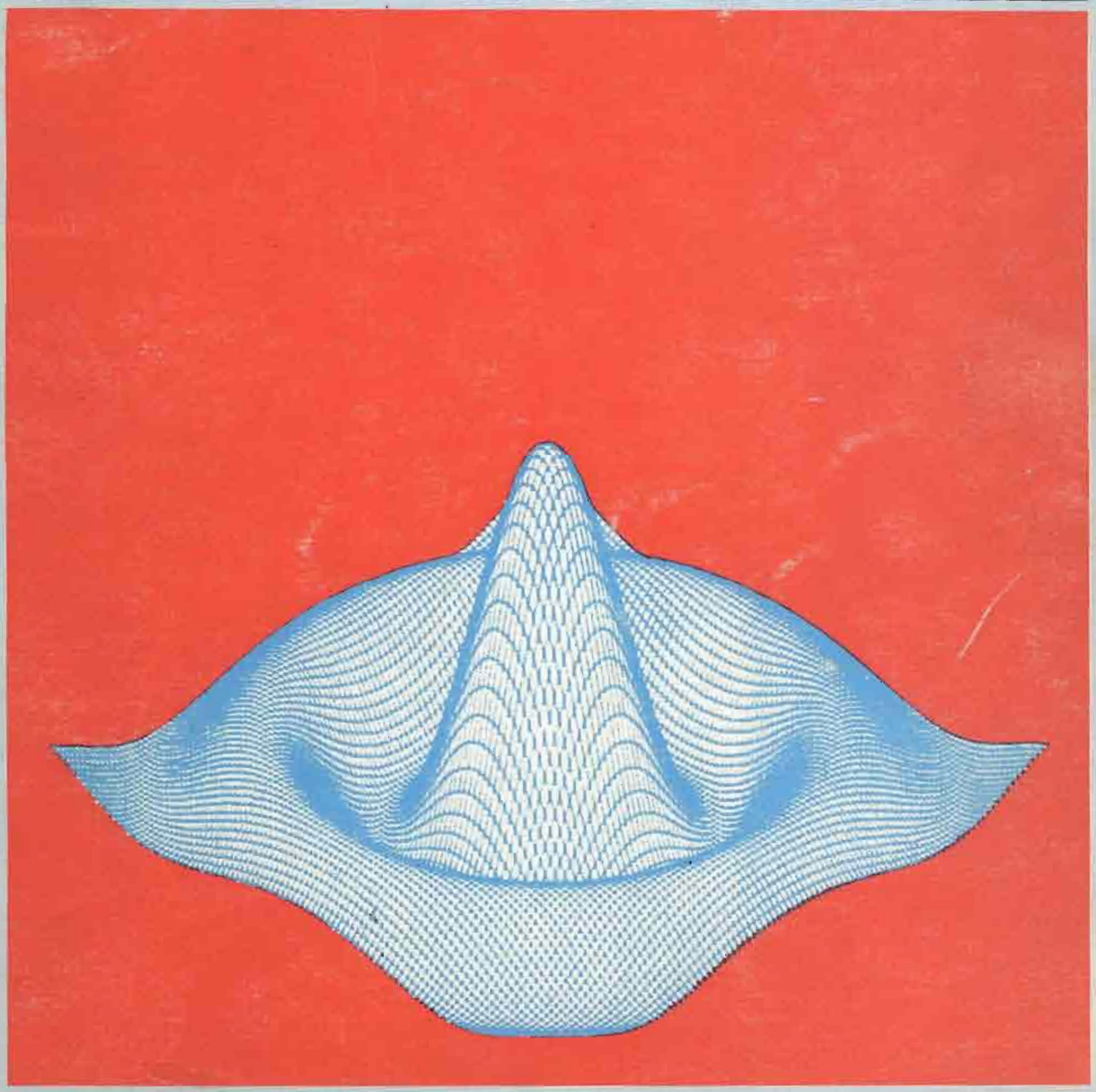
YU ISSN 0353-0469
UDK 608.3:371

Računarstvo

u nauci i obrazovanju

VOL I. BROJ 2/4 OKTOBAR 1987. NOVA KNJIGA BEOGRAD



Računarstvo
u nauci i obrazovanju

VOL. I BROJ 2/4 OKTOBAR 1987.

IZDAVAČ:
IRO "Nova Knjiga", Beograd, Ada Ciganlija 6, Poštanski pregradak 2098. Direkcija: 557-571, 556-733. Redakcija: 544-600, 541-062.

DIREKTOR IRO "NOVA KNJIGA":
Slobodan Filimonović

V.D. GLAVNOG I ODGOVORNOG UREDNIKA ČASOPISA:
prof.dr Nedeljko Parezanović

IZDAVAČKI SAVET:
prof.dr Zvonimir Damjanović, prof.dr Miodrag Sibinović, dr Miloje Rakočević, prof.dr Ramiz Abduli, general-major, Gradislav Milenović, mr Milan Zaklan, pukovnik, Slobodan Filimonović, prof.dr Nedeljko Parezanović, mr Veljko Spasić, Zona Stevanović

UREDJUJE REDAKCIJSKI KOLEGIJUM:
prof.dr Žarko Mijajlović, mr Milan Tuba, mr Dušan Vitas, mr Veljko Spasić, prof.dr Robert Lewis, dr Margaret Cox

UREDNIK:
Zona Stevanović

LIKOVNA OPREMA:
Nenad Lazović

ČASOPIS IZLAZI:
Četiri puta godišnje

CENA:
po jednom primerku u knjižarskoj mreži 4000 dinara, pretplatna cena 3000 dinara, godišnja pretplata – 9000 dinara (za inostranstvo dvostruko) Žiro račun: IRO "Nova knjiga" 60812-603-23329

ŠTAMPA:
"KULTURA", Bački Petrovac

Typeset in TEX, *Elektronika 011*, Beograd, Vojislava Ilića 64, tel 437-437

Sadržaj prethodnog broja

Sadržaj

UDK 681.32:62

Primena računara u optimalnom oblikovanju površinskih otkopa

Dušan Bratičević

Kratak sadržaj: *U radu se daje matematička formulacija jednog problema optimizacije u projektovanju površinskih otkopa. Ograničenja optimizacije izražena preko uslova stabilnosti formalizuju se preko tzv.* referentnih koordinata. *Time se omogućava potpuno približavanje matematičkog modela ograničenja, ograničenjima koja postoje u realnosti. Daje se jedan postupak približnog rešavanja problema optimizacije, koji se zasniva na dinamičkom programiranju i koji se odvija u vremenu koje je proporcionalno veličini modela. Takodje se daje i postupak za ocenu tačnosti. Na kraju, uvodi se i postupak redukcije modela. Iterativnim redukcijama mogu da se dalje poboljšavaju približna rešenja i u mnogim slučajevima tako se može doći i do samog optimuma.*

Ključne reči: *projektovanje pomoću računara, optimizacija, dinamičko programiranje, projektovanje u rudarstvu, površinski otkop*

1. Uvod

Rudna tela koja treba da se eksploatišu površinskim načinom otkopavanja, obično su prekrivena nekorisnim materijalom koji je potrebno prethodno ukloniti da bi se rudno telo "otkrilo". Taj materijal naziva se *otkrivka*. Da bi se, polazeći od neke konfiguracije površinskog otkopa, došlo do novih količina rude, potrebno je ukloniti i odredjene količine otkrivke. Ukoliko je količina rude veća, utoliko je veća i potrebna količina otkrivke. Izvesno proširenje otkopa ekonomski je opravdano samo ako je vrednost dobijene rude veća od troškova prouzrokovanih uklanjanjem otkrivke.

U vezi sa navedenim problemom, prirodno se postavlja pitanje utvrdjivanja optimalnog površinskog otkopa. Takav otkop bi imao osobinu da bi se daljim proširivanjem više utrošilo na uklanjanje otkrivke nego što bi se dobilo u vrednosti rude, i obratno — ako bi se uzeo bilo koji oblik otkopa koji je manji od optimalnog, on bi svakako imao manje troškove u uklanjanju otkrivke od optimalnog, ali bi ušteda u tim troškovima bila manja nego što je vrednost rude koja bi na taj način bila izgubljena.

Zadatak optimalnog oblikovanja površinskog otkopa zanima stručnjake iz rudarstva i računarstva još od prvih dana primene računara u rudarstvu — početkom šezdesetih godina. Jedan od prvih, a do danas i najznačajnijih radova iz te oblasti, objavili su Lersch i Grossmann 1965. godine [1]. U tom radu postavljeni su temelji za dve klase algoritama za optimalno oblikovanje površinskih otkopa - metodom grafova i metodom dinamičkog programiranja. Dok prva metoda uvek daje optimalno, druga daje samo približno rešenje. S druge strane, metoda grafova postavlja ozbiljne zahteve u pogledu korišćenja memorije i vremena rada centralnog procesora, dok je metoda dinamičkog programiranja atraktivna upravo zbog svoje efikasnosti i skromnih zahteva u korišćenju memorije.

Uopšte, podela metoda optimizacije na stroge i približne, već se može smatrati tradicionalnom, kao i neslaganja stručnjaka oko toga koje metode više vrede. Postupku optimizacije nikada se ne podvrgava realno ležište, već njegov model. Model predstavlja samo približnu sliku ležišta, pa traganje za stvarnim optimumom, bez obzira na cenu postupka, nije sasvim opravdano. S druge strane, približne metode mogu još uvek da daju rešenja koja su znatno bolja od onih koja mogu da daju inženjeri radeći na klasičan način. Ipak, značajan nedostatak približnih metoda koje su do sada korišćene je u tome što se ne dobija nikakva procena greške. Za dobijeno približno rešenje ne zna se koliko se razlikuje

od optimalnog, ni po vrednosti ni po položaju u prostoru.

U ovom članku dajemo formalnu postavku problema konstrukcije optimalnog površinskog otkopa, zatim opisujemo jedan postupak za približno rešavanje ovog zadatka i na kraju dajemo jedan iterativni postupak optimizacije. Prednost ovog postupka je u tome što se, za razliku od drugih približnih metoda, pored rešenja dobija i procena odstupanja od optimuma, kako u pogledu prostornog položaja, tako i u pogledu vrednosti.

Zbog ograničenosti prostora, izlaganje je skraćeno koliko god je bilo moguće, pri čemu se vodilo računa da to ne ide na štetu razumljivosti. Za sve detalje, čitalac se može obratiti na [2].

2. Formulacija problema

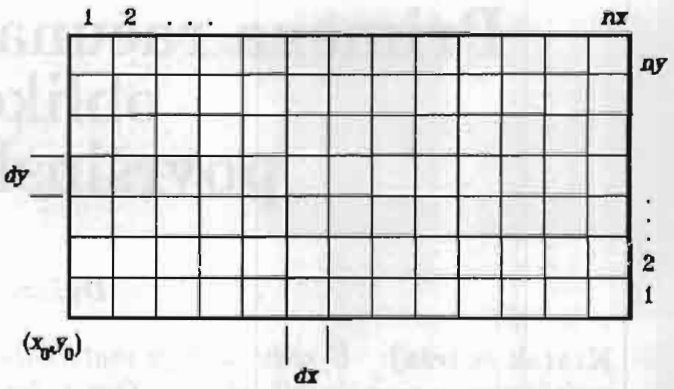
Uobičajeno je da se prostor u kojem se nalazi ležište i otkop podeli na konačan broj blokova. Zavisno od mineraloškog sastava bloka, troškova otkopavanja, transporta i prerade, može se izračunati njegova vrednost. Tako se dolazi do modela ležišta, koji se može formalizovati na sledeći način:

DEFINICIJA 1. Model ležišta L je konačan skup na kojem je definisana jedna realna funkcija f. Elementi ležišta su *blokovi*. Ako $b \in L$, onda je $f(b)$ *vrednost bloka* b.

Ovako formulisan model ne obuhvata mnoge aspekte realnog ležišta, koji bi bili od značaja u drugim problemima obrade na računaru. No, za potrebe optimalnog oblikovanja površinskog otkopa, ovako pojednostavljena formulacija sasvim je dovoljna.

Sledeća stvar koju je potrebno modelirati su kosine. Ekonomski bi bilo najpogodnije ako bi površinski otkop mogao da se kopa vertikalno, kao bunar. Medjutim, zbog sigurnosti, granica otkopa ne sme da ima nagib veći od nekog kritičnog. Taj kritični nagib zavisi od geomehaničkih osobina stena, dubine, prisustva podzemnih voda i slično. Kako se ti parametri menjaju kroz ležište, to se i maksimalno dozvoljeni ugao takodje menja i zavisi od položaja bloka u ležištu i od smera.

Za svaki blok b ležišta L odredjen je jedan podskup $K(b) \subseteq L$ blokova koji mora da se otkopaju pre bloka b, da bi se obezbedila stabilnost otkopa. Po pravilu, $K(b)$ podseća na kupu sa vrhom u b i dnom okrenutim naviše. Blokovi iz $K(b)$ koji su različiti od b prethode bloku b u bilo kom postupku otkopavanja koji poštuje uglove stabilnosti.



Slika 1. – Diskretizacija po x i y osi

Prema tome, stabilnost površinskog otkopa može se formalizovati na sledeći način.

DEFINICIJA 2. *Model uglova stabilnosti* ležišta L je jedna parcijalna relacija poretka medju blokovima. Skup $K(b)$ svih blokova koji u smislu ove relacije prethode bloku b zove se *konus stabilnosti* bloka b.

Sada se može definisati i otkop.

DEFINICIJA 3. *Površinski otkop* je skup $P \subseteq L$ koji ima osobinu da ako blok $b \in P$, onda je $K(b) \subseteq P$.

Drugim rečima, površinski otkop sadrži kompletne konuse stabilnosti svih svojih blokova.

Zadatak optimizacije može se formalno izkazati na sledeći način.

U zadatom ležištu naći površinski otkop čija je ukupna vrednost blokova maksimalna.

3. Konkretizacija modela ležišta i uglova stabilnosti

Za obradu na računaru potrebno je da se konkretnije formuliše kako model ležišta, tako i model uglova stabilnosti.

Diskretizacija ležišta obavlja se tako što se najpre vrši diskretizacija x i y koordinata. Projekcija dela trodimenzionog prostora koji se posmatra

ograničava se po x i y osi, a zatim se vrši podela na $nx \times ny$ elementarnih pravougaonika, kao što je prikazano na slici 1. Svakom pravougaoniku odgovara jedan stub ili vertikala u ležištu. Taj stub se deli horizontalnim ravnima na blokove. Dok su dimenzije blokova u pravcu x, odnosno y ose konstantne, podela po z osi ne mora da bude uniformna, već može da se prilagodi karakteristikama rudnih tela. Model ležišta može da ima nekoliko desetina hiljada do nekoliko stotina hiljada blokova.

Na ovom mestu nećemo ulaziti u detalje izračunavanja vrednosti pojedinih blokova. Značajno je, medjutim, da ćemo dalje posmatrati model ležišta kao skup *glavnih tačaka*, koje se nalaze u centrima osnova blokova (a ne u središtima blokova). Umesto funkcije f (koja predstavlja vrednosti pojedinačnih blokova), posmatraćemo funkciju F definisanu u glavnim tačkama. Pri tome je $F(t)$ ukupna vrednost blokova iznad glavne tačke t. Očigledno, F se lako izračunava iz f.

Sledećom definicijom uvodimo pojam granice.

DEFINICIJA 4. Skup glavnih tačaka koji na svakoj vertikali ima tačno jednu glavnu tačku nazivamo *granicom*. *Vrednost granice* G je

$$Vred(G) = \sum_{t \in G} F(t) .$$

Svaka granica deli skup blokova na dva podskupa — podskup blokova iznad i podskup blokova ispod granice. Očigledno, $Vred(G)$ je ukupna vrednost blokova iznad G.

Od posebnog interesa su granice površinskih otkopa, tj. granice koje zadovoljavaju uslove stabilnosti. Za uvodjenje stabilnosti koristićemo relaciju prethodjenja, koju ćemo ovde definisati pomoću tzv. referentnih koordinata.

DEFINICIJA 5. U svakom vertikalnom profilu π koji je paralelan nekoj od osa ili nekoj od dijagonala elementarnih pravougaonika, uvode se dve funkcije P_π i Q_π definisane u glavnim tačkama profila tako da zadovoljavaju sledeće uslove:

(a) U svakom profilu π i u svakoj vertikali funkcije P_π i Q_π rastu zajedno sa z koordinatom.

(b) Smatramo da su vertikale unutar profila π uredjene sleva udesno. Ako je tačka t_2 desno od tačke t_1 i $P_\pi(t_2) \geq P_\pi(t_1)$, onda je z koordinata tačke t_2 veća od z koordinate tačke t_1; ako je t_2 levo od t_1 i $Q_\pi(t_2) \geq Q_\pi(t_1)$, onda je z koordinata tačke t_2 veća od z koordinate tačke t_1.

Funkcije P_π i Q_π su *referentne koordinate* u profilu π.

Pomoću referentnih koordinata može se definisati relacija prethodjenja medju glavnim tačkama.

DEFINICIJA 6. (a) Ako su tačke t_1 i t_2 $(t_1 \neq t_2)$ na istom profilu π, tada t_2 *prethodi* tački t_1 ako i samo ako je

$$P_\pi(t_2) > P_\pi(t_1) \quad \text{i} \quad Q_\pi(t_2) > Q_\pi(t_1) .$$

(b) Relacija prethodjenja je tranzitivno i refleksivno zatvorenje relacije koja zadovoljava uslov (a).

Prema tome, relacija prethodjenja definisana je u okviru svakog profila neposredno preko referentnih koordinata, a medju tačkama koje nisu na istom profilu definisana je "preko posrednika" (što je smisao tranzitivnog zatvorenja): ako postoji tačka t_2 koja prethodi tački t_1 i kojoj prethodi tačka t_3, onda t_3 prethodi tački t_1. Uslovi definicije 5 dovoljni su da relacija uvedena definicijom 6 zaista bude relacija uredjenja (posebno da važi antisimetrija).

Na slici 2 vidi se da se pomoću referentnih koordinata P i Q mogu definisati promenljivi nagibni uglovi. U odredjenoj meri, finoća zadavanja uglova ograničena je diskretizacijom ležišta, no sa stanovišta prakse to ne predstavlja značajan problem. Važno je da se može izbeći da se lokalna odstupanja od stvarno zadatih uglova ne nagomilavaju.

Pomoću relacije prethodjenja može se definisati i stabilnost granice.

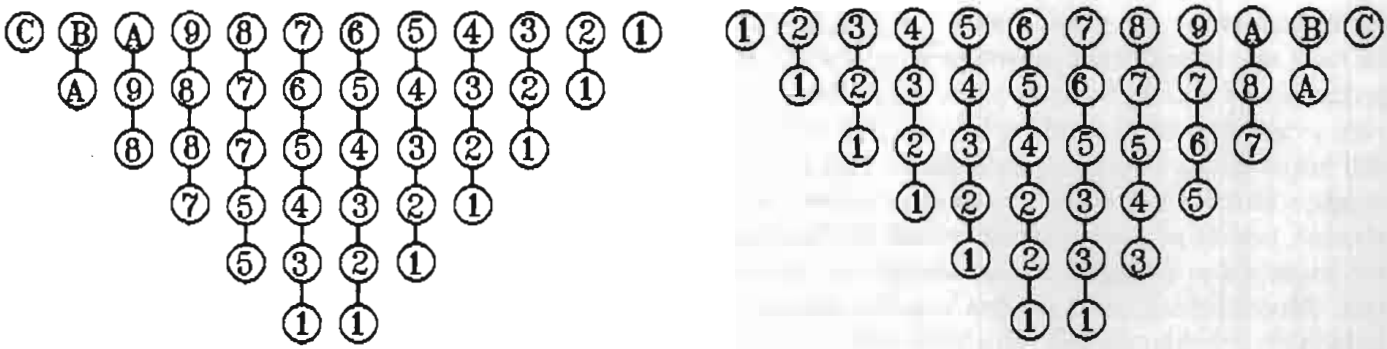
DEFINICIJA 7. Granica G je *stabilna* ako za svake dve tačke t_1 i t_2 $(t_1 \neq t_2)$ t_1 ne prethodi tački t_2 niti t_2 prethodi tački t_1.

Medju granicama može se uvesti relacija poretka.

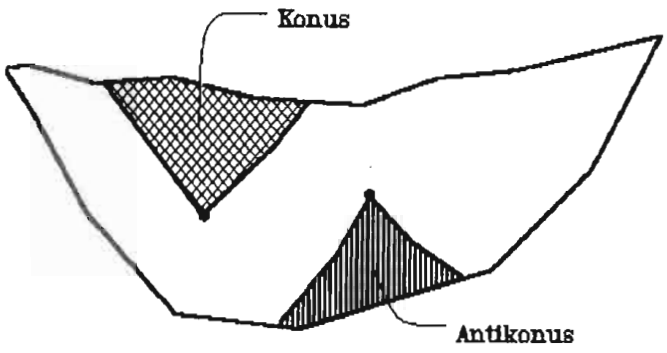
DEFINICIJA 8. Ako su G_1 i G_2 granice i ako je na svakoj vertikali tačka koja pripada G_1 iznad ili jednaka tački koja pripada G_2, onda je G_1 *iznad ili jednaka* granici G_2 i piše se $G_1 \geq G_2$.

Medju stabilnim granicama od posebnog interesa su sledeće dve:

DEFINICIJA 9. *Granica konusa stabilnosti* glavne tačke t, u oznaci $GKS(t)$ je maksimalno stabilna granica koja sadrži glavnu tačku t. *Granica antikonusa stabilnosti* $GAKS(t)$ je minimalna stabilna granica koja sadrži tačku t.



Slika 2. – Referentne koordinate – (a) referentna koordinata P, (b) referentna koordinata Q



Slika 3. – Konus i antikonus stabilnosti i njihove granice

Na slici 3 prikazani su primeri granica GKS i $GAKS$. Jasno je da sve tačke iznad $GKS(t)$ prethode tački t, dok t prethodi svim tačkama ispod $GAKS(t)$.

DEFINICIJA 10. Neka je X bilo kakav skup glavnih tačaka. *Gornja stabilizacija* skupa X je maksimalno stabilna granica $GST(X)$ ispod koje nema ni jedna tačka iz X. Slično, *donja stabilizacija* skupa X je minimalno stabilna granica $DST(X)$ iznad koje nema ni jedne tačke skupa X.

Na slici 4(a) prikazana je crtkastom linijom $GST(X)$, gde je X skup tačaka koje su istaknute na crtežu, a na slici 4(b) prikazana je $DST(X)$.

Lako se može pokazati da su GST, odnosno DST generalizacije, granice konusa stabilnosti, odnosno antikonusa stabilnosti. Naime,

$$GST(\{t\}) = GKS(t)\,,$$

$$DST(\{t\}) = GAKS(t)\,.$$

Osim toga, može se pokazati da je

$$GST(X) = \min_{t\in X} GKS(t)\,,$$

$$DST(X) = \max_{t\in X} GAKS(t)\,,$$

što je prikazano na slici 4.

4. Približno izračunavanje funkcije V_{opt}

4.1 Izračunavanje V^2_{opt} u profilu

Ako se umesto celog ležišta posmatra samo jedan profil, problem optimizacije se krajnje pojednostavljuje. Isto važi i za izračunavanje funkcije V^2_{opt} (indeks 2 označava da se radi o dvodimenzionom prostoru).

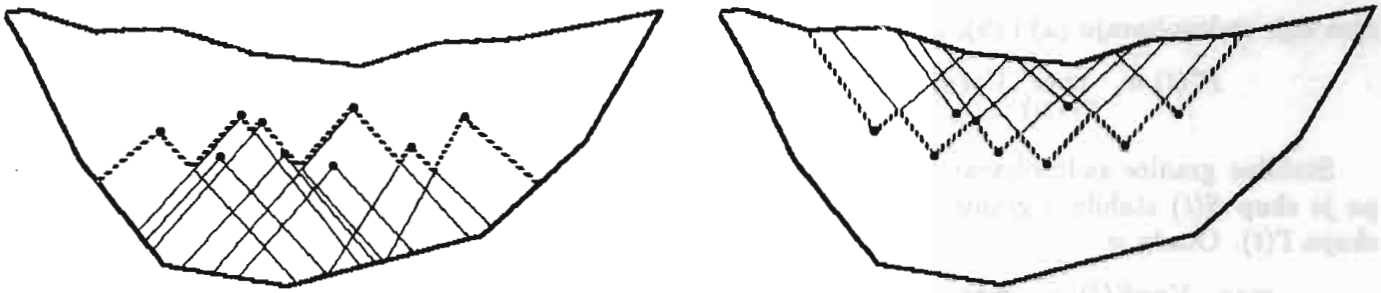
Granica u profilu je skup glavnih tačaka koji na svakoj vertikali profila ima po jednu tačku. Na osnovu definicija 5 i 6 može se pokazati da je granica G u profilu π stabilna ako za svaki par tačaka $t, t_1 \in G$, gde je t_1 na susednoj desnoj vertikali od t, važi

$$P_\pi(t) \geq P_\pi(t_1) \quad \text{i} \quad Q_\pi(t) \leq Q_\pi(t_1) \qquad . \qquad (1)$$

Optimalna granica $G^*(t)$ koja prolazi kroz neku glavnu tačku t može da se odredi tako što se posebno odredjuju levi i desni deo te granice $G^*_L(t)$ i $G^*_D(t)$. Ako označimo

$$F_L(t) = \sum_{t_1\in G^*_L(t)} F(t_1)\,,$$

$$F_D(t) = \sum_{t_2\in G^*_D(t)} F(t_2)\,,$$



Slika 4. – Gornja (a) i donja (b) stabilizacija skupa

onda je

$$V^2_{opt}(t) = F_L(t) + F_D(t) - F(t) \,. \qquad (2)$$

Izraz $F(t)$ se oduzima zato što $t \in G^*_L(t)$ i $T \in G^*_D(t)$, pa se $F(t)$ pojavljuje u oba zbira $F_L(t)$ i $F_D(t)$.

$F_D(t)$ može da se izračuna na sledeći način.

Ako je $t_1 \in G_D(t)$ tačka na susednoj desnoj vertikali od t, onda važi relacija (1). Osim toga, lako se može pokazati da je

$$G^*_D(t) \backslash \{t\} = G^*_D(t_1) \,.$$

Drugim rečima, kada se iz $G^*_D(t)$ izbaci tačka t, dobija se optimalna desna parcijalna granica za tačku t_1. Prema tome,

$$F_D(t) = F(t) + \sum_u F(u) = F(t) + F_D(t_1) \,, \qquad (3)$$

pri čemu se sumacija uzima po $u \in G^*_D(t_1)$. Na osnovu jednakosti (3), F_D može da se izračuna rekurzivno, po vertikalama, počev od krajnje desne ulevo. Na krajnjoj desnoj vertikali je $F_D = F$. Ako t nije na krajnjoj desnoj vertikali, onda se medju tačkama na susednoj desnoj vertikali traži t_1 koja zadovoljava uslov (1) i u kojoj je F_D maksimalno. Funkcija $F_D(t)$ izračunava se jednostavno na osnovu (3).

Na sličan način, krećući se sleva udesno, može se izračunati F_L, a na osnovu (2) i funkcija V^2_{opt}.

Očigledno, ovaj postupak je krajnje jednostavan i brz. Vreme računanja proporcionalno je broju glavnih tačaka. Ideja ovakve optimizacije javlja se već u [1] i dalje je primenjivana u [3] i [4]. Medjutim, u svim tim radovima, stabilnost je bila definisana na krajnje jednostavan način: podela po z osi bila je takodje uniformna (svi blokovi su imali istu visinu) i od neke glavne tačke smelo se ići samo jedan blok naviše, ravno ili naniže. Ovo je bilo ozbiljno ograničenje, s obzirom na situaciju u vezi sa nagibnim uglovima koja postoji u realnim uslovima. Razrešenje ovog problema pomoću referentnih koordinata prvi je uveo autor u [2].

4.2 Primena 2D optimizacije za približno izračunavanje V_{opt}

Atraktivnost 2D optimizacije u pogledu jednostavnosti i brzine navodila je mnoge autore da je primenjuju za približnu optimizaciju u 3D prostoru. Ovde navodimo jedan takav postupak.

Na način koji je opisan u 4.1 može se izračunati V^2_{opt} u svim profilima paralelnim sa x osom. Zatim se dobijena funkcija može uzeti kao osnovna (umesto F) za slično izračunavanje u svim profilima u pravcu y ose. Označimo sa F^* dobijenu funkciju.

Neka je t bilo koja glavna tačka. Iz postupka računanja može se pokazati da postoji granica G kroz t za koju važi sledeće:

(a) G zadovoljava uslove stabilnosti u svim profilima u pravcu x ose,

(b) G zadovoljava uslove stabilnosti u onom profilu u pravcu y ose u kojem se nalazi tačka t,

(c) $F^*(t) = Vred(G)$, i

(d) ne postoji ni jedna granica G' koja zadovoljava uslove (a) i (b) takva da je $Vred(G') > Vred(G)$.

Drugim rečima, ako sa $\Gamma(t)$ označimo skup gra-

nica koje zadovoljavaju (a) i (b), tada je

$$F^*(t) = \max_{G \in \Gamma(t)} Vred(G) .$$

Stabilne granice zadovoljavaju uslove (a) i (b) pa je skup $S(t)$ stabilnih granica kroz t podskup skupa $\Gamma(t)$. Otuda je

$$\max_{G \in \Gamma(t)} Vred(G) \geq \max_{G \in S(t)} Vred(G) .$$

tj. $F^*(t) \geq V_{opt}(t)$. Tako smo pokazali da je F^* **majoranta** funkcije V_{opt}.

Kao što je rečeno, pri traženju funkcije F poštovani su uslovi stabilnosti u pravcu x ose i samo delimično u pravcu y ose. Zbog toga se ovaj postupak u literaturi često naziva *dvoipodimenzionom (2.5D) optimizacijom.*

Postupak 2.5D optimizacije može se obaviti i tako što se prvo obavlja 2D optimizacija u pravcu y ose, a zatim u pravcu x ose. U [2] se pokazuje da postoji deset takvih shema 2.5D optimizacije, kada se uzmu u obzir i profili u dijagonalnim pravcima.

Višestruka 2.5D optimizacija može da bude od koristi. Neka su F_1^* i F_2^* funkcije dobijene 2.5D optimizacijama i neka je

$$F'(t) = \min(F_1^*(t), F_2^*(t)) ,$$

za svaku glavnu tačku t. S obzirom da su F_1^* i F_2^* majorante funkcije V_{opt}, to važi

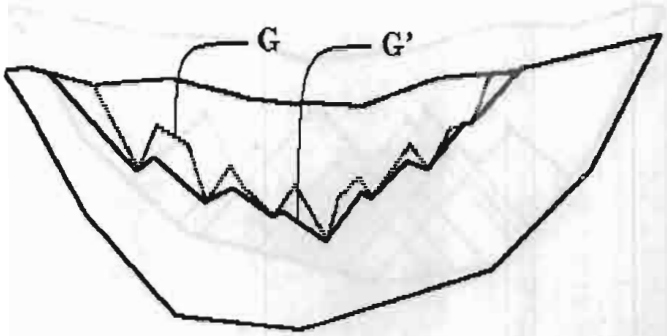
$$F_1^* \geq F' \geq V_{opt} \quad \text{i} \quad F_1^* \geq F' \geq V_{opt} .$$

Prema tome, primenom jednostavnog operatora min iz dve majorante F_1^* i F_2^* funkcije V_{opt} dobija se F' koja je takodje majoranta funkcije V_{opt}, ali je joj je bliža, ukoliko nije $F_1^* = F_2^*$.

Napominjemo da se sva računanja o kojima je do sada bilo reči obavljaju u vremenu koje je proporcionalno broju glavnih tačaka. Osim toga, algoritmi kojima se obavljaju ova računanja mogu se bez značajnih posledica u pogledu ukupnog vremena računanja organizovati tako da su istovremeno u memoriji računara samo dve susedne vertikale.

5. Približna optimizacija pomoću majorante funkcije V_{opt}

Funkcija F^* koja se dobija (eventualno višestrukom) primenom 2.5D optimizacije, može se upotrebiti za približno odredjivanje optimalne granice na sledeći način.



Slika 5. – Približno rešavanje zadatka optimizacije

Na svakoj vertikali odredi se tačka u kojoj F^* ima najveću vrednost i tako se dobija granica G, koja ne mora da bude stabilna (crtkasta linija na slici 5). Primenom operatora stabilizacije može se dobiti granica $G' = GST(G)$ (puna linija na slici 5), koja se može smatrati približnim rešenjem.

Da bismo ocenili tačnost približnog rešenja, odredićemo interval u kojem se nalazi vrednost optimuma, kao i granice u prostoru izmedju kojih se nalazi optimalna granica.

Posmatrajmo bilo koju glavnu tačku $t \in G$ koja pripada nekoj vertikali V. Kako je za svako $t_1 \in V$

$$F^*(t) \geq F^*(t_1) \quad \text{i} \quad F^*(t_1) \geq V_{opt}(t_1) ,$$

to je za svako $t_1 \in V$

$$F^*(t) \geq Vopt(t_1) .$$

Ako je G^* optimalna stabilna granica, ona ima u vertikali V bar jednu tačku t', pa je

$$F^*(t) \geq V_{opt}(t') = Vred(G^*) .$$

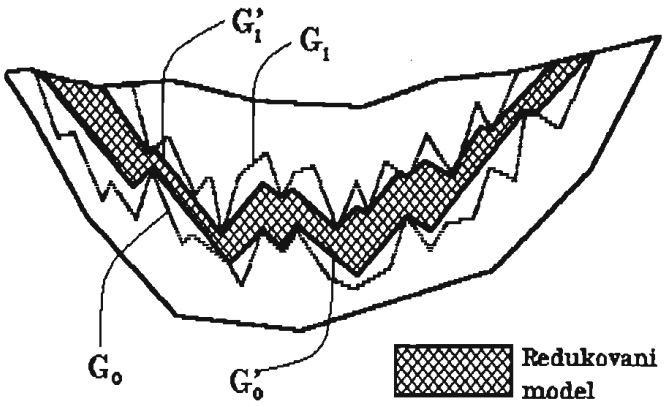
Primetimo da prethodna relacija važi za svako $t \in G$. Otuda je

$$Vred(G^*) \leq \min_{t \in G} F^*(t) .$$

Tako smo došli do jednog gornjeg ograničenja za vrednost optimalne granice. S druge strane, pošto je G' jedna stabilna granica, to je

$$Vred(G') \leq Vred(G^*) \leq \min_{t \in G} F^*(t) .$$

Prema tome, na osnovu približne majorante F^* funkcije V_{opt} može se odrediti interval u kojem se nalazi vrednost optimalne granice. Na osnovu iste majorante, mogu se u prostoru odrediti dve granice



Slika 6. – Redukcija modela

G_0 i G_1 ($G_0 \leq G_1$), a zatim i dve stabilne granice G'_0 i G'_1, tako da važi

$$G_0 \leq G'_0 \leq G^* \leq G'_1 \leq G_1 \ .$$

Neka je $g = Vred(G')$. Da bismo formirali granicu G_0 izaberimo na svakoj vertikali prvu tačku t_0 *odozdo* za koju važi $F^*(t_0) \geq g$ (može se lako dokazati da takva tačka uvek postoji). Slično, za granicu G_1 izaberimo prvu tačku t_1 *odozgo* za koju je $F^*(t_1) \geq g$ (slika 6).

Dokazaćemo da optimalna granica ne prolazi iznad G_1 niti ispod G_0.

Za sve tačke t' iznad G_1, odnosno ispod G_0, važi

$$F^*(t') < g = Vred(G') \ .$$

Kako je $Vred(G') \leq Vred(G^*)$, to je

$$F^*(t') < Vred(G^*) \ .$$

S druge strane, F^* je majoranta funkcije V_{opt}, pa je

$$V_{opt}(t') \leq F^*(t') \ .$$

Iz poslednje dve nejednakosti sledi

$$V_{opt}(t') < Vred(G^*) \ ,$$

što znači da kroz t' ne prolazi optimalna granica.

Kako je G^* stabilna granica čija ni jedna tačka nije iznad G_1, onda je $G^* \leq DST(G_1)$, jer je $DST(G_1)$ maksimalno stabilna granica čija ni jedna tačka nije iznad G_1. Tako se dobija da je $G'_1 = DST(G_1)$ gornje prostorno ograničenje optimalne granice.

Slično, kako je G^* stabilna granica čija ni jedna tačka nije ispod G_0, to je $G'_0 = GST(G_0)$ donje prostorno ograničenje za granicu G^*.

Iz svega izloženog sledi da pomoću majorante F^* funkcije V_{opt} možemo da odredimo:

(a) približnu optimalnu granicu G',

(b) interval u kojem se nalazi vrednost optimalne granice, i

(c) stabilne granice G'_0 i G'_1 izmedju kojih se nalazi optimalna granica.

Naravno, ukoliko je F^* bolja aproksimacija funkcije V_{opt}, toliko je G' bliže optimalnoj granici i bolja je procena vrednosti i položaja optimalne granice. Zbog toga je od interesa da se izračunava F^* pomoću višestruke 2.5D optimizacije ili da se potraži bolji način za izračunavanje majorante.

6. Redukcija modela

Ako se G'_0 i G'_1 ne poklapaju sa donjom, odnosno gornjom granicom modela ležišta L i ako se odbace sve glavne tačke iznad G'_1 i ispod G'_0, dobija se model L' koji ima istu optimalnu granicu kao i polazni. Ovaj postupak zove se *redukcija modela.*

U modelu L' može da se ponovi postupak 2.5D optimizacije. S obzirom da je skup glavnih tačaka u L' pravi podskup skupa glavnih tačaka u L, to je skup granica koje ispunjavaju uslove (a) i (b) u 4.2 manji u modelu L' nego u modelu L. Iz toga sledi da majorante funkcije V_{opt} dobijene 2.5D optimizacijom u L' ne mogu da budu veće nego L, tj. u L' mogu da se dobiju bolje aproksimacije funkcije V_{opt} nego u L. Posledica toga je obično da se redukovani model L' može i dalje redukovati.

Iterativni postupak uzastopnih redukcija može da se zaustavi u jednom od sledećih slučajeva:

(a) $G'_0 = G'_1$. Tada je $G'_0 = G^* = G'_1$, pa se u tom slučaju dobija tačno rešenje zadatka optimizacije;

(b) Interval u kojem je vrednost približnog rešenja i/ili položaj granice u prostoru manji je od nekog zadatog. U tom slučaju, rešen je zadatak optimizacije *sa traženom tačnošću*;

(c) G'_1 i G'_0 poklapaju se sa gornjom, odnosno donjom granicom (redukovanog) modela. Tada se dobijaju samo približno rešenje i ocena odstupanja kao što je opisano u 5.

Prema tome, opisana metoda ne može se strogo klasifikovati ni kao približna ni kao tačna. Značajno je da je u praksi slučaj (a) veoma čest.

Ideja redukcije prvi put se pominje u [4]. U tom radu, redukcija se uvodi kao sredstvo koje pomaže

da se smanji model, koji će se posle toga podvrći optimizaciji metodom grafova. Postupak koji se tamo izlaže znatno se razlikuje od postupka izloženog u ovom radu. Osim toga, model uglova je krajnje pojednostavljen u odnosu na realno stanje i u odnosu na model koji se kasnije primenjuje u metodi grafova.

Literatura

1. Lerch H, Grossmann I F: Optimum Design of Open Pit Mines, *Canadian Institute of Mines Bulletin*, Jan. 1965.
2. Dušan Bratičević: Primena računara u konstrukciji optimalne konture površinskog otkopa, *Doktorski rad*, Univerzitet u Beogradu, maj 1986.
3. Johnson T B, Sharp W R: A Three Dimensional Dynamic Programming Method for Optimum Open Pit Design, *U.S.B.M. RI7553*, 1971.
4. Barnes R J, Johnson T B: Bounding Techniques for the Ultimate Pit Limit Problem, *Proceedings of the 17^h APCOM*, Golden, Kolorado, SAD 1982.

Adresa autora:
Dušan Bratičević
11070 NOVI BEOGRAD
Omladinskih brigada 220/II

UDK 681.32.06/08

Razvoj grafičkog softvera hemijskih i hemijsko–inženjerskih informacija

J.Savković – Stevanović, G.Popović, D.Poleti, M.Mitrinović

Kratak sadržaj: *U radu je prikazan razvoj i primena grafičkog softvera kao programske podrške za projektovanje i upravljanje hemijskih procesa. Pri tome je korišćena biblioteka grafičkog softvera instalirana na mikroračunarskom sistemu Burroughs B20. Rezultati su prikazani u vidu datoteka grafičkih informacija za tipične sisteme hemijskih procesa. To su moduli za višestupnjevitu separaciju, hemijsko inženjerske parametre i strukture hemijskih jedinjenja. Prikazani su i rezultati grafičkih informacija nekih funkcija za modelovanje proizvodnih sistema.*

Ključne reči: *grafički softver, hemijski proces, modelovanje, strukturna formula jedinjenja*

1. Uvod

Grafički modul na sistemu Burroughs B20 podržava osnovne grafičke funkcije (linija, luk, krug, pravougaonik, ispunjavanje površina, karakteri, programabilne veličine i vrste slova). Ulogu grafičkog procesora obavlja Intelov mikroprocesor 8086. Na sistemu su instalirana tri programska paketa DRAW, MULTIPLAN i BGP (Business Graphics Package) (1-4).

Trećoj generaciji simulatora u hemijskom inženjerstvu pripada ASPEN - program za projektovanje hemijskih procesa, MIT 1981, (5). Takodje, postoji i grafički simulator GSCHEMER (Graphic Structured CHEM-o-SyntesizER) koji pokriva operacije hemijske reakcije, razmenu toplote, kompresiju i ekspanziju gasa.

U ovom radu je ispitivana mogućnost grafičkog prikaza različitih hemijskih i hemijsko inženjerskih informacija uporedo na tri programska paketa — DRAW, MULTIPLAN I BGP. Svrha ovih datoteka grafičkih informacija je da posluže za kreiranje blokova potrebnih za analizu, modelovanje i projektovanje hemijskih procesa (6).

Delovi ovog rada saopšteni su na II jugoslovenskom kongresu za hemijsko inženjerstvo i procesnu tehniku, Dubrovnik, maj 1987.

2. Grafička biblioteka i funkcionalne procedure

Grafička biblioteka na sistemu B20 sastoji se od dva tipa procedura (1). Glavni deo biblioteke čine nezavisne procedure koje se pozivaju pomoću grafičkog programa. Druge procedure ove biblioteke su zavisne procedure (zavisne od uredjaja) i izvršavaju se brže od nezavisnih procedura. Funkcije nezavisnih procedura su: inicijalizacija, slika, predmet, atributi, crteži, tekst, tip slova, labela, transformacija, izgled, pokazivač i dr. Zavisne procedure se mogu grupisati prema funkcijama koje obavljaju u sledeće kategorije: kontrola, boja, vektor, inverzna manipulacija i alfanumerički atributi.

Grafički softver na sistemu B20 čine programski paketi DRAW, MULTIPLAN i BGP. DRAW program omogućava predstavljanje informacija pomoću dijagrama, šema i crteža (2). U DRAW programu mogu se koristiti sledeće procedure: editovanje objekta, kreiranje templit (template) šabloniziranih objekata, otvaranje i zatvaranje prozora, promena palete i zatvaranje prozora, promena palete boja, uvećanje linije, centriranje teksta ili objekta, korišćenje predstavljene templit datoteke, izbor željene templit datoteke, ponovno vraćanje iz sistema grešaka, ponovno vraćanje iz CLEAR i prihvatanje HELP datoteka.

