```cpp
#include<iostream>
#include<string>
#include<stdlib.h>
#include<windows.h>

using namespace std;
void menue();
void decryCaesar();
void encryCaesar();
void encryKeyword();
void decryKeyword();
void encryAffine();
void decryAffine();
void encryVigenere();
void decryVigenere();
void encryAutokey();
void decryAutokey();
void encryAutokey2();
void decryAutokey2();
void encryRotor();
void decryRotor();
void encryPlayfair();
void decryPlayfair();
void encryDouPlayfair();
void decryDouPlayfair();
void encrySkytale();
void decrySkytale();
void encryPermutation();
void decryPermutation();
void encryColumn();
void decryColumn();
void encrydoubleColumn();
void decrydoubleColumn();
char(*vigenereTable(char(&)[26][26]))[26];          //打印vigenere表(类比返回二维数组)
char(*matrixkey(char(&)[5][5]))[5];
char *strtem(string,char[50]);
char *strtem2(string,char[50]);
int *wordKey(string, int[20]);
int Check(string s);
int mainMenue();
int e_gcd(int a, int b, int &x, int &y);     //caeser cipher计算密钥
int cal(int a, int m);          //caeser cipher计算密钥
char menueOptions();
char algoOptions();
char PolyalpOptions();
char PolygraOptions();
char TransOptions();

struct group {
	char ch = '\0';
	int num = -1;
};

int mainMenue() {         //主菜单
	char choose;
	string strchoose;
	do {
		system("cls");
		cout << "************************************************" << endl;
		cout << "*                    OPTIONS                   *" << endl;
		cout << "*                                              *" << endl;
		cout << "*                1 Encryption                  *" << endl;
		cout << "*                2 Decryption                  *" << endl;
		cout << "*                3 Exit                        *" << endl;
		cout << "************************************************" << endl;
		cout << "What do you want to do: ";
		cin >> strchoose;
		if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1) {
			cout << "Error!Please enter again!" << endl;
			Sleep(500);
		}
	} while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1);
```

```
        choose = strchoose[0];
        return choose;
}

char menueOptions() {
        string strchoose;
        char choose;
        do {
                system("cls");
                cout << "*************************************************" << endl;
                cout << "*                  OPTIONS                      *" << endl;
                cout << "*                                               *" << endl;
                cout << "*           1 Monoalphabetic cipher             *" << endl;
                cout << "*           2 Polyalphabetic cipher             *" << endl;
                cout << "*           3 Polygraphics cipher               *" << endl;
                cout << "*           4 Transposition cipher              *" << endl;
                cout << "*           5 Return                            *" << endl;
                cout << "*************************************************" << endl;
                cout << "What do you want to do: ";
                cin >> strchoose;
                if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) !=
'4'&&strchoose.at(0) != '5' || strchoose.length() != 1) {
                        cout << "Error!Please enter again!";
                        Sleep(500);
                }
        } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) !=
'4'&&strchoose.at(0) != '5' || strchoose.length() != 1);
        choose = strchoose[0];
        return choose;
}

char algoOptions() {
        string strchoose;
        char choose;
        do {
                system("cls");
                cout << "*************************************************" << endl;
                cout << "*                  OPTIONS                      *" << endl;
                cout << "*                                               *" << endl;
                cout << "*              1 Caesar cipher                  *" << endl;
                cout << "*              2 Keyword cipher                 *" << endl;
                cout << "*              3 Affine cipher                  *" << endl;
                cout << "*              4 Return                         *" << endl;
                cout << "*************************************************" << endl;
                cout << "What do you want to do: ";
                cin >> strchoose;
                if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4' ||
strchoose.length() != 1) {
                        cout << "Error!Please enter again!";
                        Sleep(500);
                }
        } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4' ||
strchoose.length() != 1);
        choose = strchoose[0];
        return choose;
}
char PolyalpOptions() {
        string strchoose;
        char choose;
        do {
                system("cls");
                cout << "*************************************************" << endl;
                cout << "*                  OPTIONS                      *" << endl;
                cout << "*                                               *" << endl;
                cout << "*              1 Vigenere cipher                *" << endl;
                cout << "*              2 Autokey cipher                 *" << endl;
                cout << "*              3 Autokey plaintext              *" << endl;
                cout << "*              4 Rotor cipher                   *" << endl;
                cout << "*              5 Return                         *" << endl;
                cout << "*************************************************" << endl;
                cout << "What do you want to do: ";
                cin >> strchoose;
                if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4'
&&strchoose.at(0) != '5' || strchoose.length() != 1) {
```

```
                cout << "Error!Please enter again!";
                Sleep(500);
            }
        } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4'
&&strchoose.at(0) != '5' || strchoose.length() != 1);
        choose = strchoose[0];
        return choose;
}

char PolygraOptions() {
        string strchoose;
        char choose;
        do {
            system("cls");
            cout << "*************************************************" << endl;
            cout << "*                OPTIONS                        *" << endl;
            cout << "*                                               *" << endl;
            cout << "*           1 Playfair cipher                   *" << endl;
            cout << "*           2 Double Playfair cipher            *" << endl;
            cout << "*           3 Return                            *" << endl;
            cout << "*************************************************" << endl;
            cout << "What do you want to do: ";
            cin >> strchoose;
            if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1) {
                cout << "Error!Please enter again!";
                Sleep(500);
            }
        } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1);
        choose = strchoose[0];
        return choose;
}

char TransOptions() {
        string strchoose;
        char choose;
        do {
            system("cls");
            cout << "*************************************************" << endl;
            cout << "*                  OPTIONS                      *" << endl;
            cout << "*                                               *" << endl;
            cout << "*      1 Skytale                                *" << endl;
            cout << "*      2 Permutation cipher                     *" << endl;
            cout << "*      3 Column permutation cipher              *" << endl;
            cout << "*      4 Double column permutation cipher       *" << endl;
            cout << "*      5 Return                                 *" << endl;
            cout << "*************************************************" << endl;
            cout << "What do you want to do: ";
            cin >> strchoose;
            if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) !=
'4'&&strchoose.at(0) != '5' || strchoose.length() != 1) {
                cout << "Error!Please enter again!";
                Sleep(500);
            }
        } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) !=
'4'&&strchoose.at(0) != '5' || strchoose.length() != 1);
        choose = strchoose[0];
        return choose;
}
void menue() {
        string strchoose;
        char getchoose, choose1, choose2;
        while (1) {
            getchoose = mainMenue();        //调用主菜单
            switch (getchoose) {
            case '1':
                //选择菜单
                choose1 = menueOptions();
                switch (choose1) {
                case '1':
                    choose2 = algoOptions();
                    switch (choose2) {
                    case '1':
                        encryCaesar();
```

```
                                break;
                        case '2':
                                encryKeyword();
                                break;
                        case '3':
                                encryAffine();
                                break;
                        default:
                                break;
                        }
                        break;
                case '2':
                        choose2 = PolyalpOptions();
                        switch (choose2) {
                        case '1':
                                encryVigenere();
                                break;
                        case '2':
                                encryAutokey();
                                break;
                        case '3':
                                encryAutokey2();
                                break;
                        case '4':
                                encryRotor();
                                break;
                        default:
                                break;
