

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Deathclaw

**Seminarski rad iz predmeta
Uvod u teoriju računarstva**

Zadatak broj 2011

Zagreb, lipanj 2009.

Seminarski rad iz predmeta Uvod u teoriju računarstva

Student: Deathclaw

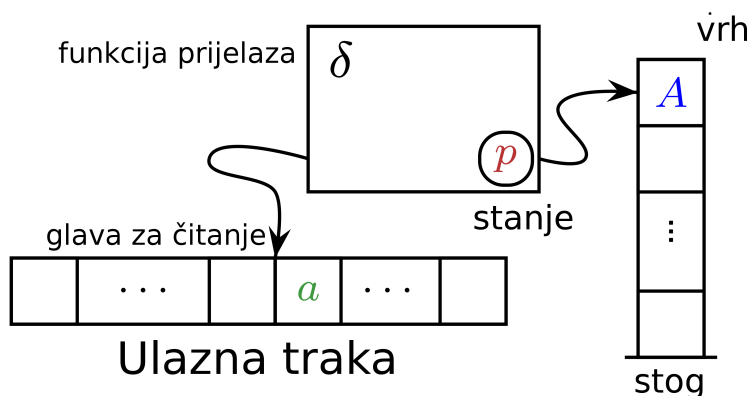
Matični broj studenta: *****

Zadatak broj 2011: Napisati program u programskom jeziku C ili C++ za simuliranje univerzalnog DPA.

Uvod

Potisni automat je konačni automat veoma sličan nedeterminističkom konačnom automatu s dodatnom komponentom – potisnim stogom (*LIFO* stog). Potisni stog omogućava dodatnu memoriju, te omogućuje potisnom automatu da prihvaća kontekсно neovisne jezike.

Rad glave za čitanje je isti kao i kod nedeterminističkog konačnog automata. Osim čitanja znakova ulazne trake, upravljačka jedinka čita i jedan znak vrha potisnog stoga. Nakon čitanja znaka, s vrha stoga uzima se pročitani znak, a na vrh stoga stavlja se niz znakova. Znakovi ulazne trake nazivaju se ulazni znakovi, dok se znakovi na stogu nazivaju znakovi stoga. Upravljačka jedinka je u jednom od konačnog broja stanja, ta stanja su podijeljena u dva skupa: prihvatljiva stanja i neprihvatljiva stanja.



Ilustracija 1: Model potisnog automata

Upravljačka jedinka donosi odluku o promjeni sadržaja vrha stoga, pomaku glave za čitanje i promjeni stanja. Odluka se donosi na temelju tri podatka:

- stanja upravljačke jedinice
- znaka koji je na vrhu stoga
- znaka ulazne trake

Odluka o prihvaćanju ulaznog niza donosi se na jedan od ova dva načina:

- prihvatljivim stanjem – niz je prihvatljiv kada je upravljačka jedinka u prihvatljivom stanju nakon pročitanih svih znakova ulazne trake, pritom stog ne mora biti prazan
- praznim stogom – niz je prihvatljiv kada se stog ispravi nakon pročitanih svih ulaznih znakova trake

Potisni automat je formalno zadan kao uređena sedmorka:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$$

gdje je

- | | |
|------------------|--|
| Q | - konačan skup stanja |
| Σ | - konačan skup ulaznih znakova |
| Γ | - konačan skup znakova stoga |
| $\delta \in Q$ | - konačan podskup $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$ |
| $q_0 \in \Gamma$ | - početno stanje |
| Z | - početni znak stoga |
| $F \subseteq Q$ | - skup prihvatljivih stanja |

Prema funkciji prijelaza potisni automat se dijeli na dvije vrste:

- deterministički potisni automati
- nedeterministički potisni automat

Potisni automat je deterministički ako su ispunjena slijedeća dva uvjeta:

- ako je $\delta(q, \varepsilon, Z)$ neprazni skup onda je $\delta(q, a, Z)$ prazni skup za bilo koji ulazni znak a iz Σ
- ako je u skupu $\delta(q, a, Z)$ najviše jedan element, i to za bilo koje stanje q iz Q , za bilo koji znak stoga Z iz Γ i za bilo koji ulazni znak a iz $\Sigma \cup \{\varepsilon\}$

