# UNIVERZITET U BEOGRADU
# ELEKTROTEHNIČKI FAKULTET

***Katedra za elektroniku***

***Predmet: Računarska elektronika***


# Projekat br. 7

## (De)šifrovanje primenom Rail Fence algoritma

Rok za predaju: 04.06.2017.

Projekat radili:


| Ime | Prezime | broj indeksa |
|---|---|---|
| Nataša | Jakišić | 293/2014 |
| Tatjana | Stefanović | 273/2013 |

# Sadržaj

# Zadatak

Napisati program koji učitava ulazni fajl koji u prvom redu sadrži :

mode ∈ {e, d} - karakter koji definiše da li tekst koji se nalazi u fajlu treba šifrovati (e) ili dešifrovati (d) primenom Rail Fence algoritma.

n - parametar koji definiše broj redova koji se koristi ili je korišćen prilikom enkripcije fajla koji su razdvojeni jednim znakom razmaka, a od drugog reda na dalje nalazi se tekst koji je potrebno obraditi. Kao rezultat izvršavanja programa, potrebno je u izlazni fajl, prema formatu specificiranom za ulazni fajl upisati mode i broj redova, kao i rezultat obrade ulaznog teksta (parametar n je isti kao i u ulaznom fajlu, parametar mode je suprotan od onog koji je definisan u ulaznom fajlu).

**Primer ulaznog fajla:**

e 4
ćufta je kul

**Primer izlaznog fajla:**

d 4
ćeujkfautl

# Opis korišćenih funkcija i procedura

U ovom delu su navedene i opisane sve funkcije i procedure redosledom kojim su korišćene u programu.

## ReadString

ReadString - Čita string koji se unosi sa tastature, prekida se pritiskom Enter-a. Potrebno je uneti offset u registar EDX i maksimalni broj karaktera za unos (plus jedan) u registar ECX.

U programu se koristi za unošenje imena tekstualnog fajla koji se obrađuje.

## OpenInputFile

OpenInputFile – Otvara postojeći fajl. Potrebno je proslediti offset imena fajla u registar EDX. Ako je uspešno otvoren fajl, EAX će sadržati filehandle.

U programu se koristi za otvaranje fajla za obradu i za proveru uspešnosti otvaranja fajla.

## ReadFromFile

ReadFromFile – Ubacuje sadržaj fajla u bafer. Potrebno je proslediti ofset bafera u registar EDX i maksimalnu veličinu bafera u registar ECX. Po završetku se proverava Carry flag, ako je nula, EAX sadrži broj procitanih karaktera, a ako je jedan, EAX sadrži gresku koja se ispisuje upotrebom funkcije WriteWindowsMsg

U programu se koristi za čitanje fajla za obradu i za proveru broja karaktera u fajlu (da li je veći od bafera tj. od 501).

## WriteToFile

Funkcija koja ispisuje u novom izlaznom fajlu string koji se nalazi u baferu.

## ExitProcess

ExitProcess – Funkcija za izlazak iz programa.

# Opis rada programa

Prvo je potrebno učitati ime fajla, otvoriti fajl i ubaciti sadržaj fajla u bafer koji služi za obradu. Ovo se izvršava upotrebom procedura koje su gore opisane.

## Čitanje fajla za obradu

Kroz bafer se prolazi upotrebom brojača koji pokazuje na karakter koji se obrađuje. U registar EBX se upisuje 0 tako da pokazuje na početak bafera tj na prvi karater za obradu i u nastavku programa se koristi kao brojač.

Na početku svakog reda učita se karakter u registar AL i poveća se EBX za 2 jer je na sledećem mestu razmak koji ne treba da se obrađuje. Iz registra AL karakter prebacimo u enc_or_dec. Taj karaktr može biti "e" ili "d" što nam pokazuje da li zadati tekst treba da šifrujemo ili dešifrujemo. Na osnovu toga skače se kasnije na labelu šifrovanje ili dešifrovanje. Sedeći deo ulaznog fajla predstavlja n. To je broj koji nam označava koliko karaktera ima u jednoj dijagonali grafičkog prikaza Rail Fence algoritma. Zatim se proverava da li je kraj reda i ako jeste da li postoji tekst u sledećem redu. Ako ne postoji ispisuje se poruka za grešku, a ako postoji pročita se tekst koji treba obraditi. Kada naiđe na razmak program ga izbaci iz teksta.

