

**UNIVERZITET U BEOGRADU  
ELEKTROTEHNIČKI FAKULTET**

*Katedra za elektroniku*

*Predmet: Računarska elektronika*



**Projekat: Otkrivanje/skrivanje tajne poruke**

***Projekat radili:***

Ivana Vasiljević 2014/0022

Marko Marković 2014/0597

***Predmetni profesor:***

Milan Prokin

***Predmetni asistent:***

Aleksandra Lekić

Jun 2017, Beograd

### ***Tekst zadatka:***

Napisati program kojim se vrši otkrivanje tajne poruke koja se nalazi u slici i zapisuje je u novi PGMA file, a zatim u ulaznu sliku sakriva proizvoljnu novu poruku.

### ***Struktura projekta:***

1. Učitavanje ulaznih datoteka
2. Obrada ulaznih datoteka
3. Ispis rezultata obrade u izlaznu datoteku

### ***1. Učitavanje ulaznih datoteka***

U okviru glavnog programa učitavaju se datoteke potrebne za rad. Prvo je u bafer buffer učitana slika iz koje treba da se izvuče tajna poruka, i u istu tu sliku biće sakrivena slika koja se naredna učitava u bafer buffer2. Mi smo u buffer učitali pepper\_msg.pgm, a u buffer2 smo učitali cassablanca\_msg.pgm.

Postupak učitavanja ulazne datoteke sastoji se iz sledećih koraka:

- Korisnik upisuje ime ulazne datoteke
- Otvara se ulazna datoteka
- Proverava se da li je doslo do greške pri otvaranju
- Sadržaj datoteke se učitava u bafer
- Proverava se da li ima greske i da li je bafer dovoljno veliki
- Ispisuje se veličina datoteke
- Zatvara se ulazna datoteka

Iz Irvine.inc biblioteke smo koristili sledeće rutine:

- ReadString
- OpenInputFile
- ReadFromFile
- WriteWindowsMsg
- WriteDec
- Crlf
- CloseFile

## 2. Obrada ulaznih datoteka

Obrada ulaznih datoteka se sastoji iz tri procedure

- process
- process2
- process3

Iz Irvine.inc biblioteke smo koristili sledeće rutine:

- IsDigit
- ParseDecimal32

Procedure pozivane iz ovih procedura:

- ***row\_copy\_paste***

Ovo je procedura za prepisivanje reda u izlazni bafer.

Brojači counter1 i counter22 su postavljeni na 0. Brojač counter 22 se inkrementira za svaki učitani znak, a brojač counter1 se uvećava za svaki učitani red koji nije komentar.

Učitani znak se poredi sa # (koja je oznaka da je taj red pocetak komentara), pa se na osnovu toga zna da li brojač counter1 treba uvećavati ili ne. Nakon toga se učitavaju ostali karakteri, pritom se proverava da li je učitani karakter 0ah, koji označava kraj reda. Kada bude učitani 0ah završava se procedura.

- ***isp1***

Ovo je procedura koja ispisuje jednu cifru i razmak.

Po dolasku u ovu proceduru razmak se čuva na steku jer je on poslednji učitani u eax. Pre učitavanja razmaka broj koji je prethodno bio u eax je sačuvan u pom4. On se vraća u eax, a zatim se tom broju briše poslednji bit, tj. postavljamo najniži bit na 0. Postavljanje najnižeg bita na 0 smo uradili sledećom operacijom: `and eax, 254`. Zatim, na mesto najnižeg bita potrebno je upisati 1 ili ostaviti 0 (trenutni broj sabrati sa 0 ili 1), u zavisnosti koja je šifra ili slika u pitanju. Vrednosti najvišeg bita slike koju želimo da sakrijemo smesteni su u bafer outBuffer7. Odredjenom pikselu iz bafera outBuffer7 pristupamo na osnovu vrednosti brojača counter2. Ta vrednost postaje vrednost prethodno izbrisanog najnižeg bita. Posle ispisivanja ovog broja, razmak se vraća sa steka pa se ispisuje i on.

- ***isp2***

Ovo je procedura koja ispisuje 2 cifre i razmak.

Po dolasku u ovu proceduru razmak se čuva na steku jer je on poslednji učitani u eax. Pre učitavanja razmaka brojevi koji je prethodno bili u eax su sačuvani u pom4 i pom5. (Redosled učitavanja je bio pom4, pom5, razmak). Odmah se ispisuje vrednost koja je sačuvana u pom4, a u najniži bit pom5 smešta se cifra određenog piksela iz bafera outBuffer7, na isti način kao i u proceduri isp1 što se radilo sa pom4. Nakon te obrade najnižeg bita, ispisuje se cifra, a zatim i razmak vraćen sa steka.

- **isp3**

Ovo je procedura koja ispisuje 3 cifre i razmak.

Po dolasku u ovu proceduru razmak se čuva na steku jer je on poslednji učitani u eax. Pre učitavanja razmaka brojevi koji je prethodno bili u eax su sačuvani u pom4, pom5 i pom6. (Redosled učitavanja je bio pom4, pom5, pom6, razmak). Odmah se ispisuju vrednosti koje su sačuvane u pom4 i pom5, a u najnižem bitu pom6 smešta se cifra određenog piksela iz bafera outBuffer7, na isti način kao i u proceduri isp1 što se radilo sa pom4. Nakon te obrade najnižeg bita, ispisuje se cifra, a zatim i razmak vraćen sa steka.