                        }
                        break;
                case '3':
                        choose2 = PolygraOptions();
                        switch (choose2) {
                        case '1':
                                encryPlayfair();
                                break;
                        case '2':
                                encryDouPlayfair();
                                break;
                        default:
                                break;
                        }
                        break;
                case '4':
                        choose2 = TransOptions();
                        switch (choose2) {
                        case '1':
                                encrySkytale();
                                break;
                        case '2':
                                encryPermutation();
                                break;
                        case '3':
                                encryColumn();
                                break;
                        case '4':
                                encrydoubleColumn();
                                break;
                        default:
                                break;
                        }
                        break;
                }
                break;
        case '2':
                choose1 = menueOptions();
                switch (choose1) {
                case '1':
                        choose2 = algoOptions();
                        switch (choose2) {
                        case '1':
                                decryCaesar();
                                break;
```

```
                        case '2':
                                decryKeyword();
                                break;
                        case '3':
                                decryAffine();
                        default:
                                break;
                        }
                        break;
                case '2':
                        choose2 = PolyalpOptions();
                        switch (choose2) {
                        case '1':
                                decryVigenere();
                                break;
                        case '2':
                                decryAutokey();
                                break;
                        case '3':
                                decryAutokey2();
                                break;
                        case '4':
                                decryRotor();
                                break;
                        default:
                                break;
                        }
                        break;
                case '3':
                        choose2 = PolygraOptions();
                        switch (choose2) {
                        case '1':
                                decryPlayfair();
                                break;
                        case '2':
                                decryDouPlayfair();
                                break;
                        default:
                                break;
                        }
                        break;
                case '4':
                        choose2 = TransOptions();
                        switch (choose2) {
                        case '1':
                                decrySkytale();
                                break;
                        case '2':
                                decryPermutation();
                                break;
                        case '3':
                                decryColumn();
                                break;
                        case '4':
                                decrydoubleColumn();
                                break;
                        default:
                                break;
                        }
                        break;
                }
                break;
        case '3':
                return;
        }
    }
}


void encryCaesar() {
    int b, j, k;
    int flag = 0;
    string st;
```

```
358         char ch;
359         do {
360               fflush(stdin);
361               system("cls");
362               cout << "------------Caesar cipher(Encryption)------------" << endl;
363               cout << "Please enter the plaintext:" << endl;
364               cout << "Plaintext:";
365               cin >> st;
366               flag = Check(st);
367
368         } while (flag);
369
370         do {
371               fflush(stdin);
372               cout << "key = ";
373               cin >> b;
374               if (b < 0 || b>26) {
375                     cout << "Avalilable!Please enter again!" << endl;
376               }
377         } while (b < 0 || b>26);
378
379         cout << "Ciphertext: ";
380         for (j = 0; j < st.length(); j++) {
381               if (st[j] >= 'a'&&st[j] <= 'z')
382                     k = 'a' + (st[j] - 'a' + b) % 26;
383               else
384                     k = 'A' + (st[j] - 'A' + b) % 26;
385               ch = k;
386               cout << ch;
387         }
388         cout << endl;
389         fflush(stdin);
390         cout << endl;
391         system("pause");
392   }
393
394   void decryCaesar() {
395         int b, j, k;
396         int flag = 0;
397         string st;
398         char ch;
399         do {
400               fflush(stdin);
401               system("cls");
402               cout << "------------Caesar cipher(Decryption)------------" << endl;
403               cout << "Please enter the ciphertext:" << endl;
404               cout << "Ciphertext:";
405               cin >> st;
406               flag = Check(st);
407         } while (flag);
408
409         do {
410               fflush(stdin);
411               cout << "key = ";
412               cin >> b;
413               if (b < 0 || b>26) {
414                     cout << "Avalilable!Please enter again!" << endl;
415               }
416         } while (b < 0 || b>26);
417
418         cout << "Ciphertext: ";
419         for (j = 0; j < st.length(); j++) {
420               if (st[j] >= 'a'&&st[j] <= 'z')
421                     k = 'a' + (st[j] - 'a' - b) % 26;
422               else
423                     k = 'A' + (st[j] - 'A' - b) % 26;
424               ch = k;
425               cout << ch;
426         }
427         cout << endl;
428         fflush(stdin);
429         //getchar();
430         cout << endl;
```

```
	system("pause");
}

void encryKeyword() {
	int i, j, k = 0, n = 0;
	int flag = 0;
	string st, keyword;
	string alpha("abcdefghijklmnopqrstuvwxyz");
	string key(20, '\0');
	string rule(50, '\0');
	system("color 2F");
	do {
		fflush(stdin);
		system("cls");
		cout << "------------Keyword cipher(Encryption)------------" << endl;
		cout << "Please enter the plaintext:" << endl;
		cout << "  Plaintext: ";
		cin >> st;
		flag = Check(st);

	} while (flag);
	do {		//输入keyword
		fflush(stdin);
		cout << endl << "Select a keyword: ";
		cin >> keyword;
		flag = Check(keyword);
	} while (flag);
	for (i = 0; i < keyword.length(); i++) {		//提取key
		flag = 1;
		for (j = 0; j < i; j++) {
			if (key[j] == keyword[i])
				flag = 0;
		}
		if (flag)
			key[k++] = keyword[i];
	}
	key[k] = '\0';		//组合对应表
	while (key[n] != '\0') {
		rule[n] = key[n];
		n++;
	}
	rule[n] = '\0';

	for (i = 0; i < alpha.length(); i++) {
		flag = 1;
		for (j = 0; j < key.length(); j++) {
			if (alpha[i] == key[j]) {
				flag = 0;
				break;
			}
		}
		if (flag)
			rule[n++] = alpha[i];
	}
	rule[n] = '\0';
	cout << "  Key: " << key << endl;
	cout << endl << "          Password Table" << endl;
	cout << "---------------------------------------" << endl;
	cout << "plaintext:  " << alpha << endl;
	cout << "ciphertext: " << rule << endl;
	cout << "---------------------------------------" << endl << endl;
	cout << "Ciphertext: ";
	for (i = 0; i < st.length(); i++) {
		if (st[i] >= 'a'&&st[i] <= 'z') {
			cout << rule[(int)(st[i] - 'a')];
		}
		else {
			cout << rule[(int)(st[i] - 'A')];
		}
	}
	cout << endl << endl;
	system("pause");
}
```

```
504
505 void decryKeyword() {
506         int i, j, k = 0, n = 0;
507         int flag = 0;
508         string st, keyword;
509         string alpha("abcdefghijklmnopqrstuvwxyz");
510         string key(20, '\0');
511         string rule(50, '\0');
512         system("color 2F");
513         do {
514                 fflush(stdin);
515                 system("cls");
516                 cout << "------------Keyword cipher(Decryption)-----------" << endl;
517                 cout << "Please enter the ciphertext:" << endl;
518                 cout << "  Ciphertext: ";
519                 cin >> st;
520                 flag = Check(st);
521
522         } while (flag);
523         do {
524                 fflush(stdin);
525                 cout << endl << "Enter the keyword: ";
526                 cin >> keyword;
527                 flag = Check(keyword);
528         } while (flag);
529         for (i = 0; i < keyword.length(); i++) {
530                 flag = 1;
531                 for (j = 0; j < i; j++) {
532                         if (key[j] == keyword[i])
533                                 flag = 0;
534                 }
535                 if (flag)
536                         key[k++] = keyword[i];
537         }
538         key[k] = '\0';
539         while (key[n] != '\0') {
540                 rule[n] = key[n];
541                 n++;
542         }
543         rule[n] = '\0';
544
545         for (i = 0; i < alpha.length(); i++) {
546                 flag = 1;
547                 for (j = 0; j < key.length(); j++) {
548                         if (alpha[i] == key[j]) {
549                                 flag = 0;
550                                 break;
551                         }
552                 }
553                 if (flag)
554                         rule[n++] = alpha[i];
555         }
556         rule[n] = '\0';
