

# **Izveštaj - Faza I**

**Byte**

Anastasija Vasić 18096

Lucija Stojković 18434

Filip Vidojković 18108

---

## Uvod:

U igri "Bytes", koju igraju dva igrača - crni i beli, na tabli dimenzija od 8,10 ili 16, moguće je pomerati figure dijagonalno za jedno mesto u sva četiri pravca. Za igru se koriste samo tamna polja table. Svaki igrač, u skladu sa svojom bojom, može pomerati samo figure te boje. Cilj je formirati stekove od 8 figura na tabli, koristeći se pravilima prenošenja figura na susedna polja. Boja vrha svakog steka označava vlasnika i taj stek se uklanja sa table. Pobednik je onaj igrač koji ima više stekova u svom vlasništvu. Igra se prekida ukoliko jedan igrač poseduje više od polovine stekova na tabli ili tabla ostane prazna.

### 1) Način predstavljanja igre

U cilju simulacije igre "Byte", koristili smo matricu kao osnovni model za reprezentaciju trenutnog stanja igre. Na ovoj matrici vizualno prikazujemo ključne elemente igre: prazna polja, dostupna mesta za postavljanje figura, pozicije formiranih stekova tokom igre, kao i sadržaj samih stekova. Stekovi su predstavljeni kao manje matrice, gde su figure X i O igrača precizno raspoređene.

Na sledećoj slici se može videti početno stanje igre, sa jasnim prikazom matrice i svih relevantnih elemenata.

Prikazana matrica je dimenzije  $n=8$ . Polja ispunjena tačkicama smatraju se crnim poljima i jedino se ona koriste prilikom igranja igre.

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | . | . | . | . | . | . | . | . |
| B | . | . | . | . | . | . | . | . |
| C | . | . | . | . | . | . | . | . |
| D | . | . | . | . | . | . | . | . |
| E | . | . | . | . | . | . | . | . |
| F | . | . | . | . | . | . | . | . |
| G | . | . | . | . | . | . | . | . |
| H | . | . | . | . | . | . | . | . |

```

14 def ispisi_veliku_tablu(velika_tabla):
15     columns_labels = ' ' + ' '.join([' ' * 2 + str(i + 1) + ' ' * 2 for i in range(velicina_vece_table)])
16     print(columns_labels)
17
18     for row_index, row in enumerate(velika_tabla):
19         row_label = chr(ord('A') + row_index)
20
21         for i in range(3):
22             print(row_label if i == 1 else ' ', end=' ')
23             for small_table in row:
24                 print(' '.join(small_table[i]), end=' ')
25             print()
26         print()
27
28 1 usage
29 def kreiraj_malu_tablu_t(): #ima tackice u sebi
30     return [['.' for _ in range(3)] for _ in range(3)]
31
32 1 usage
33 def kreiraj_malu_tablu_b(): # ima blank znake u sebi
34     return [[' ' for _ in range(3)] for _ in range(3)]
35
36 1 usage

```

Funkcija *ispisi\_veliku\_tablu()* omogućava vizualizaciju kompletnog stanja table.

Funkcija *kreiraj\_malu\_tablu\_t()* kreira i vraća matricu dimenzija 3x3 popunjenu tačkicama ('.'). Svaka od matrica je inicijalno prazna i koristi se kao jedinica za formiranje veće table u igri "Bytes".

Funkcija *kreiraj\_malu\_tablu\_b()* kreira i vraća matricu dimenzija 3x3 popunjenu prazninama (' '). Svaka od matrica je inicijalno prazna i takodje se koristi kao jedinica za formiranje veće table u igri "Bytes".

## 2) Način predstavljanja početnog stanja

```

37 def pocetnoStanje():
38
39     # CRNO - X
40     for x in range(2, velicina_vece_table, 2):
41         for y in range(2, velicina_vece_table+1, 2):
42             pristupi_specifnoj_tacki(int(x - 1), int(y - 1), char: 'X', placeX: 0, placeY: 0)
43     # BELO - O
44     for x in range(3, velicina_vece_table, 2):
45         for y in range(1, velicina_vece_table, 2):
46             pristupi_specifnoj_tacki(int(x - 1), int(y - 1), char: 'O', placeX: 0, placeY: 0)
47
48 1 usage