### ***process***

Ovo je procedura u kojoj se izvlači tajna poruka (najniža bitska ravan) iz slike koja je smeštena u baferu buffer. Tajna poruka se smešta u bafer outBuffer.

Prvo se prepisuje zaglavlje iz buffer u outBuffer pozivanjem procedure row\_copy\_paste sve dok counter1 ne postane 3 što znači da je zaglavlje prepisano. Zatim se učitava karakter i proverava, pomoću IsDigit, da li je cifra. Učitava se sve dok se ne učitava nešto što nije cifra (razmak, novi red), a zatim se skače na labelu notDigit. Na samom početku je u edx bio zapamćen početni položaj broja unutar stringa. Sada se na osnovu razlike esi i edx može zaključiti da li je učitani neki broj pa razmak ili npr. razmak pa novi red, i u tom slučaju se samo ispisuje učitani karakter. U slučaju kad je učitani neki broj, tačnije string, prvo se čuvaju vrednosti potrebnih registara na steku, a zatim pomoću ParseDecimal32 učitani sting prebacujemo u decimalni broj. Pomoću `and eax, 01h` otkrivamo da li je vrednost najnižeg bita 0 ili 1. U slučaju da je 0 u izlazni bafer outBuffer ispisujemo '0', a u slučaju da je 1 ispisujemo '2' '5' '5' da bismo imali bolji kontrast. Posle ovog ispisa vraćaju se u registre vrednosti sa steka, i ispisuje se vrednost (koja nije bila cifra) iz eax. Kada se završi sa ispisom svih piksela u outBuffer, u brojače counter1 i counter2 se upisuju nule, da bi ponovo mogli da se koriste pri pozivanju procedure row\_copy\_paste.

### ***process2***

Ovo je procedura u kojoj se izvlači najviša bitska ravan slike iz bafera buffer2. Ova bitska ravan predstavlja novu tajnu poruku koja će se u process3 ubaciti na mesto najniže bitske ravni slike u baferu buffer.

Prvo se prepisuje zaglavlje iz buffer2 u outBuffer7 pozivanjem procedure row\_copy\_paste sve dok counter1 ne postane 3 što znači da je zaglavlje prepisano. Zatim se učitava karakter i proverava, pomoću IsDigit, da li je cifra. Učitava se sve dok se ne učitava nešto što nije cifra (razmak, novi red), a zatim se skače na labelu notDigit. Na samom početku je u edx bio zapamćen početni položaj broja unutar stringa. Sada se na osnovu razlike esi i edx može zaključiti da li je učitani neki broj pa razmak ili npr. razmak pa novi red, i u tom slučaju se samo ispisuje učitani karakter. U slučaju kad je učitani neki broj, tačnije string, prvo se čuvaju vrednosti potrebnih registara na steku, a zatim pomoću ParseDecimal32 učitani sting prebacujemo u decimalni broj. Pomoću `and eax, 80h` otkrivamo da li je vrednost sedmog (najvišeg) bita 0 ili 1. U slučaju da je 0 u izlazni

bafer outBuffer ispisujemo '0', a u slučaju da je 1 ispisujemo '1'. Posle ovog ispisa vraćaju se u registre vrednosti sa steka, i ispisuje se vrednost (koja nije bila cifra) iz eax. Kada se završi sa ispisom svih piksela u outBuffer, u brojače counter1 i counter22 se upisuju nule, da bi ponovo mogli da se koriste pri pozivanju procedure row\_copy\_paste. Pre nego što se u counter22 upiše 0, vrednost koju je imao se prebacuje u novi brojač counter2. Ova vrednost pokazuje koliko karaktera ima u zaglavlju bafera outBuffer7 i ta vrednost će nam biti potrebna u process3 kada budemo hteli da pristupamo brojevima koje smo upisivali posle zaglavlja u outBuffer7 (ti brojevi su ili '0' ili '1').

### ***process3***

Ovo je procedura u kojoj se šifra izvučena u proceduri process2 (i smeštena u outBuffer7) ubacuje kao najniža bitska ravan slike koja se nalazi u baferu buffer. Rezultat ovog smeštanja se nalazi u baferu outBuffersifra.

Prvo se prepisuje zaglavlje iz buffer u outBuffersifra pozivanjem procedure row\_copy\_paste sve dok counter1 ne postane 3 što znači da je zaglavlje prepisano. Zatim se učitava karakter po karakter sve dok se učitava karakter koji nije cifra. Svaka cifra koja je učitana se smešta u pomoću promenljivu redom pom4, pom5, pom6, u zavisnosti koliko cifara ima u datom broju. Ako smo učitali jednu cifru, a već sledeći karakter nije cifra (razmak je) skaćemo na labelu notDigit31 odakle pozivamo proceduru isp1. Ako smo učitali dve cifre, a već sledeći karakter nije cifra (razmak je) skaćemo na labelu notDigit32 odakle pozivamo proceduru isp2. Na labelu notDigit33 možemo doći ako smo učitali tri cifre pa karakter koji nije cifra (razmak je) i u tom slučaju pozivamo isp3 ili ako prvi učitani karakter nije cifra (novi red) i u tom slučaju samo ispisujemo karakter.

