ATM_Vigenere

Program ATM_Viegenere to program służący do szyfrowania i deszyfrowania plików tekstowych z wykorzystaniem algorytmu Viegenere'a.

Wprowadzenie do algorytmu Viegenere

Algorytm Viegenere'a to jeden z klasycznych algorytmów szyfrujących. Jego działanie jest oparte na tablicy określanej nazwą tabulaRecta.

tabulaRecta:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Każdy wiersz tablicy odpowiada szyfrowi Cezara, przy czym w pierwszym wierszu przesunięcie wynosi 0, w drugim 1 itd.

Aby zaszyfrować tekst potrzebne jest słowo kluczowe. Słowo kluczowe jest tajne i mówi z którego wiersza (lub kolumny) należy w danym momencie skorzystać.

Załóżmy, że chcemy zaszyfrować następujący tekst:
TO JEST BARDZO TAJNY TEKST

Do tego użyjemy znanego tylko nam słowa kluczowego np. KLUCZ
Słowo to jest krótsze niż niż tekst to zaszyfrowania więc używamy jego wielokrotności:
TO JEST BARDZO TAJNY TEKST
KL UCZK LUCZKL UCZKL UCZKL

Następnie wykonujemy szyfrowanie w następujący sposób: litera szyfrogramu odpowiada literzez tabeli znajdującej się na przecięciu wiersza, wyznaczonego przez literę tekstu jawnego i kolumny wyznaczonej przez literę słowa kluczowego, np. „T" i „K" daje „D", „O" i „L" daje „Z" itd.
W efekcie otrzymujemy zaszyfrowany tekst:
DZ DGRD MUTCJZ NCIXJ NGJCE

Odszyfrowanie przebiega bardzo podobnie. Bierzemy kolejne litery szyfrogramu oraz odpowiadające im litery słowa kluczowego (podobnie jak przy szyfrowaniu). Wybieramy kolumnę (w tabulaRecta) odpowiadającą literze słowa kluczowego. Następnie w tej kolumnie szukamy litery szyfrogramu. Numer wiersza odpowiadający znalezionej literze jest numerem litery tekstu jawnego. np.
DZ DGRD MUTCJZ NCIXJ NGJCE
KL UCZK LUCZKL UCZKL UCZKL
TO JEST BARDZO TAJNY TEKST

Opis użytkowania programu

ATM_Viegenere wykorzystuje sześć znaczników które podajemy w terminalu. Musimy również podać nazwę pliku wejściowego i pliku wyjściowego.
-e => oznacza, że program będzie wykorzystywany w trybie szyfrowania
-d => oznacza, że program będzie wykorzystywany w trybie deszyfrowania
-f => jest to znacznik po którym podajemy plik wejściowy (z tekstem zaszyfrowanym dla -d i z tekstem jawnym dla -e)
-o => jest to znacznik po którym podajemy plik wyjściowy (z tekstem zaszyfrowanym dla -e i z tekstem jawnym dla -d)
-k => jest to znacznik po którym podajemy tekst klucza (dużymi bądź małymi literami)
-a => tryb kryptoanalizy. Program wyświetla tabulaRecta dzięki czemu możemy wizualnie kontrolować prawidłowość szyfrowania/deszyfrowania programu.

Znacznik -p wyświetla pomoc.

Przykładowa komenda dla szyfrowania:
./ATM_Vigenere -e -f plikZTekstemJawnym.txt -o plikZZaszyfrowanymTekstem.txt -k klucz

Przykładowa komenda dla deszyfrowania:
./ATM_Vigenere -d -f plikZZaszyfrowanymTekstem.txt -o plikZTekstemJawnym.txt -k klucz

