UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Računarska elektronika

Projekat 12, 2016/17

Grupa 14

Vučević Božidar 2014/0383

Borović Jelena 2013/0514

TEKST PROJEKTA

Potrebno je realizovati igru Pattern memory (http://www.primarygames.com/puzzles/memory/patternmemory/).

Tabla se sastoji iz matrice dimenzija N×N, pri čemu su minimalne dimenzije 3×3. Potrebno je podrŽati minimum 5 razliČitih pattern-a (5 nivoa).

Igrica se sastoji u tome da se najpre u odreĐenom vremenskom intervalu (tshow) na ekranu prikazuje zadati pattern, koji se nakon isteka tog vremena uklanja nakon čega korisnik pogaĐa zadati pattern. Pogađanje pattern-a korisnik obavlja tako što se kroz matricu kreće pomoću 4 tastera na tastaturi, i koristi dodatni taster za potvrdu polja za koje misli da pripada pattern-u. Polje koje bi se trenutno selektovalo pritiskom tastera za potvrdu, treba da bude obojeno nekom bojom koja se razlikuje od boje kojom se prikazuje pattern, a koje se nakon pritiska tastera za potrvdu boji u boju koja se koristi za prikazivanje pattern-a.

Nakon što je korisnik K puta, pritiskom tastera za potvrdu izabrao željena polja, proverava se uneti pattern. Ukoliko je pattern koji je korisnik uneo, jednak onom koji je inicijalno prikazan, korisnik dobija poene, i prelazi se na sledeĆi nivo, dok se u suprotnom završava igra.

Broj K opciono može da varira u zavisnosti od nivoa i predstavlja broj elemenata u prikazanom pattern-u. tshow je proizvoljan razuman vremenski interval koji takođe opciono može varirati između nivoa.

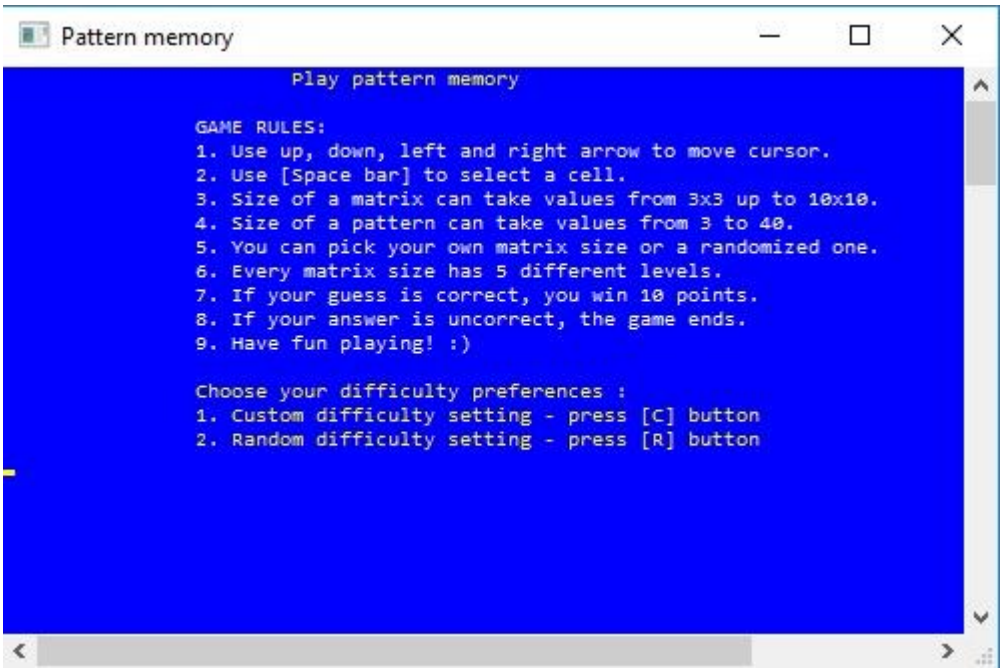
Broj N se definiše unapred i isti je za svaki nivo.

Takođe, treba obratiti pažnju da kada je korisnik označio određeno polje, ukoliko pokuša da ga u istom nivou označi ponovo, ne treba ništa raditi.

REALIZACIJA IGRE

Na početku koda main procedure definisana su opšta podešavanja igre, kao što su veličina prozora, screen *buffer*-a, boja pozadine i boja teksta.

Nakon pozdravne intro poruke, i pravila igre koja su prikazana na ekranu, korisniku se upućuje zahtev za unosom sa izborom *difficulty* opcije: Custom[C] ili Random[R].
Kako to izgleda, prikazano je na sledećoj slici:



Veličina matrice je u programu definisana promenljivom N, a izabrano je da dozvoljene veličine budu od 3x3 do 10x10 (N iz opsega {3.. 10}). Pod *Custom* varijantom podrazumeva se da korisnik sam bira veličinu matrice N, i nakon tog izbora, pritiskom na taster C/c, korisnik unosi željenu veličinu matrice. *Random* varijanta nudi automatski izbor veličine matrice, koji je realizovan pomoću random generatora. Ukoliko je korisnik odabrao ovu mogućnost, na ekranu će se prikazati dobijena vrednost N, da bi korisnik bio obavešten o veličini matrice.

Nakon što je izabrana vrednost N, počinje proračun vrednosti potrebnih za iscrtavanje šablona.

Za proračun šablona korišćene su promenljive *minK* i *maxK*, koje predstavljaju minimalnu i maksimalnu veličinu šablona za datu veličinu matrice. Za minimalnu vrednost odabrana je približna četvrtina broja polja (gde je broj polja N x N), pomoću konstante *maxP* = 4

minK = (N x N) / maxP,

a za maksimalnu vrednost približne polovine broja polja), korišćenjem *minP* = 2

maxK = (N x N) /minP.

Dobijeni rezultati se koriste za biranje random broja iz opsega *[minK, maxK)*, i dobijeni broj predstavlja veličinu K, odnosno veličinu šablona. Kod računa maksimalne vrednosti, napravljen je izuzetak za neparne vrednosti N, a samim tim i neparan broj polja matrice. Kod ovih vrednosti, dobijeni *maxK* se povećava još za jedan, obzirom da se deljenjem sa 2 dobija ostatak 1, pa je taj ostatak vraćen, kako bi se u randomizaciji dobio veći maksimalan broj.

Pre same randomizacije broja K stavljena je labela *gameLoop*, na koju se vraća nakon svakog prelaska na sledeći nivo. Razlog za to je što se *maxK* i *minK* tako računaju samo jednom, a K se sa novim nivoom bira opet kao random broj.

Promena nivoa je zamišljena kao različit šablon za istu veličinu matrice koja je uneta na početku, i ima ih 5 po individualnoj igri.

Osnovna boja za matricu je bela, dok je osnovna boja šablona, odnosno pattern-a crvena. Boja koja je odabrana za kretanje kroz matricu je žuta.