### ***3. Ispis rezultata obrade u izlaznu datoteku***

U okviru glavnog programa sadržaj bafera outBuffer, u koji je smeštena otkrivena poruka, smeštamo u izlaznu datoteku. Mi smo tu datoteku nazvali Otkrivena\_sifra.pgm. U izlaznu datoteku smešta se i sadržaj bafera outBuffersifra, u koji je smestena slika sa novom tajnom porukom. Tu datoteku smo nazvali Ubacena\_nova\_sifra.pgm.

Postupak ispisa rezultata se sastoji iz sledećih koraka:

- Korisnik unosi ime izlazne datoteke
- Pravi se izlazna datoteka
- Proverava se da li ima grešaka
- Ispisuje se bafer u izlaznu datoteku
- Zatvara se izlazna datoteka

Iz Irvine.inc biblioteke smo koristili sledeće rutine:

- ReadString
- CreateOutputFile
- WriteToFile
- CloseFile

## ***Kod programa:***

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

BUFFER_SIZE = 256 * 256 * 20

.data
buffer BYTE BUFFER_SIZE DUP(? );Bafer u koji se ubacuje ulazna datoteka - slika iz koje
se treba izvuci tajna poruka.
buffer2 BYTE BUFFER_SIZE DUP(? );Bafer u koji se ubacuje ulazna datoteka - slika koja se
ubacuje kao tajna poruka u prethodnu sliku.

infilename    BYTE 80 DUP(0)
outfilename    BYTE 80 DUP(0)

fileHandle    HANDLE ?

stringLength   DWORD ?;Duzina slike sa tajnom porukom.
stringLength2  DWORD ?;Duzina slike koja je tajna poruka.

outBuffer      BYTE BUFFER_SIZE DUP(? );Bafer u koji se smesta tajna poruka izvucena iz bafera
buffer.
outBuffer7     BYTE BUFFER_SIZE DUP(? );Bafer u koji se smesta sedma bitska ravan iz bafera
buffer2.
outBuffersifra  BYTE BUFFER_SIZE DUP(? );Bafer u koji se smesta slika iz bafera buffer,
;izmenjenog tako da
mu je najniza bitska ravan sedma bitska ravan iz bafera outBaffer7.

counter1       DWORD 0;Brojac koji broji do 3 u proceduri pri prepisivanju zaglavlja datoteke.
counter22      DWORD 0;Brojac koji broji duzinu zaglavlja datoteke.
counter2       DWORD 0;Vrednost brojaca counter22 se prepisuje u counter2 u proceduri
proccess2,
;da bi se sacuvala duzina zaglavlja buffer2 (ista kao i kod
outBuffer7),
;jer na kraju procedure u counter22 upisuje se 0.
;Inkrementiranjem njega u sledecoj proceduri se pristupa
sadrzaju datoteke posle zaglavlja.

pom4          DWORD ?;Pomocna promenljiva u koju se cuva vrednost eax registra.
pom5          DWORD ?;Pomocna promenljiva u koju se cuva vrednost eax registra.
pom6          DWORD ?;Pomocna promenljiva u koju se cuva vrednost eax registra.

.code

row_copy_paste PROC
;Procedura za prepisivanje reda u izlazni bafer:

    lodsb                ;Ako je prvi karakter u redu # to znaci da je taj red komentar,
    stosb                ;sto znaci da se taj red samo prepisuje karakter po karakter u
    dec ecx              ;izlazni bafer, a counter1 se ne uvecava. U slucaju kad prvi
karakter nije #
    inc counter22        ;counter1 se povecava. U ovu proceduru ce se dolaziti dok se
ovde, u proceduri, counter1 ne uveca na 3
```

```

        cmp eax, '#'           ;jer postoje 3 reda zaglavlja ne racunajuci komentare (zbog
toga se i ne uvecava brojac kad prepisujemo komentar).
        je paste              ;Prepisuju se tri reda zaglavlja jer sadrzaj slike koja je u
pgm formatu izgleda:
        inc counter1          ;P2 ;prvi red
paste:                                     ;broj redova broj kolona ;drugi red
        inc counter22         ;maksimalna vrednost pixela ;treci red ;pri cemu se
komentar moze naci izmedju svakog od ova tri reda
        lodsb                 ;counter22 se inkrementira pri ucitavanju svakog znaka.

        stosb                 ;counter22 ce po zavrsetku ove procedure prebrojati koliko je
znakova bilo u zaglavlju.
        cmp eax, 0ah          ;0ah je oznaka za kraj reda, kad se u eax ucita 0ah skace se
na endProc - kraj procedure.
        je endProc
        loop paste            ;Pri svakom povratku na paste, ecx se dekrementira, naredba
dec ecx je potrebna jer je jedno ucitavanje van petlje.
endProc:
        ret                   ;Povratak na mesto sa kog je pozvana ova
procedura.
row_copy_paste ENDP

```