Programski paket DRAW omogućava korisniku da u potpunosti iskoristi sve prednosti kompjuterske grafike. Veliki broj kontrolisanih tačaka (1208x912) na ekranu i poznavanje koordinata svake tačke obez-

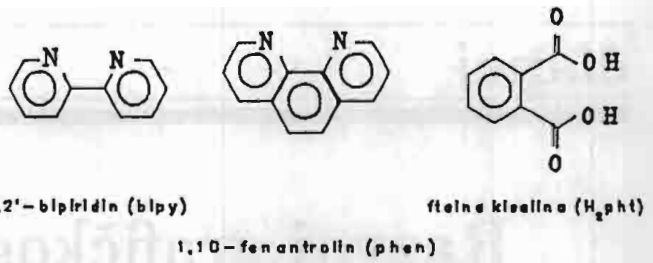
bedjuje postizanje visoke preciznosti i dobre rezolucije dobijenih slika. Moguće je kreirati, zapamtiti i kombinovati proste ili složene objekte, tehničke crteže i planove, električne, proizvodne i upravljačke šeme, blok dijagrame itd. U ovom programskom paketu već je programirano crtanje 30 prostijih geometrijskih figura (od tačke do spirale), a za crtanje šema iz elektronike i blok dijagrama postoje posebne templit datoteke sa većim brojem specifičnih simbola. Najzanimljivije je svakako što se kombinovanjem prostih slika mogu kreirati i memorisati složeni objekti koji se kasnije pozivaju pritiskom samo jednog tastera. Na taj način, svaki korisnik može imati jedan ili više svojih templit datoteka sa po 200 simbola koje često koristi u svojim crtežima. Iz TEMPLATE menija se takodje pritiskom na taster SCROLL UP, bira objekat iz sledećeg nivoa objekata, a pritiskom na SCROLL DOWN bira objekat iz prethodnog nivoa objekata.

Programski paket MULTIPLAN stoji na raspolaganju korisniku u dve verzije — MULTIPLAN i EMULTIPLAN (prošireni MULTIPLAN)(4). Programski paket MULTIPLAN omogućuje sledeće funkcije: kreiranje radne površine od 255 redova i 63 kolone, koristeći reči, brojeve i formule; proračun vrednosti radne površine, automatski ili pomoću komandi; dodavanje,brisanje, premeštanje ili kopiranje redova i kolona ; otvaranje prozora na ekranu; mogućnost da se levo od radnih površina kopiraju postojeći podaci drugih radnih površina ; štampanje radnih površina ili delova radnih površina; povezivanje sa BGP programima i pokazivanje alfabetske liste MULTIPLAN komandi kao i njihovog opisa. EMULTIPLAN omogućuje: skladištenje radnih površina u dokument promenljivog formata tako da se može koristiti radna površina programa za obradu teksta; korišćenje iteracionih opcija, korišćenje raspoložive memorije u tekućoj particiji za radnu površinu, displej radne površine u boji i slanje poruke iz B20 operativnog upravljačkog sistema.

Programski paket BGP (Business Graphics Package) veoma je pogodan za pretvaranje grafičkih informacija na osnovu tabelarnih podataka, i to u vidu tri različite interpretacije: histogrami, kružni dijagrami i linijski dijagrami (3).

3. Grafički prikaz hemijsko inženjerskih i hemijskih informacija

Simboli za predstavljanje funkcija za modelovanje proizvodnih sistema kreirani u programu DRAW u datoteci GRAFPS, prikazani su u tabeli 1. Detaljniji



Slika 2. – Primer strukturnih formula koje se nalaze u templit datoteci

simboli za predstavljanje modula hemijskih procesa kreirani su, pak, u datoteci GRAFHP, čiji je sadržaj prikazan u tabeli 2. (2–8).

Primena ovih specifičnih simbola za predstavljanje tokova hemijskih procesa ilustrovana je na primeru jednog sistema za razdvajanje destilacijom (9-13), koji je pokazan na sl.1. Ova grafička sekvenca kreirana je takodje u DRAW.

U hemiji i srodnim naukama, jedinjenja se često predstavljaju strukturnim formulama. Ako se elementi takvih formula unapred programiraju u templit datoteci, rad će biti u velikoj meri olakšan. Kao primer, date su strukturne formule nekoliko organskih jedinjenja poznatih po svojim donorskim sposobnostima, kao što je pokazano na slici 2.

Formule se po potrebi mogu modifikovati povećanjem, izduživanjem, rotiranjem itd., kao što je to pokazano na sl.3.

Tako sada jednačine:

$$Cu(aq) + phen(EtOH) = Cu(phen)(aq)$$

$$Cu(phen)(aq) + pht(aq) =$$

$$= [Cu(pht)(H_2O)(phen)] \quad (s)$$

možemo prikazati kao na sl.4.

Istraživanja su, medjutim, pokazala da prikazani kompleks ima polimernu strukturu u kojoj je bakar pentakoordiniran, a ftalat jon ima mostovnu ulogu (15). Ovo jedinjenje je šematski prikazano na slici 5.

Pod sličnim uslovima moguće je dobiti i odgovarajuće jedinjenje nikla:

$$[Ni(pht)(H_2O)(phen)].H_2O.$$

Odredjivanje njegove kristalne strukture (16) pokazalo je oktaedarski raspored liganada oko nikla, a ftalat jon se, za razliku od prethodnog slučaja, ponaša kao monodentatni ligand (sl.6).

Rezultati ispitivanja ravnotežnih hemijsko–inženjerskih parametara prikazani su pomoću programskog paketa MULTIPLAN u vidu dijagrama na sl.7.

TABELA 1

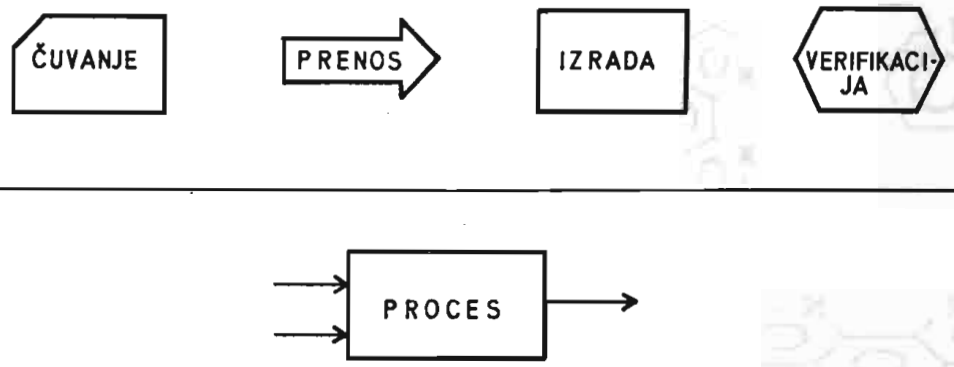


Tabela 1. – Grafički simboli za modelovanje proizvodnih informacija

TABELA 2

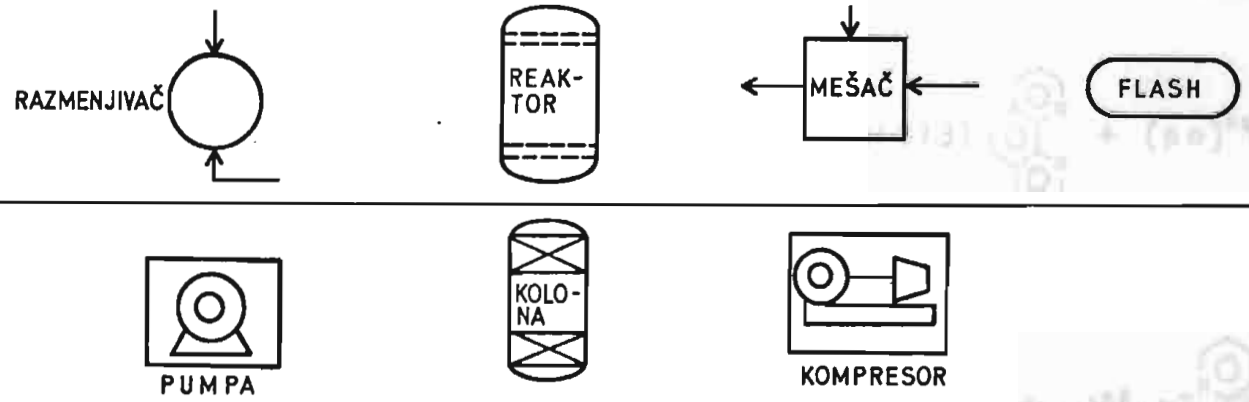
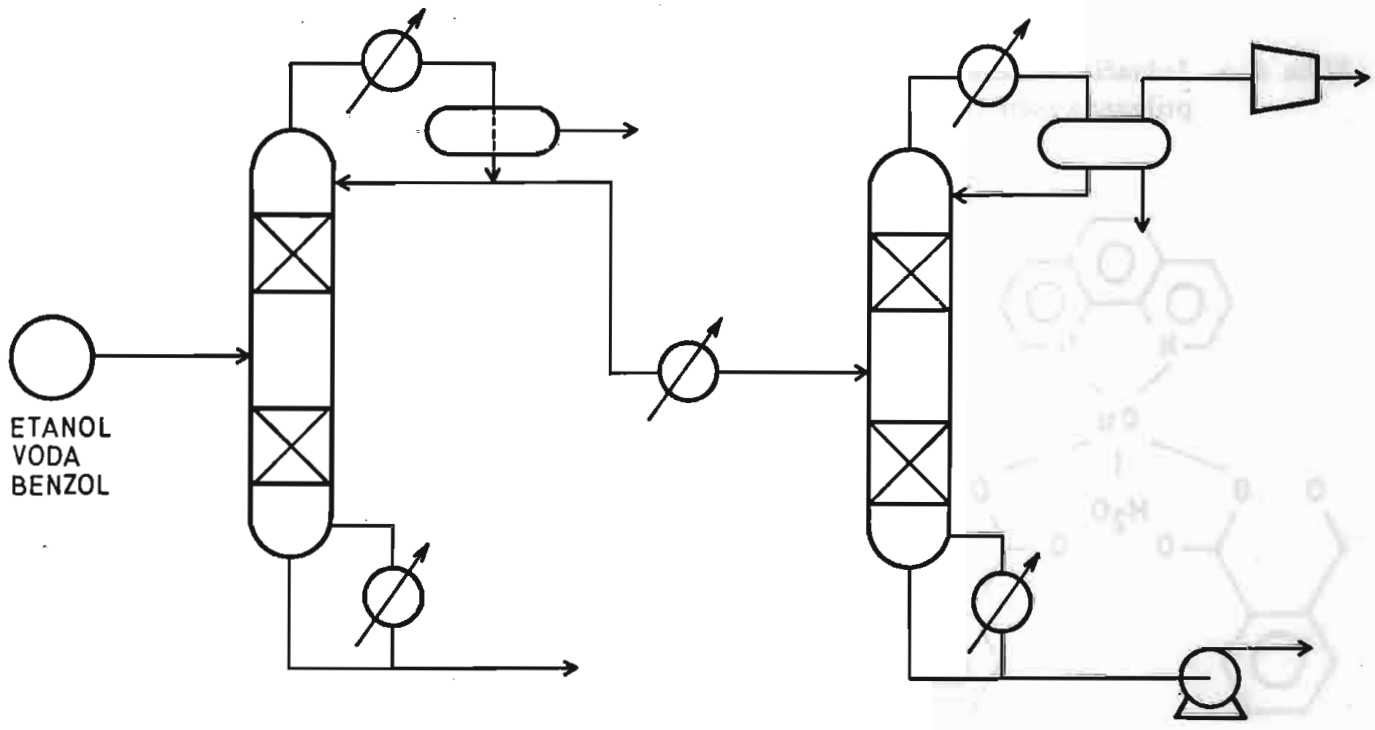
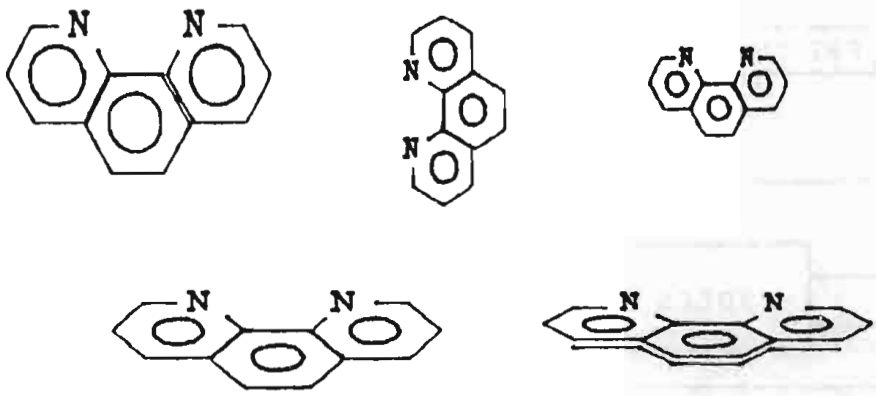


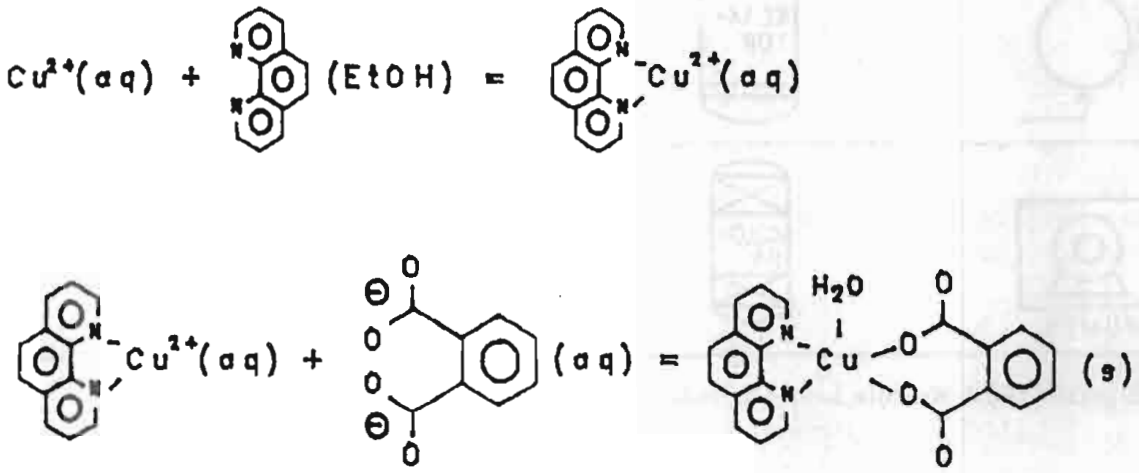
Tabela 2. – Grafički prikaz nekih modula hemijsko-inženjerskih informacija



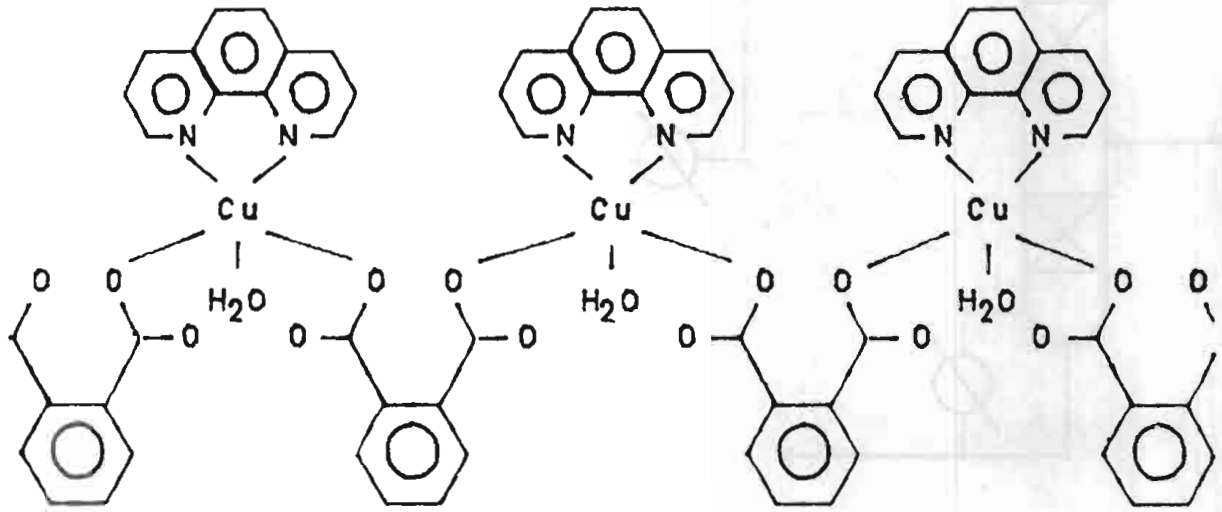
Slika 1. – Prikaz sistema za azeotropnu rektifikaciju



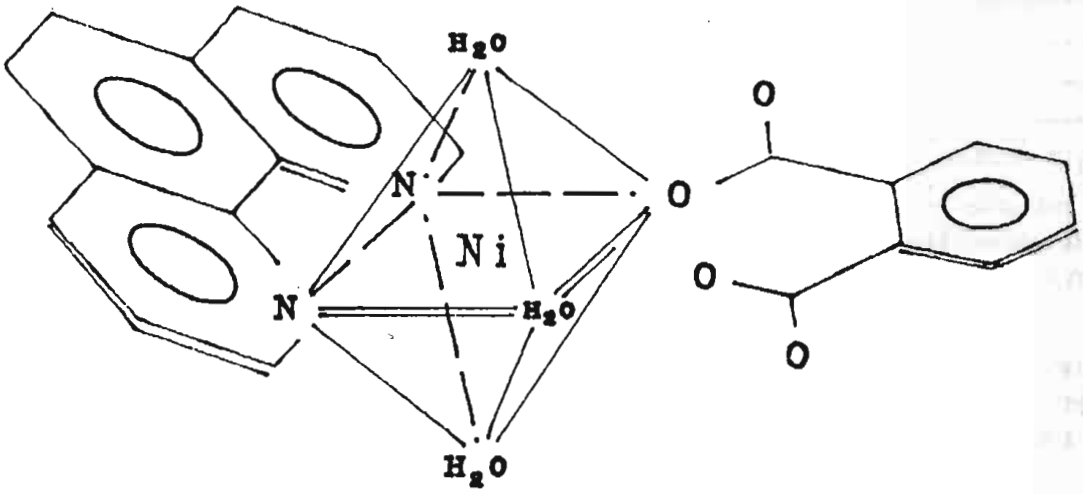
Slika 3. – Prikaz mogućnosti za modifikaciju postojećih objekata



Slika 4. – Jednačina sinteze jedinjenja $[Cu(pht)(H_2O)(phen)]$ prikazana pomoću strukturnih formula



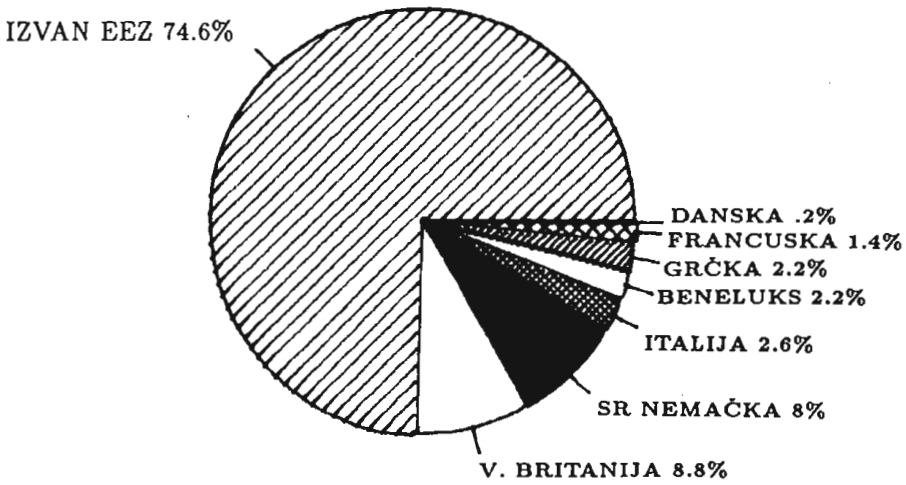
Slika 5. – Šematski prikaz kristalne strukture jedinjenja $[Cu(pht)(H_2O)(phen)]$



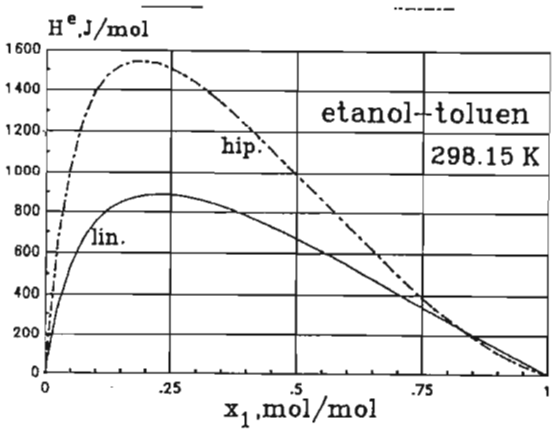
Slika 6. – Šematski prikaz kristalne strukture jedinjenja $[Ni(pht)(H_2O)(phen)].H_2O$

UVOZ OBOJENIH METALA U SFRJ PREMA ZEMLJI POREKLA

GODINA 1976



Slika 8. – Prikaz izvoza obojenih metala



Slika 7. – Dopunska entalpija mešanja za sistem etanol-toluen po dvoparametarskom Wilsonovom modelu

Ilustracija BGP programa pokazana je na predstavljanju izvoza obojenih metala (17). Prikaz je dat na sl.8 pomoću kružnih dijagrama.

4. Zaključak

Rezultati istraživanja u ovom radu predstavljeni su u više datoteka grafičkih informacija za predstavljanje: funkcije modelovanja proizvodnih sistema, tokova hemijskih procesa i specifičnih simbola za predstavljanje strukturnih formula jedinjenja. Primena specifičnih simbola za predstavljanje tokova hemijskih procesa, diskutovana je na primeru azeotropne rektifikacije. Simboli za predstavljanje strukturnih formula prikazani su na strukturama nekih kompleksnih jedinjenja bakra i nikla koji sadrže ligande sa izraženim donorskim sposobnostima.

Rezultati dobijeni u ovom radu mogu se u daljem razvoju i primeni programske podrške koristiti ravnopravno sa numeričkim i bibliografskim datotekama i bankama podataka.

Literatura

1. Programmer's Guide, B20 Systems Graphics, Relative to Relėase Level 4.0, Copyright Burroughs Corporation, Detroit, Michigan (1985).
2. User's Guide, B20 Systems - B20 Draw, Relative to Release Level 1.0, Copyright Burroughs Corporation, Detroit, Michigan (1984).
3. Reference Manual B20 System Business Graphic Package (BGP) Release Level 4.0, Copyright Burroughs Corporation, Detroit, Michigan (1984).
4. User's Guide, B20 Systems Multiplan , Relative to Release Level 1.0 Enhanced Multiplan, Level 3.0 Multiplan, printed in U.S.A., feb. (1985).
5. Sawicki S.J.,Young R.D.,Sund S.E., Comp.Chem.Eng., 10, No.3, (1986) 297.
6. Savković-Stevanović J.,Informacioni sistemi u procesnoj tehnici, Naučna knjiga, Beograd (1987).
7. CHEMSHARE DESIGN /2000, version 7.6o46 ACHEMA SHORTCUT (1985)
8. Peters B.A., Chem.Eng.,May, 13 (1987) 95.
9. Savković-Stevanović J., Sep.Sci.Techn., 19, Nr.4-5 (1984) 283.
10. Savković-Stevanović J., Chem.-Ing.-Tech., 57,Nr.4 (1985) 368.
11. Savković-Stevanović J., CHISA'84, paper C5.2,No.741, Prague sept.(1984).
12. Savković-Stevanović J., Simonović D., J.Chem.Eng.Data,21 (1976)456.
13. Savković-Stevanović J., Bilten korisnika, Informatika–Interkomerc, Beograd, Vol.1, Nr.1 (1986) 1.
14. Stojaković Dj.R., Poleti D.D., Aleksić R.R., J.Serb.Chem. Soc.,50 (1985) 241.
15. Krstanović I.,Karanović Lj.,Stojaković Dj., Acta Cryst., C41 (1985) 43.
16. Poleti D., Prelesnik B., Herak R., Stojaković Dj.,X European Crystallographic Meeting, Wroclaw, Poland,august (1986).
17. Statistika spoljne trgovine SFRJ, Savezni zavod za statistiku (1976).

Adresa autora: Računski centar Tehnološko-metalurškog fakulteta u Beogradu
11000 Beograd
Karnegijeva 4, P.O.B. 494,

UDK 681.32.06/08

Rastavljanje polinoma na faktore primenom matematičkih spektara

Jovan D. Madić

Kratak sadržaj: *U radu je izložena jedna nova spektralna metoda za rastavljanje polinoma sa celim koeficijentima na faktore. Najpre se razmatra rastavljanje pozitivnog polinoma i daje dovoljan uslov za dobijanje dva uzajamno dopunska pozitivna faktora. Proizvoljan polinom rastavlja se na faktore transformacijom u strogo pozitivan polinom (prvi način). Isto tako, daju se dovoljni uslovi za rastavljanje proizvoljnog polinoma na faktore (drugi način).*

Ključne reči: *spektar polinoma, ritam, pruga spektra, multiplikativni izomorfizam, metod pseudospektra.*

1. Uvod

Dr K. Orlov je dao dve spektralne metode za rastavljanje polinoma na faktore [1]. Medjutim, ove metode su dosta nepraktične zbog velikog ritma spektra polinoma (nadjenog teorijski). U ovom radu izložićemo novu spektralnu metodu koja se bazira na rezultatu Postnikova [2]. Za rastavljanje proizvoljnog polinoma na faktore dovoljno je rešiti problem njegove rastavljivosti i, u slučaju pozitivnog odgovora, naći bar jedan par uzajamno dopunskih faktora polinoma.

U radu ćemo razmotriti rastavljanje polinoma

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \qquad (1)$$

sa celim koeficijentima uz uslove $a_n > 0$, $n > 0$. Uvedimo sledeće definicije i pojmove, koje ćemo koristiti u radu.

DEFINICIJA 1. Polinom g sa celim koeficijentima (pozitivnog stepena) naziva se faktorom (deliocem) polinoma (1), ako postoji polinom s (dopunski faktor) sa celim koeficijentima tako da važi $f = g \cdot s$. Polinom, koji nema delioce, naziva se nerastavljivim (nesvodljivim).

DEFINICIJA 2. Spektar S polinoma (1) jeste ceo broj

$$S = f(10^h) \qquad (2)$$

gde je h ritam spektra dobiven iz uslova

$$10^h > 2 \cdot a, \quad a = \max_i \{|a_i|\} \qquad (3)$$

Broj S je pozitivan zbog uslova $a_n > 0$ i uslova (3). Ako spektar S izdelimo redom na disjunktne grupe od po h cifara zdesna ulevo dobićemo *pruge* spektra $\alpha_0, \alpha_1, \ldots, \alpha_n$. Tada važi zapis $S = \alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0$. Razlikujemo velike i male pruge [6]. Pruga je velika ako počinje velikom cifrom (5-6), a mala ako počinje malom cifrom (0-4). Pored toga, imamo tri vrednosti pruga.

Nominalna vrednost pruge je broj zapisan u prùzi. *Popravljena* vrednost pruge je jednaka nominalnoj ili za jedinicu veća ako je prva pruga zdesna mala , odnosno velika. *Efektivna* vrednost male pruge jednaka je popravljenoj vrednosti, dok je efektivna vrednost velike pruge jednaka razlici popravljene vrednosti pruge i broja 10^h. Ova poslednja definicija važi za pozitivne spektre. Efektivne vrednosti pruga α_i, $i = \overline{0,n}$ spektra S iz (2) jesu upravo koeficijenti a_i, $i = \overline{0,n}$ polinoma (1).

DEFINICIJA 3. Spektralni faktor spektra S sa ritmom h jeste svaki aritmetički faktor S_1 broja S koji sa ritmom h jednoznačno odredjuje polinom stepena $n > 0$.

Napomena: Osnivač teorije matematičkih spektara je naš veliki matematičar M.Petrović-Alas. Praktičnu primenu matematičkih spektara dao je Konstantin Orlov. Čitaoci, koji žele da se bliže upoznaju sa ovom oblašću, mogu koristiti sledeću literaturu: [1], [3], [4], [5], [6].

2. Pozitivni polinomi

DEFINICIJA 4. Polinom (1) nazivamo *pozitivnim* ako su svi njegovi koeficijenti nenegativni, tj.

$$a_n > 0, a_{n-1} \geq 0, \ldots, a_1 \geq 0, a_0 \geq 0.$$

Jasno je da je proizvod dva pozitivna polinoma pozitivan. Medjutim, obrat ne važi. Ritam h pozitivnog polinoma može se odrediti iz uslova $h = (a)$, $a = \max_i\{a_i\}$, gde je (a) broj cifara broja a. Sve tri vrednosti pruge spektra pozitivnog polinoma su jednake.

Kao prvi korak u rešavanju opšteg problema rastavljanja proizvoljnog polinoma na faktore, Jakovkin je predložio razmatranje pomoćnog problema (2).

Zadatak Jakovkina: Za proizvoljan polinom, koji je pozitivan, naći bar jedan par njegovih uzajamno dopunskih pozitivnih delioca (ili dokazati da ne postoji ni jedan takav par).

Rešenje ovog zadatka zasniva se na sledećoj lemi i teoremi.

LEMA: Najveći koeficijent proizvoda $g \cdot s$ dva pozitivna polinoma g i s nije manji od najvećeg koeficijenta svakog od faktora.

DOKAZ: Neka su b_{i_0} i c_{j_0} najveći koeficijent polinoma g i s respektivno. Kako se koeficijent $a_{i_0+j_0}$ polinoma $f = g \cdot s$ izračunava po formuli $a_{i_0+j_0} = b_{i_0} \cdot c_{j_0} + \cdots$, gde tri tačke označavaju neke dopunske nenegativne sabirke, to važi $a_{i_0+j_0} \geq b_{i_0} \cdot c_{j_0}$. Otuda, najveći koeficijent polinoma f nije manji od proizvoda $b_{i_0} \cdot c_{j_0}$, odnosno, nije manji od svakog koeficijenta $b_{i_0} \cdot c_{j_0}$.

TEOREMA 1: Svi pozitivni uzajamno dopunski delioci pozitivnog polinoma f sa ritmom h imaju takodje ritam h (ili manji), i pri tome ako važi $f = g \cdot s$, tada je $f(10^h) = g(10^h) \cdot s(10^h)$, odnosno $S_f = S_g \cdot S_s$.

Teorema se dokazuje na osnovu prethodne leme i definicije spektra polinoma. Obrat teoreme ne važi. Specijalno, ako je $S = a \cdot b$, i ako su g i s asocirani polinomi, sa ritmom h, sa brojevima a i b respektivno, tada ne mora da važi $f = g \cdot s$. Na osnovu ove teoreme možemo formulisati sledeće pravilo za rastavljanje pozitivnih polinoma na pozitivne delioce.

PRAVILO 1: Neka je f proizvoljan pozitivan polinom sa ritmom h. Formirajmo spektar $S_f = f(10^h)$ polinoma f i rastavimo ga na uzajamno dopunske spektralne faktore. Za svaki par uzajamno dopunskih spektralnih faktora S_g i S_s spektra S_f, formirajmo (sa ritmom h) asocirane sa njima pozitivne polinome g i s. Svi mogući parovi uzajamno dopunskih pozitivnih delioca polinoma f nalaze se medju tako formiranim parovima (g, s). Zatim, računajući proizvode $g \cdot s$ za sve takve parove i uporedjujući ih sa polinomom f, mi dobijamo rastavljanje datog polinoma f na proizvod dva pozitivna polinoma ili utvrdjujemo da se polinom f ne može rastaviti na pozitivne faktore.

Ukoliko je spektar S polinoma f prost broj ili se ne može rastaviti na par uzajamno dopunskih spektralnih faktora, tada je jasno da se polinom f ne može rastaviti na pozitivne faktore. Gornje pravilo možemo poboljšati rukovodeći se sledećim pravilom izbora, kojim odbacujemo nepotrebne parove (S_g, S_s) uzajamno dopunskih spektralnih faktora spektra S_f polinoma f.

Neka je

$$g = b_p x^p + \cdots + b_0 \,, \; s = c_q x^q + \cdots + c_0$$

par uzajamno dopunskih delioca polinoma (1). Tada važe sledeće jednakosti:

$$n = p + q \tag{4'}$$

$$a_n = b_p \cdot c_q \tag{5'}$$

$$a_0 = b_0 \cdot c_0 \tag{6'}$$

$$f(1) = g(1) \cdot s(1) \tag{7'}$$

Posmatrajmo ceo pozitivan broj S koji predstavlja spektar nekog polinoma f sa ritmom h. Označimo sa $\tau(s)$ broj efektivnih vrednosti pruga spektra S umanjen za jedan (stepen asociranog polinoma), a sa $\sigma(s)$ zbir efektivnih vrednosti pruga spektra S. Pored toga, neka $d(s)$ odnosno $\ell(s)$ označavaju prvu efektivnu vrednost pruge spektra S zdesna odnosno sleva. Koristeći uvedene oznake, jednakosti $(4') - (7')$ možemo zapisati i ovako:

$$\tau(s_f) = \tau(s_g) + \tau(s_s) \tag{4}$$

$$\ell(s_f) = \ell(s_g) + \ell(s_s) \tag{5}$$

$$d(s_f) = d(s_g) + d(s_s) \tag{6}$$

$$\sigma(s_f) = \sigma(s_g) + \sigma(s_s) \tag{7}$$

Pravilo izbora: Za formiranje parova (g, s) uzajamno dopunskih delioca polinoma f treba uzimati u obzir samo one parove uzajamno dopunskih spektralnih faktora (S_g, S_f) spektra S_f polinoma f za koje važe uslovi $(4) - (7)$.

Za slučaj pozitivnih polinoma i pozitivnih faktora uslovi $(4) - (7)$ nisu samo potrebni već i dovoljni, te važi sledeća teorema.

TEOREMA 2: Pozitivni polinomi g i s, asocirani sa uzajamno dopunskim spektralnim faktorima S_g

i S_s spektra S_f pozitivnog polinoma f sa ritmom h, jesu uzajamno dopunski faktori polinoma f ako i samo ako važi uslov (7).

DOKAZ: Dokažimo da je uslov dovoljan. Neka je h ritam pozitivnog polinoma f i neka je $S_f = f(10^h)$ njegov spektar. Nadalje, neka su S_g i S_s dva uzajamno dopunska spektralna faktora broja S_f (sa kojima su asocirani pozitivni polinomi g i s) takva da važi uslov (7). Dokažimo da ja tada $f = g \cdot s$.

Pretpostavimo obrnuto, tj. da je $g \cdot s = F\ (\neq f)$, pri čemu je:

$$\begin{aligned} F &= A_n x^N + \cdots + A_1 x + A_0 \\ f &= a_n x^n + \cdots + a_1 x + a_0 \\ g &= b_p x^p + \cdots + b_1 x + b_0 \\ s &= c_q x^q + \cdots + c_1 x + c_0 \end{aligned}$$

Tada važi uslov (4), odnosno $N = p+q$. Koeficijenti A_k, $k = 0, \ldots, N$ polinoma F se odredjuju kao zbirovi svih proizvoda $b_i c_j$ uz uslov $i + j = k$.

Kako su koeficijenti a_i, b_i, c_i $(i = 0, 1, 2, \ldots)$ jednaki vrednostima odgovarajućih pruga $\alpha_i, \beta_i, \gamma_i$ spektara respektivno S_f, S_g i S_s sa ritmom h, to znači da ćemo koeficijent A_k dobiti i kao zbir proizvoda $\beta_i \cdot \gamma_j$ (uz uslov $i + j = k$), tj.: $A_k = \sum_{i+j=k} \beta_i \gamma_j$. Dakle, A_k možemo dobiti i množenjem spektra S_g i S_s, ali tako da se množe redom pruge spektra S_g sa prugama spektra S_s. Pri tome, prilikom sabiranja odgovarajućih proizvoda pruga $\sum_{i+j=k} \beta_i \gamma_j$ nećemo vršiti prenos (cifara) na poziciju sledeće pruge.

S druge strane, iz uslova $S_f = S_g \cdot S_s$ pruge α_i spektra S_f (koje su jednake odgovarajućim koeficijentima a_i polinoma f) mogu se dobiti množeći spektre S_g i S_s po prugama, ali sada sa uračunavanjem prenosa (mogućeg). Simbolički:

$$\alpha_n \cdots \alpha_1 \alpha_0 = (\beta_p \cdots \beta_1 \beta_0) \cdot (\gamma_q \cdots \gamma_1 \gamma_0)$$

Pruga $\alpha_0 (= a_0)$ jednaka je proizvodu pruga $\beta_0 \cdot \gamma_0$, umanjenom za prenos ε_0 na poziciju sledeće pruge tj:

$$a_0 = A_0 - 10^h \cdot \varepsilon_0, \quad \varepsilon_0 \geq 0 \qquad (8_0)$$

Nadalje, pruga $\alpha_1 (= a_1)$ se dobija iz jednakosti $\alpha_1 = \beta_0\gamma_1 + \beta_1\gamma_0 + \varepsilon_0 - 10^h\varepsilon_1$, gde je ε_1 prenos na poziciju pruge α_2. Pošto je $\beta_0\gamma_1 + \beta_1\gamma_0 = A_1$, to važi i jednakost (8_1):

$$a_1 = A_1 + \varepsilon_0 - 10^h\varepsilon_1, \quad \varepsilon_1 \geq 0 \qquad (8_1)$$

Analogno se dobija

$$a_2 = A_2 + \varepsilon_1 - 10^h\varepsilon_2, \quad \varepsilon_2 \geq 0 \qquad (8_2)$$

i na kraju

$$a_N = A_N + \varepsilon_{N-1} - 10^h\varepsilon_N, \quad \varepsilon_N \geq 0 \qquad (8_N)$$

Time smo pokazali da je $N \leq n$.

Što se tiče koeficijenata $a_{N+1}, \ldots, a_n$ (ako postoje), oni predstavljaju vrednost pruga (sa ritmom h) broja (spektra) ε_N, tj:

$$\varepsilon_N = a_n 10^{(n-N-1)h} + \cdots + a_{N+2}10^h + a_{N+1} \qquad (9)$$

Sabiranjem jednakosti $(8_0), \ldots, (8_N)$ dobijamo:

$$\begin{aligned} a_0 + \cdots + a_N = A_0 + \cdots + A_N - \\ -(10^h - 1)(\varepsilon_0 + \cdots + \varepsilon_{N-1}) - 10^h\varepsilon_N \end{aligned}$$

Dodajući zbir $a_{N+1} + \cdots + a_n$ na obe strane prethodne jednakosti i koristeći jednakost (9) dobijamo:

$$\begin{aligned} a_0 + \cdots + a_n = A_0 + \cdots + A_N - \\ - (10^h - 1)\,(\varepsilon_0 + \cdots + \varepsilon_{N-1}) \\ - \left[a_n\,(10^{h(n-N-1)} - 1) + \right. \\ \left. \cdots + a_{N+2}(10^h - 1)\right] \end{aligned} \qquad (10)$$

Pored toga, važi sledeća jednakost

$$a_0 + \cdots + a_n = f(1) = \alpha_0 + \cdots + \alpha_n = \sigma(S_f) \qquad (11)$$

i analogno

$$\begin{aligned} A_0 + \cdots + A_N &= F(1) = g(1) \cdot s(1) = \\ &= \sigma(S_g) \cdot \sigma(S_s) \end{aligned} \qquad (12)$$

Zato, saglasno uslovu (7) važi jednakost levih strana jednakosti (11) i (12), odnosno

$$a_0 + \cdots + a_n = A_0 + \cdots + A_N$$

Iz jednakosti (10) sledi jednakost

$$\begin{aligned} (10^h - 1)(\varepsilon_0 + \cdots + \varepsilon_N) + \\ + a_n\,(10^{h(n-N-1)} - 1) + \\ + \cdots + a_{N+2}(10^h - 1) = 0 \end{aligned}$$

Poslednja jednakost je jedino moguća (zbog $h \geq 1$ i ne-negativnosti prenosa ε_i i koeficijenta a_i) uz uslov

$$\varepsilon_0 = \varepsilon_1 = \cdots = \varepsilon_N = 0,$$

$$a_n = \cdots = a_{N+2} = a_{N+1} = 0$$

Ovim je dokazano da prilikom množenja spektralnih faktora S_g i S_s neće doći do prenosa ε_i iz pruge u prugu (za svako i), odnosno da važi

$$n = N, \quad a_0 = A_0, \quad a_1 = A_1, \ldots, a_n = A_N$$

To ujedno znači da, ako važi uslov (7), tada je ritam h saglasan sa operacijom množenja spektara S_g i S_s. (Ovo bi bio još jedan interesantan način za odredjivanje ritma h za proizvod dva spektra.)

Na osnovu ove teoreme, zaključujemo da spektralni faktori S_g i S_s dobijeni navedenim pravilom izbora dovode do uzajamno dopunskih faktora g i s polinoma f.

3. Primeri

Kao ilustraciju datog spektralnog metoda navodimo nekoliko jednostavnih primera rastavljanja pozitivnih polinoma na faktore.

PRIMER 1: Polinom $f = 6x^6 + 7x^5 + 4x^2 + x + 7$ ima ritam $h = 1$ i spektar $S = f(10^1) = 6700417$. Iz tablice prostih brojeva vidimo da je broj S prost. To znači da polinom f nema pozitivnih delioca.

PRIMER 2: Polinomu $f = 7x^6 + x^5 + 4x^4 + 7x + 6$ sa ritmom $h = 1$ odgovara spektar $S = f(10) = 7140076$. Aritmetički faktori broja S su 4 i 1785019, pri čemu 4 nije spektralni faktor spektra S. Zato se f ne može rastaviti na pozitivne faktore.

PRIMER 3: Polinom $f = 2x^5 + 5x^4 + 5x^3 + 8x^2 + 5x + 5$ ima ritam $h = 1$ i spektar $S = f(10^1) = 255855$. Prosti aritmetički faktori broja S su 3, 5, 37, 461, a svi mogući parovi uzajamno dopunskih spektralnih faktora jesu $S = 15 \cdot 17057 = 37 \cdot 6915 = 111 \cdot 2305 = 185 \cdot 1383 = 461 \cdot 555$. Potreban uslov (4) ispunjavaju prvi, treći i četvrti par, dok potrebne uslove (5) i (6) ispunjava treći par. Ovaj par zadovoljava i dovoljan uslov (7), tj: $2 + 5 + 5 + 8 + 5 + 5 = (1 + 1 + 1) \cdot (2 + 3 + 0 + 5)$. Zato su uzajamno dopunski pozitivni delioci polinoma f upravo

$$x^2 + x + 1, \quad 2x^3 + 3x^2 + 5.$$

PRIMER 4: Naći sve pozitivne delioce polinoma $f = x^4 + 5x^3 + 11x^2 + 13x + 6$. Ritam saglasan sa ovim polinomom je $h = 2$, a njegov spektar je $S = f(10^2) = 105111306$. Prosti aritmetički faktori broja S su 2, 3, 17, 101, 3401 i važi rastavljanje $S = 2 \cdot 3 \cdot 3 \cdot 17 \cdot 101 \cdot 3401$. Pošto su spektralni faktori brojevi ≥ 100, to će parovi uzajamno dopunskih spektralnih faktora biti:

$$\begin{aligned} S &= 101 \cdot 104706^* = 102 \cdot 1030503^* = \\ &= 153 \cdot 687002 = 202 \cdot 520353 = \\ &= 303 \cdot 346902 = 306 \cdot 343501 = \\ &= 606 \cdot 173451 = 909 \cdot 115634 = \\ &= 1717 \cdot 61218 = 1818 \cdot 57817 = \\ &= 3401 \cdot 30906 = 3434 \cdot 30609 = \\ &= 5151 \cdot 20406 = 6802 \cdot 15453 = \\ &= 10203 \cdot 10302^* . \end{aligned}$$

Potreban uslov (4) zadovoljavaju samo parovi markirani zvezdicom. Interesantno je da i dovoljan uslov (7) zadovoljavaju ovi parovi. Zato su parovi uzajamno dopunskih delioca polinoma f upravo:

$$\begin{aligned} f &= (x+1) \cdot (x^3 + 4x^2 + 7x + 6) = \\ &= (x+2) \cdot (x^3 + 3x^2 + 5x + 3) = \\ &= (x^2 + 3x + 2) \cdot (x^2 + 2x + 3). \end{aligned}$$

Kako je $(x + 1) \cdot (x + 2) = x^2 + 3x + 2$ to važi konačno rastavljanje

$$f = (x+1)(x+2)(x^2 + 2x + 3)$$

4. Rastavljanje proizvoljnog polinoma na faktore

Proizvoljan polinom oblika (1) možemo rastaviti na faktore na dva načina korišćenjem matematičkih spektara. Prvi način se zasniva na ideji Postnikova, a drugi način je nova spektralna metoda.