```

Funkcija *pocetnoStanje()* inicijalizuje početno stanje igre "Bytes". Postavlja crne i bele figure oznakama 'X' i 'O' na tamna polja šahovske table. Crne figure postavlja na parne redove i kolone, označavajući ih 'X', dok bele figure postavlja na neparne redove i kolone, označavajući ih sa 'O'.

```

2 usages
33 def pristupi_specifichnoj_tacki(row, column, char, placeX, placeY):
34     specific_small_table = velika_tabla[row][column]
35     specific_small_table[placeX][placeY] = char
36

```

Funkcija *pristupi\_specifichnoj\_tacki()* omogućava pristup određenoj tački (polju) unutar neke male matrice. Primjenjuje se tako što prvo pristupa maloj matrici u određenom redu i koloni na tabli, a zatim postavlja vrednost prosleđenu kao 'char' poziciju unutar male matrice, koja je zadata parametrima placeX i placeY.

### 3) Mogućnost unosa početnih parametara igre

```

def podesavanje_igre():

    print("Izaberite opciju:")
    print("1 - Drugi igrač")
    print("2 - Kompjuter")
    while True:
        try:
            choice = int(input("Izaberi opciju (1 ili 2): "))
            if choice in [1, 2]:
                break
            else:
                print("Izbor nije validan. Izaberite 1 ili 2.")
        except ValueError:
            print("Izbor nije validan. Unesite broj")

    if choice == 1:
        order = input("Da li želite da igrate prvi ili drugi (Unesite 1 ili 2): ").lower()
        while order not in ['1', '2']:
            print("Izbor nije validan. Unesite 1 ili 2")
            order = input("Da li želite da igrate prvi ili drugi? ").lower()
    else:
        order = input("Da li želite da igrate prvi ili drugi (Unesite 1 or 2): ").lower()
        print(f"Vi ćete igrati {order}.")

    return (choice, order)

```

Metoda *podesavanje\_igre()* omogućava korisnicima da postave parametre igre. Prvo, korisnicima se pruža opcija da izaberu da li žele igrati protiv drugog igrača ili protiv računara. Nakon toga, korisnicima se postavlja pitanje da li žele igrati prvi ili drugi.

Kroz interaktivni unos, korisnici biraju opciju (1 ili 2) koja određuje da li će igrati protiv drugog igrača ili računara. Ukoliko je izbor nevalidan, korisnik se obaveštava o tome i traži se ponovni unos.

Nakon odabira opcije, korisnicima se postavlja dodatno pitanje o redosledu igranja. Korisniku se ponovo pruža povratna informacija u slučaju nevalidnog unosa, sa ponovljenim zahtevom za unos.

Sve ove informacije se zatim koriste za postavljanje osnovnih parametara igre, koji se na kraju vraćaju kao tuple, uključujući izbor opcije (1 ili 2) i redosled igre (1 ili 2). Ova funkcionalnost pruža korisnicima kontrolu nad početnim postavkama igre.

Unos dimenzije table:

```
322     print("Unesite dimenziju matrice: (8, 10 ili 16) ")
323     velicina_vece_table = int(input())
324     while(velicina_vece_table != 8 and velicina_vece_table != 10 and velicina_vece_table != 16):
325         print("Nije validna dimenzija")
326         velicina_vece_table = int(input("Unesite dimenziju matrice: (8, 10 ili 16) "))
```

Od korisnika se zahteva da unese parnu dimenziju matrice 8, 10 ili 16 i u petlji osigurava da se validira unos sve dok korisnik ne unese odgovarajuću vrednost.

