZ DUH BRX
how are you

Press 1:Encrypt/Decrypt
Press 2:quit
2
```

实验二　维吉尼亚密码

一、 实验目的

(1)实现维吉尼亚密码的加解密

(2)实现维吉尼亚密码的破解

二、算法描述

（1）维吉尼亚密码的加解密

1. 选择一个多字母密钥

2. 使用密钥的第一个字母加密明文的第一个字母，以此类推

3.当密钥的所有字母都使用完时，使用密钥的第一个字母开始加密，重复 2 操作

4.解密为 2 的逆操作

三、加解密的程序代码

//NewPolyalphabeticCipher.h

```
#include<cstdio>
#include<cctype>
#include<string>
#include<cmath>
using namespace std;

char vigenere_table[26][26] = {
	'A','B','C','D','E','F','G','H','I','J','K',
	'L','M','N','O','P','Q','R','S','T','U','V',
	'W','X','Y','Z',
	'B','C','D','E','F','G','H','I','J','K','L',
	'M','N','O','P','Q','R','S','T','U','V','W',
	'X','Y','Z','A',
	'C','D','E','F','G','H','I','J','K','L','M',
	'N','O','P','Q','R','S','T','U','V','W','X',
	'Y','Z','A','B',
	'D','E','F','G','H','I','J','K','L','M','N',
	'O','P','Q','R','S','T','U','V','W','X','Y',
```

'Z','A','B','C',

'E','F','G','H','I','J','K','L','M','N','O',

'P','Q','R','S','T','U','V','W','X','Y','Z',

'A','B','C','D',

'F','G','H','I','J','K','L','M','N','O','P',

'Q','R','S','T','U','V','W','X','Y','Z','A',

'B','C','D','E',

'G','H','I','J','K','L','M','N','O','P','Q',

'R','S','T','U','V','W','X','Y','Z','A','B',

'C','D','E','F',

'H','I','J','K','L','M','N','O','P','Q','R',

'S','T','U','V','W','X','Y','Z','A','B','C',

'D','E','F','G',

'I','J','K','L','M','N','O','P','Q','R','S',

'T','U','V','W','X','Y','Z','A','B','C','D',

'E','F','G','H',

'J','K','L','M','N','O','P','Q','R','S','T',

'U','V','W','X','Y','Z','A','B','C','D','E',

'F','G','H','I',

'K','L','M','N','O','P','Q','R','S','T','U',

'V','W','X','Y','Z','A','B','C','D','E','F',

'G','H','I','J',

'L','M','N','O','P','Q','R','S','T','U','V',

'W','X','Y','Z','A','B','C','D','E','F','G',

'H','I','J','K',

'M','N','O','P','Q','R','S','T','U','V','W',

'X','Y','Z','A','B','C','D','E','F','G','H',

'I','J','K','L',

'N','O','P','Q','R','S','T','U','V','W','X',

'Y','Z','A','B','C','D','E','F','G','H','I',

'J','K','L','M',

'O','P','Q','R','S','T','U','V','W','X','Y',

'Z','A','B','C','D','E','F','G','H','I','J',

'K','L','M','N',

'P','Q','R','S','T','U','V','W','X','Y','Z',

'A','B','C','D','E','F','G','H','I','J','K',

'L','M','N','O',

'Q','R','S','T','U','V','W','X','Y','Z','A',

```
        'B','C','D','E','F','G','H','I','J','K','L',
        'M','N','O','P',
        'R','S','T','U','V','W','X','Y','Z','A','B',
        'C','D','E','F','G','H','I','J','K','L','M',
        'N','O','P','Q',
        'S','T','U','V','W','X','Y','Z','A','B','C',
        'D','E','F','G','H','I','J','K','L','M','N',
        'O','P','Q','R',
        'T','U','V','W','X','Y','Z','A','B','C','D',
        'E','F','G','H','I','J','K','L','M','N','O',
        'P','Q','R','S',
        'U','V','W','X','Y','Z','A','B','C','D','E',
        'F','G','H','I','J','K','L','M','N','O','P',
        'Q','R','S','T',
        'V','W','X','Y','Z','A','B','C','D','E','F',
        'G','H','I','J','K','L','M','N','O','P','Q',
        'R','S','T','U',
        'W','X','Y','Z','A','B','C','D','E','F','G',
        'H','I','J','K','L','M','N','O','P','Q','R',
        'S','T','U','V',
        'X','Y','Z','A','B','C','D','E','F','G','H',
        'I','J','K','L','M','N','O','P','Q','R','S',
        'T','U','V','W',
        'Y','Z','A','B','C','D','E','F','G','H','I',
        'J','K','L','M','N','O','P','Q','R','S','T',
        'U','V','W','X',
        'Z','A','B','C','D','E','F','G','H','I','J',
        'K','L','M','N','O','P','Q','R','S','T','U',
        'V','W','X','Y'
};

