Projekti iz predmeta Računarska elektronika

Aleksandra Lekić, Ksenija Josipović, Haris Turkmanović

26. mart 2018

Za svaki od zadataka potrebno je dostaviti:

- **Folder sa kodom programa koji zadovoljava traženu funkcionalnost. Ukoliko program generiše izlazne fajlove, potrebno je i njih priložiti.**
- **Izveštaj u kome se opisuje funkcija zadatka i kako je sam program realizovan. Na kraju izveštaja potrebno je dodati ceo kod projekta.**

Izveštaj i programe sačuvati u folderu sa imenom `REx`, gde je `x` broj projekta. Folder u vidu arhive poslati mejlom sa *subject*-om `REx_projekat` na mejl adrese:

Aleksandra Lekić `lekic.aleksandra@etf.rs`
Ksenija Josipović `jk140124d@student.etf.bg.ac.rs`
Haris Turkmanović `th140516d@student.etf.bg.ac.rs`

Odbrana projekata će se organizovati po rasporedu koji će biti naknadno objavljen.

Projekti označeni $*$ se mogu predati prvi dan ispitnog roka: junskog, julskog, septembarskog ili oktobarskog. Ostali projekti se predaju do prvog dana junskog i julskog ispitng roka. Odbrana projekata će biti organizovana u danima pomenutog ispitnog roka po dogovoru.

PGMA format slike

PGMA je *grayscale* ASCII format slike čiji je zapis:

```
P2
# pepper.ascii.pgm created by PGMA_IO::PGMA_WRITE.
256  256
255
26  47  49  43  44  44  46  41  43  42  43  43
42  42  41  39  44  39  42  45  46  42  39  39
42  42  41  39  44  39  42  45  46  42  39  39
36  39  37  37  35  41  37  35  33  34  33  37
...
```

Prvi red čini „magični broj" koji je u slučaju PGMA formata uvek P2. Zatim sledi karakter za novi red `0x0A` i red koji počinje sa `#`. Ovaj red predstavlja komentar. U sledećem redu je data širina slike odvojena sa dva razmaka i visina slike. Četvrti red predstavlja maksimalnu vrednost piksela slike, što je u slučaju 8-bitne slike 255. Od petog reda nadalje su date decimalne vrednosti piksela međusobno odvojene razmakom ili sa razmakom i znakom za novi red.

PGM slika se može prikazati korišćenjem programa *IrfanView* koji se može naći na linku: `http://www.irfanview.com/`. Detaljnije uputstvo je dato u datoteci `Obrada_slike` pod nazivom `pgm_files.pdf` u okviru fajlova za izradu projekta. Dato je i nekoliko slika PGMA formata koje se mogu koristiti za testiranje projekata.

Primer slike.

Na sajtu `http://people.sc.fsu.edu/~jburkardt/data/pgma/pgma.html` se može naći još primera slika, pri čemu ih je za upotrebu u programu potrebno dodatno umanjiti korišćenjem programa *IrfanView*.

Prikaz i promenu dimenzija slike moguće je uraditi i korišćenjem programa MATLAB, korišćenjem sledećih naredbi:

- Učitavanje i prikaz slike:

```
pmga_img = imread('pepper.pgm');
figure; imshow(pgma_img);
```

- Promena dimenzija slike:

```
pgma_res = imresize(pgma_img,[a b]);
```

  Više informacija o načinu definisanja konstanti a i b može se dobiti pozivom komande `help imresize`.

- Čuvanje slike u PGMA formatu:

```
imwrite(pgma_res,'pepper_res.pgm','Encoding','ASCII');
```

  Jako je bitno specificirati encoding type prilikom čuvanja slike, jer ukoliko se on ne navede kao rezultat dobiće se binarni pgm fajl, odnosno PGMB format. Takođe, MATLAB generiše PGMA fajl tako što u jednom redu stoje, odvojeni razmacima, „magični broj", dimenzije slike i maksimalni intenzitet u slici, pa je radi konzistentnosti sa prethodnim opisom fajla potrebno korišćenjem nekog text editora, modifikovati fajl tako da ima formu shodno definisanim pravilima.

U narednih šest projekta potrebno je napraviti program koji po pokretanju pita za ime slike koju je potrebno obraditi. Nakon izvršene obrade slike, pita se za naziv izlazne slike i upisuju se obrađeni podaci.

Projekat 1 - skaliranje slike primenom bilinearne transformacije

Napisati program za skaliranje slike primenom bilinearne transformacije. Faktor s, sa kojim se vrši skaliranje, unosi se sa standardnog ulaza i uvek je veći ili jednak 1. Pored toga, sa standardnog ulaza na određeni način definiše se da li se vrši povećanje ili decimacija slike s puta.

.....

Povećanje dimenzija slike primenom bilinearne transformacije, vrši se na sledeći način:

Neka je $g(x', y')$ intenzitet piksela u skaliranoj slici na poziciji (x', y'). Koordinate u ulaznoj slici koje odgovaraju ovom pikselu izlazne slike su:

$$x = \frac{x'}{s} \text{ i } y = \frac{y'}{s}$$

Kako x i y ne moraju biti celi brojevi, intezitet piksela $(g(x', y'))$ dobija se posmatranjem intenziteta 4 najbliža piksela u originalnoj slici: $f(x_0, y_0)$, $f(x_0, y_0 + 1)$, $f(x_0 + 1, y_0)$ i $f(x_0 + 1, y_0 + 1)$, pri čemu je:

$$x_0 = \left\lfloor \frac{x'}{s} \right\rfloor \text{ i } x_r = x' - x_0\, s$$

$$y_0 = \left\lfloor \frac{y'}{s} \right\rfloor \text{ i } y_r = y' - y_0\, s$$

Sada se intenzitet piksela u izlaznoj slici računa upotrebom sledećih relacija:

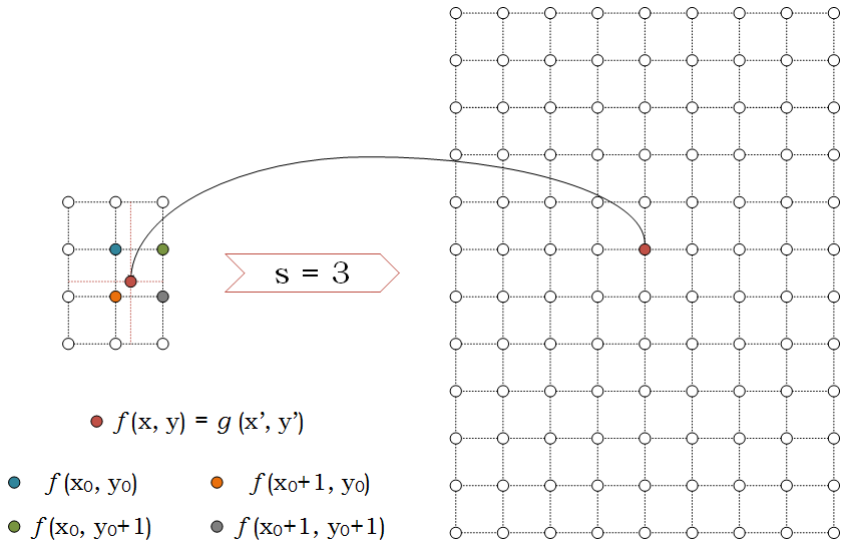
$$f(x_0, y) = \frac{f(x_0, y_0)\,(s - y_r) + f(x_0, y_0 + 1)\, y_r}{s}$$

$$f(x_0 + 1, y) = \frac{f(x_0 + 1, y_0)\,(s - y_r) + f(x_0 + 1, y_0 + 1)\, y_r}{s}$$

pa je:

$$g(x', y') = f(x, y) = \frac{f(x_0, y)\,(s - x_r) + f(x_0 + 1, y)\, x_r}{s}$$

Primer: Na slici 1 prikazan je jedan primer povećanja dimenzija slike u kome je faktor skaliranja $s = 3$. Označeni piksel u izlaznoj slici ima koordinate $(x', y') = (5, 4)$, pa su ekvivalentne koordinate ovog piksela u ulaznoj slici $(x, y) = (1.67, 1.33)$, odnosno $(x_0, y_0) = (1, 1)$ i $(x_r, y_r) = (2, 1)$. Neka je:



Slika 1: Primer skaliranja slike sa faktorom s=3

$$f(x_0, y_0) = 128,$$

$$f(x_0, y_0 + 1) = 150,$$

$$f(x_0 + 1, y_0) = 132 \text{ i}$$

$$f(x_0 + 1, y_0 + 1) = 145$$

Odavde je:

$$f(x_0, y) = \frac{128 \cdot (3-1) + 150 \cdot 1}{3} = \lfloor 135.33 \rfloor = 135,$$

$$f(x_0 + 1, y) = \frac{132 \cdot (3-1) + 145 \cdot 1}{3} = \lfloor 136.33 \rfloor = 136,$$

iz čega sledi da je:

$$g(x', y') = f(x, y) = \frac{135 \cdot (3-2) + 136 \cdot 2}{3} = \lfloor 135.667 \rfloor = 135$$

U slučaju decimacije slike dimenzije slike se određuju kao: $\left\lfloor \frac{N}{s} \right\rfloor$ i $\left\lfloor \frac{M}{s} \right\rfloor$ gde su M i N dimenzizije slike, a intenziteti piksela u izlaznoj slici određuju se prema formuli:

$$g(x', y') = f(sx', sy')$$

. . . . . .

Projekat 2 - skaliranje slike primenom metoda najbližeg suseda

Napisati program za skaliranje slike primenom metode najbližeg suseda. Faktor s, sa kojim se vrši skaliranje, unosi se sa standardnog ulaza i uvek je veći ili jednak 1. Pored toga, sa standardnog ulaza na određeni način definiše se da li se vrši povećanje ili decimacija slike s puta.

. . . . . .

Povećanje dimenzija slike primenom metoda najbližeg suseda, vrši se na sledeći način:

Neka je $g(x', y')$ intenzitet piksela u skaliranoj slici na poziciji (x', y'). Koordinate u ulaznoj slici koje odgovaraju ovom pikselu izlazne slike su:

$$x = \frac{x'}{s} \text{ i } y = \frac{y'}{s}$$

Kako x i y ne moraju biti celi brojevi, intezitet piksela ($g(x', y')$) dobija se posmatranjem 4 najbliža piksela u originalnoj slici: $f(x_0, y_0)$, $f(x_0, y_0 + 1)$, $f(x_0 + 1, y_0)$ i $f(x_0 + 1, y_0 + 1)$, pri čemu je:

$$x_0 = \left\lfloor \frac{x'}{s} \right\rfloor \text{ i } x_r = x' - x_0\, s$$

$$y_0 = \left\lfloor \frac{y'}{s} \right\rfloor \text{ i } y_r = y' - y_0\, s$$

Sada se intenzitet piksela u izlaznoj slici računa tako što se:

- ukoliko je $x_r < s - x_r$, uzima piksel u ulaznoj slici čija je x koordinata x_0, dok se u suprotnom uzima piksel čija je x koordinata $x_0 + 1$
- ukoliko je $y_r < s - y_r$, uzima piksel u ulaznoj slici čija je y koordinata y_0, dok se u suprotnom uzima piksel čija je y koordinata $y_0 + 1$

Primer: Na slici 1 prikazan je jedan primer povećanja dimenzija slike u kome je faktor skaliranja $s = 3$. Označeni piksel u izlaznoj slici ima koordinate $(x', y') = (5, 4)$, pa su ekvivalentne koordinate ovog piksela u ulaznoj slici $(x, y) = (1.67, 1.33)$, odnosno $(x_0, y_0) = (1, 1)$ i $(x_r, y_r) = (2, 1)$. Neka je:

$f(x_0, y_0) = 128$,

$f(x_0, y_0 + 1) = 150$,

$f(x_0 + 1, y_0) = 132$ i

$f(x_0 + 1, y_0 + 1) = 145$

Kako je $x_r > s - x_r$ i $y_r < s - y_r$, sledi da je:

$$g(x', y') = f(x, y) = f(x_0 + 1, y_0) = 132$$

U slučaju decimacije slike dimenzije slike se određuju kao: $\left\lfloor \frac{N}{s} \right\rfloor$ i $\left\lfloor \frac{M}{s} \right\rfloor$ gde su M i N dimenzije slike, a intenziteti piksela u izlaznoj slici određuju se prema formuli:

$$g(x', y') = f(sx', sy')$$

. . . . . .