```

process PROC
        ;Procedura za obradu ulazne datoteke:
        ;Izdvaaja se najniza bitska ravan iz datoteke jer je u njoj sakrivena poruka.
        cld
        mov esi, OFFSET buffer ;U buffer je učitana slika iz koje treba izvuci sakrivenu
poruku (najnizu bitsku ravan).
        mov edi, OFFSET outBuffer ;U outBuffer se upisuje izvucena poruka iz buffer-a.
        mov ecx, LENGTHOF buffer ;brojac za petlju

```

```

copy:
        ;Petlja kojom se prepisuje zaglavlje ulazne datoteke u izlaznu:
        cmp counter1, 3
        je move_on ; U proceduru row_copy_paste ide se dok counter1 ne postane 3, kad
postane, skace se na move_on.
        call row_copy_paste
        loop copy

```

```

move_on:
        mov edx, esi ;Pamti se pocetni polozaj broja unutar stringa.

```

```

loop1:
        ;Petlja se vrti dokle god je procitani karakter cifra, u suprotnom se skace na
notDigit.
        lodsb                 ;Pri svakom ponovnom dolasku na loop1, ecx se dekrementira, i
oznacava preostalu duzinu bafera.
        call IsDigit
        jnz notDigit
        loop loop1
        jmp finish           ;Ako je kraj bafera zavrshi obradu.

```

```

notDigit:
        push esi
        sub esi, edx
        cmp esi, 1           ;U slucaju dva uzastopna karaktera koji nisu cifre (razmak +
novi red)

```



```

jne compare
stosb          ;prepisuje se procitani karakter i
pop esi
loop move_on   ;vraca se na move_on.
jmp finish     ;(ako je kraj bafera zavrshi obradu)

compare:        ;Ukoliko je bilo cifri, umesto ispisivanja poslednjeg
procitanog karaktera dolazi se ovde, očitava se broj.
push ecx       ;Stavljaju se na stek vrednosti ecx i eax jer su ti registri
potrebni za dalji rad.
push eax
mov ecx, esi    ;U ecx se prebacuje broj cifara vrednosti pixela.
call ParseDecimal32 ;Konvertuje se string u decimalni broj.
and eax, 01h    ;and 01h je maska kojom kao rezultat ostaje samo najniza
bitska ravan.
jnz one        ;Ako je rezultat razlicit od nule, skace se na labelu one.
mov eax, '0'    ;Ako se ne skoci, rezultat je 0.
stosb          ;Ispisuje se u izlazni bafer 0.
jmp stek       ;Skace se na labelu stek

one:            ;da se preskoci ispis jedinice.
mov eax, '2'    ;Tacnije, umesto jedinice ispisuje se 255 da bi se dobilo na
kontrastu.
stosb
mov eax, '5'
stosb
stosb

stek:
pop eax        ;Skida se sa steka prethodno stavljena vrednost eax
stosb          ;i upisuje se u izlazni bafer (to je char koji nije bio
cifra).
pop ecx        ;Skidaju se vrednosti registara ecx i esi sa steka, kako bi se
nastavilo normalnim tokom.
pop esi
loop move_on   ;Ceo proces se ponavlja dok se ne dodje do kraja bafera.
Svakim ponovnim ulaskom u petlju, ecx se dekrementira.

finish:
mov counter1, 0 ;Brojac se resetuje da bi ponovo mogla da se koristi ista
procedura.
mov counter2, 0 ;Brojac se resetuje da bi ponovo mogla da se koristi ista
procedura.
ret           ;Povratak na mesto sa kog je pozvana ova procedura.
process ENDP

process2 PROC
;Procedura za obradu ulazne datoteke
;Iz datoteke se izdvaja najvisa (sedma) bitska ravan,
;koja ce u proceduri process3 biti ubacena u datoteku koja je ucitana pre ove
datoteke.
;Struktura procedure process2 je veoma slicna proceduri process,
;tako da ce biti komentarisane samo stvari koje su drugacije nego u proceduri
process.
cld
mov esi, OFFSET buffer2
mov edi, OFFSET outBuffer7

```

```

        mov ecx, LENGTHOF buffer2

copy2:
    cmp counter1, 3
    je move_on2
    call row_copy_paste
    loop copy2

move_on2:
    mov edx, esi

loop2:
    lodsb
    call IsDigit
    jnz notDigit2
    loop loop2
    jmp finish2

notDigit2:
    push esi
    sub esi, edx
    cmp esi, 1
    jne compare2
    stosb
    pop esi
    loop move_on2
    jmp finish2

compare2:
    push ecx
    push eax
    mov ecx, esi
    call ParseDecimal32
    and eax, 80h                ;and 80h je maska takva da u rezultatu su svi biti osim
sedmog nula,                    ;a sedmi bit rezultata je sedmi bit eax. Ako je on nula skace
    jz zero2                    ;se na zero2
    mov eax, '1'                ;U suprotnom, ispisuje se 1.
    stosb
    jmp stek2                    ;Preskace se ispis 0.

zero2:
    mov eax, '0'                ;Ispisuje se 0.
    stosb

stek2:
    pop eax
    stosb
    pop ecx
    pop esi
    loop move_on2

finish2:
    mov counter1, 0;Brojac se resetuje da bi ponovo mogla da se koristi ista
procedura.
    mov eax, counter22;Vrednost iz coutera22 se prepisuje u counter2 da bi bila
sacuvana
    mov counter2, eax;jer counter22 mora da se resetuje.

```