## Broj slova u jednom redu

Nakon toga kreće obrada. Prvo računamo koliko koji red ima slova. Na osnovu broja n računamo koliko ima slova u jednom ciklusu. Broj celih ciklusa dobijamo kada podelimo ukupan broj slova u tekstu sa brojem slova jednog ciklusa k.
Prvi red ima slova koliko ima celih ciklusa, poslednji može da ima 1 slovo više u zavisnosti od ukupnog broja slova. Redovi u sredini imaju duplo više slova od broja punih ciklusa, jer u svakom ciklusu po 2 slova su u istom redu, plus 0, 1 ili 2 u zavisnosti od ukupnog broja slova.

## Šifrovanje i dešifrovanje

U zavisnosti od toga šta se u fajlu traži radimo šifrovanje ili dešifrovanje ulaznog teksta.
Šifrujemo i dešifrujemo red po red, prvi i poslednji red su specifični i njih šifrujemo posebno a redove između šifrujemo i dešifrujemo sa pomeranjem reda u kom se nalazimo u odnosu na prvi red. Imamo pomoćni ulazni niz koji sadrži tekst koji treba šifrovati i iz njega prebacujemo u bafer slova algoritmom za šifrovanje ili dešifrovanje. Bafer uvećavamo za 1 i ubacujemo u njega odgovarajuće karaktere.

## Šifrovanje

Iz ulaznog niza čitamo:

| | |
|---|---|
| Za prvi red | u[0], u[k], u[2k],… |
| Za drugi red | u[0+1], u[k-1], u[k+1], u[2k-1], u[2k+1],… |
| Za treći red | u[0+2], u[k-2], u[k+2], u[2k-2], u[2k+2],… |
| Za m-ti red(koji nije poslednji) | u[0+m-1], u[k-m+1], u[k+m-1], u[2k-m+1], u[2k+m-1],… |
| Za poslednji red(n-ti) | u[0+n-1], u[k+n-1], u[2k+n-1], u[3k+n-1],… |

## Dešifrovanje

Koristimo promenljivu r koja nam označava početak reda u koji treba da ispišemo šifrovani tekst. Onda ulazni niz inkrementiramo za 1 a red po red menjamo raspored karaktera i stavljamo na svoje mesto u bafer.

| | | |
|---|---|---|
| Za prvi red | | buffer[r], buffer[r+k], buffer[r+2k],… |
| Za drugi red | s=1 | buffer[r+s], buffer[r+k-s], buffer[r+k+s],… |
| Za treći red | s=2 | buffer[r+s], buffer[r+k-s], buffer[r+k+s],… |
| Za m-ti red(koji nije poslednji) | s=m-1 | buffer[r+s], buffer[r+k-s], buffer[r+k+s],… |
| Za poslednji red(n-ti) | s=n-1 | buffer[r+s], buffer[r+s+k], buffer[r+s+2k] |

## Ispis

Prvi celokupni red ostavljamo sa izmenjenim prvim slovom “d” -> “e” ili “e” -> ”d“ kao i znak za početak novog reda. Imamo pomoćni ulazni niz koji sadrži tekst koji smo (de)šifrovali tako što smo menjali sadržaj bafera i sada ispisujemo bafer u izlazni fajl.
Takođe smo prebrojali koliko slova i znakova treba da sadrži izlazni fajl i to smo upisali u stringLength.