#### 4) Funkcija za unos novog poteza

```
211 def unos_poteza(trenutni_igrac):
212     while True:
213         try:
214             pozicija = input("Unesite poziciju polja (npr. A1): ").upper()
215             kolona_broj=None
216             if len(pozicija)==2:
217                 if(pozicija[0].isalpha() and pozicija[1].isdigit()):
218                     kolona_broj = abs(int(pozicija[1]) - 1)
219                 else: raise ValueError("Neispravan unos za poziciju.")
220             elif len(pozicija)==3:
221                 if (pozicija[0].isalpha() and pozicija[1].isdigit() and pozicija[2].isdigit()):
222                     kb1= abs(int(pozicija[1]))
223                     kb2=abs(int(pozicija[2]))
224                     spojeno = str(kb1) + str(kb2)
225                     kolona_broj=abs(int(spojeno)-1)
226                 else:
227                     raise ValueError("Neispravan unos za poziciju.")
228             else:
229                 raise ValueError("Neispravan unos za poziciju.")
230             red_slovo=pozicija[0]
231             # Pretvaranje reda u indeks
232             red_broj = ord(red_slovo) - ord('A')
233             if (kolona_broj > velicina_vece_table or red_broj > velicina_vece_table):
234                 raise ValueError("Neispravan unos za poziciju.")
235             if not (ima_figura_u_polju(red_broj, kolona_broj)):
236                 raise ValueError("Izabrana pocetna pozicija nema figure.")
237             # Da li je polje sa kojeg polazimo crno
238             if velika_tabla[red_broj][kolona_broj][1] == ' ':
239                 print("Dozvoljeno je kretati se samo po crnim poljima table.")
240                 raise ValueError("Neispravan unos za poziciju.")
241             figura_mesto = int(input("Unesite mesto figure na steku koju pomerate (celi broj): "))
242             smer = input("Unesite smer pomeranja (GL, GD, DL, DD): ").upper()
243             redNext = None
244             kolNext = None
245             redCurr = red_broj
246             kolCurr = kolona_broj
```

```

251     if smer not in ['GL', 'GD', 'DL', 'DD']:
252         raise ValueError("Neispravan unos za smer pomeranja.")
253     if smer == 'GL':
254         redNext = red_broj - 1
255         kolNext = kolona_broj - 1
256     if smer == 'GD':
257         redNext = red_broj - 1
258         kolNext = kolona_broj + 1
259     if smer == 'DL':
260         redNext = red_broj + 1
261         kolNext = kolona_broj - 1
262     if smer == 'DD':
263         redNext = red_broj + 1
264         kolNext = kolona_broj + 1
265     niz = remove_figures_from_matrix(velika_tabla[red_broj][kolona_broj], figura_mesto, brRedova: 3, brKolona: 3, trenutni_igrac, redNext, kolNext)
266     mala_matrica = velika_tabla[redNext][kolNext]
267     if (potez_validan(redNext, kolNext, redCurr, kolCurr)):
268         niz_izbacenih = niz[0]
269         pred_visina = niz[1]
270
271         k = 0
272         for i in range(3):
273             for j in range(3):
274                 if (mala_matrica[i][j] == '.'):
275                     mala_matrica[i][j] = niz_izbacenih[k]
276                     k = k + 1
277                 if (k == len(niz_izbacenih)):
278                     print(mala_matrica)
279                     ispisi_veliku_tablu(velika_tabla)
280                     return 0
281
282 except ValueError as e:
283     print(f"Greška pri unosu poteza: {e}")
284     return None

```

Funkcija `unos_poteza()` prima trenutnog igrača kao ulaz i omogućava mu da unese potez u igri koja se odvija na tabli. Prvo traži unos pozicije polja i proverava ispravnost unosa. Zatim traži unos mesta figure na steku, smeru pomeranja i izvršava određene provere na osnovu unosa.

- Proverava se ispravnost unosa pozicije i proverava se da li je polje na koje igrač želi da pomera figuru crno.
- U zavisnosti od unetog smeru, određuje se nova pozicija figure.
- Izvršava se funkcija `remove_figures_from_matrix` kako bi se uklonile figure sa trenutne pozicije na tabli.
- Proverava se validnost poteza pomoću funkcije `potez_validan()`.
- Ako je potez validan, figure se smeštaju na novo odredište na tabli, a zatim se nova tabla šalje kao rezultat.
- U slučaju bilo koje greške pri unosu, korisnik dobija odgovarajuću poruku o grešci. Ako se potez uspešno izvrši, vraća se nova pozicija figure, mesto figure na steku i smer pomeranja.