```

        mov counter22, 0;Brojac se resetuje da bi ponovo mogla da se koristi ista
procedura.
        ret
process2 ENDP

isp1 PROC
        push eax                                ;Poslednja ucitana vrednost (razmak) cuva se na steku
dok ne dodje vreme da se ispise.
        mov eax, pom4
        and eax, 254                            ;Brise se najnizi bit - postavlja se na nulu.
        mov pom4, eax                            ;Vrednost eax cuva se u pom4.
        mov eax, counter2                      ;Kad se prvi put udje u ovu proceduru, outBuffer7[counter2] je
prvi broj (0 ili 1) posle zaglavlja u tom baferu.
        xor edx, edx                            ;edx=0
        mov dl, outBuffer7[eax] ;U dl je 30 ili 31 jer je to ASCII kod za 0 i 1.
        and dl, 01h                            ;Sada je u dl 0 ili 1.
        mov eax, edx
        add eax, pom4;Sada je u eax piksel u kome je na najnižem bitu smestena vrednost
najviseg bita odgovarajućeg piksela slike koja se sakriva.
        stosb                                ; I ta vrednost se ispisuje.
        pop eax
        stosb                                ;Ispisuje se razmak.
        inc counter2 ;Vrednost brojaca se uvecava za 2 jer u baferu outBuffer7 izmedju
svaka dva broja ima razmak, a nama trebaju samo brojevi.
        inc counter2
        dec ecx                                ;U proceduri iz koje smo dosli smo ucitali 2 karaktera, a
postojala je samo jedna petlja, pa je neophodno da ecx dekrementiramo.
        ret
isp1 ENDP

isp2 PROC
        push eax
        mov eax, pom4 ;Prva ucitana cifra se odmah ispisuje.
        stosb
        mov eax, pom5 ;Za drugu ucitanu cifru vazi isti postupak kao za prvu cifru u
proceduri isp1.
        and eax, 254
        mov pom5, eax
        mov eax, counter2
        xor edx, edx
        mov dl, outBuffer7[eax]
        and dl, 01h
        mov eax, edx
        add eax, pom5
        stosb
        pop eax
        stosb
        inc counter2
        inc counter2
        dec ecx                                ;Ovde se ecx dekrementira dva puta jer smo ucitali 3
karaktera, a imamo samo jednu petlju.
        dec ecx
        ret
isp2 ENDP

isp3 PROC
        push eax

```

```

    mov eax, pom4 ;Prva ucitana cifra se odmah ispisuje.
    stosb
    mov eax, pom5 ;Druga ucitana cifra se odmah ispisuje.
    stosb
    mov eax, pom6 ;Za trecu ucitanu cifru vazi isti postupak kao za prvu cifru u
proceduri isp1.
    and eax, 254
    mov pom6, eax
    mov eax, counter2
    xor edx, edx
    mov dl, outBuffer7[eax]
    and dl, 01h
    mov eax, edx
    add eax, pom6
    stosb
    pop eax
    stosb
    inc counter2
    inc counter2
    dec ecx ;ecx se dekrementira 3 puta jer smo ucitali 4 karaktera, a
imamo samo jednu petlju.
    dec ecx
    dec ecx
    ret
isp3 ENDP

process3 PROC
    ;Procedura u kojoj se najvisa bitska ravan izvucena u proceduri process2 ubacuje
    ;kao najniza bitska ravan na mesto sifre koja je otkrivena u proceduri process
    cld
    mov esi, OFFSET buffer
    mov edi, OFFSET outBuffersifra
    mov ecx, LENGTHOF buffer

copy3:
    cmp counter1, 3
    je move_on3
    call row_copy_paste
    loop copy3

move_on3:
    mov edx, esi

loop13:
    lodsb ;Ucitava se karakter.
    call IsDigit ;Proverava se da li je cifra.
    jnz notDigit33 ;Skace se ako je ucitan karakter koji nije cifra, ili ako su
ucitane 3 cifre pa razmak.
    mov pom4, eax ;U pom4 se cuva prva ucitana cifra.
    lodsb ;Ucitava se naredni karakter.
    call IsDigit ;Ispituje se da li je cifra.
    jnz notDigit31 ;Skace se ako je ucitana jedna cifra pa razmak.
    mov pom5, eax ;U pom5 se cuva druga ucitana cifra.
    lodsb ;Ucitava se sledeci karakter.
    call IsDigit ;Ispituje se da li je cifra.
    jnz notDigit32 ;Skace se ako su ucitane dve cifre i razmak.
    mov pom6, eax ;U pom6 se cuva treca ucitana cifra

```

```

        loop loop13      ;U slucaju da su ucitane 3 cifre, sledeci karakter nije cifra
sigurno jer je najveca vrednost piksela 255.
        jmp finish3

```

```

notDigit31:
    call isp1      ;Ispis jedne cifre i razmaka.
    loop move_on3

```

```

notDigit32:
    call isp2      ;Ispis dve cifre i razmaka.
    loop move_on3

```

```

notDigit33:
    push esi
    sub esi, edx
    cmp esi, 1
    jne ispis3
    stosb          ;Ispis novog reda (jer on dolazi posle razmaka).
    pop esi
    loop move_on3
    jmp finish3

```

