Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Lea Suć

Druga domaća zadaća iz predmeta "Uvod u teoriju računarstva"

Zadatak broj 3004

Zagreb, lipanj 2010.

Druga domaća zadaća iz predmeta “Uvod u teoriju računarstva”

Student: Lea Suć

Matični broj studenta: 0036444175

Zadatak broj 3004: Implementirati na računalu generator ε-NKA, odnosno program koji iz zadanih regularnih izraza (zadanog regularnog izraza) gradi tablicu odgovarajućeg ε-NKA prema algoritmu datom na predavanjima (u udžbeniku). NAPOMENA: obratiti pažnju na prioritet operatora koji se koriste u regularnim izrazima pri izboru podautomata i redoslijeda kojim se nadograđuju u kompletan automat.

1. Uvod

1.1. Regularni jezici

Definicija regularnih jezika zasniva se na konačnom automatu: jezik jest regularan ako i samo ako postoji konačni automat koji ga prihvaća. Time je definirana istovjetnost konačnih automata i regularnih jezika: za bilo koji regularni jezik moguće je izgraditi konačni automat koji ga prihvaća, i obrnuto, bilo koji konačni automat prihvaća jedan od regularnih jezika. Regularni jezici su najjednostavniji u smislu da je za njihovo prihvaćanje moguće izgraditi najjednostavniji automat: konačni automat.

1.1.1. Nedeterministički konačni automat s ε-prijelazima (ε-NKA)

ε-NKA može promijeniti stanje a da ne pročita niti jedan ulazni znak. Formalno se zadaje uređenom petorkom:

$$\boldsymbol{\varepsilon\text{-}NKA = (Q, \Sigma, , \delta, q_0, F)}$$

gdje je :

$\boldsymbol{Q}$	- konačan skup stanja;
$\boldsymbol{\Sigma}$	- konačan skup ulaznih znakova ;
$\boldsymbol{\delta}$	- funkcija prijelaza $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$;
$\boldsymbol{q_0 \in Q}$	- početno stanje;
$\boldsymbol{F \subseteq Q}$	- skup prihvatljivih stanja;

Funkcija prijelaza $\delta(q,a)$ određuje skup stanja P. U skupu stanja P su sva ona stanja p za koja postoji prijelaz iz stanja q. Oznaka a jest ε ili neki znak iz skupa ulaznih znakova Σ.

1.1.2. Regularni izrazi

Regularni izrazi koriste se za opis regularnih jezika. Regularni jezik moguće je opisati regularnim izrazima. Dva su osnovna razloga uporabe regularnih izraza:

1) Ako je jezik moguće opisati regularnim izrazima, onda je on regularan. Neka $L(r)$ označava jezik definiran regularnim izrazima r. Za bilo koji jezik $L(r)$ zadan regularnim izrazima r moguće je izgraditi DKA M za koji vrijedi $L(M)= L(r)$.
2) Izgrađen je algoritam pretvorbe regularnih izraza u ε -NKA.

Neka je Σ abeceda. Regularni izrazi definiraju se rekurzivno nad abecedom Σ. Uz rekurzivna pravila navedeni su i jezici određeni tim pravilima. Rekurzivna pravila regularnih izraza su:

1) Ø jest regularni izraz i označava jezik $L(Ø)=\{\}$.
2) ε jest regularni izraz i označava jezik $L(\varepsilon)=\{\varepsilon\}$.
3) $\forall a \in \Sigma$, a jest regularni izraz i označava jezik $L(a)=\{a\}$. Istom oznakom a označeni su znak abecede Σ, regularni izraz i niz jedinične duljine koji je element jezik $L(a)$.
4) Ako su r i s regularni izrazi koji označavaju jezike $L(r)$ i L(s), onda:
    a) $(r)+(s)$ jest regularni izraz koji označava jezik $L((r)+(s))=L(r)\cup L(s)$. Jezik $L((r)+(s))$ nastaje unijom jezika $L(r)$ i $L(s)$. Često se koristi i oznaka $(r)|(s)$. Oznaka $(r)|(s)$ koristi se za definiranje aritmetičkih izraza kako bi se ona razlikovala od oznake aritmetičkog operatora zbrajanja +.
    b) $(r)(s)$ jest regularni izraz koji označava jezik $L(r)(s))=L(r)L(s)$. Jezik $L((r)(s))$ nastaje nadovezivanjem jezika $L(r)$ i $L(s)$.
    c) $(r)^*$ jest regularni izraz koji označava jezik $L((r)^*)=L(r)^*$. Jezik $L((r)^*)$ nastaje primjenom Kleeneovog operatora nad jezikom $L(r)$.