## 5) Provera validnosti poteza

```
93 def potez_validan(rowNext, colNext, rowCurr, colCurr, countMoved):
94     # Da li su nove koordinate u tabli
95     if rowNext < 0 or rowNext >= len(velika_tabla) or colNext < 0 or colNext >= len(velika_tabla[0]):
96         print("Izabrano polje se ne nalazi na tabli!")
97         return False
98     # Da li se krecemo dijagonalno i da li se krecemo za samo jedno polje
99     if abs(rowNext - rowCurr) != abs(colNext - colCurr) and abs(rowNext - rowCurr) != 1 and abs(colNext - colCurr) != 1:
100         return False
101     # Da li je polje na koje zelimo da odemo crno
102     if velika_tabla[rowNext][colNext][1][1] == ' ':
103         print("Dovoljeno je kretati se samo po crnim poljima table.")
104         return False
105     # Provera da na odabranom mestu ne bude previse figurica
106     count = sum(1 for row in velika_tabla[rowNext][colNext] for small_table_row in row for elem in small_table_row if elem != '.')
107     if (count + countMoved > 8):
108         print(f"Broj figurica na odabranom polju je {count}, pokušavate premestiti {countMoved}, ukupan broj figurica {count + countMoved} je veci od 8! Pokušajte ponovo")
109         return False
110
111     return True
```

Ova funkcija, *potez\_validan()*, ima za cilj proveru ispravnosti poteza uzimajući u obzir pravila igre. Ključne tačke koje obuhvata:

- Provera da li se polje nalazi na tabli: Prvo se proverava da li su indeksi odredišnog polja (rowNext, colNext) unutar validnih granica velike table. Ukoliko nisu, funkcija ispisuje poruku o grešci i vraća False.
- Provera dozvoljenog pomeranja: Funkcija proverava da li pomeranje figure ima validnu dužinu i smer. Potezi su dozvoljeni ako se figura pomeri dijagonalno i to za jedno polje. Ako uslov nije zadovoljen, funkcija vraća False.
- Provera boje polja: Proverava se boja polja na destinaciji (rowNext, colNext). Potezi su dozvoljeni samo ako je odredišno polje crno. U suprotnom, funkcija ispisuje poruku o grešci i vraća False.
- Provera broja figura na polju: Funkcija računa trenutni broj figura na odredišnom polju i proverava da li će pomeranje izazvati prekoračenje maksimalnog broja figura na jednom polju (8 figura). Ako je prekoračenje moguće, funkcija ispisuje poruku o grešci i vraća False.
- Povratna vrednost True: Ukoliko su svi uslovi zadovoljeni, funkcija vraća True, označavajući da je potez validan prema pravilima igre.

Ova funkcija ulogu osiguravanja ispravnosti poteza, pružajući informacije o tome da li je potez validan ili ne.

```
98 def ima_figura_u_polju(rowCurr, colCurr):
99     for row in velika_tabla[rowCurr][colCurr]:
100         for small_table_row in row:
101             for elem in small_table_row:
102                 if elem != '.':
103                     return True
104     return False
```

Funkcija *ima\_figura\_u\_polju(rowCurr, colCurr)* proverava da li postoje figure na određenom polju (rowCurr, colCurr) na tabli. Koristi se petlja koja prolazi kroz elemente male matrice i proverava da li se na bilo kojem polju male matrice nalazi figura, vraćajući True ukoliko pronađe figuru ili False ako ne pronađe nijednu figuru na tom polju.

## 6) Provera kraja igre

Na kraju svakog poteza funkcija *pobedio(n)* proverava da li je igra gotova.

Ova metoda proverava da li postoji pobednik u igri "Bytes" na osnovu trenutnog stanja stekova i broja figura na tabli. Ako jedan od igrača ima više od polovine svih stekova ili ako je tabla prazna, igra se završava ispisom odgovarajućih poruka.

```
137 def pobedio(n):
138     ukupanBrojStekova = int((n-2)*(n/2)/8)
139
140     if(stek1 > stek2 and (stek1 >= (ukupanBrojStekova+1)//2)):
141         print("Igrac broj 1 je pobedio")
142         print("IGRA JE ZAVRSENA")
143         sys.exit()
144     elif(stek1 < stek2 and (stek2 >= (ukupanBrojStekova//2))):
145         | print("Igrac broj 2 je pobedio")
146         print("IGRA JE ZAVRSENA")
147         sys.exit()
148     count = [sum(1 for row in matrix for elem in row if elem != '.') for matrix in velika_tabla]
149     total_count = sum(count)
150     if(total_count!= 0):
151         print("IGRA JE ZAVRSENA")
152         sys.exit()
```