```

ispis3:          ;Ispis tri cifre i razmaka.
    call isp3
    pop esi
    loop move_on3

```

```

finish3:
    ret
process3 ENDP

```

main PROC;Glavni program

;Korisnik upisuje ime ulazne datoteke, sliku iz koje se otkriva tajna poruka i u koju se ubacuje neka druga poruka/slika:

```

    mWrite "Ime ulazne datoteke, slike iz koje se otkriva tajna poruka i u koju se
ubacuje neka druga poruka/slika?: "
    mov     edx, OFFSET infilename
    mov     ecx, SIZEOF infilename
    call ReadString

```

;Otvoranje datoteke:

```

    mov     edx, OFFSET infilename
    call OpenInputFile
    mov     fileHandle, eax

```

;Proveravanje da li ima gresaka:

```

    cmp     eax, INVALID_HANDLE_VALUE ;Da li postoji greska pri otvaranju datoteke?
    jne     file_ok_in ;Ako nema gresaka, skoci.
    mWrite <"Greska prilikom otvaranja ulazne datoteke.", 0dh, 0ah>
    jmp     quit ;Ako ima gresaka, zavrshi program.

```

file\_ok\_in :

```

    ;Citanje fajla u bafer:
    mov     edx, OFFSET buffer
    mov     ecx, BUFFER_SIZE

```

```

        call    ReadFromFile
        jnc     check_buffer_size ;Proveravanje da li postoji greske pri citanju. Ako ne
postoji, skoci.
        mWrite "Greska u citanju." ;U suprotnom ispisuje se da postoji greska i zatvara se
fajl.
        call    WriteWindowsMsg
        jmp     close_file

;Proveravanje da li je bafer dovoljno veliki:
check_buffer_size :
        cmp     eax, BUFFER_SIZE ;Provera da li je bafer dovoljno veliki.
        jbe     buf_size_ok ;Ako jeste, skoci.
        mWrite <"Greska: bafer nije dovoljno veliki", 0dh, 0ah>;Ako nije, ispisi gresku
        jmp     quit;i zavrshi program.

buf_size_ok :
        mov     buffer[eax], 0 ;Ubacivanje terminatora 0.
        mWrite "Velicina datoteke: "
        mov     stringLength, eax
        call    WriteDec;Ispisivanje koliko je velika datoteka.
        call    Crlf

;Zatvaranje ulaznog fajla:
close_file :
        mov     eax, fileHandle
        call    CloseFile

;Korisnik upisuje ime ulazne datoteke, slike koja se sakriva u prethodno ucitanu sliku iz
koje se procitala skrivena poruka:
        mWrite "Ime ulazne datoteke, slike koja se sakriva u prethodno ucitanu sliku?: "
        mov     edx, OFFSET infilename
        mov     ecx, SIZEOF infilename
        call    ReadString

;Otvoranje datoteke:
        mov     edx, OFFSET infilename
        call    OpenInputFile
        mov     fileHandle, eax

;Proveravanje da li ima gresaka:
        cmp     eax, INVALID_HANDLE_VALUE ;Da li postoji greska pri otvaranju datoteke?
        jne     file_ok_in1 ;Ako nema gresaka skoci.
        mWrite <"Greska prilikom otvaranja ulazne datoteke.", 0dh, 0ah>
        jmp     quit ;Ako ima gresaka, zavrshi program.

file_ok_in1 :
        ;Citanje fajla u bafer:
        mov     edx, OFFSET buffer2
        mov     ecx, BUFFER_SIZE
        call    ReadFromFile
        jnc     check_buffer_size1 ;Proveravanje da li postoji greske pri citanju. Ako ne
postoji, skoci.
        mWrite "Greska u citanju." ;U suprotnom ispisuje se da postoji greska i zatvara
fajl.
        call    WriteWindowsMsg
        jmp     close_file1

;Proveravanje da li je bafer dovoljno veliki:

```

```

check_buffer_size1 :
    cmp     eax, BUFFER_SIZE ;Provera da li je bafer dovoljno veliki.
    jbe     buf_size_ok1 ;Ako jeste, skoci.
    mWrite <"Greska: bafer nije dovoljno veliki", 0dh, 0ah>;Ako nije, ispisi gresku
    jmp     quit;i zavrshi program.

buf_size_ok1 :
    mov     buffer2[edx], 0 ;Ubacivanje terminatora 0
    mWrite "Velicina datoteke: "
    mov     stringLength2, edx
    call    WriteDec;ispisivanje koliko je velika datoteka.
    call    Crlf

;Zatvaranje ulaznog fajla:
close_file1 :
    mov     eax, fileHandle
    call    CloseFile

    call process;Izvlacenje tajne poruke iz datoteke koja je smestena u buffer. Tajna
poruka se smesta u outBuffer

    call process2;Pravljenje tajne poruke koja ce se smestiti u ulaznu datoteku koja
je bila ucitana u buffer.
    ;Izvlacenje sedme bitske ravni iz slike koja je smestena u buffer2. Sedma bitska
ravan(tajna poruka/slika) smesta se u outBuffer7.