# Kod

Ovde su komentari pisani kao ;// da bi ih program prepoznao kao komentare i da bi se lepše videlo u izveštaju. U programu su komentari napisani kao ; .

```
INCLUDE Irvine32.inc
INCLUDE macros.inc


BUFFER_SIZE = 501;// maks velicina

.data

buffer BYTE BUFFER_SIZE DUP(? )
filenameout BYTE "output.txt", 0
filename BYTE 80 DUP(0)
fileHandle HANDLE ?
u BYTE 501 DUP(? )
enc_or_dec BYTE ?
k DWORD ?
p DWORD ?
stringLength DWORD ?
s DWORD ?
v DWORD ?
i DWORD ?
n DWORD ?
uk DWORD ?
r DWORD ?
l DWORD ?
m BYTE ?
z DWORD 501 DUP(0)

str1 BYTE "Cannot create file", 0dh, 0ah, 0
outHandle HANDLE 0
bytesWritten DWORD ?
endl EQU <0dh, 0ah>;// end of line
novired LABEL BYTE
BYTE endl


.code
main PROC
; UCITAVANJE FILE - a

mWrite "Unesite ime file-a: "
mov edx, OFFSET filename
mov ecx, SIZEOF filename
call ReadString
; Open the file for input.
mov edx, OFFSET filename
call OpenInputFile
```

```
mov fileHandle, eax
; Check for errors.
cmp eax, INVALID_HANDLE_VALUE; error opening file ?
jne file_ok; no: skip
mWrite <"Cannot open file", 0dh, 0ah>
jmp quit; and quit

file_ok :
; Read the file into a buffer.
mov edx, OFFSET buffer
mov ecx, BUFFER_SIZE
call ReadFromFile
jnc check_buffer_size; error reading ?

poruka :
	mWrite "Error reading file. "; yes: show error message
	call WriteWindowsMsg
	jmp close_file

	check_buffer_size :
cmp eax, BUFFER_SIZE; buffer large enough ?
jb buf_size_ok; yes
mWrite <"Error: Buffer too small for the file", 0dh, 0ah>
jmp quit; and quit


buf_size_ok :
mov ecx, eax
mov stringLength, eax
mov uk, ecx
mov ebx, 0
mov al, [buffer + ebx];// ucitava prvo slovo
mov enc_or_dec, al
add ebx, 1;// preskace razmak
mov p, eax
mov eax, uk
sub eax, 2
dec eax
mov uk, eax
mov eax, 0;

; citanje broja

mov edi, 0

broj :
add ebx, 1;// pokazuje na trenutni
mov eax, 0
mov al, [buffer + ebx];// cita ga

mov p, eax
mov eax, uk
sub eax, 1
mov uk, eax
mov eax, p
cmp al, 0ah; //EOF
je poruka; //ako je nema teksta
```

```
cmp al, 0dh;// EOL
je kraj_linije
imul edi, 10
sub al, 48
add edi, eax
mov n, edi
jmp broj

; pocetak ucitavanja teksta za obradu
kraj_linije :
add ebx, 1
mov r, ebx;
mov edx, 0
sub ecx, 1
mov l, ecx
mov i, 0
loop1 :
	add ebx, 1
	mov al, [buffer + ebx]
	mov m, ' '
	cmp m, al;// izbacuje razmak ako dobijemo tekst za sifrovanje
	je loop2
	mov edx, i
	mov[u + edx], al
	mov cl, [u + edx]
	mov eax, i
	add eax, 1
	mov i, eax

	jmp loop3
	loop2 :
mov eax, stringLength
sub eax, 1
mov stringLength, eax
mov eax, uk
sub eax, 1
mov uk, eax
loop3 :

cmp l, ebx
jne loop1
mov ecx, l
add ecx, 1
mov edx, n

; pocetak izracunavanja koliko koji red ima slova
mov eax, uk
mov eax, 2
mul edi
sub eax, 2; //k je koliko ima slova u jednom ciklusu
mov k, eax
mov ecx, k
mov eax, uk
mov s, 0
div ecx; //broj celih ciklusa
mov p, eax; //broj celih ciklusa
mul ecx
```

```
mov l, eax;// ukupan broj slova u punim cilkusima
mov i, 0
mov edi, 0
mov eax, n
sub eax, 1
mov v, eax
mov ecx, 0

mov eax, p
mov[z + ecx], eax

mov eax, l
inc eax
cmp uk, eax
jb  pon1; //ako je carry 1 uk je vece od l
mov eax, [z + ecx]
add eax, 1

mov[z + ecx], eax
mov eax, s
inc eax
cmp eax, n
je sif_ili_desif
dec eax
mov s, eax

pon1 : ;// posle prvog reda svaki se tu vraca
       add ecx, 4
       mov eax, s
       add eax, 1
       mov s, eax
       cmp v, eax
       je loo

       mov eax, 2
       mov edx, p

       mul edx
       mov[z + ecx], eax
       mov eax, l
       inc eax
       add eax, s
       cmp uk, eax
       jb pon1
       mov eax, [z + ecx]
       add eax, 1
       mov[z + ecx], eax
       mov eax, l
       inc eax
       add eax, k
       sub eax, s
       cmp uk, eax
       jb pon1
       mov eax, [z + ecx]
       add eax, 1
       mov[z + ecx], eax
```

```
        jmp pon1

        loo :
mov eax, p

mov[z + ecx], eax
mov eax, l
add eax, n
cmp uk, eax
jb sif_ili_desif
mov eax, [z + ecx]
add eax, 1
mov[z + ecx], eax



sif_ili_desif :
cmp enc_or_dec, 'e'
je sifrovanje
cmp enc_or_dec, 'd'
je desifrovanje

greska :
mWrite <"Pogresni zahtevi", 0dh, 0ah>
jmp quit

sifrovanje :
mov[buffer + 0], 'd'
mov s, 0
mov i, 0

mov eax, 0
mov edi, 0
mov ecx, 0
mov ebx, r
add ebx, 1
pon2 :
        mov al, [u + ecx]
        mov[buffer + ebx], al
        add ebx, 1
        add ecx, k
        mov eax, i
        add eax, 1
        mov i, eax
        cmp[z + edi], eax
        jne pon2; kraj n = 1

        pon3 :;// sledeci red se sifruje
        add edi, 4
        mov eax, s
        mov ecx, 0
        mov i, 0
        inc eax
        mov v, 0
        mov s, eax
        inc eax
```

```
       cmp eax, n;
je loop5

pon4 :
mov ecx, v    ;// zadrzavamo vrednost jer nam je potrbno, ostalo sve radimo preko v
add ecx, s
mov al, [u + ecx]
mov[buffer + ebx], al
add ebx, 1
mov eax, i
add eax, 1
mov i, eax
cmp eax, [z + edi]
je pon3
mov ecx, v
add ecx, k
mov v, ecx
sub ecx, s
mov al, [u + ecx]
mov[buffer + ebx], al
mov eax, i
add eax, 1
mov i, eax
add ebx, 1
cmp eax, [z + edi]
je pon3
jmp pon4
loop5 :
mov ecx, n
sub ecx, 1
mov eax, 0
pon5 : ;//  poslednji red se sifruje

mov al, [u + ecx]
mov[buffer + ebx], al
mov eax, i
add eax, 1
mov i, eax
add ebx, 1
add ecx, k
cmp eax, [z + edi]
jne pon5
mov[buffer + ebx], 0ah
jmp ispis


desifrovanje :
mov[buffer + 0], 'e'
mov i, 0
mov s, 0
mov p, 0
mov s, 0
mov ecx, 0
mov eax, 0
mov edi, 0
mov ebx, r
add ebx, 1
```

```
mov r, ebx

;desifrovanje prvog reda
lp:
mov al, [u + ecx]
mov[buffer + ebx], al
add ecx, 1
mov eax, i
add eax, 1
mov i, eax
add ebx, k

cmp eax, [z + edi]
jne lp
mov eax, s
inc eax
cmp n, eax
je ispis      ;//kraj_desifrovanja

lp1:
add edi, 4
mov eax, s
inc eax
mov s, eax
inc eax
mov ebx, r
mov i, 0
cmp eax, n;// edx treba da bude n - 1, ovde se skace da treba poslednji red da se sifruje
je lp3
mov eax, 0
lp2:
mov v, ebx
add ebx, s
mov al, [u + ecx]
mov[buffer + ebx], al
mov eax, i
add eax, 1
mov i, eax
add ecx, 1
cmp eax, [z + edi]
je lp1
mov ebx, v
add ebx, k
mov v, ebx
sub ebx, s
add eax, 0
mov al, [u + ecx]
mov[buffer + ebx], al
add ecx, 1
mov eax, i
inc eax
mov i, eax
cmp eax, [z + edi]
je lp1
mov ebx, v
jmp lp2
```