Prvi način: Posmatrajmo obostrano jednoznačno preslikavanje $f \mapsto f^*$ skupa polinoma f u skup polinoma f^* koji ispunjava uslove:

a) iz jednakosti $f = g \cdot s \Rightarrow f^* = g^* \cdot s^*$

b) ako za proizvoljan polinom f i polinome g' i s', za koje važi $f^* = g' \cdot s'$,postoje takvi polinomi g i s tako da važi $f = g \cdot s$ i $g^* = g'$, $s^* = s'$ (odnosno, $f = g \cdot s \Leftrightarrow f^* = g^* \cdot s^*$). Takvo preslikavanje nazivamo multiplikativni izomorfizam (2). Kao multiplikativni izomorfizam za rastavljanje proizvoljnog polinoma f na faktore koristićemo preslikavanje $f \mapsto f_c$, gde je $f_c(x) = f(x + c)$ i c neki fiksiran ceo nenegativan broj.

DEFINICIJA 5. Pozitivan polinom nazivamo strogo pozitivnim ako je svaki njegov delilac pozitivan.

TEOREMA 3: Za svaki polinom f oblika (1) postoji nenegativan broj c takav da je polinom f_c strogo pozitivan.

Dokaz teoreme zasniva se na činjenici da za broj c možemo uzeti najmanji ceo broj veći od realnih delova nula polinoma f.U tom slučaju, nule polinoma f_c imaće negativne realne delove, odnosno, polinom f_c se rastavlja na delioce sa pozitivnim koeficijentima (u opštem slučaju sa realnim koeficijentima). Na taj način, strogo pozitivni polinom f_c može se rastaviti na moguće pozitivne delioce, što je ekvivalentno sa zadatkom Jakovkina.

Sledeće pravilo nam daje prvi način rastavljanja proizvoljnog polinoma na faktore.

PRAVILO 2: Da bismo rastavili proizvoljan polinom f oblika (1) treba:

1) naći ceo nenegativan broj c, takav da je polinom $g = f_c$ strogo pozitivan,
2) rastaviti polinom g na moguće pozitivne delioce
$$g = g_1 \cdot g_2 \cdots g_r$$
3) dobiti delioce $f_1, f_2, \ldots, f_r$ polinoma f na osnovu formule
$$f_i = (g_i)_{-c}, \quad i = 1, 2, \ldots, r.$$

Za broj c možemo uzeti najmanji ceo broj veći od broja $1 + \frac{A}{a_n}$, gde je $A = \max_{i \in \{0,1,\ldots,n-1\}} \{|a_i|\}$, što je poznato iz linearne algebre.

Pored toga, broj c možemo odrediti kada su poznate približne vrednosti nula polinoma f ili za c uzeti proizvoljne cele brojeve i proveravati da li je polinom f_c strogo pozitivan nekim od kriterijuma [2].

Drugi način: Kod rastavljanja proizvoljnog polinoma f oblika (1) na uzajamno dopunske faktore g i s, možemo postupiti kao u odeljku 2, kod rastavljanja pozitivnog polinoma. Prvi problem je utvrditi ritam h koji je saglasan sa polinomima f, g, i s i sa operacijom množenja spektara polinoma g i s. Drugi problem jeste kako utvrditi da li smo našli uzajamno dopunske faktore g i s polinoma f na osnovu spektralnih faktora S_g i S_s spektra S_f polinoma f.

Prvi problem rešava ritam $h = \max\{H, h_1\}$, gde je H ritam datog polinoma f, a h_1 je najveći mogući ritam koji je saglasan sa faktorima polinoma f. Ritam h_1 faktora polinoma (1) se odredjuje po Orlovu [1] na sledeći način:

$$h_1 = \operatorname{int}(L_1) + 2 \qquad (13)$$

$$\begin{aligned} L_1 &= \log \binom{n-1}{v_1} + \\ &\quad + (n-1)\log(a_n + A_k) - (n-2)\log a_n \\ v_1 &= \operatorname{int}\left(\frac{n-1}{2}\right), \\ A_k &= \max_i \{|a_i|\}. \end{aligned}$$

Drugi problem rešavaju sledeće dve teoreme, pri čemu za prvu nemamo dokaz, pa je dajemo kao hipotezu.

Hipoteza: Polinomi g i s, asocirani sa uzajamno dopunskim spektralnim faktorima S_g i S_s broja $S = f(10^h)$, $h = \max\{H, h_1\}$, gde je H ritam polinoma f na osnovu (3) a h_1 dobijamo iz (13), jesu uzajamno dopunski faktori polinoma f ako i samo ako važe uslovi (4)-(7).

Za primenu ove hipoteze važi analogno pravilo i pravilo izbora kao i za pozitivne polinome.

Sledeća teorema bazira se na metodi pseudospektara i pogodna je takodje za primenu.

TEOREMA 4: Polinomi g i s, asocirani sa uzajamno dopunskim spektralnim faktorima S_g i S_s spektra S_f polinoma f sa saglasnim ritmom h, jesu uzajamno dopunski faktori polinoma f ako važi

$$f\left(10^{h+1}\right) = g\left(10^{h+1}\right) \cdot s\left(10^{h+1}\right) \qquad (14)$$

DOKAZ: Neka su S_g i S_s uzajamno dopunski spektralni faktori spektra S_f polinoma f sa ritmom h, koji jednoznačno odredjuju polinome g i s. Tada važi

$$S_g = g(10^h) \quad \text{i} \quad S_s = s(10^h)$$

odnosno,

$$f(10^h) = g(10^h) \cdot s(10^h) \qquad (15)$$

Dakle, ritam h saglasan je sa polinomima f, g i s. Pored toga, na osnovu (14) i (15) i metode pseudospektara ritam h je saglasan i sa operacijom množenja spektara polinoma g i s. Zato važi $f(x) = g(x) \cdot s(x)$

Napomena 1: Za primenu teoreme 4 zgodno je koristiti uslove (4)-(7) za izbor spektralnih faktora S_g i S_s spektra S_f polinoma f.

Napomena 2: Kod većine polinoma važi $h \geq h_1$, gde je h ritam polinoma f a h_1 ritam njegovih faktora. Zato za primenu hipoteze možemo na početku uzeti ritam h iz uslova (3), a zatim ga postupno povećati do teorijski moguće vrednosti h_1 iz uslova (13).

Na kraju navodimo nekoliko primera za ilustraciju teoreme 4 i hipoteze.

PRIMER 1: Rastaviti polinom $f = x^4 + 5x^3 + 3x^2 - 7x + 2$ na faktore (nerastavljive polinome sa celim koeficijentima). Ritam h saglasan sa f je broj $h = 2$ (jer je $10^2 > 2 \cdot \max_i\{|a_i|\}$), dok je spektar S polinoma f broj $S = f(10^2) = 105029302$. Razlaganjem broja S na proste faktore dobijamo $S = 2 \cdot 7 \cdot 19 \cdot 31 \cdot 47 \cdot 271$. Medju mogućim parovima uzajamno dopunskih spektralnih faktora broja S samo par $(S_g, S_s) = (10199, 10298)$ zadovoljeva uslove (4)-(7) hipoteze. Kako su nizovi efektivnih vrednosti pruga brojeva S_g odnosno S_s $1, 2, -1$ odnosno $1, 3, -2$, to je $g(x) = x^2 + 2x + 1$, $s(x) = x^2 + 3x - 2$

Napomena: U radu [1] za ritmove h_1 i h dobijeno je $h_1 = 5$, $h = 22$, što je u odnosu na našu vrednost $h = 2$ dosta neracionalno.

PRIMER 2: Rastaviti polinom $f = x^2 - 3x + 2$ na faktore. U ovom slučaju je $h = 1$ i $S = f(10^1) = 72$. Važi rastavljanje $S = 72 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3$. Uzajamno dopunski spektralni faktori broja S su $6 \cdot 12 = 8 \cdot 9$. Medjutim, samo par $8, 9$ zadovoljava uslove teoreme 4, jer važi jednakost (14), odnosno $9702 = 98 \cdot 99$ te su zato $g = x - 2$, $s = x - 1$ uzajamno dopunski faktori polinoma f.

PRIMER 3: Rastaviti polinom $f = 2x^3 + 49x^2 - 48x - 3$ na faktore. Ritam polinoma je h=2, te je njegov spektar $S = f(10^2) = 2485197$. Pri tome važi $S = 3 \cdot 3 \cdot 11 \cdot 13 \cdot 1931$. Jedini par uzajamno dopunskih spektralnih faktora broja S, koji zadovoljava uslove (4), (5) i (6) hipoteze, jeste par $S_g = 99$ i $S_s = 25103$, ali ne zadovoljava uslov (7). Zato, povećajmo ritam h za 1 i ponovimo postupak. Dobijamo: $h = 3$, $S = 2048951997$ i $S = 3 \cdot 3 \cdot 3 \cdot 37 \cdot 2051003$. Par uzajamno dopunskih spektralnih faktora broja S, koji zadovoljava uslove (4)-(7), jeste par $(Sg, Ss) = (999, 2051003)$ koji daje tražene faktore polinoma f:

$$g(x) = x - 1, \quad s(x) = 2x^2 + 51x + 3$$

Na analogni način, primena teoreme 4 dovela bi do traženih faktora polinoma f. Interesantno je da se koeficijent 51 faktora $s(x)$ "vidi" u spektralnom faktoru $S_s = 25103$, ali pošto je pruga 51 velika, ona dovodi do koeficijenata -49.

Zaključak

Primena matematičkih spektara skopčana je sa problemom izvodjenja računskih operacija sa "velikim" brojevima. Konstantin Orlov je primenu spektara realizovao na računskim mašinama (kalkulatorima), koristeći često operaciju "cepanja" spektara na dva ili više delova (6). B. Mihajlović je dao jedan program za realizaciju proizvoda dva spektra (7). Medjutim, do danas nije formirana programska podrška za automatski rad sa matematičkim spektrima. Stoga je interesantno definisanje programskog jezika za rad sa spektrima i izgradnja odgovarajućeg programskog sistema koji bi omogućio jednostavnu primenu matematičkih spektara. Realizacija ove ideje je predmet daljeg rada autora ovog članka.

Literatura

[1] K. ORLOV, Aritmetičke i analitičke primene matematičkih spektara, doktorska teza, Beograd, 1935.

[2] M. M. POSTNIKOV, Razloženie mnogočlenov na množiteli, Matematika v škole, No 3, Moskva, 1969, 18-24.

[3] M. PETROVIĆ, Les spectres mathematiques, Paris, 1919.

[4] M. PETROVIĆ, Lesons sur les spectres mathematiques, Paris, 1928.

[5] K. ORLOV, Aplikation pratique de la theorie des spectres mathematiques de M. Petrovitch au calcul numerique, La Revue scientifique, Fasc.4, Paris, 1953.

[6] K. ORLOV, Numerička spektralna aritmetika i algebra, Društvo matematičara, fizičara i astronoma Srbije, Beograd, 1973.

[7] B. MIHAJLOVIĆ, Realizacija proizvoda dva spektra na cifarskim računskim mašinama, Matematički vesnik, 4 (19), 1967, 119-122.

Adresa autora: Filozofski fakultet,
Grupa za matematiku,
18000 Niš
Ćirila i Metodija 2,

UDK 519.713:007.5

Mogućnosti primene konačnih automata u modeliranju aritmetičkih operacija

Miroslav Martinović

Kratak sadržaj: *Ovaj rad je na temu mogućnosti primene konačnih automata za modeliranje binarnog sabiranja i nemogućnosti primene kod binarnog množenja. Prvi deo rada je, u stvari, prezentacija konačnog automata koji proverava korektnost obavljenog binarnog sabiranja, dok drugi deo daje dokaz teoreme da proveru binarnog množenja nije moguće sprovesti ovim sredstvima. Dalje slede završni komentari i zaključci.*

Ključne reči: *konačni, automat, modeliranje, binarni, sabiranje, množenje.*

1. Uvod

Dobro je poznata činjenica da se konačni automati (a takodje i njihov ekvivalent — regularni jezici) mogu koristiti, a i vrlo široko koriste, u modeliranju mnogih komponenti računara. Leksička analiza u teoriji prevodilaca programskih jezika na primer, pokrivena je klasom regularnih izraza, koja je takodje pojam ekvivalentan klasi konačnih automata.

Sledeći slučaj vezan za problematiku modeliranja konačnim automatima, interesantan je kako sa praktičnog stanovišta mogućnosti izgradnje pojedinih komponenti, tako i sa teorijskog stanovišta, jer donoseći jedan prilično neočekivani rezultat govori o granicama do kojih je moguće koristiti ovaj koncept.

Prva činjenica je (i relativno poznata i očekivana) da se provera binarnog sabiranja može izraziti i izgraditi u terminima konačnih automata.

2. Modeliranje binarnog sabiranja konačnim automatom

Teorema 2.1: Postoji konačni automat koji vrši proveru binarnog sabiranja (*binary addition checker*).

Dokaz: Uzimajući za alfabet

$$A=\left\{\begin{pmatrix}0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix},\right.$$

$$\left.\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}1\\0\\1\end{pmatrix},\begin{pmatrix}1\\1\\0\end{pmatrix},\begin{pmatrix}1\\1\\1\end{pmatrix}\right\},$$

automat će prihvatati niske binarnih trojki

$$\begin{pmatrix}x_1\\y_1\\z_1\end{pmatrix}\begin{pmatrix}x_2\\y_2\\z_2\end{pmatrix}\cdots\begin{pmatrix}x_n\\y_n\\z_n\end{pmatrix}$$

ako je $z_1 \ldots z_n$ suma od $x_1 \ldots x_n$ i $y_1 \ldots y_n$ kada se ove niske tretiraju kao binarni brojevi (niz nula u početnom delu broja je dozvoljen). Niske se učitavaju sleva na desno.

Jednostavno se može pokazati da automat $K = (A, Q, \delta, F)$, gde je Q skup unutrašnjih stanja dat sa

$$Q = \{q_1, q_2, q_3\},$$

F skup završnih stanja dat sa

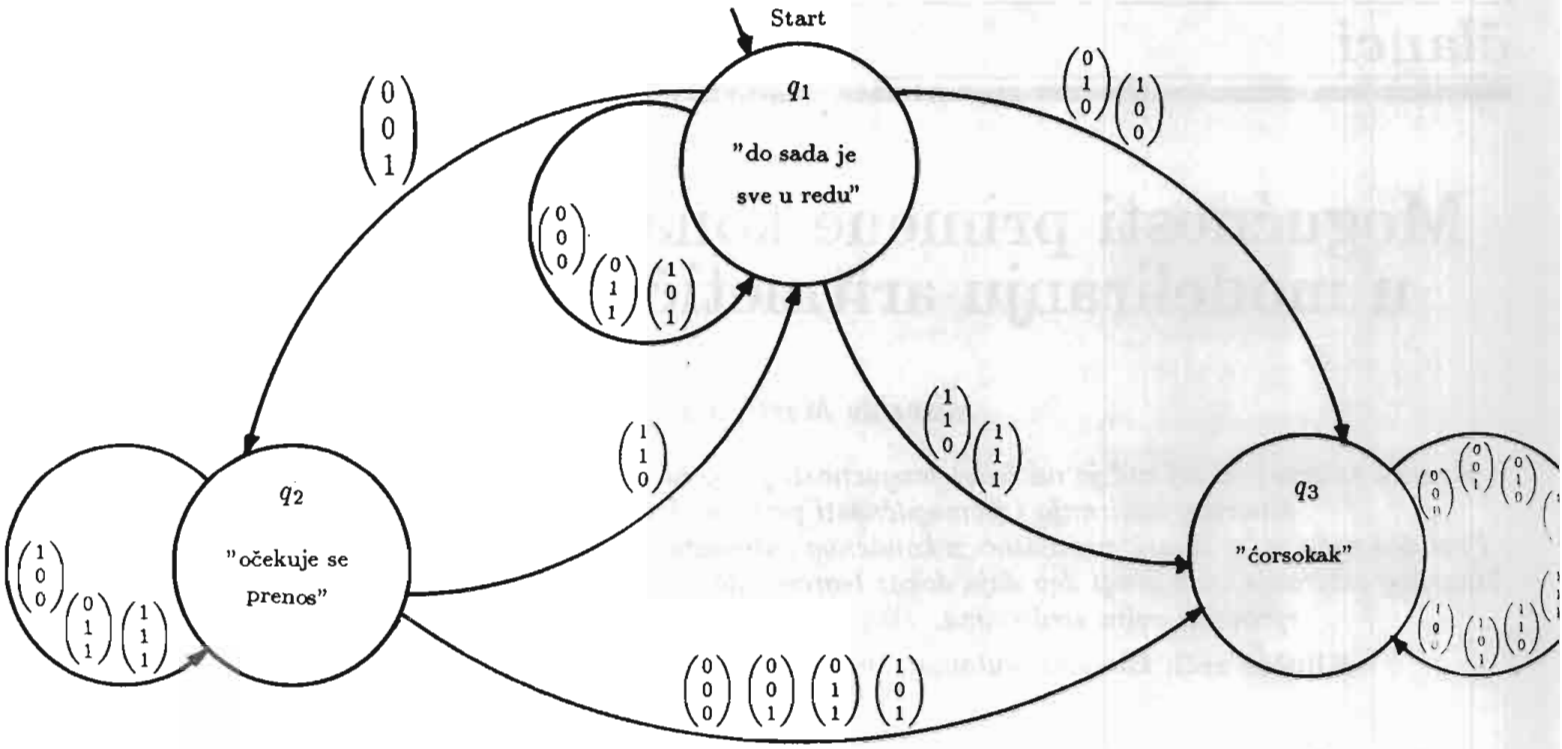
$$F = \{q_1\},$$

i δ funkcija prelaza data dole navedenim dijagramom, rešava ovaj problem.

Završno stanje je ujedno i početno stanje q_1, dok stanja q_2 i q_3 možemo okarakterisati sa "očekuje se prenos" (q_2) i "ćorsokak" (q_3) u skladu sa njihovom semantikom. Stanje q_1 je "do sada je sve u redu" stanje.

Sledeći primeri će ilustrovati rad ovog automata. Ispred i iza svakog simbola navodi se stanje u kome je automat bio pre i posle iščitavanja tog simbola, respektivno.

Prvi primer predstavlja korektno binarno sabiranje i, dakle, automat će okončati rad u svom



Slika 1. – Funkcija prelaza (δ)

završnom stanju q_1:

$$q_1 \begin{pmatrix}0\\0\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\0\end{pmatrix} q_1 \begin{pmatrix}1\\0\\1\end{pmatrix}$$
$$q_1 \begin{pmatrix}1\\0\\1\end{pmatrix} q_1 \begin{pmatrix}0\\0\\1\end{pmatrix} q_2 \begin{pmatrix}1\\0\\0\end{pmatrix} q_2 \begin{pmatrix}1\\1\\0\end{pmatrix} q_1.$$

Drugi primer nije binarno sabiranje, te automat ne staje u svom završnom stanju, već u stanju q_3, koje nije završno, pa, dakle, niska simbola i nije prihvaćena (automat je dospeo u "ćorsokak", od čega se ne može oporaviti):

$$q_1 \begin{pmatrix}0\\0\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\0\end{pmatrix} q_1 \begin{pmatrix}1\\0\\1\end{pmatrix}$$
$$q_1 \begin{pmatrix}1\\0\\0\end{pmatrix} q_3 \begin{pmatrix}0\\0\\1\end{pmatrix} q_3 \begin{pmatrix}1\\0\\0\end{pmatrix} q_3 \begin{pmatrix}1\\1\\1\end{pmatrix} q_3.$$

U trećem primeru automat konačno ostaje u stanju "očekivanja prenosa" q2, koje takodje nije završno, pa dakle niska opet nije prihvaćena:

$$q_1 \begin{pmatrix}0\\0\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\1\end{pmatrix} q_2 \begin{pmatrix}1\\1\\0\end{pmatrix} q_1 \begin{pmatrix}1\\0\\1\end{pmatrix}$$
$$q_1 \begin{pmatrix}1\\0\\1\end{pmatrix} q_1 \begin{pmatrix}0\\0\\1\end{pmatrix} q_2 \begin{pmatrix}1\\0\\0\end{pmatrix} q_2 \begin{pmatrix}1\\0\\0\end{pmatrix} q_2.$$

Korektnost ovog binarnog sabirača može se proveriti testiranjem njegove *Pascal* implementacije date u dodatku ovog teksta. Ova implementacija je uobičajena programska implementacija jednog konačnog automata.

3. Nemogućnost modeliranja binarnog množenja konačnim automatima

Na iznenadjenje, za proveru binarnog množenja moć konačnih automata se pokazuje nedovoljnom. S obzirom na činjenicu da se o množenju uglavnom razmišlja kao o "uzastopnom sabiranju", dakle nečem vrlo srodnom, razlika u njihovoj složenosti je upravo okarakterisana (ne)mogućnošću konačnih automata.

Teorema 3.1: Ne postoji konačni automat koji bi proveravao korektnost binarnog množenja.

Dokaz: Ukoliko bi takav automat postojao, on bi imao nekih k stanja i morao bi da prihvati i sledeću nisku, koja predstavlja korektno binarno množenje 2**k (2 na potenciju k) sa 2**k (sa 2**(2*k) kao

rezultatom), odnosno reč w, opisanu i datu sa:

$$w = \begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{k-1}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{k}.$$

Ovde je korištena oznaka

$$z^n \stackrel{\text{def}}{=} \underbrace{z\ldots z}_{n \text{ puta}},$$

za slovo ili reč z koja se ponavlja n puta uzastopce u nekom kontekstu.

U toku iščitavanja zadnjih k simbola automat je morao biti bar u jednom od svojih stanja dva puta. Dakle, možemo pisati:

$$\begin{pmatrix}0\\0\\0\end{pmatrix}^{k} = \begin{pmatrix}0\\0\\0\end{pmatrix}^{r}\begin{pmatrix}0\\0\\0\end{pmatrix}^{s}\begin{pmatrix}0\\0\\0\end{pmatrix}^{t}$$

gde je automat u stanju q i posle iščitavanja

$$\begin{pmatrix}0\\0\\0\end{pmatrix}^{r} \quad \text{i} \quad \begin{pmatrix}0\\0\\0\end{pmatrix}^{r}\begin{pmatrix}0\\0\\0\end{pmatrix}^{s} \quad (s \neq 0).$$

Dalje, pošto je w prihvaćena, s obzirom na prethodnu činjenicu automat će prihvatati i sledeću reč:

$$\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{k-1}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}0\\0\\0\end{pmatrix}^{r+t}.$$

Medjutim, očigledno da ovo ne predstavlja korektno binarno množenje!

4. Zaključak

Dakle, može se zaključiti da je množenje aritmetička operacija koja je prebrzo rastuća da bi se mogla modelirati konačnim automatima, jer pozicija koja se može promeniti binarnim množenjem može biti "isuviše daleko" od najjače potencije za operande (na "putu" koji je duži od broja stanja). U slučaju sabiranja, ono što se menja je u najgorem slučaju pozicija susedna najjačoj potenciji operanada, a ta razlika 1 je nešto što se može pokriti brojem stanja konačnog automata, koji je UVEK veći ili jednak jedan.

Dodatak

Programska implementacija binarnog sabirača.

```
PROGRAM BINSAB (INPUT,OUTPUT);

{$B-}

CONST MAXBROJCIFARA = 50;
      NULA = 0;  JEDAN = 1;
TYPE  BINCIFRE = NULA..JEDAN;
      NENEGATIV = NULA..MAXINT;
      SIMBOL = RECORD
         PRVI  :  BINCIFRE;
         DRUGI :  BINCIFRE;
         TRECI :  BINCIFRE
         END;
VAR NISKA : ARRAY [1..MAXBROJCIFARA] OF
            SIMBOL;
    I,N,BR: NENEGATIV;
    UREDU : BOOLEAN;

PROCEDURE UCITAVANJE;

BEGIN
   WRITELN ('UCITATI BROJ SIMBOLA ZA UN
   OSENJE');
   READ (BR);
   N:=0;
   WRITELN;
   WRITELN ('UCITATI SIMBOLE, CIFRU PO
   CIFRU');
   WRITELN ('SVAKI U PO JEDAN RED');
   WHILE N<BR DO BEGIN
      N:=N+1;
      WITH NISKA[N] DO BEGIN
         READ (PRVI, DRUGI, TRECI)
      END;
      READLN
   END;
   WRITELN;
   WRITELN ('UCITANA NISKA JE :');
   FOR I:=1 TO N DO WRITE (I, ' ');
   WRITELN;  WRITELN;
   FOR I:=1 TO N DO
    WITH NISKA[I] DO WRITE (PRVI,' ');
   WRITELN;
   FOR I:=1 TO N DO
    WITH NISKA[I] DO WRITE (DRUGI,' ');
   WRITELN;
   FOR I:=1 TO N DO
    WITH NISKA[I] DO WRITE (TRECI,' ');
    WRITELN
END;{UCITAVANJE}
```

```
FUNCTION Q3:BOOLEAN;

BEGIN
   WRITELN('NALAZIM SE U CORSOKAK STANJU');
   IF I=N+1 THEN Q3:=FALSE
   ELSE BEGIN
      WRITELN ('UCITAVAM',I,'. SIMBOL');
      WITH NISKA[I] DO BEGIN
      WRITELN (PRVI);
      WRITELN (DRUGI);
      WRITELN (TRECI)
   END;
   I:=I+1;
   Q3:=Q3
   END
END; {Q3}


FUNCTION Q2 : BOOLEAN; FORWARD;

FUNCTION Q1 : BOOLEAN;

BEGIN
   WRITELN (' NALAZIM SE U "DO SADA JE U
   REDU" STANJU');
   IF I=N+1 THEN Q1:=TRUE
   ELSE BEGIN
      WRITELN ('UCITAVAM',I,'. SIMBOL');
      WITH NISKA[I] DO BEGIN
      WRITELN (PRVI);
      WRITELN (DRUGI);
      WRITELN (TRECI);
      IF ((PRVI=0) AND (DRUGI=1)
            AND (TRECI=0)) OR
         ((PRVI=1) AND (DRUGI=0)
            AND (TRECI=0)) OR
         ((PRVI=1) AND (DRUGI=1)
            AND (TRECI=0)) OR
         ((PRVI=1) AND (DRUGI=1)
            AND (TRECI=1))
         THEN BEGIN
            I:=I+1;
            Q1:=Q3
         END ELSE IF ((PRVI=0) AND
                     (DRUGI=0) AND
                     (TRECI=1))
            THEN BEGIN
               I:=I+1;
               Q1:=Q2
            END ELSE BEGIN
               I:=I+1;
               Q1:=Q1
            END
         END
      END
   END
END; {Q1}


FUNCTION Q2;

BEGIN
   WRITELN ('NALAZIM SE U STANJU CEKANJA
 PRENOSA');
   IF I=N+1 THEN Q2:=FALSE
   ELSE BEGIN
      WRITELN ('UCITAVAM',I,'. SIMBOL');
      WITH NISKA[I] DO BEGIN
         WRITELN (PRVI);
         WRITELN (DRUGI);
         WRITELN (TRECI);
         IF ((PRVI=1) AND (DRUGI=0)
               AND (TRECI=0)) OR
            ((PRVI=0) AND (DRUGI=1)
               AND (TRECI=0)) OR
            ((PRVI=1) AND (DRUGI=1)
               AND (TRECI=1))
         THEN BEGIN
            I:=I+1;
            Q2:=Q2
         END ELSE IF ((PRVI=1) AND
                     (DRUGI=1) AND
                     (TRECI=0))
         THEN BEGIN
            I:=I+1;
            Q2:=Q1
         END ELSE BEGIN
            I:=I+1;
            Q2:=Q3
         END
      END
   END
END; {Q2}

BEGIN {BINSAB}
   UCITAVANJE;
   I:=1;
   UREDU:=Q1;
   WRITELN;
   WRITE (' NISKA PREDSTAVLJA ');
   IF UREDU THEN WRITE ('KOREKTNO ')
   ELSE WRITE ('NEKOREKTNO ');
   WRITELN ('BINARNO SABIRANJE .')
END.
```

Literatura:

1. Hopcroft J.E.,Ullman J.D. (1979)
"Introduction to Automata Theory,
Languages and Computation"
Addison-Wesley P.C.
2. Kosaraju S.R. (1974)
"Regularity Preserving Functions"
SIGACT News 6:2, 16-17.
3. Davis M.D.,Weyuker E.J. (1983)
"Compatibility, Complexity, and
Languages"
Academic Press, Inc.
4. Garey M.R.,Johnson D.S. (1979)
"Computers and Intractability :
A Guide to the Theory of NP-Completeness"
Freeman, New York
5. Harrison M. (1978)
"Introduction to Formal Languages Theory"
Addison-Wesley P.C.
6. Loveland D.W. (1978)
"Automated Theorem Proving :
A Logical Basis"
North-Holland P.
7. Rogers H. (1967)
"Theory of Recursive Functions and
Effective Computability"
McGraw-Hill, N.Y.
8. Seiferas, J.I.,McNaughton, R. (1978)
"Regularity Preserving Relations"
Theoretical Computer Science
9. Arden, D.N. (1960)
"Delayed Logic and Finite State Mashines"
Theory of Computing Mashine Design
10. Arbib, M.A. (1970)
"Theories of Abstract Automata"
Prentice Hall, N.J.

Adresa autora: Prirodno-matematički fakultet
Univerziteta u Beogradu
OOUR MMA
11000 Beograd
Studentski trg 16

UDK 519.682+800.92

Mizar MSE – sistem za kompjutersku podršku učenja osnova matematike

Roman Matuzewski

Kratak sadržaj: *U radu je dat kratak prikaz jezika MIZAR i opisane mogućnosti njegove primene u nastavi. Na kraju rada izložene su mogućnosti drugih primena ovog jezika.*

Ključne reči: *Mizar, računar, učenje, dokaz, dedukcija.*

1. Uvod

Familija jezika Mizar je kompjuterski orijentisana formalizacija matematike, a istovremeno je i semiotička rekonstrukcija jezika matematike. Namenjeni su pisanju matematičkih tekstova u obliku koji omogućuje automatsku verifikaciju ispravnosti rasudjivanja. Autor jezika je Andrzej Trybulec sa Warsaw University, Bialystok College.

Jedan od tih jezika je MIZAR MSE(1), jezik multisortnog računa predikata i reda sa jednakošću (*Multi Sorted with Equality*). Služi za učenje uz pomoć kompjutera u oblasti klasične logike, a takodje i nekih drugih delova matematike (npr. geometrije, teorije skupova itd.). Način rekonstruisanja dokaza zasnovan je na sistemima prirodne dedukcije (uslovni dokazi – *conditional proofs*) koji su tradicija poljske škole logike (Borkowski, Slupecki, Jaskowski).

2. Prikaz jezika MIZAR

Detaljan opis jezika Mizar MSE dodat je u referenci [1]. Za kratak prikaz jezika, u ovom radu, koristićemo se ilustracijom jezika na sledećim primerima:

```
   environ
      let x,y,z  denote  point;
      let K,L,M  denote  cercle;
      == "<>" je nejednakost
      == "in" je predikat
      ==  pripadanja
axiome: for  K,L  st
    K<>L
    ex x  st  in[x,K] & not in[x,L]
```

```
begin
      exercise: (ex K,L,M st
            K<>L & L<>M & M<>K)
          implies
          (ex x,y,z st
        c     x<>y & y<>z & z<>x)
      proof   given K,L,M
   such that
1:          K<>L & L<>M & M<>K;
   consider x such that
2:          in[x,K] & not in[x,L]
   by 1, axiome;
   consider y such that
3:          in[y,M] & not in[y,K]
   by 1, axiome;
   consider z such that
4:          in[z,L] & not in[z,M]
   by 1, axiome;
5:          x <> y
   by 1, 2, 3 ;
6:    '     y <> z
   by 1, 3, 4 ;
         x <> z
   by 1, 2, 4 ;
   hence    ex x, y, z  st
      x<>y & y<>z & z<>x
   by 5, 6
end ==dokaza
```

Tekst Mizar-a sastoji se iz dva dela. Prvi, koji počinje rečju **environ** predstavlja aksiomatiku koja se sastoji od aksioma teorije i rezervacije tipova za promenljive. Primetimo da taj deo teksta (aksiomatika) i pravila izvodjenja predstavljaju logički aparat teorije. U aksiomatici kompjuter proverava samo sintaksnu ispravnost.

U našem primeru rezervisana su dva tipa `point` i `circle` za promenljive i `axiom` koji govori da za dva razna kruga postoji tačka koja pripada prvom od njih i ne pripada drugom. U Mizaru MSE `for` je skraćenica za *for every*, `st` – *such that*, `ex` – *exist*, `<>` – simbol nejednakosti, `==` – je znak iza kojeg stoji komentar.

Drugi deo teksta počinje od reči `begin` i sastoji se pretežno od dokaza teoreme. U tom delu, kompjuter osim sintaksne ispravnosti proverava i logičku ispravnost grana dokaza, kao i ispravnost konstrukcije dokaza. U našem primeru dat je za dokazivanje sledeći fakt: Ako postoje tri razna kruga, tada postoje tri razne tačke. Oslanjajući se na `axiom`, tvrdjenje je dokazano uz polaznu pretpostavku o egzstenciji tri razna kruga. Zatim, uvedene su tri pomoćne tačke x, y, z, koje zadovoljavaju `axiom` (rečenice `2:`, `3:`, `4:`). Koristeći pravilo ekstenzionalnosti za jednakost u sledeća tri koraka dokaza, pokazano je da su te tri tačke medjusobno različite. Dokaz je završen konstrukcijom da postoje tri razne tačke, pri čemu je korišteno pravilo izvodjenja *existential generalization rule*.

Dokazivanje u Mizaru MSE je zasnovano na tri pravila konstruisanja dokaza: generalizacija (*generalization rule*), jednoj verziji teoreme o dedukciji Tarskog (*deduction rule*). Osim toga, nastaju takozvani pomoćni elementi dokaza kao posredni koraci u provodjenju rasudjivanja. Napisati ispravan dokaz u Mizaru, to znači napisati dokaz koji će biti akceptovan od strane checkera sistema Mizar MSE. Checker je procedura koja proverava:

1. Je li zaključak oblika:

$$\frac{\beta_1, \ldots, \beta_k}{\alpha}$$

za tekst u Mizaru MSE

$1:$ β_1 ;
$\vdots$
$k:$ β_k;
α by $1, \ldots, k$;

ispravan.

2. Je li teza dokazana.

Checker koristi pravilo *dictum de omni* i izmedju ostalog prihvata izvodjenja računa iskaza, pravila de Morgana za račun predikata, može raditi sa osnovnim osobinama jednakosti (refleksivnost, simetrija, tranzitivnost, ekstezionalnost).

Algoritmizacija proveravanja ispravnosti izvodjenja i rasudjivanja za ciljeve učenja, zahteva precizno definisanje i opis moći *checker*-a i istovremeno daje mogućnost izbora takvog checker-a kakav nam je potreban. Na taj način možemo precizno odrediti obim logičke umešnosti koju zahtevamo od učenika ili studenta.

3. Primena u nastavi

Praktično učenje pomoću Mizara MSE odvija se po sledećoj šemi:

1. Nastavnik priprema učenicima aksiomatiku potrebnu za rešavanje zadataka.
2. Učenik piše na kompjuteru, pomoću editora, tekst dokaza zadane teoreme.
3. Kompjuter analizira tekst i proverava dokumentaciju dokaza sa preciznom dijagnostikom grešaka, uključujući i greške u konstrukciji dokaza, a takodje govori o nedostacima u rasudjivanju.
4. Učenik, nakon eventualnih konstrukcija, popravlja i ponovo unosi tekst u kompjuter, ponavljajući taj ciklus sve dok ne dobije ispravan dokaz.

Na taj način realizovana didaktika daje učeniku vlastitog mentora kakav je kompjuterski sistem koji proverava rešavanje zadataka. Nastavnik je potreban za konsultaciju u slučaju ozbiljnijih grešaka.

Prezentovana metoda učenja pomoću kompjutera primenjivana je izmedju ostalih u:

- dopisnom kursu logike u matematičkom listu DELTA, Poljska [2],
- kursu diskretne matematike na University of Conecticut, USA [3],
- kursu metodologije fizike na University of Warsaw, POLJSKA,
- kursu formalnih sistema u informatici na University of Alberta, Canada.

Nabrojane praktične primene Mizara MSE i iskustva u vezi sa njima, dozvoljavaju mogućnost niza primena tog sistema:

- didaktika logike i nekih elementarnih teorija računa predikata (geometrija, klasična analiza),
- učenje konstruisanja ispravnih dokaza i njihovog redigovanja,
- razvoj intuicije raspoznavanja ispravnosti koraka rasudjivanja,
- razvoj razumevanja pojma dedukcije (npr. u metodologiji fizike),

- učenje logike kroz praksu ovladavanja pojmovima,
- alternativno učenje nekih matematičkih teorija (npr. aksiomatske verzije projektivne geometrije, nacrtne geometrije).

Izlažemo logiku kada pomoću nje želimo da naučimo slušaoce da ispravno provode rasudjivanje. Testiramo, dakle, logiku kao orudje rasudjivanja. To orudje moguće je osetiti kroz učenje konstruisanja ispravnih matematičkih dokaza. Korišćenje formalnog jezika pod kontrolom kompjutera zahteva od učenika izvodjenje rigoroznih rasudjivanja, verifikacije koje leme treba koristiti, koje pomoćne korake, koje pomoćne objekte.

U uskom smislu, Mizar MSE može biti primenjen za dokazivanje osobina programa, a takodje za dokumentaciju matematičkih radova [4].

4. Implementacija jezika

Mizar MSE implementiran je u jeziku PASCAL na sledećem kompjuterima: APPLE II, IBM PC, Amdahl, Rainbow, VAX, PDP-11/45, IBM 370/158, HP-9000, AMSTRAD/SCHNEIDER. Za IBM PC, Mizar MSE je dostupan kroz electronic mail: INFO - IBM PC SRI. ARPA.

5. Mogućnosti drugih primena

Provodjenje je takodje jedna od primena Mizara MSE. Matematički tekst zapisan u tom jeziku, preveden na prirodni jezik (*natural language*) može služiti kao didaktička pomoć pri učenju osnovnih teorema i teorija. U slučaju regionalnih jezika, korištenih od strane male grupe ljudi, brzo i jeftino automatsko prevodjenje moglo bi imati primene pri izdavanju priručnika iz oblasti osnova matematike.

Slične primene moguće su i u slučaju rasprostranjenih jezika, koji se karakterišu malom popularnošću van odredjenih granica (kineski [5], japanski, arapski). Sistem prevodjenja povezan sa checkerom, davao bi tekst pogodan za didaktiku, očišćen od meritoričkih grešaka i jasno sistematizovan.

Prevodjenje sa jezika Mizar MSE na naturalni jezik zasniva se na pretvaranju teksta Mizara na unutrašnji semantički oblik, a zatim sledi sinteza naturalnog teksta. Prevod se odnosi na jezičke konstrucije koje ulaze u sastav dokaza.

Na osnovu tih parcijalnih prevoda, gradi se prevod celog dokaza, a takodje prelaz sa strukture u zagradama (*parenthese*), svojstvne mašinskom tekstu, na strukturu u alineama (*alinea*), svojstvene tekstu naturalnog jezika. Važan je izbor odgovarajućeg bogatog rečnika prevodjenih izraza reči jezika Mizar MSE, svojstvenih gramatičkim konstrukcijama Mizara i karakterističnih jezičkih obrta matematičkog teksta. Translator izvodi strukturnu analizu ulaznog teksta i multiplikativnu analizu gramatičkih konstrukcija, u cilju promene strukture u zagradama u strukturu u alineama i istovremeno za odstranjivanje monotonosti dobivenog teksta. Istraživanja nad tim prevodjenjem provode se u Poljskoj [6] i USA [5].

Bibliografija

[1] Trybulec A.
- Jezyk informacyjno - logiczny Mizar MSE (Logic Iformation Language Mizar MSE), ICS PAS Reports Nr 465, Warszawa 1982, (in Polish)

[2] Mostowski M, Trubulec Z.
- A Certain Experimental Computer Aided Course of Logic in Poland, Proceedings of World Conference on Computer in Edocation IFIP/AFIPS, Norfolk 1985, North Holland, (to appear)

[3] Trybulec A, Blair H.
- Computer Aided Reasoning, Logic of Programs, Brooklyn 1985, Lecture Notes in Computer Science No 193, Springer - Verlag

[4] Cryszczyszyn H.
- An exemple of the use of computer in mathematical work, Studies in Logic, Grammar and Rhetoric, Bialystok 1984

[5] Qin Bin
- Translation Mizar MSE texts into Chinese, University of Connecticut T.R., 1985 (to appear)

[6] Matuszewski R.
- On Automatic Translation of texts from Mizar - QC Language into Polish, Studies in logic, Grammar and Rhetoric, Bialystok 1984, (in Polish)

[7] Matuszewski R.
- Mizar MSE - Enseignement des Fondements de la Mathematique appuye par Ordinateurs, Monastir (Tunis), 1986

Adresa autora: 02 - 792 Warszawa - Natolin
Lasek Bizozowy 15 m.9.
POLAND

UDK 371.6:681.32

Basic u početnoj nastavi računarstva i informatike

Nedeljko Parezanović

Kratak sadržaj: *Članak obradjuje probleme početne nastave računarstva i informatike zasnovane na programskom jeziku* BASIC. *U ovoj nastavi neophodno je uskladiti više ciljeva koji su prisutni u ovakvom kursu. To su opšteobrazovni cilj kursa, sticanje znanja iz računarstva i informatike potrebnog u nastavnu školovanja, učenje programiranja i algoritamskog rešavanja zadataka. Sve ovo mora biti realizovano kroz* BASIC*-jezik. Izlaganje u članku je zasnovano u udžbeniku RAČUNARSTVA I INFORMATIKE u okviru predmeta OTP za I razred srednje škole. Cilj članka jeste da nastavnicima pruži pomoć u realizaciji kursa, tako što će kroz članak sagledati bitne sadržaje kursa i njihovu vrednost sa gledišta, pre svega, opšteg obrazovanja, a zatim i drugih ciljeva kursa. U celom kursu* BASIC*-jezik se javlja kao sredstvo za realizaciju nastave, a ne kao objekat izučavanja u nastavi. Naravno, to se ne može u potpunosti postići, ali je neophodno koristiti iz* BASIC*-a one mogućnosti koje najbolje doprinose ovom cilju.*

Ključne reči: BASIC, *početna nastava, računarstvo i informatika, podatak, struktura podataka, struktura programa, mikroračunar.*

1. Uvod

U stručnim krugovima u svetu, pa i u nas, dosta se raspravljalo, a i sada se raspravlja, na temu prvog programskog jezika u obrazovanju. Odmah da bude jasno da ovaj članak nema te namere, iako autor smatra da je to interesantna i do kraja neiscrpna tema. Zapravo, u članku se polazi od toga da je BASIC taj prvi jezik koji se uči u školi. Razlog za to je jednostavna činjenica da je, po novim nastavnim programima, u našoj srednjoj školi BASIC predvidjen za prvo učenje programiranja i prvi susret je predvidjen nastavnim programom *Osnova tehnike i proizvodnje*, kao blok nastava u I razredu srednje škole. Kako ovaj predmet pripada grupi opšteobrazovnih predmeta, to i sadržaj računarstva i informatike, u okviru ovog predmeta, treba da ima opšteobrazovni karakter. Pored toga, izvesna znanja iz računarstva i informatike, koja će biti korisna u daljem obrazovanju u ovoj oblasti, moraju biti obuhvaćena i u *početnoj nastavi računarstva.* S druge strane, programom je predvidjeno da se izučava BASIC kao programski jezik. Naravno, to znači da će i neke specifičnosti BASIC-jezika biti neizostavno prisutne, koje sa gledišta prethodna dva cilja ove nastave neće biti bitne, a u nekim slučajevima biće i nepoželjne. Svih ovih elemenata, nastavnici, kao izvodjači nastave, moraju biti u potpunosti svesni i moraju imati u ovom pogledu čvrsta opredeljenja. Dakle, nastavnik mora znati koji sadržaj imaju opšteobrazovne vrednosti, koji su neophodni kao znanje iz računarstva, a koji su specifičnost BASIC-jezika.

