Titkosírás haladóknak I.

(Cipher - exercises for advanced users - Part I)

Pitlik Marcell, Pitlik László, Pitlik Mátyás, Pitlik László (jun) – MY-X team

Kivonat: A titkosírások története számtalan megoldást ismer és vélelmezhetően végtelen sok trükk alakítható ki szövegek a triviális olvashatóságának nehezítésére. A három részes cikksorozat első részében bemutatandó kódoló, majd a második részben prezentálandó dekódoló eljáráshoz az alapötletet a mobiltelefonok tárcsázási logikája adta, ahol egy gombon több betű is elhelyezkedik. A harmadik rész a kódoló program finomhangolásáról szól, vagyis a kezdő és a haladó programozók szemlélet-különbségei kerülnek példaértékűen bemutatásra. Egy rel. nehéz titkosírás esetén elvárható, hogy a szavak határát ne lehessen felismerni, ill. az is elvárható, hogy egyes betűket/jeleket ne csak egy jelsorozat helyettesítsen, ill. lehetőség szerint legyen a titkosírásban értelmetlen/zavaró jelből is valamennyi – minél inkább véletlenszerűnek tűnő módon, sőt egy-egy üzenet ne azonos jelsorozattá alakuljon a titkosítás során mindenkor lehetőség szerint. A mellékelt titkosító demo egy C++ állomány, melyet a repl.it oldalon lehet futtatni. A cikk tartalmaz demo-üzeneteket is a titkosítási folyamat input-output-viszonyainak demonstrálására.

Kulcsszavak: titkosírás tervezése, titkosírások nehézségi szintjei

Abstract: The history of the ciphering offers a lot of solutions and theoretically, quasi unlimited ideas should be existing for causing problems during the interpretation of messages. This article is one of series having three parts. The first part presents a cipher algorithm created by an author having just a little experience in programming. The part II. demonstrates the appropriate decipher algorithm where the basic idea was delivered through the old-style mobile phone’s keyboard offering more letters per button. The last (third) part shows the same functionality like in the first part – however this source code got realized by a professional author. Based on the comparing of the more and less trained thinking methodologies (of more and less optimized source codes), it is possible to demonstrate the development possibilities concerning abstraction/complexity of approximations. A rel. robust cipher ensures that the starting and/or ending position of words can not be identified. It is also a sign of higher complexity of ciphering, if one particular letter in the raw message will be substituted through more than one code. Besides, it is necessary to have meaningless/random parts in the ciphered messages. Parallel, it is a more complex level, when the same message can be described through different ciphers in an identical way. The presented demo is a C++ program which can be used through the online services of repl.it. The article also includes demo messages and their ciphered versions.

Keywords: planning of ciphering, levels of ciphering

# Bevezetés

Előzményként említésre méltó egy új évi jó kívánság 2013-ból: <https://miau.my-x.hu/miau2013_2014.png>, melyre megfejtés azóta sem érkezett – bár vélhetően csak azért nem, mert nem feltétlenül áll az érdeklődés középpontjában egy olyan kép, mely kódtartalmára szinte semmi nem utal egyébként (<https://miau.my-x.hu/miau2009/index.php3?x=news>).

Friss impulzusként említendő, hogy 2018-ban a KJE sikeresen akkreditálta a BPROF képzését (üzemmérnök informatikus területen), mely képzés a tananyagfejlesztés kapcsán jelentős fejlesztéseket vár el. Ennek a folyamatnak a szerves része ez a cikk és ennek folytatása. Hiszen a BA és a BSC képzések mellett egyedülálló BPROF képzés gyakorlatiassága megkérdőjelezhetetlen kell, hogy legyen. Az akkreditáció során két specializáció került kialakításra: az agrárinformatika és az IT-biztonság, mely utóbbi értelemszerűen kötődik a titkosírásokhoz is – még esetlegesen, ha csak gamifikációs effektként és/vagy szemléletformáló erőtérként. A kódírás és a kódtörés a leendő programozók/programtervezők/rendszerüzemeltetők számára vélhetően kellően érdekes terület ahhoz, hogy egy hagyományos példánál nagyobb motiváltságot lehessen ezen keresztül kiváltani az érintett Hallgatóságból.

