TUGAS 2

PRAKTIKUM KRIPTOGRAFI

Laura Azra Aprilyanti 140810200036

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2022

Program

```
/*
Nama        = Laura Azra Aprilyanti
NPM         = 140810200036
Deskripsi   = Program Hill Cipher
*/

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

int key[3][3]; // Sebagai kunci global batas ordo matriks

// fungsi modulo
int mod26(int x){
    return x >= 0 ? (x % 26) : 26 - (abs(x) % 26);
}

// Mencari determinan matriks
int cariDet(int m[3][3], int n){
    int det;
    if (n == 2){ // jika ordo matriks = 2
        det = m[0][0] * m[1][1] - m[0][1] * m[1][0];
    }
    else
        det = 0; // invalid input
    return mod26(det);
```

```
}


// Mencari invers matriks
int cariDetInv(int R, int D = 26){
    // R = sisa atau determinan
    int i = 0;
    int p[100] = {0, 1};
    int q[100] = {0}; // hasil bagi

    while (R != 0){ // jika sisa tidak sama dengan 0
        q[i] = D / R;
        int oldD = D;
        D = R;
        R = oldD % R;
        if (i > 1){
            p[i] = mod26(p[i - 2] - p[i - 1] * q[i - 2]);
        }
        i++;
    }

    if (i == 1)
        return 1;
    else
        return p[i] = mod26(p[i - 2] - p[i - 1] * q[i - 2]);
}


int gcd(int m, int n){
```

```
    if (n > m)
        swap(m, n);

    do{
        int temp = m % n;
        m = n;
        n = temp;
    } while (n != 0);
    return m;
}

void multiplyMatrices(int a[1000][3], int a_baris, int a_kolom, int
b[1000][3], int b_baris, int b_kolom, int res[1000][3]){
    for (int i = 0; i < a_baris; i++){
        for (int j = 0; j < b_kolom; j++){
            for (int k = 0; k < b_baris; k++){
                res[i][j] += a[i][k] * b[k][j];
            }
            res[i][j] = mod26(res[i][j]);
        }
    }
}

void findKey(){
    // deklarasi
    string plainteks, cipherteks;
    int key[2][2], det, detInv, adj[2][2], plainTeksInv[2][2],
plainMatrix[2][2], CMatriks[2][2], counter;
```

```
    int p, c;
    int transpose[2][2];

    // input plainteks
    cout << "Masukkan Plainteks : ";
    cin.ignore();
    getline(cin, plainteks);

    // menetapkan plainteks ke plainMatrix
    counter = 0;
    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            p = toupper(plainteks[counter]) - 65;
            plainMatrix[i][j] = p;
            counter++;
        }
    }

    // input cipherteks
    cout << "Masukkan Cipherteks : ";
    getline(cin, cipherteks);

    // menetapkan cipherteks ke CMatriks
    counter = 0;
    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            c = toupper(cipherteks[counter]) - 65;
```

```
            CMatriks[i][j] = c;
            counter++;
        }
    }

    // determinan
    det = (plainMatrix[0][0] * plainMatrix[1][1]) - (plainMatrix[0][1] *
plainMatrix[1][0]);
    if (gcd(det, 26) == 1){
        // inverse dari determinan mod 26
        detInv = cariDetInv(det, 26);

        // mencari adjoin
        adj[0][0] = plainMatrix[1][1];
        adj[0][1] = (-1) * plainMatrix[0][1];
        adj[1][0] = (-1) * plainMatrix[1][0];
        adj[1][1] = plainMatrix[0][0];

        // menghitung matriks invers dari plainteks
        for (int i = 0; i < 2; i++){
            for (int j = 0; j < 2; j++){
                plainTeksInv[i][j] = detInv * adj[i][j];
                if (plainTeksInv[i][j] < 0){
                    plainTeksInv[i][j] = 26 - (abs(plainTeksInv[i][j]) %
26);
                }else{
                    plainTeksInv[i][j] = plainTeksInv[i][j];
                    plainTeksInv[i][j] = plainTeksInv[i][j] % 26;
```

```
            }
        }
    }

    // Search key
    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            key[i][j] = 0;
            for (int k = 0; k < 2; k++){
                key[i][j] += (plainTeksInv[i][k] * CMatriks[k][j]);
            }
            key[i][j] %= 26;
        }
    }

    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            transpose[j][i] = key[i][j];
        }
    }