1.1 Opšteobrazovni značaj računarstva

O opšteobrazovnom značaju računarstva postoje podeljena mišljenja. Od onih koji smatraju da se ova oblast može izučavati samo kao opštestručni ili užestručni predmet, do onih koji smatraju da se savremeno obrazovanje ne može zamisliti bez opšteg obrazovanja u oblasti računarstva i informatike. Ima dosta razloga zbog kojih treba verovati da je to oblast koja mora biti prisutna u opšteobrazovnim sadržajima savremene škole. Navešćemo neke od njih.

a) *Poznavanje računarstva i informatike postaje neophodno za razumevanje organizacije i funkcionisanja mnogih službi savremenog društva.*

Zaista, u savremenom društvu sve javne službe su opremljene računarima, sve evidencije gradjana

se nalaze u računarskim memorijama, celokupno finansijsko poslovanje banaka i drugih organizacija sa kojima gradjani svakodnevno komuniciraju opremljene su računarima. Radni ljudi se sreću sa računarom u proizvodnim halama, kancelarijama i gotovo svim drugim radnim mestima. Ovome svakako treba dodati i sve veću prisutnost računara u našim domovima.

b) *Programiranje osposobljava učenike da logički misle i ove svoje sposobnosti primene u rešavanju raznovrsnih zadataka iz škole i svakodnevnog života.*

Programiranje nije jednostavna primena gotovih algoritama, kao što se često misli, već se sastoji pre svega u iznalaženju algoritama za rešavanje odredjene klase problema. To je kreativan posao koji zahteva dobro razumevanje problema, kritički osvrt na postavku problema, a zatim stvaralački razvoj ideja za rešavanje problema, koji uvek podrazumeva apstraktni način mišljenja.

Na ovaj način programiranje, za koje redovno postoji unutrašnja i spoljašnja motivacija učenika, dobija izuzetan značaj u razvoju sposobnosti rešavanja problema kod učenika. Visoka unutrašnja motivisanost potiče od zadovoljstva konstruktorskog rada, jer se programiranjem računara da rade odredjen posao, učenik javlja kao konstruktor mašine u kojoj vidi ostvarenje svojih ideja. Spoljašnja motivisanost potiče iz svesti učenika da radi nešto što je moderno i veoma rasprostranjeno u svetu u kome živimo.

c) *Programiranje je kreativan posao koji kod učenika podstiče i razvija osobine, kao što su: upornost, tačnost, sistematičnost, urednost i smisao za samostalni rad.*

Razvoj postupka za rešavanje odredjene klase problema zahteva upornost, kao što je slučaj pri svakom traženju rešenja problema. U slučaju primene računara to je posebno potencirano traženjem efikasnih algoritama za rešavanje problema. Razvoj složenih algoritama ne može se uraditi bez odredjene sistematičnosti i urednosti u radu koja u ovom slučaju nije veštački nametnuta učeniku već uslov da se uspešno radi. Nije potrebno posebno naglašavati da se problem može rešiti pomoću računara samo vrlo tačnim prenošenjem uputstva za rešavanje problema računaru u vidu programa. Najmanje nepreciznosti u zapisu programa zahtevaće od učenika njihovo otkrivanje i otklanjanje u programu. Za ovakvu komunikaciju sa računarom neophodno je samopouzdanje učenika u svoje sposobnosti da samostalnim radom ostvari program na računaru. Za ovakvu upornost učenik je nagradjen neposrednim uvidom u ponašanje računara saglasno svojim ugradjenim idejama u program.

d) *Računari nalaze sve veću primenu u procesu izvodjenja nastave i učenja, pa je familijarizacija učenika sa njima osnovna pretpostavka za uspeh ovakve primene računara.*

Danas se ozbiljno računa na primenu računara u nastavi, kako za prezentaciju nastavnih materijala od strane nastavnika, tako i za uvežbavanje i učenje uz pomoć računara od strane učenika. Na ovom planu se ne može ništa ozbiljno učiniti bez poznavanja mogućnosti i načina korišćenja računara od strane svih nastavnika i učenika. To se može ostvariti samo masovnim osposobljavanjem učenika kroz opšteobrazovne sadržaje iz računarstva.

1.2 Znanja iz računarstva

U početnoj nastavi računarstva i informatike moraju se obraditi i izvesni pojmovi koji će predstavljati osnovu za dalje sticanje i proširivanje znanja iz računarstva. Neki od ovih pojmova mogu biti obradjeni u početnoj nastavi, ali mnogi se moraju postupno uvoditi, razvijati i produbljivati kroz programske sadržaje u višim razredima. Za ovo postoje dva razloga: prvi, što bi mnogo detalja u početnom kursu iz programiranja opteretilo nastavne sadržaje i umanjilo efikasnost usvajanja bitnih pojmova; drugo, što neki pojmovi ne mogu biti objašnjeni u potpunosti na početnom kursu s obzirom na uzrast učenika i njihovo matematičko znanje.

1.3 Specifičnosti BASIC-jezika

Programski jezik u početnom kursu iz računarstva i informatike ima vrlo važnu ulogu. Učenjem programskog jezika učenici dobijaju mogućnost prenošenja problema na računar, što u velikoj meri ima motivišuću ulogu u učenju sadržaja iz računarstva. Kroz programski jezik se najbrže uočavaju mogućnosti računara u obradi različitih tipova podataka, zatim različite strukture podataka i programa. Iako su ovo elementi koji su obeležja konkretnog programskog jezika, oni odražavaju i mogućnosti računara, jer se sve ove mogućnosti, pri izvršavanju na računaru uvek ostvaruju kroz mašinski jezik računara. Na taj način programski

jezik dobija oblik nosioca, a ne objekta nastave iz računarstva.

Medjutim, neke konstrukcije BASIC-jezika mogu kod učenika da deluju zbunjujuće, pa na njih treba posebno obratiti pažnju. Zapravo, učenici će moći da korektno koriste konstrukcije BASIC-jezika u različitim programskim situacijama samo ako o svakoj konstrukciji jezika steknu korektan semantički model u svojim mislima. Neke specifičnosti BASIC- jezika u smislu loše struktuiranosti i loše čitljivosti programa mogu biti prevazidjene veštinom nastavnika da na adekvatan način izlaže materiju. Za ovo je neophodno da nastavnik poznaje sve prednosti i nedostatke BASIC-jezika, što dobija posebnu težinu kad je reč o početnom kursu programiranja.

1.4 Način izvodjenja nastave

U nastavi računarstva i informatike treba se držati opštepoznatih didaktičkih principa, kao što su: aktivnost, naučnost, očiglednost, postupnost i sistematičnost. Ovde posebno treba istaći aktivnost učenika, s obzirom da će se često dogadjati da učenici istog razreda imaju različito predznanje iz računarstva. Nastavnik može ovakve situacije da iskoristi kroz dijaloški metod nastave i na taj način učini nastavu interesantnijom za sve učenike. U svakom slučaju, u nastavi programiranja poseban značaj pripada praktičnom radu učenika sa računarom.

Cilj ovog članka jeste da ukaže na sve navedene probleme u početnoj nastavi računarstva i informatike. Dakle, šta su opšteobrazovni sadržaji u ovoj nastavi, koja znanja imaju širi značaj u oblasti računarstva, o kojim specifičnostima BASIC-jezika treba posebno voditi računa, kao i kako izvoditi nastavu.

Pre nego što predjemo na izlaganje, moramo ukazati da će ono pratiti sadržaje obradjene u udžbeniku za I razred srednje škole [1]. U udžbeniku je materija podeljena u četiri dela:

1. SAMO POČETAK...
2. BOGATSTVO PODATAKA
3. KA MOĆNIJIM PROGRAMIMA
4. KORIŠĆENJE I ČUVANJE PROGRAMA

Prvi deo izlaže pojmove i materiju koji se odnose na prvi kontakt sa računarom kao informacionom mašinom. Drugi deo obradjuje različite vrste podataka koji omogućavaju rad sa brojevima, tekstom, grafikom i znakom. U trećem delu izlažu se elementarne strukture podataka i programske strukture. Četvrti deo obradjuje korišćenje spoljnih memorija za čuvanje gotovih programa, ali i za čuvanje tekućih verzija u procesu razvoja programa.

2. Samo početak...

Ovaj deo materije u udžbeniku ima opšteobrazovni značaj i predstavlja nezaobilazno znanje iz računarstva. U ovoj materiji ne postoje specifičnosti BASIC-jezika koje bi opteretile izlaganje detaljima koji nisu bitni sa gledišta pojmova i znanja koje treba preneti učenicima. Navešćemo neke bitne napomene sa gledišta realizacije programskih tema iz ovog dela gradiva.

2.1 Prvo upoznavanje sa računarom

Danas, zahvaljujući, pre svega, štampi, časopisima i televiziji, veliki broj učenika stiče izvesna znanja van škole. Tako će i većina učenika imati izvesno predznanje i o računarima. Medjutim, to znanje nije takvo da bi se moglo računati kao predznanje na koje treba dalje nadgradjivati nova znanja iz računarstva jer će ono biti vrlo raznoliko medju učenicima, kako po nivou, tako i po razumevanju osnovnih pojmova. Medjutim, to znanje nastavnik ne treba u potpunosti da ignoriše. Problem je sličan onome koji se javlja medju sedmogodišnjim prvacima od kojih neki znaju slova, neki znaju da čitaju i pišu, a neki ne znaju ništa od toga. Takva situacija zahteva da nastavnik na prvim časovima, kroz razgovor otkrije predznanja učenika i da ih dovede na zajedničku osnovu sa koje će svi dalje sticati nova znanja.

Na samom početku računar treba predstaviti kao i svaku drugu mašinu koja ima odredjenu namenu. Navesti primere raznih mašina i diskutovati njihovu namenu, naučiti šta je ulazna sirovina a šta rezultat obrade na mašini. Tako se lako dolazi do zaključka da je računar *informaciona mašina*, tj. mašina koja vrši obradu podataka. Ovde je neophodno uspostaviti vezu izmedju informacije i podataka. Objasniti na primerima pojam informacije i uvesti podatak kao zapis informacije. Prema tome, informacije se saopštavaju računaru, obradjuju i izdaju u obliku podataka. Tako se lako dolazi do funkcija pojedinih uredjaja računarskog sistema, kao što su ulazni organ, memorija, procesor i izlazni organ. U sastav računarskog sistema uvrstiti najmanji broj uredjaja, tj. one koje će učenici neposredno koristiti, a to su tastatura, računar i ekran. Uvesti pojam programa u intuitivnom smislu, koji će se kroz

dalje sadržaje produbljivati. Dakle, za razgovor o računaru moramo obraditi termine procesor, memorija, računar, računarski sistem, tastatura, ekran i program, uz to, svakako, i smisao prefiksa mikro za mikroprocesor, mikroračunar i mikroračunarski sistem.

Kada su učenici upoznali funkciju pojedinih uredjaja dobro je da izvrše i njihovo povezivanje, pri čemu im treba skrenuti pažnju na sve predostrožnosti pri povezivanju električnih uredjaja, pa i računara. Kada je računarski sistem povezan može se izvršiti njegovo uključivanje u električno napajanje, pri čemu učenici treba da znaju šta je *početna poruka sistema*, a nastavnik mora proveriti da li je ova poruka dobijena na svim ekranima u učionici. Učenicima se posebno detaljno mora objasniti tastatura i ekran, jer su to uredjaji preko kojih čovek komunicira sa računarom, pa je familijarizacija učenika sa ovim uredjajima bitna za dalji rad.

2.2 Prva komunikacija sa računarom

Za početak objašnjenja komunikacije sa računarom dobro je jednostavno tražiti od učenika da na računaru saberu dva cela broja. Nastavnik treba da pogleda različite ideje učenika da ostvare ovakvo sabiranje na računaru. Odmah će videti koji učenici imaju neko predznanje, a koji nemaju. U svakom slučaju, to će stvoriti lepu atmosferu za uvodjenje pojma programskih jezika. Velika prednost BASIC-jezika jeste što je to interpretativan jezik u kojem se naredbe mogu neposredno izvršavati. Ovo omogućuje da učenici već pri prvom kontaktu sa računarom rade zadatke koji imaju odredjen smisao. To ohrabruje učenike i podiže unutrašnju motivisanost učenika za učenje programiranja. To je osnovni razlog što u ovoj fazi treba ovladati upotrebom računara kao kalkulatora - tačnije, kao uredjaja za izračunavanje vrednosti aritmetičkih izraza. Ovo treba iskoristiti za dve stvari: prvo, da učenici bolje upoznaju tastaturu i ekran i drugo, da upoznaju pisanje brojeva, znakova aritmetičkih operacija i aritmetičkih izraza uopšte. Takodje je važno da učenici uoče da računar prihvata samo sintaksno ispravne poruke, a da na poruke koje ne može da "razume" odgovara sa izveštajem o sintaksnoj grešci. To je dobra prilika da se uoči da računar kontroliše samo sintaksnu ispravnost poruka koje prima, a da ne može otkriti semantičke greške u našim porukama.

2.3 Prvi program

Učenici već znaju intuitivnu definiciju programa, da je to uputstvo računaru kako treba da radi pri rešavanju odredjenog problema. Kada sastavimo prvi program, tada treba produbiti i definiciju programa. Da bi jedan spisak naredbi mogao da čini *program* mora biti jasno definisano sledeće:

1. koja je prva naredba programa
2. kako se po izvršenju jedne (osim poslednje) naredbe programa prelazi na sledeću naredbu i
3. koja je poslednja naredba programa.

Jasno, ovo je u BASIC-jeziku precizno definisano uvodjenjem brojeva programskih redova. Medjutim, u mnogim varijantama BASIC-jezika poslednja naredba programa je naredba sa najvećim brojem reda. Ovo ne treba koristiti, već kao poslednju naredbu obavezno pisati naredbu END. Naravno, ona će imati najveći broj programskog reda, ali će bez ikakvog dvoumljenja ukazati na fizički kraj programa. Dakle, učenicima treba dati samo ovakvu definiciju fizičkog kraja programa, pa zahtevatii od onih koji znaju za druge mogućnosti da koriste samo ovako definisan kraj programa.

Tako sve bitne elemente programa ima jednostavan zapis:

```
10 PRINT 1
20 PRINT 2
30 PRINT 3
40 END
```

Program počinje naredbom čiji je broj programskog reda najmanji, sledeća naredba je u programskom redu čiji je broj programskog reda prvi veći broj, a program se završava programskim redom sa naredbom `END`.

Objasniti *komande* za rad sa programima (`NEW`, `LIST` i `RUN`). Jasno razgraničiti razliku izmedju komande i naredbe BASIC-jezika, bez obzira što u nekim varijantama BASIC-jezika ova razlika praktično ne postoji. Dakle, na ovoj razlici treba strogo insistirati i kroz ceo kurs se strogo držati. Nikada u okviru programa ne koristiti komande. Na ovaj način se stiče navika nemešanja naredbi jezika i komandi programske sredine u kojoj se jezik koristi. U drugim programskim jezicima, osim BASIC-a, biće to komande operativnog sistema.

Odmah kada unesemo prvi program u memoriju računara moramo govoriti i uputiti učenike u jednostavne mogućnosti ispravki programa. Za ovo

se mogu koristiti standardne mogućnosti BASIC-interpretatora na nivou programskih redova. Medjutim, dobro je uvesti i minimalne mogućnosti uredjenja unutar programskog reda. Ovo obuhvata zamenu, izbacivanje i ubacivanje znakova u okviru programskog reda. Šire mogućnosti uredjenja koje često stoje na raspolaganju korisniku ne treba obradjivati, jer to nepotrebno opterećuje kurs.

Posebno značajan pojam u programiranju, kao i u matematici, jeste pojam *promenljive*. Promenljivoj može biti dodeljena izračunata vrednost ili vrednost sa ulaza. Važno je na primerima pokazati razliku izmedju programa koji sadrže ulazne veličine i onih koji ne sadrže ovakve veličine. Prvi daju rezultate koji zavise od ulaznih veličina, a drugi, pri svakom izvršavanju, daju iste rezultate.

Tako sledeći program uvek daje isti rezultat

```
10  A=47
20  B=12
30  C=A*B
40  PRINT C
50  END
```

Ali, program u kome promenljive `A` i `B` dobijaju vrednost sa ulaza

```
10 INPUT A
20 INPUT B
30 C=A*B
40 PRINT C
50 END
```

pri svakom izvršenju može da daje različite rezultate, što zavisi od dodeljenih vrednosti promenljivim A i B. Razlika je slična onoj u matematici izmedju aritmetičkog izraza (u našem primeru 47*12) i algebarskog izraza (u našem primeru A*B).

Ukratko, ovaj deo knjige upoznaje učenike sa sastavom računarskog sistema, funkcijama pojedinih uredjaja, načinom povezivanja uredjaja, uključivanjem sistema, posebno sa tastaturom i ekranom i načinom komunikacije sa računarom. Ovde se stiču prva znanja o programiranju, upoznaje se suština programskog rada i operacije nad programom, kao što je izvršavanje programa, listanje programa na ekranu i brisanje programa u memoriji. Na ovom nivou treba postići i što bolju familijarizaciju učenika sa računarom (tastaturom i ekranom). Naravno, ovo se može postići samo praktičnim radom učenika na računaru, što će se sprovoditi kroz ceo kurs.

3. Bogatstvo podataka

Kada je ostvaren prvi kontakt učenika sa računarom, može se preći na dalje upoznavanje mogućnosti računara. To je najbolje ostvariti preko različitih tipova podataka. Jak razlog za ovo je jednostavna činjenica da se naše razmišljanje i očekivanje o mogućnosti računara zasniva na podacima, a ne na programu i strukturi programa. Dakle, prirodno je nastavu vezivati za podatke i probleme, a ne za pisanje programa što je samo posledica zahteva za odredjenim obradama podataka. Za početni kurs programiranja izbor podataka je najbolje zasnovati na brojevima, tekstom, grafikom i znakom. Ovo je učenicima blisko i interesantno, pa i najviše motivišuće za rad sa računarom.

3.1 Rad sa brojevima

Učenici su upoznali različite zapise brojeva, brojčanih promenljivih i brojčanih izraza. Kao dalju nadgradnju rada sa brojevima treba upoznati rad sa brojčanim funkcijama i posebno tačnost računanja. Izbor brojčanih funkcija treba izvršiti saglasno uzrastu učenika, a takodje i izlaganje o tačnosti računanja. I jedno i drugo će se kroz dalje školovanje svakako produbljivati, ali ove probleme ne treba izostaviti iz početnog kursa iz dva razloga: prvo, što će učenici često uočiti, u praktičnom radu sa računarom, da su neka izračunavanja približna; drugo, što je tačnost računanja nezaobilazna opšteobrazovna tema kada je reč o radu sa brojevima.

U ovom delu udžbenika obradjene su još dve teme: preglednost ispisa brojeva na ekranu i programski ciklusi. Prva tema doprinosi boljoj čitljivosti ispisa na ekranu, na kojoj se mora insistirati u cilju unapredjenja komunikacije korisnika sa računarom. Na taj način se razvija osećaj učenika šta je dobro, a šta loše, sa gledišta dobre komunikacije, što je deo opšte kulture o korišćenju računara. Druga tema, programski ciklusi, omogućuje pisanje interesantnijih programa od strane učenika, čime se doprinosi većoj motivisanosti za rad. Medjutim, naravno, ova tema produbljuje znanje učenika u tehnici programiranja. Važno je istaći da je sa gledišta postupnosti uvodjenja programskih struktura u nastavi, opravdano uvesti programski ciklus pre razgranate strukture. Ovo je tačno samo ako se programski ciklus definiše posebnim naredbama, bez eksplicitnog navodjenja izlaznog kriterijuma, što je slučaj sa naredbama `FOR...NEXT` u BASIC-u.

Jednostavni primeri mogu lepo ilustrovati potrebu i efikasnost programskih ciklusa u pisanju programa. Navodimo kao primer izračunavanje kvadrata prvih 10 prirodnih brojeva:

```
10 FOR I=1 TO 10
20 PRINT I, I*I
30 NEXT I
40 END
```

Alternativa za ovako elegantan ciklus bilo bi ponavljanje naredbe **PRINT** u obliku:

```
10 PRINT 1, 1*1
20 PRINT 2, 2*2
.
.
.
100 PRINT 10, 10*10
110 END
```

Prednosti su očigledne, a biće još ubedljivije ako naš program hoćemo da radi ne za 10, već za 100 prvih prirodnih brojeva. Ostavite učenicima da isprave gornji program u ovom smislu.

3.2 Rad sa tekstom

Za opšte obrazovanje o mogućnosti računara, rad sa tekstom ima vrlo značajno mesto. Zapravo, u mnogim primenama računara rad sa tekstom ima centralno mesto. To je slučaj i sa mnogim ličnim korišćenjem računara, gde računar zamenjuje pisaću mašinu. U ovom delu udžbenika sa minimumom naredbi i funkcija BASIC-jezika izložene su mogućnosti unošenja, obrade i izdavanja teksta.

Uz rad sa tekstom izložene su i mogućnosti organizacije programske kolekcije podataka. Ovo će omogućiti da se u mnogim primerima sa tekstom izbegne unošenje teksta, što bi zadatke učinilo neprirodnim, dosadnim i neinteresantnim, posebno pri izvršavanju programa. Sa druge strane, ovo je dalje produbljivanje znanja iz programiranja, koje je ravnomerno distribuirano kroz ceo udžbenik.

Pogledajmo jednostavan program koji ilustruje korist od programske kolekcije podataka:

```
10 U=0
20 FOR I=1 TO 5
30 READ D$
40 PRINT "UNESITE PROIZVODNJU ZA
   ";D$
50 INPUT P
60 U=U+P
70 NEXT I
80 PRINT "PROSECNA DNEVNA PROIZV
   ODNJA:"; U/5
90 DATA PONEDELJAK,UTORAK,SREDA,
   CETVRTAK,PETAK
99 END
```

Da bismo ostvarili ovakvu komunikaciju korisnika sa programom bez korišćenja programske kolekcije podataka morali bismo ponavljati telo ciklusa 5 puta!

3.3 Rad sa grafikom

Korišćenje grafičkih mogućnosti računara moguće je na različitim obrazovnim nivoima, jer su naredbe za grafiku vrlo jednostavne i prihvatljive i za najmladji uzrast učenika. Tako da grafika može da zauzme značajno mesto u početnim kursevima iz programiranja, što je i slučaj u nekim programskim jezicima (npr. LOGO). Složenost rada sa grafikom ogleda se u traženju efikasnih algoritama u različitim grafičkim prikazima, posebno kada je reč o trodimenzionalnoj grafici. Ozbiljno ograničenje u radu sa grafikom predstavlja raspoloživa realizacija na mikroračunaru koji se koristi u nastavi. Kako je niska rezolucija (znakovna) prisutna kod svih mikroračunara, to je za početni kurs izabrana ova vrsta rezolucije. To ima neke prednosti i nedostatke. Prednost je što se rad sa grafikom uvodi kroz naredbe koje učenici već poznaju, dakle ne proširuje se skup naredbi. Na ovaj način grafika se javlja kao izvanredna prilika da se uvežbavaju već naučene naredbe, ali i da se pišu programi, posebno oni sa različitim programskim ciklusima. Svi bitni pojmovi u grafici, kao što su elementarni grafički objekti, složeni crteži i animacija mogu se demonstrirati. Posebno je značajno uočiti diskretnost slike koja se proizvodi pomoću računara. Jedini nedostatak ovog pristupa jeste relativno loš kvalitet slike uslovljen niskom rezolucijom, što umanjuje izbor zadataka iz grafike, a i interesovanje učenika za rad sa grafikom. Ovde svakako ne treba zaboraviti da najčešće na početnom kursu ne postoji ni potrebno matematičko predznanje za ozbiljniji rad u grafici srednje i visoke rezolucije. Otud, znakovna grafika predstavlja dobro rešenje za sticanje utiska o grafičkim mogućnostima računara, bez većeg matematičkog znanja neophodnog za ozbiljniji rad sa grafikom.

Tako sledeći program izdaje 11 grafičkih znakova po dijagonali ekrana počev od gornjeg levog ugla ka donjem desnom uglu ekrana:

```
10 CLS
20 FOR I=0 TO 10
30 LOCATE I, I
```

```
40 PRINT CHR$ (219)
50 NEXT I
60 GOTO 60
```

Jedno upozorenje. U ovom programu je uzeto da se prva pozicija prvog reda zadaje sa (0, 0), što kod nekih mikroračunara nije slučaj, već se ova pozicija zadaje sa (1, 1). To morate znati da biste dobijali crteže na ekranu onako kako ih zamislite.

3.4 Rad sa zvukom

Postojanje zvuka na mikroračunarima omogućuje učenicima da pišu programe sa zvučnim efektima ili da programiraju melodije. Naročito treba istaći da programiranje zvuka omogućuje svim učenicima da eksperimentišu sa zvukom, pa i onima koji nikada do tada nisu uspeli da odsviraju bilo kakvu melodiju na nekom od muzičkih instrumenata. U udžbeniku se koristi elementarna naredba za generisanje zvuka u kojoj su parametri frekvenca i trajanje zvučnog signala. Tako se učenici vezuju za fizička svojstva zvuka, a muzički tonovi se javljaju kao posebna vrsta zvuka odredjene učestanosti.

Programiranje melodija treba iskoristiti da učenici uoče prednost čuvanja podataka u memoriji (naredbom DATA) u odnosu na unošenje podataka preko tastature. Jednostavno čovek ne može postići unošenje učestanosti i trajanja tona sa ulaza u cilju sviranja odredjene melodije. Trajanje pojedinih tonova u melodiji je znatno kraće od potrebnog vremena za unošenje parametara tona preko tastature. Ako se parametri tona čuvaju u memoriji, uzimaće se onda kada budu potrebni u izvodjenju melodije. To je izvanredno lepa prilika da se učenicima ukaže da neki procesi diktiraju brzinu rada računara i računar ih mora pratiti. Na ovaj način se može lepo objasniti i demonstrirati rad računara u **realnom vremenu**. To je jedna od važnih osobina savremenih računara koja se sreće pre svega u problemima upravljanja procesima, kontroli eksperimenata, upravljanja letilicama i sl.. dakle, primenama koje se teško mogu bliže objasniti i demonstrirati učenicima.

Ovo lepo ilustruje razlika izmedju programa koji svira skalu tonova unetih sa ulaza:

```
10 FOR I=1 TO 8
20 INPUT F
30 SOUND F, 10
40 NEXT I
60 END
```

i programa koji istu skalu svira koristeći podatke iz programske kolekcije podataka:

```
10 FOR I=1 TO 8
20 READ F
30 SOUND F, 10
40 NEXT I
50 DATA 262, 294, 330, 349, 392,
   440, 494, 523
60 END
```

U oba slučaja treba svirati skalu sa četvrtinskim tonovima. Medjutim, u prvom programu to se može ostvariti sa velikim razmacima izmedju tonova, jer se mora čekati na sporu manipulaciju korisnika preko tastature. U drugom programu tonovi se uzimaju onom brzinom koja je potrebna.

4. Ka moćnijim programima

Važna karakteristika programa sa kojima se sreću i koje mogu da pišu učenici u prva dva dela udžbenika jeste jednostavna struktura. To su programi u kojima se nižu naredbe i programski ciklusi. U ovim programima nema naredbi grananja što ih čini jednostavnim za pisanje i razumevanje. Medjutim, uvodjenje naredbi grananja dovodi do složenih programskih struktura, a to znači da će biti potrebno mnogo više napora da se sagleda postupak za rešavanje zadataka, a tek zatim, kada smo ovo uradili, možemo pristupiti pisanju programa. Kako je sada reč o složenijim programima, to se mora povesti računa i o tome kako je zapisan program, što dovodi do razmatranja programskog stila.

4.1 Programski stil

Razmatranje programskog stila mora se zasnovati na znanju učenika u pisanju programa. To se može učiniti ilustracijama različitih zapisa konstanti, promenljivih, izraza, naredbi i programskih struktura (ciklusa). Kroz ove ilustracije treba pokazati zapise koji su lošiji i one koji su bolji sa gledišta dobrog stila. Ovo je jedan aspekat programskog stila. Medjutim, za pisanje složenijih programa mora biti jasno da je presudno naše razmišljanje o stukturi postupka koji kreiramo za rešavanje odredjenog problema. Ova struktura postupka (algoritma) najbolje se uočava korišćenjem algoritamskih šema. Vizuelna predstava strukture algoritama ima veliku pedagošku vrednost. Medjutim, ovo je tačno samo do izvesnog stepena složenosti šeme, jer veća složenost šeme umanjuje vizuelnu preglednost pa i smisao njenog korišćenja.

Tako, razmatranje programskog stila u smislu pisanja čitljivih programa i korišćenja algoritamskih šema za sagledavanje strukture postupaka za

rešavanje problema, predstavlja uvod u pisanje složenijih programa.

4.2 Razgranati programi

Naredba kojom se ispituje istinitost odredjenog uslova omogućava da programiramo računar da rešava zadatke u kojima se vrši izbor i donosi odluka. Ovo je jedna od suštinskih mogućnosti računara koja se može dobro razumeti samo kroz primere programa u kojima se vidi vrednost ovakve mogućnosti računara. Tako se može razumeti kako računar nalazi najbolje od mogućih rešenja ili kako u velikom skupu podataka pronalazi onaj koji nas interesuje i sl. Razumevanje ove mogućnosti računara ima vešestruki značaj. Opšteobrazovni, jer daje pravu sliku o mogućnosti računara; proširuje znanje iz programiranja i omogućuje rešavanje znatno složenijih i interesantnijih problema na računaru.

Neophodno je obraditi različite vrste relacijskih izraza, koji će pokazati da računar može da ispituje uslove brojčanog karaktera, ali isto tako uslove u kojima se poredi tekst. Dok je poredjenje brojčanih vrednosti sasvim razumljivo, na osnovu matematičkog predznanja učenika, to poredjenje teksta zahteva mnogo veće objašnjenje. Prvo, mora se razumeti prikazivanje teksta u *ASCII-kôdu*, i drugo, mora se razumeti način ispitivanja da li su dva znakovna podatka u navedenoj relaciji ili nisu. Za ovo uvežbavanje dobro je navoditi kratke primere programa koje će učenici čitati i analizirati njihove uslove. Posle usmene analize ovih programa od strane učenika treba pokazati rad programa na računaru. Tako, sledeći program

```
10 INPUT A$
20 INPUT B$
30 IF A$<B$ THEN PRINT "JESTE ";
   ELSE PRINT "NIJE ";
40 PRINT A$; "<"; B$
50 GOTO 10
```

učenici treba usmeno da objasne šta radi, a zatim da odgovore šta je rezultat rada programa za sledeće ulazne znakovne zadatke:

a)	BOR		b)	BEOGRAD
	BAR			BOR
c)	456	d)	BOR	e)	TIVAT
	178		1785		TISA

Zapisati neke odgovore učenika na tabli, naročito pogrešne, a zatim izvršiti program i porediti odgovore učenika sa onim dobijenim na računaru. Sve pogrešne odgovore analizirati i objasniti kako se dolazi do tačnog odgovora. U gornjem programu promeniti relacijski znak, a zatim ponoviti prethodnu analizu za već navedene primere ili neke druge. Na kraju, sastaviti program koji omogućava unošenje i relacijskog znaka sa ulaza. Ovakav program omogućava različita poredjenja unetih reči i predstavlja korisnu vežbu u pisanju programa i uvežbavanju poredjenja znakovnih podataka.

Ako verzija BASIC-jezika raspolaže naredbom `IF...THEN...ELSE`, tada ovu naredbu treba prvo obraditi, a naredbu `IF..THEN` izvesti kao poseban slučaj naredbe `IF..THEN..ELSE`. Pri pisanju programa primenjivati uvlačenje opcije ELSE, čak i u slučaju kada interpretator ne dozvoljava uvlačenje redova. Ovo će učiniti preglednijim zapis programa na papiru, ako već nije moguće na ekranu računara, iako svaka bolja verzija BASIC-a dozvoljava uvlačenje redova.

4.3 Nizovi podataka

Nizove podataka je vrlo jednostavno uvesti za učenike koji znaju pojam matrice. Medjutim, za mladji uzrast to je nešto teže. Zato je najbolje pojam niza objasniti potrebom za organizovanjem prolaska preko većeg broja promenljivih u okviru programskog ciklusa. Evo primera. Mi možemo napisati naredbu za sabiranje vrednosti 10 promenljivih:

$$Y = A + B + C + D + E + F + G + H + I + J$$

Ali, šta da radimo ako treba da saberemo vrednosti stotinu promenljivih. To se u okviru programskog ciklusa i niza jednostavno rešava:

```
10    FOR I=1 TO 100
20       Y=Y+A(I)
30    NEXT I
```

Ovde treba uvesti pojam *strukture podataka*, iako se učenici neće upoznati sa drugim strukturama podataka. Medjutim, nizom se zapravo najbolje i ilustruje suština i svrha organizovanja podataka u strukturu. Ukratko se može reći da su strukture podataka logične celine koje omogućavaju lakše manipulisanje sa podacima i izražavanje potrebnih obrada podataka koji čine strukturu.

Osnovna svojstva niza, kao strukture podataka, jesu da se u svakom elementu u nizu može nezavisno pristupiti, pomoću indeksa, kao i da svi podaci u nizu moraju biti iste vrste (tipa). Što se tiče dimenzije niza, treba obraditi jednodimenzione i dvodimenzione nizove. Ovakve nizove podržavaju sve verzije BASIC-jezika. Medjutim, čak ako verzija BASIC-jezika dozvoljava i višedimenzione nizove ne

treba ih uključiti u nastavu. Jednostavno zato, što je rad sa jednodimenzionim i dvodimenzionim nizovima očigledan, dok rad sa višedimenzionim zahteva izvestan nivo apstrakcije, a i u praktičnim primenama redje se koristi.

4.4 Potprogrami

Koncept potprograma u programiranju ima višestruki značaj. To je koncept koji je uveden u programiranje pre svega u cilju uštede memorijskog prostora. Razvojem programskih jezika ovaj koncept je dobio i drugi značaj, koji se ogleda u mogućnosti izgradnje biblioteke potprograma i u primeni modularnog programiranja.

Koncept potprograma u BASIC-jeziku je, u većini verzija ovog jezika, loše ostvaren. Nemogućnost prenosa parametara potprograma posredstvom fiktivnih i stvarnih parametara umanjuje bitna svojstva potprograma u programiranju. To otežava stvaranje i praktično korišćenje biblioteka potprograma, a takodje i razvoj složenih programa korišćenjem ideje modularnog programiranja. Ono što se ipak može učiniti, jeste uvodjenje koncepta potprograma u rešavanju složenih zadataka, razbijanjem zadataka na podzadatke. Svakom podzadatku odgovara jedan potprogram. Posebno je važno da se uoči veza izmedju programa i potprograma, kao i kako ova veza može biti ostvarena. Na ovaj način, učenici će se upoznati sa konceptom potprograma i lako će ga prihvatiti i u drugim programskim jezicima.

5. Korišćenje i čuvanje programa

Problem korišćenja i čuvanja programa učenicima postaje brzo očigledan i to iz dva razloga: prvi je potreba korišćenja gotovih programa, a drugi čuvanje sopstvenih programa. Korišćenje gotovih programa pripada opšteobrazovnom sadržaju. Ako nastavnik raspolaže gotovim programima koje može dati učenicima na korišćenje, treba da osposobi učenike za njihovo korišćenje odmah na početku kursa. To će obogatiti kurs novim sadržajima i povećati interesovanje učenika. Potreba da učenici čuvaju sopstvene programe nastaje u trenutku kada njihovi programi postanu toliko opširni da ih ne mogu razviti u toku jednog školskog časa. Tada treba sačuvati tekuću verziju programa zbog nastavka rada na programu na sledećem času. Drugi razlog za čuvanje programa jeste sopstveni razvoj interesantnih programa koje učenici mogu razmenjivati. To će, najčešće, biti slučaj sa programima koji sadrže grafičke mogućnosti i rad sa zvukom. Tako da je dobro, pre ovih lekcija, osposobiti učenike za čuvanje programa na spoljnim memorijama. Prema tome, sadržaji o korišćenju i čuvanju programa, u udžbeniku su obradjeni kao četvrti deo, ali u realizaciji nastave mogu slobodno biti rasporedjeni po nahodjenju nastavnika, od početka do kraja kursa. To zavisi od raspoloživosti gotovih programa i vrste spoljnih memorija.

U svakom slučaju, ovaj sadržaj treba iskoristiti i za to da se učenici upoznaju sa vrstama spoljnih memorija (magnetnom trakom i disketom), kao i štampačem kao izlaznim organom. Magnetna traka ima prednost što je jeftin medijum, tako da učenici mogu imati svoje lične kasete, ali ima i nedostatak da je spor i manje pouzdan medijum što može da znači veliko zadržavanje u izvodjenju nastave. Zato je disketa bolji medijum, pa kad god je to moguće treba koristiti disketu pre nego kasetu. Na tržištu se nalaze dve vrste disketa od 3.5 inči i 5 1/4 inči. Za rad sa učenicima bolja fizička svojstva ima disketa od 3.5 inči, jer nije podložna oštećenjima kao što je to disketa od 5 1/4 inči. Naravno, nastavnik tu ne može ništa učiniti, to zavisi od disketne jedinice sa kojom raspolaže naš mikroračunarski sistem.

Literatura

1. N. Parezanović: OTP - računarstvo i informatika, udžbenik za I razred srednjeg usmerenog obrazovanja i vaspitanja, za sve struke, Naučna knjiga, Beograd i Zavod za izdavanje udžbenika, Novi Sad, 1987.

Adresa autora: Prirodno-matematički fakultet,
11000 Beograd
Studentski trg 16

UDK 681.32.06/08

"Stari" i "novi" programi

Duško Vitas

1. Problem

Već sredinom šezdesetih godina je uočeno da testiranje, kao postupak verifikacije programa, predstavlja kritičnu tačku u razvoju i održavanju programa. Količina vremena koja se u to doba trošila na testiranje programa je iznosila i do 70% ukupno potrebnog vremena za razvoj jednog programskog sistema. Pri tome, kao što je Dejkstra primetio [2], "testiranjem se može uočiti prisustvo mušica ali nikako ne i njihovo odsustvo". I najbrižljivije testirani programi su u kritičnim fazama svoje eksploatacije otkazivali zbog pojave mušica (na primer, u misiji svemirskih istraživanja "Apolo").

Pojava mušice, koja se često naziva i programska greška, predstavlja odstupanje onoga šta program zaista radi od onoga što bismo mi želeli da on radi. U samom programu, kao ni u računaru, ne postoji ništa što bi ukazivalo na pojavu mušice: i računar i program funkcionišu besprekorno ali ne onako kako smo mi to očekivali. Problem potiče otuda što se faktičko značenje napisanog programa razlikuje od onog značenja koje smo želeli da mu dodelimo. Stoga su nastojanja da se postupak testiranja prevazidje kao verifikaciona metoda vodila ka utvrdjivanju onih metoda kojima se može precizno opisati značenje programa i ispitati, nezavisno od aktuelnog izvršavanja, da li on odgovara svojoj specifikaciji.

Navedeni problem ima svoje dalekosežne i duboke posledice, a put kojim je on rešavan vodi ka konstituisanju programiranja u strogo zasnovanu naučnu oblast. Najveća zasluga za to pripada, bez sumnje, briljantnom holandskom programeru Edsgeru Viibu Dejkstru (Edsger Wybe Dijkstra). Pored brojnih fundamentalnih programskih rešenja (podsetimo se samo na bafere ili sinhronizaciju paralelnih procesa), najveći Dejkstrin doprinos se sastoji u stalnom osvetljavanju onih misaonih aktivnosti koje se nalaze u korenu procesa programiranja. Dubinu promena koje su se odigrale u samom poimanju pojma programa, Dejkstra je izložio u predgovoru nedavno objavljenoj Grisovoj (Gries) knjizi "Nauka programiranja" [5]: "Tokom poslednje decenije moguće značenje reči 'program' se duboko promenilo. I programi kakve smo pisali pre deset godina i programi kakve možemo pisati danas, mogu se izvršavati na računaru ali im je to gotovo jedina zajednička crta. Osim ove površinske sličnosti, oni su toliko suštinski različiti da bi moglo doći do zabune ako ih nazivamo istim imenom. Razlika izmedju 'starih' i 'novih' programa je onoliko duboka koliko i razlika izmedju ... prenaučnog znanja o matematičkim činjenicama i strogo izvedenih zaključaka iz zadatih postulata."

U okviru ovog teksta ćemo nastojati da na primeru evolucije rešenja jednog zadatka ilustrujemo ovu suštinsku evoluciju u shvatanju procesa programiranja.

2. Jednostavan zadatak

Neka je potrebno konstruisati program koji će za zadato x, $x > 0$, izračunavati z tako da je $z = [\sqrt{x}]$. Na primer, za $x = 7$, biće $z = 2$, a za $x = 17, z = 4$.

U većini programskih jezika postoje funkcije "celobrojni deo" i "kvadratni koren", pa se ovaj zadatak može, primenom metode "grube sile", rešiti njihovom upotrebom. Na primer, u FORTRAN-u se ovaj problem može rešiti iskazom dodele

```
IZ = IFIX ( SQRT ( FLOAT (IX) ) )
```

ili u PASCAL-u iskazom:

$$z := trunc(sqrt(float(x))).$$

Ipak, funkcija *sqrt* je, po pravilu, definisana za realne argumente, pa se u gornje "praktično" rešenje jednog celobrojnog problema nepotrebno (i ne bez opasnosti) uvode elementi realne aritmetike. S druge strane, budući da je funkcija *sqrt* predefinisana, to je postupak izračunavanja korena prikriven, pa nad njim ne možemo (a priori) uspostaviti nikakvu misaonu kontrolu.