557         cout << "  Key: " << key << endl;
558         cout << endl << "          Password Table" << endl;
559         cout << "---------------------------------------" << endl;
560         cout << "plaintext:  " << alpha << endl;
561         cout << "ciphertext: " << rule << endl;
562         cout << "---------------------------------------" << endl << endl;
563         cout << "Plaintext: ";
564         for (i = 0; i < st.length(); i++) {
565                 if (st[i] >= 'a'&&st[i] <= 'z') {
566                         cout << alpha[rule.find(st[i])];
567                 }
568                 else {
569                         cout << alpha[rule.find(st[i] + 32)];
570                 }
571         }
572         cout << endl << endl;
573         system("pause");
574 }
575
576 int Check(string s) {           //检查输入的字符串是否为纯英文
```

```
	int i = 0;
	while (s[i] != '\0') {
		if (!((s[i] >= 'a'&&s[i] <= 'z') || (s[i] >= 'A'&&s[i] <= 'Z'))) {
			cout << "Error!Please enter again!" << endl;
			Sleep(1000);
			return 1;
		}
		i++;
	}
	return 0;
}
/*
int CheckNum(int s) {
if (!(s >=0 &&s <= 20) ) {
cout << "Error!Please enter again!" << endl;
Sleep(1000);
return 1;
}
return 0;
}
*/

void encryAffine() {
	int flag = 0;
	int a, b, c, x, i, key;
	int y = 26;
	char ch;
	string st;
	do {
		fflush(stdin);
		system("cls");
		cout << "------------Affine cipher(Encryption)------------" << endl;
		cout << "Encryption Algorithm: c=ap+b mod 26" << endl;
		cout << "*  gcd(a,26)=1" << endl;
		cout << "*  two numbers are selected(a,b)between 0 and 25" << endl;
		cout << "*  p and c represent letters" << endl;
		cout << "*  p is a plaintext letter and c is a ciphertext letter" << endl << endl;
		cout << "Please enter the plaintext:" << endl;
		cout << "Plaintext:";
		cin >> st;
		flag = Check(st);
	} while (flag);
	do {		//判断a是否与26互素
		cout << "a = ";
		cin >> x;
		a = x;
		y = 26;
		while (x != y) {
			if (x > y)
				x = x - y;
			else
				y = y - x;
		}
		if (a > 26 || a < 0 || y != 1) {
			cout << "Avalilable!Please enter again!" << endl;
		}
	} while (y != 1 || a > 26 || a < 0);
	do {
		cout << "b = ";
		cin >> b;
		if (b < 0 || b>26) {
			cout << "Avalilable!Please enter again!" << endl;
		}
	} while (b < 0 || b>26);
	cout << endl;

	//计算密钥
	key = cal(7, 26);
	if (key < 0)
		cout << "Public Key = " << a << "   Key = " << key + 26 << endl;
	else
		cout << "Public Key = " << a << "   Key = " << key << endl;

```

```
650         cout << "Ciphertext: ";
651         for (i = 0; i < st.length(); i++) {
652               if (st[i] >= 'a'&&st[i] <= 'z') {
653                     c = 'a' + ((st[i] - 'a')*a + b) % 26;
654               }
655               else {
656                     c = 'A' + ((st[i] - 'A')*a + b) % 26;
657               }
658               ch = c;
659               cout << ch;
660         }
661         cout << endl << endl;
662         system("Pause");
663   }
664
665   //以下为计算模乘法逆元
666   int e_gcd(int a, int b, int &x, int &y) {
667         if (b == 0) {
668
669               x = 1;
670               y = 0;
671               return a;
672         }
673         int ans = e_gcd(b, a%b, x, y);
674         int temp = x;
675         x = y;
676         y = temp - a / b*y;
677         return ans;
678   }
679
680   int cal(int a, int m) {
681         int x, y;
682         int gcd = e_gcd(a, m, x, y);
683         if (1 % gcd != 0)
684               return -1;
685         x *= 1 / gcd;
686         m = abs(m);
687         int ans = x%m;
688         return ans;
689   }
690
691   void decryAffine() {
692         int flag = 0;
693         int a, b, c, i;
694         int y = 26;
695         char ch;
696         string st;
697         do {
698               fflush(stdin);
699               system("cls");
700               cout << "------------Affine cipher(Decryption)------------" << endl;
701               cout << "Decryption Algorithm: p=a'(c-b) mod 26" << endl;
702               cout << "*  p and c represent letters" << endl;
703               cout << "*  p is a plaintext letter and c is a ciphertext letter" << endl << endl;
704               cout << "Please enter the Ciphertext:" << endl;
705               cout << "Ciphertext:";
706               cin >> st;
707               flag = Check(st);
708         } while (flag);
709
710         do {
711               cout << "b = ";
712               cin >> b;
713               if (b < 0 || b>26) {
714                     cout << "Avalilable!Please enter again!" << endl;
715               }
716         } while (b < 0 || b>26);
717
718         cout << "Key = ";
719         cin >> a;
720
721         cout << "Plaintext: ";
722         for (i = 0; i < st.length(); i++) {
```

```
723                 if (st[i] >= 'a'&&st[i] <= 'z') {
724                       c = 'a' + ((st[i] - 'a' - b + 26)*a) % 26;
725                 }
726                 else {
727                       c = 'A' + ((st[i] - 'A' - b + 26)*a) % 26;
728                 }
729                 ch = c;
730                 cout << ch;
731           }
732           cout << endl << endl;
733           system("Pause");
734
735           cout << endl;
736     }
737
738
739     char(*vigenereTable(char(&list)[26][26]))[26]{              //返回二维数组
740           int i, j, k;
741     char ch;
742     for (i = 0; i < 26; i++) {
743           for (j = 0; j < 26; j++) {
744                 k = 'a' + (i + j) % 26;
745                 ch = k;
746                 list[i][j] = ch;
747           }
748     }
749     cout << "  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" << endl;
750     for (i = 0; i < 26; i++) {
751           cout << list[0][i] << " ";
752           for (j = 0; j < 26; j++) {
753
754                 cout << list[i][j] << " ";
755           }
756           cout << endl;
757     }
758     cout << endl;
759     return list;
760     }
761
762     void encryVigenere() {
763           int m, n;
764           int i = 0;
765           int j = 0;
766           int flag = 0;
767           char(*list)[26];
768           char list1[26][26] = { 0 };
769           char cipher[26][26] = { 0 };
770           string key, st;
771           string rule(50, '\0');
772
773           do {
774                 fflush(stdin);
775                 system("cls");
776                 cout << "-------------Vigenere cipher(Encryption)------------" << endl;
777                 cout << endl << "Vigenere table" << endl << endl;
778                 list = vigenereTable(list1);
779                 cout << "Please enter the plaintext:" << endl;
780                 cout << "Plaintext:";
781                 cin >> st;
782                 flag = Check(st);
783
784           } while (flag);
785           do {          //输入key
786                 fflush(stdin);
787                 cout << "Please enter the key: ";
788                 cin >> key;
789                 flag = Check(key);
790           } while (flag);
791
792           j = 0;
793           for (i = 0; i < st.length(); i++) {
794                 j = j % key.length();
795                 rule[i] = key[j];
```

```
796             j++;
797         }
798         rule[i] = '\0';
799         cout << endl;
800         cout << "---------------------------------------" << endl;
801         cout << "Keyword:    " << rule << endl;
802         cout << "Plaintext:  " << st << endl;
803         cout << "---------------------------------------" << endl << endl;
804         cout << "Ciphertext: ";
805         for (i = 0; i < st.length(); i++) {
806             m = rule[i] - 'a';
807             n = st[i] - 'a';
808             cout << list[n][m];
809         }
810         cout << endl << endl;
811 
812         system("PAUSE");
813 
814 }
815 
816 void decryVigenere() {
817         int m, n;
818         int i = 0;
819         int j = 0;
820         int flag = 0;
821         char(*list)[26];
822         char list1[26][26] = { 0 };
823         char cipher[26][26] = { 0 };
824         char plain;
825         string key, st;
826         string rule(50, '\0');
827 
828         do {
829             fflush(stdin);
830             system("cls");
831             cout << "------------Vigenere cipher(Decryption)-----------" << endl;
832             cout << endl << "Vigenere table" << endl << endl;
833             list = vigenereTable(list1);
834             cout << "Please enter the ciphertext:" << endl;
835             cout << "Ciphertext:";
836             cin >> st;
837             flag = Check(st);
838 
839         } while (flag);
840         do {        //输入key
841             fflush(stdin);
842             cout << "Please enter the key: ";
843             cin >> key;
844             flag = Check(key);
845         } while (flag);
846 
847         j = 0;
848         for (i = 0; i < st.length(); i++) {
849             j = j % key.length();
850             rule[i] = key[j];
851             j++;
852         }
853         rule[i] = '\0';
854         cout << endl;
855         cout << "---------------------------------------" << endl;
856         cout << "Keyword:    " << rule << endl;
857         cout << "Ciphertext:  " << st << endl;
858         cout << "---------------------------------------" << endl << endl;
859         cout << "Plaintext: ";
860 
861         //以下是解密关键代码
862         for (i = 0; i < st.length(); i++) {
863             m = rule[i] - 'a';
864             for (j = 0; j < 26; j++) {
865                 if (list[j][m] == st[i])
866                     break;
867             }
868             n = 'a' + j;
```

```
869             plain = n;
870             cout << plain;
871         }
872         cout << endl << endl;
873 
874         system("PAUSE");
875 
876 }
877 
878 void encryAutokey() {
879         int i, j, m, n;
880         int flag = 0;
881         string st, key;
882         char(*list)[26];
883         string rule(50, '\0');
884         char list1[26][26] = { 0 };
885         do {
886             fflush(stdin);
887             system("cls");
888             cout << "------------Autokey cipher(Encryption)-----------" << endl;
889             cout << endl << "Vigenere table" << endl << endl;