Pravila asocijativnosti i prednosti operatora definiraju se na sljedeći način:

1) Unarni operator * jest lijevo asocijativan i najveće je prednosti.
2) Operator nadovezivanja jest lijevo asocijativan i veće je prednosti od operatora +.
3) Operator + jest lijevo asocijativan i najmanje je prednosti.

2. Ostvarenje

2.1. Algoritam izgradnje ε-NKA na temelju zadanih regularnih izraza

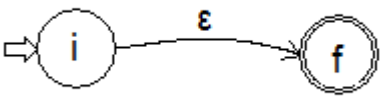
Za ostvarenje rješenja problema koristi se sljedeći algoritam:

Za bilo koji regularni izraz *r* moguće je izgraditi ε-NKA *M* tako da vrijedi $L(M)=L(r)$:

*p*1) Za regularni izraz Ø koji definira jezik $L(Ø)=\{\}$ konstruira se ε-NKA $M=(\{i,f\}, \Sigma, \{\}, i, \{f\})$. Početno stanje jest *i*, a prihvatljivo stanje jest $f \in F$. Dijagram stanja izgrađenog ε-NKA je:

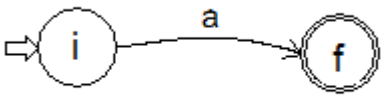
Za bilo koji $b \in (\Sigma \cup \{\varepsilon\})$, skup $\delta(i, b)$ jest prazan skup. Ne postoji niti jedan prijelaz u prihvatljivo stanje f. Budući da je početno stanje *i* neprihvatljivo, ε-NKA *M* ne prihvaća niti jedan niz, uključujući i prazni niz ε.

*p*2) Za regularni izraz ε koji definira jezik $L(\varepsilon)=\{\varepsilon\}$ konstruira se ε-NKA $M==(\{i,f\}, \Sigma, \{\delta(i, \varepsilon)=f\}, i, \{f\})$. Početno stanje jest *i*, a prihvatljivo stanje jest $f \in F$. Dijagram stanja izgrađenog ε-NKA je:



Za bilo koji $b \in \Sigma$, skup $\delta(f, b)$ jest prazan skup. Prijelaz $\delta(i, \varepsilon)=\{f\}$ omogućuje prihvaćanje praznog niza. ε-NKA *M* prihvaća isključivo prazni niz ε.

*p*3) Za regularni izraz *a* koji definira jezik $L(a)=\{a\}$ konstruira se ε-NKA $M==(\{i,f\}, \Sigma, \{\delta(i, a)=f\}, i, \{f\})$. Početno stanje jest *i*, a prihvatljivo stanje jest $f \in F$. Dijagram stanja izgrađenog ε-NKA je:



Za bilo koji $b \in (\Sigma \cup \{\varepsilon\})$ za koji vrijedi b≠a, skup $\delta(f, b)$ jest prazan skup. Prijelaz $\delta(i, a)=\{f\}$ omogućuje prihvaćanje niza *a*. Nadalje, ε-NKA *M* prihvaća isključivo niz *a*. Ne prihvaća se ni prazni niz ε.

*p*4) Za regularni izraz r_1+r_2 koji definira jezik $L(r_1+r_2)=L(r_1) \cup L(r_2)$ konstruira se ε-NKA *M* na sljedeći način: pretpostavimo da su prethodno izgrađeni ε-NKA $M_1=(Q_1, \Sigma_1, \delta_1, i_1, \{f_1\})$ i $M_2=(Q_2, \Sigma_2, \delta_2, i_2, \{f_2\})$ takvi da vrijedi $L(M_1)=L(r_1)$ i $L(M_2)=L(r_2)$ i da nema prijelaza iz stanja f_1 i f_2 niti za jedan ulazni znak (tj. $\delta_1(f_1, a)=\emptyset, \forall a \in (\Sigma_1 \cup \{\varepsilon\})$ i $\delta_2(f_2, b)=\emptyset, \forall b \in (\Sigma_2 \cup \{\varepsilon\})$). Promjenom imena stanja u Q_1 i Q_2 postiže se da je $Q_1 \cap Q_2=\{\}$. Konstruira se ε-NKA $M=(Q_1 \cup Q_2 \cup \{i,f\}, \Sigma_1 \cup \Sigma_2, \delta, i, \{f\})$. Novo početno stanje jest *i*, a novo prihvatljivo stanje jest *f*. Stanja i_1 i i_2 nisu više početna stanja, te stanja f_1 i f_2 nisu više prihvatljiva. Funkcija δ određuje se na sljedeći način :