3. "Stari program"

Kao što je Dejkstra naznačio, "stari prograi" su pisani pre deset i više godina. Na jedno rešenje postavljenog zadatka iz tog doba nailazimo kod Mane [7], u poglavlju o verifikaciji programa. Tu se zadatak sastoji u tome da se za program prikazan dijagramom toka na slici 1. dokaže parcijalna korektnost u odnosu na ulazno tvrdjenje

$$\varphi(x):\ x \geq 0$$

i izlazno tvrdjenje:

$$\Phi(x,z):\ z^2 \leq x < (z+1)^2$$

koje je ekvivalentno sa:

$$z = [\sqrt{x}].$$

Postupak izračunavanja se zasniva na poznatoj činjenici da je zbir prvih n neparnih brojeva jednak kvadratu broja $(n+1)$:

$$(*)\ 1+3+5+\cdots+(2n+1) = (n+1)^2$$

Izračunavanje se tada odvija tako što se za svako $n = 0, 1, 2, \ldots$, vrednost n pamti u promenljivoj y_1, pridruženi neparni broj $2n+1$ u promenljivoj y_3, a suma prvih n neparnih brojeva u y_2. Kada je ispunjen uslov jedine petlje: $y_2 > x$, tada je

$$\sum_{i=0}^{i=y_1}(2i+1) > x,$$

odnosno zbog $(*)$, $(y_1+1)^2 > x$. Kako je u prethodnom koraku bilo $y_1^2 \leq x$, to je tražen z, koje zadovoljava Φ, upravo y_1.

Mana dokazuje dalje da je ovaj program parcijalno korektan u odnosu na zadatu ulazno-izlaznu specifikaciju.

U sasvim pojednostavljenom obliku, dokaz se sastoji u sledećem: presečnoj tački B se dodeljuje invarijanta

$$P:\ y_1^2 \leq x \wedge y_2 = (y_1+1)^2 \wedge y_3 = 2 * y_1 + 1$$

Ova invarijanta ima vrednost *tačno* za bilo koju vrednost koju tokom izračunavanja u petlji, polazeći od datih početnih vrednosti, mogu dobiti promenljive y_1, y_2 i y_3. Prvi član gornje konjunkcije izražava da je tokom izvršavanja petlje, u y_1 prirodan broj čji je kvadrat manji od zadatog x, drugi član izražava vezu ostvarenu formulom $(*)$, dok treći član opisuje da se preko y_3 brojaču y_1 pridružuje odgovarajući neparan broj $(2y_1+1)$.

Da ovo tvrdjenje ostaje uvek tačno u petlji, možemo se "empirijski" uveriti probnim izvršavanjem programa sa slike 1. za neko utvrdjeno x. Sama tehnika dokazivanja korektnosti ovog programa bi nepotrebno opteretila ovo izlaganje i biće izložena na drugom mestu.

4. Interludijum

Rešenje koje je Mana predložio se značajno razlikuje od "prastarih" programa koji se razvijaju isključivo na bazi programerskog iskustva i intuicije i za koje je testiranje jedini način verifikovanja. Opisani program je opremljen dokazom korektnosti, pa je izvesno da će za svako x koje zadovoljava ulazno tvrdjenje φ, rezultat z zadovoljavati izlazno tvrdjenje Φ. Kako se na sličan način pokazuje da se ovaj program završava u odnosu na ulazno tvrdjenje φ, testiranje nije potrebno.

Ipak, navedeno rešenje ima nekoliko očiglednih nedostataka.

U pogledu formalne strukture programa na slici 1, primećujemo da se uslov petlje nalazi izmedju dve dodele, u sredini petlje. U doslovnoj linearizaciji datog dijagrama toka na neki programski jezik se, zbog toga, ne može izbeći korišćenje bezuslovnog skoka iz sredine petlje. Sa stanovišta valjanog strukturiranja programa, pak, trebalo bi ovakvu petlju preformulisati u petlju tipa **repeat** ili **while**. U jednom kasnijem povratku na isti primer [9], Mana je bio prinudjen da preformuliše ovo rešenje na sledeći način:

$$\begin{array}{ll} & input(x) \\ L_0: & (y_1,y_2,y_3) \leftarrow (0,1,1) \\ L_1: & \textbf{if } y_2 > x \textbf{ then } L_2: output(y_1) \\ & \qquad \textbf{else } (y_1,y_3) \leftarrow (y_1+1, y_3+2) \\ & \qquad\quad y_2 \leftarrow y_2 + y_3 \\ & \qquad\quad \textbf{goto } L_1. \end{array}$$

gde su L_0, L_1 i L_2 obeležja koja odgovaraju presečnim tačkama programa sa slike 1.

Ozbiljniji nedostatak izloženog rešenja se ogleda u tome što je ono konstruisano sasvim nezavisno od svoje formalne specifikacije. Kao što je već rečeno, ulazno-izlazna specifikacija izražava "ono što bismo želeli da program radi", dok sam program "radi ono što je u njemu zapisano da treba da radi". U tom smislu, dokazom korektnosti se ispituje da li je zapis programa saglasan sa zadatom specifikacijom iz koje se, pak, ne može sagledati nikakva organska veza predloženog rešenja i same specifikacije. S druge strane, sam dokaz korektnosti implicitno ukazuje upravo na iznesenu činjenicu: njime se pokazuje da se datim programom ulazno tvrdjenje

može transformisati u izlazno tvrdjenje. Odavde dalje slede važne primedbe.

S jedne strane, izabrani postupak izračunavanja $[\sqrt{x}]$, zasnovan na (*), nije posledica zadate specifikacije. Umesto (*), mogao se koristiti ma koji drugi "trik" kojim se promenljive x i z dovode u vezu izrečenu pomoću Φ (na primer, jedno drukčije rešenje se može dobiti koristeći geometrijsku interpretaciju (*), prema [11]). Moguće je, dakle, imati bezbroj rešenja, zavisno od dovitljivosti, koja su sva podjednako dobra jer zadovoljavaju zadatu specifikaciju. Ali isto tako, za svako od takvih rešenja se može postaviti pitanje: "Zašto baš takvo rešenje?" ili "Kako smo se tog rešenja dosetili?", koje nužno ostaje bez odgovora. Radi se, dakle, o rešenjima koja se temelje na dosetljivosti ili intuitivnom poznavanju problema, a nikako o sistematičnom postupku koji je nametnut razumevanjem specifikacije problema.

S druge strane, ovaj isti problem se ogleda i u konstrukciji invarijante P u tački B. Mana ([7], p.177) to jednostavno komentariše: "Sada možemo verifikovati dati program. S tim ciljem pridružimo induktivno tvrdjenje P ... tački preseka B ..." Odavde vidimo da se i sama invarijanta P, ponaša kao eksterna specifikacija petlje. U nastojanju da ovaj nedostatk prevazidje i da invarijante generiše iz samog programa, Mana [8] je kasnije predložio jedan postupak koji to omogućava. U tom postupku se prvo heuristički generiše invarijanta a zatim se ona "gura" ka početku i kraju programa, čime se dobijaju ulazno-izlazne specifikacije na osnovu samog teksta programa. Ipak, ni ovaj postupak ne osvetljava organsku vezu koja mora postojati izmedju korisničke specifikacije programa i samog programa. U tom svetlu, invarijanta P predstavljau osnovi samo drukčiji iskaz već opisanog postupka izračunavanja $[\sqrt{x}]$. Iz izloženih komentara proizilazi da je, umesto naknadnog anotiranja već napisanog programa u cilju njegove verifikacije, prirodniji onaj postupak koji bi omogućio da se iz same formalne specifikacije izvede tekst programa.

5. Ambijent za "nove programe"

Da bi se otklonili nedostaci "starih" programa, potrebno je, pre svega, imati na raspolaganju jezik koji će to omogućavati. Jedan takav "mali" jezik je predložio Dejkstra [3], uvodeći pojam *zaštićene komande* (engl. guarded command). Sintaksa ovog jezika je u osnovi ALGOL-lika, a osnovnu razliku u odnosu na druge jezike predstavlja to što svakoj listi uobičajenih programskih iskaza prethodi *zaštita* (engl. guard): bulovski izraz kojim se odredjuje pod kojim uslovom će se ta lista iskaza izvršiti. Skraćeni formalni zapis u BNF-u, dat u [3], je:

⟨zaštićena komanda⟩ ::=
  ⟨zaštita⟩ → ⟨zaštićena lista⟩
⟨zaštita⟩ ::= ⟨bulovski izraz⟩
⟨zaštićena lista⟩ ::= ⟨iskaz⟩ { ; ⟨iskaz⟩ }
⟨zaštićeni skup komandi⟩ ::=
  ⟨zaštićena komanda⟩ { [] ⟨zaštićena komanda⟩ }
⟨alternativni konstrukt⟩ ::=
  if ⟨zaštićeni skup komandi⟩ **fi**
⟨repetitivni konstrukt⟩ ::=
  do ⟨zaštićeni skup komandi⟩ **od**
⟨iskaz⟩ ::= ⟨alternativni konstrukt⟩ |
  ⟨repetitivni konstrukt⟩ |
  ⟨"ostali iskazi"⟩

Pri tome, pod "ostalim iskazima" se podrazumevaju razni iskazi dodele, pozivi procedura, i sl. Simboli {...} se čitaju kao "praćeno jednim ili više pojavljivanja onoga što je medju zagrade uključeno" a | kao "ili". Simbol [] je separator zaštićenih komandi.

Na primer, zaštićenom komandom

$$\textbf{if } y > x \rightarrow z := y \,[]\, y \leq x \rightarrow z := x \textbf{ fi}$$

se izračunava $z = max(x, y)$. Ovaj jezik zahteva da se, pri pisanju programa, za svaki linearni niz iskaza precizno odrede uslovi pod kojima će se taj niz iskaza izvršiti.

Obeležimo, dalje, sa S ma koji iskaz u ovom programskom jeziku, a sa Q i sa R, redom pridruženi preduslov i pauslov iskaza S. Oznaka, koja potiče od Hora [6]:

$$\{Q\}\ S\ \{R\}$$

prema Dejkstri označava da za ma koju vrednost za koju je predikat Q tačan, izvršavanje iskaza S se završava i za nove vrednosti programskih promenljivih, ostvarene izvršavanjem S, je tačan pauslov R. Medju mogućim preduslovima Q iskaza S u odnosu na R jedan odredjuje skup *svih* stanja za koja će, po završenom izvršavanju iskaza S, R biti tačno. Takav preduslov se naziva *najslabiji preduslov* (engl. weakest precondition) iskaza S i funkcija je iskaza S i pauslova R. Obeležavamo ga, stoga, sa $wp(S, R)$. Na primer, za iskaz

$$S\colon\ i := i + 1$$

i pauslov R: $i \leq 1$, najširi preduslov je

$$wp(\text{“}i := i + 1\text{”}, i \leq 1) = (i \leq 0),$$

odnosno, da bi po izvršavanju S važilo R, mora *pre* izvršavanja S biti $i \leq 0$. U suprotnom, tj. ako je $i > 0$, R ne može biti tačno kada se izvrši S. Slikovito govoreći, transformator *wp* "potiskuje unazad" predikat R kroz iskaz S.

Koristeći se transformatorom *wp* moguće je precizno definisati dejstvo svakog pojedinačnog iskaza. Ograničićemo se na opis iskaza koje ćemo koristiti u daljem izlaganju.

Iskaz *skip* (koji je sličan npr. sa CONTINUE u FORTRAN-u) se definiše sa

$$wp(\text{“}skip\text{”}, R) = R$$

odnosno, izvršavanjem iskaza *skip* se ne vrši nikakva transformacija nad programskim promenljivim. Iskaz dodele $x := e$, gde je e izraz (u uobičajenom smislu) se može definisati sa

$$wp(\text{“}x := e\text{”}, R) = domen(e) \wedge R_e^x$$

gde je $domen(e)$ predikat koji opisuje skup stanja u kojima se izraz e može izračunati, a R_e^x je rezultat zamene svakog pojavljivanja x u pauslovu R izrazom e.

Na sličan, ali daleko složeniji način se opisuje i značenje alternativnog i repetitivnog konstrukta.

6. Anatomija petlje

U tački 5. je u najkraćim crtama izložen jezički aparat koji je Dejkstri potreban za izgradnju "novih" programa. Preostaje nam da ukažemo na osnovni metodološki okvir za razvoj novih programa. Do sada izloženi aparat predstavlja osnovu za aksiomatske dokaze korektnosti već napisanog programa. Ono što predstavlja osnovnu razliku izmedju "starih" i "novih" programa je činjenica da se "novi" programi razvijaju neposredno iz svoje formalne specifikacije: oni se javljaju kao zaključak u jednom deduktivnom procesu čije su premise zadate specifikacijom. Ovakvi programi se, stoga, jasno razlikuju od onih koji nastaju na bazi programerske intuicije jer su, već prema samom načinu konstruisanja, korektni a postupak njihove konstrukcije sadrži eksplicirano objašnjenje korišćenog načina izračunavanja. Takav postupak se, prema [5], zasniva na skupu pažljivo odabranih principa, kojima se uokviruje način razmišljanja o problemu. Sama konstrukcija programa je rezultat primene odredjene strategije u rešavanju zadataka.

Kako zadatak postavljen u tački 2. ilustruje upotrebu programske petlje, razmotrimo detaljnije samo *strategiju razvoja petlje*:

Razviti prvo uslov petlje B tako da važi $P \wedge \neg B \Rightarrow R$, gde je P invarijanta petlje, a R pauslov petlje. Razviti potom telo petlje na takav način da se obezbedi u konačnom vremenu ispunjenost uslova petlje B uz očuvanje važenja invarijante P [5].

Petlja koja se realizuje ovom strategijom ima oblik:

$$\{P\} \textbf{ do } B \rightarrow telo \textbf{ od } \{R\}$$

Kada se petlja završi, mora važiti $\neg B$, pa je zato $P \wedge \neg B \Rightarrow R$. Telo petlje mora obezbediti takvo izračunavanje u kome je, s jedne strane, invarijanta uvek tačna a sa druge, kojim se polazeći od nekih početnih uslova za konačno vreme ispunjava uslov izlaska iz petlje. Pri tome, s obzirom na rečeno o najslabijem preduslovu, mora biti ispunjeno i

$$P \wedge B \Rightarrow wp(\text{“}\textbf{do } B \rightarrow telo \textbf{ od}\text{”}, R)$$

Otuda sledi da se izračunavanjem *wp* za dato R može dobiti predikat koji sadrži i invarijantu P i uslov petlje B. Tehniku identifikovanja predikata P i B navešćemo kasnije, uz odgovarajuće primere.

7. "Novi programi"

Pogledajmo sada evoluciju rešenja zadatka postavljenog u tački 2. u svetlu izloženih koncepata [5]. Navedena rešenja potiču od Dejkstre, prema [4].

Sam zadatak iz tačke 2. se može preformulisati na sledeći način:

Sastaviti program kojim se za zadati ceo broj $x \geq 0$ uspostavlja istinitost predikata:

$$R\colon\ 0 \leq z^2 \leq x < (z+1)^2 \qquad (1)$$

ili, što je ekvivalentno, istinitost sledeće konjunkcije:

$$R\colon\ 0 \leq z^2 \wedge z^2 \leq x \wedge x < (z+1)^2. \qquad (2)$$

Iz ove izlazne specifikacije programa se njenim "slabljenjem" može dobiti moguća invarijanta petlje programa koji treba napisati. Jedna tehnika slabljenja se sastoji u *brisanju člana konjunkcije*. Kandidati koje treba ukloniti iz uslova (2) da bi se ovaj uslov "oslabio" su ili drugi ili treći član konjunkcije (jer oni vezuju ulazne i izlazne promenljive). Ako iz (2) izostavimo, na primer, treći član, dobijamo kao moguću invarijantu sledeći predikat:

$$P\colon\ 0 \leq z^2 \wedge z^2 \leq x. \qquad (3)$$

Kako je $x \geq 0$, inicijalno se istinitost invarijante P postiže dodelom $z := 0$. Kao uslov petlje (s obzirom na izloženo u tački 6), može se uzeti negacija odbačenog člana konjunkcije, tj. $(z+1)^2 \leq x$.

U tom slučaju, kada se izvršavanje petlje završi zato što uslov nije više ispunjen, biće izbrisani član konjunkcije, *a fortiori*, tačan. Otuda dobijamo prvu aproksimaciju traženog programa:

$$z := 0;\ \textbf{do}\ (z+1)^2 \leq x \rightarrow ?\ \textbf{od}$$

Iskaz koji treba da zameni ? u petlji, treba da obezbedi uvećavanje promenljive z kako bi gornji uslov bio u jednom trenutku zadovoljen. S obzirom na (3), z je odozgo ograničeno sa $\sqrt{x}$, a najjednostavniji način da se z povećava se ostvaruje dodelom $z := z + 1$, pa otuda imamo program:

$$z := 0;\ \textbf{do}\ (z+1)^2 \leq x \rightarrow z := z+1\ \textbf{od} \qquad (4)$$

Lako se sada može pokazati da je

$$P \wedge B = wp(\text{“}z := z+1\text{''}, P)$$

tj. da je P invarijanta petlje.

Ovo rešenje je na žalost “sporo”, jer je vreme izvršavanja proporcionalno sa $\sqrt{x}$. Sporost je, pri tome, neposredna posledica izabranog načina napredovanja kroz petlju iskazom $z := z + 1$. Brzina se može povećati ako se umesto linearnog napredovanja obezbedi “bisekciono” napredovanje, inspirisano algoritmom za binarno pretraživanje. U tom smislu, umesto zahteva izraženog predikatom (1), zatražimo da se zadati broj x nalazi u nekom intervalu (z^2, y^2), gde je y nova promenljiva. Pokušajmo da “pronadjemo” $[\sqrt{x}]$ tako što ćemo gornji interval prepoloviti a rad nastaviti na onoj polovini koja sadrži x. Ovim je implicitno izložen još jedan postupak “slabljenja” predikata R, postupak *zamene konstante promenljivom*: umesto izraza $z + 1$, uvodimo novu promenljivu y tako da bude:

$$0 \leq z^2 \leq x < y^2. \qquad (5)$$

Da bi (5) bilo tačno, mora biti $y > z$. Inicijalno se istinitost predikata (5) utvrdjuje dodelom:

$$z, y := 0, x+1$$

Nova promenljiva y je tada ograničena sa $z+1$ (zbog $y > z$) i sa $x+1$ (zbog gornje dodele). Imamo otuda sledeću invarijantu buduće petlje:

$$P\colon\ z < y \leq x+1 \wedge z^2 \leq x < y^2. \qquad (6)$$

Koristeći se strategijom iz prethodne tačke, može se utvrditi da iz

$$P \wedge \neg B \Rightarrow R$$

sledi da B treba da bude oblika $z + 1 \neq y$. Imamo otuda:

$$z, y := 0, x+1$$
$$\textbf{do}\ z+1 \neq y \rightarrow ?\ \textbf{od}$$

Preostaje još da se odredi telo petlje. Petlja se završava kada je $z + 1 = y$, a iz (6) je $z + 1 \leq y$. Sledi da je potrebno da se izračunavanjem u telu petlje iterativno smanjuje razlika $y - z$. “Spori” način za ostvarenje ovog cilja se postiže dodelom $z := z+1$, a brži se zasniva, kao što je već rečeno, na polovljenju intervala (y, z) što se postiže sledećom alternativnom komandom:

$$\textbf{if}\ ? \rightarrow z := (z+y) \div 2\ [\!]\ ? \rightarrow y := (z+y) \div 2\ \textbf{fi}$$

Izvršavanje ovog iskaza treba da obezbedi da invarijanta (6) ostane tačna posle svake iteracije. Najslabiji preduslov za prvu dodelu je

$$\begin{aligned} wp\quad & (\text{“}z := (z+y) \div 2\text{''}, P) = \\ & (z+y) \div 2 < y \leq x+1 \wedge \\ & ((z+y) \div 2)^2 \leq x \wedge x < y^2. \end{aligned}$$

Kako invarijanta mora zadovoljavati

$$P \wedge B \Rightarrow wp(\text{“}z := (z+y) \div 2\text{''}, P)$$

lako se nalazi da je potrebna zaštita **if**-komande

$$((z+y) \div 2)^2 \leq x$$

Slično za drugu dodelu nalazimo uslov $((z + y) \div 2)^2 > x$. Uvodjenjem pomoćne promenljive w koja čuva medjurezultat izračunavanja $(z + y) \div 2$, dobija se konačan program

$$\begin{aligned} & z, y := 0, x+1 \\ & \textbf{do}\ z+1 \neq y \rightarrow w = (z+y) \div 2 \\ & \qquad\qquad \textbf{if}\ w * w \leq x \rightarrow z := w \\ & \qquad\qquad [\!]\ w * w > x \rightarrow y := w \\ & \qquad\qquad \textbf{fi} \\ & \textbf{od} \end{aligned}$$

Dobijeni program ima vreme izvršavanja proporcionalno sa $ln\, x$.

Sledeći korak u preformulisanju ostvarenog rešenja je podstaknut namerom da se “skupi” operator celobrojnog deljenja $\div$ ($x \div y$ je najveći ceo broj manji ili jednak x/y) zameni običnim deljenjem. To nas vodi u *promenu reprezentacije* podataka iz prethodnog rešenja.

Zamena $\div$ sa $/$ bi bila trivijalna kada bi $z + y$ bilo uvek parno (jer je onda $(z + y) \div 2 = (z + y)/2$). Kako to nije slučaj, treba uvesti “smenu”— novu promenljivu t koja će obezbediti da razlika $z-y$ ostane tokom celog izračunavanja parna. Ovaj uslov se najlakše postiže sledećom smenom:

$$\exists p \geq 1\colon\ t = 2^p$$

Novu promenljivu y iz prethodnog rešenja, izaberimo tako da je

$$y = z + t$$

Sada je, s obzirom da je $y - z = t$ parno, biće

$$w = (z + y) \div 2 = z + (y - z)/2 = z + t/2.$$

Zaštita petlje $z + 1 \neq y$ se neposredno transformiše u uslov $t \neq 1$, a invarijanta (6) u predikat:

$$P\colon\ 0 \leq z^2 \leq x < (z+t)^2 \wedge \exists p \geq 1\colon\ t = 2^p \quad (7)$$

Preostaje još da se promenljive y i w izraze preko promenljivih z i t. Ako je $w^2 > x$, to će biti $y := w$ zamenjeno sa $y := z + t/2$, jer je

$$w = (z + y) \div 2 = (z + z + t) \div 2 = z + t/2$$

i $y := z + t$, pa je odgovarajuća smena $t := t/2$.

Za $w^2 \leq x$, slično dobijamo:

$$z := z + t/2$$

Iz $w := (z+y) \div 2 = z + t/2$ i $w = z + t/2$, vidimo da se w može eliminisati. Prva aproksimacija petlje daje:

do $t \neq 1$ **if** $(z + t/2)^2 \leq x \rightarrow z := z + t/2$
$\quad [\!]\ (z + t/2)^2 > x \rightarrow t := t/2$
fi
od

Sada se može lako dodela $t := t/2$ izvući ispred alternativne komande, što daje konačno program:

$z, t := 0, 1$; **do** $t^2 \leq x \rightarrow t := t * 2$ **od**
do $t \neq 1$ $t := t/2$
$\quad$ **if** $(z + t)^2 \leq x \rightarrow z := z + t$
$\quad [\!]\ (z + t)^2 > x \rightarrow$ *skip*
$\quad$ **fi**
od

Inicijalna dodela je, pri tome, rezultat potrebe da invarijanta (7) bude zadovoljena pri prvom ulasku u petlju. U samom procesu izračunavanja može se prepoznati još jedan elementarni algoritam: algoritam za celobrojno hardversko deljenje.

8. Budućnost "novih" programa

Kada je krajem 60-tih godina, Dejkstra "osudio" GOTO-iskaz na progonstvo, mnogi medju programerima odraslim na klasičnim programskim jezicima bili su zapanjeni radikalnošću ovog predloga. Prisetimo se samo upornosti kojom je Donald Knut pokušavao da rehabilituje ovaj iskaz [10]. Trebalo je da prodje više od decenije da bi se naširoko shvatilo da se iza progonstva GOTO-iskaza krije, zapravo, zahtev za strogom i sadržajnom vezom izmedju načina razmišljanja o jednom problemu i načina zapisivanja programa. Čini mi se da slična sudbina čeka i "nove" programe.

Misaoni napor koji se zahteva u procesu stvaranja "novog" programa može izgledati preteran u odnosu na jednostavnost nadahnuća kojim se intuitivnim putem stvaraju programi. Ipak, izloženi primeri ubedljivo ilustruju prednosti "novih" programa a kompenzacija za uloženi trud je višestruka: realizovani program je korektan (dakle, ne zahteva testiranje), elegantan (konstruisan bez poziva na "trikove") i, što je možda najvažnije, uz njegovu izgradnju je izvršen "prinudni" razvoj programske dokumentacije (koji je bolna tačka svih klasičnih tehnika razvoja programa). Takodje, ostvarena rešenja su izuzetnog kvaliteta i ne mogu se dobiti drugim putem.

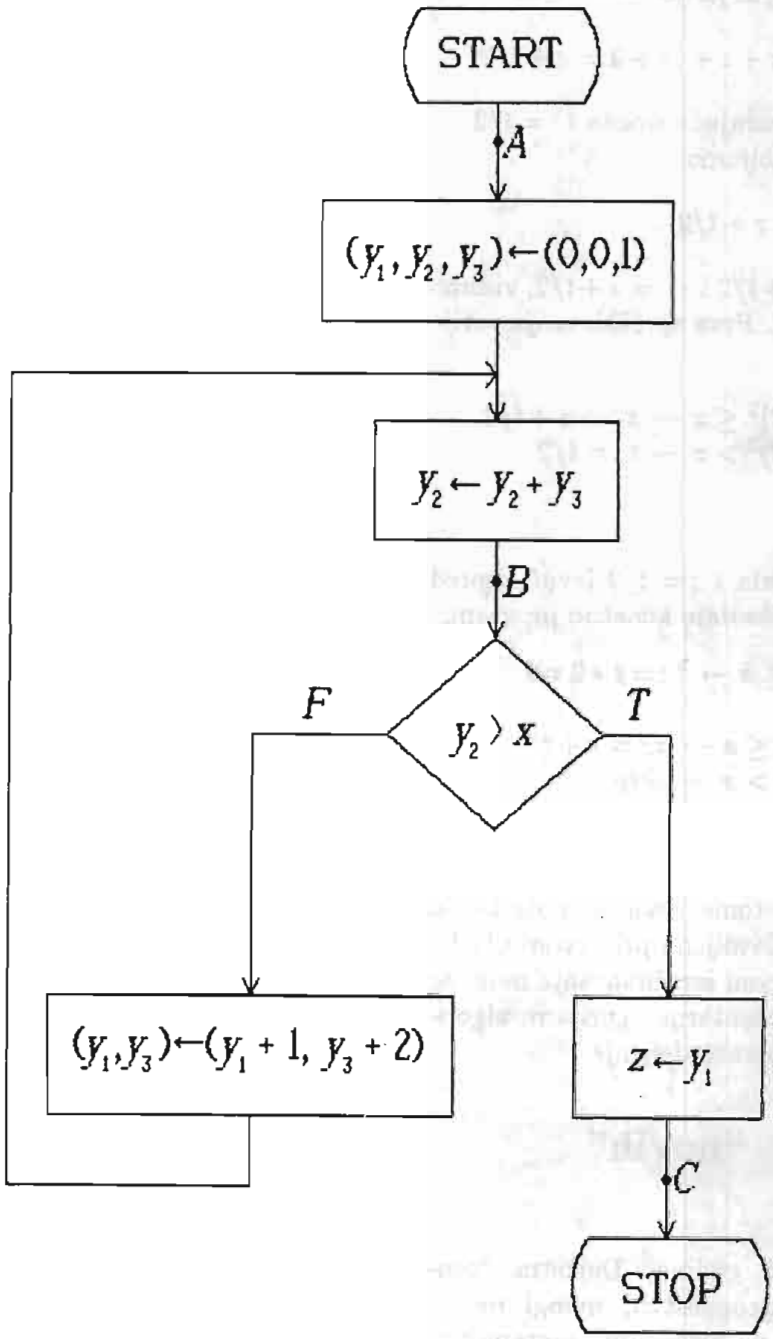
Biće potrebno, bez sumnje, još mnogo vremena da ideje o "novim" programima postanu programerska svakodnevica. Neki preduslovi, sistematsko izučavanje elementarnih algoritama i integrisanje potrebne podrške matematičke logike u programersko obrazovanje, moraju prvo biti ispunjeni. S druge strane, "novi" programi nose u sebi neophodne preduslove kojima programiranje dobija svoju stabilnu metodološku osnovu, a to je prvi korak ka razdvajanju jedne nauke od puke madjioničarske veštine.

Literatura

1. Dijkstra, E.W.: Go to statement considered harmful, Comm. of the ACM 11(3), March 1968, pp. 147–148
2. Dijkstra, E.W.: Notes on Structured Programming, In: Dahl, O.-J., Hoare, C.A.R. and Dijkstra, E.W.: Structured Programming, Academic Press, New York, 1972.
3. Dijkstra, E.W.: Guarded commands, nondeterminacy and the formal derivation of programs, Comm of the ACM 18(8), August 1975, pp. 453–457
4. Dijkstra, E.W.: A Discipline of Programming, Prentice Hall, Englewood Cliffs, 1976.
5. Gries, David: The Science of Programming, (Texts and monographs in Computer Science), IV ed. Springer-Verlag, New York, 1987.
6. Hoare, C.A.R.: An Axiomatic Basis for Computer Programming, Comm. of the ACM, 12(10), October 1969, pp. 576–580, 583

7. Manna, Z.: Mathematical Theory of Computation, McGraw-Hill, New York, 1974
8. Manna, Z., Katz, S.: Logical Analysis of Programs, Comm. of the ACM, 19(3), March 1976, pp.
9. Manna, Z.: Six Lectures on the Logic of Computer Programming, Stanford AI Laboratory, Memo AIM-318, November 1978
10. Knuth, D.E.: Stuctured Programming with GOTO statements, Computing Surveys, 6(4), December 1974. pp.261–301
11. Knuth, D.E.: The Art of Computer Programming, vol. 1, Addison-Wesley, Reading Mass., 1968

Adresa autora: Duško Vitas
Računarska laboratorija
Prirodno-matematički fakultet
Beograd
Studentski trg 16



UDK 681.32:3

Laički pogled na kompjutersko prosvetiteljstvo

Dušanka Ban

Kratak sadržaj: *Uvodjenje kompjutera u bibliotekarske i dokumentalističke poslove zahteva, kao i u svim drugim oblastima primene, dobro razumevanje specifičnosti struke. Kompjuteraši profesionalci ukazuju redovno samo na tehničku stranu uvodjenja kompjutera, a ne spominju organizacione i metodološke promene koje novi alat nužno izaziva u samoj struci i u poimanju njegovih korisnika. U ovom izlaganju, autorica opisuje svoje vidjenje i svoj put u postizanju kompjuterske pismenosti i svakodnevne rutinske upotrebe personalnog kompjutera. Da bibliotekare ohrabri da počnu, ali i da pokaže da se tim putem ne može ići bez podrške kompjuteraša, autorica daje primere iz tri tipična područja primene: a) pisanja kompjuterom (obrada teksta); b) planiranja i izrade specijalizovanog programskog proizvoda; c) kompleksne upotrebe personalnog kompjutera u istraživanju. Namerno nije pomenut nikakav primer korišćenja velikih kompjuterskih baza podataka posredstvom mikrokompjutera, jer su takvi ciljevi obično daleki i teško dosežni, a svakodnevna praksa nosi dovoljno problema koji se mogu realno rešiti mikrokompjuterom.*

Ključne reči: *kompjuter, laički pogled, bibliotekarstvo, dokument.*

1. Moj mikro

Ja nisam kompjuteraš, dakle laik sam za područje programiranja i tehnologiju računara. Korisnik sam ovog novog alata za svoje dokumentalističke, bibliotekarske, istraživačke i publicističke potrebe.

Sticajem srećnih okolnosti početkom 1984. godine personalni mikrokompjuter naselio se u moju radnu sobu. To je u nas ponešto egzotični japanac OKI IF 800 kapaciteta .... i ostale tehničke detalje ni danas ne znam napamet (videti Prilog).

Tada sam stekla i svog kompjuterskog gurua - mislim da je moja totalna bespomoćnost pred svakom napravom složenijom od kafemila naprosto izazvala sažaljenje. Na stranu šala, i svaka pedagogija potvrdiće da je pripremanje uz strpljivog učitelja uvek efikasnije od suvoparnih knjiških lekcija.

Profesionalno znanje u kompjuteraštvu najbrže zastareva i veoma je specijalizovano. Nemojmo se zavaravati da ga je moguće savladati tek za usputnu upotrebu kompjutera kao alata. Reklamni slogani tipa "Lako ćete vladati našim mikrobom XY" znači samo da su vam resursi neograničeno lično dostupni, i ni u kom slučaju ne odnosi se na stvarno znanje programiranja i održavanja kako hardvera tako i softvera kojim se služite. Zato je u svakom početnom učenju, ali i tokom rutinskog rada, nužna podrška eksperta i ja se od nje ni danas ne odvajam.

Volela bih zato da vam ukratko ispričam kondenzat trogodišnjeg procesa učenja, za koje vreme sam popunila 8 disketa i proizvela preko 1.500 strana raznih finalnih dokumenata (ne računajući naravno medjuverzije i kopije).

Možda će vam poneki detalji izgledati trivijalni, a neki sitni, ali se oni obično ne nalaze u oficijelnim uputstvima (manual), koje pretežno pripremaju kompjuteraši za kompjuteraše (njima je npr. autorsko pisanje nepoznat proces, bibliotekarski problemi su im "španska sela" itd).

Iako vam ovaj tekst neće uskratiti muke sopstvenog probijanja kroz višeslojni proces kompjuterskog opismenjavanja, možda izazove radoznalost i ponukati vas da probate.

Svoja zapažanja izložiću kroz primere od kojih svaki predstavlja po jednu tipičnu situaciju:

* WordStar ili upotreba gotovog programskog proizvoda,
* FIKUS ili kreiranje novog programskog proizvoda,
* BC ili kompleksna upotreba kompjutera u istraživanju.

2. Pisanje kompjuterom

Mada je obrada teksta relativno kasno počela da se primenjuje u razvoju kompjutera, postala je najrasprostranjenija upotreba upravo personalnih kompjutera (Kalow, 1984). Naglašavam termin — personalni kompjuter — za kompjutere u ličnoj upotrebi (pa i u posedu).

Igre ne komentarišem, mene ni najmanje ne privlače.

Zato ću prvo govoriti o toj funkciji (iako pisanje tekstova u ukupnom bibliotečkom poslovanju na prvi pogled nije mnogo zastupljeno), jer je smatram najprirodnijom osnovom (početkom) za svaku drugu upotrebu mikrokompjutera.

Za pisanje teksta, kompjuteraši to zovu obrada teksta (4), koristim WordStar (WS) firme MicroPro SAD, jedan od najrabljenijih programa za tu svrhu u svetu.

Pišući na kompjuteru, istovremeno sam savladala tri procesa:

- učenje samog programa - WS tj. komandi kojima se piše,
- nova pravila grafičkog oblikovanja teksta,
- nova pravila stvaranja dokumenta (članka, pregleda i dr.).

Učenje pisanja WordStarom počela sam bez naročitih priprema - morala sam hitno da **prepišem** kratak prilog za časopis, a pisaća mašina bila je pokvarena. U tom tenutku, jedino mi tastatura nije bila strana, još svoj maturski rad sama sam prekucala, a već desetak godina pisala sam direktno pisaćom mašinom.

Šest najbitnijih komandi tada zabeleženih i danas imam na istoj kartici. To su komande za micanje kursora (treptajuća oznaka na ekranu za mesto pisanja — *svetlac ?!?*) po ekranu (levo, desno, gore, dole), komanda za brisanje pojedinog znaka (slova, cifre, interpunkcije) i komanda za dozvolu umetanja znakova. Pripremne radnje da bi se tekst pojavio na ekranu i pohranio, obavio je moj učitelj.

Kod sledećeg prepisivanja zabeležila sam komande za pozivanje programa WS, otvaranje i zatvaranje fajle, brisanje reči, redova, za micanje teksta po ekranu i druge ... a onda su apetiti rasli. Naredna faza je bilo pisanje na osnovu koncepta, zatim na osnovu najgrubljih odrednica planiranog dokumenta, a onda "pravo na ekran".

Vremenom sam postigla izvesnu ekonomičnost u micanju kursora prema vrsti komande koja sledi (što je najvažnije kod najučestanije komande za brisanje jednog znaka ili reči). Na žalost, to se ne da ukratko prepričati.

Pored svojih beležaka sa komandama stigla sam, ipak, samo do skraćenog uputstva WORDSTAR COMMAND CARD od dve nepune strane; do danas nisam dospela da bacim ni pogleda na oficijelni priručnik *(manual)*, a kamoli da ga proučim. Ne kažem da je ovo najbolji metod, ali je prilično efikasan, jer sprečava rasplinjavanje u mnoštvu mogućnosti programa WS, koje prevazilaze osnovne potrebe kod pisanja.

Grafičko oblikovanje teksta pri velikoj raspoloživosti raznolikih fontova, slova različite veličine, lakog izvodjenja podebljavanja, spacioniranja, promena leve i desne margine, postaje veliki problem znanja, mere i ukusa. Upotrebe raznih znakova: ("# $ % & ' ( ) = - + * < > / _") je napast kojoj je teško odoleti.

Kako nema posrednika u finalnoj grafičkoj obradi teksta, odnosno dokumenta, autor se iznenada nadje u ulozi stvaraoca grafizma (grafički urednik) svog dokumenta, najčešće bez ikakvog prethodnog iskustva (jer je malo autora do sada imalo prilike da o tome brine – to je bio "niževredan" posao daktilografkinja).

Mene je ovaj šok mimoišao jer sam, prethodno radeći u izdavačkoj delatnosti, prošla "oštru" školu redaktora i urednika publikacija.

Zavodljivost brojnih grafičkih mogućnosti u oblikovanju teksta mnoge je dokumente učinila besmisleno iskićenim poput božićnog drvca. Uočavajući ovu pojavu, pojedine redakcije časopisa i izdavači u svetu, posebno oni koji primenjuju *electronic publishing*, izdaju detaljna uputstva za autore o oblikovanju teksta [1], [6].

Kao ilustracija neka posluži priručnik *Publication Manual of American Psychological Association* [6] na 208 strana formata A-4, sa 35 referenci i 42 dela preporučene literature, uz naglasak da je prvo izdanje bilo 1983, drugo revidirano 1984, a ovo treće 1985. godine.

O izboru slova, proreda, prelomu strane, o visini, mestu i sadržaju zaglavlja (*heading*) i napomene na dnu stranice (*footing*), smeštaju paginacije, broju redova na strani, broju slovnih mesta u redu, veličini pasusa i ostalim "cakama" – drugi put. Ili pročitajte u navedenom *Publication Manual* ....

Ustanovila sam takodje i pravilo o redosledu operacija grafičkog doterivanja. Prvo korigovanje slovnih grešaka od početka do kraja fajle, zatim poravnavanje desne margine u celoj fajli, onda

stvaranje razmaka izmedju pasusa i podnaslova, tek na kraju prelom strane. Tako praktično morate proći celu fajlu četiri puta — mislite o tome kada dimenzionirate deo teksta za jednu fajlu.

Jednu stvar je, medjutim, gotovo nemoguće udobno izvesti prilikom pisanja na "običnim" (čitaj jeftinim) tipovima mikrokompjutera — umetanje fusnota (podnožnih napomena), tako omiljenih u radovima teoretičara. Ja ih obično izostavljam u svojim tekstovima, ali kada su baš nužne stavljam ih u poseban pasus unutar strane označen ovako /** **/. Profesionalni kompjuteri i programi rešavaju elegantno ovaj problem.

Stvaranje dokumenta u odnosu na proces ručnog pisanja ili pisanje pisaćom mašinom, razlikuje se u nekoliko bitnih tačaka.

Na ekranu je **vidljiv** samo deo teksta, otprilike oko polovina strane uobičajenog formata A4. To iziskuje bolje pamćenje prethodno napisanog teksta, što nije teško za jednu, dve strane. Više od prethodno napisanih pet strana mislim da nije moguće "držati u glavi"; pogotovo u početnoj verziji. Istina, tekst se može pomicati po ekranu gore i dole, tj. može se baciti pogled na prethodno ili potonje napisano, ali kod dužih tekstova čak i izvršenje komande ubrzanog pomicanja zametno traje.

Velika **fleksibilnost premeštanja** i brisanja reči, rečenica i celih pasusa dozvoljava pisanje na preskok i sukcesivno pisanje uporednih varijanti delova teksta. Brisanje uporednih varijanti delova teksta, barem u početnoj obuci, bolje je činiti tek na otisnutom tekstu.

Neekonomično je pisati sa **proredom**, pošto je tada vidljiv još manji deo teksta. Medjutim, na otisnutom tekstu bez proreda zapažanje slovnih grešaka je veoma otežano. Čitljivost teksta umanjuje i poravnata desna margina jer na nju nismo navikli u rukopisu, a ni kod teksta pisanog pisaćom mašinom. S početka sam uredno ravnala desnu marginu čim bih napisala pasus, a sada to radim na samom kraju oblikovanja teksta.

Zavisno od **obima dokumenta** (ako je duži od pet – šest strana), tekst razdvajam u manje fajlove kojima se lakše barata (micanje teksta na ekranu je brže, preoblikovanje, korigovanje i otiskivanje je nezavisno). Fajlove radim po šemi: naslovna strana i sadržaj; tekst 1; tekst 2; prilozi; literatura.

Svaki tekst pisan na kompjuteru mora da ima svoju **identifikaciju** da bi se moglo baratati fajlovima na disketi. WS na CP/M operativnom sistemu dozvoljava 12 znakova, poslednja četiri rezervisana su za tip fajle. Za osam prvih znakova smišljam raznorazne asocijativne skraćenice, pazeći da ostavim neko slobodno mesto za oznake povezanih fajlova. Fajle ovog dokumenta identifikovane su kao "kuh0.doc", "kuh.doc" i "kuha.doc". To su skraćenice od izraza "kuharica" iz pijeteta prema imenu prvog uputstva koje sam dobila za pisanje kompjuterom.

Uopšteno govoreći, svaka **promena teksta** prilikom pisanja predstavlja novu varijantu teksta. Medjutim, ovakav pristup je nepraktičan, jer je moguće izvesti ogroman broj promena, a nije ekonomično printati tekst kod svake promene. Potrebno je fiksirati neko obeležje (granicu) po kojoj se tekst može jednoznačno obeležiti kao celina. Takvu celinu nazivam **verzijom** i označavam je sa ver. 0–n. Ovu oznaku stavljam u zaglavlje otiska i u podsetnik za fajlu (koji je vidljiv samo na ekranu i ne ispisuje se). Kao obeležje za verzije koristim nekad datum pisanja, nekad smisaonu celinu, nekada čin printanja teksta, tj. fajle, ili pak čin korigovanja slovnih grešaka.

Ovo su situaciona obeležja – svako ih može odabrati po svom nahodjenju, jer je teško definisati uvek pogodno pravilo, osim, da se celine, naročito otisnute, moraju nekako obeležiti. U protivnom, stvara se zbrka pošto razne verzije teksta nisu od prve raspoznatljive. Tek posle dve godine iskustva uspela sam da uvedem običaj da otisnuti tekst bacim kada isprintam narednu verziju.