890             list = vigenereTable(list1);
891             cout << "Please enter the plaintext:" << endl;
892             cout << "Plaintext:";
893             cin >> st;
894             flag = Check(st);
895 
896         } while (flag);
897         do {         //输入key
898             fflush(stdin);
899             cout << "Please enter the key: ";
900             cin >> key;
901             flag = Check(key);
902         } while (flag);
903 
904         j = 0;
905         for (i = 0; i < key.length(); i++) {
906             j = j % key.length();
907             rule[i] = key[j];
908             j++;
909         }
910         rule[i] = '\0';
911         for (i = 0; i < st.length() - key.length(); i++) {
912 
913             m = rule[i] - 'a';
914             n = st[i] - 'a';
915             rule[i + key.length()] = list[n][m];
916 
917         }
918         rule[i + key.length()] = '\0';
919 
920         cout << endl;
921         cout << "---------------------------------------" << endl;
922         cout << "Keyword:    " << rule << endl;
923         cout << "Plaintext:  " << st << endl;
924         cout << "---------------------------------------" << endl << endl;
925         cout << "Ciphertext: ";
926 
927         for (i = 0; i < st.length(); i++) {
928             m = rule[i] - 'a';
929             n = st[i] - 'a';
930             cout << list[n][m];
931         }
932         cout << endl << endl;
933 
934         system("PAUSE");
935 }
936 
937 void decryAutokey() {
938         int i, j, m, n;
939         int flag = 0;
940         string st, key;
941         char(*list)[26];
```

```
	char plain;
	string rule(50, '\0');
	char list1[26][26] = { 0 };
	do {
		fflush(stdin);
		system("cls");
		cout << "------------Autokey cipher(Decryption)-----------" << endl;
		cout << endl << "Vigenere table" << endl << endl;
		list = vigenereTable(list1);
		cout << "Please enter the ciphertext:" << endl;
		cout << "Ciphertext:";
		cin >> st;
		flag = Check(st);

	} while (flag);
	do {		//输入key
		fflush(stdin);
		cout << "Please enter the key: ";
		cin >> key;
		flag = Check(key);
	} while (flag);

	j = 0;
	for (i = 0; i < key.length(); i++) {
		j = j % key.length();
		rule[i] = key[j];
		j++;
	}
	rule[i] = '\0';
	for (i = 0; i < st.length() - key.length(); i++) {
		rule[i + key.length()] = st[i];

	}
	rule[i + key.length()] = '\0';

	rule[i] = '\0';
	cout << endl;
	cout << "-------------------------------------" << endl;
	cout << "Keyword:     " << rule << endl;
	cout << "Ciphertext:  " << st << endl;
	cout << "-------------------------------------" << endl << endl;
	cout << "Plaintext: ";

	//以下是解密关键代码
	for (i = 0; i < st.length(); i++) {
		m = rule[i] - 'a';
		for (j = 0; j < 26; j++) {
			if (list[j][m] == st[i])
				break;
		}
		n = 'a' + j;
		plain = n;
		cout << plain;
	}
	cout << endl << endl;

	system("PAUSE");
}
void encryAutokey2() {
	int i, m, n;
	int flag = 0;
	string st, key;
	char(*list)[26];
	string rule(50, '\0');
	char list1[26][26] = { 0 };
	do {
		fflush(stdin);
		system("cls");
		cout << "------------Autokey plaintext(Encryption)-----------" << endl;
		cout << endl << "Vigenere table" << endl << endl;
		list = vigenereTable(list1);
		cout << "Please enter the plaintext:" << endl;
		cout << "Plaintext:";
```

```
1015            cin >> st;
1016            flag = Check(st);
1017
1018        } while (flag);
1019        do {        //输入key
1020            fflush(stdin);
1021            cout << "Please enter the key: ";
1022            cin >> key;
1023            flag = Check(key);
1024        } while (flag);
1025
1026        for (i = 0; i < key.length(); i++) {
1027            rule[i] = key[i];
1028        }
1029        rule[i] = '\0';
1030        for (i = 0; i < st.length() - key.length(); i++) {
1031            rule[i + key.length()] = st[i];
1032
1033        }
1034        rule[i + key.length()] = '\0';
1035
1036        cout << endl;
1037        cout << "---------------------------------------" << endl;
1038        cout << "Keyword:    " << rule << endl;
1039        cout << "Plaintext:  " << st << endl;
1040        cout << "---------------------------------------" << endl << endl;
1041        cout << "Ciphertext: ";
1042
1043        for (i = 0; i < st.length(); i++) {
1044            m = rule[i] - 'a';
1045            n = st[i] - 'a';
1046            cout << list[n][m];
1047        }
1048        cout << endl << endl;
1049
1050        system("PAUSE");
1051    }
1052
1053    void decryAutokey2() {
1054
1055        int i, j, m, n;
1056        int flag = 0;
1057        string st, key;
1058        char(*list)[26];
1059        string pl(50, '\0');
1060        string rule(50, '\0');
1061        char list1[26][26] = { 0 };
1062        do {
1063            fflush(stdin);
1064            system("cls");
1065            cout << "------------Autokey plaintext(Decryption)------------" << endl;
1066            cout << endl << "Vigenere table" << endl << endl;
1067            list = vigenereTable(list1);
1068            cout << "Please enter the ciphertext:" << endl;
1069            cout << "Ciphertext:";
1070            cin >> st;
1071            flag = Check(st);
1072
1073        } while (flag);
1074        do {        //输入key
1075            fflush(stdin);
1076            cout << "Please enter the key: ";
1077            cin >> key;
1078            flag = Check(key);
1079        } while (flag);
1080
1081        //解密与对应关系同时进行
1082        for (i = 0; i < key.length(); i++) {
1083            rule[i] = key[i];
1084        }
1085        rule[i] = '\0';
1086        for (i = 0; i < st.length(); i++) {
1087            m = rule[i] - 'a';
```

```
1088                for (j = 0; j < 26; j++) {
1089                      if (list[j][m] == st[i])
1090                            break;
1091                }
1092                n = 'a' + j;
1093                pl[i] = n;
1094                if (i < st.length() - key.length()) {
1095                      rule[i + key.length()] = n;
1096                      rule[i + key.length() + 1] = '\0';
1097                }
1098                //rule[i + key.length()] = st[i];
1099          }
1100
1101          pl[i] = '\0';
1102          cout << endl;
1103          cout << "---------------------------------------" << endl;
1104          cout << "Keyword:     " << rule << endl;
1105          cout << "Ciphertext:  " << st << endl;
1106          cout << "---------------------------------------" << endl << endl;
1107          cout << "Plaintext: ";
1108          cout << pl;
1109          cout << endl << endl;
1110          system("PAUSE");
1111    }
1112    void encryRotor() {
1113          system("cls");
1114          cout << "No algorithm included..." << endl;
1115          system("PAUSE");
1116    }
1117    void decryRotor() {
1118          system("cls");
1119          cout << "No algorithm included..." << endl;
1120          system("PAUSE");
1121    }
1122
1123    //Playfair cipher构造matrix key
1124
1125    char(*matrixkey(char(&matrix)[5][5]))[5]{
1126          int i = 0,j = 0,k = 0,n = 0;
1127    string keyword;
1128    int flag = 0;
1129    string key(20, '\0');
1130    string rule(50, '\0');
1131    string alpha("abcdefghijklmnopqrstuvwxyz");
1132
1133    do {          //输入keyword
1134          fflush(stdin);
1135          cout << "select a keyword:";
1136          cin >> keyword;
1137          flag = Check(keyword);
1138    } while (flag);
1139    for (i = 0; i < keyword.length(); i++) {            //提取key
1140          flag = 1;
1141          if (keyword[i] == 'j')
1142                keyword[i] = i;
1143          for (j = 0; j < i; j++) {
1144                if (key[j] == keyword[i])
1145                      flag = 0;
1146          }
1147          if (flag)
1148                key[k++] = keyword[i];
1149    }
1150    key[k] = '\0';              //组合对应表
1151    while (key[n] != '\0') {
1152          rule[n] = key[n];
1153          n++;
1154    }
1155    rule[n] = '\0';
1156
1157    for (i = 0; i < alpha.length(); i++) {
1158          flag = 1;
1159          for (j = 0; j < key.length(); j++) {
1160                if (alpha[i] == key[j] || alpha[i] == 'j') {    //i,j占据同一个字符
```

```
                    flag = 0;
                    break;
                }
        }
        if (flag)
                rule[n++] = alpha[i];
}
rule[n] = '\0';
// 组建 matrix key 并打印到屏幕
cout << endl << "    matrix key" << endl << endl;
k = 0;
for (i = 0; i < 5; i++) {
        cout << "   ";
        for (j = 0; j < 5; j++) {
                matrix[i][j] = rule[k];
                cout << " " << rule[k];
                k++;
        }
        cout << endl;
}
cout << endl << endl;
return matrix;
}

//Playfair cipher 处理输入的明文/密文，将格式一致化
char *strtem(string st,char tem[50]) {
        int i = 0, j = 0;
        int flag = 0;

        //两两组合配对，检查每组元素是否相同，若相同，则添加元素'q'
        for (i = 0; i < st.length(); i++) {
                if (st[i] == 'j') {       //字母j用字母i替代
                        tem[i] = 'i';
                        continue;
                }
                tem[i] = st[i];
        }
        tem[i] = '\0';

        for (i = 0; i < strlen(tem) - 1; i++) {
                if ((i % 2 == 0) && (tem[i] == tem[i + 1])) {   //检测分组元素
                        for (j = strlen(tem) - 1; j >= i; j--) {
                                tem[j + 1] = tem[j];
                        }
                        tem[i + 1] = 'q';
                }
        }
        //若长度为奇数，则在末尾添加元素'q'，使其成为偶数
        if ((strlen(tem) % 2 != 0) && (tem[strlen(tem) - 1] == 'q')) {
                cout << "Error!Please choose another encryption algorithm!" << endl;
                exit(-1);
        }
        if ((strlen(tem) % 2 != 0) && (tem[strlen(tem) - 1] != 'q')) {
                tem[strlen(tem)] = 'q';
        }
        tem[strlen(tem)] = '\0';
        return tem;

}

char *strtem2(string st,char tem[50]) {
        int i = 0, j = 0;
        int flag = 0;

        for (i = 0; i < st.length(); i++) {
                if (st[i] == 'j') {       //字母j用字母i替代
                        tem[i] = 'i';