További részletek: <https://miau.my-x.hu/miau/253/titkosiras/>

# Üzenetek

Természetesen, aki vérbeli kódfejtő/kódíró, annak az alábbi üzeneteket előbb illik saját hatáskörben megfejtenie, mielőtt tovább olvassa a cikket:

Üzenet1:

00000000000100072516061170002116131040002102000702150000912190002012160140065000802120200060051411723018181401070030000711506101750280061190618800082000412318408130500009313000811209018000800510301420151202200050170040053111780518000460311411650320616060008140604900040319180005000703600313171036000410803790081305051300033010700007160310400300315103156000211801093630188030500020792015505106006000331500051261040151611070000902318418031511506700088020051205140004161310411411170004060310417310071181140003120003300650401608111111111

Üzenet2:

00000000000140060061130000908121600081840079160160180004603114000480161022000091189363118000513114010908118803170003131351603106512107020600211500050202201216000540604900009016136301800061506111911236000801200020431123130711353018000096301600052911610616001740852000201841179061060180000700219108000601701301603150031060000710171401537815180000911511650003061310410650321614000610406140009000361040161130360008311419307000070516030165121070004811301413600061677910303108100943018116000410506131160050028000423000091062070008116016017118000811417000481341028000305310711601220000760600510310179118143008106000250137900041126014115160700062000811301305131165140069020017800031141017000316030045211840801616

A megoldásokat természetesen a cikk maga is tartalmazza.

# A kódoló eljárás

Az alábbi (tudatosan nem a végső készültségi állapotot jelentő) kód tovább-gondolásra hívja fel az Olvasókat és maguk az üzenetek is a fejlesztőmunka egy adott állapotában felmerült gondolatokat osztják meg az Érdeklődőkkel:

// Example program

#include <iostream>

#include <string>

#include <map>

#include <sstream>

#include <bitset>

#include <stdlib.h>

#include <time.h>

using namespace std;

struct kod2 {

 int mod;

 int kong;

 int tarcsa;

 kod2(){};

 kod2(int m, int k, int t) {

 mod = m;

 kong = k;

 tarcsa = t;

 }

};

/\*map<char,kod2> kod;

struct kodolo {

 //int forgatas;

 string uzenet;

 kodolo(string u) {

 uzenet = u;

 }

 string kodol() {

 ide ki kene meg valahogy szedni

 na meg a tobbi alegyseget is

 }

};\*/

int main()

{

 srand (time(NULL));

 string uzenet;

 string initdatum;

 cout << "Mi az uzenet?"<< endl;

 getline (cin, uzenet);

 cout << "Esedekesseg datuma? (MM/DD)"<< endl;

 getline (cin, initdatum);

 stringstream sho, snap;

 int ho,nap,nap1,nap2;

 sho << initdatum[0] << initdatum[1];

 sho >> ho;

 snap << initdatum[3] << initdatum[4];

 snap >> nap;

 nap1=nap/10;

 nap2=nap%10;

 string bho = bitset<4>(ho).to\_string();

 string bnap1 = bitset<2>(nap1).to\_string();

 string bnap2 = bitset<4>(nap2).to\_string();

 map<char,kod2> kod;

 kod['a']= {3,0,2};

 kod['b']= {3,1,2};

 kod['c']= {3,2,2};

 kod['d']= {3,0,3};

 kod['e']= {3,1,3};

 kod['f']= {3,2,3};

 kod['g']= {3,0,4};

 kod['h']= {3,1,4};

 kod['i']= {3,2,4};

 kod['j']= {3,0,5};

 kod['k']= {3,1,5};