Iscrtavanje matrice se obavlja procedurom *drawMatrix* koja prima argumente a, b i M, pri čemu a i b predstavljaju početne x i y koordinate iscrtavanja matrice.

Procedura *drawMatrix* koristi proceduru *drawCell*, koja iscrtava samo jedno polje matrice i ima dva argumenta: x i y koordinatu gornjeg levog ugla polja. Polje se iscrtava u 3 reda, po 3 karaktera koda *0DBh*.

U .data segmentu programa definisana su dva niza:

jumpCell BYTE 12, 16, 20, 24, 28, 32, 36, 40, 44, 48

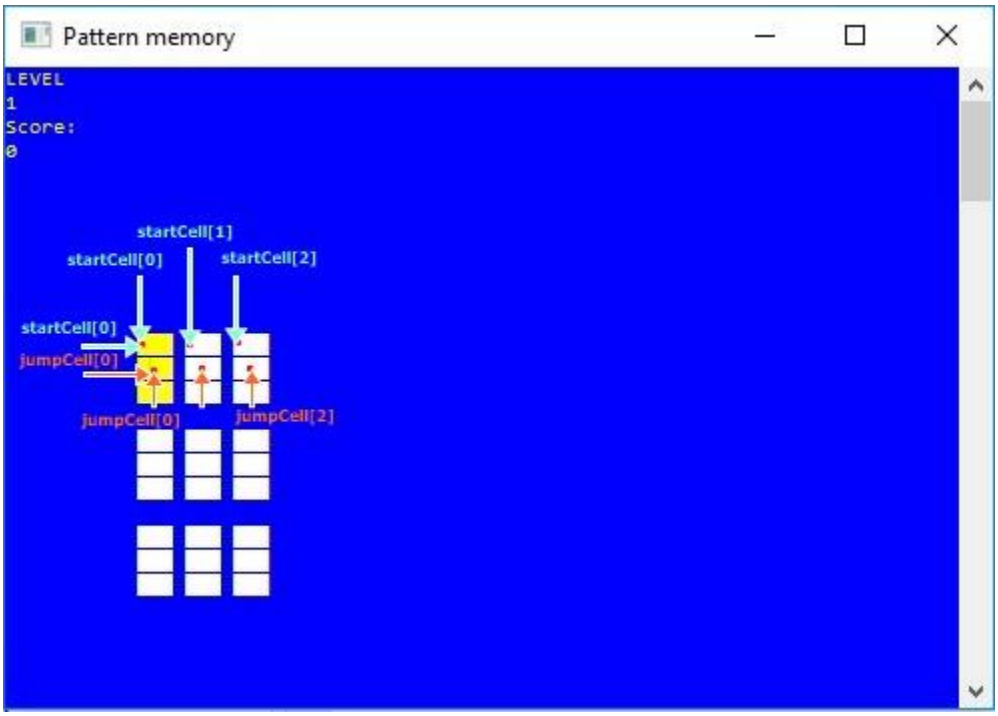
startCell BYTE 11, 15, 19, 23, 27, 31, 35, 39, 43, 47

Namena *jumpCell* niza je da obezbedi koordinate za poziciju kursora kada se vrši kretanje istog. Niz *startCell* obezbeđuje koordinate početnog karaktera jednog polja matrice, odnosno gornjeg levog ugla polja. Razmaci između koordinata u istom nizu su 4, i to je iskorišćeno za kretanje kroz matricu. Veza jumpCell i startCell niza je

startCell[i] = jumpCell[i] – 1

što je u kodu iskorišćeno za povezivanje pozicije početka iscrtavanja polja sa pozicijom kursora na tom polju. Koordinate su izabrane simetrično, da bi bio potreban samo jedan niz za obe ose, i x i y. Nakon unosa veličine N, nizovi se koriste samo do N-1-og elementa.

Na slici je prikazan primer za matricu 3x3:



Niz *startCell* je iskorišćen za iscrtavanje šablona, koji je takođe randomizovan. U ovom slučaju random procedurom se bira broj u opsegu [0, N-1], koji će predstavljati indeks u datom nizu. Ovo se radi dva puta, za x I y koordinatu. Sa ovim vrednostima se poziva procedura *drawCell*, koja sada boji polje u crvenu boju, što je pre iscrtavanja podešeno pozivom *SetTextColor*. U toku ove randomizacije formira se niz *patternArray* sa elementima velicine WORD u kom se čuvaju dobijene coordinate u format dh|dl, da bi se kasnije koristio za poređenje sa unosom korisnika, i te koordinate predstavljaju gornji levi ugao polja.

Kretanje kroz matricu vrši se strelicama levo, gore, desno i dole, a biranje određenog polja za pogađanje space tasterom. Kursor je u slučaju kretanja postavljen na sredinu polja.
Pomeraji kroz matricu su dati kao promene dh ili dl koordinata za vrednost 4, što je objašnjeno nizom jumpCell i detaljno komentarisano u kodu koji će biti predstavljen na narednim stranicama.

Prilikom korisnikovog unosa se takođe formira niz, sa nazivom *userArray*, u kom se čuvaju koordinate polja koje je korisnik obeležio crvenom bojom. Pomoću ovog niza vrši se provera da li je polje već selektovano ili ne.

Provera korisnikovog unosa se vrši dok god se ne naiđe na prvu grešku, s tim što je indikator greške *mistake* postavljen samo u pozadini igre, tako da korisnik i nakon greške ima pravo da izabere K polja, odnosno onoliko polja koliko je i veličina šablona.

Ukoliko je unos tačan prelazi se na sledeći nivo, pri čemu se čiste svi indikatori i nizovi *patternArray* i *userArray*, i skače se na labelu *gameLoop*, dok god broj nivoa ne dođe do broja 6. Tada se prelazi na prozor sa porukom čestitke i upita korisniku da li želi da igra ponovo.

Ukoliko je unos pogrešan, završava se igra sa obaveštenjem GAME OVER! i, takođe, slanjem upita korisniku da li želi novu igru.

.DATA SEGMENT PROGRAMA

Slede kratka objašnjenja poruka u kodu, u sekciji Messages, čije je komentarisanje izostavljeno u samom kodu, radi lakše čitljivosti.