Verzija može imati više podverzija, označenih brojem iza broja verzije i tačke (0.1 – 0.n; 1.1 – 1.n). Podverzije sam u poslednje vreme povezivala sa korigovanjem otisnutog teksta.

U toku učenja pisanja kompjuterom najveću teškoću predstavljalo mi je upravo pamćenje nagomilanog nevidljivog teksta i zbrka sa verzijama i fajlovima. Zato sam razvijala naročitu **taktiku stvaranja dokumenta**.

Prilikom pisanja pisaćom mašinom obično sam imala tri celine dokumenata:

- početnu skiciranu rukom,
- radnu verziju koju sam sama tipkala dva/tri puta, smisaono ispravljala, uredjivala za prepis i korigovala,
- prepis završnog dokumenta (radio profi daktilograf).

Kod pisanja kompjuterom dokumenat prolazi kroz nekoliko faza.

Verzoja 0.0 je početak pisanja bez ikakvog zapisanog koncepta ili samo sa grubom skicom (tema/naslov, delovi dokumenta). Nulta verzija ima dve do tri podverzije u jednoj smisaonoj celini (nastavak pisanja narednih dana, direktne dopune), a onda je otisnem.

Verzija 1.0 nastaje posle čitanja otisnute nulte verzije. U njoj smisaono preoblikujem tekst prepisivanjem izmena sa papira, ali i direktnim unošenjem promena. Deo korekcija slovnih grešaka unosim direktno u verziju 1.2 ili 1.3, kao i formatizaciju desne margine. Verzija 1 ima obično dva do tri otiska.

Verzija 2.0 obuhvata sve promene sa otisnute prethodne verzije. Kroz podverzije direktno korigujem i grafički uredjujem celokupan tekst (podnaslovi, masna slova, proredi, prelom stranica). Otisak verzije 2.3 obično je poslednji ukoliko tekst ide u štampu.

Verziju 3.0 obično radim ako tekst treba da predstavlja samostalni dokument (ako neće biti štampan, već fotokopiran ili printan u više primeraka).

Lakoća pisanja i dodavanja izazvala je produžavanje mojih tekstova. Začetke sklonosti ka perfekcionizmu u grafičkom oblikovanju teksta oštro sam sprečila. Takodje proizvodim i više tekstova uz manji osećaj zamora — pisanje kompjuterom zaista me motiviše — ko to tvrdi da mašina otudjuje?

Jedna od retkih domaćih knjiga (možda trenutno i jedina) sa naslovom Kompjuterska obrada teksta [5], bavi se, medjutim, pretežno dociranjem o hardveru. Oblikovanju teksta posvećeno je jedva 30 strana (od 192). Još je najinteresantnije poglavlje "Psychoware", o psihološkim aspektima kompjutera. Sama knjiga oblikovana je kompjuterom, ali tako zbrljano, da predstavlja baš primer kako dokument ne sme da izgleda (ni sa kompjuterom ni bez njega) — titrav slog gotovo je nemoguće čitati, a ima i drugih "bisera" (npr. popis literature u jedinstvenom pasusu na dve strane). Knjiga ne sadrži nikakav podatak o autoru koji mi ne izgleda dovoljno osposobljen za pisanje priručnika o toj tematici, čak ni u vlastitom izdanju.

3. FIKUS

Svi vi znate kako je tegobna izrada godišnjeg sadržaja časopisa sa predmetnim indeksom. Posle deset godina ručnog rada za časopis "Fizička kultura", Beograd, i preko 1.000 obradjenih bibliografskih jedinica (u furioznom tempu na deset dana pre štampanja poslednjeg godišnjeg broja), svanulo mi je kada je FIKUS proradio.

Uzusi za kreiranje FIKUSA bili su unapred čvrsto odredjeni već ustanovljenim funkcijama i oblikom Godišnjeg sadržaja koji je obuhvatao:

- autorski indeks
  (sa bibliografskim opisom članka uz prvog autora),
- indeks pojmova (predmetni)
  (veže se na redni broj bibliografske jedinice iz Autorskog indeksa).

Guru je ustvrdio da će program za ovu obradu proizvesti za dva dana - tipično za kompjuteraše. Ja sam - tipično za dokumentalistu, smatrala da su zahtevi bezazleni i potpuno definisani (redosled unutar bibliografskog opisa, moguća veličina polja, način sortiranja i klasifikacije itd.).

Ispostavilo se da je niz detalja trebalo revidirati, a neke nove smisliti. Bilo je vrlo komplikovano postići da se u Autorskom indeksu kod drugog autora pojavi uputnica na prvog autora, gde je bibliografski opis. Takodje je problema bilo sa modifikacijom sortnog niza za jugoslovenske latinične znakove i istu sortnu težinu za velika i mala slova (da bi se ključne reči mogle pisati i malim i velikim slovom po potrebi). I sortiranje bibliografskih jedinica istog prvog autora, prema rastućem broju sveske u kojoj je objavljen članak, nije prošlo bez muke.

Pokazalo se da treću funkciju nije ekonomično raditi kompjuterom, izrada programa trajala bi puno dana, a sistem bi bio veliki barem kao pola FIKUSA. U stvari, bilo nam je već dosta FIKUSA. Tako pregled članaka po rubrikama i brojevima sveski i dalje radim ručno.

Izrada programa, koji se na koncu razvio u programski sistem od 12 funkcionalno povezanih programa, trajala je više od mesec dana (naravno uz druge poslove), njegovo dokumentiranje još mesec i po dana, a korekcije su radjene kroz naredna tri meseca. Programska dokumentacija ima 64 kompjuterske strane (tj. 127 "normalnih" strana ili 8 autorskih tabaka).

Za rad i održavanje baze podataka koristi se programski jezik dBase II Ashton-Tate verzija 2.4. Programi za obradu i ispis autorskog indeksa i indeksa pojmova napisani su u jeziku PASCAL/MT+ verzija 5.5, a za formatizirani ispis baze podataka korišćen je 'QUERRY' jezik sistema dBase II. Sva sortiranja izvode se programskim sistemom SUPERSORT MicroPro verzija 1.6.

Ovaj programski sistem može se primeniti za izradu godišnjeg sadržaja bilo kojeg časopisa sa istim funkcijama, delomično i za kumulativni sadržaj više godišta, ali ni za jednu drugu svrhu. Tu je još jedno pravilo: ne isplati se generalizacija specijalizovanog programskog sistema.

Izgleda mi da smo ime sistema pre stvorili po asocijaciji na trnje kroz koje smo prošli u njegovoj

proizvodnji, nego kao akronim tako običnog naslova "Fizička kultura - Sadržaj".

4. Istraživač i mikrokompjuter

BC je bilo bibliometrijsko istraživanje jednog uzorka stručne literature i u njemu citiranih referenci (oko 1200 entiteta), uz primenu statističke analize neparametrijskih varijabli - karakteristika izvora i referenci.

U ovom istraživanju kompjuter sam koristila u svim fazama istraživačkog procesa.

U fazi prikupljanja podataka, analizirane parametre varijabli direktno sam unosila u dve baze podataka na mikrokompjuteru. Programi za unos i formiranje baza uradjeni su u programskom jeziku dBase II Ashton-Tate.

Statističku obradu (program ANTE Pavičić/-Štalec, 1979. na programskom jeziku SS) provela sam on-line, preko mikrokompjutera u kući priključenog telefonom, na računaru UNIVAC 1100 u zagrebačkom Sveučilišnom računskom centru.

Rezultate statističke obrade na "velikom bratu" (u formi kontingencionih tabela) pretočila sam i pohranila u moj računar, iz kojeg su printane direktno u elaborat.

Dokumentalističku obradu pojedinih podataka - npr. razna sortiranja, radila sam direktno iz baza, uz pohranu dobijenih rezultata u mikrokompjuteru. Time je omogućeno da se npr. Indeks citiranih autora direktno otisne u elaborat.

Interpretaciju rezultata i druge delove teksta, od prvog koncepta do finalne verzije, pisala sam na mikrobu (služeći se WordStarom).

Kompletan istraživački elaborat (obim 183 strane) **otisnula sam i umnožila** (u 15 primeraka) takodje u kući tokom dva vikenda.

Kompleksna upotreba računara u istraživanju obezbedjuje operativnu dostupnost istraživačke dokumentacije (koja je u slučaju BC zauzela samo šest disketa) za:

- sekundarne analize prikupljenih podataka,
- lagane reorganizacije teksta u nove članke,
- doštampavanje novih primeraka elaborata.

Da bih istraživanje izvela na opisani način saradjivala sam sa kompjuterskim ekspertima koji su se svojski naradili. Ne samo da su pripremili programe za unos podataka, statističke i dokumentalističke procedure, kreirali baze podataka, uspostavili on-line obradu, te brinuli o hardveru, već su i konstruktivno učestvovali u postavljanju koncepcije celokupnog istraživanja. Hvala im i ovom prilikom.

U ovom procesu bilo je i teškoća i propusta.

Prilikom on-line obrade nužno je enormno strpljenje - telefonska veza se često prekida, operateri na velikom sistemu nisu uvek gostoljubivi, u prenesenim podacima mogu se pojaviti "fantomi". Iako je sama statistička obrada BC na velikom sistemu trajala oko 30 minuta, za provodjenje ove procedure utrošili smo oko 12 časova, a telefonsku liniju zauzeli smo puna četiri časa.

Posle završene on-line obrade, u kontingencionim tabelama pojavio se znak e koji je, iz nikada utvrdjenih razloga, "proždirao" podatke u tabelama prilikom svakog micanja kursora. Sistematska eliminacija ovog "fantoma" angažovala je svu maštu i znanje gurua tokom dvadesetak mučnih sati.

Unošenje podataka je dosadno, ali je zato ukupna faza prikupljanja bibliometrijskih podataka trajala znatno kraće (oko 30% vremena potrebnog za ručni rad).

Najveći propust učinila sam što popis literature (131 citirana bibliografska jedinica i oko 60 tematski bliskih dela) nisam uradila kao bazu podataka - sjajno bi mi poslužila za prenos liste referenci u nove radove.

U ovom primeru mikrokompjuter je poslužio kao prenosnik kondenzovanog znanja (direktno su primenjeni gotovi programski proizvodi za statističku obradu, kreiranje baza podataka, sortiranje i sl.). Upotreba ovog alata dodaje istraživaču funkciju operatera i funkcije vezane za obradu teksta (opisane u primeru WS), ali mu dozvoljava komoditet rada u kući (u papučama — što je san mnogih) i nezavisnost od pomoćnih službi (daktilografa i dr.). Jedina nužnost je podrška kompjuteraša, ali je to ravnopravna, kreativna i podstičuća saradnja.

Još nekoliko napomena

Strah

I sada ga osećam. Mala muka u želucu, koju izaziva "little knowledge", upravo se pojavljuje čim sednem pred ekran. To je ipak zanemarljivo prema drhtanju prstiju i kolena, ubrzanom pulsu, suvim ustima, sa početka mojeg saobraćanja sa mikrobom i guruom.

Nenaviknutost na tvrda pravila (npr. poštovanje blenkova i komandi), stvarni rizik opasne komande "era"— "izbriši" (dva puta sam "pregazila" podugačke datoteke vredne višečasovnog unosa), posledice pogrešnih komandi nesvesno datih usled umora, mogu izazvati pravu paniku. Uprkos tome, postepeno, uz pomoć sportskog iskustva i dve joga vežbe, te ustaljene procedure za početak rada na mikrobu, naučila sam da strah preobratim u podsticajnu tremu iščekivanja. Mašina mi najzad blagonaklono uzvraća moje misli na svetlucavom ekranu — pošto sam osvojila izvesno znanje i smirila plahovitost, postali smo partneri u spisateljskom trudu.

Radni ambijent

Za udobni rad na mikrobu dosta je važno puno, belo i ravnomerno **svetlo** u prostoriji. Od svetlucanja i zračenja ekrana oči se često upale — meni tada "Proculin" kapi olakšavaju tegobe.

Idealno je da su tastatura i ekran na pomičnom stoliću čija se visina može podešavati, da ispred tastature ima petnaestak santimetara prostora za oslanjanje podlaktica prilikom kucanja, da je sa desne strane radni sto pod uglom od devedeset stepeni, da se stolica kotrlja... — da, ja upravo opisujem svoj radni **nameštaj i prostor**. Jedino što mi još nedostaje je pomični zaslon sa leve strane ekrana za papire iz kojih se prepisuje.

O štetnosti padanja **električnog napona** u našoj mreži ne moram posebno razglabati. Čak i uz veliki stabilizator koji mi po cele noći šuška, ekran povremeno zastrašujuće bleska. Tada već vidim zauvek zbrisane sve svoje mudrosti i obliva me hladni znoj. Ipak, to mi se desilo samo jednom zbog udara groma u obližnju kulu "Lotršćak" i nikada više nisam ni pomislila da pišem uz idilično rominjanje kiše.

Osim toga, od tada svako malo "sejvam" (pohranjujem) tekst, tj. gotovo posle svakog ovećeg pasusa. To je i inače korisno, jer npr. usled umora pogrešno pritisnuta tipka, može izazvati brisanje teksta u obradi.

Snabdevanje potrošnim materijalom

Priča o "Paperless society" uz upotrebu kompjutera, meni tek sada izgleda kao bajka. Za desetak prethodnih godina pisanja nisam poarčila toliku količinu **papira** i to najbolje kakvoće, kao uz kompjuter. S tog stanovišta, ova je zabava vrlo skupa. U cenu svojih tekstova sada zaista računam i iznos za papir.

Broj potrebnih **disketa** takodje se brzo povećava. Ni jedna disketa ne sme se potpuno popuniti. Npr. za pisanje u WordStaru na disketi mora biti slobodno najmanje dva puta više prostora od najveće datoteke (jer se prilikom aktiviranja datoteka automatski kopira). Diskete obeležavam hronološkim rednim brojem uz posebnu oznaku arhivskih. Držim ih u bajnderu (plastičnom registratoru) po dve u plastičnom fasciklu u koji je umetnut karton—patent iz domaće manufakture, jer kutije sa disketama imaju opasnu tendenciju prevrtanja.

Ispostavilo se, ipak, da je **traka** na printeru najslabija tačka moje opreme — ili bar najnedostupnija. Domaće trake, ako se ne poderu, dosta brzo se otaru, pa otisak postaje nečitljiv. Spasonosna procedura zove se "reinkanje" trake što uz učešće dve spretne osobe traje najmanje dva sata uz sitnice od kojih bi Kišon napravio bar tuce priča. Ad hoc rešenje je otiskivanje celog teksta masnim slovima, što sam više puta uradila u zoru kritičnog datuma.

Novogovor

Ne spadam u puriste i nisam se opirala kompjuteraškom slengu. U ovom izlaganju, videli ste, koristila sam izraze: datoteka (čak u različitom rodu), sejvati, mikrob, kursor, printanje, i to namerno.

Ipak, književnici, lingvisti, filolozi, informatičari ili već ko je zadužen za mentalno zdravlje naroda, trebalo bi da se dosta potrude na ovom području. Znam da se terminologija ne može propisivati, ali se o njoj može pisati, može se opisivati, tumačiti i koristiti za pisanje, tj. pisati o ovim poslovima na našim jezicima.

Držim pred sobom novi višejezični tehnički rečnik o obradi podataka (7), gde za pojam "file" piše da je to na hrvatskom "datoteka", pojma "save" (pohrani) nema, "directory" je imenik, adresar (u WS je to komanda kojom se na ekranu prikaže sadržaj diskete). U jednom starijem englesko-srpskohrvatskom rečniku iz informatike (2), ovaj termin znači "popis", a "data file directory" znači "spisak datoteka".

"Cursor" je "značka" kod Turka (7), "pozicionoindikacioni simbol" kod Kontića (2), a u leksikonu računarskih pojmova (4) je "treptajuća markica". Usudjujem se da predložim ime "svetlac".

Mnogih termina iz obrade teksta novijeg datuma nema u pomenutim rečnicima (2, 4, 7).

Stručni rečnici su retki, medjusobno nepodudarni, sadrže suviše podrazumevanja koje je strano laicima i nemam utisak da ih kompjuteraši baš respektuju. Stoga mi je bliži stručni sleng, jer ga us-

vajam neposredno uz aktivnost. Dokle god profesionalci govore u slengu, ni laicima ništa drugo ne preostaje.

5. Dakle, da rezimiram

Da bi upotrebili mikrokompjuter morate tačno znati (ili bar saznati u realnom vremenskom okviru) šta hoćete, bilo da pomoću kompjutera simulirate postupak koji već primenjujete, bilo da, iz praktičnog iskustva ili iz svoje vizije, kreirate novi postupak.

Nužno je učiti od profesionalca koji je spreman za dugoročnu podršku, tj. ima strpljenja i ne zgraža se na neuke komentare, besmislena pitanja i glupe zahteve.

Posao mora da ima stvarnu korist. Kompjuter ne služi za kočoperenje i statusni prestiž, već da obavi one radnje kojima se efikasno može olakšati redovna delatnost.

Postoje (UTUVI!) četiri kabalistička pravila:

- Započni rad na mikrobu komandom koja na ekranu prikazuje sadržaj diskete.
- Svako malo tokom pisanja daj komandu "save" (pohrani).
- Uvek uradi arhivsku kopiju ako preradjuješ fajlu (datoteku).
- Vodi urednu dokumentaciju u toku rada, a ne posle, za:

fajlove	(imena fajlova, verzija i smeštaj),
diskete	(sadržaj, pozajmica),
softver	(popis, smeštaj, pozajmica),
priručnike	(popis, smeštaj, pozajmica),
hardver	(oštećenja, servisiranje),
eksperte	(telefoni i adrese) jer se nikad ne zna kad će doći nevolja.

I ovo izlaganje nastalo je pred ekranom. Pišući ga, radosno sam se podsetila na dogodovštine svoje kompjuterske avanture koja još traje, a guru se, čitajući, slatko cerekao.

Literatura

Do poslednje stranice uzdržavala sam se da zavirim u rafove svoje biblioteke. Ipak nisam odolela.

Pomenuta dela:

1. "The Electronic Manuscript Project" - Project plan for the development of publishing industry standards and author guidelines on electronic manuscript preparation. Association of American Publishers, New Technology Committee - Electronic Manuscript Subcommittee, 1984, 6 str.
2. Englesko-srpskohrvatski rečnik stručnih termina iz oblasti informatike sa glosarom. Intertrade, Ljubljana, 1974, 408 str. /preveo na srpskohrvatski M. Kontić/
3. Kalow, S. J.: A comprehensive overview of the initial creation and development of word processing from 1964 to the 1980's. International Course on World Processing, Inter-University Centre of postgraduate studies, Dubrovnik, 14-19.5.1984.
4. Maćešić, N.: Leksikon računarskih pojmova. Vjesnikova press agencija-VPA, Zagreb, 1986, 267 str.
5. Matešić, K.: Kompjuterska obrada teksta. Izdanje autora, Zagreb, 1986, 192 str.
6. Publication Manual of American Psychological Association - Third Edition, APA, Washington, 1985, 208 str.
7. Tehnički riječnik - Obrada podataka i programiranje (englesko/nemačko/francusko/rusko/hrvatski). Tehnička knjiga, Zagreb, 1984, 285 str. (autor hrvatskog dela S. Turk)

Preporučena dela:

8. Naiman, A.: Introduction to WordStar. Sybex Berkeley/Paris/Duesseldorf, 1982,202 str.
9. Wedse, J.: Računarski rečnik - vodič za kompjuterski žargon. Tehnička knjiga i Zavod za izdavanje udžbenika, Beograd, 1985, 160 str./preveo A.Jakupović/

** Uveravam vas da sam ovu knjigu pročitala tek pošto sam napisala svoje izlaganje. Svaka sličnost je dakle situaciona i dokaz je da ljudi mogu imati bliske, iako medjusobno nepoznate, zamisli. **

OKI IF 800 - Karakteristike

Mikrokompjuterska radna stanica ili personalni kompjuter ili mikrob, sastoji se od četiri osnovna sastavna dela: ekrana, tastature, arhivskog medija i štampača.

Mikrokompjuter OKI IF 800 model 20, na kojem radim, ima sledeće karakteristike:

- Ekran: 25 × 80 znakova, dijagonala 12 inča, kolor.
- Tastatura sa svim znakovima definisanim ASCII standardom (American Standard Code for Information Interchange) i sa adaptacijom za latinične znakove (č, ć, š, dj, ž) po JUS-u.
- Arhivski medij sa dve 5 inčne diskete, svaka kapaciteta 350.000 znakova.
- Integrisani igličasti štampač, 80 znakova na sekundu.

Umesto OKI štampača koristim matrični štampač TRS 836 sa brzinom od 150 znakova na sekundu.

Ovaj mikrokompjuter baziran na 8-bitnom procesoru ZILOG Z-80A i ima memorijski kapacitet 64 KB. Radi pod operacionim sistemom CP/M-2.2

OKI IF više ne spada u elitne proizvode, već je pomalo "demode", ali meni i danas izvrsno služi.

Adresa autora: Dr Dušanka Ban
Voditelj Bibliotečno-informacijskog centra
Fakultet za fizičku kulturu
41000 Zagreb,
Horvaćanski zavoj 15

prikazi

U ovoj rubrici želimo da dajemo prikaze svega što se dogadja u nas i u svetu od značaja za oblast računarstva. Biće to prikazi računarske opreme, programa, knjiga i časopisa, projekata, naučno-stručnih skupova, sajmova, kao i doktorskih i magistarskih teza iz oblasti računarstva.

Prikazi RAČUNARSKE OPREME imaju za cilj da upoznaju čitaoce sa savremenom računarskom opremom. Prikaz sadrži funkciju opreme, opis principa rada i karakteristike opreme. Ovde će naglasak biti na karakteristikama opreme koju treba da poznaje korisnik opreme da bi mogao da izvrši pravilan izbor pri nabavci opreme, odnosno, da bi mogao da na najbolji način upotrebi opremu pri njenom korišćenju.

Prikazi PROGRAMA imaju za cilj da informišu čitaoce o postojećem, aplikativnom i sistemskom softveru. Kroz ove prikaze čitaoci će dobiti informaciju o funkciji softvera, teorijskim osnovama na kojima počiva softver i zahtevanom hardveru i softverskom okruženju neophodnom za eksploataciju istog. Prvenstvo u ovim prikazima imaće softver za potrebe nauke i obrazovanja razvijen u nas.

Prikaz KNJIGA I ČASOPISA, NAUČNO – STRUČNIH SKUPOVA I SAJMOVA imaju uobičajen smisao da informišu čitoace o novim knjigama i časopisima, kao i o interesantnim radovima u časopisima iz premene računara u nauci i obrazovanju. Želimo na vreme da informišemo čitaoce o naučno-stručnim skupovima i njihovim programima rada, kao i o novostima u opremi i softveru koji se sreću na sajmovima.

Posebno želimo da istaknemo prikaze naučno-istraživačkih PROJEKATA iz oblasti računarstva. Mislimo da će ovo biti interesantna informacija kako za naučne radnike koji rade na projektima tako i za sve druge čitaoce koje interesuje razvoj ove oblasti u nas. Na ovaj način se može ostvariti i potrebna komunikacija izmedju istraživačkih timova koja je u nas nedovoljno prisutna.

Prikazivanjem DOKTORSKIH I MAGISTARSKIH TEZA želimo da doprinesemo boljoj informisanosti šireg kruga stručnjaka medju profesorima, naučnim radnicima, kao i studentima postdiplomskih studija, koje teme obradjuju i kakve originalne rezultate postižu mladi ljudi u svojim radovima za sticanje najviših akademskih zvanja iz oblasti računarstva. Biće to korisna informacija i za uključavanje mladih ljudi sa stečenim akademskim zvanjima u radne timove u naučnim i privrednim organizacijama.

Optički disk u hijerarhiji pomoćne memorije

Miroslav Jeličić

Memorija kao element računarskog sistema deli se na glavnu i pomoćnu. Glavna memorija je malog kapaciteta (nekoliko desetina Mb) i služi za privremeno smeštanje programa i podataka dok ih procesor koristi. Programi i podaci se dugoročno smeštaju na pomoćnoj memoriji koja je velikog kapaciteta (nekoliko desetina/stotina Gb).

Pomoćna memorija sastoji se od komponenata sa različitim karakteristikama. Zato se obično govori o pomoćnoj memoriji kao hijerarhiji memorijskih komponenata. U tabeli 1. dat je prikaz hijerarhije pomoćne memorije kakva se može uočiti kod većine današnjih računarskih sistema.

Kao što se iz tabele 1. vidi, od vrha prema dnu hijerarhije razlikuju se tri glavne grupe memorije:

- Memorija sa direktnim pristupom (ONLINE STORAGE).
- Memorija sa gotovo direktnim pristupom (NEARLINE STORAGE). Ova grupa je manje poznata, a označava sve automatske sisteme pomoćne memorije gde se kao medijum koristi magnetna vrpca, a za pronalaženje i donošenje medijuma do uredjaja za čitanje/pisanje umesto čoveka/operatera koristi procesorski vodjen mehanizam (npr. mehanička ruka).
- Neautomatska memorija (OFFLINE STORAGE).

Porast zahteva za pomoćnom memorijom je oko 40% godišnje, što je posledica povećanja broja korisnika računara i postepenog osvajanja aplikacija koje umesto alfanumeričkih podataka rukuju slikama vrlo visoke rezolucije (za smeštanje jedne stranice teksta, formata A4, potrebno je 2 do 4 Kb, a za jednu sliku istog formata potrebno je oko 100 puta više memorije).

Proizvodjači magnetne memorije povećavaju gustinu upisa na magnetnom disku sa stopom od oko 20% godišnje, nude produženu glavnu memoriju do 1 Gb i programske sisteme koji treba da obezbede efikasnije korišćenje pomoćne memorije. Gustina upisa na magnetnom disku ne može da se unedogled povećava zbog ograničenja tehnologije. Način da se zadovolje rastuće potrebe za pomoćnom memorijom mogao bi da bude u komercijalnoj upotrebi digitalnog optičkog diska kao medijuma za smeštanje podataka.

1. Digitalni optički disk

Ideja o korišćenju optičke tehnologije za smeštanje podataka stara je oko 25 godina, da bi tek poslednjih godina doživela komercijalni uspeh u oblasti audio i video industrije. Ovaj uspeh otvorio je vrata novim istraživanjima i primenama, posebno u računarskoj industriji, tako da sada i veliki proizvodjači magnetne memorije puno ulažu u razvoj optičke tehnologije.

Postoje tri osnovna tipa optičkih diskova: video, audio i digitalni optički disk koji redom predstavljaju medijume za smeštanje video, audio i digitalnih informacija. Tabela 2. prikazuje tipove informacija i tehnike smeštanja.

U daljem tekstu biće rači o digitalnim optičkim diskovima, koji se razvijaju posebno za potrebe računarske industrije. Osnovna razlika izmedju magnetne i optičke tehnologije je način na koji se podaci upisuju na površinu diska. Glava za čitanje/pisanje jedinice magnetnih diskova proizvodi magnetno polje velike gustine koje menja orijentaciju memorijske ćelije (najmanjeg elementa diska koji može da bude nosilac informacije: + ili -, odn. 0 ili 1). Kod očitavanja, promena orijentacije magnetnog polja, na površini magnetnog diska, dovodi do promena napona u namotajima glave koje se dekodiraju u podatke. Optička glava za čitanje/pisanje koristi jak laserski zrak da na površini diska izazove trajne promene u vidu klobuka ili rupa, mikron-

	VRSTA MEMORIJE	KAPACITET	VREME PRISTUPA
DIREKTAN PRISTUP	CACHE	do 32 Mb	< 1 ms
	MAGNETNI DISK	do 1.2 Gb	oko 30 ms
SKORO DIREKTAN PRISTUP	MASS STORAGE	do 500 Gb	oko 16 s
	AUTOMATSKA BIBLIOTEKA MAG. KASETA	do 10000 Gb	oko 11 s
NEAUTOMATSKI PRISTUP	MAGNETNE TRAKE/ KASETE	do 200 Mb (po kolutu/ kaseti)	oko 120 s

Tabela 1. Pregled hijerarhije pomoćne memorije

		NAČIN UPISIVANJA	
		ANALOGNI	DIGITALNI
TIP INFORMACIJE	ANALOGNE	TV TEHNOLOGIJA (video disk)	HIBRIDNI SISTEM (dig.audio disk)
	DIGITALNE	HIBRIDNI SISTEM (video disk bez mogućnosti ponovnog upisa)	RAČ. TEHNOLOGIJA (digitalni disk sa mogućnošću jedno/višestrukog upisa)

Tabela 2. Tipovi informacija i tehnike smeštanja

ske veličine, koje se nalaze na malom rastojanju jedna od druge. Slab laserski zrak, kod očitavanja, raspoznaje promene na osnovu količine odbijene svetlosti od površine diska i dekodira ih u podatke.

Optički diskovi imaju nekoliko glavnih prednosti i mana u poredjenju sa magnetnim diskovima. Glavne prednosti su:

- gustina upisa je oko 10 puta veća što omogućava veći kapacitet diska,
- upis se ne oštećuje upotrebom tj. praktično je trajan,
- disk je promenljiv, nije fiksno vezan za jedinicu diskova, što omogućava izgradnju automatske biblioteke diskova (JUKEBOX),
- optički disk namenjen za masovnu distribuciju informacija može se vrlo jevtino umnožavati, kao gramofonska ploča.

Glavne mane su:

- vreme pristupa podacima je, u proseku, 10 do 20 puta veće,
- jednom upisani podaci ne mogu se brisati (ovo je privremena mana koja će biti otklonjena narednih godina, jer već postoje ohrabrujući eksperimentalni rezultati u realizaciji optičkih diskova sa mogućnošću brisanja i višestrukog upisivanja).

2. Tipovi digitalnih optičkih diskova

U zavisnosti od mogućnosti upisa postoje tri vrste digitalnih optičkih diskova:

- disk bez mogućnosti ponovnog upisa (ROOD – Read Only Optical Disk ili OROM – Optical Read Only Memory),
- disk sa mogućnošću jednokratnog upisa (WOOD – Write Once Optical Disk ili WORM – Write Once Read Many times),
- disk sa mogućnošću brisanja i višestrukog upisa (EROD – Erasable Rewritable Optical Disk ili WMRA – Write Many times Read Always).

	VELIČINA MEM. ĆELIJE	GUSTINA Mb/inch	KAPACITET Mb	VREME PRISTUPA	TRAJNOST UPISA
MAG. DISK	8 mikrona	15	1200 (15 ploča u pakovanju)	16-30 milisec	2-3 god.
OPT. DISK	0.7 mikrona	oko 100	4000 (ploča od 14 inch-a)	80-500 milisec	> 10 god.

Tabela 3. Karakteristike magnetnih i optičkih diskova

ROOD koji se može naći na tržištu je disk od 5.25 inča koji je podeljen u sektore tako da se očitava konstantnom ugaonom brzinom (CAV -Constant Angular Velocity), za razliku od audio diska CD-ROM (Compact Disk - Read Only Memory) koji se očitava konstantnom linearnom brzinom (CLV -Constant Linear Velocity). Kapacitet diska je do 400 Mb, a vreme pristupa je od 100 do 200 milisekundi. CAV omogućava brz direktan pristup, jer svaki slog može da se adresira preko broja staze i rednog broja sektora (bloka) u okviru staze, kao što se adresira slog na magnetnom disku. Medjutim, kapacitet diska je približno dvostruko manji nego odgovarajućeg CLV diska, jer je kapacitet svake staze isti, bez obzira da li je staza unutrašnja ili spoljna. ROOD se može primenjivati za distribuciju baza podataka, publikacija, softvera itd.

WOOD, koji se u literaturi često spominje je Philipsov dvostrani optički disk prečnika 12 inča i kapaciteta 2 Gb. Prosečno vreme pristupa je 137.5 milisekundi . Ovi diskovi mogu da budu ugradjeni u jukebox sa kapacitetom do 64 diska odn. 128 Gb. Dimenzije ove jedinice su 210· 150· 250 cm. Poredjenja radi, IBM-ova najmodernija komercijalna jedinica diskova IBM 3380 E ima kapacitet 5 Gb u kutiji približnih dimenzija. Glavna primena WOOD diskova, a naročito biblioteka WOOD diskova, mogla bi da bude zamenjivanje postojećih biblioteka magnetnih traka i kaseta (MAS memorija). Praksa pokazuje da se podaci arhivirani na trakama gotovo nikada ne menjaju, a da je čak u nekim primenama nepoželjano da se menjaju, pa tako mana WOOD može da se pokaže i kao prednost.

Za razliku od WOOD, kod kojih laserski zrak pravi trajne promene u obliku rupa ili klobuka na površini diska, kod EROD se menja samo stanje memorijske ćelije na površini diska (kao i kod magnetnih diskova). Uglavnom se eksperimentiše sa dve grupe materijala od kojih bi trebalo da bude izradjena površina diska i to:

- materijali kojima pomoću laserskog zraka može da se promeni stanje iz amorfnog u kristalno i obrnuto,
- magnetno-optički materijali kod kojih laser zagrevanjem jedne tačke mikronske veličine i njenim hladjenjem u prisustvu magnetnog polja dovodi do namagnetisanja tačke (do istog efekta kao i kod magnetnog diska, samo na manoj površini). Proizvodjači M-0 materijala pokušavaju da spoje dobre osobine magnetne i optičke tehnologije: privremeno menjanje stanja magnetne memorijske ćelije i sposobnost laserskog zraka da se koncentriše na mikronsku površinu.

Kristalizirajući materijali imaju manu da se olenje posle duže upotrebe, pa ih je teško naterati da promene stanje (pretpostavlja se da mogu da podnesu oko 1000 promena po memorijskoj ćeliji tj. 1000 ponovnih upisivanja ili brisanja). Izgleda da su od M-O materijali, koji pokazuju mnogo bolja svojstva za brisanje i ponovno upisivanje, prava budućnost EROD. Gustina upisa trebalo bi da bude približno jednaka gustini upisa kod WOOD. Mada još nema preciznih rezultata, smatra se da brzina pristupa, ako se ne koristi više glava za čitanje/pisanje, ne može da bude manja od 80 milisekundi.

3. Mesto digitalnog optičkog diska u hijerarhiji pomoćne memorije

Očekuje se da mali računarski sistemi prokrče put ideji o delimičnom zamenjivanju magnetnih memorijskih jedinica optičkim. To se pogotovu odnosi na uredske sisteme koji su najprirodniji korisnici kapacitativne memorije, kakva je optička. Već su instalirani takvi sistemi koji se za sada baziraju na WOOD (Filipsov MEGADOC, Kodakov KIMS,

	VRSTA MEMORIJE	KAPACITET	VREME PRISTUPA
DIREKTAN PRISTUP	CACHE	do 32 Mb	< 1 ms
	MAGNETNI DISK	do 1.2 Gb	oko 30 ms
SKORO DIREKTAN PRISTUP	AUTOMATSKA BIBLIOTEKA OPT. DISKOVA	nekoliko stotina Gb	oko 5 s

Tabela 4. je jedno vidjenje buduće hijerarhije automatizovane pomoćne memorije.

Olivetijev FILENET). Posebna je prednost ovakvih sistema što omogućavaju potpunu integraciju podataka sa raznih medijuma: magnetnih i optičkih diskova, pa i mikrofilma.

Ako ponovo pogledamo tabelu 1. i karakteristike WOOD i EROD, jasno je da digitalni optički diskovi ozbiljno konkurišu postojećoj NEARLINE i OFFLINE pomoćnoj memoriji i to zbog sledećih prednosti:

- veći kapacitet,
- manja zapremina,
- direktan pristup,
- brži pristup,
- trajnost podataka.

Ako se još ima u vidu da put od procesora do magnetnog diska ili trake, koji se sastoji i od kanala i inteligentnih kontrolnih jedinica, ne mora da pretrpi bitne izmene, može da se zaključi da je zamenjivanje npr. magnetne trake digitalnim optičkim diskom bliska budućnost (uzmimo u obzir samo podatak da jedan optički disk ima isti kapacitet kao i 40 velikih kolutova magnetnih traka).

Ostaje otvoren problem programske podrške digitalnom optičkom disku tj. izgradnja pristupnih metoda koje treba da obezbede udoban direktan i sekvencijalan pristup podacima, kakav je postignut kod magnetnog diska. To znači da treba da se obezbedi direktorijum koji sadrži tablu preslikavanja izmedju imena datoteke i njenih ekstenata na disku tako da korisnik pristupa datoteci navodjenjem imena, a ne pozicije datoteke.

Postoji samo nekoliko sistema za upravljanje memorijskom hijerarhijom koji stvaraju kod korisnika privid da je sva pomoćna memorija kojom raspolaže ONLINE. Ovi sistemi na poslednjem nivou hijerarhije rukuju sa neautomatizovanim ili automatizovanim bibliotekama traka (MAS memorija). Dobit koja bi se ostvarila uvodjenjem digitalnih optičkih diskova umesto magnetnih traka (vidi prednosti WOOD i EROD nad postojećom NEARLINE i OFFLINE memorijom), dobar je motiv da se prevazidju problemi i predrasude u povezivanju magnetnih i optičkih jedinica u istom računarskom sistemu i istoj memorijskoj hijerarhiji. Ključ je u razvoju inteligentnih kontrolnih jedinica optičkih diskova, koje obezbedjuju standardne priključke za kanalske procesore postojećih računara, i programskih sistema koji automatski upravljaju memorijskom hijerarhijom.

Slika trenutnog stanja memorijske hijerarhije u većini računarskih sistema mogla bi da se potpuno promeni narednih godina. OFFLINE memorija će prestati da postoji, jer će biti zamenjena NEARLINE memorijom: prvo magnetnom, a zatim optičkom. Tako će podaci do kojih je sada teško ili nemoguće doći biti na raspolaganju za desetak sekundi, bez angažovanja čoveka/operatera, na terminalu, bez obzira na kom se medijumu nalaze.

4. Literatura

1 Videodiscs, Compact discs and Digital Optical Disk Systems, Tony Hendley, 1985.

2 BYTE, May 1986, broj posvećen MAS memoriji.

3 Laser optical disk: the coming revolution i online storage, Communications of ACM, June 1984.

4 Pinning hopes on a vision of storage, Datamation, February 1987.

5 How to breathe life into dead data, Computer weekly, January 1987.

6 An eye on optical disks, Datamation, March 1986.

7 Cache-DASD storage design for improving system performance, IBM Systems Journal, 1985.

8 Impact of memory systems on computer architecture and system organization, IBM Systems Journal, 1986.

Pogled na TEX

Cvetana Krstev

Nije redak slučaj da neki programski sistem stekne u našoj sredini, posredstvom naučnih i stručnih knjiga i časopisa, popularnost daleko pre nego što se sam sistem pojavi u upotrebi. Takav je slučaj i sa programskim sistemom TEX koji je namenjen pripremi, pre svega matematičkih i tehničkih dokumenata za štampu [7]. Jedan deo svoje velike popularnosti u nas, TEX duguje svom autoru, izuzetnom matematičaru i programeru Donaldu Knutu. Računarska javnost je nestrpljivo očekivala da se upozna sa produktom kome je ovaj čuveni programer posvetio osam godina rada. Prva verzija TEX-a pojavila se 1979. godine a 1985. Knut rad na TEX-u smatra završenim [6]. Sada, kada je TEX stigao i u našu sredinu, možemo se uveriti da problem štampanja matematičkog teksta nije mogao biti zadovoljavajuće rešen sve dok ga se neki matematičar nije poduhvatio.

1. Klasifikacija sistema za pripremanje i obradu dokumenata

Svedoci smo nezadrživog prodiranja računara u kancelarije i štamparije, gde njihova pomoć u pripremanju raznih dokumenata za štampu postaje neizbežna. Raznovrsnost programskih sistema koji su namenjeni ovim poslovima, kao i raznovrsnost računarske opreme na kojoj su ti programski sistemi instalirani je tolika da je neupućeni potencijalni korisnik često u nedoumici koji od tih sistema najviše odgovara njegovim potrebama. Analiza većeg broja ovih programskih sistema za obradu dokumenata pokazuje da ne postoji "najbolji" od njih, već se jedino može reći da su neki od njih najbolji u svojoj kategoriji.

Programski sistemi za pripremanje i obradu dokumenata mogli bi se, najgrublje govoreći, podeliti na uredjivače i formatere. Uredjivači su programski sistemi koji služe za pripremanje dokumenata, a to znači za definisanje njegovog sadržaja i strukture. (Uredjivači su, u stvari, mnogo univerzalniji jer se, osim za pripremanje dokumenata, koriste i za programe i podatke.) Formateri su zaduženi da generišu na papiru ili na zaslonu tvrdu, odnosno meku kopiju dokumenta, iz obeležja kojima se specifikuje njegov izgled. Programski sistem TEX pripada ovoj drugoj kategoriji [3].

Programski sistemi za uredjivanje i formatiranje dokumenata mogu biti strogo razdvojeni i u prošlosti se i jesu nezavisno razvijali. Tekst dokumenta se priprema pomoću uredjivača i u njega su, takodje pomoću uredjivača, unose sva obeležja njegove strukture i izgleda. "Čisti" formater, u kakve spada i TEX, prihvata ovako pripremljen tekst i od njega formira dokument pripremljen za štampu. Kasnije su razvijeni mnogi takozvani sistemi za obradu reči (engl. *word processor*) koji u sebi integrišu funkcije uredjivača i formatera. Oni korisniku omogućavaju da u toku pripremanja i menjanja dokumenta na zaslonu vide izgled dokumenta, koji, više ili manje, odgovara izgledu koji će dokument dobiti na tvrdoj kopiji. Svaka izmena u sadržaju, strukturi ili opisu izgleda dokumenta, trenutno se odražava na izgledu dokumenta na zaslonu. Danas je široka upotreba ekrana visoke rezolucije omogućila razvoj takozvanih WYSIWYG sistema (od engl. *what-you-see-is-what-you-get*, ono što vidiš to ćeš i dobiti). Po analogiji sa programskim jezicima, ove dve vrste formatera se često nazivaju i kompilatori dokumenta, odnosno interpretatori dokumenta.

Imajući u vidu način rada ovih formatera, oni se još nazivaju i formateri po porcijama, odnosno interaktivni formateri.

I u odnosu na korisničku sumedju, tj. na jezik formatiranja koji koriste, formateri se mogu podeliti u dve osnovne kategorije. Jedni od njih, u koje spada i TEX, koriste proceduralni jezik koji omogućava da se njime tačno preciziraju izgled, položaj, veličina itd. pojedinih objekata dokumenta. Tako u TEX-u postoje npr. komande `\centerline` kojom se tekst koji sledi centrira, `\hsize` kojom se definiše

širina retka ili \parindent kojom se definiše koliko će prvi redak pasusa biti uvučen u odnosu na levu marginu. Druga kategorija formatera koristi deklarativni jezik kojim korisnik može da specifikuje apstraktne objekte dokumenta, kakvi su poglavlje, odeljak, fusnota i sl., dok se sam izgled tih objekata može definisati izvan dokumenta i zavisiće od toga za koje svrhe se dokument štampa, na kom izlaznom uredjaju, itd. Podela formatera na one koji koriste proceduralni, odnosno deklarativni jezik, nije uvek ovako stroga. Tako se npr. u TEX-u, koji u osnovi koristi proceduralni jezik, mogu pomoću tzv. makro-definicija specifikovati i apstraktni objekti, o čemu će kasnije biti više reči.