Projekat 3 - (de)šifrovanje primenom Vigenèr-ovog/Beaufort-ovog algoritma

Napisati program koji učitava ulazni fajl koji u prvom redu sadrži

mode $\in \{e, d\}$ - karakter koji definiše da li tekst koji se nalazi u fajlu treba šifrovati (e) ili dešifrovati (d)

type $\in \{v, b\}$ - karakter koji definiše da li se (de)šifrovanje obavlja primenom Vigenèr-ovog (v) ili Beaufort-ovog (b) algoritma

key_type $\in \{s, a\}$ - karakter koji definiše da li (de)šifrovanje obavlja korišćenjem standardne (s) ili autokey metode (a)

koji su razdvojeni jednim znakom razmaka, u drugom redu nalazi se reč koja predstavlja ključ koji se koristi prilikom formiranja šifrovane poruke, a od trećeg reda na dalje nalazi se tekst koji je potrebno obraditi. Kao rezultat izvršavanja programa, potrebno je u izlazni fajl, prema formatu specificaranom za ulazni fajl upisati mode, type, key_type i ključ, kao i rezultat obrade ulaznog teksta (parametari type, key_type i ključ su isti kao i u ulaznom fajlu, parametar mode je suprotan od onog koji je definisan u ulaznom fajlu).

Na slici 2 prikazana je matrica koja se koristi prilikom upotrebe ovih algoritama.

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

Slika 2: Matrica koja se koristi prilikom šifrovanja

Naime, ova dva algoritma su jako slična, samo se razlikuju delovi matrice koji se koriste kao ključ:

Vigenèr-ovo algoritam matricu sa slike 2 interpretira na sledeći način:

Prva kolona predstavlja slova ključa, prvi red predstavlja slova poruke koja se šifruje, a u preseku reda koji odgovara slovu ključa i kolone koja odgovara slovu poruke nalazi se slovo šifrovane poruke. Ovo se takođe može predstaviti formulom:

$$C_i = (M_i + K_i) \bmod 26$$

Dešifrovanje poruke šifrovane ovim algoritmom može se predstaviti formulom:

$$M_i = (C_i - K_i) \bmod 26$$

Beaufort-ov algoritam matricu sa slike 2 interpretira na sledeći način:

Prvi red predstavlja slova poruke koja se šifruju, a unutrašnja matrica predstavlja slova ključa. Slovo šifrovane poruke dobija se tako što se u koloni koja odgovara slovu poruke nađe slovo ključa, i slovo iz prve kolone kome odgovara taj red predstavlja slovo šifrovane poruke. Ovo se takođe može predstaviti formulom:

$$C_i = (K_i - M_i) \bmod 26$$

Dešifrovanje poruke šifrovane ovim algoritmom može se predstaviti formulom:

$$M_i = (K_i - C_i) \bmod 26$$

gde je M_i slovo poruke koja se šifruje, K_i slovo ključa i C_i slovo dobijeno šifrovanjem.

U slučaju da je poruka koja se šifruje kraća od dužine ključa, koristi se samo potreban broj slova iz ključa, nezavisno da li se koristi standardna ili autokey metoda. Ukoliko je poruka koja se šifruje duža od ključa, način produžavana ključa razlikuje se u slučaju ove dve metode:

Standardna metoda zasniva se na periodičnom ponavljanju ključa potreban broj puta, tako da se pokriju sva slova poruke

Autokey metoda zasniva se na dodavanju slova iz poruke koja se šifruje na originalni ključ, dok se ne pokriju sva slova poruke

. . . . . .

Projekat 4∗ - Snake

Potrebno je realizovati igru zmijica (*Snake*).

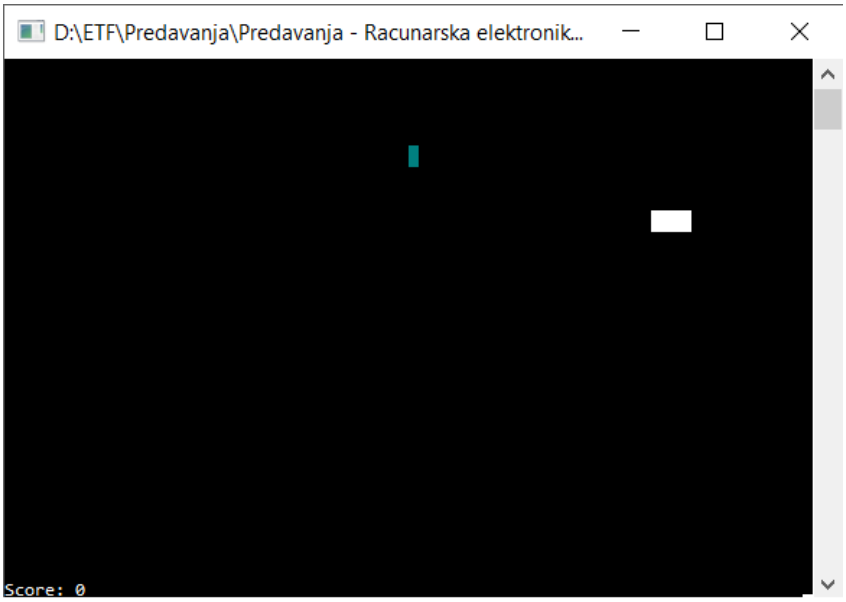
Koristeći kursor tastere na tastaturi pomerati zmijicu u četiri pravca. Blokove zmijice realizovati u vidu praznog prostora (*spacebar* – FFFFh), tako da je prazan prostor kojim se iscrtava zmijica obojen u boju drugačiju od pozadine. Hrana se pojavljuje na nasumično generisanim lokacijama (korišćenjem *random* funkcije) i takođe se realizuje korišćenjem praznog prostora obojenog u boju drugačiju od zmijice i pozadine.

Početna pozicija zmijice je na sredini igrališta i početna dužina zmijice je četiri polja. Igrica se napušta pritiskom na taster ESC, prilikom kolizije sa preprekom ili prilikom kolizije zmijice sa samom sobom. Nakon završetka igre, ispisuje se broj osvojenih poena.

Prilikom pokretanja programa, i nakon napuštanja igre, program treba da uđe u glavni meni, u kojem je moguće odabrati početak igre, napuštanje programa i brzinu kretanja zmijice.

Na slici 3 je prikazan primer izgleda prozora prilikom igranja igrice.

. . . . . .



Slika 3: Igrica Snake.

Projekat 5∗ - Pong

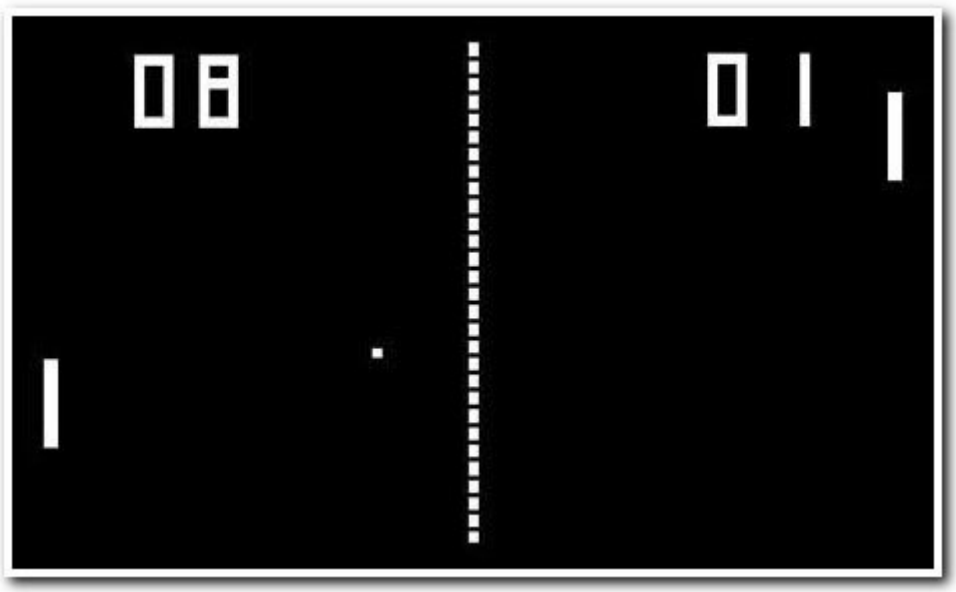
Napaviti igricu Pong za slučaj jednog i dva igrača (pogledati: `http://www.ponggame.org/`).

Igrači su predstavljeni pedalama veličine četiri prazna bloka. Kuglica je takođe oblika praznog bloka, i početna pozicija joj je na sredini ekrana. Svaki put kada kuglica udari u pedalu ili gornju ili donju ivicu prozora, ona se odbija pod istim uglom u odnosu na normalu pedale/ivice, samo u suprotom smeru. Takođe, svaki put kada udari pedalu, brzina kuglice se povećava. Ukoliko kuglica dotakne levu ili desnu ivicu prozora, onda se završava takuće „dodavanje" kuglice. Broj poena igrača koji je odigrao poslednji potez dodavanja se inkrementira za jedan i počinje nova partija.

Ukoliko se izabere igranje ponga u dva igrača, potrebno je aktivirati 4 kursora na tastaturi: po dva za pokretanje oba igrača. U slučaju da se izabere igranje sa jednim igračem, aktiviraju se dva tastera za pokretanje pedale. Drugog igrača je potrebno realizovati u programu tako da se kreće u smeru u kom se odbila kuglica od pedalu/ivicu prozora konstantnom brzinom.

Igrica se završava kada se odigra 10 partija ili pritiskom tastera ESC. Nakon završetka igrice se ispisuje pobednik sa osvojenim brojem bodova i ponovo se ulazi u glavni meni gde se može izabrati igranje u paru, sa jednim igračem ili izlazak iz igrice.

Na slici 4 je prikazan primer izgleda prozora prilikom igranja igrice.



Slika 4: Igrica Pong.

. . . . . .

Projekat 6∗ - Asteroids

Potrebno je realizovati igricu Asteroids (pogledati: `http://www.freeasteroids.org/`).

Na početku igrice na sredini ekrana se nalazi svemirski brod veličine 4 prazna bloka. Svake sekunde se pojavljuje po jedan asteroid sastavljen od 8 praznih blokova na proizvoljnim pozicijama. Svemirski brod pritiskom tastera SPACE ispaljuje mine oblika jednog praznog bloka. Mina se kreće unapred zadatom brzinom, a ako se i svemirski brod kreće onda je brzina mine jednaka:

$$v_M = v_{M,const} + 3\, v_B,$$

gde je v_M brzina mine, $v_{M,const}$ unapred zadata konstantna brzina mine i v_B brzina broda.