- *gameName BYTE "Pattern memory", 0* – Naziv igre.
- *introMsg BYTE "Play pattern memory ", 0ah, 0dh, 0* – Intro poruka.
- *levelMsg BYTE "LEVEL ", 0ah, 0dh, 0* – Poruka ispod koje se nalazi broj trenutnog nivoa igre.
- *matrixMsg BYTE 0ah, 0dh, " Size of matrix [3-10] ? ", 0dh, 0ah, 0* – Upit korisniku da unese veličinu matrice.
- *nMsg BYTE 0ah, 0dh, " Size of matrix is: ", 0dh, 0ah, 0* – Poruka koja se prikazuje u Random izboru veličine matrice.
- *nextMsg BYTE* - Prelazna poruka za promenu nivoa
- *playMsg BYTE*– Upit u više redova, upućen korisniku za ponovnu igru sa uputstvima za izbor (y/Y taster za potvrdan odgovor I n/N za odričan).
- *pointsMsg BYTE 0ah, 0dh, "Score:", 0ah, 0dh, 0* – Poruka ispod koje se nalazi broj ostvarenih bodova.
- *startMsg BYTE "Let's begin! :) ", 0ah, 0dh, 0* – Poruka za označavanje početka igre.
- *winMsg BYTE "CONGRATULATIONS!", 0ah, 0dh, 0* – Čestitka za pređenih svih 5 nivoa.
- *gameRules BYTE* – Poruka u više redova u kojoj su prikazana pravila igre.
- *difficultyMsg BYTE* – Poruka u više redova za izbor načina biranja veličine matrice, sa uputstvima za izbor(c/C taster za custom I r/R za random).
- *errorMsg BYTE 0dh, 0ah, "Your input is invalid. Please recheck the rules and enter again ", 0dh, 0ah, 0* – Poruka za signal greške u unosu.
- *gameOverMsg BYTE " GAME OVER!", 0dh, 0ah, 0* – Signal da je igra završena.

