Semestralni rad iz Diskretne matematike

Tjuringova mašina

Nikola Ilić, 2018/0063
Matija Dodović, 2018/0072

October 10, 2019

1 Uvod

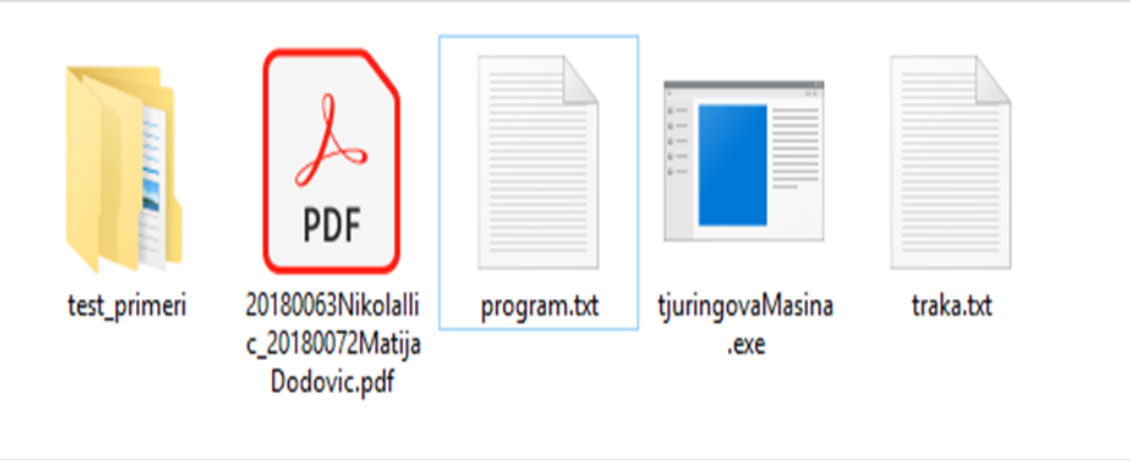
Tjuringova mašina je uređena sedmorka (S, b, Q, q_0, q_+, q_-, f), gde je S azbuka, b blanko znak (prazno polje), Q skup stanja sa izdvojenim q_0 - početnim, i q_+ i q_- - završnim stanjima sa pozitivnim, odnosno negativnim odgovorom. f je program mašine i on je seldećeg oblika:

$$f(q_i, a_j) = (q_i', a_j', r)$$

gde su q_i i q_i' trenutno i naredno stanje, a a_j i a_j' trenutni i novi znak. r je iz skupa $\{+1, -1\}$ i određuje da li se glava mašine pomera u levo ili u desno.
Program za realizaciju Tjuringove mašine učitava početnu traku i program, što je opisano na *Slici 2*. Traka može sadržati elemente bilo koje azbuke, uz ograničenje da mora početi i završiti se po jednim znakom b. Glava Tjuringove mašine je pozicionirana na poziciju 2, uz indeksiranje pozicija od 1, i ukoliko taj element bude prazan (b) program javlja grešku (što je ilustrovano na *Slici 4*). Pri pokretanju programa ispisuje se polazna traka.
Funkcija se zadaje u standardnom formatu, uz obavezno postojanje stanja q_0. Ne mogu postojati blanko znaci u funkciji. Ukoliko je izvršavanje programa korektno, ispisuju se traka i sledeći poziv funkcije. Ukoliko je odogovor pozitivan ispisuje se finalna traka u suprotnom se prijavljuje greška, *Slika 3*.