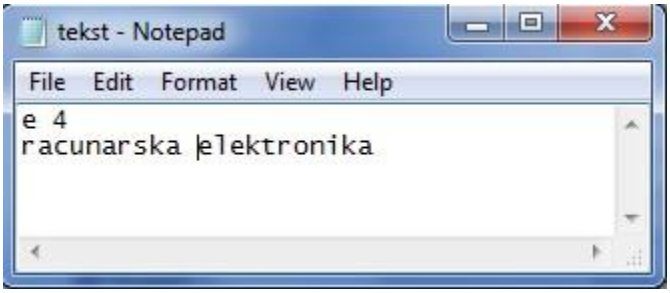
```
lp3 :
mov eax, 0
mov ebx, r
add ebx, s
lp4 : mov al, [u + ecx]
       mov[buffer + ebx], al
       add ecx, 1
       mov eax, i
       inc eax
       mov i, eax
       add ebx, k
       cmp eax, [z + edi]
       jne lp4
       ;//kraj_desifrovanja

ispis :

; Create a new text file
mov edx, OFFSET filenameout
call CreateOutputFile
mov fileHandle, eax
; Check errors.
cmp eax, INVALID_HANDLE_VALUE;  error found ?
jne file_ok1; no: skip
mWrite <"Ne moze se ispisati izlazni fajl", 0dh, 0ah>
jmp quit
file_ok1 :
mov eax, fileHandle
mov edx, OFFSET buffer
mov ecx, stringLength
call WriteToFile
mov bytesWritten, eax; save return value
call CloseFile

jmp quit
close_file :
mov eax, fileHandle
call CloseFile

quit :
invoke ExitProcess, 0
main ENDP
END main
```