Postoje i drugi kriterijumi prema kojima se formateri mogu razvrstavati. Tako postoje formateri koji obradjuju dokumente koji sadrže i tekst i sliku za razliku od onih koji, kao TEX, obradjuju samo tekst. Razlikuju se takodje formateri koji pripremaju dokument prevashodno za računarski izlaz, kakav je štampač, od onih koji slažu (engl. *typesetting*) dokument za fotoslog (koji se, iako takodje može biti izlazni uredjaj računara, ne može smatrati uobičajenim računarskim izlazom).

Iz ove grube klasifikacije formatera moglo bi se zaključiti da TEX ima nekih nedostataka u odnosu na druge ovde spomenute, koji npr. integrišu funkcije uredjivača i formatera, tekst i sliku, itd. Ne treba, medjutim, izgubiti iz vida da je TEX pre svega sistem za slaganje teksta čiji cilj nije toliko predusretljivost prema autoru koliko želja da se proizvede dokument iz tehničkog domena vrhunskog kvaliteta. Da bi se taj cilj ostvario, u TEX su ugradjena neka izuzetno moćna sredstva, koja ga čine veoma fleksibilnim i sposobnim da savlada najteže zadatke. O najznačajnijim od njih sada će biti više reči.

2. Standardizacija formalizma za zapis matematičkih formula

Iz vremena ručnog slaganja teksta potiče uvrenje da je matematički tekst posebno težak za slaganje. Složenost matematičkog teksta ne potiče ponajpre od upotrebe velikog broja specijalnih karaktera i istovremene upotrebe više alfabeta, već pre od upotrebe višestrukih supskripta i superskripta, od neophodnosti da se podese sve jednakosti, jednom rečju, od dvodimenzionalnosti zapisa matematičkih formula. Problem linearizacije ovakvog zapisa poznat je i iz klasičnih programskih jezika, ali je za potrebe slaganja matematičkog teksta potreban nešto drukčiji pristup. U TEX-u je razvijen mnemonički jezik kojim se mogu specifikovati sve zamršenosti matematičkog teksta. Osnovu mnemoničkog jezika TEX-a čine, takozvane, kontrolne sekvencije koje, neformalno govoreći, predstavljaju nisku slova koja počinje isključnim karakterom (obično je to obrnuta kosa crta \). Na taj način, TEX ne koristi ključne reči pa nije potrebno da se svaki token teksta sravnjuje sa nekim ugradjenim rečnikom ključnih reči. Posebno je značajno da za unošenje teksta nije potrebna nikakva specijalna tastatura obogaćena mnogobrojnim matematičkim simbolima – korišćenje kontrolnih sekvencija omogućava da se i najsloženije matematičke formule mogu predstaviti pomoću svake uobičajene računarske tastature.

Linearizovane formule koje su zapisane ovim jezikom nisu, naravno, lake za čitanje kao formule u uobičajenom dvodimenzionalnom zapisu. Iskustvo, medjutim, pokazuje da i matematičari bez iskustva u slaganju teksta kao i daktilografi bez iskustva u slaganju matematičkog teksta mogu u relativno kratkom vremenu da nauče da prevedu uobičajeni dvodimenzionalni u jednodimenzionalan TEX zapis.

Značaj standardizacije zapisa matematičkih formula ogleda se i u oblasti koja je tesno povezana sa automatskom pripremom teksta za štampu. Budućnost nagoveštava uvodjenje "elektronskih" časopisa u biblioteke [8]. Čuvanje časopisa u vidu grafičke slike visoke rezolucije, zbog različitosti korišćenog hardvera i količine memorije koju takvo rešenje zahteva, neće u skoroj budućnosti biti moguće. Prirodan oblik za skladištenje časopisa biće izvorni oblik teksta. Ovakav oblik časopisa postaje deo bibliografske baze podataka. Standardizovan zapis matematičkih formula je od suštinskog značaja za mogućnost pretraživanja ovakvih baza podataka.

3. Izlaz nezavisan od izlaznog uredjaja

Tekst obradjen TEX procesorom je u formi koja ne zavisi od izlaznog uredjaja. U toku slaganja (kompiliranja) teksta, TEX, naime, rukuje samo apstraktnim pojmom "kućice" i interesuju ga samo dimenzije tih "kućica", njihova visina, širina i dubina. Sadržaj ovih kućica predstavlja grafički izgled svakog karaktera koji, pak, zavisi od karakteristika konkretnog izlaznog uredjaja. Takodje, sve verifikovane verzije TEX-a, bilo da su to verzije na centralnom računaru ili nekom personalnom računaru, prihvataju potpuno iste ulazne tekstove (teke) i za njih proizvode potpuno iste izlazne teke. To omogućava da se jednom pripremljen tekst može

obraditi na svakom računaru na kome je TEX instaliran, a da se, takodje, jednom obradjeni tekst može odštampati na raznim izlaznim uredjajima: matričnom štampaču, laserskom štampaču, uredjaju za fotoslog. Kvalitet izlaza zavisiće samo od kvaliteta izlaznog uredjaja. Potrebno je jedino da za konkretan štampač i za korišćene pismovne vrste budu razvijeni odgovarajući pogonaši [10].

4. Makro definicije

Mogućnosti makro definicija koje su ugradjene u TEX su izuzetno velike. Na najjednostavnijem nivou one korisniku omogućavaju da imenuje delove teksta koj se često ponavlja (kakav je, npr., zapis $x_1, x_2, \ldots, x_n$ u matematici). Na višem nivou one omogućavaju korisniku ili grupi korisnika da prilagodi TEX tako da što bolje odgovori njegovim potrebama. Pri tome se izvorni kôd programa TEX ne mora menjati što je od izuzetnog značaja. Kao što je već pomenuto, svaki program koji se pod imenom TEX oglašava za novi računarski sistem proveren je, što znači da se može sa izvesnošću tvrditi da isti ulazni tekst kompiliran ma kojim valjano implementiranim TEX-om daje uvek isti kôd nezavisan od uredjaja. Svaka intervencija korisnika u izvornom kôdu dovela bi do nekompatibilnosti raznih implementacija. Korišćenjem makro komandi ne narušava se stabilnost TEX-a, a u isto vreme se ne gubi ni osnovna prednost ovog sistema: mogućnost razmene TEX ulaznih teka. Potrebno je samo uz ulaznu teku priložiti i teku koja sadrži makro definicije.

Razvijeno je više opštih paketa makro komandi koji su namenjeni širokoj upotrebi. Za potrebe Američkog matematičkog društva razvijen je paket AMS-TEX kojim se matematičarima koji objavljuju u ovom časopisu omogućava da, ukoliko to žele, sami slože svoj članak u formi koju zahteva dizajn nekog od časopisa ovog društva. LATEX je kolekcija makro komandi koja korisniku omogućava da se više skoncentriše na strukturu dokumenta nego na tipografske komande [1]. Time se upotreba TEX-a olakšava, premda se gubi ponešto od njegovih kvaliteta. Ipak, sve primarne komande TEX-a su korisniku, ukoliko su mu potrebne, i dalje na raspolaganju.

5. Mogućnost dizajniranja fontova

METAFONT je sistem za automatsko dizajniranje fontova, čiji je autor takodje D. Knut [5]. Da bi se shvatio značaj funkcionalne povezanosti jednog ovakvog sistema sa sistemom za slaganje teksta kakav je TEX, vratićemo se malo u prošlost.[5] U vreme ručnog ili poluautomatskog slaganja teksta, u metalu izlivena slova (odgovarajućeg dizajna i veličine) slagala su se u ploču sa koje se pravio otisak. Uredjaji za fotoslog doneli su revoluciju u grafičku industriju. U prvim uredjajima ovog tipa, oblici slova bili su uskladišteni kao mali negativi na rotirajućem disku. Ploča potrebna za štampu dobijana je osvetljavanjem filma preko negativa odgovarajućeg slova. Revolucija u grafičkoj industriji nije se ovde zaustavila. Sledeća generacija uredjaja za fotoslog zamenila je ovaj "analogni" proces "digitalnim". Stranica negativa filma podeljena je na milione kvadrata koji se nazivaju pikseli - ima ih oko 1000 po inču. Za svaki od ovih piksela uredjaj za fotoslog treba da odluči da li će se, tankim elektronskim ili laserskim zrakom, osvetliti ili će ostati neosvetljen.

Kako su uskladišteni oblici slova u ovom poslednjem slučaju? Oni se, lako se može zaključiti, pamte u vidu matrica 0 i 1 gde, na primer, 0 označava neosvetljeno a 1 osvetljeno mesto na filmu. Dimenzija matrice, očigledno, zavisi od rezolucije izlaznog uredjaja za fotoslog. Postavlja se, medjutim, pitanje kako ovakve matrice formirati? Najčešća praksa je da se, uz pomoć složenih uredjaja i kompleksnih programa , "digitalizuju" postojeći metalni fontovi. Naknadna ručna dorada ovako dobijenih fontova je neizbežna.

Zašto ovakav pristup nije najpogodniji? Pre svega, za "digitalizaciju" fonta potrebna je skupa oprema koja mnogima nije dostupna. Zatim, većina poznatih fontova je zaštićena te je potrebna dozvola za njihovo presnimavanje, a ovakvu dozvolu grafička industrija obično ne daje. Dok su ovi razlozi uglavnom tehničke prirode, ovakvom pristupu mogu se prebaciti i suštinski nedostaci. Kako ćemo u jednom digitalizovani font uvesti novi simbol? Ako nam uredjaj za digitalizaciju nije pri ruci, ostaje nam jedino da ručno formiramo matricu 0 i 1, što je pri rezoluciji od 1000 linija po inču vrlo mukotrpan, ako ne i sasvim neizvodljiv posao. I na kraju, font se digitalizuje imajući u vidu rezoluciju odredjenog izlaznog uredjaja, npr. za rezoluciju od 300 linija po inču kakvu ima laserski štampač. Ukoliko isti font želimo da upotrebimo na uredjaju za fotoslog koji ima rezoluciju od 1000 linija po inču, postupak se mora ponoviti.

METAFONT primenjuje sasvim drukčiji pristup. Oblici slova definišu se u formi koja ne zavisi od izlaznog uredjaja. Za svako slovo ili simbol korisnik piše "program" u METAFONT jeziku [4]. Za razliku od uobičajenih računarskih programa, programi u

METAFONT -u su deklarativni, a ne proceduralni. Programom se definišu glavne karakteristike oblika koji se želi dobiti, kao što su njegove dimenzije i položaj referentnih tačaka. Kako program treba da simulira iscrtavanje željenog oblika kaligrafskim perom, definišu se i oblik i debljina takvog pera, njegov položaj i jačina "pritiskanje" po papiru. Na osnovu ovako postavljenog zadatka, računar sam postavlja i rešava odgovarajuće jednačine koje treba da daju najprijatniju krivu koja zadovoljava postavljene uslove [2]. Promenom samo jednog od ovih parametara, može se iz iste definicije fonta dobiti mnogo različitih stilova: npr. zadebljana slova, iskošena slova i sl. Svi ti stilovi su, medjutim, medjusobno povezani i odražavaju zajedničko poreklo. Tako se, na primer, može definisati font koji se koristi za supskripte i superskripte, kao i fontovi grčkih slova i matematičkih simbola koji su potpuno konzistentni sa osnovnim fontom u kome se neki matematički tekst slaže.

Medjutim, METAFONT nije samo program za dizajniranje fontova. On, takodje, kreira računarske teke koje su neophodne da bi se dizajnirani font predstavio TEX-u i izlaznom uredjaju. Za svaki novi izlazni uredjaj potrebno je još samo napisati relativno jednostavan program koji povezuje METAFONT sa tim uredjajem. Ali čim se to uradi, velika i praktično neograničena familija METAFONT fontova postaje odmah dostupna na tom uredjaju.

6. Finese slaganje teksta

Kroz viševekovnu istoriju štampane reči postavljeni su mnogi iskustveni zakoni kojih, ma koliko knjiga pročitali, često nismo ni svesni sve do onog trenutka dok ne vidimo narušen neki od tih zakona. Svi ti zakoni, osim što doprinose lepšem izgledu štampanog teksta, služe, pre svega tome da omoguće bolju čitljivost. Prilikom ručnog ili poluautomatskog slaganja teksta, zavisilo je od savesnosti i stručnosti slagača da se ne naruši neki od ovih zakona. Prelazak na potpuno automatsko slaganje teksta, svakako ne bi smelo da, u ime brzine i drugih ekonomskih razloga, dovede do umanjivanja kvaliteta. Naprotiv, kvalitet bi trebalo da se poboljšava. I sam autor sistema TEX smatra da njegov sistem, u odnosu na mnoge druge sisteme slične namene koji su razvijeni u poslednjih petnaestak godina, donosi možda samo nešto više "finoće" u mnogim detaljima. Pomenimo ovde samo neke od njih.

Prilikom slaganja matematičkih formula, TEX sam vodi računa o mnogim zakonitostima koje važe u ovom području. U matematičkom režimu rada, npr., TEX automatski prelazi na korišćenje kurziva, sam vodi računa o specifičnostima korišćenja razmaka u formulama, o razlikama u predstavljanju istaknutih formula i formula unutar teksta, sam bira veličine odredjenih simbola, kao što su razne vrste zagrada ili oznaka za kvadratni koren. Ukoliko neka formula ne može cela da stane u jedan redak teksta, TEX se sam stara da formulu podeli na najpogodnijem mestu.

Na čitljivost teksta u najvećoj meri utiče pravilna upotreba razmaka. Pravilo je da unutar jednog retka razmaci izmedju reči moraju biti jednaki i da, pri tome, ne smeju biti ni preveliki ni premali. Postoje, ipak, i izuzeci od ovog pravila. Poželjno je, na primer, da razmaci posle interpunkcijskih znakova budu nešto veći. O svemu ovom TEX sam vodi računa.

Podešavanje jednakog razmaka izmedju reči u retku predstavljalo je kod ručnog slaganja teksta netrivijalan problem. Sada, kada slaganje jednog retka zapravo podrazumeva pravilno podešavanje 0 i 1 u matrici koja predstavlja redak, problem se svodi na proračunavanje broja 0 koje treba postaviti izmedju reči. Sa ovim povezani problem odredjivanja kraja retka, ne rešava se ovako trivijalno. Kod ručnog slaganja teksta, slagač je sam odlučivao da li je bolje da reč koja pretiče "zgura" u redak, smanjujući razmak izmedju reči ili je bolje da tu reč prebaci u novi redak, proširujući razmak. Slagač se nikada nije vraćao unazad: jednom složeni redak nije se više popravljao. Većina računarskih programa postupa na isti način, što često dovodi do poražavajućih rezultata: "labavo" složen redak iza koga neposredno sledi "gusto" složen radak.

U TEX je ugradjen neobično promišljen algoritam za rešavanje ovog problema. Definitivne granice izmedju redaka se ne odredjuju sve dok se ne učita ceo pasus. Tada se usvajaju one granice koje će dati najbolje rezultate u odnosu na strogo postavljene kriterijume: veličina razmaka izmedju reči treba da teži optimalnoj, veličina razmaka izmedju reči u susednim redova ne bi trebalo da se drastično razlikuje, dva susedna retka ne bi trebalo da se završavaju podeljenom reči kao ni pretposlednji redak u pasusu i sl.

7. TEX na malim računarima

Prve verzije TEX-a su instalirane samo na velikim računarima, što se može delom pripisati veličini samog programa. Kako veliki računarski sistemi nisu ni približno tako predusretljivi kao mali računari, upotreba TEX-a na velikom računaru je odbijala mnoge potencijalne korinike koji nisu pripadali računarskoj profesiji.

Ulaskom u sve širu upotrebu malih računara, počele su se i za njih pojavljivati verifikovane verzije TEX-a. Najpoznatije verzije TEX-a za IBM-kompatibilne personalne računare su PCTEX, proizvod Presonal TEX, Inc. i MicroTEX, proizvod Addison-Wesley Publishing Co. Medjutim, TEX se na malim računarima našao u konkurenciji sa velikim brojem raznolikih i izuzetno predusretljivih programa za obradu reči. U Vesniku Američkog matematičkog društva [9] objavljen je januara 1986. godine izveštaj Bostonskog računarskog društva o projektu evaluacije softvera za obradu reči u tehničkom domenu na personalnim računarima. Ovo istraživanje je obuhvatilo 38 raznih programa za obradu reči a medju njima i dve gore spomenute verzije TEX-a. Softver je procenjivan u odnosu na korisničku sumedju, prilagodljivost, mogućnost prenosivosti na različite sisteme, kvalitet tvrde kopije, kompetenciju u domenu tehničkog teksta i lakoću i brzinu kojom se softverom može ovladati.

Ova analiza je pre svega istakla da TEX nije klasični program za obradu reči, što podrazumeva integraciju uredjivača i formatera. Kao što je na početku ovog prikaza istaknuto, TEX je formater, ili pre program za slaganje teksta, pri čemu se proces slaganja odvija po partijama: u toku ukucavanja teksta, korisnik ne može da vidi kako će sve to što kuca na kraju izgledati. Na personalnom računaru mu jedinu pomoć mogu pružiti programi za prikazivanje kompiliranog teksta na zaslonu, koji korisniku omogućavaju da tekst pregleda na zaslonu umesto da za svaku od radnih verzija pravi tvrdu kopiju. Analiza je, ipak, pokazala da TEX, iako se ne može podičiti predusretljivošću programa za obradu reči, poseduje snagu i fleksibilnost koja daleko nadmašuje svaki od njih. Zaključak ovog izveštaja je, da će svako ko uloži napor koji je nesumljivo potreban da bi se TEX-om ovladalo, biti bogato nagradjen mogućnošću da svoje tehničke, ali i sve druge tekstove, priredi na svoje zadovoljstvo i zadovoljstvo budućih čitalaca.

Literatura

1. Lamport, L.: LaTEX: A Document Preparation System, Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.
2. Knuth, D.E.: TEX and METAFONT : New Directions in Typesetting, Digital Press & American Mathematical Society, 1979.
3. Knuth, D.E.: The TEXbook, Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.
4. Knuth, D.E.: A Course on METAFONT Programming, TUGboat 5 (2), November 1984. pp. 105–118.
5. Knuth, D.E.: The METAFONT book, Addison-Wesley Publishing Company, Reading, Massachussetts, 1986.
6. Knuth, D.E. (interview): Computer Science Considerations, Byte, February 1986. pp. 169–172.
7. Krstev, C.: Prikaz sistema TEX i METAFONT D.E. Knuta, Zbornik VIII naučno-stručnog simpozijuma INTERGRAFIKA'85, Zagreb, mart 1985.
8. Palais, R.S.: Message from the Chairman, TUGboat 1 (1), Octobar 1980, pp. 3–7.
9. Palais, R.S.: Mathematical Text Processing, Notices of American Mathematical Society 33 (1), Januray 1986.
10. Whitney, R.: Introduction to TEX and TUG for the New Users, TUGboat 3 (2), May 1982. pp. 9–12.

Prikaz "Časopisa o računarski podržanom učenju"

Veljko Spasić

U Velikoj Britaniji četiri puta godišnje izlazi "Časopis o računarski podržanom učenju" ("Journal of Computer Assisted Learning"). Primena računara u obrazovanju intenzivno se razvija u Velikoj Britaniji već preko petnaest godina, a časopis je nastao 1985, dakle u vreme kada su se stekla značajna iskustva u ovoj oblasti.

Časopis pokriva osnovne oblasti primene računara u obrazovanju, kao što su koncepti obrazovnog softvera, provere vrednosti ove vrste softvera, metodologiju i probleme vezane za pisanje programa koji se primenjuju u obrazovanju. Zastupljeni su tekstovi o potrebnoj računarskoj opremi, njenim karakeristikama i stalno prisutnom razvoju. Časopis se takodje bavi problematikom vezanom za neophodno usavršavanje nastavnog kadra koji uvodi novu računarsku tehnologijuu u proces obrazovanja. Konačno, prikazi knjiga, časopisa i publikovanih radova iz oblasti računarski podržanog učenja, zaključuju svaki broj.

Tekstovi su pretežno britanskih autora, a gradiraju od prikaza do originalnih doprinosa. Dijapazon tema je nužno širok što odslikava stanje u multidisciplinarnoj oblasti primene računara u obrazovanju.

Kao ilustraciju navodimo sadržaje dva broja iz 1985. i 1987. godine:

VOL 1 BROJ 3 1985.

- Upravljanje računarskom opremom u školama: D. Esterson
- Računari i diskusije u okviru manjih grupa: R. Cummings
- Perspektive inteligentnog računarski podržanog učenja: J. Self
- Rešavanje problema pomoću programa za unakrsna izračunavanja: P. Catterall & R. Lewis
- Prikazi
- E.R.I.C. abstrakti

VOL 3 BROJ 1 1987.

- Kako deca vežbaju matematiku pomoću jezika LOGO: R. Noss
- Iskustvo uvodjenja koncepta veštačke inteligencije u srednje škole: C. Casey, A. O. Adewumi, A. Hart & G. O Hare
- Na putu za Jerusalim: računarska simulaciona igra za nastavu teologije: A. Martin & I. Smyth
- Politika regionalnih obrazovnih institucija pri uvodjenju računara u obrazovanje: F. J. Burdett
- Evaluacija mikroračunarskih programa: područje debata: V. M. Johnston
- Prikazi
- E.R.I.C. Abstrakti

Časopis je namenjen svima koji se bave računarski podržanim učenjem kako na univerzitetu tako i u školama, a svakako nastavnicima (i njihovim učenicima) koji već primenjuju računare u nastavi svog predmeta ili direktno u nastavi računarstva.

Sa ovim časopisom najtešnju saradnju ima naše "Računarstvo u nauci i obrazovanju" kroz sporazum o razmeni tekstova i preko redakcije u kojoj saradjuju pojedini članovi uredjivačkog odbora britanskog časopisa. Planira se i izdavanje zajedničkog godišnjeg izbora teksova iz oba časopisa na oba jezika.

Časopis o računarski podržanom učenju izdaje britanski izdavač Blackwell Scientific Publications, Osney Mead, Oxford, OX2 OEL, GB. Izlazi tromesečno uz godišnju pretplatu od 14.50 funti za Evropu ako su pretplatnici privatna lica. Za institucije godišnja pretplata iznosi 54 funte.

Glavni i odgovorni urednik časopisa je Robert Lewis, Univerzitet u Lankasteru, a redakciju sačinjavaju stručnjaci sa nekoliko britanskih univerziteta i računskih centara.

Logic colloquium 87.

Žarko Mijajlović

U nizu nekoliko stotina malih i velikih skupova matematičara koji se održavaju svake godine širom sveta, na velikom broju ovih konferencija jedna od glavnih tema odnosi se na računarstvo. Ovo je slučaj i sa medjunarodnim naučnim simpozijumom Logic Colloquium 87.

Već dve decenije medjunarodno udruženje za matematičku logiku (Association for Symbolic Logic - ASL) održava svake godine takozvanu letnju konferenciju u nekom od evropskih gradova pod nazivom Logic Colloquium. Prošle godine ova konferencija je održana u Halu (Kingston upon Hull - britanska luka na severnom moru odakle je isplovio Aleksandar Selkerk, glavni junak Defoovog romana Robinson Kruso), 1985. god. u Parizu, 1984. god. u Mančesteru, 1983. god. u Ahenu, 1982. god. u Firenci, 1981. god. u Marseju,.... Ove godine Logic Colloquium održan je u Granadi u vremenu 20-25 juli u organizaciji španskog nacionalnog matematičkog časopisa *Teorema* i Univerziteta u Granadi, kao i uz pomoć drugih lokalnih ali i medjunarodnih organizacija (na primer, *International Union of History and Philosophy*, koja takodje ove godine ima svoj kongres u Moskvi). Glavne teme predavanja po pozivu i saopštenja učesnika bile su iz oblasti: a) teorija modela i teorija dokaza, b) teorija skupova, c) filozofija i matematička logika, d) računarstvo i matematička logika. U ovom pregledu osvrnućemo se na predavanja i saopštenja koja se odnose na ovu poslednju oblast.

Evo najpre jedne male statistike. Od četrnaest predavanja po pozivu, pet se odnosilo ili je bilo u vezi sa računarstvom: J.Fenstad, (Oslo), *Logic and natural language systems*; G.Jaeger (Cirih), *Type theory and explicit mathematics*; A. Kučera (Prag), *Properties of diagonally non-recursive functions*; J. Van Benthem (Amsterdam) *Paralles in the semantics of natural languages and programming languages*; J.Meseguer (Kalifornija), *On the semantics of logical programming languages*. Od oko šezdeset saopštenja, otprilike petnaest bilo je sa temama iz računarstva. Po tematici, ovi radovi su se mogli podeliti na tri grupe: a) teorija algoritama, b) semantika programskih jezika i prirodni jezici, c) logike (kao formalni sistemi) nastale u vezi sa računarstvom (kao što je na primer dinamička logika).

Radovi iz oblasti teorije algoritama bili su najdalje od konkretnih primena u računarstvu. Naime, oni su više osvetljavali neke teorijske vidove zasnivanja računarstva. Verovatno je najinteresantniji prilog ove vrste bio već pomenut Kučerov rad koji se bavi izučavanjem aritmetičkih funkcija sa ovom osobinom: $f(e) \neq \pi_e(\ell)$ za sve $e \in N$, N je skup prirodnih brojeva, gde je π_e efektivna enumeracija svih parcijalno izračunljivih funkcija. Kučera u radu raspravlja pitanje složenosti ove klase funkcija u smislu teorije (Turigovih) stepena nerešivosti.

Druge dve teme neposrednije su vezane za konkretne probleme računarstva. Ispostavlja se da je naročito aktuelan problem definisanja (formalne) semantike programskih jezika, kao i fragmenata prirodnih jezika (o značaju ovog problema u računarstvu čitalac se može upoznati iz prikladnog članka u prvom broju ovog časopisa). Zašto se u ovaj kontekst stavljaju prirodni jezici, jasno je ako se zna da se postojeći programski jezici šire a novi projektuju tako da sadrže značajne fragmente prirodnih jezika. Naravno, ovaj trend odgovara predvidjanju da će se komunikacija sa računarima pete generacije obavljati na jezicima sličnim prirodnim. Programski jezik PROLOG je u ovom kontekstu zauzimao značajno mesto. Ipak, na konferenciji smo imali prilike da čujemo tek nekoliko novih tehničkih rezultata. To je i razumljivo, jer od otkrića Montegueovih gramatika (logičkog formalizma pogodnog za analizu prirodnih jezika) i denotacijskih semantika D.Scotta, nije bilo velikih prodora u ovoj oblasti. S druge strane, Van Benthem je održao zanimljivo pregledno predavanje na temu primena specijalnih logika u računarstvu. Videli smo da su donedavno egzotične logike, kao što su modalne, temporalne i druge našle važne primene u računarstvu. Interesantno je da je Van Benthem prošle jeseni bio gost

Matematičkog instituta u Beogradu, gde je održao slično predavanje.

Bilo je sporadičnih priloga koji ne spadaju u ove tri oblasti, ali su ipak u nekoj vezi sa računarstvom. Tako smo čuli saopštenje P.Nanda (Piza) o rasplinutim (fuzzy) skupovima, dok je Van Lambalgen (Amsterdam) imao interesantan rad o slučajnim nizovima u kojem je dao komparativnu analizu izmedju poluformalne Von Misesove definicije i Martin-Löfove definicije slučajnih nizova. Podsećamo čitaoca da u Martin-Löfovoj definiciji ključnu ulogu ima pojam rekurzivne (tj. izračunljive) funkcije. Naravno, na ranijim konferencijama imali smo prilike da čujemo i egzotičnije teme. Sećam se, u Parizu 1985. godine, čuli smo pregledno predavanje iz primene logike u kriptografiji. Apstrakti radova prezentiranih na ovom skupu biće objavljeni u časopisu *Journal of Symbolic Logic*, službenom glasilu ASL-a.

Kao i na svim dosadašnjim konferencijama, velika većina učesnika bila je smeštena na jednom mestu - u studentskom domu. Ovo je pružilo priliku učesnicima da se druže i van zvaničnog programa i da eventualno završe svoje naučne rasprave započete na sednicama konferencije. Skupu je prisustvovalo oko 150 učesnika iz većeg broja zemalja (SAD, Poljska, Čehoslovačka, V.Britanija, Francuska, Madjarska, Italija, Španija itd.). Iz Jugoslavije bilo je dva učesnika; jedan od njih je autor ovog prikaza.

Prikaz 2. jugoslovenske konferencije "Računar u obrazovanju"

Duško Vitas, Dragan Vasić

U Novoj Gorici je od 1. do 3.10.1987. godine održana 2. jugoslovenska konferencija o politici modernizacije obrazovne tehnologije sa temom "Računar u obrazovanju". Konferenciju su organizovali Stalna konferencija zavoda za obrazovanje i vaspitanje republika i pokrajina, Zavod za školstvo SR Slovenije i Zavod za prosvjetno-pedagošku službu SR Hrvatske. Generalni pokrovitelj je bila Iskra-Delta, u čijem je obrazovnom centru održana konferencija.

Konferenciji je prisustvovalo preko 160 učesnika iz svih krajeva zemlje. Pored domaćih učesnika, konferenciji su prisustvovali i predstavnik projekta OECD-CERI, gospodin Pier Dige, i predstavnik UNESCO-a, gospodin Anri Diezed. Po prvi put su na jednom ovakvom jugoslovenskom skupu uzeli istovremeno učešće stručnjaci iz oblasti informatike i stručnjaci iz oblasti pedagogije i psihologije, kao i predstavnici nadležnih republičkih i pokrajinskih organa.

Zamisao organizatora je bila da problem uvodjenja računara u obrazovanje razloži na pet nezavisnih i komplementarnih tema. Ove teme, koje obuhvataju ukupnost problema informatizacije škole, su: obrazovanje i usavršavanje nastavnika, programska oprema (softver), mašinska oprema (hardver), naučno-istraživački rad i finansiranje.

Na plenarnom zasedanju prvog dana rada Konferencije su podneti referati, koji su poslužili kao osnova za diskusije po radnim grupama. Uvodni referat "Obrazovanje i nove informacione tehnologije" podneo je dr Vladimir Srića, predsednik Komiteta za nauku, tehnologiju i informatiku SR Hrvatske. O "Obrazovanju nastavnika za korišćenje računara u vaspitno obrazovnom procesu" govorio je Borko Boranić iz Zavoda za PS SRH, a dr Vladimir Batagelj je podneo referat — "Neka razmišljanja o obrazovnoj računarskoj programskoj opremi". Boris Sović iz Elektro-kovine, Maribor, obavestio je skup o katastrofalnom stanju na području školskog hardvera u referatu "Računarska mašinska oprema u obrazovanju". Prof. dr Vladimir Mužić sa Sveučilišta u Zagrebu podneo je referat pod naslovom "Naučno-istraživački rad na području upotrebe računara u vaspitno-obrazovnom radu". Dragoljub Pavlović, sekretar Republičke zajednice osnovnog obrazovanja SR Srbije, je izložio finansijske probleme sa kojima se susreću škole pri uvodjenju informatike i računarstva. Najveći deo ovih referata je objavljen u posebnom dodatku "Kompjutor u školi", školskih novina iz Zagreba.

Na popodnevnom zasedanju, mr Nada Tomić iz Zagreba je u referatu "Prikaz projekta obrazovanje i nove informacione tehnologije", izložila rezultate prikupljene u okviru 2. faze medjunarodnog projekta OECD-CERI posvećenog navedenoj temi.

Pjer Dige je, potom, u izuzetno nadahnutom izlaganju "Nove informacione tehnologije u obrazovanju—projekat OECD-CERI", dao pregled osnovnih trendova na ovom području u zemljama — članicama OECD-a. Posebno je naglašena primarna uloga obrazovanja nastavnika u procesu informatizacije škole kao i neophodnost veza izmedju škole i univerziteta u ovom procesu. O problemu obrazovnog softvera izneo je zanimljivo zapažanje da se u postojećim primerima edukativnih programa prepoznaju zapravo različite pedagoške teorije, te da se i kroz obrazovni softver ogleda nasledje klasične škole. Naročit značaj je pridao mogućnosti ekspliciranja meta-kognitivnih procesa (razvoju sposobnosti sinteze, kritičkog duha...) u funkciji kompjuterizacije škole.

Anri Diezed je u referatu "Nove informacione tehnologije u obrazovanju u državama — članicama UNESCO-a" izložio neke probleme sa kojima se sreću pojedini obrazovni sistemi.

Drugog dana rada konferencije, radilo se po radnim grupama. Pripremljeni su zaključci konferencije koje navodimo integralno u Prilogu. Redakcija pojedinih zaključaka je protekla u živoj i žustroj diskusiji. Tokom popodneva drugog dana kon-

ferencije, organizator je priredio obilazak oglednih učionica u Staroj i Novoj Gorici.

Trećeg dana konferencije, na plenarnom zasedanju, predstavljene su knjige dr V. Batagelja "S računarom u matematiku" (izdanje Državne založbe SRS, 1987), prof. dr V. Mužića "Kompjutor u preobražaju škole" (Školska knjiga, Zagreb, 1986) i prof. dr N. Parezanovića "Računarstvo i informatika", kao i našeg časopisa.

Potom su usvojeni zaključci konferencije uz opšte izraženu želju da se na planu računarstva i informatike više nego do sada saradjuje na jugoslovenskom planu.

Ostaje utisak da je konferencija prikazala svu složenost sadašnje situacije na planu informatizacije škole: nepostojanje dugoročne racionalne politike na planu računarstva i informatike u zemlji se u potpunosti odrazilo i na ovaj proces. S druge strane, posebno kroz sagledavanje inostranih iskustava, može se reći da još uvek postoje realne mogućnosti za uključivanje u svetske tokove na ovom planu. Naime, na planu koncepcije uvodjenja novih informacionih tehnologija u proces obrazovanja, može se reći da je stvoren jedan jasan i savremen plan: ono što bitno nedostaje u ovom trenutku je značajnija finansijska podrška naučno-istraživačkom radu, koji treba da stvori osnovu za sistematsku i organizovanu produkciju obrazovnog softvera.

Zaključci radnih grupa

A. Radna grupa za obrazovanje nastavnika

Konferencija preporučuje da se u proces vaspitanja i obrazovanja odlučnije unose dostignuća razvoja informatike i da se prati razvoj informatičkog društva.

Na području obrazovanja radnika u vaspitanju i obrazovanju Konferencija preporučuje:

(1) da se dopune programi studija nastavnika u redovnom školovanju opštim informatičkim sadržajima i informatičkim sadržajima usmerenim na struku;

(2) da se obezbedi redovno školovanje nastavnika informatičkih predmeta za rad na svim nivoima obrazovanja;

(3) da se organizuje dopunsko obrazovanje svih radnika u vaspitanju i obrazovanju kako bi ovladali opštim informatičkim sadržajima i bili osposobljeni da primenjuju informatičke metode i tehnike u vaspitno-obrazovnom procesu;

(4) da se u svakoj vaspitno-obrazovnoj organizaciji postepeno zapošljavaju specijalisti – stručno i pedagoški osposobljeni nastavnici, koji će pomagati ostalim nastavnicima, podučavati ih i savetovati u tehničkim i pedagoškim pitanjima učenja pomoću računara.

Da bi ovo obrazovanje bilo uspešno treba pripremiti sredstva za obrazovanje nastavnika, stručnu literaturu, programsku opremu i drugo.

B. Radna grupa za programsku opremu

Izrada obrazovne programske opreme je složen stručni posao koji treba da omogući ostvarivanje ciljeva i zadataka u obrazovanju. Nedostatak programa onemogućava realizaciju uvodjenja računara u škole.

(1) Potrebno je planirati i obezbediti sredstva za izradu i distribuciju obrazovnih programa, kao i za njihov uvoz. U svakoj republici, odnosno pokrajini treba zadužiti ustanovu koja će se starati o razvoju i verifikaciji obrazovnih programa. Treba izraditi metodologiju i standarde za izradu obrazovnih programa. Programi treba da imaju status didaktičkog materijala (udžbenika) i treba da se biraju putem javnog konkursa.

(2) Škole i fakultete treba stalno podsticati da stvaraju ideje za domaće obrazovne programe. Potrebno je pregledati, izabrati i prevesti dobre strane obrazovne programe.

(3) Treba izraditi bazične programe ("školjke") koji bi nastavnicima omogućili da jednostavno pripreme nastavne jedinice na računaru. Treba, takodje, izraditi module (potprograme) za standardne elemente obrazovnih programa. Treba pripremiti, izabrati i prilagoditi programsku podršku za školsku administraciju i ostale školske službe.

(4) Treba ustanoviti stručnu saradnju na nivou Jugoslavije i šire. Treba obezbediti informisanje i koordinaciju. Problematika obrazovne programske opreme mora da dobije svoje mesto u računarskim i pedagoškim časopisima. Treba organizovati stručne sastanke o problemima obrazovne programske opreme.

C. Radna grupa za aparaturnu opremu

(1) Aparaturna oprema – definicija

Pojam aparaturne opreme obuhvata opremu pomoću koje je moguće realizovati računarsko radno mesto, kao i dodatnu periferijsku opremu. Računarsko radno mesto obuhvata:

- ako je realizovano kao terminal:
  - tastaturu,
  - zaslon,
  - vezu s centralnim računarom;
- ako je realizovano kao mikroračunar:
  - tastaturu,
  - zaslon,
  - centralnu procesorsku mikroračunarsku jedinicu.

Periferna oprema je:

- štampač,
- digitalizator, miš, skaner, svetlosna olovka,
- modem, oprema za računarsku mrežu,
- AD/DA pretvarač,
- specijalna periferijska oprema za posebne zahteve.

(2) Strategija opremanja

Predškolske organizacije treba opremati odgovarajuće konfigurisanim kućnim računarima.

Sve osnovne i srednje škole treba opremati ličnim računarima. Razlike u potrebama pojedinih škola moguće je realizovati odgovarajućom programskom i periferijskom opremom. U slučaju da je računarsko radno mesto realizovano kao terminal, preporuke koje slede odnose se na minimalne mogućnosti koje su na raspolaganju pojedinom korisniku.

(3) Lični računari

Škole treba opremati IBM PC kompatibilnim računarima.

Prvi lični računar u školi treba da ima sledeće karakteristike:

- 640 KB RAM,
- ugradjen matematički koprocesor,
- Hercules ili CGA kompatibilnu grafiku (ako je zaslon crno–beli),
- EGA-Hercules-CGA kompatibilnu grafiku (ako je zaslon u boji),
- disketnu jedinicu 1 × 360 kB,
- tvrdi disk 1 × 30 MB
- serijko sučelje (interfejs) – RS 232 C,
- paralelno sučelje (interfejs) – Centroniks kompatibilno,
- Microsoft ili Mouse System kompatibilni miš.

Ostali lični računari treba da imaju bar sledeće karakteristike:

- 256 kB,
- disketnu jedinicu 1 x 360 kB,
- Hercules ili CGA kompatibilnu grafiku (ako je zaslon crno–beli).

Lični računar i na zaslonu i na tastaturi mora da ima sve YU-karaktere (uključujući i č,ć,š,ž). Mora postojati mogućnost biranja izmedju YU i US ASCII karakterskog skupa.

Lični računari moraju da budu tako koncipirani da ih je moguće proširivati. Proširenja zavise od specifičnih potreba pojedinih škola.

(4) Štampači

Štampač mora da bude kompatibilan sa ličnim računarom. Minimalne karakteristike su:

- 80 kolona,
- grafički način rada ("bit image", definisanje sopstvenih karaktera),
- traktorski papir,
- štampanje YU i/ili US ASCII karaktera.

(5) Lokalna mreža

Lokalna mreža mora da omogućava:

- komunikaciju izmedju servera i satelita,
- nesmetan rad servera, kada mreža radi,
- centralno učitavanje programa,
- korišćenje periferijske opreme servera od strane satelita.

(6) Komunikacijska oprema

Komunikacijska oprema mora da omogući školama pristup javnim bazama podataka, povezivanje izmedju škola i sl. Modem mora da ima brzinu od najmanje 300 Bauda.

(7) Ostala periferijska oprema

U zavisnosti od zahteva vaspitno-obrazovnog programa, potrebno je obezbediti i dodatnu opremu, kao što su crtači, digitalizatori i sl. Posebnu pažnju treba posvetiti video-izlazu iz video sučelja.

(8) Kućni računari

Ubuduće škole treba opremati pre svega ličnim računarima. Postojeće kućne računare u školama treba povezati sa ličnim računarima pomoću računarskih mreža. Kućni računari mogu pomoću odgovarajuće periferijske opreme (pretvarači, sučelja) postati pomoćna sredstva pri merenju, simulaciji, regulaciji i upravljanju.

Pri eventualnim nabavkama novih kućnih računara za navedene potrebe, ne bi trebalo širiti broj različtih tipova kućnih računara, osim ako njihove karakteristike nisu znatno bolje.

(9) Instalacijska infrasturktura

Instalacijska infrastruktura mora u prostoru u kome se računari koriste da obezbedi:

- priključak na gradsku mrežu,
- telefonski priljučak,
- priključak za lokalnu mrežu (ukoliko ona u školi postoji),
- priključak za video uredjaj (TV zaslon, itd.).

(10) Minimalni nivo opremanja

Treba obezbediti da u SFRJ ne bude više ni jedne škole bez računara. U svakoj školi treba obezbediti bar:

- lični računar,
- štampač,
- modem.

(11) Ergonomija

Pri rasporedjivanju radnih mesta treba voditi računa da položaj zaslona i njegova okolina budu takvi da ne dolazi do refleksije na zaslonu. Radna mesta treba primereno osvetliti.

Gabarite radnog mesta treba prilagoditi ergonomskim zahtevima uzrasta koji opremu pretežno koristi.

Pri rasporedjivanju radnih mesta treba voditi računa da bude omogućena komunikacija kako izmedju nastavnika i učenika tako i izmedju samih učenika.

(12) Organizacija rada

U svakoj školi, uvodjenjem računara i obrazovne tehnologije uopšte u vaspitanje i obrazovanje, treba da rukovodi nastavnik – savetnik.

Rad s računarima treba u školama organizovati tako da se računari i računarske učionice koriste tokom celog dana.

D. Radna grupa za naučno-istraživačku delatnost

(1) Naučno-istraživačka delatnost na području upotreba računara u vaspitno-obrazovnom radu treba da pruži osnovu za prihvatanje odluka. Investicije treba da budu naučno zasnovane i provedene tako da se obezbede optimalni koraci razvoja na ovom području.

(2) Istraživačke organizacije, zavodi za školstvo, instituti pri radnim organizacijama i matični fakulteti moraju da organizuju istraživačku delatnost na području uvodjena računara u obrazovanje. One moraju da definišu probleme, razrade zadatke i organizuju istraživače. Istraživanja treba da obuhvate razne, a naročito složenije strategije (simulaciju, rešavanje problema i sl.). Treba se koncentrisati na ona područja obrazovanja koja se bez računara ne mogu izvoditi ili ih je moguće izvoditi samo na vrlo niskom nivou.

Mogućnosti programske i mašinske opreme treba maksimalno iskoristiti. Osnovna polazna tačka istraživanja treba da bude: računari moraju da pomažu učenicima pri učenju a nastavnicima pri podučavanju. Nastavnike treba rasteretiti od administrativnog posla.

Treba podsticati istraživanja razvoja kreiranja znanja kod učenika i upotrebe računara u vaspitno-obrazovnom procesu.

(3) Istraživačke organizacije i matični fakulteti iz oblasti vaspitanja i obrazovanja treba da dobiju dovoljno računarske opreme i finansijska sredstva za istraživačku delatnost u vaspitno-obrazovnom radu u toj oblasti. Potrebna je, takodje, adekvatna motivacija, koja se može ostvariti, na primer, uvo-

djenjem boniteta u opremi, finansiranju i slično za posebne uspehe.

