Information Security HW1

分工

四資工三 B10615033 王璽禎 Encrypt
四資工三 B10615035 陳靜萱 Decrypt

開發環境

Visual studio 2017 C++

範例 **Encrypt**：

1. Caesar cipher
   Input:
      Key: 5
      Plaintext: doyourbestandthenletgo
   Output:
      Encrypt: **ITDTZWGJXYFSIYMJSQJYLT**

```
void Caesar(string Key, string plaintext)
{
    int key = stoi(Key);
    for (int i = 0; i < plaintext.size(); i++)
    {
        plaintext[i] += key;
        if (plaintext[i] >= 123)     // 超過'z', -26回到'a'~'z'
            plaintext[i] -= 26;
        plaintext[i] = toupper(plaintext[i]);
    }
    cout << plaintext;
}
```

2. Playfair cipher:
   Input:
      Key: COMP
      Plaintext: doyourbestandthenletgo
   Output:
      Encrypt: **IDWPQSDFTUGUFRKBHNFSDA**

```
void Playfair(string key, string plaintext)
{
    vector<char> alph = { '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'};
    for (int i = 0; i < key.size(); i++)    //I J同一個字
        if (key[i] == 'J') key[i] = 'I';
    // 建立 5*5 matrix
    char matrix[5][5] = {};
    int m = 0, n = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (n < key.size())     // 將key放入matrix
            {
                vector<char>::iterator it;
                for (it = alph.begin(); it != alph.end(); it++)
                {
                    if (*it == key[n]) // key重複字母 跳過
                    {
                        it = alph.erase(it);
                        break;
                    }
                }
                if (it == alph.end())   // 在alph中沒找到, 返回上一格繼續
                {
                    j--; n++;
                }
```

```
                else
                    matrix[i][j] = key[n++];
            }
            else            // 將剩下的字母放入matrix
                matrix[i][j] = alph[m++];
        }
    }
    string ciphertext = "";
    for (int i = 0; i < plaintext.size(); i++)
    {
        char a = plaintext[i] - 32; char b = plaintext[++i] - 32; //重複 插入X (插入後plaintext為偶數?)
        if (a == b)
        {   b = 'X'; i--;   }
        if (a == 'J') a = 'I';
        if (b == 'J') b = 'I';
        int pax = 0, pay = 0, pbx = 0, pby = 0;
        int ck = 0;
        // 從matrix中找字
        for (int x = 0; x < 5; x++)
        {
            for (int y = 0; y < 5; y++)
            {
                if (a == matrix[x][y])
                {
                    pax = x; pay = y;
                    ck++;
                }
```

```
            else if (b == matrix[x][y])
            {
                pbx = x; pby = y;
                ck++;
            }
            if (ck == 2) break;
        }
    }
    // 轉成ciphertext
    if (pay == pby)      //右
    {
        ciphertext += matrix[(pax + 1)%5][pay];
        ciphertext += matrix[(pbx + 1)%5][pby];
    }
    else if (pax == pbx) //下
    {
        ciphertext += matrix[pax][(pay + 1) % 5];
        ciphertext += matrix[pbx][(pby + 1) % 5];
    }
    else                 // 對角
    {
        ciphertext += matrix[pax][pby];
        ciphertext += matrix[pbx][pay];
    }
}
cout << ciphertext;
}
```

3. Vernam proposed the autokey system:

   Input:

   Key: TEC

   Plaintext: doyourbestandthenletgo

   Output:

   Encrypt: **QK[N[JPQDSE`QTKH_MA_NK**

```
void Vernam(string Key, string plaintext)
{
    string key = Key;
    string ciphertext = "";
    for (int i = 0; i < plaintext.size(); i++)
    {
        plaintext[i] = toupper(plaintext[i]);
        key += plaintext[i];    // autokey: key + plaintext
        char temp = (plaintext[i] - 65) ^ (key[i] - 65);    // 變成數字做xor
        temp += 65;
        ciphertext += temp;
    }
    cout << ciphertext;
}
```

4. Row transposition:

   Input:

   Key: 45362178

   Plaintext: doyourbestandthenletgo

   Output:

Encrypt: **RTOUDGYAEDSNOTLONTBHEE**

```
void Row(string Key, string plaintext)
{
    map<int, int> key; // key的數字->對應順序
    int max = 0;
    for (int i = 0; i < Key.size(); i++)
    {
        int num = Key[i] - 48;
        if (num > max) max = num;
        key[num] = i;
    }
    // 建立 row=max 的matrix
    int col = (plaintext.size() / max) + 1;
    vector<vector<char>> matrix;
    matrix.resize(col);
    int n = 0;
    for (int i = 0; i < plaintext.size(); i++)
    {
        matrix[n].push_back(plaintext[i]);
        if (matrix[n].size() == max)
            n++;
    }
```

```
    // 轉成ciphertext
    string ciphertext = "";
    for (int i = 0; i < key.size(); i++)
    {
        int t = 0;
        while (t < col && key[i + 1] < matrix[t].size())
        {
            ciphertext += toupper(matrix[t++][key[i + 1]]);
        }
    }
    cout << ciphertext;
}
```