# Pokretanje programa i test

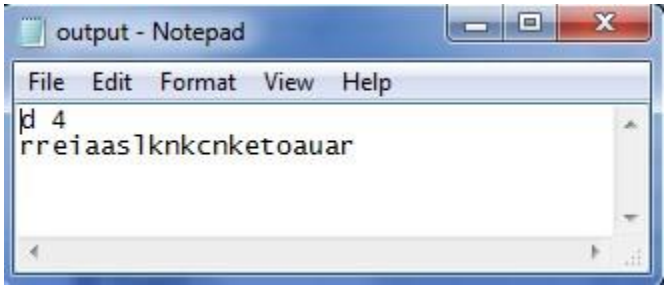
Test fajl se zove tekst.txt a izlazni output.txt i nalaze se u folderu gde i sam program. Program će tražiti unošenje imena fajla gde se treba uneti tekst.txt i pritisnuti enter. Test fajl treba da bude napisan u odgovarajućem formatu i da sadrži manje od 501 karaktera kako program ne bi izbacio grešku.

## Primer 1

Sadržaj fajla tekst.txt je:

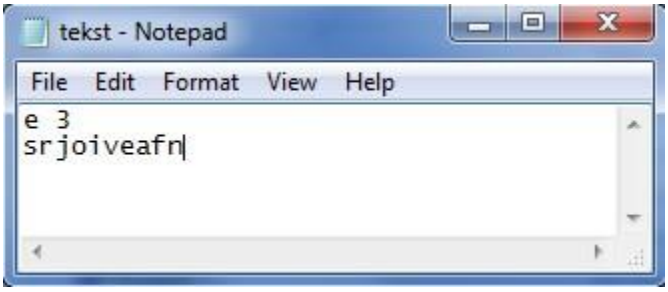



Rezultat je:

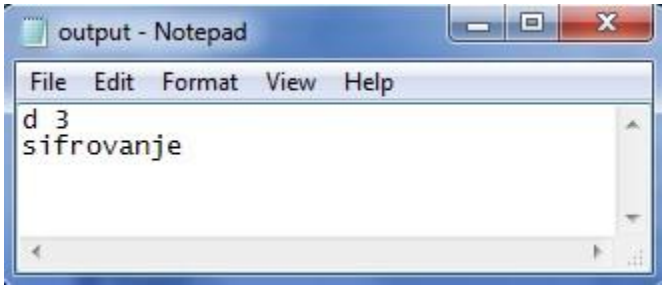


## Primer 2

Sadržaj fajla tekst.txt je:




Rezultat je:



# Literatura


1. Vežbe iz predmeta Računarska elektronika
2. Kip R. Irvine Assembly Language for x86 Processors