Brod može da se drugačije orijentiše pritiskom tastera levo i desno na tastaturi gde se svakim pristkom obće u smeru kazaljke na satu (odnosno obrnutom smeru od kazaljke na satu) za približno 10°. Pritiskom kursora na gore se brod ubrzava tako što se doda inkrement na njegovu brzinu, a taster na dole ga usporava. U toku kretanja u svemiru postoji i trenje koje se ogleda u tome što se na svakih nekoliko polja smanjuje brzina broda koji staje ukoliko nije pritisnut taster na gore (pogledati video: `Asteroids.mp4`).

Kada mina udari u asteroid, asteroid biva raznet i nestaje, a broj bodova osvojen u igrici se uvećava za 100. Ukoliko svemirski brod udari u asteroid ili se novi asteroid pojavi na svemirskom brodu, onda se igrica završava i ispisuje se ukupan broj osvojenih bodova. Igrica se završava i pritiskom tastera ESC.

. . . . . .

Projekat 7∗ - Bejeweled

Napraviti igricu Bejeweled (`http://www.classicgamesarcade.com/game/21730/bejeweled.html`). Prozor je podeljen na 8×8 polja u kojima su poređani kvadrati u 4 različite boje: crvene, žute, plave i zelene. Na početku igrice se popunjavaju polja kvadratima nasumičnih boja. Kursorima na tastaturi se pomera kroz kvadrate, a tasterom SPACE se markiraju blokovi koji bi trebalo da zamene mesta. Mesta mogu da zamene samo susedni blokovi u vrsti ili koloni i to

samo ukoliko s zamenom dobijaju u vrsti ili koloni bar tri kvadrata iste boje. Program treba da registruje i grupe veća od tri polja iste boje.

Svaki put kada se složi drupa od više od tri polja, data polja se brišu. Polja koja su iznad izbrisanih polja se pomeraju ka dole, a dobijena prazna polja na vrhu se popunjavaju poljima nasumične boje (crvene, žute, plave ili zelene).

Igrica traje 3 minuta. Svaki put kada se složi više od tri polja iste boje, broj osvojenih poena se povećava za $100\,n$, gde je n broj polja. Na kraju izvršavanja igrice se ispisuje broj osvojenih poena. Igrica se može prekinuti pritiskom tastera ESC.

Slika 5: Igrica Bejeweled.

. . . . . .

Projekat 8∗ - Minesweeper

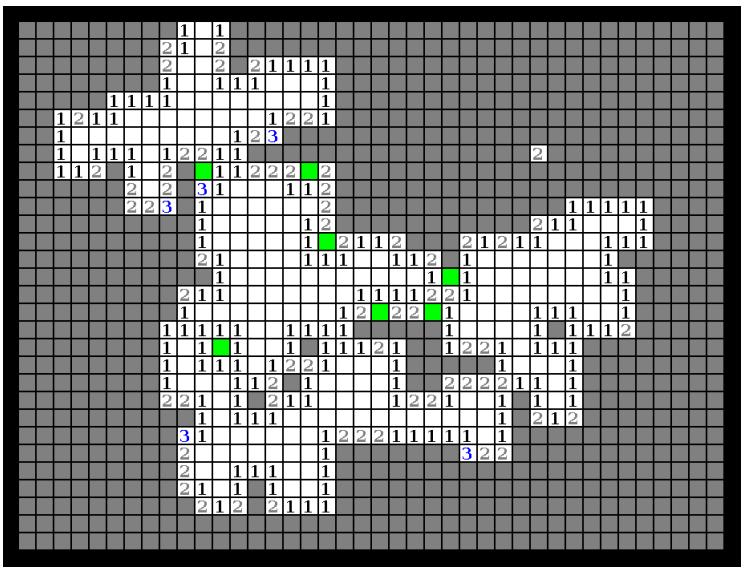
Napraviti igricu Minesweeper:
`http://www.classicgamesarcade.com/game/21649/minesweeper.html`,
`https://en.wikipedia.org/wiki/Minesweeper_(video_game)`.

Prozor igrice (tabla) je podeljen na 9×9 polja koja su na početku igrice sva iste boje. Na 10 lokacija na tabli, odnosno iza 10 polja se nalaze mine koje su prikrivene. Kroz polja se kreće kursorima na tastaturi (na neki način označiti polje na kome se trenutno nalazi kursor). Polja napraviti veličine 2×2.

Pritiskom tastera SPACE se otvara polje. Ukoliko se na polju ne nalazi mina, rekurzivno se otvaraju i sva susedna polja (promeniti boju ovih polja) na kojima nema mine. Na ivicama otvorene oblasti bez mina se ispisuje broj koliko se mina nalazi pored datog polja u redu, koloni i djagonalno. Pritiskom na minu se završava igrica. Ako se pretpostavlja da se iza datog polja nalazi mina, onda se to polje označava pritiskom tastera Left Shift i tada dato polje postaje druge boje.

Igrica se završava kada se otvore sva polja na kojima nema mine i označe sva polja sa minama. Ukoliko se otvori polje na kome se nalazi mina, takođe se završava igrica. Pritiskom tastera ESC omogućiti prekid programa.

. . . . . .



Slika 6: Igrica Minesweeper.

Projekat 9∗ - 2048

Napraviti igricu 2048 (`https://en.wikipedia.org/wiki/2048_(video_game)`).

Igrica se igra na 4×4 mreži kao na slici 7. Na početku igrice na mreži se popunjava nasumično jedno polje kvadratom na kome je ispisan broj koji predstavlja stepen dvojke. Svi stepeni dvojke imaju odgovarajuću boju polja. Pritiskom kursora na tastaturi se popunjena polja pomeraju skroz levo, desno, gore ili dole.

Ukoliko se pomeranjem dodirnu dva polja sa istim stepenom dvojke, ona se zamenjuju poljem čija je vrednost jednaka zbiru ova dva polja, a ukupan broj poena se inkrementira za tu istu vrednost. U slučaju ako se desi da postoje 3 susedna polja koja imaju istu vrednost, onda se dva koja su bliža onoj granici mreže na koju ukazuje pritisnuti kursor zamenjuju jednim poljem sa duplo većom vrednošću.

Svaki put kada se pritisne neki od kursora na tastaturi, nakon pomeranja svih polja u smeru tog kursora, na neko od praznih polja se nasumično generiše novi element. Vrednost na polju se generiše nasumično i to 2 sa verovatnoćom 5/8, 4 sa verovatnoćom 2/8 i 8 sa verovatnoćom 1/8.

Igrica se završava pritiskom tastera ESC ili kada na mreži ne postoji više nijedno prazno polje, a susedni elemenati ne mogu pomeranjem da daju jedno polje.

Slika 7: Igrica 2048.

Projekat 10 - Školski dnevnik*

Potrebno je napraviti program koji će pomoći profesorima da srede dnevnik na kraju jednog polugođa. Program radi sa datotekama i sa podacima koje profesor u program unosi preko tastature. Program treba da izračuna prosečne ocene za svaki predmet i za svakog učenika ponaosob.

Slika 10.1 Logička podela interfejsa programa

Na početku program otvara konzolu proizvoljne veličine. Struktura sadržaja konzole treba da bude kao na slici 1. Sadržaj konzole se može podeliti u dva logička dela uokvirena crvenom bojom. Jedan deo predstavlja *PREGLED* unetih podataka dok drugi deo predstavlja deo konzole koji služi za *UNOS PODATAKA*. Nakon otvaranja konzole tabela se formira na osnovu sadržaja tekstualne datoteke *odeljenjeinfo.txt* koja sadrži :

- Ime i prezime razrednog starešine
- Odeljenje
- Školsku godinu
- Skraćenice predmeta koje učenici slušaju , kao i binarni podatak da li se predmet , čija se skraćenica nalazi u fajlu, ulazi u prosek (1 – ulazi u prosek , 0 – ne ulazi u prosek)

Struktura tekstualnog fajla *odeljenjeinfo.txt* prikazana je na slici 2. Prilikom čitanja sadržaja

```
Ime Prezime ; // Ime i prezime odeljenskog starešine
Odeljenje; // Odeljenje (npr. 4-3, 4-2, 3-2)
Sk.Godina; // 2017/18
Predmet1 1 , Predmet2 1, Predmet3 0,
Predmet4 1, ... , Predmet N 0; // Skraćenice predmeta sa informacijom da li dati
predmet ulazi u ------//prosek (1) ili ne (0).
```

iz ovog fajla i prepisivanja u konzolu, deo teksta iza znaka // predstavlja komentar koji treba ignorisati.

Nakon što se formira tabela , od korisnika programa se u delu programa označenim sa UNOS PODATAKA traži da unese ime fajla u kome se nalaze ocene učenika. Nakon unetog imena fajla , vrši se prepisivanje ocena u tabelu. Struktura fajla u kome se nalaze ocene učenika data je na slici

1. Marko Markovic , 5 4 4 ... 3 2 5;
2. Ivana Ivanović , 5 5 4 5 ... 4 3 5 5 ;

Predpostavka je da su ocene predmeta navedene u poretku u kom su navedeni i predmeti u fajlu *odeljenjeinfo.txt.* Nakon unetih ocena potrebno je izračunati tražene proseke i ažurirati tabelu. Nakon izračunavanja proseka program pita korisnika da li želi da sačuva sadržaj tabele. Ukoliko je odgovor potvrdan od korisnika se traži da unese ime fajla u koji želi da sačuva podatke a zatim se sadržaj tabele prepisuje u tekstualni fajl.

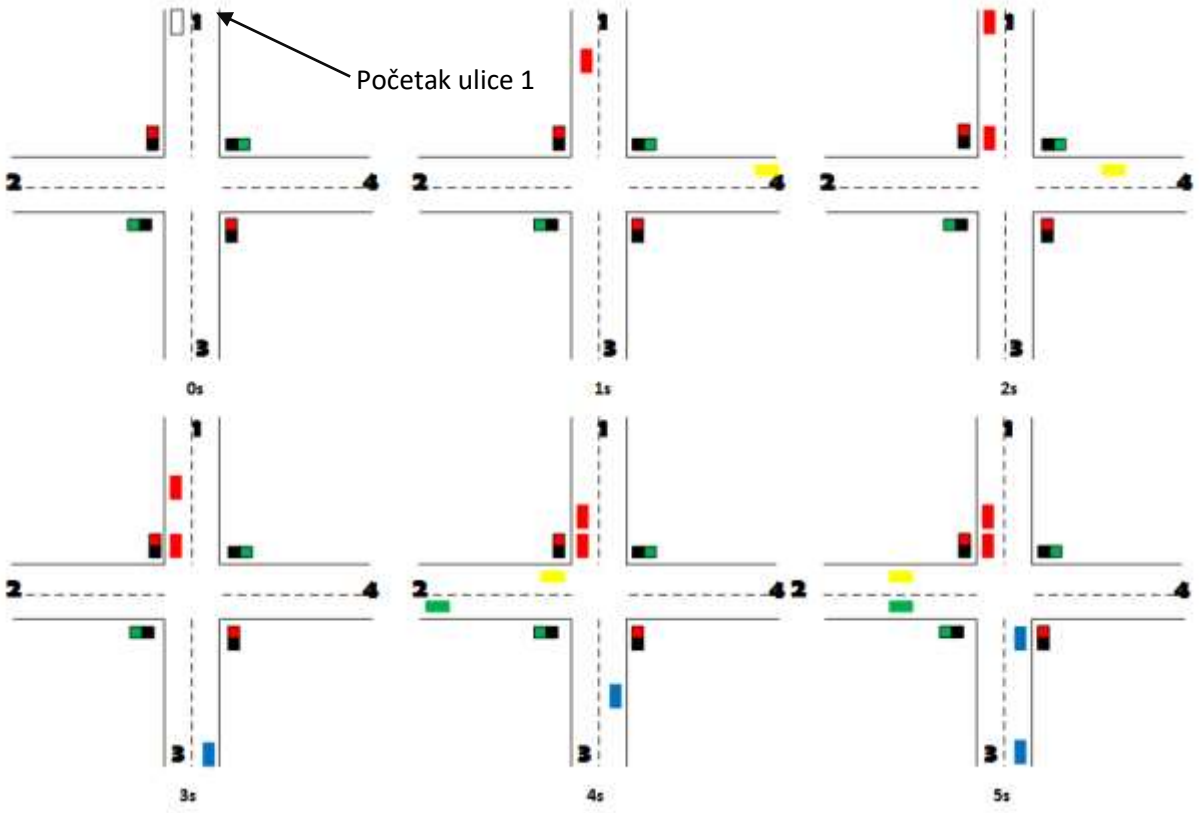
Ispravnost programa potvrditi sa dva različita fajla *odeljenjeinfo.txt*. Napisani kod treba da bude pregledan i propraćen komentarima.

