Univerzitet u Beogradu – Elektrotehnički fakultet

Katedra za elektroniku

Projekat iz računarske elektronike
Repeat Pattern Game

Studenti :
Ćirić Andrea 2016/0202
Cvejić Miloš 2016/0145

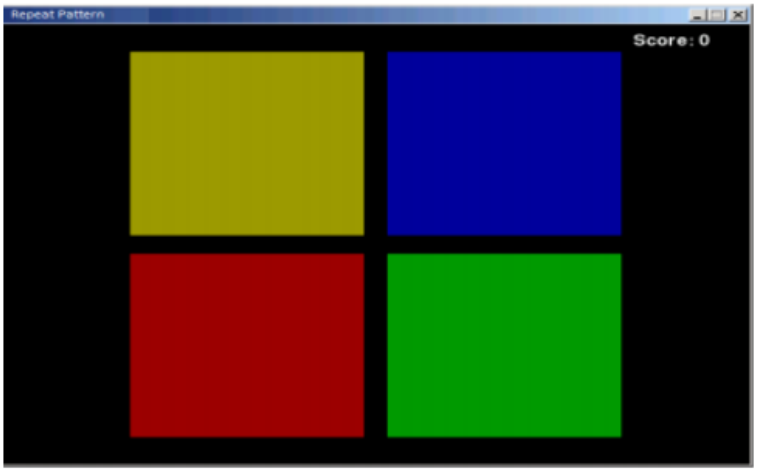
Beograd,
septembar 2020.

Sadržaj:

Projektni zadatak 3
Uputstvo 4
Programski kod 6
main.asm 6
procedure.inc 7
Upotrebljene gotove funkcije iz biblioteke Irvine32.inc 9

Projektni zadatak

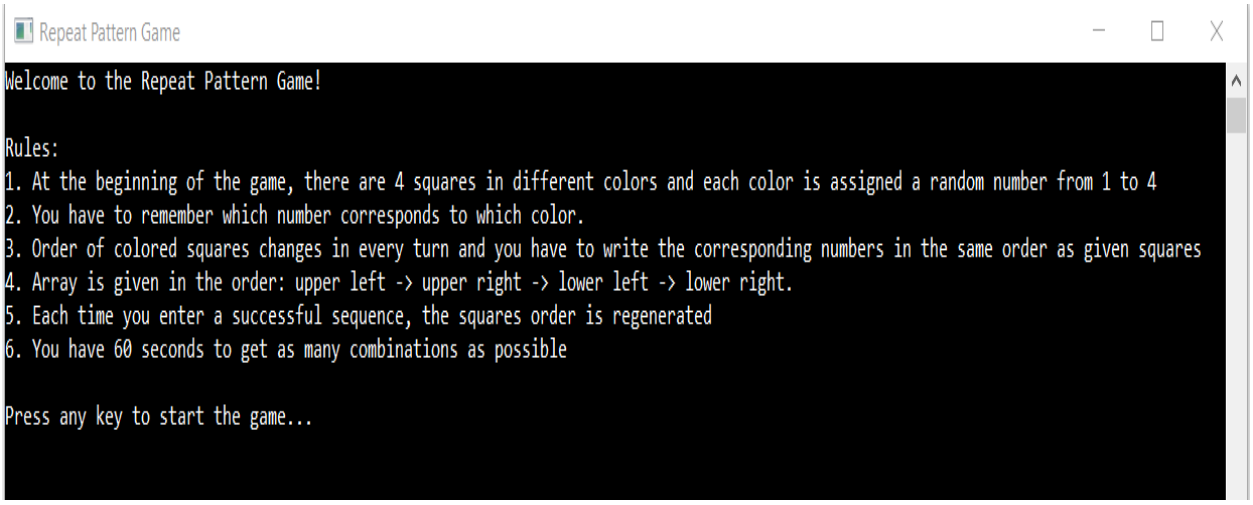
Data je tebela 2x2 polja (polja veličine 8x8) sa različitim bojama. Cilj igrice je da se svaki put pogodi odgovarajuća kombinacija boja koja se prikazuje u tabeli pritiskom na odgovarajuće tastere u odgovarajućem redosledu gde svakoj od boja koje se mogu pojaviti u tabeli odgovara jedan taster. Boje koje se mogu pojaviti su: plava, crvena, žuta i zelena. Bojama se na svakom početku igrice po random šablonu dodeljuju brojevi od 1 do 4 koji zapravo predstavljaju taster koji je neposredno dodeljen toj boji. Nakon ove dodele parovi boja-taster se prikazuju igraču sve dok on ne pritisne taster "space" na tastaturi kako bi zapamtio kombinaciju, odnosno kojoj boji odgovara koji taster(broj), a zatim se kreće u izvršavanje igrice. Boje se pojavljuju u blokovima po slučajnom redosledu i poenta igrice je da se pogodi odgovarajući redosled boja pritiskom na odgovarajće tastere koji pripadaju datim bojama. Sve dok igrač ispravno pogađa boje, tabela se iznova generiše. Ukoliko igrač pogreši igra se prekida uz odgovarajuću poruku i mogućnost da se krene ispočetka.