2 Postupak korišćenja

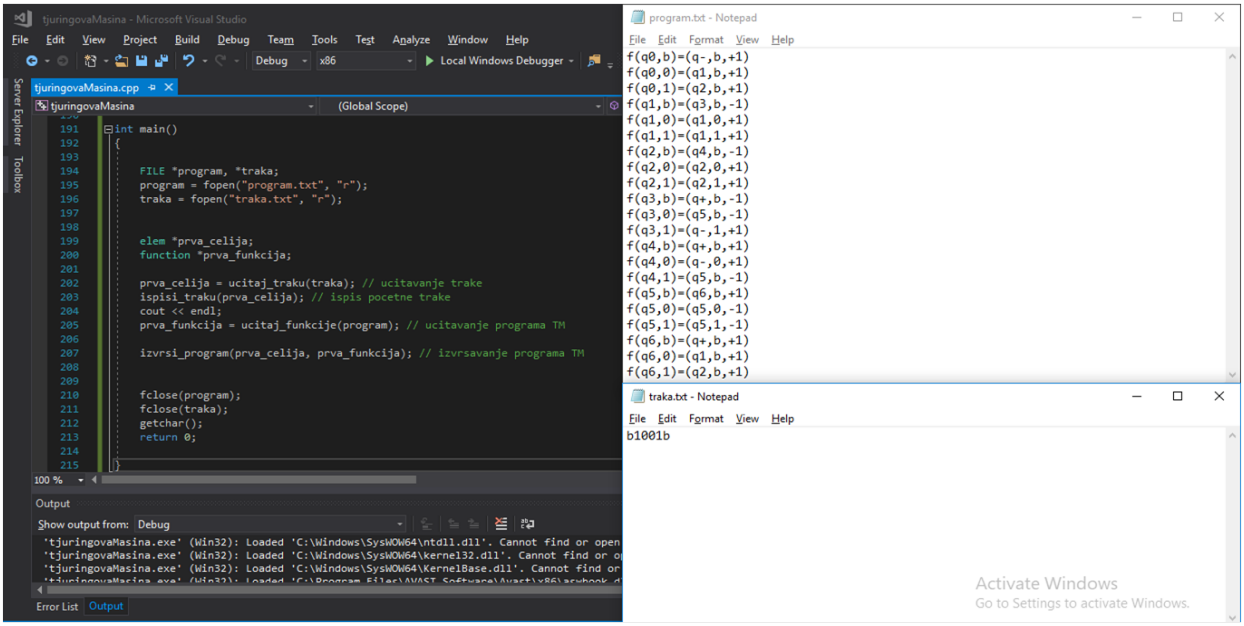
Na *Slici 1* je prikazan folder sa sadržajem potrebnim za pokretanje programa za izvršavanje Tjuringove mašine.



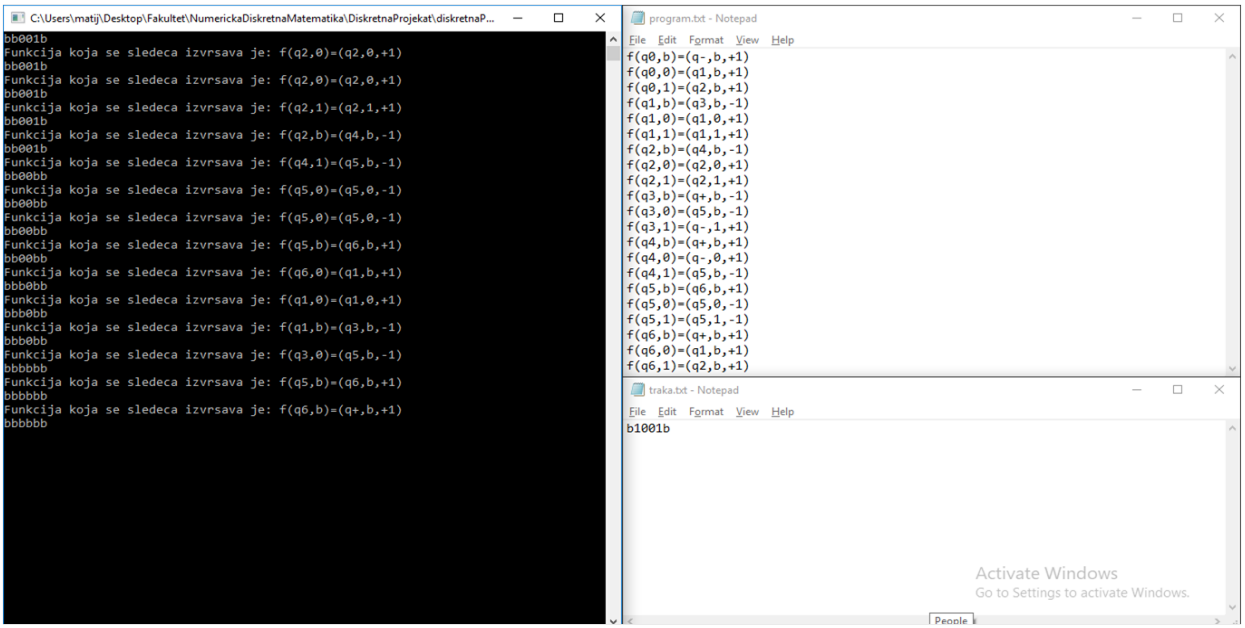
Slika 1: Prikaz foldera za pokretanje Tjuringove mašine.