KOD PROGRAMA

U nastavku je dat detaljno iskomentarisan kod programa.

```
INCLUDE Irvine32.inc
INCLUDE macros.inc

; Segment poruka koda(Messages, Rules, Error) je detaljno objasnjen u fajlu
PatternMemory_Izvestaj.pdf
; Komentari su izostavljeni u samom kodu, radi lakse citljivosti datog dela koda.

.data
gameName BYTE "Pattern memory", 0

; ------------------------------MESSAGES---------------------------------------------------------- -
introMsg BYTE "                         Play pattern memory ", 0ah, 0dh, 0

levelMsg BYTE "LEVEL ", 0ah, 0dh, 0

matrixMsg BYTE 0ah, 0dh, "              Size of matrix [3-10] ? ", 0dh, 0ah, 0

nMsg BYTE 0ah, 0dh, "                   Size of matrix is: ", 0dh, 0ah, 0

nextMsg BYTE 0ah, 0dh, "                CORRECT!", 0ah, 0dh
        BYTE "                  Next level", 0ah, 0dh, 0

playMsg BYTE 0ah, 0dh, "                Play again? ", 0ah, 0dh
        BYTE "                  Yes - Press [Y] button", 0ah, 0dh
        BYTE "                  No - Press [N] button", 0ah, 0dh, 0

pointsMsg BYTE 0ah, 0dh, "Score:", 0ah, 0dh, 0

startMsg BYTE "                         Let's begin! :) ", 0ah, 0dh, 0

winMsg BYTE "                   CONGRATULATIONS!", 0ah, 0dh, 0
```

```
; ------------------------------ - RULES------------------------------------------------------------ -
gameRules       BYTE 0ah, 0dh, "                GAME RULES:", 0ah, 0dh
                BYTE "          1. Use up, down, left and right arrow to move cursor.", 0ah, 0dh
                BYTE "          2. Use [Space bar] to select a cell.", 0ah, 0dh
                BYTE "          3. Size of a matrix can take values from 3x3 up to 10x10.", 0ah,
        0dh
                BYTE "          4. Size of a pattern can take values from 3 to 40.", 0ah, 0dh
                BYTE "          5. You can pick your own matrix size or a randomized one.", 0ah,
        0dh
                BYTE "          6. Every matrix size has 5 different levels.", 0ah, 0dh
                BYTE "          7. If your guess is correct, you win 10 points.", 0ah, 0dh
                BYTE "          8. If your answer is uncorrect, the game ends.", 0ah, 0dh
                BYTE "          9. Have fun playing! :)", 0ah, 0dh, 0

; ------------------------------ - DIFFICULTY-------------------------------------------------------
difficultyMsg   BYTE 0ah, 0dh, "                Choose your difficulty preferences :", 0ah, 0dh
                BYTE "          1. Custom difficulty setting - press [C] button", 0ah, 0dh
                BYTE "          2. Random difficulty setting - press [R] button", 0ah, 0dh, 0

; ------------------------------ - ERROR------------------------------------------------------------ -
errorMsg BYTE  0dh, 0ah, "Your input is invalid. Please recheck the rules and enter again ", 0dh,
0ah, 0

gameOverMsg BYTE "          GAME OVER!", 0dh, 0ah, 0

; ------------------------------ - MATRIX_&_PATTERN_VARIABLES----------------------------------------
K DWORD ? ; Izracunata velicina sablona(pattern - a) na osnovu zadate velicine matrice, K < N.
maxK BYTE ? ; Maksimalna smislena velicina sablona odredjena na osnovu velicine matrice.
minK BYTE ? ; Minimalna smislena velicina sablona odredjena na osnovu velicine matrice.
maxP BYTE 4 ; Pomocna velicina za odredjivanje minK.
minP BYTE 2 ; Pomocna velicina za odredjivanje maxK.
N DWORD ? ; Velicina matrice N x N, uneta od strane korisnika, ili izabrana random
generatorom.
P DWORD ? ; P = 2 * K, pomocna promenljiva za indeksiranje kroz nizove.
tempNum DWORD ? ; Privremena promenljiva za unos.
```

```
; ------------------------------ - SCORE_VARIABLES---------------------------------------------- -
correct DWORD 0 ; Trenutni broj tacnih pogodaka sablona.
hits DWORD 0; Trenutni broj izabranih polja matrice.
levelNo BYTE 1, 2, 3, 4, 5; Niz nivoa igre.
mistake BYTE 0; Indikator da je u korisnikovom pogadjanju pronadjena greska.
points BYTE 0; Broj osvojenih bodova.

; ------------------------------ - POSITION_VARIABLES-----------------------------------------------
currentPos WORD ? ; Trenutna pozicija kursora
currentRed BYTE 0; Indikator da je trenutno polje vec selektovano od strane korisnika.
jumpCell BYTE  12, 16, 20, 24, 28, 32, 36, 40, 44, 48; Niz simetricnih koordinata(dl, dh->svaki po BYTE) pozicija kursora (sredista polja)
startCell BYTE 11, 15, 19, 23, 27, 31, 35, 39, 43, 47; Niz simetricnih koordinata(dl, dh->svaki po BYTE) gornjeg levog ugla polja
previousPos WORD ? ; Prethodna pozicija kursora
previousRed BYTE 0; Indikator da je prethodno polje vec selektovano od strane korisnika.
tempx BYTE ? ; Pomocna promenljiva za x koordinatu, za pozivanje procedure.
tempy BYTE ? ; Pomocna promenljiva za y koordinatu, za pozivanje procedure.
patternArray WORD 49 dup(? ); Niz koordinata(dh | dl = dx(WORD)) sablona koji je dobijen randomizacijom
userArray WORD 49 dup(0) ; Niz korisnikovih unosa koji se poredi sa patternArray
userMove WORD ? ; Korisnikov unos sa tastature.
userTemp WORD ? ; Pomocna promenljiva za proveru da li je polje ranije vec selektovano.

; ------------------------------ - COLORS---------------------------------------------- -
colorW DWORD white + (lightBlue * 16); Bela boja sa plavom pozadinom, osnovna boja matrice i odredjenog dela teksta.
colorY DWORD yellow + (lightBlue * 16); Zuta boja sa plavom pozadinom, za veci deo ispisanog teksta i za kretanje kursora.
colorR DWORD lightRed + (lightBlue * 16); Crvena boja sa plavom pozadinom, za poruke greski ili pobede, i prikazivanje / selektovanje sablona.

; ------------------------------ - CONSOLE_CONTROL---------------------------------------------- -
outHandle DWORD ? ; handle za standardni izlaz.
consoleHandle HANDLE 0; handle za podesavanje prozora konzole
windowRect SMALL_RECT <0, 0, 20, 80>; levo, vrh, desno, dno
cursorInfo CONSOLE_CURSOR_INFO <>; struktura kursora
consoleInfo CONSOLE_SCREEN_BUFFER_INFO <>; struktura screen buffer - a
```

```
scrSize COORD <90, 50>;velicina prozora u koordinatama (broj redova i kolona)

.code
; ------------------------------DRAWCELL_PROCEDURE---------------------------------------- -
drawCell PROC c uses eax ecx edx, y:BYTE, x : BYTE

	mov ecx, 3;Broj prelazaka u novi red.
row :
mov al, 0DBh; Karakter koji se koristi za iscrtavanje polja smesta se u al deo eax registra.
call WriteChar
call WriteChar
call WriteChar
inc dh; Nakon 3 ispisa datog karaktera prelazi se u novi red.
mov dl, x; Vrednost broja kolone se postavlja na pocetnu, koja je prosledjena kao argument
procedure.
call Gotoxy

loop row; Petlja jednog reda polja se ponavlja 3 puta.

mov dl, x
mov dh, y
call Gotoxy; Kursor se vraca na pocetnu poziciju iscrtavanja(gornji levi ugao polja).

ret
drawCell ENDP

; ------------------------------DRAWMATRIX_PROCEDURE---------------------------------------- -
drawMatrix PROC c uses eax esi edi edx, a:BYTE, b : BYTE, M : DWORD

LOCAL y1 : BYTE, x1 : BYTE, xcnt : BYTE, ycnt : BYTE

mov xcnt, 0
mov ycnt, 0; Lokalne pomocne promenljive xcnt i ycnt se postavljaju na 0.
mov eax, 0;Vrednost eax registra se takodje postavlja na 0.

rows :
mov xcnt, 0;Pocetna vrednost lokalne promenljive xcnt se postavlja na 0.
```

```
row :
mov al, xcnt
mov ah, 0
imul  ax, 4; Vrednost lokalne promenljive xcnt se smesta u ax registar, mnozi sa 4 i sabira sa argumentom b