U ovom projektu je neophodno koristiti *floating-point* jedinicu da bi se moglo realizovati necelobrojno deljenje brojeva. Uputstvo za rad sa ovom jedinicom može se naći u knjizi **KIP R. IRVINE** *Assembly Language for x86 Processors* poglavlje 12.

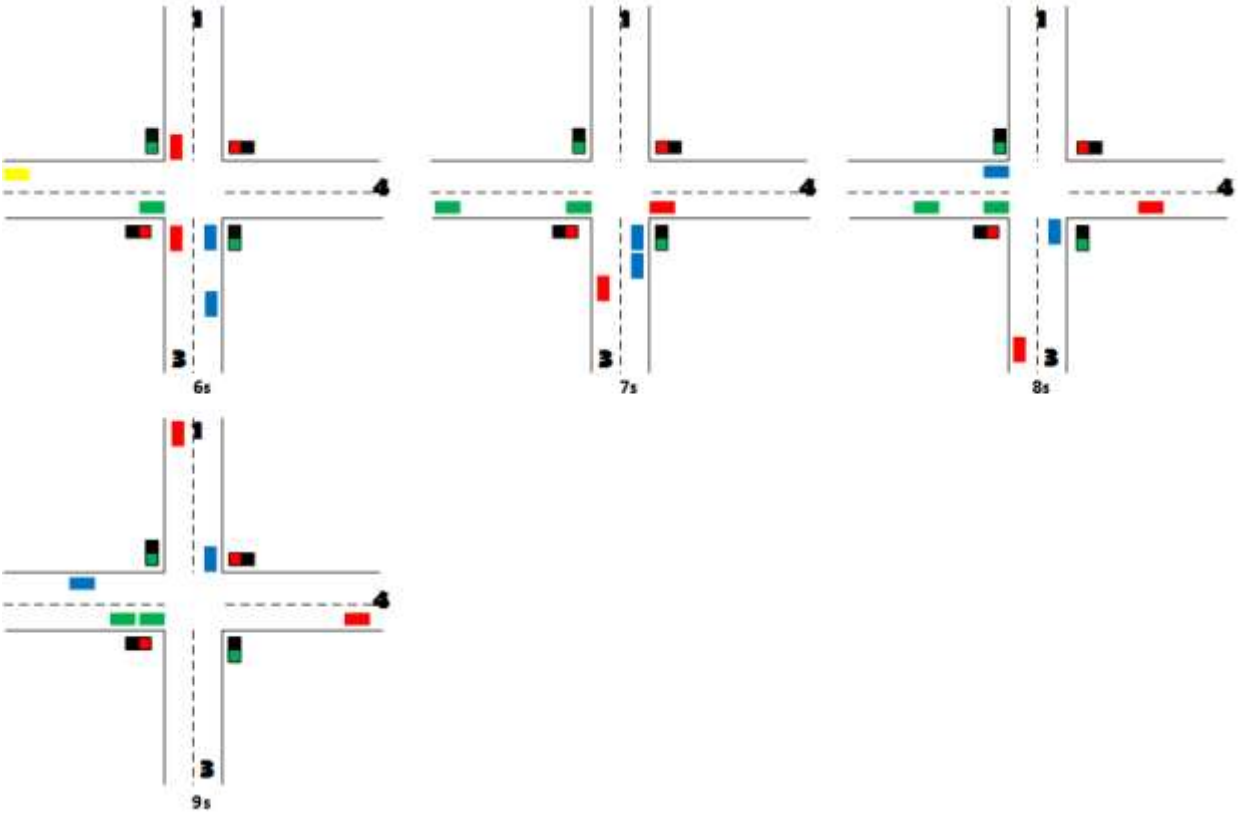
Projekat 11 - Simulacija raskrsnice *

Na raskrsnici se ukrštaju 4 ulice (označene brojevima 1-4). U svakoj ulici se proizvoljno sa periodom od 1s generiše po jedan automobil. Automobil se uvek generiše na početku jedne od 4 ulice (Početak ulice – deo ulice koji se nalazi na krajevima konzole). U jednoj ulici se maksimalno mogu nalaziti dva automobila dok simulator može maksimalno prikazivati šest automobila na sve četiri ulice. Svakom od automobila treba 2s da stigne od mesta na kome je generisan do semafora raskrsnice. Semafori menjaju stanje sa periodom od 6s i rade u parovima 1-3, 2-4.

Potrebno je napisati program koji simulira kretanje automobila na raskrsnici. Ukoliko je na semaforu crveno svetlo za automobile koji dolaze iz neke od 4 ulice, ti automobili čekaju sve dok se ne uključi zeleno svetlo. Ukoliko je na semaforu zeleno svetlo automobili ulaze na raskrsnicu pod uslovom da neki od automobila iz suprotne ulice , u kojoj je takođe zeleno svetlo, ne želi da preseče put nekom od automobila. U ovoj situaciji se poštuje pravilo desne strane koje prednost daje onom automobilu koji dolazi sa vozačeve desne strane. Princip generisanja automobila , kretanja automobila i funkcionisanja saobraćaja dat je na slikama 1 i 2.



Slika 11.1 a)

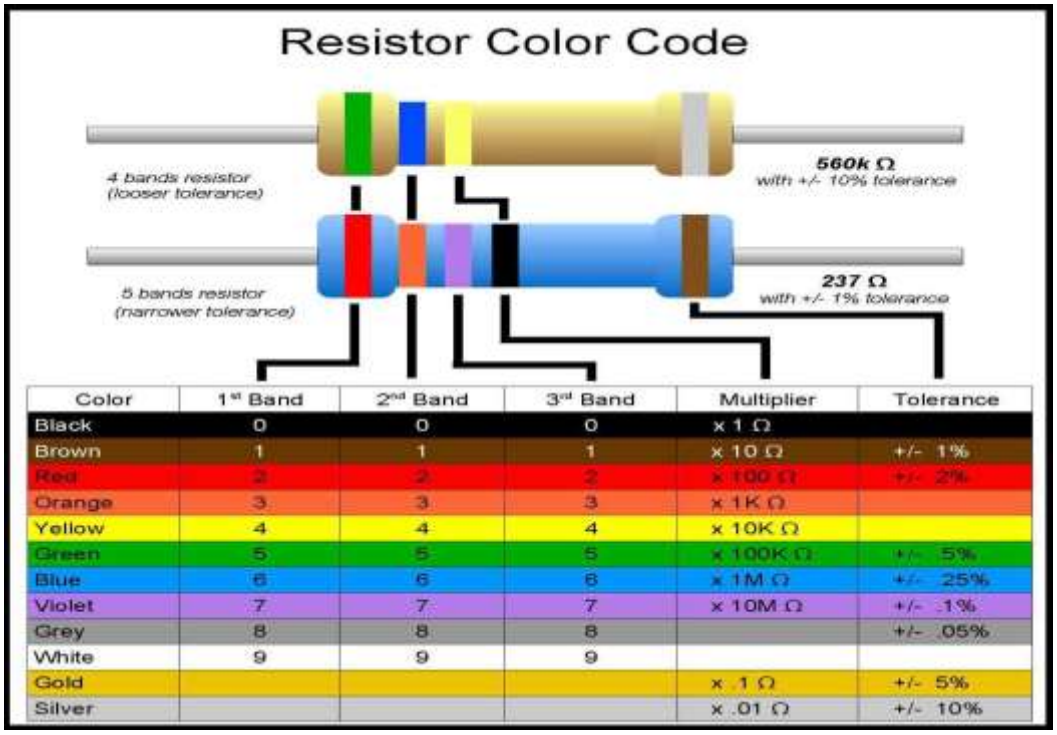


Slika 11.1 b)

Simulaciju je u svakom trenutku moguće zaustaviti pritiskom na taster SPACE. Ponovnim pritiskom na taster SPACE simulacija nastavlja sa izvršavanjem.

Projekat 12 - Pronađi otpornik

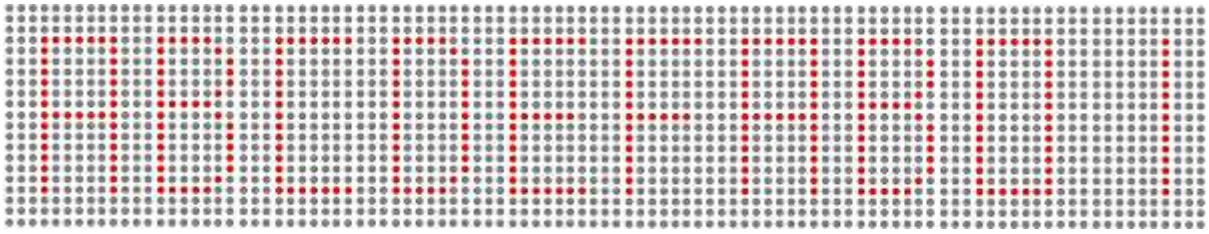
Potrebo je realizovati program koji će studentima u lab. 39 ☺ pomoći da na laboratorijskim vežbama brže nađu vrednost otpornika. Student unosi celobrojnu vrednost koja predstavlja željenu vrednost otpornika a program treba u konzoli da iscrta četiri pravougaonika čije boje predstavljaju tu



Slika 12.1 Tabela otpornosti

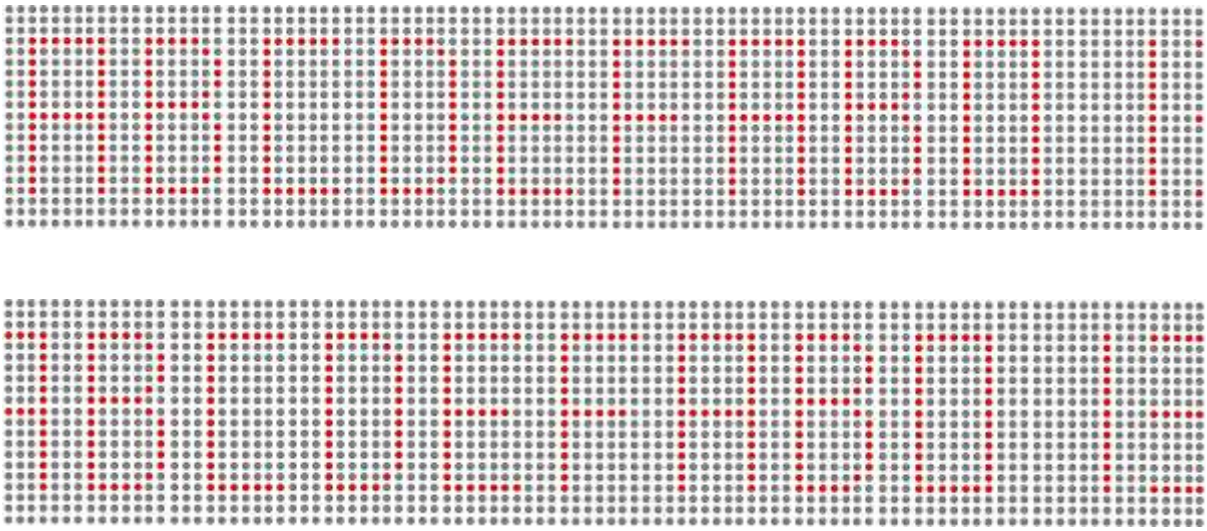
vrednost (pravougaonik koji odgovara toleranciji se izostavlja). Program treba pretragom tabele otpornosti, koja je data na slici 1., da nađe boje koje odgovaraju unetoj vrednosti. Ukoliko korisnik unese otpornost koja ne postoji u tabeli neophodno je obavestiti korisnika o tome a zatim naći vrednost u tabeli koja je najbliža unetoj vrednosti i za nju iscrtati pravougaonike u odgovarajućim bojama.