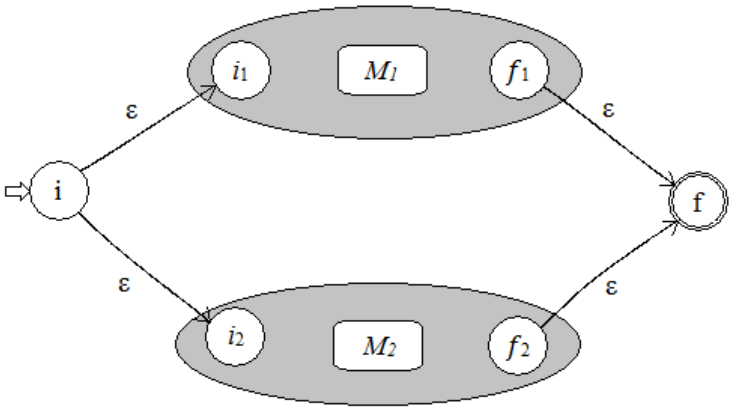
a) $\delta(i, \varepsilon)=\{i_1, i_2\}$,

b) $\delta(q, a)=\delta_1(q, a), \forall q \in (Q_1 \setminus \{f_1\})$ i $\forall a \in (\Sigma_1 \cup \{\varepsilon\})$,

c) $\delta(q, b)=\delta_2(q, b), \forall q \in (Q_2 \setminus \{f_2\})$ i $\forall b \in (\Sigma_2 \cup \{\varepsilon\})$,

d) $\delta(f_1, \varepsilon)= \delta(f_2, \varepsilon)=\{f\}$

Dijagram stanja izgrađenog ε-NKA M koji prihvaća jezik $L(M)=L(r_1+r_2)$ je:



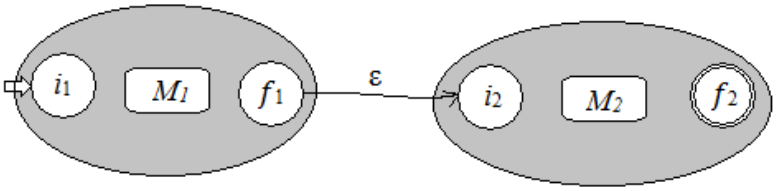
*p*5) Za regularni izraz r_1r_2 koji definira jezik $L(r_1r_2)=L(r_1)L(r_2)$ konstruira se ε-NKA M na sljedeći način: pretpostavimo da su prethodno izgrađeni ε-NKA $M_1=(Q_1, \Sigma_1, \delta_1, i_1, \{f_1\})$ i $M_2=(Q_2, \Sigma_2, \delta_2, i_2, \{f_2\})$ takvi da vrijedi $L(M_1)=L(r_1)$ i $L(M_2)=L(r_2)$ i da nema prijelaza iz stanja f_1 i f_2 niti za jedan ulazni znak (tj. $\delta_1(f_1, a)=\emptyset$, $\forall a\in(\Sigma_1\cup\{\varepsilon\})$ i $\delta_2(f_2, b)=\emptyset$, $\forall b\in(\Sigma_2\cup\{\varepsilon\})$). Promjenom imena stanja u Q_1 i Q_2 postiže se da je $Q_1\cap Q_2=\{\}$. Konstruira se ε-NKA $M=(Q_1 \cup Q_2, \Sigma_1\cup\Sigma_2, \delta, i_1, \{f_2\})$. Novo početno stanje jest i_1, a novo prihvatljivo stanje jest f_2. Stanje i_2 nije više početno stanje, te stanje f_1 nije više prihvatljivo. Funkcija δ određuje se na sljedeći način :

a) $\delta(q, a)=\delta_1(q, a)$, $\forall q\in(Q_1\setminus\{f_1\})$ i $\forall a\in(\Sigma_1\cup\{\varepsilon\})$,

b) $\delta(q, b)=\delta_2(q, b)$, $\forall q\in Q_2$ i $\forall b\in(\Sigma_2\cup\{\varepsilon\})$,

c) $\delta(f_1, \varepsilon)=\{i_2\}$

Dijagram stanja izgrađenog ε-NKA M koji prihvaća jezik $L(M)=L(r_1r_2)$ je:

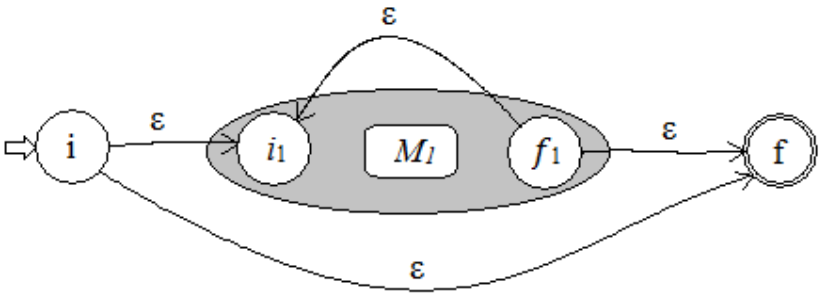


*p*6) Za regularni izraz r_1* koji definira jezik $L(r_1^*)=L(r_1)^*$ konstruira se ε-NKA M na sljedeći način: pretpostavimo da je prethodno izgrađen ε-NKA $M_1=(Q_1, \Sigma_1, \delta_1, i_1, \{f_1\})$ za koji vrijedi $L(M_1)=L(r_1)$ i nema prijelaza iz stanja f_1 niti za jedan ulazni znak (tj. $\delta_1(f_1, a)=\emptyset$, $\forall a \in (\Sigma_1 \cup \{\varepsilon\})$). Konstruira se ε-NKA $M=(Q_1 \cup \{i,f\}, \Sigma_1, \delta, i, \{f\})$. Novo početno stanje jest i, a novo prihvatljivo stanje jest f. Stanje i_1 nije više početno stanje, te stanje f_1 nije više prihvatljivo. Funkcija δ određuje se na sljedeći način :

a) $\delta(i, \varepsilon)= \delta(f_1, \varepsilon)= \{i_1, f\}$,

b) $\delta(q, a)=\delta_1(q, a)$, $\forall q \in (Q_1 \setminus \{f_1\})$ i $\forall a \in (\Sigma_1 \cup \{\varepsilon\})$

Dijagram stanja izgrađenog ε-NKA M koji prihvaća jezik $L(M)=L(r_1^*)$ je:



*p*7) Budući da je $L((r))=L(r)$, za ε-NKA M regularnog izraza (r) uzima se ε-NKA M_1 regularnog izraza r, jer je $L(M_1)=L(r)=L((r))$.

2.2. Programsko ostvarenje

Priloženi generator ε-NKA ostvaren je u programskom jeziku C++. Generator očekuje ulaznu datoteku „UlazG.txt" u istom direktoriju u kojem se i sam generator nalazi. U ulaznoj datoteci se nalazi jedan ulazni niz koji predstavlja regularni izraz za koji ćemo izgraditi ε-NKA. Ulaznu abecedu Σ ε-NKA čine svi znakovi ASCII tablice koji se mogu ispisati, s tim da se ulazni znakovi '+', '*', '(' i ')' tumače kao operatori, a znak 'E' predstavlja ε (prazan niz). Pretpostavlja se da je regularni izraz ispravno zadan, a moguće je i ne upisati ništa u ulaznu datoteku, što će se

tumačiti kao Ø. Kompletan programski kod ostvarenja i primjer ulazne datoteke nalaze se na mediju priloženom uz rad. Rezultat izvođenja programa vidljiv je u komandnom prozoru.

Sam generator može se podijeliti u 3 cjeline:

1) Glavni dio programa u kojem se učitava izraz iz ulazne datoteke, poziva rekurzivni potprogram za obradu regularnog izraza, te poziva funkcija *ispis* koja ispisuje konačni ε-NKA.

2) Rekurzivni potprogram *potp* koji poziva sebe i druge potprograme, u kojem se svakim korakom smanjuje veličina regularnog izraza, te kad dođe do jednostavne cjeline za koju je moguće izgraditi podautomat, poziva odgovarajući potprogram koji će stvoriti automat za taj jednostavni oblik regularnog izraza. U potprogramu *potp*, dakle, provjerava se postoje li odvojive cjeline, bilo da su razdvojene plusom, zagradama, ili je u pitanju konkatenacija, te se prema tome razdvoji prvi dio za koji se stvara automat i ostatak niza. Kasnije se podautomati spajaju u konačni automat ε-NKA koji odgovara zadanom regularnom izrazu, te potprogram *potp* vraća gotov automat. U potprogramu *potp* pazi se na prioritete operatora, pa se tako najprije pozivaju funkcije koje predstavljaju operatore najmanjeg prioriteta (najprije *zbroj*, zatim *konkaten*, zatim *klin*) kako bi se tijekom rekurzivnih poziva operatori najvećeg prioriteta prvi vremenski izveli.

3) Ostali potprogrami u kojima se rješavaju situacije kada se dođe do jednostavnog oblika regularnog izraza za koji se odmah može stvoriti gotov ε-NKA. Potprogrami za niz veličine jedan (*znak*, koji gradi automat za jedan ulazni znak $\neq \varepsilon$ i potprogram *epsilon*, koji gradi automat za ulaz ε), potprogrami za zbroj (*zbroj*), nadovezivanje (*konkaten*) i Kleeneov operator (*klin*) napravljeni su prema zadanom algoritmu. Funkcija *ispis* ispisuje rezultat izvođenja programa za svaki izgrađeni podautomat.