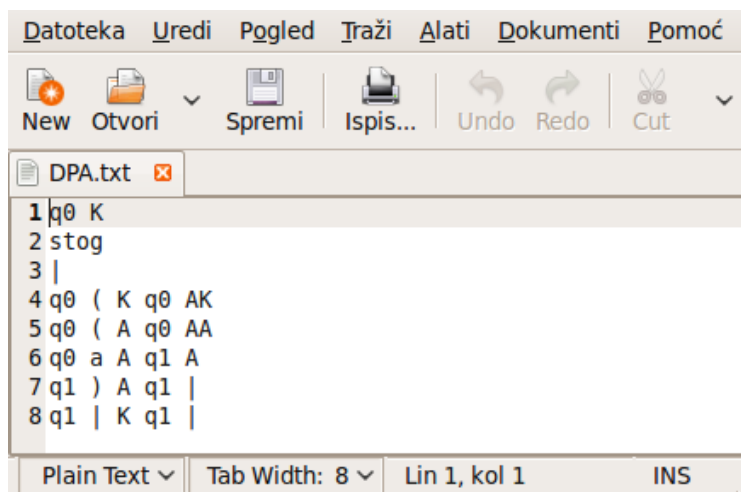
Ostvarenje

Simulator univerzalnog determinističkog potisnog automata ostvaren je u programskom jeziku C.

Simulator se može razložiti na tri osnovne cjeline:

- cjelina za čitanje tablice prijelaza, početnog stanja stoga, početnog stanja te načina prihvaćanja ulaznog niza iz datoteke
- cjelina za izvršavanje funkcije prijelaza
- cjelina za donošenje odluke prihvatljivosti ulaznog niza

Simulator očekuje ulaznu datoteku "DPA.txt" u istom direktoriju simulatora. Ulazna datoteka može imati ovakav izgled:



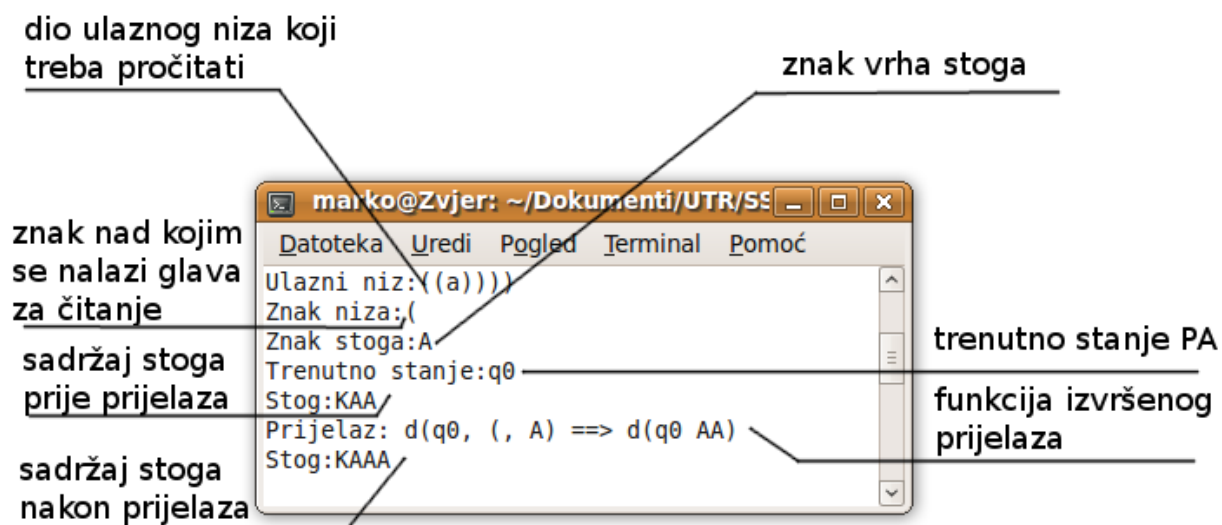
Ilustracija 2: Ulazna datoteka

gdje u prvoj liniji "q0" predstavlja početno stanje, a "K" predstavlja početni znak stoga. U drugoj liniji se bira način prihvaćanja ulaznog niza stanjem ili stogom. Za odabir prihvaćanja stanjem potrebno je upisati "stanje", a za prihvaćanje stogom potrebno je upisati "stog". Na trećoj liniji ulazne datoteke nalazi se prostor za upis prihvatljiva stanja, ako ih nema ili nisu potrebna onda se upisuje znak "|" koji predstavlja prazni skup \emptyset . Od četvrte linije pa nadalje upisuje se tablica prijelaza koja ima slijedeći oblik:

$$\delta(\text{trenutno stanje } q, \text{ ulazni znak } a, \text{ znak stoga } Z) = (\text{sljedeće stanje } p, \text{ sljedeći znak stoga } y) \text{ .}$$