void Decrypt(string in,string &out,string k){
        int i = 0;

        for(string::iterator it = in.begin(); it != in.end();it++){
                if((*it) != ' '){
                        int column = toupper(k[i % k.length()]) - 'A';
                        int row;
                        for(row = 0;row < 26;row++){
                                if(vigenere_table[row][column] == *it) break;
                        }
                        out += 'A' + row;
                        i++;
                }else{
                        out += ' ';
                }

        }
}

void Encrypt(string in,string &out,string k){
        int i = 0;
        for(string::iterator it = in.begin(); it != in.end();it++){
                if((*it) != ' '){
                        int row = toupper(*it) - 'A';
                        int column = toupper(k[i % k.length()]) - 'A';
                        out += vigenere_table[row][column];
                        i++;
                }else{
                        out += ' ';
                }

        }
}
```

//testPolyalphabeticCipher.cpp

```
#include "NewPolyalphabeticCipher.h"
#include "EndPolyalphabeticCipher.h"
#include<algorithm>
#include<string>
#include<iostream>
using namespace std;

int main(){
    string input,output,key;
```

```
	int choice = 0;

	while(true){
		cout << "----------------" << endl;
		cout << "1:Encrypt "<<endl
				<<"2:Decrypt "<<endl
			<<"3:EndDecrypt "<<endl
			<<"4:quit"<<endl;
		cout << "----------------" << endl;

		try{
			cin >> choice;
			cin.ignore();

			if(choice != 1 && choice != 2 && choice != 3 && choice != 4){
				throw "Incorrect Choice";
			}
		}catch(const char* chc){
			cerr << "Incorrecr Choice "<< endl;
			return 1;
		}

		if(choice == 4) break;

		try{
			cout << endl << "Enter cipher text:";
			getline(cin,input);

			for(int i = 0;i < input.size();i++){
				if( (!(input[i] >= 'a'&& input[i] <= 'z')) && (!(input[i] >= 'A' && input[i] <=
'Z')) && (!(input[i] == ' ')) ){
					throw "Incorrect string ";
				}
			}
		}catch(const char* str){
			cerr <<"input string have some digits or special sysbol"
				  <<endl;
			cerr << "please enter only alphabets" << endl;
```

```
                return 1;
            }

            if(choice == 1 || choice == 2){

                cout << "Enter Key(alphabets/words):";
                getline(cin,key);

                if(choice == 1){
                    Encrypt(input,output,key);
                    cout<<endl<<"Cipher text:"<<output<<endl;
                }else{
                    Decrypt(input,output,key);
                    cout<<endl<<"Plain text:"<<output<<endl;
                }

            }else if(choice == 3){

                transform(input.begin(),input.end(),input.begin(),::tolower);

                getKey(input);

                print();

                getAns(input);

                cout << endl;
            }

            input.clear();
            output.clear();
            key.clear();
        }
        return 0;
}
```

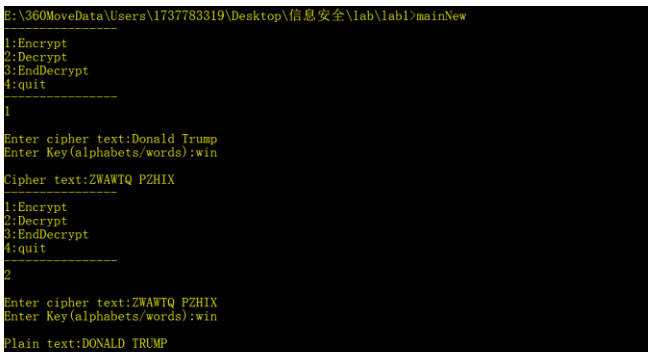
四、破解的程序代码

//EndPolyalphabeticCipher.h