Slika 1 - slika uz projekat

Uputstvo

Na početku svake igre se prvo pristupa ekranu (slika 2) koji prikazuje pravila po kojima igra funkcioniše čime se korisniku jasno prikazuju smernice neophodne za razumevanje igre.

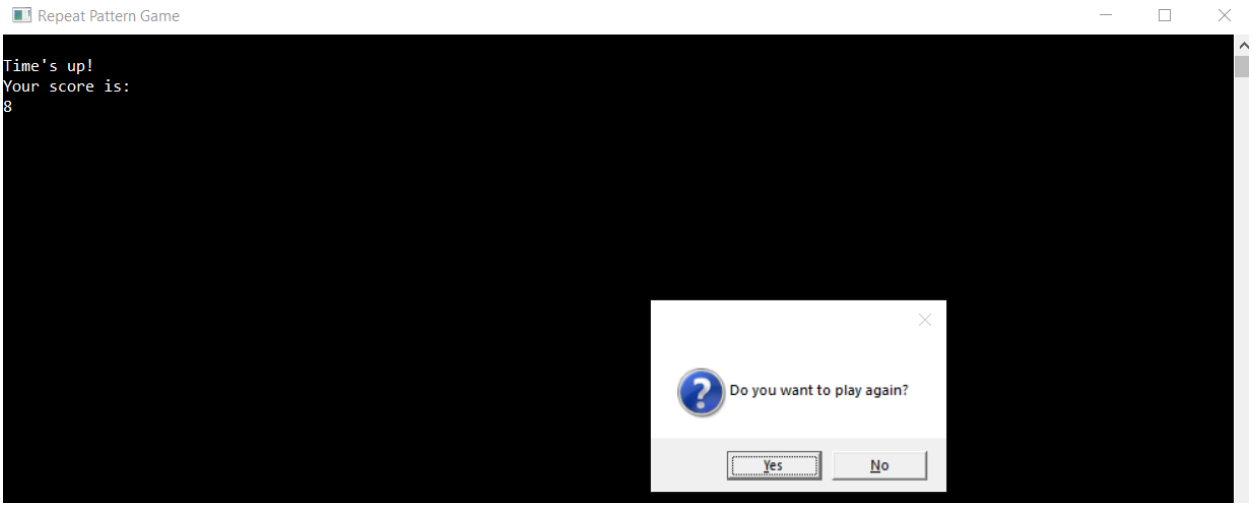


Slika 2 - Pravila igre

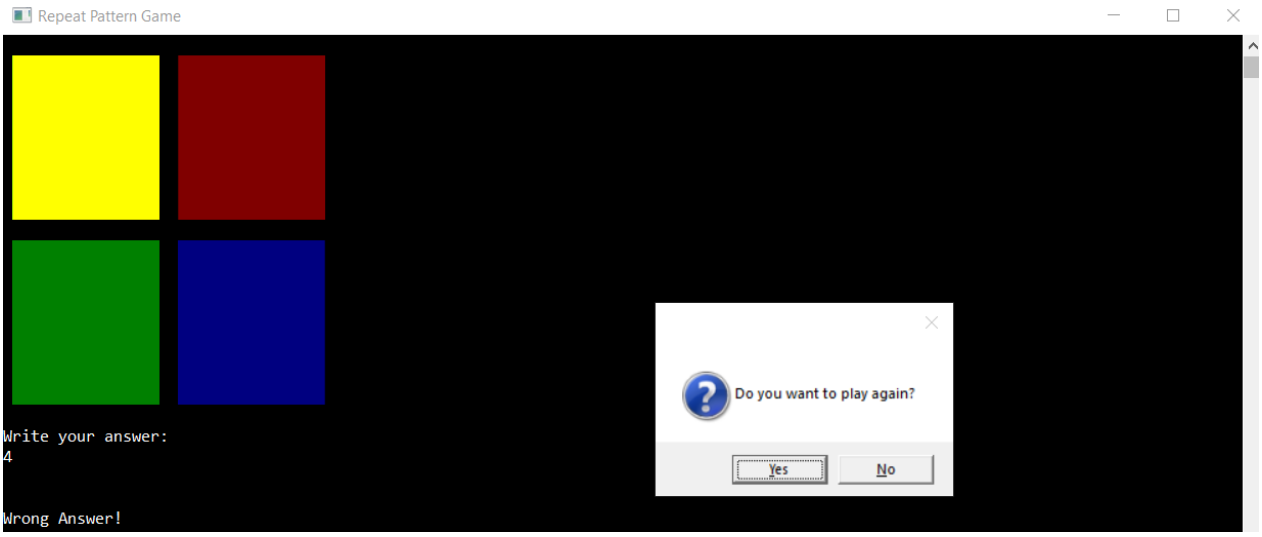
Nakon toga sledi test runda (slika 3) u kojoj svaka od boja dobija svoj ekvivalent na tastaturi. Pristupa se test primeru koji služi za proveru ispravnosti dodele boja.

Slika 3 - test runda

Nakon uspešnog test primera, pritiskom na taster otpočinje igra u kojoj za 60 sekundi treba pogoditi što više kombinacija, od kojih svaka tačna nosi 1 poen. Na kraju svake igre korisnik dobija informaciju o svom rezultatu sa odgovarajućom porukom i mogućnost da ponovo igra (slika 4). Ukoliko dođe do pogrešnog unosa igra se prekida (slika 5).



Slika 4 - kraj igre kada istekne vreme



Slika 5 - kraj igre nakon pogrešnog odgovora

Programski kod

main.asm

U main delu programa se pozivaju procedure ***start_screen*** u kojoj se definiše izgled prozora pri pokretanju igrice, ***example_screen*** u kojoj se definiše izgled prozora u toku test primer i ***game_screen*** u toku igranja igrice. Takođe i prikaz prozora za upit o ponovnom igranju na kraju igrice. Pomenute