Opis budowy programu

Program został podzielony na 5 plików.
ATM_Viegenere.c => to program rozruchowy wykorzystujący funkcje z pozostałych plików. Zawiera „logikę” znaczników, funkcję wyświetlającą pomoc i wywołuje fukcję silnika szyfrującego
tabulaRecta.h => plik nagłówkowy zawierający prototyp funkcji generującej tablicę tabulaRecta i prototyp funkcji wyświetlającej tabulaRecta
tabulaRecta.c => plik implementujący generację tabulaRecta i wyświetlania tabulaRecta
silnikSzyfrujacy.h => plik nagłówkowy zawierający prototypy trzech funkcji:
- podfunkcja *szyfrowanie*
- podfunkcja *deszyfrowanie*
- funkcja agregująca dwie powyższe *silnikSzyfrujący*

silnikSzyfrujący.c => implementacja funkcji *szyfrowanie*, *deszyfrowanie* i *silnikSzyfrujący*

Listingi plików

(ATM_Viegenere.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tabulaRecta.h"
#include "silnikSzyfrujacy.h"

void pomoc()
{
    printf("Szyfr Vigenere.\n");
    printf("-f => plik wejściowy\n");
    printf("-o => plik wyjściowy\n");
    printf("-e => szyfrowanie\n");
    printf("-d => deszyfrowanie\n");
    printf("-k => klucz\n");
    printf("-a => kryptoanaliza\n");
    printf("-p => pomoc");

}

int main(int argc, char **argv)
{
    char operacja = '0';
    char nazwaPlikuWejsciowego[50];
    char nazwaPlikuWyjsciowego[50];
    char klucz[100] = "0";
    char kryptoanaliza = '0';
    for (int i=0; i<argc; i++)
    {
        if (strcmp(argv[i], "-d") == 0)
        {
            operacja='d';
        }
        if (strcmp(argv[i], "-e") == 0)
        {
            operacja='e';
        }
        if (strcmp(argv[i], "-f") == 0)
        {
            strcpy(nazwaPlikuWejsciowego, argv[i+1]);
        }
        if (strcmp(argv[i], "-o") == 0)
        {
            strcpy(nazwaPlikuWyjsciowego, argv[i+1]);
        }
        if (strcmp(argv[i], "-k") == 0)
        {
            strcpy(klucz, argv[i+1]);
        }
        if (strcmp(argv[i], "-a") == 0)
        {
            kryptoanaliza='1';
        }
        if (strcmp(argv[i], "-p") == 0)
        {
            pomoc();
        }
        //printf("%s\n", argv[i]);
    }

    silnikSzyfrujacy(operacja, nazwaPlikuWejsciowego, nazwaPlikuWyjsciowego, klucz, kryptoanaliza);
    return 0;
}
```

(tabulaRecta.h)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_ROWS 26
#define MAX_COLS 26

void generujTabulaRecta(char tabulaRecta[MAX_ROWS][MAX_COLS]);

void wyswietlTabulaRecta(char tabulaRecta[MAX_ROWS][MAX_COLS]);
```

(tabulaRecta.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_ROWS 26
#define MAX_COLS 26

void generujTabulaRecta(char tabulaRecta[MAX_ROWS][MAX_COLS])
{
   int pos, row, col;

   for (row=0; row<MAX_ROWS; row++)
   {
      pos = 0;
      for (col=0; col<MAX_COLS; col++)
      {
         if ('A'+row+col > 'Z' && row > 0)
            {
               tabulaRecta[row][col] = 'A'+pos;
               pos++;
            }
         else if  ('A' + row+col == 'Z')
            {
               tabulaRecta[row][col] = 'Z';
            }
         else
            {
               tabulaRecta[row][col] = 'A'+row+col;
            }
      }
   }
}

void wyswietlTabulaRecta(char tabulaRecta[MAX_ROWS][MAX_COLS])
{
   int row, col;
   printf("Tabula Recta:\n");
   for (row=0; row<MAX_ROWS; row++)
   {
      for (col=0; col<MAX_COLS; col++)
      {
         printf("%c", tabulaRecta[row][col]);
      }
      printf("\n");
   }
}
```

(silnikSzyfrujacy.h)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tabulaRecta.h"

void szyfrowanie(char operacja,
                 char nazwaPlikuWejsciowego[],
                 char nazwaPlikuWyjsciowego[],
                 char klucz[]);

void deszyfrowanie(char operacja,
                 char nazwaPlikuWejsciowego[],
                 char nazwaPlikuWyjsciowego[],
                 char klucz[]);

void silnikSzyfrujacy(char operacja,
                    char nazwaPlikuWejsciowego[],
                    char nazwaPlikuWyjsciowego[],
                    char klucz[],
                    char kryptoanaliza);
```