Projekat 13 - Simulacija Led reklame*



Slika 13.1 Izgled LED displeja

Potrebno je napraviti program koji će u konzoli simulirati rad led displeja. Led displej je širine 100x21 piksela. Jedna dioda na led displeju predstavlja piksel. Jedno slovo na displeju se predstavlja grupom isključenih ili uključenih led dioda koje formiraju izgled nekog slova. Svako slovo zauzima površinu 15x7 piksela. Tekst koji možemo ispisati je maksimalne dužine 50 karaktera. Razmak između karaktera je 3 px. Izgled dela led displeja prikazan je na slici 1. Ukoliko je dužina unetog teksta veća od 10 karaktera, koliko je moguće prikazati odjednom na displeju , potrebno je rotirati tekst na displeju tako da se omogući prikaz svih unetih ASCII karaktera. Ovaj mehanizam prikazan je na slici 13.2.



Slika 13.2 Simulacija kružnog pomeranja slova

U konzoli je neophodno simulirati gore navedene osobine led displeja. Na početku programa se od korisnika traži da unese sekvencu ASCII karaktera . Zatim se uneta sekvenca ispisuje na ekranu na prethodno opisani način. Brzina rotacije se povecava pritiskom na taster UP(strelica na gore) a smanjuje pritiskom na taster DOWN (srelica na dole).

☆ Prilikom realizacije ovog projekta preporučeno je koristiti neke od funkcija za iscrtavanje u konzoli čiji je opis dat na linku https://docs.microsoft.com/en-us/windows/console/console-functions sa akcentom na procedure WriteConsoleOutputA, WriteConsoleOutput, ... U knjizi **KIP R. IRVINE** *Assembly Language for x86 Processors* obratiti pažnju na sekciju 11 koja pomaže u razumevanju rada funkcija sa gore navedenog linka.

Projekat 14 - RE syntax validator*

Kao i svaki programski jezik , i programski jezik RE ima svoju sintaksu. Da bi olakšali programerima koji pišu programe na ovom jeziku , potrebno je napraviti program koji će u konzoli ispisivati greške koje su programeri napravili prilikom kucanja koda. U slučaju sintaksno ispravnog koda program generiše dva tekstualna fajla:

- *function.txt*
- *variable.txt*

Struktura ovih fajlova biće navedena u nastavku projektnog zadatka. Programski jezik je *case sensitive* što znači da razlikuje velika i mala slova.

Cilj ovog programa jeste da upozori programera o postojanju greške u njegovom kodu, pa stoga ispisane greške treba da što tačnije daju programeru uvid u tip greške koju je napravio i da mu omoguće lako lociranje greške u kodu. Struktura poruka greške je sledeća:

ERROR ID | ERROR POSITION | ERROR MESSAGE

ERROR ID – jedinstveni identifikator greške
ERROR POSITION – broj linije u kojoj se nalazi kog gde je detektovana greška
ERROR MESSAGE - poruka o grešci koja daje detaljniji uvid u tip greške

Primer ispisa jedne greške usled neispravnog koda dat je u primeru:

2.1.1 | Linija 5 | Naziv promenljive nije ispravan. Prvi karakter u imenu promenljive mora biti slovo.

Kod programa se piše u tekstualnom fajlu *code.txt*.

1. Struktura programskog jezika RE

Kao i većina programskih jezika i ovaj programski jezik omogućava apstrahovanje određenih delova koda u zasebne logičke celine koje se zovu funkcije. I sam program pisan u

```
func func_name (type1 var1, type2 var2, ... )
      // deklaracija promenljivih
      // telo funkcije
endf func_name;
```

ovom programskom jeziku počinje od funkcije START. Sintaksa funkcija je:

func_name – ime funkcije koje mora počinjati slovom
type1,type2 – tipovi promenljivih var1 i var2. Tipovi mogu biti neki od podržanih tipova promenljivih o čemu će biti reči u nastavku.

Neke od grešaka koje mogu nastati prilikom pisanja funkcija su date u tabeli 1.1

ERROR CODE	ERROR MESSAGE
1.1	Naziv funkcije mora počinjati slovom.
1.2	Niste ispravno deklarisali argumente funkcije. Struktura liste argumenata je: var_type *var_name*
1.3	Argumenti se moraju odvojiti zarezom
. . .	

U slučaju ispravno napisane funkcije, u datoteku *funkcije.txt* se dodaje funkcija koja je sintaksno ispravna. Jedna linija fajla *funkcije.txt* ima sledeću strukturu:

ID | FUNC NAME | FUNC ADDR | NO OF ARG| TYPE1 | ARG1 | TYPE2 | ARG2 | …

ID – Jedinstven ID funkcije. Funkcija START uvek ima ID 0 dok se ostalim funkcijama dodeljuje jedinstven ID

FUNC NAME - Naziv funkcije
FUNC ADDR - Ovo polje ostaje prazno
NO OF ARG - Broj argumenata funkcije
TYPEi - Tip i-tog argumenta
ARGi - Naziv i-tog argumenta

Prilikom definisanja funkcija , neophodno je prvo deklarisati promenljive (da bi se formirala tabela promenljvih) a zatim vršiti neku obradu nad deklarisanim podacima i argumentima u telu funkcije.

2. Tipovi podataka

2.1 Osnovni tipovi

RE programski jezik podržava sledeće tipove podataka:

```
[0]uVar1 – neoznačeni celobrojni podatak dužine 1 bajt
[1]uVar2 - neoznačeni celobrojni podatak dužine 2 bajta
[2]uVar4 - neoznačeni celobrojni podatak dužine 4 bajta
[3]Var1  - označeni celobrojni podatak dužine 1 bajt
[4]Var2  - označeni celobrojni podatak dužine 2 bajta
[5]Var4  - označeni celobrojni podatak dužine 4 bajta
[6]rVar  - realni brojevi
```

Sintaksa deklaracije promenljivih je:

```
type [const] var_name [=var_value] ;
```

Promenljiva može biti deklarisana kao konstanta. Rezervisana reč *const* onemogućava dalju promenu vrednosti promenljive kroz kod. Promenljivoj se može dodeliti vrednost prilikom deklaracije. Ukoliko se ne izvrši inicijalizacija promenljive , promenljivoj se dodeljuje neka slučajna vrednost. U slučaju da se promenljiva proglasi konstantom , inicijalizacija je neophodna.

2.2 Izvedeni tipovi

Jedan od izvedenih tipova podataka je *niz*. Deklaracija promenljive tipa niz ima sledeću sintaksu:

```
[7] arr(num_of_elem) type [const] arr_name [=elem_values];
```

arr_name predstavlja adresu prvog elementa niza tipa *type* gde niz ima *num_of_elem* elemenata. Opciono svi elementi niza mogu biti konstante ili mogu biti inicijalizovane na određene vrednosti. Sve ostalo je isto kao i za osnovne tipove.

Nazivi promenljivih bilo kog tipa moraju počinjati slovom. Vrednosti promenljivih mogu biti:

1) decimalne (10,13, 0 , 100)
2) binare (1011101b)
3) heksadecimalne (0xA300013C)

Neke od grešaka koje treba detektovati:

ERROR CODE	ERROR MESSAGE
2.1.1	Naziv promenljie mora počinjati slovom
2.1.2	Konstante moraju biti inicijalizovane
2.1.3	arr tip podataka zahteva broj elemenata niza
. . .	

Neki od primera deklarisanja promenljvih i grešaka koje mogu nastati usled neispravnog deklarisanja su sledeći:

```
//Primer 1
1  uVar1 a = 4 ;
2  uVar4 const b; ! 2.1.2 |Line 2| Konstante moraju biti inicijalizovane

//Primer 2
1  uVar4 const a
= 4 ;
2  uVar1 b =10101111b;
3  Var2  c;
4  uVar4 Pointer = 0x0000AB13;
5  uvar4 12ptr2  = ox0000AB1E; 2.1.1 | Line 5 | Naziv promenljive mora
počinjati slovom
```

3. Telo funkcije

Ovaj deo funkcije predstavlja deo gde se vrši obrada podataka. Obrada podataka se vrši direktno (preko *operatora*, *while* i *for* petlji) ili pozivom nekih drugih funkcija.

ID | FUNC ID | VAR_NAME | TYPE ID | NO_OF_ELEM | VAR(ELEM) VALUE(S)

3.1.Operatori

Postoje unarni i binarni operatori. Operatori zajedno sa operandima čine *izraze* koji se mogu dodeliti nekoj promenljivoj. Podržani operatori su:

1) *Operator dodele vrednosti* (=)

   Levi operand je promenljiva kojoj dodeljujemo vrednost. Desni operand može biti ili promenljiva ili neki složeniji aritmetički/logički izraz

2) *Dereferenciranje* (#)

Unarni prefiksni operator koji iza sebe očekuje promenljivu tipa uVar4. Dereferencirana promenljiva se može nalaziti sa leve strane znaka = što znači da je sledeća naredba validna :

```
uvar4 ptrNoName=0x001133AC;

#ptrNoName = 3h;
```

3) *Operatori poređenja* (>, < , >= , <=)
4) *Aritmetički operatori* (+, -, *, /)

3.2 While petlja

```
while statemen

      [code]
endw;
```

While petlje u ovom programskom jeziku imaju sledeću sintaksu:

Predvideti mogućnost ulančavanja petlji kao korektnu situaciju

3.3 Uslovni skok

Ovaj programski jezik podržava uslovni skok. Skok se vrši ukoliko je ispunjen uslov

```
if statemen

      [code]
[
else

      [code]

]

endif;
```

dat izrazom *statement* . Sintaksa ove naredbe je:

3.4 Pozivi funkcija

U telu funkcije je moguće pozvati neku prethodno definisanu funkciju i proslediti joj argumente. Tip prosleđenih argumenata mora da se slaže sa tipom argumenata navedenih u definiciji funkcije ili se u suprotnom prijavljuje greška

3.5 Komentari.

Bilo gde u kodu je moguće napisati komentare što povećava čitljivost koda. Komentari započinju sa // a preostali deo reda u tekstualnom fajlu se ignoriše.