 kod['l']= {3,2,5};

 kod['m']= {3,0,6};

 kod['n']= {3,1,6};

 kod['o']= {3,2,6};

 kod['p']= {4,0,7};

 kod['q']= {4,1,7};

 kod['r']= {4,2,7};

 kod['s']= {4,3,7};

 kod['t']= {3,0,8};

 kod['u']= {3,1,8};

 kod['v']= {3,2,8};

 kod['w']= {4,0,9};

 kod['x']= {4,1,9};

 kod['y']= {4,2,9};

 kod['z']= {4,3,9};

 cout << endl << "A feldolgozott uzenet:" << endl << '"' << uzenet << '"' << endl;

 int forgatas;

 stringstream ss;

 for (int i=0; i<uzenet.length(); i++) {

 char betu = uzenet[i];

 if (betu != 32 && (betu<97 || betu>122)) {

 cout << "ERROR: UNEXPECTED CHAR!" << endl << "Csak az angol abc kisbetui megengedettek!" << endl;

 exit(1);

 }

 //cout << betu << endl;

 if (isspace(betu)) {

 int sprand = rand()%8 + 1;

 map<int,string> spkod;

 spkod[1]= "0002";

 spkod[2]= "0003";

 spkod[3]= "0004";

 spkod[4]= "0005";

 spkod[5]= "0006";

 spkod[6]= "00007";

 spkod[7]= "0008";

 spkod[8]= "00009";

 ss << spkod[sprand];

 //cout << ss.str() << endl;

 } else {

 forgatas = 0;

 for (int j=0; j<ss.str().length(); j++) {

 if (ss.str()[j] == '0' || ss.str()[j] == '1') {

 forgatas++;

 }

 }

 //cout << forgatas % kod[betu].mod << endl;

 while (forgatas % kod[betu].mod != kod[betu].kong) {

 bool forrand = rand()%2;

 ss << forrand;

 forgatas++;

 //cout << ss.str() << endl;

 }

 ss << kod[betu].tarcsa;

 //cout << ss.str() << endl;

 }

 }

 while (forgatas % 12 != 0) {

 ss << "1";

 forgatas++;

 }

 cout << "A kodolt uzenet:" << endl << 0 << bho << bnap1 << bnap2 << 1 << ss.str() << endl;

 return 0;

}

# Futtatási keretek

A repl.it oldalon a jobb felső sarokban látható „new repl” gomb mögött lehet választani programozási nyelvet (jelen esetben a C++-t). Majd a felkínált kódszerkesztési felületre a már kész (jelen esetben a titkosírást előállító) kód kerül bemásolásra. A „run” gomb (középen felül zöld háttérrel) teszi majd lehetővé magát a futtatást a megfelelő előkészítő lépések után. Amint fut a program, akkor a dialógus logikáját követve kell bemásolni a kódolandó üzenetet, mely alapvetően egyelőre az angol karakterkészletet engedi meg. Ugyanazon üzenet nem ugyanazon kóddá fordul minden esetben, vagyis egy üzenetnek több alternatív kódnézete is van:







# Konklúziók

Az oktatás gamifikálása alapvető kihívásként nehezedik az oktatási rendszerre. Látszat célok esetén a motiváció alábbhagy a Hallgatóságban. Valódi (piacképes/innovatív) célok esetén pedig a megfutamodás eseteivel állhat olykor/gyakran szemben az oktató. Jelen esetben gyakorló feladatként a kódolás/dekódolás érdekes játék – valódi innováció azonban akkor keletkezik az újabb és újabb kódolási logikákon túl, ha a már kódolt (ismeretlen logikájú) üzenet visszafejtésére alkalmas egyre inkább univerzális robot kerül kialakításra (vö. KNUTH: tudás az, ami forráskódba átírható – minden más emberi aktivitás művészet…)

A három rész egyben is letölthető: <https://miau.my-x.hu/miau/253/cipher1-2-3.docx>