5. Rail fence cipher:

   Input:

   Key: 2

   Plaintext: doyourbestandthenletgo

   Output:

   Encrypt: **DYUBSADHNEGOORETNTELTO**

```
void Rail_fence(string Key, string plaintext)
{
    int key = stoi(Key);
    vector<vector<char>> fence;
    fence.resize(key);
    for (int i = 0; i < plaintext.size(); i++)
    {
        int len = (key * 2) - 2; // 計算一圈的長度 1->2->3->2 (4)
        int n = i % len;     // 取餘數判斷去程還回程
        if (n < key)     // 去程
            fence[n].push_back(plaintext[i]);
        else             // 回程
            fence[len - n].push_back(plaintext[i]);
    }
    string ciphertext = "";
    for (int i = 0; i < key; i++)
        for (int j = 0; j < fence[i].size(); j++)
            ciphertext += toupper(fence[i][j]);
    cout << ciphertext;
}
```

範例 **Dncrypt**：

1. Caesar cipher

   Input:

   Key: 7

   ciphertext: RLLWNVPUNULCLYNPCLBW

   Output:

   Decrypt: keepgoingnevergiveup

```
if (type == "caesar") {
    string plaintext = "";
    int num_key = 0;
    for (int i = key.size() - 1, j = 1; i >= 0; i--, j *= 10) {
        num_key += j * (key[i] - 48);//將key從string變成int
    }
    for (int i = 0; i < cipher.length(); i++) {
        int p = cipher[i] - 97;
        p -= num_key;//向左移動
        if (p < 0)
            p += 26;//鄉減小於零時做調整
        plaintext += p + 97;//化成英文
    }
    cout << plaintext << "\n";
}
```

2. Playfair cipher:

Input:

Key: COMP

ciphertext: IDWPQSDFTUGUFRKBHNFSDA

Output:

Decrypt: doyourbestandthenletgo

```
else if (type == "playfair") {
    char matrix[5][5];
    for (int i = 0; i < 5; i++)
        for (int j = 0; j < 5; j++)
            matrix[i][j] = ' ';//初始化
    string new_key = "";
    for (int i = 0; i < key.size(); i++) {
        if (key[i] == 'j')
            key[i] = 'i';//把所有j變成i
        size_t found = new_key.find(key[i]);
        if (found == std::string::npos)//去掉key中重複的字母
            new_key += key[i];
    }
    for (int i = 0; i < new_key.size(); i++) {
        matrix[i / 5][i % 5] = new_key[i];//將key放進matrix
    }
```

```
string alph = "abcdefghiklmnopqrstuvwxyz";
int index = 0;
for (int i = new_key.size(); i < 25; i++) {
    while (index < 26) {
        char temp = alph[index];
        index++;
        size_t found = new_key.find(temp);
        if (found == std::string::npos) {
            matrix[i / 5][i % 5] = temp;//將其他字母放進matrix
            break;
        }
    }
}
for (int i = 0, j = 1; i < cipher.size() - 1; i += 2, j += 2) {
    if (cipher[i] == 'j')//把cipher中的j變成i
        cipher[i] = 'i';
    if (cipher[j] == 'j')
        cipher[j] = 'i';
}
```

```
int i_x = 0, i_y = 0, j_x = 0, j_y = 0;
for (int i = 0, j = 1; i < cipher.size() - 1; i += 2, j += 2) {
    for (int m = 0; m < 25; m++) {
        if (matrix[m / 5][m % 5] == cipher[i])
            i_x = m / 5, i_y = m % 5;//找到第一個字母的位置
        if (matrix[m / 5][m % 5] == cipher[j])
            j_x = m / 5, j_y = m % 5;//找到第二個字母的位置
    }
    if (i_x != j_x && i_y != j_y) {//當兩個字母不同行列，+=對角線字母
        plaintext += matrix[i_x][j_y];
        plaintext += matrix[j_x][i_y];
    }
    else if (i_x == j_x && i_y != j_y) {//當兩個字母同列，+=左邊字母
        if (i_y - 1 < 0)
            i_y = 5;
        if (j_y - 1 < 0)
            j_y = 5;
        plaintext += matrix[i_x][i_y - 1];
        plaintext += matrix[j_x][j_y - 1];
    }
    else if (i_x != j_x && i_y == j_y) {//當兩個字母同列，+=上面字母
        if (i_x - 1 < 0)
            i_x = 5;
        if (j_x - 1 < 0)
            j_x = 5;
        plaintext += matrix[i_x - 1][j_y];
        plaintext += matrix[j_x - 1][i_y];
    }
}
cout << plaintext << "\n";
```