Egzekutabilni fajl *tjuringovaMasina.exe* se pokreće na standardan način. U istom folderu kao i taj fajl je potrebno da se nalaze dva tekstualna fajla: *program.txt* i *traka.txt*, gde su funkcija i traka, respektivno. Fajlovi su oblika opisanog u uvodu. Egzekutabilni fajl izvršava program Tjuringove mašine čitajući i obrad ujući tekstualne datoteke i svoje izlaze ispisuju na konzoli (*Slike 2* i 3). U folderu *test_primeri* nalaze se zasebni folderi sa tri test primera izvršavanja programa Tjuringove mašine. U sva tri foldera su fajlovi sa funkcijom i trakom, koje treba kopirati u direktnorijum sa egzekutabilnim fajlom, radi njihovog pokretanja. Ukoliko se program izvrši sa pozitivnim odgovorom ispisuje se nova traka, a ukoliko je odgovor negativan izlazi poruka: "Greška na traci!".

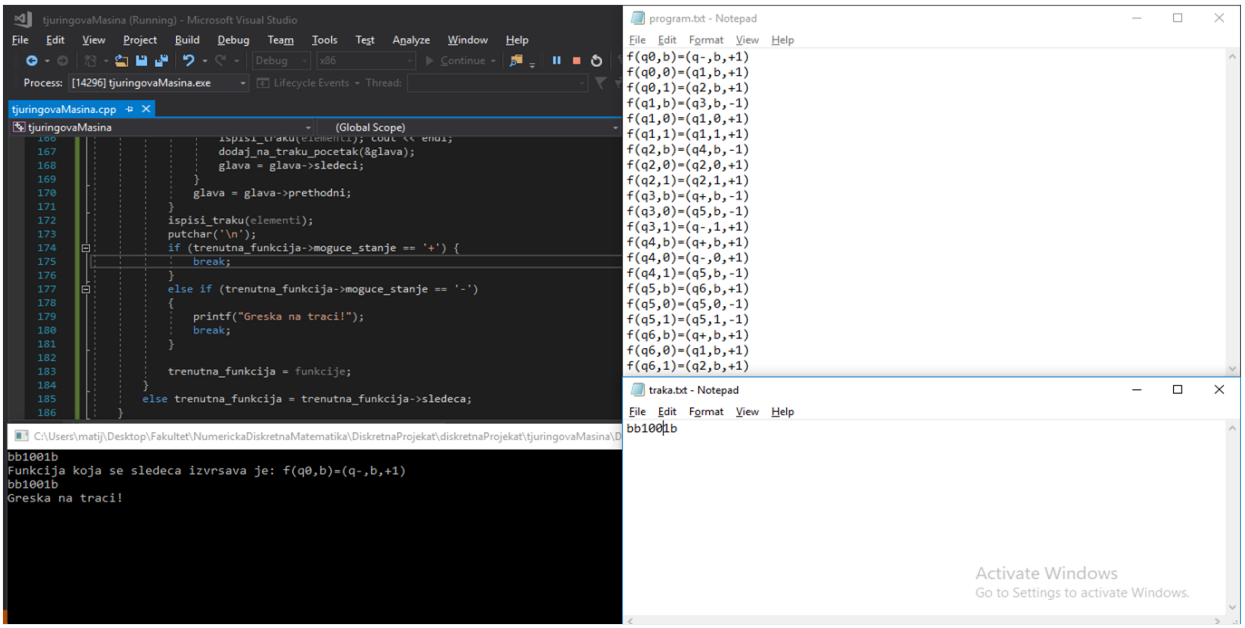
3 Prikaz rada koda



Slika 2: Prikaz *main* funkcije koda i primer po jedne datoteke *program.txt* i *traka.txt*. Pri pokretanju koda, linije koda 194-200 su zadužene za deklaracije i definicije, potrebne u kasnijem radu. Na liniji 202 se nalazi poziv procedure za učitavanje trake iz datoteke, koja se u naredne dve linije ispisuje (pozivom procedure *ispisi_traku*). To je početna traka koja služi radi poređenja ispravnosti rada programa, tj izvršavanja funkcija. Na linijama 205 i 207 se nalaze pozivi procedura za učitavanje programa Tjuringove mašine (iz odgovarajuće datoteke) i njegove relaizacije, respektivno. Ispis međukoraka i finalne trake se nalazi u samoj proceduri za izvršavanje programa Tjuringove mašine.



Slika 3: Primer korektnog izvršavanja programa koji ispituje da li je binaran broj palindrom. Prilikom učitavanja trake, koja počinje i završava se sa b, glava mašine pokazuje na broj (ne b). Izvršava se funkcija zadata u datoteci (*program.txt*), i u svakom koraku se ispisuje naredni poziv funkcije i trenutna traka. Pošto se poruka "Greška na traci!" nije ispisala, tj. u stanje q_- se nije ušlo, program se korektno završio (stanje q_+) i ispisuje se promenjena traka, odakle se zaključuje da zadani broj jeste palindrom.