Funkciju prijelaza simulator vrši tako da u petlji prolazi kroz sve učitane prijelaze iz datoteke te uspoređuje s pročitanim znakom niza, skinutim znakom sa stoga i s trenutnim stanjem potisnog automata. Simulator vrši prijelaze sve dok u tablici prijelaza postoji prijelaz za trenutne znakove ulaznog niza i stoga, te trenutnog stanja automata. Kada simulator izvrši sve moguće prijelaze ispituje prihvatljivost ulaznog niza. Prihvatljivost provjerava tako da provjeri da li je pročitani cijeli niz, te provjeri da li je stog prazan ili je automat u prihvatljivom stanju ovisno o načinu prihvaćanja ulaznog niza.

Izgled ispisa rezultata simulacije nakon jednog izvršenog prijelaza prikazano je slijedećom slikom:



Ilustracija 3: Primjer ispisa

Treba zamijetiti da vrh prikazanog stog je zadnji desni znak, a dno stoga prvi lijevi znak.

Primjer prihvatanja stogom:

Simulirani automat M_1 ima slijedeći oblik:

$M_1 = (\{q_0, q_1\}, \{(, a,)\}, \{A, K\}, \delta, q_0, K, \{\emptyset\})$ uz funkcije prijelaza :

$$\delta(q_0, (, K) = (q_0, AK)$$

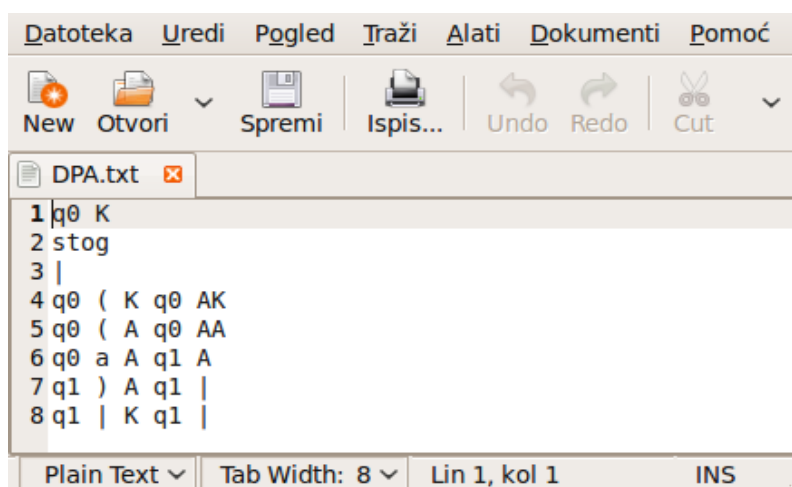
$$\delta(q_0, (, A) = (q_0, AA)$$