3. Vernam proposed the autokey system:

   Input:

   Key: TEC

   ciphertext: QK[N[JPQDSE`QTKH_MA_NK

   Output:

   Decrypt: doyourbestandthenletgo

```
else if (type == "vernam") {
    string new_key = "";
    string plaintext = "";
    for (int i = 0; i < cipher.size(); i++)
        cipher[i] = toupper(cipher[i]);//將cipher變大寫
    for (int i = 0; i < key.size(); i++)
        key[i] = toupper(key[i]);//將key變大寫
    int len = key.size();
    while (cipher.size() != 0)
    {
        string temp = "";
        if (cipher.size() < len)
            len = cipher.size();//最後一次轉換的時候計算要做幾個字
        for (int i = 0; i < len; i++)
            temp += (cipher[i] - 65 ^ key[i] - 65) + 65;//做xor轉換
        key = temp;
        cipher.erase(0, len);//將做完的字刪除
        plaintext += temp;
    }
    for (int i = 0; i < plaintext.size(); i++)
        plaintext[i] = tolower(plaintext[i]);//把plaintext變小寫
    cout << plaintext << "\n";
}
```

4. Row transposition:

   Input:

   Key: 45362178

   ciphertext: RTOUDGYAEDSNOTLONTBHEE

   Output:

   Decrypt: doyourbestandthenletgo

```
else if (type == "row") {
    string plaintext = "";
    int c = key.size(), r = cipher.size() / c;
    if (cipher.size() % c == 0)//計算要幾個row
        r = cipher.size() / c;
    else
        r = cipher.size() / c + 1;
    char **matrix = (char **)malloc(sizeof(char*) * r);
    for (int i = 0; i < r; i++)
        matrix[i] = (char(*))malloc(sizeof(char) * c);
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            matrix[i][j] = ' ';//初始化
    for (int i = 0; i < cipher.size(); i++) {
        matrix[i / c][i%c] = cipher[i];//把cipher放進matrix
    }
    for (int i = 1; i <= key.size(); i++) {
        size_t found = key.find(i + 48);//按照順序找到對應的數字
        int f = found;
        for (int j = 0; j < r; j++)
            if (matrix[j][found] != ' ')//如果matrix不為空，把字母放進matrix中
                matrix[j][found] = cipher[0], cipher.erase(0, 1);
    }
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            plaintext += matrix[i][j];
    cout << plaintext << "\n";
}
```

5. Rail fence cipher:

   Input:

   Key: 2

   ciphertext: DYUBSADHNEGOORETNTELTO

   Output:

   Decrypt: doyourbestandthenletgo

```
else if (type == "rail_fence") {
    string plaintext = "";
    int num_key = 0;
    for (int i = key.size() - 1, j = 1; i >= 0; i--, j *= 10) {
        num_key += j * (key[i] - 48);//將key從string變int
    }
    char **matrix = (char **)malloc(sizeof(char*) * num_key);
    for (int i = 0; i < num_key; i++)
        matrix[i] = (char(*))malloc(sizeof(char) * cipher.size());
    for (int i = 0; i < num_key; i++)
        for (int j = 0; j < cipher.size(); j++)
            matrix[i][j] = ' ';//初始化
    int command = 0, index = 0;
    for (int i = 0; i < cipher.size(); i++)//把cipher用frence放進matrix
    {
        matrix[index][i] = cipher[i];
        if (command == 0) {
            index++;
            if (index == num_key - 1)
                command = 1;
        }
        else if (command == 1) {
            index--;
            if (index == 0)
                command = 0;
        }
    }
```

```
int *num = (int(*))malloc(sizeof(int) * num_key);
string split[100];
for (int i = 0; i < num_key; i++) {
    int count = 0;
    for (int j = 0; j < cipher.size(); j++)
        if (matrix[i][j] != ' ')//計算每個row有幾個字
            count++;
    num[i] = count;
}
int len = cipher.size();
for (int i = 0; i < num_key; i++) {
    split[i] = split[i].append(cipher, 0, num[i]);//按照上面計算的數字分割cipher
    cipher.erase(0, num[i]);
}
```

```
command = 0, index = 0;
for (int i = 0; i < len; i++)//用fence的規則把字母放到plaintext
{
    plaintext += split[index][0];
    split[index].erase(0, 1);
    if (command == 0) {
        index++;
        if (index == num_key - 1)
            command = 1;
    }
    else if (command == 1) {
        index--;
        if (index == 0)
            command = 0;
    }
}
cout << plaintext << "\n";
```