(silnikSzyfrujacy.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "tabulaRecta.h"


// Funkcja szyfrująca
int szyfrowanie(char operacja,
                char nazwaPlikuWejsciowego[],
                char nazwaPlikuWyjsciowego[],
                char klucz[])
{
   char tabulaRecta[MAX_ROWS][MAX_COLS];
   generujTabulaRecta(tabulaRecta);
   int keyLenght = strlen(klucz);

   // Otwarcie pliku z tekstem do zaszyfrowania
      FILE *plikWe = fopen(nazwaPlikuWejsciowego, "r");
      if (plikWe == NULL)
      {
         printf("Nie udało się otworzyć pliku z tekstem do zaszyfrowania.\n");
         return -1;
      }
      else
      {
         printf("Udało się otworzyć plik z tekstem do zaszyfrowania\n");
      }

      // Obliczenie ilości znaków w pliku z tekstem do zaszyfrowania
      int counter = 0;
      while (!feof(plikWe))
      {
         fgetc(plikWe);
         counter++;
      }

      // Alokacja bufora przechowującego tekst z pliku z tekstem do zaszyfrowania
      int *tekst = (int*)malloc(sizeof(int)*counter);
      if (tekst == NULL)
      {
         printf("Nie udało się zaalokować tablicy 'tekst'");
         return -1;
      }

      rewind(plikWe);
      int i = 0;
      int c = 0;
      // Wszystkie znaki w buforze przechowującym tekst z pliku mają być zapisane DUŻYMI LITERAMI
      while (!feof(plikWe))
      {
         c = toupper(fgetc(plikWe));
         tekst[i]=c;
         i++;
      }
      // Zamknięcie pliku z tekstem, teraz operujemy tylko na buforze przechowującym tekst
      if (plikWe!=NULL)
      {
         fclose(plikWe);
      }

      // Alokacja tablicy pomocniczej przy szyfrowaniu. Znaki z bufora są pokryte kluczem
      int *pokrytyKluczem = (int*)malloc(sizeof(int)*counter);
      if (pokrytyKluczem == NULL)
      {
         printf("Nie udało się zaalokować tablicy 'pokrytyKluczem'");
         free(tekst);
         return -1;
      }
      // Alokacja tablicy docelowej, w której znajdzie się zaszyfrowany tekst
      int *zakodowanyTekst = (int*)malloc(sizeof(int)*counter);
      if (zakodowanyTekst == NULL)
      {
         printf("Nie udało się zaalokować tablicy 'zakodowanyTekst'");
         free(tekst);
         free(pokrytyKluczem);
         return -1;
      }
      int charCounter = 0;

      i=0;

      // Wypisanie niezaszyfrowanego tekstu z bufora
      while (tekst[i]!='\0')
      {
         printf("%c", tekst[i]);
         i++;
```

```
        }
        printf("\n");
        i=0;

        // Pokrycie tekstu kluczem (zapisanie tablicy pokrytyKluczem)
        while (tekst[i]!='\0')
        {
           if (tekst[i] == ' ' || ( tekst[i]>'Z' || tekst[i]<'A' ))
           {
              pokrytyKluczem[i] = ' ';
           }
           else
           {
              pokrytyKluczem[i] = klucz[charCounter%keyLenght];
              charCounter++;
           }
           printf("%c", pokrytyKluczem[i]);
           i++;
        }
        printf("\n");


        i=0;