$$\delta(q_0, a, A) = (q_1, A)$$

$$\delta(q_1,), K) = (q_1, \varepsilon)$$

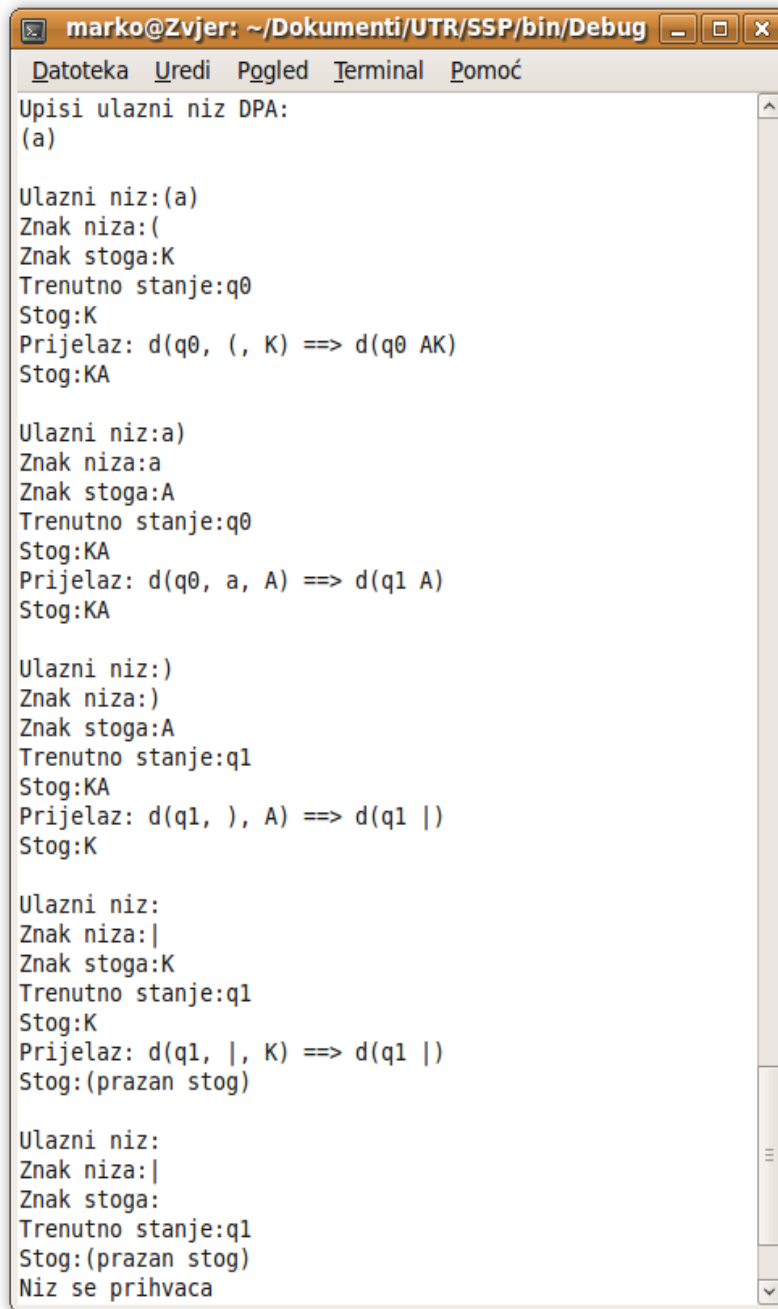
$$\delta(q_1, \varepsilon, K) = (q_1, \varepsilon)$$

Ulazna datoteka automata M_1 izgleda:



Ilustracija 4: Ulazna datoteka

Za ulazni niz "(a)" koji se upisuje nakon pokretanja simulatora ispis ima slijedeći oblik:



```
marko@Zvjer: ~/Dokumenti/UTR/SSP/bin/Debug
Datoteka  Uredi  Pogled  Terminal  Pomoć
Upisi ulazni niz DPA:
(a)

Ulazni niz:(a)
Znak niza:(
Znak stoga:K
Trenutno stanje:q0
Stog:K
Prijelaz: d(q0, (, K) ==> d(q0 AK)
Stog:KA

Ulazni niz:a)
Znak niza:a
Znak stoga:A
Trenutno stanje:q0
Stog:KA
Prijelaz: d(q0, a, A) ==> d(q1 A)
Stog:KA

Ulazni niz:)
Znak niza:)
Znak stoga:A
Trenutno stanje:q1
Stog:KA
Prijelaz: d(q1, ), A) ==> d(q1 |)
Stog:K

Ulazni niz:
Znak niza:|
Znak stoga:K
Trenutno stanje:q1
Stog:K
Prijelaz: d(q1, |, K) ==> d(q1 |)
Stog:(prazan stog)

Ulazni niz:
Znak niza:|
Znak stoga:
Trenutno stanje:q1
Stog:(prazan stog)
Niz se prihvaca
```

Ilustracija 5: Rezultat simulacije

Primjer prihvatanja stanjem:

Simulirani automat M_1 ima slijedeći oblik:

$M_2 = (\{q_1, q_2, q_3\}, \{0, 1, 2\}, \{N, J, K\}, \delta, q_0, K, \{q_3\})$ uz funkcije prijelaza :

$\delta(q_1, 0, K) = (q_1, NK)$

$\delta(q_1, 0, N) = (q_1, NN)$

$\delta(q_1, 0, J) = (q_1, NJ)$

$\delta(q_1, 1, K) = (q_1, JK)$

$\delta(q_1, 1, N) = (q_1, JN)$

$\delta(q_1, 1, J) = (q_1, JJ)$

$\delta(q_1, 2, K) = (q_2, K)$

$\delta(q_1, 2, N) = (q_2, N)$

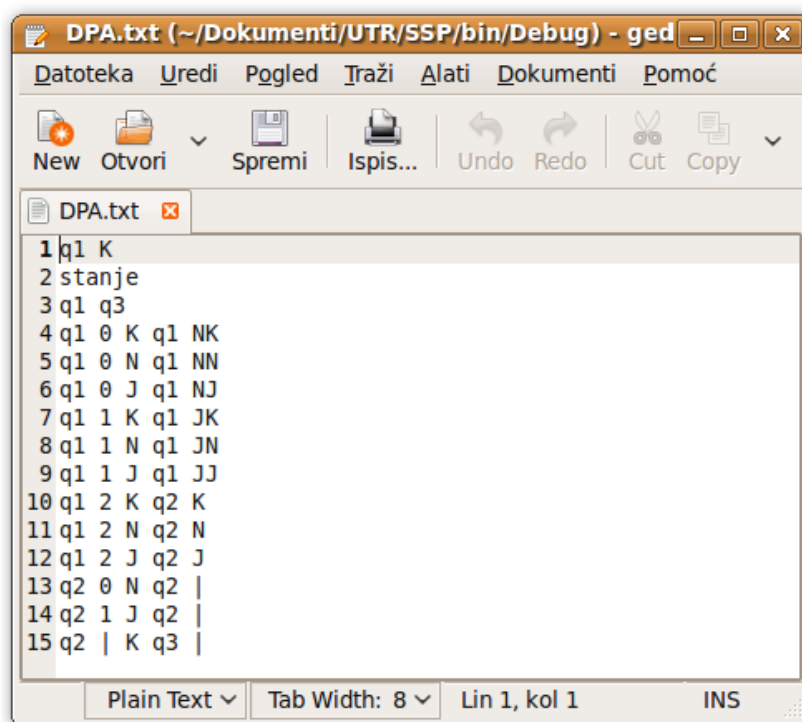
$\delta(q_1, 2, J) = (q_2, J)$

$\delta(q_2, 0, N) = (q_2, \varepsilon)$

$\delta(q_2, 1, J) = (q_2, \varepsilon)$

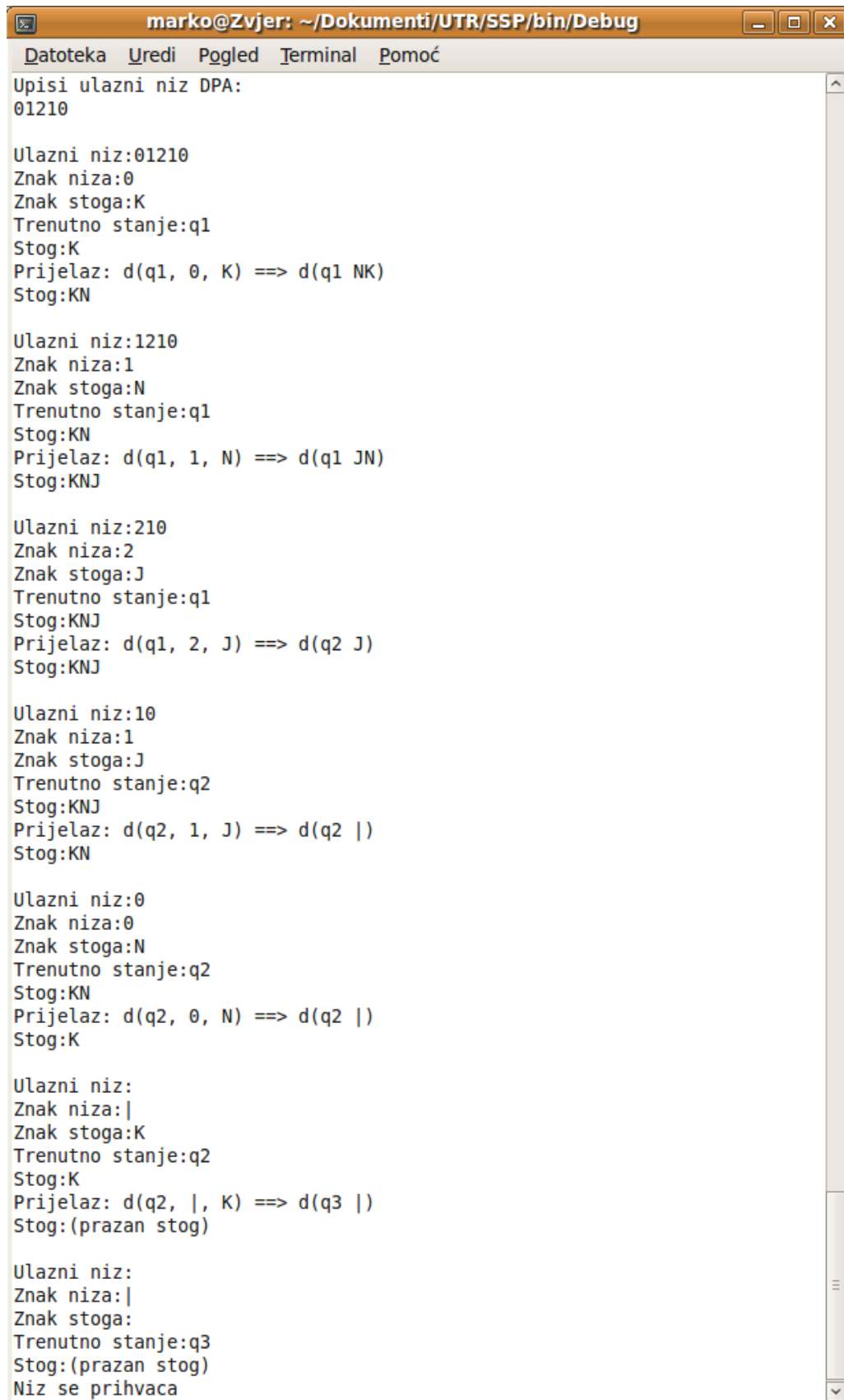
$\delta(q_2, \varepsilon, K) = (q_3, \varepsilon)$

Ulazna datoteka automata M_2 izgleda:



Ilustracija 6: Ulazna datoteka

Za zadani niz "01210" ispis simulatora determinističkog potisnog automata ima oblik:



```
marko@Zvjer: ~/Dokumenti/UTR/SSP/bin/Debug