; Argument b predstavlja pocetnu x koordinatu crtanja matrice.
add al, b
mov x1, al; Dobijeni rezultat se smesta u lokalnu promenljivu x1, koja predstavlja x koordinatu sledeceg polja matrice.

mov al, ycnt
mov ah, 0
imul ax, 4
add al, a
mov y1, al; Lokalna promenljiva y1 se dobija na isti nacin kao x1, s tim sto ona predstavlja y koordinatu sledeceg polja matrice.

push DWORD PTR x1
push DWORD PTR y1; Dobijene vrednosti se prosledjuju na stek za poziv procedure drawCell.

mov dl, x1
mov dh, y1
call Gotoxy
call DrawCell; Nakon postavljanja kursora na poziciju datih vrednosti, poziva se procedura.

inc xcnt
mov al, BYTE PTR M; Vrednost xcnt se inkrementuje i poredi sa M(velicina matrice)
cmp xcnt, al
jl row; Dok god je xcnt < M, ponavlja se petlja row, odnosno iscrtavaju se polja u jednom redu matrice.

        inc ycnt
        mov al, BYTE PTR M; Vrednost ycnt se takodje inkrementuje i poredi sa M.
        cmp ycnt, al
        jl rows; Dok god je ycnt < M ponavlja se petlja rows, odnosno iscrtavaju se novi redovi matrice.

        inc b
```

```
        inc a
mov dl, b
mov dh, a
call Gotoxy; Kursor se postavlja na poziciju sredista gornjeg levog polja.

ret
drawMatrix ENDP

; ----------------------------- - ENDWINDOW PROCEDURE---------------------------------------- -
endWindow PROC c uses eax edx, msg1:DWORD, msg2 : DWORD, msg3 : DWORD

call Clrscr; Brise trenutno stanje ekrana.

push colorY
push colorR
push msg1
call changeColorMsg; Poziva proceduru za ispis poruke sa promenom boje.
; Poruka zavisi od ishoda igre, odnosno da li je predjen poslednji nivo, ili je igra zavrsene nakon pogresnog unosa.

mov edx, msg2
call WriteString

call ReadChar; Ceka na korisnikov unos karaktera Y(es) ili N(o)

.IF al == 059h || al == 079h; Y[Ascii] = 59h, y[Ascii] = 79h
jmp Restart; Ukoliko korisnik zeli da igra ponovo, skace se na Restart labelu

.ELSEIF al == 04Eh || al == 06Eh; N[Ascii] = 04h, n[Ascii] = 06h
jmp endGame; Ukoliko korisnik ne zeli da igra ponovo, izlazi se iz igre.

.ELSE; Ukoliko je unet neki drugi karakter, ispisuje se poruka o gresci, i ponovo se poziva procedura.
push msg3
call writeRedMsg

mov eax, 1000
call Delay
```

```
push msg3
push msg2
push msg1
call endWindow
.ENDIF

ret
endWindow ENDP

; ------------------------------ - CHANGE_COLOR_MESSAGE_PROCEDURE------------------------------- -
changeColorMsg PROC c uses eax edx, msg: DWORD, color1 : DWORD, color2 : DWORD

mov eax, color1
call SetTextColor; podesava boju teksta u boju prosledjenu proceduri kao prvi argument

mov edx, msg
call WriteString; stampa poruku koja je prosledjeni proceduri kao offset date poruke

mov eax, color2
call SetTextColor; podesava novu boju teksta(ili vraca staru) nakon ispisa

ret
changeColorMsg ENDP

; ------------------------------ - RED_MESSAGE_PROCEDURE------------------------------- -
writeRedMsg PROC c uses eax edx, msg: DWORD

push colorY
push colorR
push msg
call changeColorMsg; detaljnija varijanta procedure changeColorMsg, za ispise poruka crvenom
bojom

ret
writeRedMsg ENDP

; ------------------------------COLORCELL_PROCEDURE-----------------------------------------
colorCell PROC c uses eax edx, color:DWORD, t : WORD
```

```
LOCAL tx : BYTE, ty : BYTE

mov eax, color
call SetTextColor; Podesava se boja teksta u boju prosledjenu procedurom

mov dx, t; U registar dx se smesta trenutna pozicija kursora
dec dh
dec dl; Obe koordinate se smanjuju za 1, posto vazi relacija StartCell[i] = JumpCell[i] - 1
call Gotoxy; Na taj nacin se kursor postavlja u gornji levi ugao polja, koji koristimo za iscrtavanje polja

mov tx, dl
mov ty, dh
push DWORD PTR tx
push DWORD PTR ty
call DrawCell; poziva se procedura za iscrtavanje polja, cime se trenutno polje ponovo ispisuje novom bojom koja nam je potrebna

inc dl
inc dh
call Gotoxy; vraca kursor u srediste polja

ret
colorCell ENDP

; ----------------------------- - MAIN_PROCEDURE-----------------------------------------------
main PROC

INVOKE GetStdHandle, STD_OUTPUT_HANDLE; Uzima se handle za standardni izlaz.
mov outHandle, eax

INVOKE SetConsoleScreenBufferSize,
consoleHandle, scrSize; podesava se velicina screen buffer - a kao broj redova i broj kolona

INVOKE SetConsoleWindowInfo,
consoleHandle,
TRUE,
```

```
ADDR windowRect; podesava se velicina i pozicija prozora konzole relativno u odnosu na screen buffer

call Clrscr
INVOKE SetConsoleTitle, ADDR gameName;Podesava se naslov konzole na ime igre.

;------------------------------ INTRO ----------------------------------------------------------
gameStart:

mov eax, colorY
call SetTextColor; Podesava zutu boju teksta
call Clrscr; Brise trenutno stanje ekrana u konzoli.

mov edx, offset introMsg
call WriteString; Ispisuje Intro poruku - Play pattern memory -

push colorY
push colorW
push offset gameRules
call changeColorMsg; Poziva proceduru za promenu boje teksta i ispisuje pravila igre belom bojom.
add esp, 12

mov eax, 1000
call Delay;Cekanje da korisnik procita pravila.

;------------------------------ INPUT ----------------------------------------------------------
diffInput :
mov edx, offset difficultyMsg
call WriteString; Ispisuje poruku za izbor tezine igre(velicina matrice), pri cemu postoji Custom i Random varijanta
call ReadChar; Cita karakter unet sa tastature od strane korisnika.

.IF al == 043h || al == 063h; C[Ascii] = 43h, c[Ascii] = 63h
jmp CUSTOM; Skok na Custom varijantu unosa velicine matrice, odnosno korisnik sam bira velicinu.

.ELSEIF al == 052h || al == 072h; R[Ascii] = 52h, r[Ascii] = 72h
```

```
jmp RANDOM; Skok na Random varijantu unosa velicine matrice, odnosno velicina je izabrana random generatorom.

.ELSE
push offset errorMsg
call writeRedMsg; Bilo koji karakter osim c / C i r / R ce ispisati gresku crvenom bojom i ponoviti poruku za izbor tezine
add esp, 4
jmp diffInput;Skok za ponovni izbor tezine
	.ENDIF

;------------------------------ CUSTOM PART ------------------------------------------------
CUSTOM :
xor eax, eax;Resetuje eax registar na 0.

matrixInput :
	mov edx, offset matrixMsg
	call WriteString; Ispisuje poruku korisniku da unese velicinu matrice
	call ReadDec; Cita decimalni broj unet sa tastature koji se smesta u eax.

	mov tempNum, eax; Sadrzaj eax registra se smesta u pomocnu promenljivu za proveru validnosti.

	.IF tempNum < 3 || tempNum > 10
		push offset errorMsg
		call writeRedMsg; Ukoliko korisnik unese nevalidnu velicinu matrice, ispisuje gresku crvenom bojom
		add esp, 4
		jmp matrixInput; I skace na ponovni ispis poruke za unos velicine matrice.

	.ENDIF

	mov N, eax; Ukoliko je uneta velicina u validnom opsegu, smesta se u promenljivu N.
	jmp skipRandom;Preskace Random varijantu unosa velicine matrice.

;------------------------------ RANDOM PART ------------------------------------------------
RANDOM :
	xor eax, eax
```

```
	call Randomize; Poziv procedure Randomize koja postavlja novi seed za random generator u odnosu na trenutno vreme.
	mov eax, 11; Gornja granica za random generator je 11 - 1 = 10, odnosno maksimalna velicina matrice.
	call RandomRange; Poziv random generatora koji daje vrednosti iz skupa[0, 10]

	.IF eax < 3;U slucaju da random generator da vrednost manju od minimalne velicine matrice {0,1,2}
		add eax, 3; Dodaje se 3 da bi se dobila smislena velicina.
	.ENDIF

