M. Essert

Željeni lik informatičara danas ?

Sažetak

U članku je ukratko, u osobnoj impresiji, predočen informatički razvitak i pregled informatičkih znanja koja su danas nužna u izobrazbi tehnici usmjerenog studenta. Prikazana su temeljna informatička područja: programiranje, baze podataka, procesna računala, računalne mreže i multimedija. Posebno su dotaknuta znanja potrebna informatičaru, kako bi se on mogao lako uklopiti u kompjutorski svijet tehnologijski najvećih, generičkih promjena. Člankom se posredno pokušava odgovoriti ne samo što, nego i zašto su nam baš ta znanja potrebna, te kako ih povezati s drugim znanjima u današnjoj fakultetskoj naobrazbi.

1. Uvod

Svrha ovog članka je na popularan način pokazati bitne značajke informatike i smjerove kojima se ona razvija, kako bi iz tog razmatranja izronio željeni lik informatičkog stručnjaka danas, sposobnog da bude dovoljno širok da stvari razumije u širem tehničkom i humanističkom smislu, te dovoljno dubok da to svoje znanje umije brzo i kvalitetno primijeniti. 'Popularan način' nužno unosi pojednostavljenja, što implicitno znači stanovitu netočnost, a često puta stvara opasnu nedorečenost. Zbog toga se stručnog ili savjesnog čitatelja moli za ispriku i upućuje na literaturu. Članak ne pretendira da bude "state-of-the-art" ili "survey" područja.

Posebnu težinu ima i riječ 'danas', što nas upućuje na datum otisnut na naslovnici časopisa. Sigurno ste i sami primjetili da Vaše računalo svake godine gubi 50% svoje vrijednosti i da potpuno zastarjeva nakon 3-5 godina. Razlog je jednostavan: gotovo eksponencijalni razvitak ovog područja (od očvrsja, engl. *hardware*-a do programa, engl. *software*-a, kraticom: hw/sw) daje svakih 6 mjeseci novu tehnologiju, nova poboljšanja, nove smjerove.

Pisati o razvitku samo jednog segmenta, npr. razvitku programskih jezika iziskuje puno prostora i vremena, a nosi za moju generaciju i osobna obilježja (u šaljivom tonu):

- rođen u godini kad se pojavio Fortran, prvi programski jezik;
- gimnaziju pohađao kad je Fortran već doživio svoju 4. verziju (Fortan IV), a uz njegov bok već su se pojavili Forth, Logo, Cobol, PL/I, B, Simula, Algol, Smalltalk, Basic, Lisp, Snobol;
- završio fakultet kad su dominirali Pascal, C, Ada, Modula II, Mesa, Smalltalk-78, awk, csh, MS Basic 2.0, Scheme MIT, ML, Icon;
- doktorirao u vremenu pojavljivanja Object Pascal-a, Concurrent C-a, Common Lisp-a, SML-a. Caml-a, Haskell-a, Perl-a, Eiffel-a, OO Forth-a, MUMPS(FIPS)-a, APL 2, Module 3, ABC, Oberon-a, Tcl/Tk;
- prošao domovinski rat i godine obnove dok su Zapadom 'vladali' Delphi, Ada 95, LiveScript, Java 1, Ruby, Self 4.0, PHP/FI, Sather 1.1, Objective Caml, Haskell 1.3, PostScript, Prolog, Object Rexx;
- uči svoje studente C++/Java 2 i PHP, smatrajući da se s tim znanjem mogu pokriti i ostali jezici koji su se do sada održali: Fortran 2000, Tcl/Tk 8.4.1, Delphi 7, Python 2.3a2, C#, Ruby 1.6.8, Self 4.1.6, Perl 5.8.0 i O'Caml 3.06.

Pod pojmom informatike podrazumijevamo skupljanje, obradbu i predočivanje informacije - sveukupnog ljudskog znanja. Znanje kao apstraktna kategorija, ovom se tehnologijom obrađuje u digitalnom obliku, temeljenom na dvije osnovne veličine: *jest* i *nije*, ili - što se češće označuje - s *1* i *0*, *istina* i *laž*, *visoko* i *nisko* logičko stanje ili naponska razina.