    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            cout << (transpose[i][j]) << "\t";
        }
        cout << endl;
    }
```

```
    }else{
        cout << "Determinan tidak relatif " << endl;
        cout << "Kunci tidak ditemukan" << endl << endl;
    }
}

/* Invers = (matriks * det Invers) mod 26 */
/* cari Invers(matrix , order_of_matrix , result_matrix) */
void CariInvers(int m[3][3], int n, int m_inverse[3][3]){
    int adj[3][3] = {0};
    int det = cariDet(m, n); // ini menggunakan fungsi cariDet(matrix , order_of_matrix)
    int detInverse = cariDetInv(det);

    if (n == 2){ // jika ordo matrik 2x2
        adj[0][0] = m[1][1];
        adj[1][1] = m[0][0];
        adj[0][1] = -m[0][1];
        adj[1][0] = -m[1][0];
    }

    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            m_inverse[i][j] = mod26(adj[i][j] * detInverse);
        }
    }
}
```

```
// C = PK
string encrypt(string pt, int n){
    int P[1000][3] = {0}; // plaintext
    int C[1000][3] = {0}; // cipher text
    int ptIter = 0;

    while (pt.length() % n != 0){
        pt += "x"; // pad extra x, ini digunakan jika plaintext di module dengan matrik tidak sama dengan 0
    }
    int baris = (pt.length()) / n; // jumlah baris dalam plaintext

    for (int i = 0; i < baris; i++){
        for (int j = 0; j < n; j++){
            P[i][j] = pt[ptIter++] - 'a';
        }
    }

    // multiplyMatrices(mat_a , baris_a , kolom_a , mat_b, baris_b, kolom_b , mat_result)
    multiplyMatrices(P, baris, n, key, n, n, C);

    string ct = "";
    for (int i = 0; i < baris; i++){
        for (int j = 0; j < n; j++){
            ct += (C[i][j] + 'a');
        }
```

```
    }
    return ct;
}

// P = C*(KeyInvers)
string decrypt(string ct, int n){
    int P[1000][3] = {0}; // plain text
    int C[1000][3] = {0}; // cipher text
    int ctIter = 0;

    int baris = ct.length() / n; // banyak baris di chipertext

    for (int i = 0; i < baris; i++){
        for (int j = 0; j < n; j++){
            C[i][j] = ct[ctIter++] - 'a';
        }
    }

    int KeyInverse[3][3] = {0};
    /* CariInvers(matrix , order_of_matrix , result_matrix) */
    CariInvers(key, n, KeyInverse);

    /* multiplyMatrices(mat_a , baris_a , kolom_a , mat_b, baris_b,
kolom_b , mat_result) */
    multiplyMatrices(C, baris, n, KeyInverse, n, n, P);

    string pt = "";
```

```
    for (int i = 0; i < baris; i++){
        for (int j = 0; j < n; j++){
            pt += (P[i][j] + 'a');
        }
    }
    return pt;
}

int main(void){
    bool menu = true;
    string pt, ct;
    int n;
    int pilih;

    while (menu){
        cout << "\nProgram Hill Cipher" << endl;
        cout << "Menu : " << endl;
        cout << "1. Enkripsi" << endl;
        cout << "2. Dekripsi" << endl;
        cout << "3. Find Key" << endl;
        cout << "4. Exit" << endl;
        cout << "Pilih Menu : ";
        cin >> pilih;
        switch (pilih){
        case 1:
            cout << "Masukkan kata   : ";
            cin >> pt;
```

```
        cout << "Masukkan ordo matriks harus 2x2 : ";
        cin >> n;

        for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++){
                cout << "Masukkan matriks : ";
                cin >> key[i][j];
            }
        }

        cout << "\nPlaintext  : " << pt << endl;

        ct = encrypt(pt, n);
        cout << "Hasil Enkripsi : " << ct << endl;
        break;
    case 2:
        cout << "Masukkan kata   : ";
        cin >> ct;

        cout << "Masukkan ordo matriks harus 2x2 : ";
        cin >> n;

        for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++){
                cout << "Masukkan matriks : ";
                cin >> key[i][j];
```

```
                }
            }

            cout << "\nChipertext  : " << ct << endl;
            cout << "Hasil Dekripsi : " << decrypt(ct, n) << endl;
            break;
        case 3:
            cout << endl;
            findKey();
            break;
        default:
            cout << "\nInvalid Input" << endl;
            break;
        }
    }
}
```