                        continue;
                }
                tem[i] = st[i];
        }
        tem[i] = '\0';

```

```
1234        //若长度为奇数，则在末尾添加元素'x'，使其成为偶数
1235        if ((strlen(tem) % 2 != 0)) {
1236              tem[strlen(tem)] = 'x';
1237        }
1238        tem[strlen(tem)] = '\0';
1239        return tem;
1240
1241  }
1242  void encryPlayfair() {
1243        int i = 0, j = 0, k = 0, n = 0, m = 0;
1244        int flag = 0;
1245        int flag1, flag2, flag3, flag4;
1246        char(*matrix)[5];
1247        char matrix1[5][5] = { 0 };
1248        char plain[20][2] = { 0 };
1249        char tem[50] = { 0 };
1250        string st;
1251        do {
1252              fflush(stdin);
1253              system("cls");
1254              cout << "-------------Playfair cipher(Encryption)------------" << endl;
1255              cout << "               the null letter is 'q'               " << endl;
1256              cout << "Please enter the plaintext:";
1257              cin >> st;
1258              flag = Check(st);
1259        } while (flag);
1260
1261        strtem(st, tem);
1262        matrix = matrixkey(matrix1);
1263
1264        //将一维数组转化为二维数组，便于加密
1265        for (i = 0; i < strlen(tem) / 2; i++) {
1266              for (j = 0; j < 2; j++) {
1267                    plain[i][j] = tem[k];
1268                    k++;
1269              }
1270        }
1271
1272        //以下为加密
1273        flag = i;
1274        cout << "CipherText:";
1275        for (i = 0; i < flag; i++) {
1276              j = 0;
1277              for (m = 0; m < 5; m++) {
1278                    for (n = 0; n < 5; n++) {
1279                          j = 0;
1280                          if (matrix[m][n] == plain[i][j]) {
1281                                flag1 = m;
1282                                flag2 = n;
1283                          }
1284                          if (matrix[m][n] == plain[i][(j + 1)]) {
1285                                flag3 = m;
1286                                flag4 = n;
1287                          }
1288                    }
1289              }
1290              if (flag1 == flag3) {
1291                    cout << matrix[flag1][((flag2 + 1) % 5)] << matrix[flag3][((flag4 + 1) % 5)];
1292              }
1293              else if (flag2 == flag4) {
1294                    cout << matrix[((flag1 + 1) % 5)][flag2] << matrix[((flag3 + 1) % 5)][flag4];
1295              }
1296              else {
1297                    cout << matrix[flag1][flag4] << matrix[flag3][flag2];
1298              }
1299        }
1300        cout << endl << endl;
1301        system("PAUSE");
1302  }
1303
1304  void decryPlayfair() {          //解密和加密基本相同，只是修改解密关键代码
1305
1306        int i = 0, j = 0, k = 0, n = 0, m = 0;
```

```
1307        int flag = 0;
1308        int flag1, flag2, flag3, flag4;
1309        char(*matrix)[5];
1310        char matrix1[5][5] = { 0 };
1311        char plain[20][2] = { 0 };
1312        char tem[50] = { 0 };
1313        string st;
1314        do {
1315              fflush(stdin);
1316              system("cls");
1317              cout << "------------Playfair cipher(decryption)-----------" << endl;
1318              cout << "Please enter the ciphertext:";
1319              cin >> st;
1320              if ((st.length() % 2) != 0) {
1321                    cout << "Error!There must be even number of characters in the ciphertext!Please enter again!" <<
endl;
1322                    exit(-1);
1323              }
1324              flag = Check(st);
1325        } while (flag);
1326
1327        strtem(st, tem);
1328        matrix = matrixkey(matrix1);
1329
1330        //将一维数组转化为二维数组，便于解密
1331        for (i = 0; i < strlen(tem) / 2; i++) {
1332              for (j = 0; j < 2; j++) {
1333                    plain[i][j] = tem[k];
1334                    k++;
1335              }
1336        }
1337
1338        //以下为解密
1339        flag = i;
1340        cout << "PlainText:";
1341        for (i = 0; i < flag; i++) {
1342              j = 0;
1343              for (m = 0; m < 5; m++) {
1344                    for (n = 0; n < 5; n++) {
1345                          j = 0;
1346                          if (matrix[m][n] == plain[i][j]) {
1347                                flag1 = m;
1348                                flag2 = n;
1349                          }
1350                          if (matrix[m][n] == plain[i][(j + 1)]) {
1351                                flag3 = m;
1352                                flag4 = n;
1353                          }
1354                    }
1355              }
1356              //解密关键代码
1357              if (flag1 == flag3) {
1358                    cout << matrix[flag1][((flag2 + 4) % 5)] << matrix[flag3][((flag4 + 4) % 5)];
1359              }
1360              else if (flag2 == flag4) {
1361                    cout << matrix[((flag1 + 4) % 5)][flag2] << matrix[((flag3 + 4) % 5)][flag4];
1362              }
1363              else {
1364                    cout << matrix[flag1][flag4] << matrix[flag3][flag2];
1365              }
1366        }
1367        cout << endl << endl;
1368        system("PAUSE");
1369
1370  }
1371
1372  void encryDouPlayfair() {
1373        int i = 0, j = 0, k = 0, n = 0, m = 0, t = 0, q = 0;
1374        int period;
1375        int flag = 0;
1376        int flag1, flag2, flag3, flag4;
1377        int flag_1, flag_2, flag_3, flag_4;
1378        char(*matrix_1)[5];
```

```
1379        char(*matrix_2)[5];
1380        char matrix1[5][5] = { 0 };
1381        char matrix2[5][5] = { 0 };
1382        char plain[20][2] = { 0 };
1383        char test[10][30] = { 0 };
1384        char tem[50] = { 0 };
1385        string st;
1386        do {
1387              fflush(stdin);
1388              system("cls");
1389              cout << "---------Double Playfair cipher(Encryption)---------" << endl;
1390              cout << "               the null letter is 'x'               " << endl;
1391              cout << "Please enter the plaintext:";
1392              cin >> st;
1393              flag = Check(st);
1394        } while (flag);
1395
1396
1397
1398        cout << "Please enter the cipher period:";          //*此处可改进以下，检测输入的是不是数字，防止运行时出错
1399        cin >> period;
1400
1401        strtem(st, tem);
1402        matrix_1 = matrixkey(matrix1);
1403        matrix_2 = matrixkey(matrix2);
1404
1405        //打印组合的matrix key
1406        cout << endl << endl << "              Playfair squares" << endl << endl;
1407        for (i = 0; i < 5; i++) {
1408              cout << "    ";
1409              for (j = 0; j < 5; j++) {
1410                    cout << matrix_1[i][j];
1411                    cout << "  ";
1412              }
1413
1414              cout << "        ";
1415              for (k = 0; k < 5; k++) {
1416                    cout << matrix_2[i][k];
1417                    cout << "  ";
1418              }
1419              cout << endl;
1420        }
1421        cout << endl;
1422        cout << "   Group Of The Plaintext" << endl;
1423
1424        k = 0;
1425        for (i = 0; i < strlen(tem); ) {
1426              if (tem[k] != '\0') {
1427
1428                    if ((strlen(tem) - k) >= (2 * period)) {
1429
1430                          for (m = 0; m < 2; m++) {
1431                                cout << "    ";
1432                                for (j = 0; j < period; j++) {
1433                                      test[t][j] = tem[k];
1434                                      cout << " " << test[t][j];
1435                                      k++;
1436                                      i++;
1437                                }
1438                                cout << endl;
1439                                t++;
1440                          }
1441                    }
1442
1443                    else {
1444                          q = (strlen(tem) - k) / 2;
1445
1446                          for (m = 0; m < 2; m++) {
1447                                cout << "    ";
1448                                for (j = 0; j < q; j++) {
1449                                      test[t][j] = tem[k];
1450                                      cout << " " << test[t][j];
1451                                      k++;
```

```
                                    i++;
                              }
                              t++;
                              cout << endl;
                        }

                  }
            }
            else {
                  break;
            }

      }

      cout << endl;
      cout << "CipherText: ";
      flag = t - 1;
      m = 0;
      for (i = 0; i < flag; ) {
            if ((i == (flag - 1)) && q != 0) {
                  period = q;
            }
            for (j = 0; j < period; j++) {
                  for (m = 0; m < 5; m++) {
                        for (n = 0; n < 5; n++) {
                              if (matrix_1[m][n] == test[i][j]) {
                                    flag1 = m;
                                    flag2 = n;
                              }
                              if (matrix_2[m][n] == test[(i + 1)][j]) {
                                    flag3 = m;
                                    flag4 = n;
                              }
                              flag_1 = flag1;
                              flag_2 = flag2;
                              flag_3 = flag3;
                              flag_4 = flag4;
                        }
                  }
                  if (flag1 == flag3) {
                        for (m = 0; m < 5; m++) {
                              for (n = 0; n < 5; n++) {
                                    if (matrix_1[m][n] == matrix_2[flag_3][(flag_4 + 1) % 5]) {
                                          flag1 = m;
                                          flag2 = n;
                                    }
                                    if (matrix_2[m][n] == matrix_1[flag_1][(flag_2 + 1) % 5]) {
                                          flag3 = m;