procedure nalaze se u fajlu ***pages.asm***. Unutar ovih procedura za izgled prozora pozivaju se ostale procedure za obradu podataka. Sve procedure su definisane u ***procedure.inc*** fajlu prikazanom u sledećem poglavlju.

```
INCLUDE Irvine32.inc
INCLUDE procedure.inc

.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD

.const
	n = 4		; //number of elements in array

.data

	assign_array_indicator	BYTE	0
	score			DWORD	0
	PlayAgain_message	BYTE	"Do you want to play again?",
0


.data?

	arrayGame	BYTE n dup(?)		;//array of numbers generated
during the game
	arraySetup	BYTE n dup(?)		;//numbers assigned to colors
	arrayOutColors	BYTE n dup(?)		;//order of colors on screen
	arrayOut	BYTE n dup(?)		;//true answer array

.code
    main PROC

		INVOKE start_screen
	again:
		INVOKE example_screen, OFFSET arrayGame, OFFSET
arrayOutColors, OFFSET arraySetup, assign_array_indicator
		INVOKE game_screen, OFFSET arrayGame, OFFSET arrayOutColors,
OFFSET arraySetup, score
```

```
            ;// play again?
            mov ebx, 0
            mov edx, OFFSET PlayAgain_message
            mov assign_array_indicator, 0
            mov score, 0
            call MsgBoxAsk
            cmp eax, IDNO
            jne again

            INVOKE ExitProcess, 0

      main ENDP
END main
```