;Korisnik unosi naziv izlazne datoteke, datoteke u koju je smestena otkrivena tajna
poruka:
    mWrite "Ime datoteke u koju se smesta otkrivena tajna poruka?: "
    mov     edx, OFFSET outfilename
    mov     ecx, SIZEOF outfilename
    call    ReadString

;Pravljenje izlazne datoteke:
    mov     edx, OFFSET outfilename
    call    CreateOutputFile
    mov     fileHandle, eax

;Proveravanje da li ima gresaka:
    cmp     eax, INVALID_HANDLE_VALUE;Da li ima greske prilikom pravljenja izlazne
datoteke?
    jne     file_ok_out ;Ako nema greske, skoci.
    mWrite <"Greska prilikom pravljenja izlazne datoteke.", 0dh, 0ah>;Ako ima greske,
ispisi poruku
    jmp     quit ;i zatvori program.

file_ok_out :
    ;Ispisivanje bafera u izlaznu datoteku.
    mov     eax, fileHandle
    mov     edx, OFFSET outBuffer
    mov     ecx, LENGTHOF outBuffer
    call    WriteToFile
    mov     eax, fileHandle
    call    CloseFile

```

```
    call process3;Smestanje tajne poruke (sedma bitska ravan slike) koja je izvucena
u proceduri process2
    ;u sliku iz koje je u proceduri process otkrivena tajna poruka. Slika u slici se
smesta u outBuffersifra
```

```
;Korisnik unosi naziv izlazne datoteke, datoteke u koju se smestena slika u kojoj je
skrivena slika:
```

```
    mWrite "Ime izlazne datoteke, slike u koju je sakrivena druga slika?: "
    mov     edx, OFFSET outfile_name
    mov     ecx, SIZEOF outfile_name
    call    ReadString
```

```
;Pravljenje izlazne datoteke:
```

```
    mov     edx, OFFSET outfile_name
    call    CreateOutputFile
    mov     fileHandle, eax
```

```
;Proveravanje da li ima gresaka:
```

```
    cmp     eax, INVALID_HANDLE_VALUE;Da li ima greske prilikom pravljenja izlazne
datoteke?
```

```
    jne     file_ok_out2 ;Ako nema greske, skoci.
```

```
    mWrite <"Greska prilikom pravljenja izlazne datoteke.", 0dh, 0ah>;Ako ima greske,
ispisi poruku
```

```
    jmp     quit;i zatvori program.
```

```
file_ok_out2 :
```

```
    ;Ispisivanje bafera u izlaznu datoteku
```

```
    mov     eax, fileHandle
    mov     edx, OFFSET outBuffersifra
    mov     ecx, LENGTHOF outBuffersifra
    call    WriteToFile
    mov     eax, fileHandle
    call    CloseFile
```