        // Zakodowanie tekstu. Pracujemy na tabluaRecta. Bierzemy literę tekstu niezakodowanego
        // i przypisujemy ją wierszowi tabulaRecta. Bierzemy literę z tablicy pokrytykluczem
        // i przypisujemy ją kolumnie. Litera na przecięku wiersza i kolumny to zaszyfrowana litera tekstu
        for (int i=0; i<counter; i++)
        {
           if ( tekst[i]>'Z' || tekst[i]<'A' )
           {
              zakodowanyTekst[i] = ' ';
           }
           else
           {
              zakodowanyTekst[i] = tabulaRecta[tekst[i]-'A'][pokrytyKluczem[i]-'A'];
              //charCounter++;
           }
           printf("%c", zakodowanyTekst[i]);
        }

        printf("\n");

        // Otwieramy plik do zapisu zaszyfrowanego tekstu
        FILE *plikWy = fopen(nazwaPlikuWyjsciowego, "w");
        if (plikWy==NULL)
        {
           printf("Nie udało się otworzyć pliku do przechowywania zakodowanego tekstu.\n");
           free(tekst);
           free(pokrytyKluczem);
           free(zakodowanyTekst);
           return -1;
        }
        else
        {
           printf("Udało się otworzyć plik do przechowywania zakodowanego tekstu.\n");
        }

        // Zapisujemy zaszyfrowany tekst
        while (zakodowanyTekst[i]!='\0')
        {
           fputc(zakodowanyTekst[i], plikWy);
           i++;
        }

        // Zwalniamy tablice z tekstem, pokrytą kluczem i z zakodowanym tekstem
        free(tekst);
        free(pokrytyKluczem);
        free(zakodowanyTekst);

        // Zamykamy plik wyjściowy
        fclose(plikWy);
        return 0;
}

// Funkcja deszyfrująca
int deszyfrowanie(char operacja,
                  char nazwaPlikuWejsciowego[],
                  char nazwaPlikuWyjsciowego[],
                  char klucz[])
{
   char tabulaRecta[MAX_ROWS][MAX_COLS];
   generujTabulaRecta(tabulaRecta);
   int keyLenght = strlen(klucz);

   // Otwieramy plik z zaszyfrowanym tekstem
   FILE *plikWe = fopen(nazwaPlikuWejsciowego, "r");
      if (plikWe == NULL)
      {
         printf("Nie udało się otworzyć pliku z zaszyfrowanym tekstem.\n");
```

```
    return -1;
}
else
{
    printf("Udało się otworzyć plik z zaszyfrowanym tekstem.\n");
}

// Obliczamy ilość znaków w pliku z zaszyfrowanym tekstem
int counter = 0;
while (!feof(plikWe))
{
    fgetc(plikWe);
    counter++;
}

// Alokujemy tablicę do przechowywania zaszyfrowanego tekstu
int *tekst = (int*)malloc(sizeof(int)*counter);
if (tekst == NULL)
{
    printf("Nie udało się zaalokować tablicy 'tekst'");
    return -1;
}
rewind(plikWe);
int i = 0;
int c = 0;

// Wszystkie znaki w zaszyfrowanego tekstu muszą być zapisane DUŻYMI LITERAMI
while (!feof(plikWe))
{
    c = toupper(fgetc(plikWe));
    tekst[i]=c;
    i++;
}

// Zamykamy plik wejściowy, teraz pracujemy tylko na tablicy z zaszyfrowanym tekstem
if (plikWe!=NULL)
{
    fclose(plikWe);
}

// Alokujemy tablicę pomocniczą do pokrycia zaszyfrowanego tekstu kluczem
int *pokrytyKluczem = (int*)malloc(sizeof(int)*counter);
if (pokrytyKluczem == NULL)
{
    printf("Nie udało się zaalokować tablicy 'pokrytyKluczem'");
    free(tekst);
    return -1;
}
// Alokujemy tablicę do przechowywania odkodowanego tekstu
int *odkodowanyTekst = (int*)malloc(sizeof(int)*counter);
if (odkodowanyTekst == NULL)
{
    printf("Nie udało się zaalokować tablicy 'odkodowanyTekst'");
    free(tekst);
    free(pokrytyKluczem);
    return -1;
}
int charCounter = 0;

i=0;

// Wypisujemy zaszyfrowany tekst
while (tekst[i]!='\0')
{
    printf("%c", tekst[i]);
    i++;
}
printf("'\n");

i=0;