procedure.inc

U procedure.inc fajlu nalaze se dekleracije procedura koje se pozivaju u projektu. Procedure su odvojene u .asm fajlove po celinama.

```
.386
.model flat, stdcall

;//-----------------------------------------------------------------------
----------------------------------------
;// This procedure generates a random four element array with different
values [1,4]

random_array PROTO,
                    Arr: PTR BYTE,
                    ArrSetup: PTR BYTE,
                    indicator: BYTE

;//-----------------------------------------------------------------------
----------------------------------------
;// This procedure draws a square in given color at given starting position

draw_square PROTO,
                    xposition: BYTE,
                    yposition: BYTE,
                    color: PTR BYTE,
                    k: PTR DWORD

;//-----------------------------------------------------------------------
----------------------------------------
;//This procedure sets parameters for all four squares  and calls draw_square
procedure for each square
```

```
draw_squares PROTO,
                    Arr: PTR BYTE,
                    OutArr: PTR BYTE

;//-------------------------------------------------------------------------
-----------------------------------------
;//This procedure generates a true answer (AnswerArr)

true_answer PROTO,
                        Arr: PTR BYTE,
                        OutArrColors: PTR BYTE,
                        AnswerArr: PTR BYTE

;//-------------------------------------------------------------------------
-----------------------------------------
;//This procedure reads input and compares it with real (true) answers

get_answer PROTO,
                        Arr: PTR BYTE,
                        indicator: PTR BYTE

;//-------------------------------------------------------------------------
-----------------------------------------
;//This function sets startup screen with game instructions

start_screen PROTO

;//-------------------------------------------------------------------------
-----------------------------------------
;//This function sets example screen

example_screen PROTO,
                    Arr: PTR BYTE,
                    OutArrColors: PTR BYTE,
                    ArrSetup : PTR BYTE,
                    assign_array_indicator: BYTE

;//-------------------------------------------------------------------------
-----------------------------------------
;//This function sets game screen

game_screen PROTO,
                        Arr: PTR BYTE,
                        OutArrColors: PTR BYTE,
                        ArrSetup : PTR BYTE,
                        score: DWORD

;//-------------------------------------------------------------------------
-----------------------------------------
```

Upotrebljene gotove funkcije iz biblioteke Irvine32.inc

Clrscr - The *Clrscr* procedure clears the console window.

WriteString - The *WriteString* procedure writes a null-terminated string to the console window. Pass the string's offset in EDX.

WriteDec - The *WriteDec* procedure writes a 32-bit unsigned integer to the console window in decimal format with no leading zeros. Pass the integer in EAX.

GetMseconds - The *GetMseconds* procedure gets the number of milliseconds elapsed since midnight on the host computer, and returns the value in the EAX register. The procedure is a great tool for measuring the time between events. No input parameters are required.

SetTextColor - The *SetTextColor* procedure sets the foreground and background colors for text output. When calling *SetTextColor*, assign a color attribute to EAX.

ReadChar - The *ReadChar* procedure reads a single character from the keyboard and returns the character in the AL register.

WriteChar - The *WriteChar* procedure writes a single character to the console window. Pass the character (or its ASCII code) in AL.

SetConsoleTitle - The *SetConsoleTitle* function lets you change the console window's title.

Randomize - The *Randomize* procedure initializes the starting seed value of the Random32 and RandomRange procedures. The seed equals the time of day, accurate to 1/100 of a second. Each time you run a program that calls Random32 and RandomRange, the generated sequence of random numbers will be unique. You need only to call Randomize once at the beginning of a program.

RandomRange - The *RandomRange* procedure produces a random integer within the range of 0 to n-1, where n is an input parameter passed in the EAX register. The random integer is returned in EAX.

Gotoxy - The *Gotoxy* procedure locates the cursor at a given row and column in the console window. By default, the console window's X-coordinate range is 0 to 79 and the Y-coordinate range is 0 to 24. When you call *Gotoxy*, pass the Y-coordinate (row) in DH and the X-coordinate (column) in DL.