Slika 4: Primer nekorektnog izvršavanja programa koji ispituje da li je binaran broj palindrom. U ovom primeru traka je zadata tako da glava pokazuje na element b. Po propozicijama, glava treba da pokazuje na početak broja, odnosno traka treba da bude zadata samo sa po jednim graničnim b znakom. Pokušava se sa identifikacijom nultog stanja i naredno stanje, usled glave koja pokazuje na b, jeste q_-. Ispisuje se traka do tog trenutka i poruka "Greška na traci!", uz prekidanje daljeg rada programa.

Kod 1. *Kod:*

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include <stdio.h>
#include <iostream>
#include <string>

using namespace std;

//NOVA STRUKTURA KOJA PREDSTAVLJA CELIJU TRAKE
typedef struct elem
{
     char podatak;
     struct elem *sledeci, *prethodni;
};

//NOVA STRUKTURA KOJA PREDSTAVLJA JEDNU FUNKCIJU
typedef struct function
{
     int trenutno_stanje, novo_stanje;
     char trenutni_znak, novi_znak, pomeranje_glave, moguce_stanje;
     struct function *sledeca;
};

//FUNKCIJA KOJA PRETVARA char BROJA U int
int vrati_broj(char c)
{
     return c - '0';
}

//FUNKCIJA ZA UCITAVANJE TRAKE IZ TEKSTUALNE DATOTEKE traka.txt
elem* ucitaj_traku(FILE *traka)
{
     elem *prvi = NULL, *novi, *posl = NULL;
     char c = fgetc(traka);

     while (c != EOF)
     {
          novi = (struct elem *) malloc(sizeof(elem));
          if (!novi) exit(1);
          novi->podatak = c;
          novi->sledeci = NULL;
          novi->prethodni = posl;
          if (prvi == NULL) prvi = novi;
          else posl->sledeci = novi;
          posl = novi;

          c = getc(traka);
     }
     return prvi;
}

//FUNKCIJA ZA UCITAVANJE FUNKCIJA TM IZ TEKSTUALNE DATOTEKE funkcije.txt
function* ucitaj_funkcije(FILE *funkcije)
{
     function *prva = NULL, *nova, *posl = NULL;
```