```
#include<cstring>
#include<string>
#include<vector>
#include<set>
#include<map>
#include<algorithm>
#include<iostream>
using namespace std;

struct Node{
        double value;
        int length;
};

vector<Node> key;
set<int> keyLen;

double g[] = {
        0.08167,0.01492,0.02782,0.04253,0.12702,0.02228,0.02015,
        0.06094,0.06966,0.00153,0.00772,0.04025,0.02046,0.06749,
        0.07507,0.01929,0.00095,0.05987,0.06327,0.09056,0.02758,
        0.00978,0.02360,0.00150,0.01974,0.00074
};

bool cmp(Node a,Node b){
        return a.value < b.value;
}



double coincidenceIndex(string cipher,int start,int length){
        double index = 0.000;
        int sum = 0;
        int num[26];
        memset(num,0,sizeof(num));
```

```
    //keyPoint
    while(start <= cipher.length()){
        num[cipher[start] - 'a']++;

        start += length;

        sum++;
    }

    for(int i = 0;i < 26;i++){
        if(num[i] <= 1)continue;

        index += (double) (num[i] * (num[i] - 1)) / (double) ((sum) * (sum -1));
    }
    return index;
}

void findSame(string cipher){

    for(int i = 3;i < 5;i++){
        for(int j = 0;j < cipher.length() - 1;j++){
            string p = cipher.substr(j,i);
            for(int k = j + i;k < cipher.length() - i;k++){
                string tmp = cipher.substr(k,i);
                if(tmp == p){
                    Node x;

                    x.length = k - j;

                    key.push_back(x);
                }
            }
        }
    }

}

int gcd(int a,int b){
        if(b == 0) return a;
        else return gcd(b,a % b);
}

void print(){
        for(int i = 0;i < key.size();i++){
                cout << key[i].length<< " and " << key[i].value << endl;
        }

}

void getKey(string cipher){
        findSame(cipher);

        for(int i = 0;i < key.size();i++){
                int x = key[i].length;

                for(int j = 0;j < key.size();j++){
                        if(key[i].length > key[j].length){
                                keyLen.insert(gcd(key[i].length,key[j].length));
                        }else{
                                keyLen.insert(gcd(key[j].length,key[i].length));
                        }
                }
        }

        key.clear();

        set<int>::iterator it = keyLen.begin();

        while(it != keyLen.end()){
                int length = (*it);

                if(length == 1){
                        it++;
                        continue;
                }
                double sum = 0.000;

                cout << length << " ";

                for(int i = 0;i < length;i++){
                        cout << coincidenceIndex(cipher,i,length) << " ";
                        sum += coincidenceIndex(cipher,i,length);
                }

                cout << endl;

                Node x;
                x.length = length;
                x.value = (double)fabs(0.065 - (double)(sum / (double)length));
                if(x.value <= 0.1) key.push_back(x);

                it++;
        }
        sort(key.begin(),key.end(),cmp);
}

void getAns(string cipher){

        int lss = 0;

        while(lss < key.size() && lss < 10){
                Node x = key[lss];

                int ans[cipher.length()];
                memset(ans,0,sizeof(ans));

                map<char,int> mp;

                for(int i = 0;i < x.length;i++){
                        double maxPg = 0.000;

                        for(int k = 0;k < 26;k++){
                                mp.clear();
                                double pg = 0.000;
                                int sum = 0;

                                for(int j = i;j < cipher.length();j += x.length){
                                    char c = (char)((cipher[j] - 'a' + k) % 26 + 'a');
                                    mp[c]++;
                                    sum++;
                                }

                                for(char j = 'a'; j <= 'z';j++){
                                    pg += ( (double)mp[j] / (double)sum ) * g[j - 'a'];
                                }

                                if(pg > maxPg){
                                    ans[i] = k;
                                    maxPg = pg;
                                }
                        }
                }
                cout << endl << "Clear text:"<< endl;

                for(int i = 0;i < cipher.length();i++){
                        cout << (char) ((cipher[i] - 'a' + ans[i % x.length]) % 26 + 'a');
                }

                cout << endl;
                lss++;
        }
}
```

五、程序运行后的结果截图



```
E:\360MoveData\Users\1737783319\Desktop\信息安全\lab\lab1>mainNew
----------------
1:Encrypt
2:Decrypt
3:EndDecrypt
4:quit
----------------
1

Enter cipher text:Donald Trump
Enter Key(alphabets/words):win

Cipher text:ZWAWTQ PZHIX
----------------
1:Encrypt
2:Decrypt
3:EndDecrypt
4:quit
----------------
2

Enter cipher text:ZWAWTQ PZHIX
Enter Key(alphabets/words):win

Plain text:DONALD TRUMP
```