2.3. Primjer rada programa

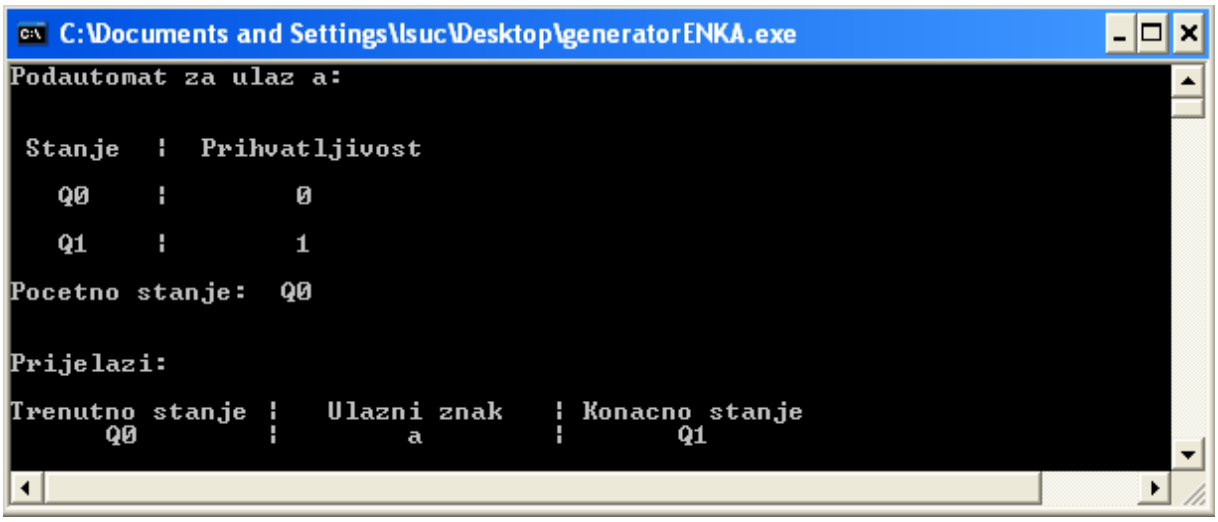
Za konkretni primjer regularnog izraza r=a*b+b program se odvija na sljedeći način:

1) Regularni izraz r razdvoji se na izraze $r=r_1+r_2$, gdje su r_1=a*b i r_2=b, te se poziva funkcija *zbroj* koja prima gotove automate za r_1 i r_2.

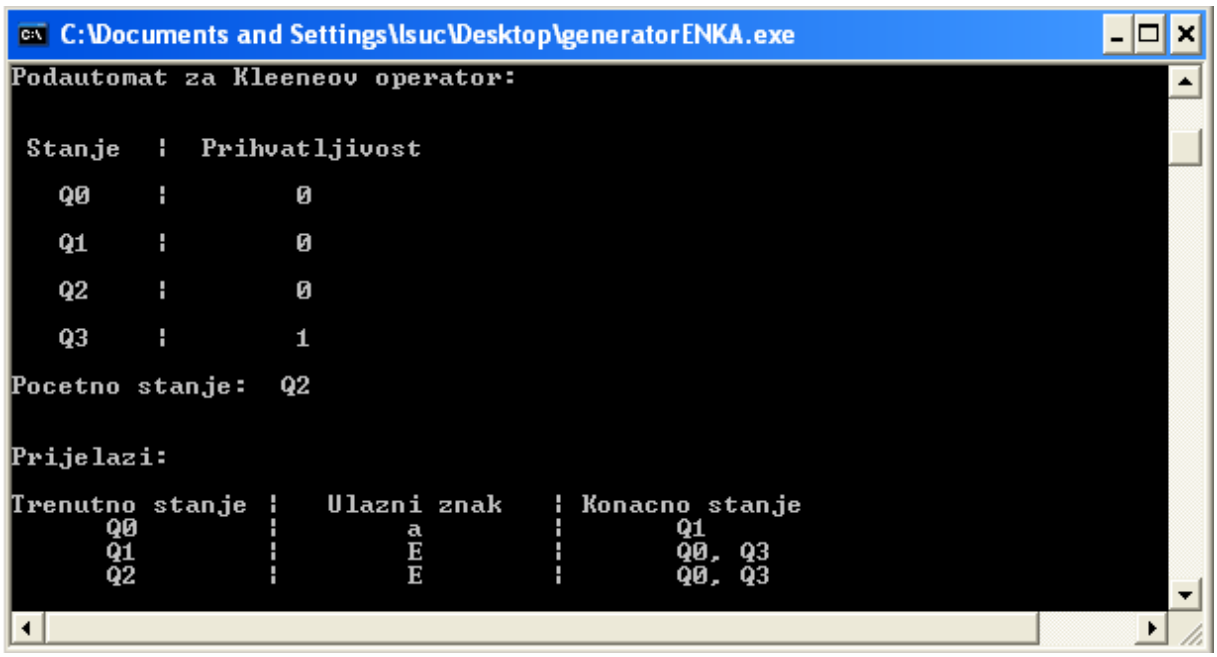
2) Regularni izraz r_1=a*b rastavi se na izraze $r_1= r_{11}r_{12}$, gdje su r_{11}=a* i r_{12}=b i poziva se funkcija *konkaten* koja prima gotove automate za r_{11} i r_{12}.