Capture Run Program

- **Enkripsi**

```
Program Hill Cipher
Menu :
1. Enkripsi
2. Dekripsi
3. Find Key
4. Exit
Pilih Menu : 1
Masukkan kata   : laura
Masukkan ordo matriks harus 2x2 : 2
Masukkan matriks : 5
Masukkan matriks : 8
Masukkan matriks : 17
Masukkan matriks : 3

Plaintext  : laura
Hasil Enkripsi : dkzdbr
```

Terdapat 1 huruf tambahan, karena huruf yang ada berjumlah ganjil

- **Dekripsi**

```
Program Hill Cipher
Menu :
1. Enkripsi
2. Dekripsi
3. Find Key
4. Exit
Pilih Menu : 2
Masukkan kata   : dkzdbr
Masukkan ordo matriks harus 2x2 : 2
Masukkan matriks : 5
Masukkan matriks : 8
Masukkan matriks : 17
Masukkan matriks : 3

Chipertext  : dkzdbr
Hasil Dekripsi : laurax
```

Huruf lebih ditandai sebagai x

- **Cari key matriks**

```
Program Hill Cipher
Menu :
1. Enkripsi
2. Dekripsi
3. Find Key
4. Exit
Pilih Menu : 3

Masukkan Plainteks : laurax
Masukkan Cipherteks : dkzdbr
5       17
8       3
```

Penjelasan

- Fungsi cariDet merupakan fungsi yang bertugas mencari determinan dari matriks key, jika matriks memiliki ordo 2x2 maka akan dihitung, dan jika tidak maka akan invalid.
- Fungsi cariDetInv erupakan fungsi untuk mencari invers dari determinan matriks key.
- Fungsi mod digunakan untuk memodulokan rumus
- Fungsi cariInv digunakan untuk mencari invers matriks.
- Fungsi enkripsi untuk mengenkripsi plaintext. Jika huruf%ordo matriks == 0 maka akan diubah menjadi matriks dan dikalikan key. Jika kurang akan ditambahkan x pada bagian belakang plaintext. setelah itu di enkripsi dan ditambahkan ‘a’ agar sesuai dengan Ascii.
- Fungsi dekripsi digunakan sama dengan fungsi ekripsi.
- Fungsi gcd digunakan untuk memastikan gcd harus =1
- Fungsi cariKey digunkan untuk menemukan kunci dari plainteks dan ciphertext. Jika gcd = 1 maka proses berlanjut, jik tida maka determinan tidak relatif kunci tidak ditemukan
- Kekurang pada program:
    - Hanya dapat memproses 1 kata
    - Jika kata memiliki jumlah huruf ganjil, maka akan ditambahkan x
    - Tidak dapat melakukan proses untuk matriks berordo yang tidak sama dengan 2x2
    - Huruf yang digunakan harus huruf kecil