```
;kraj programa
```

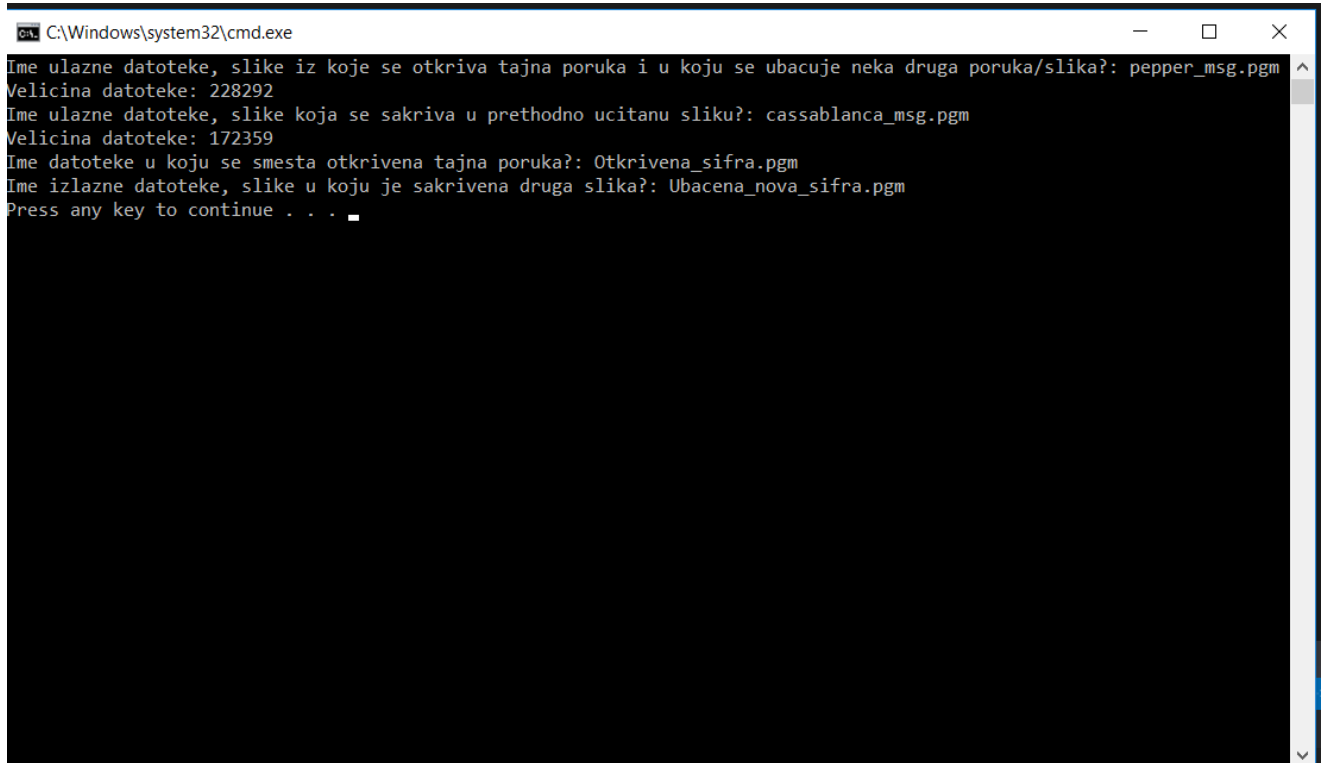
```
quit :
```

```
    exit
    main ENDP
```

```
END main
```

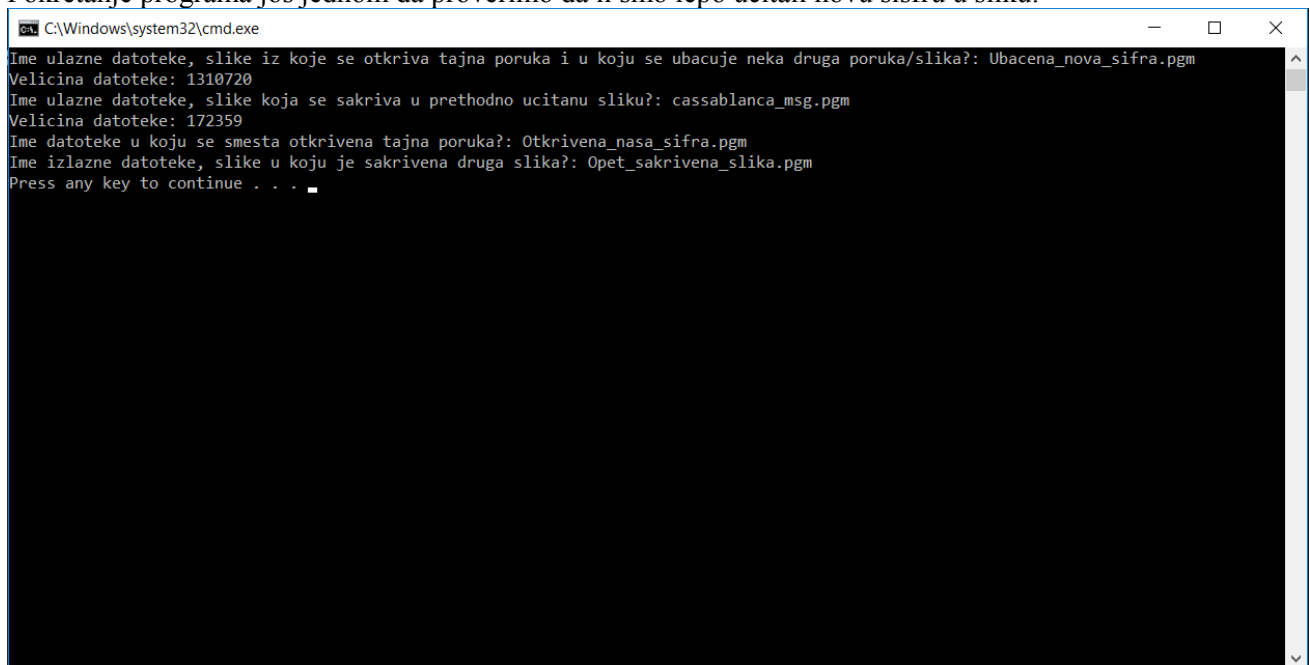


## *Pokretanje programa:*



```
C:\Windows\system32\cmd.exe
Ime ulazne datoteke, slike iz koje se otkriva tajna poruka i u koju se ubacuje neka druga poruka/slika?: pepper_msg.pgm
Velicina datoteke: 228292
Ime ulazne datoteke, slike koja se sakriva u prethodno ucitanu sliku?: cassablanca_msg.pgm
Velicina datoteke: 172359
Ime datoteke u koju se smesta otkrivena tajna poruka?: Otkrivena_sifra.pgm
Ime izlazne datoteke, slike u koju je sakrivena druga slika?: Ubacena_nova_sifra.pgm
Press any key to continue . . .
```

Pokretanje programa još jednom da proverimo da li smo lepo učitali novu šifru u sliku:



```
C:\Windows\system32\cmd.exe
Ime ulazne datoteke, slike iz koje se otkriva tajna poruka i u koju se ubacuje neka druga poruka/slika?: Ubacena_nova_sifra.pgm
Velicina datoteke: 1310720
Ime ulazne datoteke, slike koja se sakriva u prethodno ucitanu sliku?: cassablanca_msg.pgm
Velicina datoteke: 172359
Ime datoteke u koju se smesta otkrivena tajna poruka?: Otkrivena_nasa_sifra.pgm
Ime izlazne datoteke, slike u koju je sakrivena druga slika?: Opet_sakrivena_slika.pgm
Press any key to continue . . .
```