// Pokrywamy zaszyfrowany tekst kluczem
while (tekst[i]!='\0')
{
    if ( tekst[i]>'Z' || tekst[i]<'A' )
    {
        pokrytyKluczem[i] = ' ';
    }
    else
    {
        pokrytyKluczem[i] = klucz[charCounter%keyLenght];
        charCounter++;
    }
    printf("%c", pokrytyKluczem[i]);
    i++;
}
printf("'\n");

i=0;
```

```
        // Deszyfrujemy zakodowany tekst. Wykorzystujemy tablicę tabulaRecta. Bierzemy literę z tablicy pokrytyKluczem i ustawiamy
        // na kolumnie tabulaRecta. Następnie szukamy w tej kolumnie litery z tablicy z zaszyfrowanym tekstem. Pierwsza litera wiersza
        // przecięciu litery z pokrytyKluczem i litery z tekstu zaszyfrowanego to odkodowana litera.
        while (tekst[i]!='\0')
        {
           if (tekst[i]>'Z' || tekst[i]<'A')
           {
              odkodowanyTekst[i] = ' ';
           }
           else
           {
              for (char A='A'; A<='Z'; A++)
              {
                 if (tabulaRecta[A-'A'][pokrytyKluczem[i]-'A'] == tekst[i])
                 {
                    odkodowanyTekst[i]=A;
                    break;
                 }
              }
           }
           printf("%c", odkodowanyTekst[i]);
           i++;
        }
        printf("\n");

        // Otwieramy plik do zapisu odkodowanego tekstu
        FILE *plikWy = fopen(nazwaPlikuWyjsciowego, "w");
        if (plikWy==NULL)
        {
           printf("Nie udało się otworzyć pliku do przechowywania odkodowanego tekstu.\n");
           free(tekst);
           free(pokrytyKluczem);
           free(odkodowanyTekst);
           return -1;
        }
        else
        {
           printf("Udało się otworzyć plik do przechowywania odkodowanego tekstu.\n");
        }

        i=0;

        // Zapisujemy odkodowany tekst do pliku
        while(odkodowanyTekst[i]!='\0')
        {
           fputc(odkodowanyTekst[i], plikWy);
           i++;
        }

        // Zwalniamy tablicę z tekstem zakodowanym, pokrytą kluczem i z odkodowanym tekstem
        free(tekst);
        free(pokrytyKluczem);
        free(odkodowanyTekst);

        // Zamykamy plik wyjściowy
        fclose(plikWy);
        return 0;
}

// Funkcja silnikSzyfrujący agregująca funkcję szyfrującą i deszyfrującą.
// Dodatkowo umożliwia kryptoanalizę z wykorzystaniem tabulaRecta
void silnikSzyfrujacy(char operacja,
                      char nazwaPlikuWejsciowego[],
                      char nazwaPlikuWyjsciowego[],
                      char klucz[],
                      char kryptoanaliza)
{
   printf("Uruchomiono silnik szyfrujący.\n");
   char tabulaRecta[MAX_ROWS][MAX_COLS];
   generujTabulaRecta(tabulaRecta);
   int keyLenght = strlen(klucz);

   // Litery klucza muszą być zapisane DUŻYMI LITERAMI
   for (int i=0; i<keyLenght; i++)
   {
      klucz[i]=toupper(klucz[i]);
   }

   // Włączenie kryptoanalizy. Użytkownik może samodzielnie szyfrować i deszyfrować
   // tekst na wyświetlonej tabulaRecta
   if (kryptoanaliza=='1')
   {
      wyswietlTabulaRecta(tabulaRecta);
   }

   // Włączenie operacji szyfrowania
   if (operacja=='e')
   {
      szyfrowanie(operacja, nazwaPlikuWejsciowego, nazwaPlikuWyjsciowego, klucz);
   }
```

```
    // Włączenie operacji deszyfrowania
    if (operacja == 'd')
    {
        deszyfrowanie(operacja, nazwaPlikuWejsciowego, nazwaPlikuWyjsciowego, klucz);
    }
}
```