	mov N, eax; Broj dobijen randomizacijom se smesta u promenljivu N.
	mov edx, offset nMsg
	call WriteString;Ispisuje string Size of matrix is :
	call WriteDec; Ispisuje decimalnu vrednost sadrzaja eax registra, odnosno dobijeni random broj.

	mov eax, 1000
	call Delay; Poziv funkcije za kasnjenje, da bi korisnik stigao da procita rezultat randomizacije.

	call Clrscr
	mov edx, offset startMsg
	call WriteString;Ispisuje string za pocetak igre Let's begin! :)

	mov eax, 1000
	call Delay; Poziv funkcije za kasnjenje, da bi korisnik stigao da procita poruku.

;------------------------------ PATTERN_CALCULATION ------------------------------------------
skipRandom :
	xor eax, eax
	mov al, BYTE PTR N; Smesta promenljivu N u al deo eax registra.
	; Obzirom da je maksimalna velicina N = 10, koja zauzima 1BYTE, koriscenjem PTR operatora se ne gube informacije.
	mul N; N x N odnosno daje broj polja matrice date velicine N.
	div minP; Deli broj polja sa minP = 2
	mov maxK, al; Dobijeni rezultat smesta u maxK, odnosno maksimalnu velicinu sablona.
```

```
        mov al, BYTE PTR N
        shr al, 1; Shiftuje velicinu N udesno za 1, da bi se dobila informacija o najnizem bitu, odnosno bitu parnosti.
        jnc skipOdd; Ukoliko je C = 0, odnosno poslednji bit, preskace se racun za neparne brojeve.

        inc maxK; Ukoliko je broj neparan, onda se maksimalan broj povecava jos za 1.
        ; Razlog za ovo je sto se deljenjem sa 2 neparnog broja, dobija paran broj.
        ; Velicina maxK se koristi u pozivu RandomRange procedure i daje brojeve iz opsega[0, maxK - 1]
        ; Inkrementovanjem u randomizaciji dobijamo velicinu sablona iz opsega[0, maxK]
        ; Odnosno za neparne velicine matrica dobijamo parnu maksimalnu velicinu sablona, a za parnu velicinu matrice  neparnu velicinu sablona.

skipOdd :
        xor eax, eax
        mov al, BYTE PTR N
        mul N
        div maxP; Deli broj polja sa maxP = 4
        mov minK, al; Dobijeni rezultat smesta u minK, odnosno minimalnu velicinu sablona.

; ------------------------------ - LEVEL_LOOP ----------------------------------------
gameLoop :
        call Randomize
        mov al, maxK; Smesta promenljivu maxK u al deo eax registra koji se koristi za poziv RandomRange
        and eax, 0FFh;Podesava deo eax registra iznad al na 0, da bi se obrisali ostaci od prethodnih igranja.
        call RandomRange

        .IF al < minK; U slucaju da random generator da vrednost manju od minimalne velicine sablona{ 0...minK-1 }
                add al, minK; Dodaje se minK da bi se dobila smislena velicina.
        .ENDIF

        mov K, eax; Dobijeni rezultat se smesta u promenljivu K.
        mul minP; Sadrzaj eax registra se mnozi sa 2
```

```
        mov P, eax; Dobijeni rezulat se smesta u promenljivu P, koju koristimo za indeksiranja kroz nizove velicine WORD.

; ------------------------------GAME_START---------------------------------------------------------- -
mov eax, colorY
call SetTextColor

call Clrscr
mov edx, offset levelMsg
call WriteString;Ispis stringa Level:

xor eax, eax
mov al, levelNo
call WriteDec;Ispis decimalnog sadrzaja eax registra, odnosno trenutnog nivoa igre.

mov edx, offset pointsMsg
call WriteString; Ispis stringa Points :

mov al, points
call WriteDec; Ispis decimalnog sadrzaja eax registra, odnosno trenutnog broja bodova.

mov eax, colorW
call SetTextColor

mov dh, startCell[0]
mov dl, startCell[0]
call Gotoxy; Podesava poziciju kursora na pocetnu poziciju koordinata pocetka polja.

push N
push 11
push 11
call drawMatrix; Poziv procedure za iscrtavanje matrice sa parametrima : velicina matrice, pocetne koordinate iscrtavanja
add esp, 12

INVOKE GetConsoleCursorInfo, outHandle,
ADDR cursorInfo
```

```
mov  cursorInfo.bVisible, 0
INVOKE SetConsoleCursorInfo, outHandle,
ADDR cursorInfo; Podesavanje nevidljivosti kursora  dok se iscrtava i prikazuje sablon

.IF levelNo > 1; Ukoliko je nivo veci od 1
        mov eax, 1000
        call Delay; Dodaje pauzu na prelasku izmedju nivoa
.ENDIF

mov eax, colorR
call SetTextColor; Podesava boju teksta za iscrtavanje sablona

; --------------------------------PATTERN-------------------------------------------------------
mov ebx, 0;Pocetna vrednost ebx registra se podesava na 0, i predstavlja indeks u patternArray
nizu.

patternLoop :
        call Randomize
        mov eax, N
        call RandomRange; Poziv random generatora za opseg[0, N - 1]
        mov esi, eax; Dobijeni random rezultat smestamo u esi registar.

        mov eax, N
        call RandomRange
        mov edi, eax; Dobijeni random rezultat smestamo u edi registar.

        mov dh, startCell[esi]; Dobijene random rezultate koristimo kao indekse u nizu startCell
        mov tempy, dh; esi za y koordinatu
        mov dl, startCell[edi]
        mov tempx, dl; I edi za x koordinatu.
        call Gotoxy; Postavlja poziciju kursora na dobijene koordinate, da bi dato polje bilo
obojeno u crveno.

        .IF ebx == 0; Uslov za prvo polje sablona, da bi se smestio prvi element patternArray
niza.
                mov patternArray[ebx], dx; Trenutna pozicija kursora(koordinate dh, dl) se
smestaju u patternArray niz.
                inc ebx
```

```
            inc ebx; Inkrementira ebx na sledecu poziciju u nizu.
        .ELSEIF
            mov ecx, ebx;Ukoliko patternArray niz nije prazan, proverava se da li
randomizacija daje dupli rezultat.

alreadyExists:
        cmp dx, patternArray[ecx]; Poredi trenutnu poziciju kursora sa elementima niza
patternArray.
        jz patternLoop; Ukoliko je nadjen isti rezultat vraca se na pocetak randomizacije.
        dec ecx; Dekrementuje ecx registar da bi prosao kroz citav niz.
        loop alreadyExists; ecx se dekrementuje samo jednom, jer loop to radi automatski drugi
put.
        cmp dx, patternArray[0]; Poredi i nulti element, obzirom da loop prestaje nakon sto
dodje do ecx = 0.
        jz patternLoop; Ukoliko je nadjen isti rezultat vraca se na pocetak randomizacije.

        mov patternArray[ebx], dx; Kada je dobijen jedinstven rezultat smesta se u novi element
niza patternArray.
        inc ebx
        inc ebx
        .ENDIF

        push DWORD PTR tempx
        push DWORD PTR tempy
        call drawCell; Poziv procedure za crtanje polja sa dobijenim random koordinatama.
        add esp, 8

        cmp ebx, P
        jne patternLoop; Petlja za randomizaciju sablona se ponavlja dok se ne stigne do 2 * K.

        mov eax, 2000
        call Delay; Pauza za prikaz sablona.

; ------------------------------ - GUESSING-----------------------------------------------------
        mov eax, colorW
        call SetTextColor
```

```
        mov dh, 11
        mov dl, 11
        call Gotoxy
        push N
        push 11
        push 11
        call drawMatrix; Ponovno iscrtava praznu(belu) matricu.
        add esp, 12

        mov dh, jumpCell[0]
        mov dl, jumpCell[0]
        call Gotoxy; Postavlja kursor na gornje levo polje matrice.
        mov currentPos, dx; Postavlja inicijalnu vrednost za trenutnu poziciju, odnosno gornje
levo polje matrice.

        push currentPos
        push colorY
        call colorCell; Polje na kom se nalazi kursor je obojeno u zutu boju.
        add esp, 6

        INVOKE GetConsoleCursorInfo, outHandle,
        ADDR cursorInfo

        mov cursorInfo.dwSize, 100