4. Primer

U nastavku je prikazan sadržaj tekstualnog fajla code.txt kao i sadržaj datoteka function.txt i variable.txt nakon provere koda bez sintaksnih grešaka.

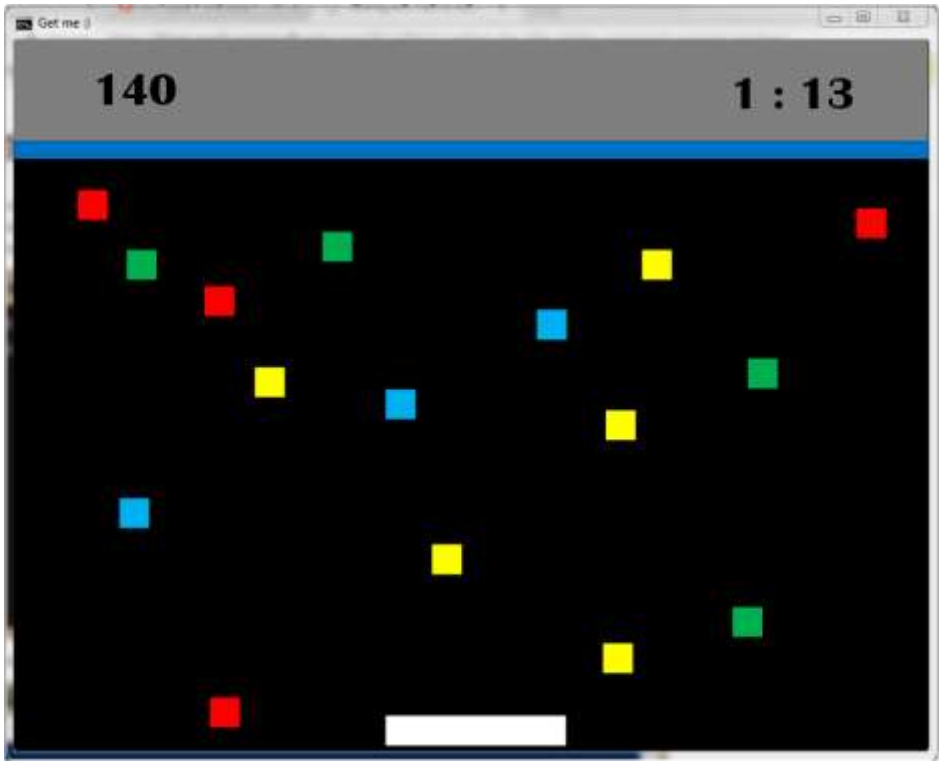
```
//Primer 2
1 func find_max(uVar4 arr, uVar1 arr_length, uVar4 max_value)
2     uVar4 tempmax = 0;
3     uVar4 temp_arr_ptr;
4     tempmax = #arr;
5     temp_arr_ptr = arr;
6     temp_arr_ptr = temp_arr_ptr + 1;
7     while arr_length
8           if(temp_max > # temp_arr_ptr)
9                 temp_max > #temp_arr_ptr;
10          endif;
11          arr_length = arr_length - 1;
12          temp_arr_ptr = temp_arr_ptr +1;
13    endw;
14    #max_value = temp_max;
15 endf;
16
17
18 func START
19    arr(10) uVar4 my_arr = 0;
20    uVar4 max = 0;
21    uVar4 arr_length = 10;
22    uVar4 max_ptr;
23    uVar4 temp_var = 0;
24    uVar4 temp_ptr;
25    temp_var = arr_length;
26    temp_ptr = my_arr;
27    while temp_var
28          #temp_ptr = temp_var;
29          temp_var = temp_var - 1;
30          temp_ptr = temp_ptr + 1;
31    endw;
32    call find_max (my_arr,
arr_length, max_ptr);
33    max = #max_ptr;
34
35 endf;
```

```
//variable.txt
0 | 1 | arr | 2 | | A // A – Argument funkcije
1 | 1 | arr_length | 2 | | A
2 | 1 | max_value | 2 | | A
3 | 1 | temp_max | 2 | | 0
4 | 1 | temp_arr_ptr  | 2 | | 4 // Posto nije inicijalizovana dodeljena joj je slucajna vrednost
5 | 0 | my_arr | 7  | 10 |  0 , 0, 0, 0 ,0 ,0 , 0 , 0 ,0 ,0
6 | 0 | max | 2 | | 0
7 | 0 | arr_length | 2 | | 10
8 | 0 | max_ptr | 2 | | 333
9 | 0 | temp_var | 2 | | 0
10 | 0 | temp_ptr | 2 | | 111

//function.txt
0 | START | | 0 |
0 | find_max || 3| 2 | arr | 0 | arr_length | 2 | max_value
```

Napravljeni program u asembleru testirati na bar još dva programa napisana na RE programskom jeziku koji potvrđuju uspešnost realizovanog programa!

Projekat 15 - Take box *



Slika 15.1 Grafički interfejs igrice

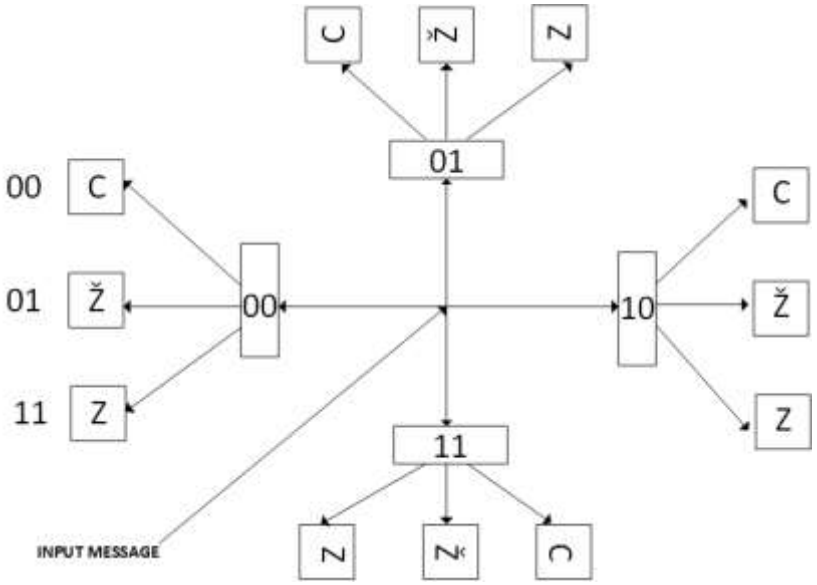
Na slici 15.1 je prikazan interfejs igrice. Igrač za fiksno vreme treba da sakupi što više poena(kvadratića) a da izbegava crvene kvadratiće. Ukoliko igrač uzastopno uhvati dva kvadratića iste boje, broj poena koji nosi ta boja se duplira (ako uzme dva kvadratića plave boje broj poena je 320). Ukoliko igrač PAD-om dohvati crveni kvadratić ili istekne vreme predviđeno za jednu partiju igrice , konzola se zatvara.

Na konzoli se u slučajnim vremenskim intervalima generišu kvadrati. Kvadrati se generišu na slučajnoj poziciji odmah ispod plave linije i počinju da se kreću ka donjem delu konzole. Ukoliko PAD (bele boje) uhvati neki od kvadratića zelene, žute ili plave boje dobija 100,40 ili 80 poena respektivno. U slučaju da uhvati crveni kvadratić igrica se prekida i izlazi se iz konzole. Kretanje pada se ostvaruje pomoću navigacionih tastera (strelica levo i desno) na tastaturi. Veličina PADa , kvadratića kao i ostalih delova interfejsa se mogu usvojiti po želji programera ali tako da se ne naruši funkcionalnost igrice.

☆ Prilikom realizacije ovog projekta preporučeno je koristiti neke od funkcija za iscrtavanje u konzoli čiji je opis dat na linku https://docs.microsoft.com/en-us/windows/console/console-functions sa akcentom na procedure WriteConsoleOutputA, WriteConsoleOutput, ... U knjizi **KIP R. IRVINE** *Assembly Language for x86 Processors* obratiti pažnju na sekciju 11 koja pomaže u razumevanju rada funkcija sa gore navedenog linka.

Projekat 16 - Network protocol

Na slici 16.1 je prikazana topologija jedne mreže. Cilj ovog projekta je simulacija



Slika 16.1 Topologija mreže

protoka poruka kroz mrežu i reagovanje uređaja (00 - 11) na odgovarajuće poruke. Preko mreže se može upravljati svim uređajima odjednom ili svakim uređajem ponaosob. Da bi se omogućilo upravljanje svakim uređajem ponaosob, svaki uređaj ima svoju jedinstvenu adresu (00,01,10,11).

Svaki uređaj preko mreže dobija odgovarajuće instrukcije koje treba da prepozna i da ih izvrši. Instrukcije koje se mogu poslati preko mreže su :

- Uključi diodu
- Isključi diodu
- IDLE

Instrukcija „Uključi diodu“ uključuje diodu odgovarajuće boje dok instrukcija „Isključi diodu“ isključuje diodu odgovarajuće boje. Koju diodu treba isključiti ili uključiti se takođe šalje preko mreže. Instrukcija „IDLE“ ne sprovodi

D7	D6	D5	D4	D3	D2	D1	D0
C	B	ADDR		FUNC		COLOR	

C = 1 – Sistemska poruka
C = 0 – Druge poruke
B = 1 – Broadcast poruka
B = 0 – Druge poruke
ADDR = 00 – Uređaj 0
ADDR = 01 – Uređaj 1
ADDR = 10 – Uređaj 2
ADDR = 11 – Uređaj 3
FUNC = 00 – NOPE instrukcija
FUNC = 01 – Ukljuci diodu zadate boje
FUNC = 10 – Iskljuci diodu zadate boje

nikakvu akciju već samo uređaj prepoznaje da je bio adresiran. Poruke se „izbacuju“ na mrežu sa periodom T koju je potrebno konfigurisati.

Poruke koje se šalju na mrežu se čitaju iz fajla „*protocol.txt*“ koji u jednom redu ima osmobitnu poruku. Poruke se iz fajla čitaju sa periodom T i šalju se ka svim uređajima. Struktura osmobitnih poruka je prikazana na slici

Bit D7 osmobitne poruke nosi informaciju da li je u pitanju sistemska poruka. Sistemska poruka služi za startovanje prenosa, zaustavljanje prenosa i podešavanje periode slanja i primanja poruka po magistrali. Sistemske poruke su sledeće sadržine:

- 10000001 – Perioda čitanja 1s
- 10000010 – Perioda čitanja 0.5s
- 10000000 – Zaustavi prenos
- 11111111 – Započni prenos

Ukoliko je bit D7 setovan , u pitanju je sistemska poruka u suprotnom se posmatraju niži biti.

Ukoliko je setovan BIT D6 poruka koja se šalje je BROADCAST poruka. Na BROADCAST poruku reaguju svi uređaji na mreži. U slučaju BROADCAST poruke ne posmatraju se biti D5 i D4 ali se posmatraju biti D3-D0 koji nose informaciju o tome koju akciju treba sprovesti na uređajima. Ukoliko bit D6 nije setovan posmatraju se niži biti.