Datoteka Uredi Pogled Terminal Pomoć
Upisi ulazni niz DPA:
01210

Ulazni niz:01210
Znak niza:0
Znak stoga:K
Trenutno stanje:q1
Stog:K
Prijelaz: d(q1, 0, K) ==> d(q1 NK)
Stog:KN

Ulazni niz:1210
Znak niza:1
Znak stoga:N
Trenutno stanje:q1
Stog:KN
Prijelaz: d(q1, 1, N) ==> d(q1 JN)
Stog:KNJ

Ulazni niz:210
Znak niza:2
Znak stoga:J
Trenutno stanje:q1
Stog:KNJ
Prijelaz: d(q1, 2, J) ==> d(q2 J)
Stog:KNJ

Ulazni niz:10
Znak niza:1
Znak stoga:J
Trenutno stanje:q2
Stog:KNJ
Prijelaz: d(q2, 1, J) ==> d(q2 |)
Stog:KN

Ulazni niz:0
Znak niza:0
Znak stoga:N
Trenutno stanje:q2
Stog:KN
Prijelaz: d(q2, 0, N) ==> d(q2 |)
Stog:K

Ulazni niz:
Znak niza:|
Znak stoga:K
Trenutno stanje:q2
Stog:K
Prijelaz: d(q2, |, K) ==> d(q3 |)
Stog:(prazan stog)

Ulazni niz:
Znak niza:|
Znak stoga:
Trenutno stanje:q3
Stog:(prazan stog)
Niz se prihvaca
```