        INVOKE SetConsoleCursorInfo, outHandle,
        ADDR cursorInfo
        mov  cursorInfo.bVisible, 1
        INVOKE SetConsoleCursorInfo, outHandle,
        ADDR cursorInfo;Vraca vidljivost kursora.

        mov edi, 0

; ----------------------------- MOVING-----------------------------------------------------------
.REPEAT

LookForKey :
        mov  eax, 50
        call Delay
```

```
        call ReadKey; Ceka unos sa tastature.
jz   LookForKey; Petlja se ponavlja dok god korisnik ne izvrsi unos sa tastature.
mov userMove, dx; Korisnikov unos se pamti u promenljivu userMove.

mov eax, N
sub eax, 1
mov esi, eax; esi registar se postavlja na vrednost N - 1.

; Sva pomeranja(levo, gore, desno, dole) su radjena po istom principu, tako da ce biti detaljno iskomentarisano  samo jedno od njih.

; ------------------------------ - LEFT-----------------------------------------------------------------
        .IF userMove == 025h; Virtual Key VK_LEFT = 25H

                mov dx, currentPos
                cmp dl, jumpCell[0]; Poredi vrednost trenutne kolone sa prvom koordinatom iz jumpCell, odnosno krajnje levom kolonom.
                jz stopLeft; Ukoliko je vrednost poredjenja jednakost, ne krece se dalje ulevo.

                mov ax, dx
                dec ah
                dec al; Smestanjem dx u ax registar, i dekrementovanjem ah i al dela dobijamo odgovarajuci gornji levi polozaj polja za datu poziciju kursora, odnosno srediste tog istog polja.
                mov esi, 0; Inicijalizovanje indeksa esi na 0, kojim prolazimo kroz niz polja koje je korisnik selektovao.

                .REPEAT
                        .IF userArray[esi] == ax
                                mov previousRed, 1; Ukoliko je data vrednost koordinata u nizu korisnikovih unosa, pamti se stanje u indikator previousRed
; razlog imenovanja previous(prethodni), jeste taj sto ce do kraja if petlje za odredjeni unos trenutna pozicija biti sacuvana kao prethodna.
                        .ENDIF
                        sub al, 4; Vrsi proveru unapred i za sledecu poziciju unosa, za istu vrednost esi registra.
                        .IF userArray[esi] == ax
```

```
                    mov currentRed, 1; Ukoliko se vrednost koordinata nalazi u nizu
korisnikovih unosa, pamti se stanje u indikator currentRed.
; razlog imenovanja current(trenutni), jeste taj sto ce do kraja if petlje ta pozicija biti u
promenljivoj currentPos.
                    .ENDIF
                    add al, 4; Vraca al za proveru previousRed indikatora.
                    inc esi
                    inc esi
                .UNTIL esi >= P; Nastavlja dalje kroz niz dok ne dodje do maksimuma korisnikovih
unosa.

                mov previousPos, dx; Trenutnu poziciju pamti kao prethodnu.
                sub dl, 4;Vrednost kolone za 4 manja od prethodne je pomeranje ulevo.
                mov currentPos, dx; U novu trenutnu poziciju se pamti ta pozicija (srediste polja
levo od trenutnog)

                .IF previousRed != 1
                    push previousPos
                    push colorW
                    call colorCell; Ukoliko polje sa kog se pomera kursor nije vec
selektovano(crveno), ono se boji u belu boju pozivom procedure colorCell.
                    add esp, 6
                    .ENDIF
                    .IF currentRed != 1
                    push currentPos
                    push colorY
                    call colorCell; Ukoliko polje na koje se pomera kursor nije vec
selektovano(crveno), ono se boji u zutu boju pozivom procedure colorCell.
                    add esp, 6
                .ENDIF

            stopLeft : ; Labela na koju se skace ukoliko nema pomeranja dalje ulevo, i zavrsetak za
gornje if petlje
                mov dx, currentPos
                call Gotoxy; Kursor se postavlja na vrednost trenutne pozicije.
                mov previousRed, 0
                mov currentRed, 0; Indikatori za selektovanje se vracaju na 0 za sledeci potez.
```

```
; ------------------------------UP-------------------------------------------------------------
		.ELSEIF userMove == 026h; VK_UP = 26H
			mov dx, currentPos
			cmp dh, jumpCell[0];  Poredi vrednost trenutnog reda sa prvom
koordinatom niza jumpCell, odnosno najvisim redom.
			jz stopUp

			mov ax, dx
			dec ah
			dec al
			mov esi, 0

			.REPEAT
				.IF userArray[esi] == ax
					mov previousRed, 1
				.ENDIF
				sub ah, 4
				.IF userArray[esi] == ax
					mov currentRed, 1
				.ENDIF
				add ah, 4
				inc esi
				inc esi
			.UNTIL esi >= P

			mov previousPos, dx
			sub dh, 4; Vrednost reda za 4 manja od trenutnog je pomeranje na gore.
			mov currentPos, dx

			.IF previousRed != 1
			push previousPos
			push colorW
			call colorCell
			add esp, 6
			.ENDIF

			.IF currentRed != 1
			push currentPos
```

```
                push colorY
                call colorCell
                add esp, 6
        .ENDIF
        stopUp :
                mov dx, currentPos
                call Gotoxy
                mov previousRed, 0
                mov currentRed, 0

; ------------------------------ - RIGHT----------------------------------------------------------
        .ELSEIF userMove == 027h; VK_RIGHT = 27H
                mov dx, currentPos
                cmp dl, jumpCell[esi]; Poredi vrednost trenutne kolone sa N - 1 - om
koordinatom niza jumpCell, odnosno krajnje desnim redom.
                jz stopRight

                mov ax, dx
                dec ah
                dec al
                mov esi, 0

                .REPEAT
                        .IF userArray[esi] == ax
                                mov previousRed, 1
                        .ENDIF
                        add al, 4
                        .IF userArray[esi] == ax
                                mov currentRed, 1
                        .ENDIF
                        sub al, 4
                        inc esi
                        inc esi
                .UNTIL esi >= P

                mov previousPos, dx
                add dl, 4;Vrednost kolone za 4 veca od trenutne je pomeranje udesno.
                mov currentPos, dx
```

```
                .IF previousRed != 1
                push previousPos
                push colorW
                call colorCell
                add esp, 6
                .ENDIF

                .IF currentRed != 1
                push currentPos
                push colorY
                call colorCell

                add esp, 6
                .ENDIF

                stopRight :
                mov dx, currentPos
                call Gotoxy
                mov previousRed, 0
                mov currentRed, 0

        ; ------------------------------DOWN-----------------------------------------------------------------
                .ELSEIF userMove == 028h; VK_DOWN = 28H

                mov dx, currentPos
                cmp dh, jumpCell[esi]; Poredi vrednost trenutnog reda sa N - 1 - om
koordinatom niza jumpCell, odnosno najnizim redom.
                jz stopDown

                mov ax, dx
                dec ah
                dec al
                mov esi, 0

                .REPEAT
                        .IF userArray[esi] == ax
                                mov previousRed, 1
```

```
                .ENDIF