Bitima D5 i D4 određena je adresa uređaja u slučaju da je bit D6 jednak 0. To znači da na željenu instrukciju koja se šalje bitima D3 i D2 odgovara jedino uređaj čija adresa se poklapa sa vrednošću grupe bita D5 i D4.

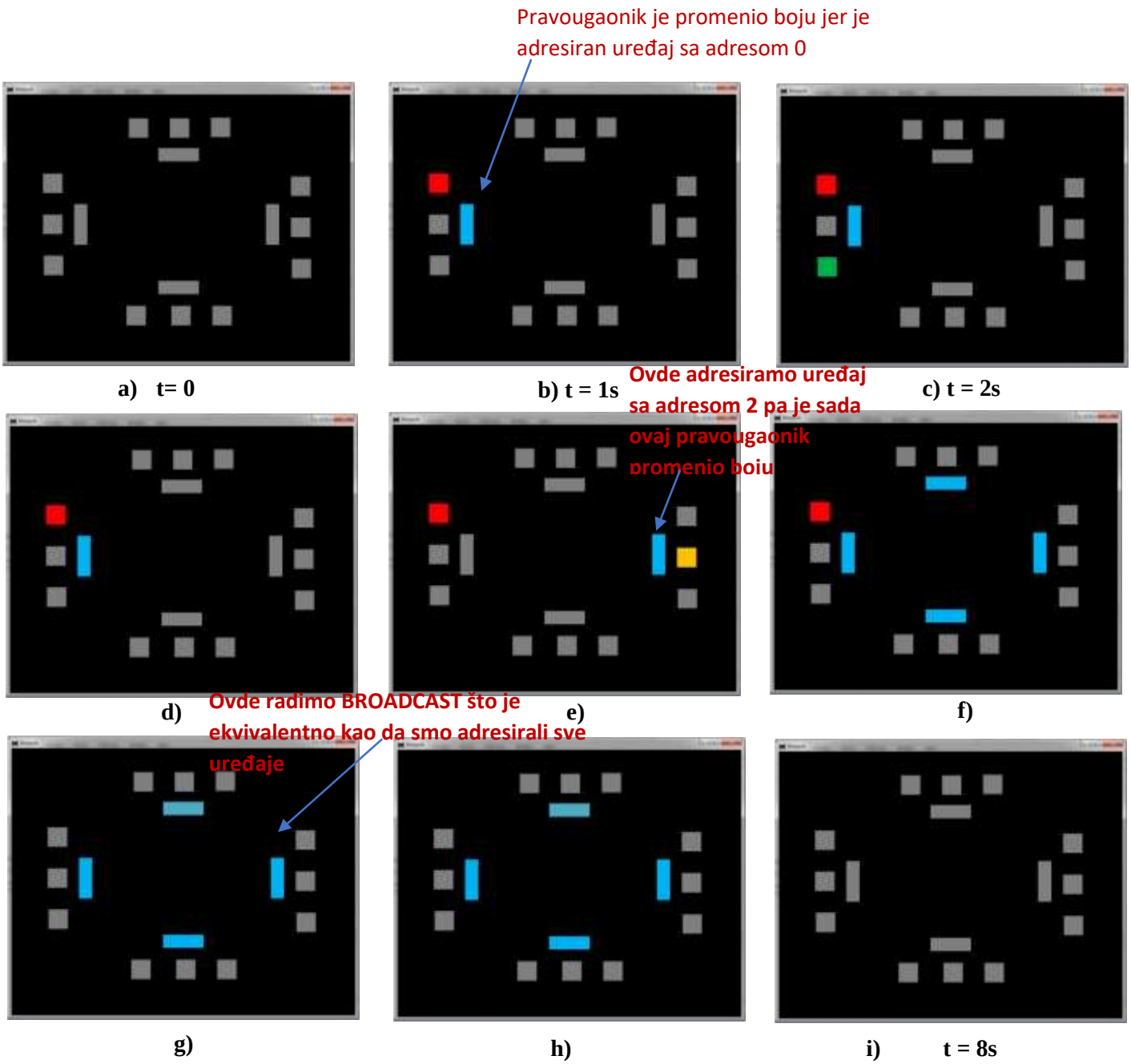
Biti D3 i D2 nose informaciju o instrukciji koja se šalje uređajima. Biti D1 i D0 nose informaciju o diodi koju treba uključiti ili isključiti.

Primer jednog *protocol.txt* fajla kao i željena funkcionalnost u konzoli dati su u sledećem primeru:

```
11111111 – Započni prenos (a)
10000001 – Perioda čitanja iz fajla je 1s (a)
00000100 – Uključi crvenu diodu na uređaju čija je adresa 0 (b)
00000111 – Uključi zelenu diodu na uređaju čija je adresa 0  (c)
00001011 – Isključi zelenu diodu na uređaju čija je adresa 0 (d)
00100101 – Uključi žutu diodu na uređaju čija je adresa 2 (e)
01101001 – Na svim uređajima isključi žutu diodu (f)
01101000 – Na svim uređajima isključi crvenu diodu (g)
01100000 – Svi uređaji dobili Nope Instrukciju , zadržavaju stanje. (h)
10000000 – Zaustavi prenos (i)
```

Pravougaonik plave boje se aktivira uvek kada je neki uređaj adresiran. Pravougaonik menja boju sa periodom jednakoj periodi čitanja fajla.

Ukoliko neke pojedinosti nisu navedene tekstom projektnog zadatka usvojiti razumne predpostavke. Potrebno je potrvditi funkcionalnost projekta na barem 3 različita fajla *protocol.txt* .



Slika 16.2 Ilustracija funcionalnosti mreže

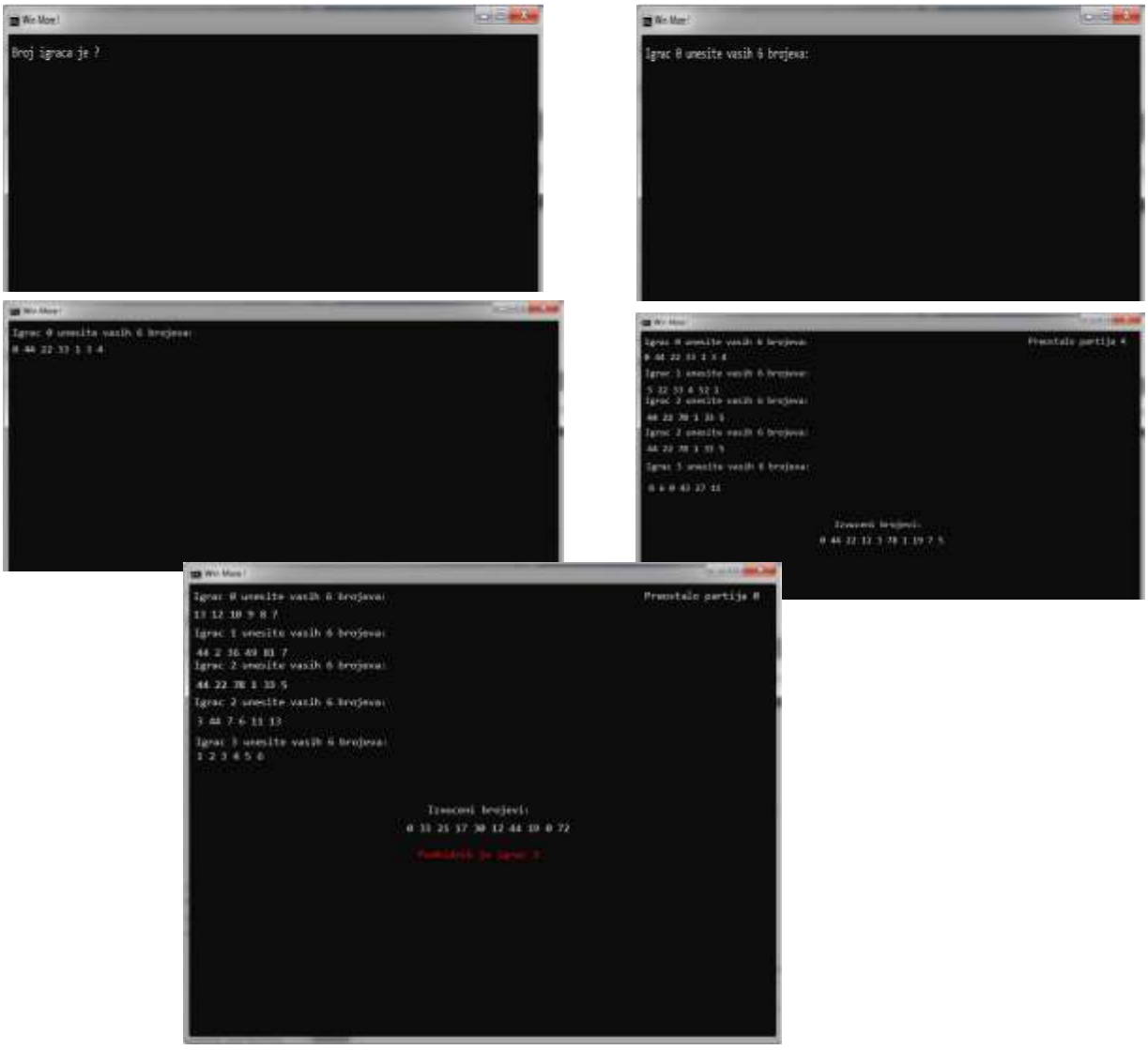
Projekat 17 - Win more!

U ovoj igrici cilj je da igrači predvide koje brojeve će kompjuter izbaciti. Igrica ima 5 partija i u svakoj partiji kompjuter izbacuje 10 brojeva od kojih igrači moraju da pogode najmanje 4 da bi imali neke poene. Brojevi koje kompjuter izbacuje su u opsegu od 0 do 77. Brojevi ispisuju na ekranu sa periodom od 1s.

Na početku igrice od korisnika se traži da unese broj igrača . Nakon unosa broja igrača kreira se onoliko izlaznih fajlova koliko ima igrača. Svaki izlazni fajl nosi naziv *PlayerSum_i.txt* gde je *i* redni broj igrača. Izlazni fajl treba da ima sledeću strukturu:

```
ID Igrača:
-------------------
Broj partije:
Kombinacija:
Izvučeni brojevi:
Broj osvojenih poena:
--------------------
Broj partije:
Kombinacija:
Izvučeni brojevi:
Broj osvojenih poena:
---------------------
....
---------------------
Ukupno osvojenih poena:
```

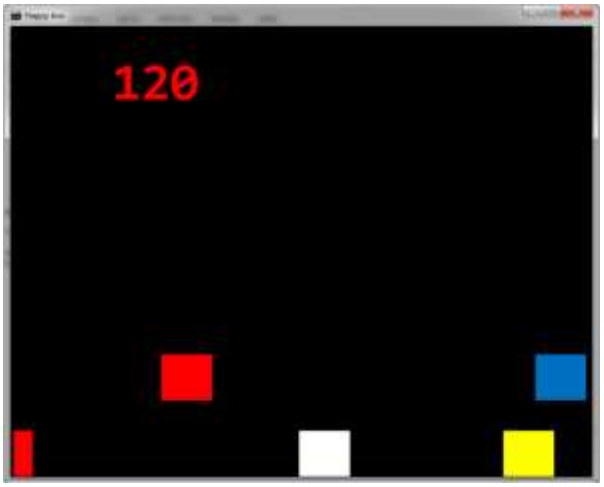
Na početku svake partije svaki od igrača unosi 6 brojeva. Ukoliko igrač unese broj koji nije u opsegu od 0 – 77 tražiti od korisnika da ponovo unese 6 brojeva ispočetka. Ukoliko igrač pogodi 4 i manje od 4 broja dobija 0 poena. Ukoliko pogodi 5 brojeva dobija 10 poena a u slučaju da pogodi svih 6 brojeva dobija 50 poena. Nakon odigranih 5 partija, svakom igraču se u njegov izlazni fajl upisuje broj osvojenih poena , dok se u konzoli ispisuje broj igrača koji ima najviše poena. Na sledećim slikama je prikazan interfejs programa prema korisniku



Projekat 18 - Happy box *

U ovoj igrici cilj je sakupiti što više žutih i plavih kocki a izbeći crvene kocke. Bela kutija, koja skuplja plave i žute kocke , može se nalaziti samo na dve horizontalne pozicije (gornja i donja). Prelazak iz jedne u drugu horizontalnu poziciju vrši se preko navigacionih tastera (strelica ka gore i strelica ka dole) na tastaturi. Pomeranje duž jedne horizontale je takođe moguće pomoću preostala dva navigaciona tastera (strelica levo i strelica desno) . Kocke u gornjoj i donjoj horizontali se ne kreću istom brzinom. Kocke se generišu u proizvoljnim vremenskim trenucima.