                                          flag4 = n;
                                    }
                              }

                        }
                        if (flag1 == flag3) {
                              cout << matrix_2[flag3][(flag4 + 1) % 5] << matrix_1[flag1][(flag2 + 1) % 5];
                        }
                        else {
                              cout << matrix_2[flag1][flag4] << matrix_1[flag3][flag2];
                        }
                  }
                  else {
                        for (m = 0; m < 5; m++) {
                              for (n = 0; n < 5; n++) {
                                    if (matrix_1[m][n] == matrix_2[flag_1][flag_4]) {
                                          flag1 = m;
                                          flag2 = n;
                                    }
                                    if (matrix_2[m][n] == matrix_1[flag_3][flag_2]) {
                                          flag3 = m;
                                          flag4 = n;
                                    }
                              }
                        }
                  }
```

```
                        if (flag1 == flag2) {
                                cout << matrix_2[flag3][(flag4 + 1) % 5] << matrix_1[flag1][(flag2 + 1) % 5];
                        }
                        else {
                                cout << matrix_2[flag1][flag4] << matrix_1[flag3][flag2];
                        }

                }

            }
            i = i + 2;
        }
        cout << endl << endl;
        system("PAUSE");


}

void decryDouPlayfair() {
        int i = 0, j = 0, k = 0, n = 0, m = 0, t = 0, q = 0;
        int period;
        int flag = 0;
        int flag1, flag2, flag3, flag4;
        int flag_1, flag_2, flag_3, flag_4;
        char(*matrix_1)[5];
        char(*matrix_2)[5];
        char matrix1[5][5] = { 0 };
        char matrix2[5][5] = { 0 };
        char plain[20][2] = { 0 };
        char test[10][30] = { 0 };
        char plaintext[20][20] = { 0 };
        char tem[50] = { 0 };
        string st;
        do {
                fflush(stdin);
                system("cls");
                cout << "----------Double Playfair cipher(Decryption)--------" << endl;
                cout << "Please enter the ciphertext:";
                cin >> st;
                if ((st.length() % 2) != 0) {
                        cout << "Error!There must be even number of characters in the ciphertext!Please enter again!" <<
endl;
                        exit(-1);
                }
                flag = Check(st);
        } while (flag);

        cout << "Please enter the cipher period:";
        cin >> period;

        strtem(st, tem);
        matrix_1 = matrixkey(matrix1);
        matrix_2 = matrixkey(matrix2);

    //打印组合的matrix key
        cout << endl << endl << "             Playfair squares" << endl << endl;
        for (i = 0; i < 5; i++) {
                cout << "   ";
                for (j = 0; j < 5; j++) {
                        cout << matrix_1[i][j];
                        cout << "  ";
                }

                cout << "        ";
                for (k = 0; k < 5; k++) {
                        cout << matrix_2[i][k];
                        cout << "  ";
                }
                cout << endl;
        }
        cout << endl;
        cout << "   Group Of The Ciphertext" << endl << endl;

```

```
1597        //整理密文为矩阵形式
1598        k = 0;
1599        t = 0;
1600        for (i = 0; i < strlen(tem); ) {
1601              if (tem[k] != '\0') {
1602
1603                    if ((strlen(tem) - k) >= (2 * period)) {
1604                          for (m = 0; m < period; m++) {
1605
1606                                for (j = t; j < (t + 2); j++) {
1607                                      test[j][m] = tem[k];
1608                                      k++;
1609                                      i++;
1610                                }
1611                          }
1612                          t = t + 2;
1613                    }
1614
1615                    else {
1616                          q = (strlen(tem) - k) / 2;
1617
1618                          for (m = 0; m < q; m++) {
1619
1620                                for (j = t; j < (t + 2); j++) {
1621                                      test[j][m] = tem[k];
1622                                      k++;
1623                                      i++;
1624                                }
1625                          }
1626                          t = t + 2;
1627                    }
1628              }
1629              else {
1630                    break;
1631              }
1632        }
1633
1634
1635        //打印矩阵密文
1636        k = 0;
1637        t = 0;
1638        for (i = 0; i < strlen(tem); ) {
1639              if (tem[k + 1] != '\0') {
1640
1641                    if ((strlen(tem) - k) >= (2 * period)) {
1642                          for (j = 0; j < 2; j++) {
1643
1644
1645                                for (m = 0; m < period; m++) {
1646                                      cout << "   ";
1647                                      cout << test[t][m];
1648                                      k++;
1649                                      i++;
1650
1651                                }
1652                                cout << endl;
1653                                t++;
1654                          }
1655                    }
1656
1657                    else {
1658                          q = (strlen(tem) - k) / 2;
1659                          for (j = 0; j < 2; j++) {
1660                                for (m = 0; m < q; m++) {
1661                                      cout << "   ";
1662                                      cout << test[t][m];
1663                                      k++;
1664                                      i++;
1665                                }
1666                                cout << endl;
1667                                t++;
1668                          }
1669                    }
```

```
            }
            else {
                  break;
            }
      }

      cout << endl;

      //寻找明文
      flag = t - 1;
      m = 0;
      for (i = 0; i < flag; ) {
            if ((i == (flag - 1)) && q != 0) {
                  period = q;
            }
            for (j = 0; j < period; j++) {
                  for (m = 0; m < 5; m++) {
                        for (n = 0; n < 5; n++) {
                              if (matrix_2[m][n] == test[i][j]) {
                                    flag3 = m;
                                    flag4 = n;
                              }
                              if (matrix_1[m][n] == test[(i + 1)][j]) {
                                    flag1 = m;
                                    flag2 = n;
                              }
                              flag_1 = flag1;
                              flag_2 = flag2;
                              flag_3 = flag3;
                              flag_4 = flag4;
                        }
                  }
                  if (flag1 == flag3) {
                        for (m = 0; m < 5; m++) {
                              for (n = 0; n < 5; n++) {
                                    if (matrix_2[m][n] == matrix_1[flag_1][(flag_2 + 4) % 5]) {
                                          flag3 = m;
                                          flag4 = n;
                                    }
                                    if (matrix_1[m][n] == matrix_2[flag_3][(flag_4 + 4) % 5]) {
                                          flag1 = m;
                                          flag2 = n;
                                    }
                              }

                        }
                        if (flag1 == flag3) {
                              plaintext[i][j] = matrix_1[flag1][(flag2 + 4) % 5];
                              plaintext[i + 1][j] = matrix_2[flag3][(flag4 + 4) % 5];
                        }
                        else {
                              plaintext[i][j] = matrix_1[flag3][flag2];
                              plaintext[i + 1][j] = matrix_2[flag1][flag4];
                        }
                  }
                  else {
                        for (m = 0; m < 5; m++) {
                              for (n = 0; n < 5; n++) {
                                    if (matrix_2[m][n] == matrix_1[flag_3][flag_2]) {
                                          flag3 = m;
                                          flag4 = n;
                                    }
                                    if (matrix_1[m][n] == matrix_2[flag_1][flag_4]) {
                                          flag1 = m;
                                          flag2 = n;
                                    }
                              }
                        }
                        if (flag1 == flag2) {
                              plaintext[i][j] = matrix_1[flag1][(flag2 + 4) % 5];
                              plaintext[i + 1][j] = matrix_2[flag3][(flag4 + 4) % 5];
                        }
                        else {
```

```
1743                                plaintext[i][j] = matrix_1[flag3][flag2];
1744                                plaintext[i + 1][j] = matrix_2[flag1][flag4];
1745                          }
1746                    }
1747              }
1748              i = i + 2;
1749        }
1750
1751        j = 0;
1752        k = 0;
1753        cout << "PlainText:";
1754        for (i = 0; i < st.length();) {
1755              if (plaintext[j][k] != '\0') {
1756                    cout << plaintext[j][k];
1757                    k++;
1758                    i++;
1759                    continue;
1760              }
1761              k = 0;
1762              j++;
1763        }
1764        cout << endl << endl;
1765        system("PAUSE");
1766  }
1767
1768  void encrySkytale() {
1769        fflush(stdin);
1770        system("cls");