```
55      char c;
        int broj = 0;

        while ((c = getc(funkcije)) != EOF)
        {
60          nova = (struct function*) malloc(sizeof(function));
            if (!nova) exit(2);

            getc(funkcije);
            getc(funkcije);
65          while ((c = getc(funkcije)) != ',')
            {
                broj = broj * 10 + vrati_broj(c);
            }
            nova->trenutno_stanje = broj;
70          broj = 0;
            nova->trenutni_znak = getc(funkcije);
            getc(funkcije);
            getc(funkcije);
            getc(funkcije);
75          getc(funkcije);
            while ((c = getc(funkcije)) != ',')
            {
                if (c == '+' || c == '-')
                {
80                  nova->moguce_stanje = c;
                    break;
                }
                broj = broj * 10 + vrati_broj(c);
            }
85          if (c == '+' || c == '-') getc(funkcije);
            nova->novo_stanje = broj;
            broj = 0;
            nova->novi_znak = getc(funkcije);
            getc(funkcije);
90          nova->pomeranje_glave = getc(funkcije);
            getc(funkcije);
            getc(funkcije);
            getc(funkcije);
            nova->sledeca = NULL;
95          if (!prva) prva = nova;
            else posl->sledeca = nova;
            posl = nova;
        }

100     return prva;
    }

    //FUNKCIJA ZA ISPISIVANJE TRAKE
    void ispisi_traku(elem *prvi)
105 {

        elem *trenutni = prvi;
```

```
     while (trenutni != NULL)
     {
          printf("%c", trenutni->podatak);
          trenutni = trenutni->sledeci;
     }
}

//FUNKCIJA ZA DODAVANJE ELEMENTA b NA POCETAK TRAKE
void dodaj_na_traku_pocetak(elem **elementi)
{
     elem* novi = new elem;
     novi->sledeci = *elementi;
     novi->prethodni = NULL;
     novi->podatak = 'b';
     *elementi = novi;
}

//FUNKCIJA ZA DODAVANJE ELEMENTA b NA KRAJ TRAKE
void dodaj_na_traku_kraj(elem *elementi)
{
     elem* novi = new elem;
     elementi->sledeci = novi;
     novi->sledeci = NULL;
     novi->prethodni = elementi;
     novi->podatak = 'b';
}


//FUNKCIJA ZA IZVRSAVANJE PROGRAMA TM
void izvrsi_program(elem *elementi, function *funkcije)
{
     int stanje = 0;
     function *trenutna_funkcija = funkcije;
     elem *glava = elementi->sledeci;
     bool kraj = false;

     while (!kraj)
     {
          if (trenutna_funkcija->trenutno_stanje == stanje
               && trenutna_funkcija->trenutni_znak == glava->podatak)
          {
               if (trenutna_funkcija->moguce_stanje == '+' || trenutna_funkcija->moguce_stanje == '-
               {
                    printf("Funkcija koja se sledeca izvrsava je: f(q%d,%c)=(q%c,%c,%c)\n", trenutna
               }
               else
               {
                    printf("Funkcija koja se sledeca izvrsava je: f(q%d,%c)=(q%d,%c,%c)\n", trenutna
               }

               stanje = trenutna_funkcija->novo_stanje;
               glava->podatak = trenutna_funkcija->novi_znak;
               if (trenutna_funkcija->pomeranje_glave == '+') {
```

```
                    if (glava->sledeci == NULL) dodaj_na_traku_kraj(glava);
                    glava = glava->sledeci;
               }
               else {
                    if (glava->prethodni == NULL) {
                         ispisi_traku(elementi); cout << endl;
                         dodaj_na_traku_pocetak(&glava);
                         glava = glava->sledeci;
                    }
                    glava = glava->prethodni;
               }
               ispisi_traku(elementi);
               putchar('\n');
               if (trenutna_funkcija->moguce_stanje == '+') {
                    break;
               }
               else if (trenutna_funkcija->moguce_stanje == '-')
               {
                    printf("Greska na traci!");
                    break;
               }

               trenutna_funkcija = funkcije;
          }
          else trenutna_funkcija = trenutna_funkcija->sledeca;
     }

}


int main()
{

     FILE *program, *traka;
     program = fopen("program.txt", "r");
     traka = fopen("traka.txt", "r");


     elem *prva_celija;
     function *prva_funkcija;

     prva_celija = ucitaj_traku(traka); // ucitavanje trake
     ispisi_traku(prva_celija); // ispis pocetne trake
     cout << endl;
     prva_funkcija = ucitaj_funkcije(program); // ucitavanje programa TM

     izvrsi_program(prva_celija, prva_funkcija); // izvrsavanje programa TM


     fclose(program);
     fclose(traka);
     getchar();
     return 0;
```

```
215 }
```