Broj poena koje nosi plava kocka iznosi 100 dok broj poena koje nosi žuta kocka iznosi 20. Igrica traje sve dok se ne pokupi crvena kocka ili dok igrač ne pritisne ESC taster. Za vreme trajanja igre se u gornjem delu ekrana ispisuje trenutni broj osvojenih poena. Interfejs igrice prikazan je na slici.



Slika 18.1 Grafički intefejs igrice

Projekat 19 - Igra memorije

Potrebno je napraviti igru memorije koja ima sledeće funckionalnosti: igrač na početku bira da li će tabla biti 4x5 ili 5x4. Zatim se nasumično postavljaju karakteri (poželjno je da to budu brojevi) i prikazuju se tabla sa raspoređenim znakovima. Takva tabela ostaje vidljiva 15 sekundi i onda se prikazuje sakrivena tabela. Korisnik unosi željenu koordinatu (npr. AA, ukoliko želi da proveri prvu koordinatu) a zatim i drugu i ukoliko je par pronađen on se zadržava otkriven na tabli a ukoliko nije, sakriva se nakon 5 sekundi. Igra je gotova kada se svi parovi pogode.

	A	**B**	**C**	**D**	**E**
A	1	3	8	7	5
B	2	0	6	4	1
C	5	7	8	9	0
D	4	9	2	3	6

Slka 19.1 4x5 tabela sa otkrivenim brojevima

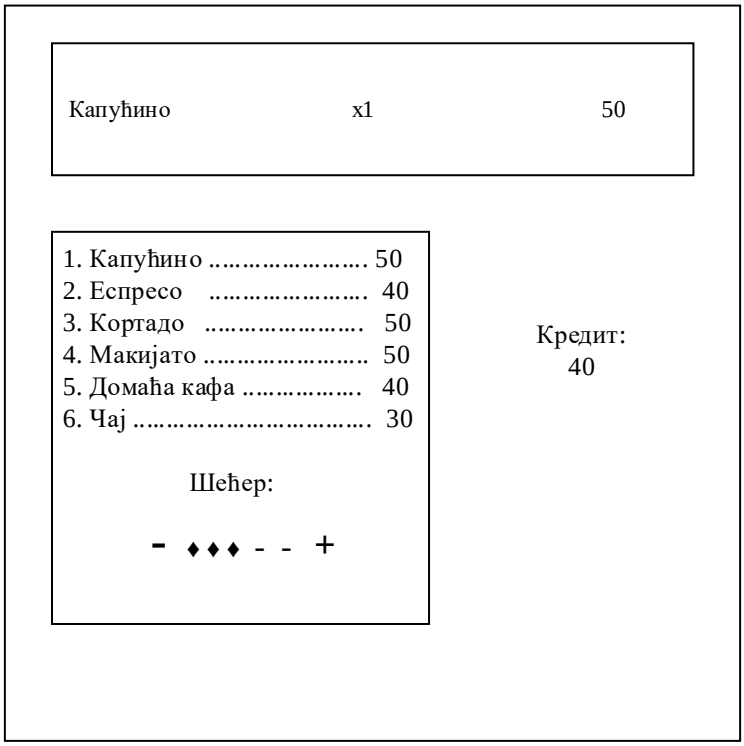
	A	**B**	**C**	**D**	**E**
A	♥	♥	♥	♥	♥
B	♥	♥	♥	♥	♥
C	♥	♥	♥	♥	♥
D	♥	♥	♥	♥	♥

Slka 19.2 4x5 tabela sa skrivenim znakovima

Za pomoć i korisne funkcije pogledati *Irvine* biblioteku.

Projekat 20 - Automat za kafu

Ideja je da se isprojektuje automat za kafu kakav se viđa na hodnicima fakulteta. Automat ima sledeće funkcije: prikaz raspoloživih toplih napitaka, nivo šećera, trenutni iznos kredita i, ukoliko postoji, kusur. Pomoću broja se bira željena vrsta napitka a ukoliko se unesu broj i znak plus ili minus(u kombinaciji 1+, 1-), naručuje se više tj manje napitaka iste vrste. Može se naručiti više napitaka od jednom. Priprema svakog napitka traje 2s. Šećer se podešava unosom znaka + ili -. Voditi računa da se ni jedan brojač ne može prevrteti. Ukoliko nema dovoljno kredita, ispisati poruku, i stornirati porudzbinu. Na odeljku kusur ispisivati ogovarajuću cifru nakon isteka vremena za pripremu napitaka.



Slika 20.1 Interfejs automata za kafu

Projekat 21 - Izbegavanje padajućih smetnji

Padajuće smetnje se realizuju kao blok karakteri koji padaju sa vrha ekrana sa nasumičnih pozicija. Igrač se predstavlja „☺“ karakterom i kreće se po dnu ekrana, levo i desno, pritiskom na slova A(ide se levo) i D(ide se desno). Na početku, igrač poseduje tri života koji se umanjuju ukoliko ga dotakne padajuća smetnja. Pored padajućih smetnji, postoje i padajuće nesmetenje (predstavljaju se preko ♣ karaktera, ili nekog drugog karaktera po izboru). Igrač treba da skuplja padajuće nesmetnje i na taj način osvaja poene. Kada se izgube sva tri života igra se završava i potrebno je prikazati broj poena osvojenih u igrici.

Projekat 22 - Šah*

Cilj je napraviti partiju šaha za dva igrača. Tabla se može predstaviti na sledeći način:

Slika 22 1 Predlog izgleda table za šah

Poželjno je figurice predstavljati slovima kao na slici. Igra se naizmenično tako što se unosi koordinata željenog polja (npr a1 ili f4). Voditi računa o specijalnim načinima kretanja određenih figurica. Partija se završava ili posle 54 poteza ili ukoliko se regularno izgubi partija.

Projekat 23 - GUI kalkulator*

Ideja je da se jednostavne računske operacije kao što su sabiranje, oduzimanje, množenje, deljenje i stepenovanje predstave preko lepo uređenog grafičkog interfejsa. Operacijama se pristupa unosom operanada i znakova operacije preko tastature(+, -, *, / i ^). Potrebno je formirati dugmiće na kojima će biti predstavljeni brojevi i operacije i displej na kome će se ispisivati rezultat u zavisnosti od unetih parametara.

Projekat 24 - Čoveče ne ljuti se*

Igra je poznata, cilj je uterati sve igrače u kućicu pre ostalih. Igra se u četvoro, tabla ima 16 polja raspoređenih u kvadrat. Kockica se generiše pomoću slučajnih brojeva koji se prikazuju na vrhu ekrana za igru. Igrači se predstavljaju uz pomoć blok karaktera crvene, žute, zelene i plave boje. Boje igrača proizvoljno raspodeliti, s tim da igru počinje gornji levi igrač, i ide se u smeru kazaljke na satu. Igra se završava kada prvi igrač popuni celu kućicu. Protivnički igrači se smeju preskakati i jesti, i kada se pojedu, vraćaju se u svoju bazu.

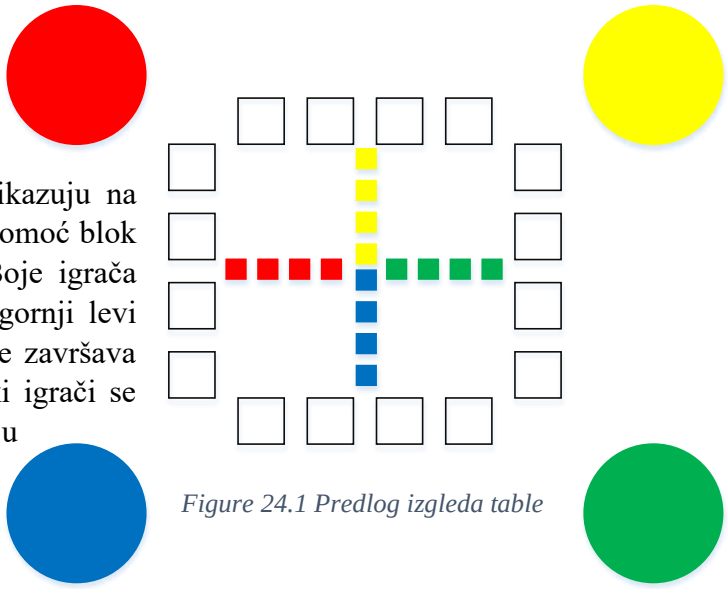


Figure 24.1 Predlog izgleda table

Projekat 25 - Color picker

Ideja je da studenti prođu kroz generisanje različitih boja u asmebleru. Prvo se na početnom ekranu prikazuju 3 kvadrata u osnovnim bojama, proizvoljne veličine(crvena,

plava, žuta). Ispod kvadrata se nalaze respektivno slova R, B i Y. Od korisnika se tada očekuje unos maksimalno dva slova i tada se prelazi na biranje nijanse. Ukoliko se unese samo jedno slovo, prelazi se na kvadrate koji su i nijansi izabrane boje, s tim da se pocinje od bele i zavrsava sa crnom. Ispod ovih kvadrata se nalaze brojevi, i svaka boja ima 9 nijansi. Ukoliko se unesi 2 slova, ulazi se u novu boju (narandžasta, ljubičasta i zelena) a zatim se ponovo biraju njihove nijanse kako je navedeno.

Slika 25.1 Izgled početnog ekrana

Slika 25.2 Izgled ekrana sa nijansama

Nijanse birati proizvoljno. Na kraju potrebno je odabranom bojom ispisati na konzoli unapred spremljeni tekst: „*Color picker*“.

Projekat 26 – Maze 3D*

Ideja je da se napravi igra u kojoj se prolazi kroz lavirint sa grafičkim prikazom iz 'first person' perspektive. Lavirint se čuva kao dvodimenzioni niz kojim se predstavlja postojanje zidova/kvadrata (kao na slici). 3D efekat se postiže tako što se na ekranu crta 'first person' projekcija koja se sastoji od zidova, poda i plafona kao na slici. Igrač se unosom sa tastature može pomeriti napred za jedan kvadrat, ili izvršiti rotaciju od 90 stepeni u levo ili desno. Igra se završava kada se dođe do izlaza, koji je predstavljen drugačijom bojom poda. Tada se ispisuje poruka da je igra završena i ukupno potrošeno vreme.