1771        cout << "    A scytale a tool used to perform a transposition cipher, consisting of a cylinder with a strip of
      parchment wound" << endl;
1772        cout << " around it on which is written a message.(from Wikipedia)" << endl << endl;
1773        cout << "No algorithm included..." << endl;
1774        system("PAUSE");
1775  }
1776
1777  void decrySkytale() {
1778        fflush(stdin);
1779        system("cls");
1780        cout << "    A scytale a tool used to perform a transposition cipher, consisting of a cylinder with a strip of
      parchment wound" << endl;
1781        cout << " around it on which is written a message.(from Wikipedia)" << endl << endl;
1782        cout << "No algorithm included..." << endl;
1783        system("PAUSE");
1784  }
1785
1786
1787  void encryPermutation() {
1788        int flag = 0;
1789        int i, j, t;
1790        int k[20] = { 0 };
1791        int *p;
1792        char tem[20] = { 0 };
1793        char keygroup[20] = { 0 };
1794        char plain[30] = { 0 };
1795        char cipher[30] = { 0 };
1796        string st, key;
1797        group list[10];
1798        do {
1799              fflush(stdin);
1800              system("cls");
1801              cout << "------------Permutation cipher(Encryption)-----------" << endl;
1802              cout << "               the null letter is 'x'               " << endl;
1803              cout << "Please enter the plaintext:";
1804              cin >> st;
1805              flag = Check(st);
1806        } while (flag);
1807
1808
1809        do {          //输入key
1810              fflush(stdin);
1811              cout << "Please enter the key: ";
1812              cin >> key;
1813              flag = Check(key);
```

```
1814        } while (flag);
1815
1816
1817        //明文不是key的整数倍补'x'
1818        t = st.length();
1819        if (t%key.length() != 0)
1820              t = ((t / (key.length()) + 1)*key.length());
1821        for (i = 0; i < t; i++) {
1822              if (i < st.length())
1823                    plain[i] = st[i];
1824              else
1825                    plain[i] = 'x';
1826        }
1827        p = wordKey(key, k);
1828
1829        //展示
1830
1831        cout << "   ";
1832        for (i = 0; i < (strlen(plain) / key.length()); i++) {
1833              for (j = 0; j < key.length(); j++) {
1834                    cout << p[j];
1835              }
1836              cout << "   ";
1837        }
1838        cout << endl;
1839
1840
1841        for (i = 0; i < strlen(plain); i++) {
1842              if ((i%key.length()) == 0)
1843                    cout << "   ";
1844              cout << plain[i];
1845
1846        }
1847        cout << endl << endl;
1848        //整理密文
1849        for (i = 0; i < t / (key.length()); i++) {
1850              for (j = 0; j < key.length(); j++) {
1851                    cipher[i*key.length() + p[j] - 1] = plain[i*key.length() + j];
1852              }
1853        }
1854
1855
1856        //输出密文
1857        cout << "CipherText:";
1858        for (i = 0; i < strlen(cipher); i++) {
1859              cout << cipher[i];
1860        }
1861
1862        cout << endl << endl;
1863        system("PAUSE");
1864  }
1865
1866  void decryPermutation() {
1867        int flag = 0;
1868        int i, j;
1869        int k[20] = { 0 };
1870        int *p;
1871        char tem[20] = { 0 };
1872        char keygroup[20] = { 0 };
1873        char plain[30] = { 0 };
1874        char cipher[30] = { 0 };
1875        string st, key;
1876        group list[10];
1877        do {
1878              fflush(stdin);
1879              system("cls");
1880              cout << "------------Permutation cipher(Decryption)-----------" << endl;
1881              cout << "Please enter the ciphertext:";
1882              cin >> st;
1883              flag = Check(st);
1884        } while (flag);
1885
1886        do {          //输入key
```

```
1887                fflush(stdin);
1888                cout << "Please enter the key: ";
1889                cin >> key;
1890                flag = Check(key);
1891          } while (flag);
1892
1893          if (((st.length()) % (key.length())) != 0) {
1894                cout << endl << "Error!The length of the ciphertext must be an integral multiple of the length of the key!" << endl << endl;
1895                exit(-1);
1896          }
1897          p = wordKey(key, k);
1898
1899          //展示
1900
1901          cout << "   ";
1902          for (i = 0; i < (st.length() / key.length()); i++) {
1903                for (j = 0; j < key.length(); j++) {
1904                      cout << p[j];
1905                }
1906                cout << "   ";
1907          }
1908          cout << endl;
1909
1910
1911          for (i = 0; i < st.length(); i++) {
1912                if ((i%key.length()) == 0)
1913                      cout << "   ";
1914                cout << st[i];
1915
1916          }
1917          cout << endl << endl;
1918
1919          cout << "PlainText:";
1920          for (i = 0; i < (st.length() / key.length()); i++) {
1921                for (j = 0; j < key.length(); j++) {
1922                      cout << st[(i*key.length() + p[j] - 1)];
1923                }
1924          }
1925          cout << endl << endl;
1926          system("PAUSE");
1927    }
1928
1929    //处理密钥,返回指针数组
1930    int *wordKey(string key, int k[20]) {
1931          int i, j;
1932          //     int k[20] = { 0 };
1933          int *p = k;
1934          int flag = 0;
1935          char temp1;
1936          char tem[20] = { 0 };
1937          char keygroup[20] = { 0 };
1938
1939          for (i = 0; i < key.length(); i++) {//设置临时数组
1940                keygroup[i] = key[i];
1941          }
1942          //冒泡排序
1943          for (i = 0; i < key.length(); i++) {
1944                flag = 0;
1945                for (j = 0; j < key.length() - i - 1; j++) {
1946                      if (keygroup[j] > keygroup[j + 1]) {
1947                            temp1 = keygroup[j];
1948                            keygroup[j] = keygroup[j + 1];
1949                            keygroup[j + 1] = temp1;
1950                            flag = 1;
1951                      }
1952                }
1953                if (flag == 0)
1954                      break;
1955          }
1956
1957          //设置临时数组
1958          for (i = 0; i < key.length(); i++) {
```

```
1959
1960            tem[i] = keygroup[i];
1961        }
1962
1963        //对应的密钥顺序
1964        cout << endl << "key = (d, f)" << endl;
1965        cout << "d = " << key.length() << "   ";
1966        cout << "f = (";
1967        for (i = 0; i < key.length(); i++) {
1968            for (j = 0; j < key.length(); j++) {
1969                if (keygroup[j] == key[i] && tem[j] != '#') {
1970                    k[i] = j + 1;
1971                    tem[j] = '#';
1972                    cout << " " << k[i];
1973                    break;
1974                }
1975
1976            }
1977        }
1978        cout << " )" << endl << endl;
1979        return p;
1980
1981 }
1982
1983
1984 void encryColumn() {
1985        int flag = 0;
1986        int k[20] = { 0 };
1987        int t, i, j, m, n;
1988        string st, key;
1989        char tem[50] = { 0 };
1990        char cipher[10][10] = { 0 };
1991        int *p;
1992        do {
1993            fflush(stdin);
1994            system("cls");
1995            cout << "--------Column Permutation cipher(Encryption)--------" << endl;
1996            cout << "               the null letter is 'x'                " << endl;
1997            cout << "Please enter the plaintext:";
1998            cin >> st;
1999            flag = Check(st);
2000        } while (flag);
2001
2002
2003        do {        //输入key
2004            fflush(stdin);
2005            cout << "Please enter the key: ";
2006            cin >> key;
2007            flag = Check(key);
2008        } while (flag);
2009
2010        //明文不是key的整数倍补'x'
2011        t = st.length();
2012        if (t%key.length() != 0)
2013            t = ((t / (key.length()) + 1)*key.length());
2014        for (i = 0; i < t; i++) {
2015            if (i < st.length())
2016                tem[i] = st[i];
2017            else
2018                tem[i] = 'x';
2019        }
2020
2021        //打印plain matrix
2022        cout << "Plaintext Matrix" << endl << endl;
2023        m = 0;
2024        n = (strlen(tem)) / (key.length());
2025        for (i = 0; i < n; i++) {
2026
2027            for (j = 0; j < key.length(); j++) {
2028                cipher[i][j] = tem[m];
2029                cout << " " << cipher[i][j];
2030                m++;
2031            }
```

```
2032            cout << endl;
2033        }
2034
2035        p = wordKey(key, k);
2036        //加密
2037        cout << "CipherText:";
2038        //cout << key.length() << "  " << n << endl;
2039        for (i = 0; i < key.length(); i++) {
2040            for (j = 0; j < n; j++) {
2041                cout << cipher[j][(p[i] - 1)];
2042            }
2043        }
2044        cout << endl << endl;
2045        system("PAUSE");
2046    }
2047
2048    void decryColumn() {
2049        int flag = 0;
2050        int k[20] = { 0 };
2051        int  i, j, m, n;
2052        string st, key;
2053        char tem[50] = { 0 };
2054        char plain[10][10] = { 0 };
2055        int *p;
2056        do {
2057            fflush(stdin);
2058            system("cls");
2059            cout << "--------Column Permutation cipher(Decryption)--------" << endl;
2060            cout << "Please enter the ciphertext:";
2061            cin >> st;
2062            flag = Check(st);
2063        } while (flag);
2064
2065
2066        do {        //输入key
2067            fflush(stdin);
2068            cout << "Please enter the key: ";
2069            cin >> key;
2070            flag = Check(key);
2071        } while (flag);
2072
2073        //判断是否符合要求
2074        if (((st.length()) % (key.length())) != 0) {
2075            cout << endl << "Error!The length of the ciphertext must be an integral multiple of the length of the key!"