Ilustracija 7: Rezultat simulacije

Prevođenje i pokretanje:

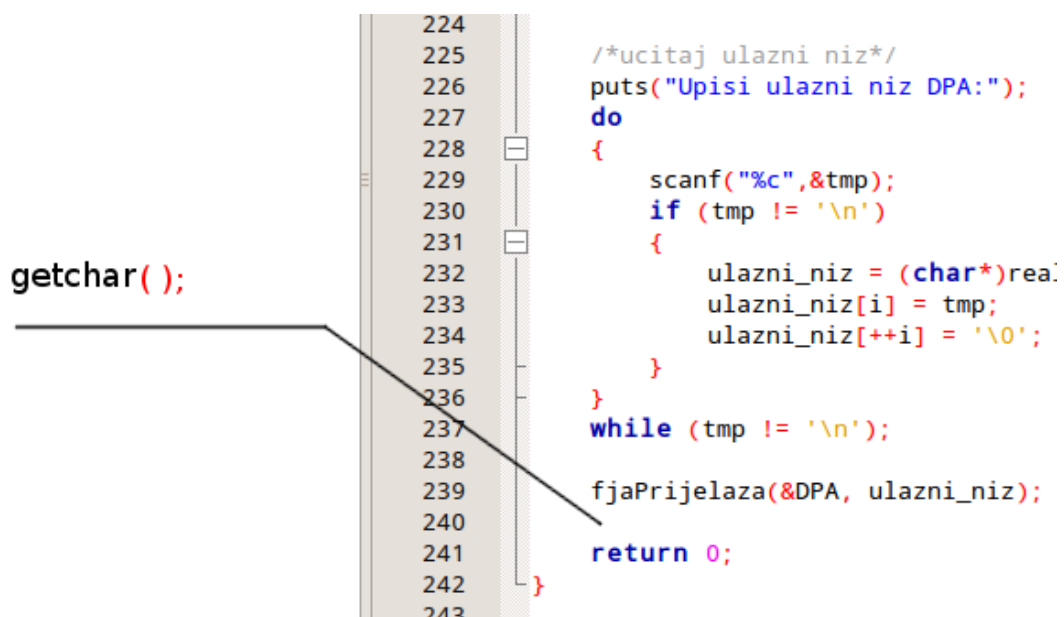
Prevođenje u strojni kod se vrši u jednoj od programske okoline poput MS Visual Studio, CodeBlocks ili DevC++, a može se i iz konzole ovisno o vrsti operacijskog sustava.

Za pokretanje potrebno je imati datoteku "DPA.txt" pokraj izvršnog programa kako bi izvršni program mogao učitati postavke PA kojeg želimo simulirati.

Ako se izvršni program ne pokreće iz neke programske okoline, potrebno je dodati pred kraj izvornog koda dodati slijedeću liniju

getchar();

kako bi se mogli očitati rezultati simulacije. Dodavanje ove linije može se izbjeći pokretanjem izvršnog programa iz konzole (na Windowsima "cmd", a na Unixoidima "bash").



```
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243

/*ucitaj ulazni niz*/
puts("Upisi ulazni niz DPA:");
do
{
    scanf("%c",&tmp);
    if (tmp != '\n')
    {
        ulazni_niz = (char*)realloc(ulazni_niz, (i+1)*sizeof(char));
        ulazni_niz[i] = tmp;
        ulazni_niz[++i] = '\0';
    }
} while (tmp != '\n');
fjaPrijelaza(&DPA, ulazni_niz);
return 0;
}
```

getchar();

Zaključak

U ovom seminarskom radu napravio sam simulator univerzalnog determinističkog potisnog automata, te je u gornjem tekstu opisan način pokretanja simulatora. Prikazao sam i osnovnu ideju rada simulatora kroz dva primjera.

Makar je algoritam ovog simulatora veoma jednostavan, imao sam problema pri realizaciji tog algoritma. Jedan od glavnih uzroka tog problema je što sam morao veoma paziti na način korištenja i alokacije memorije, u suprotnom su se pojavljivale različite slučajne pogreške prilikom pokretanja programa koje nije bilo lako otkriti. Ovaj simulator mogao bi se jednostavnije ostvariti u nekom višem objektnom programskom jeziku.