3) Regularni izraz r_{11}=a* rastavi se na izraze r_{11}= r_{111}*, gdje je r_{111} =a i poziva se funkcija *klin* koja prima gotov automat za r_{111} .

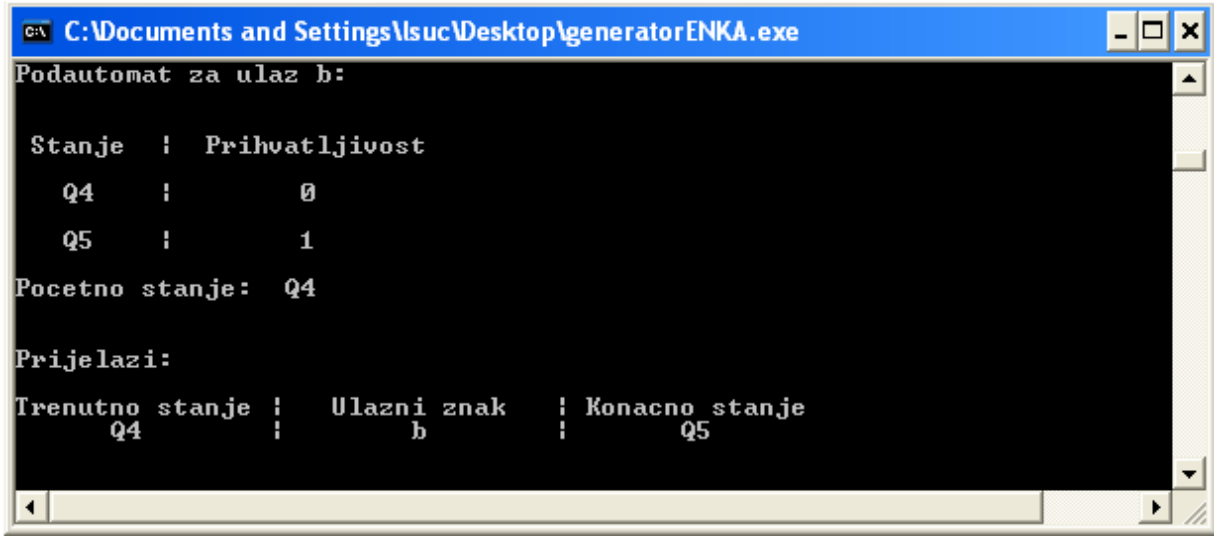
4) Za r_{111} =a poziva se funkcija *znak* koja konstruira ε-NKA prema pravilu (*p*3), koji ima sljedeća stanja i prijelaze:



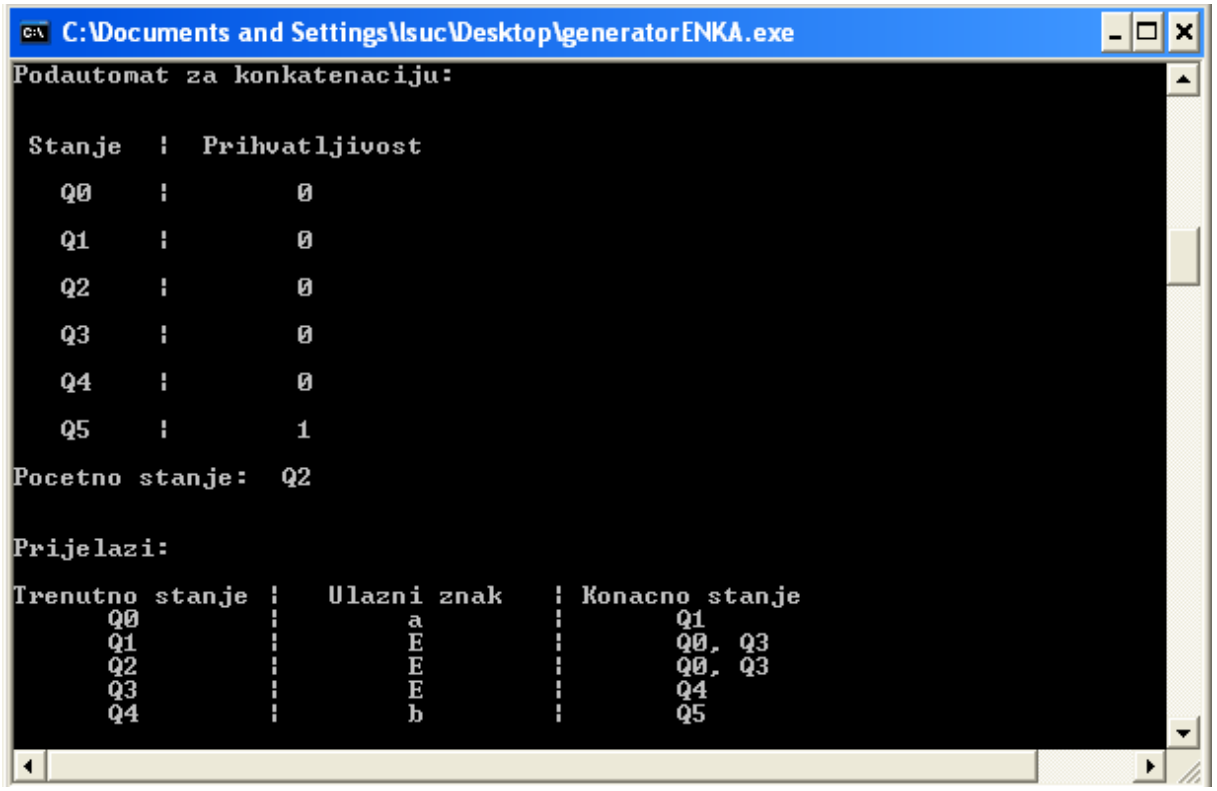
5) Za r_{11} konstruira se ε-NKA koji nastaje izvršavanjem funkcije *klin* koja koristi pravilo (*p*6), te ima sljedeća stanja i prijelaze:



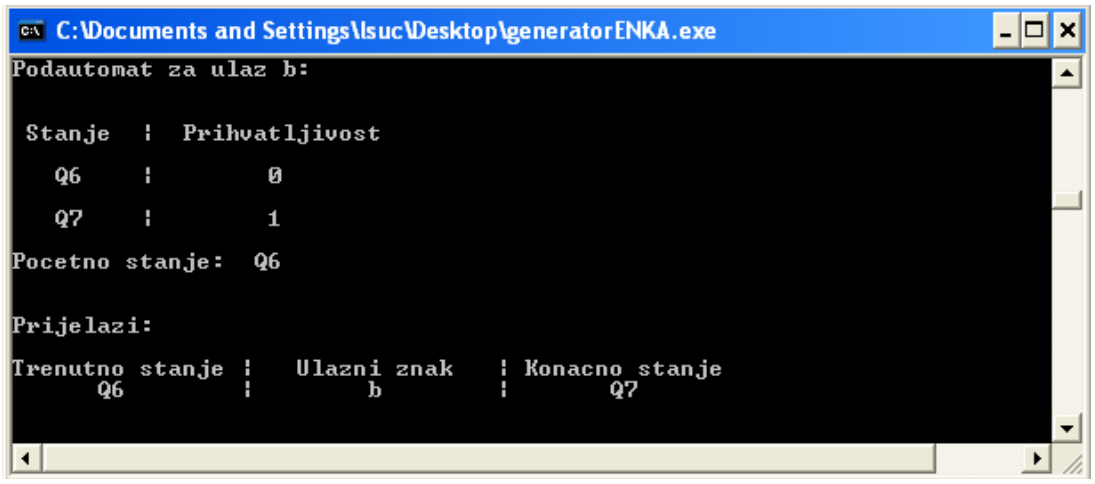
6) Za r_{12}=b poziva se funkcija znak koja konstruira ε-NKA prema pravilu ($p3$), koji ima sljedeća stanja i prijelaze:



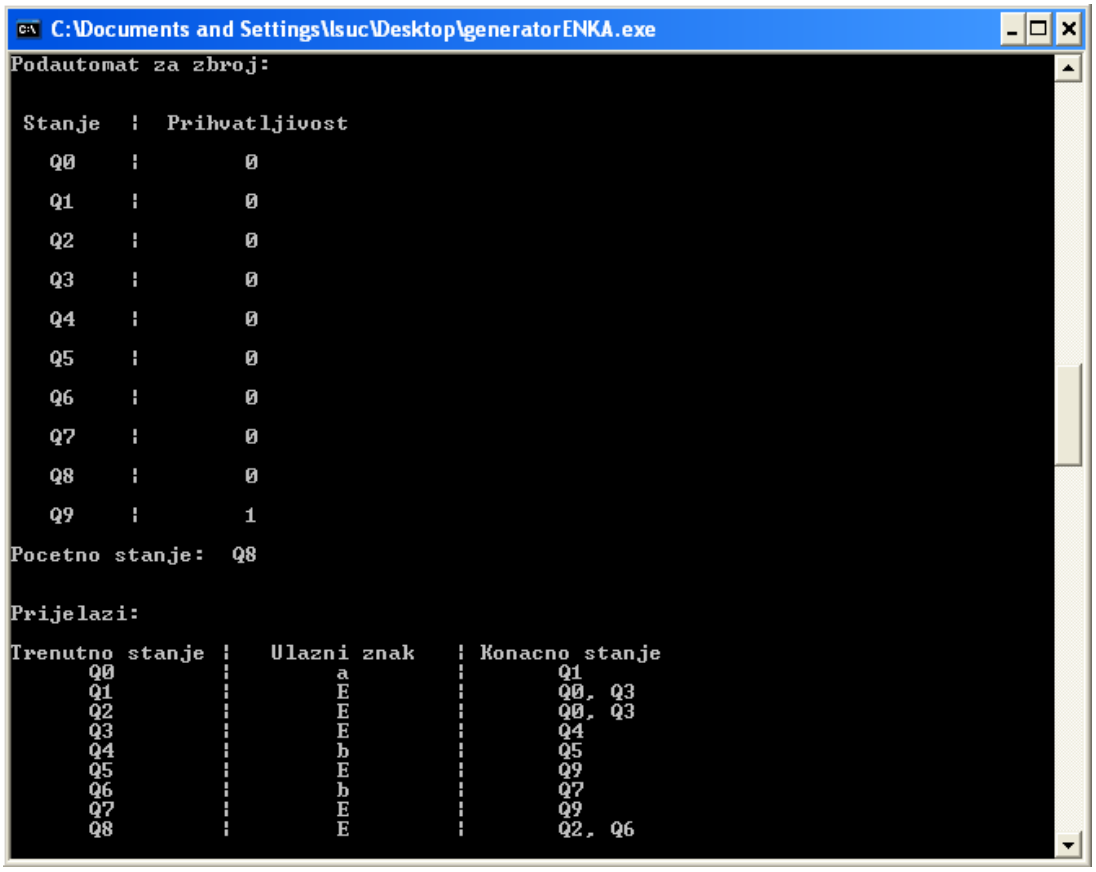
7) Za $r_1 = r_{11}r_{12}$ konstruira se ε-NKA koji nastaje izvršavanjem funkcije *konkaten* koja koristi pravilo ($p5$), te ima sljedeća stanja i prijelaze:



8) Za r_2=b poziva se funkcija znak koja konstruira ε-NKA prema pravilu (p3), koji ima sljedeća stanja i prijelaze:



9) Za $r=r_1+r_2$ konstruira se ε-NKA koji nastaje izvršavanjem funkcije *zbroj* koja koristi pravilo (p5), te ima sljedeća stanja i prijelaze:



Automat nastao u ovom koraku upravo je traženi ε-NKA kojeg je trebalo izgraditi iz zadanog regularnog izraza.

3. Zaključak

Konstrukcija ε-NKA na temelju zadanih regularnih izraza pomoću opisanog algoritma idejno je vrlo jednostavna, ali programski nije tako lako ostvariva zbog nužnosti razlikovanja prioriteta operatora koji se koriste u regularnim izrazima. Konkretno, u programu je potrebno paziti na prioritete prilikom izrade podautomata i pri odabiru redoslijeda kojim se podautomati ugrađuju u kompletan automat. Problem je riješen pažljivim odabirom redoslijeda poziva funkcija podautomata u potprogramu koji se rekurzivno poziva.

Za bilo koji ε-NKA M izgrađen ovim algoritmom karakteristično je da broj stanja nikad nije veći od $2|r|$, gdje je $|r|$ broj znakova u regularnom izrazu r. Naime, u pojedinim koracima konstrukcije ε-NKA ne stvara se više od dva nova stanja. Isto tako, ε-NKA = $(Q, \Sigma, \delta, i, \{f\})$ ima samo jedno završno stanje f. Završno stanje nema niti jedne usmjerene grane iz stanja f u neko drugo stanje. Skup $\delta(q, a)$ sadrži najviše jedno stanje za ulazni znak a iz skupa Σ, dok skup $\delta(q, \varepsilon)$ sadrži najviše dva stanja. Ima li čvor dvije izlazne grane, obje grane označene su ε prijelazima. Sva ova svojstva znatno su olakšala implementaciju generatora ε-NKA, a svako pojedino svojstvo vidi se u i rezultatima izvršavanja programa.