    << endl << endl;
2076            exit(-1);
2077        }
2078
2079        p = wordKey(key, k);
2080
2081        //组建明文序列
2082
2083        m = 0;
2084        n = (st.length()) / (key.length());
2085        for (i = 0; i < key.length(); i++) {
2086
2087            for (j = 0; j < n; j++) {
2088                plain[j][(p[i] - 1)] = st[m];
2089                m++;
2090            }
2091        }
2092
2093        //打印plain matrix
2094
2095        cout << "Plaintext Matrix" << endl << endl;
2096        for (i = 0; i < n; i++) {
2097            for (j = 0; j < key.length(); j++) {
2098                cout << "  " << plain[i][j];
2099            }
2100            cout << endl;
2101        }
2102        cout << endl;
2103
```

```
	//输出明文
	cout << "Plaintext:";
	for (i = 0; i < n; i++) {
		for (j = 0; j < key.length(); j++) {
			cout << plain[i][j];
		}
	}
	cout << endl << endl;
	system("PAUSE");
}

void encrydoubleColumn() {
	int flag = 0;
	int k[20] = { 0 };
	int k2[20] = { 0 };
	int t, i, j, m, n, q;
	string st, key, key2;
	char tem[50] = { 0 };
	char tem2[50] = { 0 };
	char text[50] = { 0 };
	char cipher[10][10] = { 0 };
	char cipher2[10][10] = { 0 };
	int *p, *p2;
	do {
		fflush(stdin);
		system("cls");
		cout << "--------Column Permutation cipher(Encryption)--------" << endl;
		cout << "               the null letter is 'x'               " << endl;
		cout << "Please enter the plaintext:";
		cin >> st;
		flag = Check(st);
	} while (flag);


	do {		//输入key
		fflush(stdin);
		cout << "Please enter the key1: ";
		cin >> key;
		flag = Check(key);
	} while (flag);



	//明文不是key的整数倍补'x'
	t = st.length();
	if (t%key.length() != 0)
		t = ((t / (key.length()) + 1)*key.length());
	for (i = 0; i < t; i++) {
		if (i < st.length())
			tem[i] = st[i];
		else
			tem[i] = 'x';
	}


	p = wordKey(key, k);
	//第一次加密
	cout << "After the first step" << endl;

	//打印plain matrix
	cout << "Plaintext Matrix" << endl << endl;
	m = 0;
	n = (strlen(tem)) / (key.length());
	for (i = 0; i < n; i++) {

		for (j = 0; j < key.length(); j++) {
			cipher[i][j] = tem[m];
			cout << " " << cipher[i][j];
			m++;
		}
		cout << endl;
	}
```

```
2177
2178        q = 0;
2179        cout << "Middle CipherText:";
2180        for (i = 0; i < key.length(); i++) {
2181              for (j = 0; j < n; j++) {
2182                    text[q] = cipher[j][(p[i] - 1)];
2183                    cout << cipher[j][(p[i] - 1)];
2184                    q++;
2185              }
2186        }
2187
2188        cout << endl << endl;
2189
2190        do {          //输入key
2191              fflush(stdin);
2192              cout << "Please enter the key2: ";
2193              cin >> key2;
2194              flag = Check(key2);
2195        } while (flag);
2196
2197
2198        //明文不是key的整数倍补'x'
2199        t = strlen(text);
2200        if (t%key2.length() != 0)
2201              t = ((t / (key2.length()) + 1)*key2.length());
2202        for (i = 0; i < t; i++) {
2203              if (i < strlen(text))
2204                    tem2[i] = text[i];
2205              else
2206                    tem2[i] = 'x';
2207        }
2208
2209        p2 = wordKey(key2, k2);
2210
2211
2212        //第二次加密
2213        cout << "After the second step" << endl;
2214
2215        //打印plain matrix
2216        cout << "Plaintext Matrix" << endl << endl;
2217        m = 0;
2218        n = (strlen(tem2)) / (key2.length());
2219        for (i = 0; i < n; i++) {
2220
2221              for (j = 0; j < key2.length(); j++) {
2222                    cipher2[i][j] = tem2[m];
2223                    cout << " " << cipher2[i][j];
2224                    m++;
2225              }
2226              cout << endl;
2227        }
2228        cout << "CipherText:";
2229        for (i = 0; i < key2.length(); i++) {
2230              for (j = 0; j < n; j++) {
2231                    cout << cipher2[j][(p2[i] - 1)];
2232              }
2233        }
2234
2235
2236        cout << endl << endl;
2237        system("PAUSE");
2238  }
2239
2240  void decrydoubleColumn() {
2241        int flag = 0;
2242        int k[20] = { 0 };
2243        int k2[20] = { 0 };
2244        int  i, j, m, n, q, d;
2245        string st, key, key2;
2246        char tem[50] = { 0 };
2247        char tem2[50] = { 0 };
2248        char plain[10][10] = { 0 };
2249        char plain2[10][10] = { 0 };
```

```
2250        int *p, *p2;
2251        do {
2252              fflush(stdin);
2253              system("cls");
2254              cout << "--------Column Permutation cipher(Decryption)--------" << endl;
2255              cout << "Please enter the ciphertext:";
2256              cin >> st;
2257              flag = Check(st);
2258        } while (flag);
2259
2260        //第一次解密
2261        do {          //输入key
2262              fflush(stdin);
2263              cout << "Please enter the key2: ";
2264              cin >> key;
2265              flag = Check(key);
2266        } while (flag);
2267
2268        if (((st.length()) % (key.length())) != 0) {
2269              cout << endl << "Error!The length of the ciphertext must be an integral multiple of the length of the key!"
    << endl << endl;
2270              exit(-1);
2271        }
2272
2273        p = wordKey(key, k);
2274
2275
2276        //组建明文序列
2277
2278        m = 0;
2279        n = (st.length()) / (key.length());
2280        for (i = 0; i < key.length(); i++) {
2281
2282              for (j = 0; j < n; j++) {
2283                    plain[j][(p[i] - 1)] = st[m];
2284                    m++;
2285              }
2286        }
2287
2288        //打印plain matrix
2289        q = 0;
2290        cout << "Plaintext Matrix" << endl << endl;
2291        for (i = 0; i < n; i++) {
2292              for (j = 0; j < key.length(); j++) {
2293                    cout << "  " << plain[i][j];
2294                    q++;
2295              }
2296              cout << endl;
2297        }
2298        cout << endl;
2299
2300        //输出明文
2301        cout << "Middle Plaintext:";
2302        for (i = 0; i < n; i++) {
2303              for (j = 0; j < key.length(); j++) {
2304                    cout << plain[i][j];
2305              }
2306        }
2307        cout << endl << endl;
2308
2309        //第二次解密
2310        do {          //输入key
2311              fflush(stdin);
2312              cout << "Please enter the key1: ";
2313              cin >> key2;
2314              flag = Check(key2);
2315        } while (flag);
2316
2317        i = 0;
2318        d = st.length();
2319        n = (st.length()) / (key.length());
2320        if ((d % key2.length()) != 0) {
2321              //检查第一次加密是否使用了无效字符'x'，若使用则删除字符'x'
```

```
2322            while ((d%key2.length() != 0) && (plain[n - 1][(p[i] - 1)] == 'x') && i < key.length()) {
2323                  plain[n - 1][p[i] - 1] = '\0';
2324                  d--;
2325                  i++;
2326            }
2327            //如果删除字符'x'后仍然不可整除key2，则打印错误信息，提示输入有误
2328            if ((d % key2.length()) != 0) {
2329                  cout << endl << "Error!The key or the ciphertext that you have entered is wrong!" << endl << endl;
2330                  exit(-1);
2331            }
2332            //组建临时中间明文
2333            q = 0;
2334            for (i = 0; i < n; i++) {
2335                  for (j = 0; j < key.length(); j++) {
2336                        if (plain[i][j] != '\0')
2337                              tem[q] = plain[i][j];
2338                        q++;
2339                  }
2340            }
2341      }
2342      else {
2343            q = 0;
2344            for (i = 0; i < n; i++) {
2345                  for (j = 0; j < key.length(); j++) {
2346                        tem[q] = plain[i][j];
2347                        q++;
2348                  }
2349            }
2350      }
2351
2352      p2 = wordKey(key2, k2);
2353      m = 0;
2354      n = (strlen(tem)) / (key2.length());
2355      for (i = 0; i < key2.length(); i++) {
2356
2357            for (j = 0; j < n; j++) {
2358                  plain2[j][(p2[i] - 1)] = tem[m];
2359                  m++;
2360            }
2361      }
2362
2363      //打印plain matrix
2364      cout << "Plaintext Matrix" << endl << endl;
2365      for (i = 0; i < n; i++) {
2366            for (j = 0; j < key2.length(); j++) {
2367                  cout << " " << plain2[i][j];
2368            }
2369            cout << endl;
2370      }
2371      cout << endl;
2372
2373      //输出明文
2374      cout << "Plaintext:";
2375      for (i = 0; i < n; i++) {
2376            for (j = 0; j < key2.length(); j++) {
2377                  cout << plain2[i][j];
2378            }
2379      }
2380      cout << endl << endl;
2381      system("PAUSE");
2382 }
2383 int main() {
2384
2385      system("color 2F");
2386      cout << "=============================================" << endl;
2387      cout << "---------Encryption/Decryption System---------" << endl;
2388      cout << "=============================================" << endl;
2389      menue();
2390
2391      return 0;
2392 }
```