(4) Istraživačka delatnost u oblasti informatike treba da proučava uvodjenje i upotrebu računara u osnovnoj školi, srednjoj školi i visokom školstvu. Treba istražiti kada, kako i šta od računarstva i informatike treba uvoditi u obrazovanje.

(5) Treba organizovati koordinaciju, razmenu informacija i operativnu saradnju u jugoslovenskim razmerama uz adekvatno korišćenje inostranih iskustava a, takodje, organizovati više operativnih sastanaka na tu temu.

E. Radna grupa za finansiranje

Predlažu se sledeća opredeljenja za finansiranje uvodjenja informacione tehnologije u škole:

U uslovima uvodjenja informacione tehnologije u obrazovanje, neophodno je dugoročno i srednjoročno planiranje finansijskih sredstava i obezbedjivanje realizacije tekućom politikom.

Sredstva za finansiranje bi trebalo formirati na nivou republičkih i pokrajinskih samoupravnih interesnih zajednica.

Sredstva treba formirati po vrstama namene i to: obrazovanje nastavnika, proizvodnja i nabavka programske opreme, nabavka aparaturne opreme i naučno-istraživački rad.

(1) Sa ukupnim i uskladjenim finansiranjem potrebno je svake godine izraditi jedinstvene projekte opremanja škola, u okviru kojih bi bilo moguće voditi politiku — čime i kako se škole opremaju.

Škole koje primaju opremu moraju ispuniti sledeće uslove:

- sopstveno finasijsko učešće pri nabavci opreme,
- prostor i nastavnike za upotrebu opreme,
- optimalno korišćenje opreme i
- permanentno obrazovanje nastavnika za upotrebu opreme.

Dobavljač opreme mora ipuniti sledeće uslove:

- kompletnost opreme (aparaturna oprema, programska: sistemska, aplikativna i obrazovna),
- povoljna cena i rok isporuke,
- servisiranje i isporuka rezervnih i dodatnih delova i
- obrazovanje nastavnika za upotrebu isporučene opreme.

(2) Zajedničkim sredstvima treba finansirati područja po sledećim delovima:

		u početku opremanja	kasnije, kad su škole opemljene
1.	obrazovanje nastavnika	20%	40%
2.	programska oprema	20%	30%
3.	aparaturna oprema	40%	20%
4.	naučno--istraživački rad	20%	10%

(3) Treba finasirati opremanje nastavničkih fakulteta, koji permanentno obrazuju nastavnike za upotrebu računara u vaspitno-obrazovnom radu.

Adrese autora:

Duško Vitas
Računarska laboratorija
Prirodno-matematički fakultet
Beograd
Studentski trg 16

Dragan Vasić
Republički zavod za unapredjivanje vaspitanja i obrazovanja SR Srbije
Beograd
Kneza Miloša 101

Prikaz medjunarodnog simpozijuma "Kompjuter na sveučilištu"

Veljko Spasić

Medjunarodni simpozijum "Kompjuter na sveučilištu" ("Computer at the University") održava se svake godine u Jugoslaviji. Pokrovitelj je Jugoslovenska akademija nauke i umetnosti, a organizator računski centar Zagrebačkog univerziteta u različitim oblastima nauke.

Simpozijum je medjunarodni, održava se devet godina, a na poslednjem devetom (Cavtat 18–22. 5. 1987) bilo je 146 radova. 83 rada prezentirano je na engleskom jeziku, a 63 na jezicima jugoslovenskih naroda i narodnosti. Radove je pripremilo 238 autora i koautora iz 17 zemalja. Prethodno, 1986. godine simpozijum je imao približno jednaku strukturu, broj autora i zemalja.

Na ovogodišnjem simpozijumu izlagani su radovi iz sledećih oblasti:

- informatika i obrazovanje;
- računarski sistemi i mreže, personalni računari;
- softversko inženjerstvo;
- informacioni sistemi i baze podataka;
- analiza podataka, statistika i statistički softver;
- modeliranje, simulacija i optimizacija;
- dizajn i proizvodnja pomoću računara;
- veštačka inteligencija i ekspertni sistemi;
- matematički aspekti informatike.

Zbornik radova je izdat u vidu publikacije obima približno 500 stranica koja objedinjuje radove i saopštenja.

Simpozijum ocenjujemo kao veoma uspeo, sa raznovrsnim i sadržajnim radovima i konstruktivnom atmosferom što je izmedju ostalog omogućeno i dobrom organizacijom.

doktorske i magistarske teze

Fakultet: Prirodno-matematički fakultet, Institut za matematiku, Beograd
Doktorska teza: Dokazivanje teorema izvodjenjem uz pomoć računara.
Kandidat: mr Irena Pevac, dipl.mat.
Komisija: dr Marica Prešić, van.prof. PMF, Beograd
dr Nedeljko Parezanović, red.prof. PMF, Beograd
dr Dragoš Cvetković, red.prof. ETF, Beograd
Datum odbrane: 6. 07. 1987.

Iz referata:

... Teorijski doprinos teze je u kreiranju strategije vodjenja dokaza na nivou kvantifikatorske formule prilagodjene uvedenoj formalizaciji teorije grafova. Preuzeta je globalna ideja razbijanja na potciljeve koji se dokazuju umesto polaznog cilja, korišćenje raznih pojednostavljenja i delimične kanonizacije, upotreba lema u dokazivanju potciljeva i strategija rada unazad. Najznačajniji teorijski doprinos predstavljaju sledeći delovi inteligentne strategije vodjenja dokaza:
1. Heuristika za odabiranje relevantne definicije ili leme. Njome se vrši izbor definicije, odnosno leme, pomoću koje će se izvršiti netrivijalna transformacija tekućeg potcilja. Zasniva se na pretpostavci da u teoremi oblika $H_1 \wedge H_2 \wedge .. \wedge H_n \Rightarrow C$ najčešće ne koristimo sve pretpostavke za izvodjenje zaključka. Značaj i originalnost heuristike za izvodjenje zaključaka. Značaj i originalnost heuristike ogleda se u činjenici da se izbor relevantne definicije, odnosno leme, ne vrši samo na osnovu formule tekućeg potcilja već na osnovu globalnog znanja o strukturi cele formalne teorije u kojoj se izvodi dokaz.
2. Heuristika za redukovanje broja kvantifikatora. Na ovaj način se izostavljaju kvantifikatori koji ekvivalentnim logičkim transformacijama mogu da postanu univerzalni, a da im domen dejstva bude cela formula. Njigovo brisanje se izvodi bez transformisanja potcilja. S druge strane prikazuje se postupak za redukovanje broja kvantifikatora sa prelaskom na logički jače tvrdjenje. Značaj heuristike je u mogućnosti rešavanja nekih problema na nivou unifikacije u automatskom radu.
3. Heuristika za manipulaciju sa n-arnim konjukcijama i disjunkcijama. Ovom heuristikom se predlaže strategija za utvrdjivanje redosleda konjunkta, odnosno disjunkta, u slučaju višestrukog pojavljivanja istog predikata, tako da termi na nivou argumenata čine odredjenu strukturu.

Praktični doprinos teze sastoji se u implementaciji interaktivnog dokazivača koja je vodjena pod rukovodstvom i u saradnji sa prof. D.Cvetkovićem. Sistem izvodi dokaz potpuno samostalno ili uz veću ili manju pomoć korisnika u zavisnosti od izabranog nivoa interaktivnosti. Izvodjenja su prirodna u smislu da se imitiraju metodi i koraci koje matematičari primenjuju u dokazivanju teorema...

Fakultet: Prirodno-matematički fakultet, Institut za matematiku, Beograd
Magistarska teza: Prilog analizi performansi lokalne mreže računara (tipa ETHERNET)
Kandidat: Dragan Stanojević, dipl.mat.
Komisija: dr Radivoj Petrović, red.prof. ETF, Beograd
dr Nedeljko Parezanović, red.prof. PMF, Beograd
dr Dušan Tošić, docent, PMF, Beograd
Datum odbrane: 19. 06. 1987.

Iz referata:

... glavu "Bas mreže" autor posvećuje mrežama čija je topologija bas tipa, a korisnici pristupaju medijumu u slučajnim vremenima. Posledica toga je pojava kolizija, što će biti glavni predmet ove magistrature. U sledećoj glavi "ETHERNET", autor detaljno opisuje jedan mehanizam pristupa medijumu koji su zajednički razradile i predložile tri poznate američke firme: Digital, Intel i Xerox. Tvorac ovog mehanizma R.Metcalfe (1976) predložio je da pauza izmedju slanja dva uzastopna paketa bude slučajna veličina čija je vrednost u funkciji broja uzastopnih kolizija. Autor ovog rada pozabavio se pitanjem dobrog izbora te funkcije, i njenog uticaja na performanse mreže i iskorišćenost kanala, kašnjenje i broj kolizija. Izabran je simulacioni pristup kao način rešavanja ovog problema. Bilo je potrebno razviti kompletan (razumno uprošćen) model komunikacionog mehanizma tipa ETHERNET i korespondentni računarski program. Odabran je i korišćen FORTRAN jezik. Pretpostavljeno je da su: (a) zahtevi za komuniciranjem slučajni, sa vremenima izmedju dva zahteva eksponencijalno rasporedjenim, (b) dužine paketa su uniformno raspodeljene. Simulacija je izvedena za različite saobraćajne zahteve i različite brojeve uključenih stanica u mrežu.

Dobijeni su interesantni dijagrami iskorišćenja kanala, broja kolizija, kašnjenja u funkciji intenziteta saobraćaja. Kandidat je izveo valjane zaključke. Najpre, ne postoji univerzalno dobar (ili najbolji) back-off algoritam. Svi testirani algoritmi imaju dobre osobine CSMA/CD protokola u smislu iskorišćenja kanala, stabilnosti pri velikom saobraćaju, kao i ravnomeran odnos prema stanicama u pristupu kanalu. Vredan je zaključak da su stepeni i eksponencijalni algoritmi dali najbolje iskorišćenje kanala pri vrlo velikom saobraćajnom opterećenju...

Da li je kompjuter računar?

Duško Vitas

1. Šta je terminologija?

U svakoj delatnosti postoji izvestan broj reči čije je značenje precizno definisano. Postoji, na primer, rečnik koji opisuje jezik ribara: alaske alate, ribe, karakteristične situacije za ovaj zanat. Skup svih takvih reči, koje nazivamo terminima, čini terminologiju delatnosti u kojoj je nastao.

Neke terminologije su vrlo stare i stabilne: u njima se retko javlja potreba za novim terminima. Druge su, pak, u nastajanju i potrebno je da prodje mnogo vremena pre nego što pojedina reč bude prihvaćena i prepoznata kao termin. Jedna takva mlada terminologija je i terminologija našeg računarstva.

Na srpskohrvatskom govornom području se, na primer, čuje i računar i kompjuter i računalo i kompjutor ali, mada retko, i elektronski mozak, ordinator ili, prosto, komputer... Sve ove reči označavaju isti pojam kome na engleskom odgovara termin *computer* ili na francuskom *l'ordinateur*. Ova pojava, u kojoj jednom pojmu odgovara više različitih reči, naziva se sinonimija: dve reči su sinonimne ako se u svakom kontekstu jedna može zameniti drugom. Tako bi, na primer, reči *računar* i *kompjuter* mogle biti sinonimi. Ali, ako se njihova upotreba analizira u jeziku dnevnih listova, onda se može primetiti da dopisnici iz inostranstva radije koriste *kompjuter* dok se njihove domaće kolege češće opredeljuju za *računar*. Otuda bi se vremenom mogla razviti razlika u značenju ovih dveju reči: *kompjuter* bi mogao označavati, možda, *inostrani računar*.

Jedno takvo razlikovanje nekada sinonimičnih termina može se primetiti u upotrebi reči *računarstvo* i *informatika* (videti, na primer, belešku u 1. glavi knjige "Računarstvo sa programskim jezikom paskal" Vlade Rajkoviča i Ivana Bratka, Nolit, 1986). Sredinom sedamdestih godina kod nas je u upotrebi bio termin *računarske nauke* kao ekvivalent engleskom *computer science* dok se pod terminom *računarstvo* podrazumevala ukupnost aktivnosti na obradi podataka (čemu bi, u engleskom odgovaralo, na primer, *data processing* ili *computing*). Vremenom je termin *računarske nauke* zamenjen terminom *računarstvo* dok je ovaj deo svog prvobitnog značenja ustupio terminu *informatika*. Reč *informatika*, pak, dolazi iz francuskog gde ju je još 1962. iskovao Filip Drajfus od reči *informacija* i *automatika* kao zamenu za englesko *data processing* [3]. Tako se u našem jeziku odomaćila sintagma *računarstvo i informatika* koju bismo sa mukom preveli nazad, na engleski ili francuski.

Primer postupne evolucije termina iz oblika nastalog pod uticajem engleskog jezika u "domaći" ekvivalent se može sagledati na sledećoj transformaciji. Godine 1977. u Sarajevu je održan naučni skup pod nazivom "Kompjuterska obrada lingvističkih podataka". Drugi skup sa ovim sadržajem je održan na Bledu 1982. pod naslovom "Računarska obrada lingvističkih podataka", dok je treći, iz 1985, nosio naziv "Računarska obrada jezičkih podataka". Bilo je potrebno, dakle, skoro deset godina da se kompjuter zameni sa računar, a lingvistički, u ovom kontekstu, sa jezički.

2. Kako nastaju termini?

Kao što se može videti iz prethodnih primera, put kojim nastaje termin može biti vrlo složen rezultat raznih, često neobičnih, uticaja i okolnosti. Za osnovni zahtev kome jedan termin mora da udovolji, a to je da je njime moguće jasno i precizno označiti odgovarajući pojam — nije nimalo lako pronaći pravo rešenje. Posebno, u situacijama kada se termin preuzima pod uticajem drugih jezika, mogu se lako pojaviti konfuzna i nedosledna terminološka rešenja koja sprečavaju da se poruka iz izvornog jezika nedvosmisleno prenese u ciljni jezik. Posmatrajmo, primera radi, objašnjenje kojim se u Oksfordskom rečniku računarstva [1] definiše jedno od značenja reči *string*:

> *String*: any one-dimensional array of characters...

Koristeći se rečnikom koji je nedavno objavljen u Računarima [6], ovu rečenicu bismo preveli sa:

> *Niz*: ma koji jednodimenzionalni niz znakova...

Očigledno, prevod je lišen značenja jer ono čime se definiše se poziva na ono što se definiše. S druge strane, u matematičkoj terminologiji pronalazimo da se za *string* koristi reč *niska* pa možemo homonimni *niz* rastaviti na dva termina *niz* (kao ekvivalent za *array*) i *niska* (za *string*). Pogledajmo, dalje, kako je u rečniku Računara definisan termin *program*. Nalazimo da je "program niz povezanih naredbi...". Ali, program nije ni *string* niti *array* već, ponekad, *sequence* naredbi. Reči *sequence* u istom rečniku odgovara *sekvenca* koja je još sinonim sa rečju *order*. No, to nije kraj: u prvom primeru se koristi još i reč *znak* kojoj u engleskom odgovaraju *character* i *sign* a u drugom, reč *naredba* koja je u engleskom, prema ovom rečniku, ili *instruction* ili *statement*. Sličnu konfuziju (u obrnutom smeru) nalazimo i u glosaru "Računalniška aBCDa" [7], koji je kompilacija iz [5]. Tu je *command ukaz*, *instrukcija*, *logički operator*, *ipmuls* (?), dok se za *instruction* i *statement* ne daju ekvivalenti. Broj mogućih interpretacija navedenih primera na engleskom jeziku se tako višestruko umnožava, pa se ne može sa sigurnošću tvrditi šta bi oni zapravo sve mogli da znače.

Odavde sledi da se terminologija mora brižljivo i precizno razvijati kako bi se načinio adekvatan terminološki sistem unutar jedne oblasti. U našem računarstvu se može uočiti nekoliko osnovnih pristupa izgradnji termina od kojih svaki ima svoje vatrene pobornike i svoje ozbiljne nedostatke ali koji se u praksi obično primenjuju svi zajedno.

3. Metoda malodušnih ili anglo-srpskohrvatski

Najmalodušniji medju nama se, po pravilu, radije predaju pred problemima koje postavlja izgradnja terminologije i koriste se engleskim terminima transkribovanim prema Vukovom principu "piši kao što govoriš...". Tako su nastali *softver*, *hardver*, *firmver*, *midlver*, *interfejs*, *kartridž*, *bag*, *butstrap*, *listing*, *damp*,... Kada se jednom pojavi u ulozi termina u našem jeziku, na ovakvu reč se, po pravilu, primenjuju sva raspoloživa morfološka sredstva srpskohrvatskog pa tako nastaju reči

- od *softver* — pridev softverski, zatim softveraš,
- od *damp* — dampovanje, izdampovati,
- od *debug* — debagovati, debagiranje i debagovanje pored debaginga,
- od *submit* — subnuti ili sabmitovati....

Nedostaci ovog rešenja su višestruki. Pre svega, semantička motivacija koja u engleskom jeziku stoji iza ovih termina se sasvim gubi. Na primer, *bag* (od engl. *bug*) je prosto *buba*, *mušica*, dakle, nešto što, govoreći slikovito, veoma dosadjuje i što treba istrebiti. *Butstrep* (od engl. *bootstrap*) potiče iz priče o baronu Minhauzenu koji se izvlači iz blata potežući gajke na svojim čizmama.

Jedan od centralnih pojmova u računarstvu, *softver*, je nastao kao jezička igra polazeći od reči *hardver* koja, inače, ima svoje značenje i u običnom, svakodnevnom jeziku. Opozicija *hard* (=tvrdo)/ *soft* (=meko), koja je u korenu ove igre, kasnije je poslužila kao inspiracija za stvaranje čitave klase novih reči po istom principu: *firmware*, *midleware*, *brainware*, *bridgeware*, *bookware*, *freeware*, itd. Doslovno prenošenje čitave ove klase reči u naš jezik uvodi potencijalno na stotine različitih -*verova* koji, za razliku od engleskog, u našem jeziku nemaju nikakvu semantičku pozadinu. Na primer, saglasno objašnjenju iz Oksfordkog rečnika, *midlver* duguje svoje ime tome što je "proizvod koji se nalazi izmedju softvera i hardvera". Pri tome, postaje sasvim nerešiv problem neologizma kakav je francusko *didacticiel*, iskovanog po uzoru na francusko rešenje za *softver*: *logiciel*, itd. *Didacticiel* je pojam vezan za uvodjenje računara u obrazovanje. U srpskohrvatskom ga možemo preuzeti ili kao *didakticijal* ili, da stvar bude još gora, kao *didaktover* (kako bi se očuvala srodnost sa klasom -*verova*).

Drugi ozbiljan nedostatak ove metode je što većina ovakvih reči sadrži u sebi konsonantske grupe koje je ili vrlo teško ili, čak, nemoguće izgovoriti kao što su -*ftv*-, -*rdv*-, -*mv*-, -*js*, -*tstr*-, -*mpj*-,... Ova činjenica krije u sebi opasnost da se inicijalno transkribovani anglicizam vremenom veoma udalji od podražavanja svog engleskog originala (kao što je to, na primer, slučaj sa fudbalskom terminologijom) čime se gubi jedina prednost ovakvog pristupa.

4. Metoda neopreznih ili "to je otprilike isto"

Manje malodušni u računarskom svetu nastoje da pronadju srpskohrvatski ekvivalent koristeći se relacijom "otprilike isto". U već navedenim primerima, imamo da je *string* otprilike isto što i *array* pa kako je *array* niz, to će i *string* otprilike biti *niz*. Pri

tome, razlikovanje koje postoji u engleskom jeziku nije slučajno, pa se ono mora očuvati i u našem jeziku. Pri tome, ova metoda nosi u sebi izuzetne opasnasti jer vodi ka postupnom poistovećivanju sasvim različitih fenomena. Pogledajmo još jedan primer iz rečnika Računara. Reč *greška* je istovremeno prevod za engleske reči *bug*, *error*, *failure* i *fault*. *Bug* je, doduše, *programska greška* ali se javlja i kao *greška* u okviru termina *program za otklanjanje grešaka*. Možemo pretpostaviti da je *greška* zamena zapravo samo za *error*. *Bug* se može povezati sa pojmom greške ali vrlo zaobilaznim putem. Naime, *bug* u radu, na primer, jednog programa predstavlja odstupanje onoga što program zaista radi od onoga što bismo mi želeli da on radi. U tom smislu treba praviti razliku izmedju greške koja je odstupanje od interne konvencije (kakva je npr. sintaksička greška) i odstupanja od eksterne specifikacije programa. Program koji ima *bug* je zapravo tačan prevod onoga što smo mi zaista u njega ugradili ali odstupa od onoga što smo želeli da u njega ugradimo. Slično, *failure* i *fault*, koji su gotovo sinonimi, označavaju radije *otkaz* sistema negoli njegovu grešku.

Ovoj metodi pripada i postupak kombinovanja ovog prilaza sa prethodnim (uporediti, na primer, u rečniku Računara, *driver* je *drajver* ali je *drive* = *pogon*!). Ovom tehnikom se može doći i do razjednačavanja značenja engleskog termina (npr. engl. *operator* može biti i *operator* i *operater*).

Rešenja se ponekad traže i tako što se zamena za "neprevodivi" engleski, traži u drugim evropskim jezicima. Tako je, na primer, iz nemačkog preuzeta reč *datoteka* kao zamena za englesko *file*. Ovaj termin, kovanica od latinskog *datum* — *podatak* i grčkog sufiksa *-teka* (koji inače koristimo u rečima kao što su *biblio-teka*, *disko-teka*, pa čak i *vico-teka* ili *smeho-teka*, itd.) je dugo vremena besprekorno funkcionisao. Svoje nedostatke je počeo da pokazuje prilikom prevodjenja izraza kao što su *program file* ili *data file*. *Program file* je *file* koji sadrži programe a ne podatke, dok se *data file* ponaša pleonastično u izrazu *podatkovna datoteka*. U rečniku Računara nailazimo na još jedan primer. Englesko *office automation* je prevedeno kao *automatizacija biroa*. *Biro* je reč francuskog porekla koja u francuskom označava *kancelariju* baš kao što je to slučaj sa rečju *office* u engleskom. Kod nas se reč *biro* javlja u izrazima "biro za nadjene stvari", "turistički biro", "projektantski biro", ali se teško može reći da je "činovničko radno mesto u birou". Pri tome, u francuskom se za *office automation* koristi neologizam *bureautique* (npr. *birotika*, koje je prihvaćeno i u engleskom kao *burotics* [4]), te je predloženo rešenje iz Računara zapravo nedosledna kompilacija ova dva izvora.

5. Metoda opreznih ili definiciona literatura

Jedan od mogućih prilaza se sastoji u izbegavanju problema. Autor teksta koji primenjuje ovaj postupak strogo vodi računa o tome da "nezgodni" termin zameni najčešće delom a ponekada i čitavom njegovom definicijom. Ovaj postupak je mogao naći svoju potvrdu, na primer, u našoj zakonodavnoj terminologiji gde pronalazimo primere termina koji u sebi sadrže petnaest i više reči (npr. čuveni termin "radni čovek koji samostalno obavlja delatnost ličnim radom sredstvima rada u svojini gradjana i drugih fizičkih ili pravnih lica..." se u ZUR-u koristi kao zamena za *zanatliju*).

Ovaj postupak podseća na jednu zabavnu igru koju je svojevremeno predložio francuski pisac (i matematičar) Remon Keno, poznat našoj publici po svojim "Stilskim vežbama" i "Caci u metrou". Igra, nazvana definiciona literatura, se sastoji u sledećem: polazeći od jedne proste rečenice, igrač iterativno zamenjuje svaku značeću reč odgovarajućom rečničkom definicijom. Posle samo nekoliko koraka dobija se tekst zavidne dužine čije je osnovno značenje vrlo blisko značenju polazne rečenice ali koje se, ako polazna rečenica nije poznata, ne može dešifrovati.

U računarskoj terminologiji, blagi primeri ovog postupka su upotrebe izraza *programska oprema* kao zamene za *softver* ili *program za otklanjanje (programskih) grešaka* kao zamene za *debuger*. U glosaru [7] nailazimo, pored *hardvera* i na *fizičnu konfiguraciju sistema*, *fizične enote računalniškega sistema*, *fizični del računalniškega sistema*. U srpskohrvatskoj verziji, *hardveru* je ekvivalentno *fizička konfiguracija sistema*, *fizičke jedinice kompjuterskog sistema*, *fizički deo kompjuterskog sistema*. Da bi nedoslednost bila potpuna, *software* je samo *softver*; niz programa, postupaka, pravila i odgovarajuće dokumentacije (obezbedjuje ih proizvodjač ili ih formira korisnik) (!?).

Glavni nedostatak ovakvog pristupa je što je tekst, proizveden na ovaj način, iako obično korektan, redundantan i teško razumljiv. Štaviše, jasno je da su naše kolege sa anglo-saksonskog govornog područja mogle i same da upotrebe svaku od ovih definicija kao odgovarajući termin: na primer, *physical units of a computer system*. Stoga se u svakodnevnom informatičkom govoru ovakva rešenja ne mogu dosledno primeniti pa se ovom metodom im-

plicitno forsiraju dva gore opisana postupka.

6. Besmislice ili "kako vam drago"

Naročitu metodu ponekad koriste autori rečnika iz računarstva. Reč je o zameni engleskog termina sasvim proizvoljnom našom rečju na koju se naišlo u nekom opštem englesko-srpskohrvatskom (ili drugom) rečniku. Na primer, u Rečniku V. Tasić [5], nalazimo da je *hash*, jedna od osnovnih metoda pretraživanja, na našem jeziku *besmislica*! (Drugo navedeno značenje, kontrolna suma, takodje, nije od koristi.) Sada, ako bismo pokušali da saznamo iz Oksfordskog rečnika šta je *hashing*, dobili bismo "koristan" savet da je to organizovanje tablice radi bržeg pretraživanja besmislica! U jednom drugom, starijem rečniku ćemo naći da je *push-down store* posuvraćena gomila [2].

Ovakva proizvoljna rešenja se nalaze i u autorskim tekstovima. Na primer, u poslednjem broju Računara, na strani 6, možemo videti da je TEX D. Knuta editor, mada je možda jedina mana ovog sjajnog programa što on to nikako nije (TEX je formater). U broju 9 slovenačkog časopisa BIT (koji više ne izlazi), se kao prevod za *interface* može pronaći *prirezovalnik papirja*!

Veselih primera ove vrste ima možda ponajviše. Očigledna posledica je da se sasvim jasna, nedvosmislena rečenica engleskog jezika prenosi u naš jezik kao puka dezinformacija. Opasnost je očigledna a moguće štete je teško utvrditi i proceniti.

7. Postoji li rešenje?

Prilikom prenošenja termina iz stranog u naš jezik mogu se koristiti izvesne olakšice: termini sa grčkim ili latinskim korenom se preuzimaju, na primer, iz engleskog kao da ih preuzimamo iz grčkog ili latinskog. Na primer, *compiler* nije *kompajler* već *kompilator*. Takvim postupkom se mogu rešiti neki naizgled nerešivi problemi. Englesko *bus*, koje se često prevodi sa *bas* ili *magistrala* (jer je sinonim sa *highway*) ili sa *sabirnica*, je zapravo enklitički oblik dativa latinske zamenice *omnis*: *omnibus*, te se može preuzeti kao *bus*. Zanimljiv primer valjane primene ovog pravila nalazimo u rečniku Računara: kad god se autor držao ovog pravila, dobijao je kompaktno i dosledno rešenje. Naprotiv, svako odstupanje vodi u zamršenu igru mogućih značenja na već izloženi način. Na primer, nalazimo da je englesko *permanent storage* u rečniku Računara *neizbrisiva memorija* mada je jasno da *permanentan* i *neizbrisiv* nisu sinonimi. Štaviše, termin *nonerasable memory* ostaje bez svog adekvatnog para u našem jeziku.

U traženju podesnog rešenja za "neprevodivi" engleski termin, moguće je ponekad koristiti reči koje su skovali drugi narodi. U nekim zemljama, kao što je Francuska, problemu računarske terminologije se pridaje izuzetan značaj. Tako su pojedini francuski neologizmi našli mesta i u engleskoj računarskoj terminologiji. Primer je *telematique* (=*telematika*). Primernu brigu o svojoj računarskoj terminologiji vode i naše slovenačke kolege: medju brojnim, vrlo uspelim rešenjima, tu se nalaze i retko korišćene reči *meščina* i *trdnina* kao zamene za *softver* i *hardver*. Medju frekventnim neologizmima nalazimo: *vmesnik* za *interface*, *povezovalnik* za *linker*, *pomnilnik* za *memory*, *medpomnilnik* ili *izravnalnik* za *buffer*, *posnemovalnik* za *emulator*, *tolmač* za *interpreter*, itd.

Najzad, izuzetno retko, pojavljuju se uspešne "domaće" zamene engleskih termina. Ovakva rešenja imaju više izvora: ponekad se radi o oživljavanju starih, već zaboravljenih reči, ponekad o novom značenju uobičajene reči, a katkad o srećno iskovanom neologizmu. Tako, na primer, već zaboravljena reč *zaslon*, koju beleži Matičin rečnik, je sasvim podesna zamena za *display*, a za njeno oživljavanje zasluga pripada slovenačkim kolegama. Reč *stog*, kako predlaže Turkov rečnik, je, pak, uspešna zamena za "neprevodivo" *stack*. Na sličan način su, bez sumnje, nastali *računar*, *štampač*, *čitač*, *bušilica*, *bušač*,... u samu zoru naše informatike. U novije vreme pojavile su se vrlo uspešne zamene, na primer, za *push-down automaton* — *potisni automat*, *interface* — *sumedja*, *sučelje* ili *uskladnik*, ili značajno razlikovanje izmedju *memory*, *store* i *storage* pomoću reči *memorija*, *skladište* i *spremište* (u knjizi Fraja: Računari za početnike, Nolit, 1985). Na žalost, ovakva rešenja su izuzetno retka: ona zahtevaju ne samo odgovarajuća znanja o računaru i jeziku već i mnogo mašte i hrabrosti. S druge strane, nameće se kao neophodno iznalaženje upravo ovakvih rešenja: računari ulaze u svakodnevni život i žargon profesije prestaje da biva prikladan način za govor o njima. Štaviše, ako se o računarima može učiti već u osnovnoj školi, onda je prirodno da takva nastava bude na maternjem jeziku.

Zapitajmo se na kraju da li je neophodno ulagati truda u jednu beskrajnu i, u izvesnoj meri, već izgubljenu trku. Moj je utisak da je osnovno opravdanje za takav rad u činjenici da se prenos novih tehnologija odvija prvenstveno kroz prenos znanja a ne kroz prenos samih mašina. Pri tome,

kao što se videlo na primerima, prenos znanja o računarima nije moguć jezikom koji je mutan i konfuzan. Čini mi se, stoga, da je neophodno pronaći pre svega forme sistematskog rada na prikupljanju i praćenju računarske terminologije, kao što to, uostalom, celi ostali svet već odavno čini.

8. Zahvalnost

Najveći deo ovog izlaganja je rezultat dugotrajnih i plodotvornih diskusija koje je autor imao sa prof. Nedeljkom Parezanovićem i Slobodanom Djordjevićem, urednikom Nolit-a, na čemu im se i u ovoj prilici osobito zahvaljujem.

Literatura

1. Illingworth, V. (ed.): Dictionary of computing, Oxford Univ. Press, 2. ed, Oxford, 1985.
2. Kontić, M.: Englesko-srpskohrvatski rečnik stručnih termina iz oblasti informatike sa glosarom, Intertrade, Ljubljana, 1974.
3. Morvan, P. (ed.): Dictionnaire de l'informatique, Larousse, Paris, 1981.
4. Naffah, N. (ed.): Proc. of symp. on burotics, North-Holland, 1980.
5. Tasić, V.: Rečnik računarskih termina (engleskosrpskohrvatski), NIRO Tehnička knjiga i Zavod za udžbenike i nastavna sredstva, Beograd, 1986.
6. Tošić, Ž.: Mali leksikon računarstva i informatike, Računari 30, BIGZ, Beograd, septembar 1987.
7. ***: "Računarska aBCDa", srpskohrvatska i slovenačka verzija: "Moj mikro", ČGP Delo, Ljubljana, oktobar 1987. Adrese autora:

Duško Vitas
Računarska laboratorija
Prirodno-matematički fakultet
Beograd
Studentski trg 16

Verifikacija i korektnost

Annotated program
Anotirani program

Tekst programa sa umetnutim tvrdjenjima kao sastavnim delom dokumantacije programa.

Assertion
Tvrdjenje

Bulovski izraz u kome učestvuju promenljive programa, dodeljen odredjenoj tački u programu. Kad god izvršavanje programa stigne u tu tačku, ovaj izraz mora imati vrednost TAČNO za vrednost učestvujućih promenljivih. Na primer, u FORTRAN-u, za niz deklarisan sa DIMENSION A(100), prilikom svakog indeksiranja elementa niza A indeksom i, mora važiti tvrdjenje da je $1 \leq i \leq 100$.

Axiomatic semantics
Aksiomatska semantika

Jedan pristup opisu semantike programskih jezika u kome se značenje pojedinih konstrukata jezika opisuje skupom aksioma.

Bug
Mušica

Uzrok odstupanja rezultata izvršenog programa od zadate ulazno-izlazne specifikacije koji se najčešće može pripisati pogrešci u implementiranju programa.

Cutpoint
Tačka preseka

Proizvoljna, ali fiksna tačka unutar programske petlje (ili modula) kojoj je pridruženo odgovarajuće tvrdjenje (koje se tada naziva invarijanta).

Debugging
Trebljenje

Svaka metoda koja omogućava otkrivanje i korigovanje mušica.

Inductive assertion
Induktivno tvrdjenje

Sinonim za tvrdjenje u tehnici dokaza korektnosti koju je predložio R. Flojd.

Input assertion
Ulazno tvrdjenje

Tvrdjenje, dodeljeno ulaznoj tački u program, koje ima vrednost TAČNO za one vrednosti ulaznih promenljivih koje će se koristiti kao ulazni podaci. Npr. Za program koji izračunava $sqrt(x)$ mora na ulazu važiti $x \geq 0$.

Intermediate assertion
Medjutvrdjenje

Tvrdjenje koje opisuje relaciju koja postoji izmedju radnih promenljivih programa i ulaznih i izlaznih promenljivih programa.

Loop invariant
Invarijanta petlje

Tvrdjenje pridruženo tački preseka koja se nalazi unutar petlje a koje čuva vrednost TAČNO pri svakom prolasku kontrole programa kroz tačku preseka.

Outline of a proof
Skica dokaza

Teskt programa u kome je izmedju svaka dva iskaza umetnuto tvrdjenje i koji, stoga, predstavlja osnovu formalnog dokaza korektnosti programa.

Output assertion
Izlazno tvrdjenje

Tvrdjenje, dodeljeno izlaznoj tački iz programa, koje ima vrednost TAČNO za realizovane vrednosti ulaznih i izlaznih promenljivih. Na primer, za program koji polazeći od zadate vrednosti ulazne promenljive x izračunava celobrojni deo korena iz x, mora na izlazu važiti $x \geq 0 \wedge z = [sqrt(x)]$.

Partial correctness
Parcijalna korektnost

Za zadato ulazno tvrdjenje φ i izlazno tvrdjenje ψ, dokaz parcijalne korektnosti programa P je dokaz teoreme oblika

> Ako važi φ i izračunavanje programa P se završi, onda važi ψ.

Point
Tačka

Fiktivno mesto u zapisu programa na kome se tokom izvršavanje jednog programa, jedan iskaz izvršio a sledeći još nije počeo da se izvršava. U dijagramima toka, tački odgovara luk koji povezuje dve naredbe.

Precondition (of a statement S)
Preduslov (iskaza S)

Tvrdjenje koje ima vrednost TAČNO neposredno pre izvršenja iskaza S.

Program correctness proof
Dokaz korektnosti programa

Formalni dokaz da će dejstvo programa biti u skladu sa korisnički definisanom (ulazno/izlaznom) specifikacijom tog programa izraženom (ulazno/izlaznim) tvrdjenjima.

Program synthesis
Sinteza programa

Postupci konstruisanja programa iz ulazno/izlaznog tvrdjenja koji će, stoga, s obzirom na način konstrukcije, biti totalno korektan u odnosu na zadatu specifikaciju.

Postcondition (of a statement S)
Pauslov (iskaza S)

Tvrdjenje koje ima vrednost TAČNO neposredno po izvršenju iskaza S.

Program specification
Programska specifikacija

Precizan izraz dejstva koje jedan program treba da ostvari bez pozivanja na način kojim to treba da izvrši. Metode specifikacije se kreću u širokom rasponu od neformalizovanih iskaza u prirodnom jeziku do visoko formalizovanih iskaza posredstvom matematičke logike.

Semantics
Semantika

Deo definicije programskog jezika kojim se opisuje dejstvo programskih konstrukata koji su u skladu sa sintaksičkom definicijom jezika.

Termination
Završavanje

Za dato ulazno tvrdjenje φ, dokaz završavanja programa P se svodi na dokaz teoreme:

> Ako važi φ, tada se izračunavanje programa P završava.

Testing
Testiranje

Provera da li program zadovoljava zadatu (ulazno/izlaznu) specifikaciju na osnovu aktuelnog izvršavanja programa. Kako je primetio Dejkstra, "testiranjem se može otkriti prisustvo mušica ali nikada njihovo odsustvo".

Total correctness
Totalna korektnost

Program je totalno korektan u odnosu na ulazno tvrdjenje φ i na izlazno tvrdjenje ψ ako je parcijalno korektan u odnosu na φ i ψ i ako se završava u odnosu na φ.

Verification
Verifikacija

Opšti pojam kojim su obuhvaćeni različiti postupci provere jednog programa u cilju utvrdjivanja njegove saglasnosti sa zadatom (ulazno/izlaznom) specifikacijom programa.

Verifier
Verifikator

Programski sistem koji za anotirani program proverava automatski njegovu korektnost u odnosu na zadatu specifikaciju.

Izbor termina pripremio Duško Vitas

Na Stenfordu je od 10. do 14. marta 1986. godine, pod pokroviteljstvom UNESCO-a, održan simpozijum o ulozi medjunarodnih istraživanja u primeni računara u obrazovanju. Smpozijum je okupio učesnike iz sledećih zemalja: SAD, Meksiko, SSSR, Francuska, Kanada, SR Nemačka, Australija, Švajcarska, Indija, Švedska, Maroko, Japan, Čile, Bugarska, Velika Britanija, Kamerun i Madjarska. Prema izveštaju sa ovog skupa [1], prenosimo

Deklaracija sa simpozijuma u Stenfordu

Istraživači iz 18 zemalja, okupljeni na Univerzitetu u Stenfordu, od 10. do 14. marta 1986. godine na inicijativu Univerziteta i UNESCO-a, da bi ispitali ulogu medjunarodnih istraživanja primene računara u obrazovanju, usvojili su sledeću deklaraciju:

Gradjani moraju biti valjano pripremljeni da žive u jednom društvu u kome će većinom imati dodira sa računarima, bilo u svom profesionalnom životu, bilo u privatnom ili u drugim prilikama. Skrećemo pažnju licima koja donose odluke na njihov zadatak da se efikasno suoče sa novim izazovima što iz toga proizilaze na nacionalnom i na medjunarodnom nivou.

Tokom poslednjih godina je u porastu broj zemalja koji je na nacionalnom nivou doneo odluke kojima se teži da se obezbedi masovno uvodjenje informatike u obrazovanje a slične mere se izučavaju u mnogim drugim zemljama, kako u zemljama u razvoju tako i u industrijalizovanim. Svuda u svetu bi trebalo da lica, odgovorna za obrazovanje u ovoj oblasti koja se brzo razvija, podrže njihovu akciju u pogledu doprinosa nauke kao i u pogledu sistema metodičnog praćenja i objektivnog procenjivanja ostvarenih rezultata.

Već postoji obiman uzorak znanja koji bi, malim naporima oko reorganizovanja, mogao biti stavljen na raspolaganje zemljama koje žele da ga koriste.

Kolege i istraživači iz raznih zemalja trebalo bi da ujedine svoje napore u jačanju informativne razmene i korišćenju metoda istraživanja podesnim da daju rezultate koji se mogu objedinjavati na medjunarodnom nivou.

Medjunarodne organizacije koje rade u ovoj oblasti, na nevladinom nivou ili na nivou vlada, na regionalnom ili svetskom, trebalo bi da pojačaju one svoje aktivnosti koje obuhvataju razmene i pribegavaju kooperaciji, naročito u pogledu onih tendencija i promena koje su pozvane da obeleže budućnost.

Potvrdjujemo našu čvrstu nameru da unapredimo istraživanja koja su takve prirode da, mimo razlika ekonomske i kulturne prirode, vode pojačanom ovladavanju računarima i njihovom optimalnom korišćenju u obrazovanju. Pozivamo stoga sve one koji su kompetentni za ovu oblast da učestvuju u usavršavanju multinacionalnih istraživačkih projekata ili oglednih projekata od medjunarodnog interesa.

[1] Martin Carnoy, Liza Loop (eds): Informatique et éducation: quel est le rôle de la recherche internationale? UNESCO, Paris, août 1986.

S francuskog prevela Dana Milošević

uputstvo za autore

Čitaoci-autori mogu slati redakciji članke kao i priloge za sve druge rubrike u časopisu. Članci mogu biti naučni, stručni i pregledni. Kategorizacija članaka vrši se pri recenziji. Izveštaj recenzenta autor dobija najkasnije dva meseca od prijema rada u redakciju. Rukopis treba pisati sa jedne strane A/4 lista pisaćom mašinom sa proredom (najviše oko 15 lista). Članak treba da sadrži: naslov, ime autora, kratak sadržaj, ključne reči (najviše 6 reči), uvodni deo, rezultate istraživanja, zaključak i spisak literature. Na kraju rada navesti ime i prezime autora sa titulama i adresom. Referisanje na bibliografske jedinice u radu vrši se navodjenjem broja reference u spisku literature (npr. [5]). Bibliografske jedinice se navode u spisku literature pod rednim brojevima bez uglastih zagrada (npr. 5). Ako se u literaturi navodi rad iz časopisa, tada se piše: ime autora, naslov rada, naziv časopisa, broj i godina publikovanje. Kod knjiga se piše: ime autora, naslov knjige, izdavač i godina izdanja. Crteži moraju biti crno-beli na pausu ili belom crtaćem papiru. Crteži mogu biti uvećani tako da se pri štampanju mogu umanjiti, ali se pri štampanju ne mogu uvećati. Crteži moraju biti pripremljeni za štampu, a tekst pisan pisaćom mašinom, ali se posebno priprema za štampu.

Autori šalju redakciji original teksta članka i crteže pripremljene za štampu. Rukopisi se ne vraćaju. Za svaki publikovani članak autor dobija 10 separata članaka. Članci se ne honorišu, osim ako su pisani po posebnom zahtevu redakcije.

Prilozi za sve druge rubrike časopisa šalju se u jednom primerku pisani pisaćom mašinom. Prilozi koji se prihvate za publikovanje honorišu se, a u slučaju da se ne prihvate ne vraćaju se autorima.Poželjno je da autori sav materijal za časopis (članke i priloge) šalju i na disketama. Datoteke moraju biti u ASCII-kôdu. Po završenom publikovanju diskete se vraćaju autorima.

Rukopise članaka i priloge slati na adresu: IRO "Nova knjiga", za časopis RAČUNARSTVO U NAUCI I OBRAZOVANJU, Ada Ciganlija 6, Poštanski pregradak 2098, 11000 Beograd.

SA NAJLEPŠIM ŽELJAMA.
ZA USPEŠNU SARADNJU!