                add ah, 4
                .IF userArray[esi] == ax
                        mov currentRed, 1
                .ENDIF
                sub ah, 4
                inc esi
                inc esi
        .UNTIL esi >= P

        mov previousPos, dx
        add dh, 4;Vrednost reda za 4 veca od trenutne je pomeranje na dole.
        mov currentPos, dx

        .IF previousRed != 1
                push previousPos
                push colorW
                call colorCell
                add esp, 6

        .ENDIF
        .IF currentRed != 1
                push currentPos
                push colorY
                call colorCell

                add esp, 6
        .ENDIF
        stopDown :
                mov dx, currentPos
                call Gotoxy
                mov previousRed, 0
                mov currentRed, 0

; ----------------------------SPACE BAR---------------------------------------------------- -
                .ELSEIF userMove == 020h; VK_SPACE = 20H

                mov dx, currentPos
```

```
dec dl
dec dh
mov userTemp, dx; Gornji levi ugao polja na kom je trenutno kursor se cuva u pomocnu promenljivu userTemp.

mov ebx, 0
.REPEAT
    mov ax, userArray[ebx]; Smesta u ax registar element niza userArray sa indeksom ebx.

    .IF userTemp != ax; poredi se korisnikov unos sa nizom dosadasnjih unosa
        inc ebx
        inc ebx
    .ELSEIF
    mov ebx, P
    inc ebx

    .ENDIF
    .UNTIL ebx >= P

    .IF ebx == P;ukoliko je ebx jednako P, znaci da nije do sad unet taj element
        mov dx, userTemp
        call Gotoxy
        mov userArray[edi], dx
        inc edi
        inc edi

        mov eax, colorR
        call SetTextColor

    mov tempy, dh
    mov tempx, dl
    push DWORD PTR tempx
    push DWORD PTR tempy
    call drawCell; poziva proceduru za iscrtavanje polja crvenom bojom
```

```
                add esp, 8

                inc hits; Povecava broj zabelezenih selektovanja.
            .ELSE
                jmp skip
            .ENDIF

        mov ecx, 0

; ------------------------------CHECK PATTERN------------------------------------------------- -
cmp mistake, 1; Ukoliko je zabelezena greska provera sablona se preskace u pozadini.
;Korisnik i dalje ima pravo na unos sablona.
jz skip
CheckPattern:
    cmp dx, patternArray[ecx]; poredi vrednost dx registra, koja u ovom trenutku sadrzi
vrednost gornjeg levog ugla polja sa nizom patternArray.
    jz oneCorrect; Ukoliko je pronadjena saglasnost, skace se na labelu oneCorrect.
    inc ecx
    inc ecx
    cmp ecx, P
    jnz CheckPattern; Petlja se ponavlja dok ecx registar ne dostigne vrednost P, odnosno
dupliranu vrednost velicine sablona posto su nizovi velicine WORD.

    mov correct, 0
    mov mistake, 1; Ukoliko nije pronadjena saglasnost, postavlja se indikator greske.
    jmp skip;Nakon postavljanja indikatora greske, preskace se ubrajanje tacnog pogotka
bezuslovnim skokom.

; ------------------------------ ONE CORRECT GUESS ----------------------------------------------
oneCorrect:
    add correct, 1; Inkrementuje vrednost pogodaka za 1 nakon sto je pronadjena
saglasnost u proveri unosa.

    mov ecx, K
    .IF ecx == hits
        jmp nextLevel;Ukoliko se u ovom delu programa dodje do K-og selektovanja,
znaci da je citav sablon unet tacno i prelazi se na sledeci nivo.
    .ENDIF
```

```
        skip:
                        mov dx, currentPos
                        call Gotoxy
        .ENDIF

        mov eax, K

.UNTIL hits == eax; Citava petlja unosa sablona od strane korisnika se ponavlja dok se ne dostigne K selektovanja polja.

cmp mistake, 1;Ukoliko je nakon citavog unosa indikator greske postavljen, preskace se na labelu wrongAnswer.
jz wrongAnswer

; ------------------------------ NEXT LEVEL ------------------------------------------------------
nextLevel:
                mov  cursorInfo.bVisible, 0
                INVOKE SetConsoleCursorInfo, outHandle,
                ADDR cursorInfo; Iskljucuje se vidljivost kursora.

                add points, 10; Vrednost bodova se uvecava za 10.
                inc levelNo; Povecava se vrednost nivoa.

                mov eax, 1000
                call Delay

                mov hits, 0
                mov correct, 0; Vrednosti broja selektovanja i broja tacnih pogodaka se postavljaju na 0.

                mov ecx, 0
                .REPEAT
                        mov userArray[ecx], 0
                        mov patternArray[ecx], 0
                        inc ecx
                        inc ecx
                .UNTIL ecx == P; Vrednosti elemenata niza sablona i niza korisnikovih unosa se postavljaju na 0.
```

```
                    .IF levelNo < 6
                        call Clrscr; Ukoliko je broj nivoa manji od 6 {1..5}, postavlja se prelazni ekran sa obavestenjem o prelasku na sledeci nivo.
                        push colorY; Brisanjem sadrzaja ekrana i pozivanjem procedure changeColorMsg za ispis obavestenja.
                        push colorW
                        push offset nextMsg
                        call changeColorMsg
                        add esp, 12

                        mov eax, 1000
                        call Delay
                    .ENDIF

                    cmp levelNo, 6
                    jnz gameLoop; Citava petlja randomizovanja sablona, iscrtavanja matrice, sablona, i korisnikovog unosa se ponavlja dok je vrednost nivoa manja od 6.

; ------------------------------LEVEL_5 + ---------------------------------------------- ------
                    .IF points == 50
                        push offset errorMsg
                        push offset playMsg
                        push offset winMsg
                        call endWindow;Ukoliko je zavrsen peti nivo, poziva se zavrsni ekran sa cestitkom i pitanjem da li korisnik zeli da igra ponovo.
        add esp, 12
.ENDIF

; ------------------------------ WRONG_GUESS ---------------------------------------------- ------
wrongAnswer :
                push offset errorMsg
                push offset playMsg
                push offset gameOverMsg
                call endWindow; Ukoliko je korisnikov uneti sablon pogresan, poziva se zavrsni ekran sa obavestenjem da je igra gotova i pitanjem da li korisnik zeli da igra ponovo.
                add esp, 12
```

```
; ------------------------------RESTART---------------------------------------------- ------
Restart::
                mov hits, 0
                mov correct, 0
                mov mistake, 0
                mov levelNo, 1
                mov points, 0
                mov ecx, 0

                .REPEAT
                        mov userArray[ecx], 0
                        mov patternArray[ecx], 0
                        inc ecx
                        inc ecx
                .UNTIL ecx == P
                jmp gameStart; Sve vrednosti indikatora i elementi nizova se postavljaju na 0, i
skace se na labelu gameStart od koje se igra pocinje ponovo, sa ispisom pravila i unosom
velicine matrice.

                mov eax, 1000
                call Delay

; ------------------------------END_GAME---------------------------------------------- ------
        endGame::
                exit
                main ENDP;Zavrsetak igre i izlazak iz main procedure.
END main
```