Doista je fascinantno da se i slovo i zvuk i slika (pa prema tomu i film kao niz slika) dadu predstaviti nizom takvih 0 i 1. Jedan takav, laserom zapisan niz od otprilike 65 km duljine, zapis je na našem držaču zvuka (popularnom CD-u) s jednim ili deset (ako se komprimira) sati glazbe ili enciklopediji s par milijuna pojmova ili biblioteci s nekoliko ormara knjiga, tj. oko 700 milijuna

znakova (za usporedbu - Biblija ima 3.4 milijuna znakova), čak i s cjelovečernjim filmom spremljenim u komprimiranom formatu.
Nerijetko se pod znanjem informatike smatra poznavanje nekih korisnih programskih paketa, npr. *MS Office*-a za uredsko poslovanje, *Catia* ili *Ideas* paketa za tehničko projektiranje, *Adobe Photoshop*-a za grafičku pripravu i sl. No, njihovo poznavanje danas je nužno tajnici, tehničaru i grafičaru. Normalno je da to znanje posjeduje i inženjer, koje on može postići ili samostalnim učenjem ili preko tečajeva, te kao popratni (usputni) sadržaj predmeta koji se bave zadanom problematikom (konstruiranje, kompjutorska grafika). Znanje pak koje informatičar mora imati je da stvara, razvija i mijenja takva ili slična programska rješenja, te ima temeljna znanja svih smjerova koje informatika pokriva. Ti smjerovi danas su: kompjutorsko programiranje, baze podataka, procesna računala, računalne mreže i čovjek-stroj komunikacija. Ovakva podjela informatike razlikuje se od uobičajenih klasifikacija (npr ACM), a temelji se na funkcionalnoj projekciji objekta (računala) u područja proučavanja (informatika). Svaki od tih smjerova ima svoje podsmjerove, koji, kao i život, imaju svoj rast, uspon i pad. Ne vidjeti to, znači odlučiti se baviti okaminama, umjesto diviti se i proučavati život koji buja oko nas.

2. Programiranje

Elektronički krugovi unutar računala upravljaju se (uključuju i isključuju: 1 ili 0) putem strojnog programskog kôda koji obuhvaća stotinjak naredbi predstavljenih binarnim zapisom, istim digitalnim zapisom kojim je predstavljena bilo koja apstraktna informacija. Razlika između naredbe i podatka u zapisu prema tomu ne postoji, ona je rezultat dobre definicije programskog kôda i njegovog slijednog, taktom upravljanog, izvođenja. O frekvenciji takta ovisi brzina izvođenja osnovnih naredbi. Programi pisani u jezicima više razine uvijek će se morati prevesti u programski kôd niže razine, tj. u niz naredbi strojnog kôda. Ovaj proces je automatiziran i uvjetuje, definira, svaki programski jezik da bude konzistentan sa zadanom fizikalnom strukturom stroja, njegovom arhitekturom. Analogno nabrajanju programskih jezika iz uvoda, bilo bi nabrajanje fizikalnih struktura i arhitektura strojeva, od prvih kompjutorskih soba, preko trokrilnih ormara, velikih i malih kutija, do današnjih računala na dlanu. I dok se u tehnološki nerazvijenoj sredini poput naše razvitak software-a može donekle pratiti, projektiranje i izgradnju računala možemo očekivati tek u slučaju premještanja kompjutorske tehnologije s Dalekog istoka, što se u ovom trenutku ne nazire. Stoga je izobrazba našeg informatičkog kadra uvjetovana našom trenutačnom gospodarskom situacijom, pa se poduka software-skog znanja preferira pred hardwer-skim.

2.1. Računalni programi

Kompjutorski programi su tekstovi koji određuju kako računalo treba djelovati, što treba načiniti - izračunati, nacrtati, pokrenuti, ispitati ... Oni su ograničeni programskim jezikom u kojem su napisani i, s nekoliko časnih iznimki, njihov jezik se ne može proširiti radom takvog programa. Točnije, nije moguće mijenjati postavljenu jezičnu semantiku. Rješenje bilo kojeg zadanog problema nalazi se na višoj razini, u apstraktnom algoritmu, koji se preslikava u taj niz programskih naredbi koje izvode zamišljeni postupak nad zadanim podacima. Zato je za pisanje računalnih programa prije svega nužno razumjeti i naučiti algoritme, barem one temeljne, kao što je za matematičara nužno razumjeti matematičku indukciju i pravila zaključivanja.

Razlika između programa i programskog jezika (zvat ćemo ga jednostavno: jezik) je stvar dogovora, ali ona ima neke korisne posljedice - redovito je pisanje novog jezika puno teže nego pisanje korisničkog programa, što znači da ljudi moraju naučiti mali broj jezika da bi mogli učitati u računalo i zatim pokrenuti veliki broj programa.

Zato se programiranje (rješavanje problema kompjutorskim programom) može razvijati u dva smjera: razvitkom boljih algoritama i razvitkom jezika u kojima se algoritmi izražavaju.

Algoritmi su zbog svog apstraktnog karaktera nepromjenjivi i stoga je njihovo poznavanje donekle jedini čvrst oslonac u dinamičnoj kompjutorskoj tehnologiji. Njihov pak razvitak jednako je tako buran – treba pratiti npr. nove genetičke algoritme, algoritme za NP-teške probleme, algoritme mrežnih optimiranja i sl.

S druge strane, u razvitku jezika koje je imalo poseban procvat 80-tih godina, sad već prošlog stoljeća, došlo je do stanovite unificiranosti – nove verzije programskih jezika nastoje uključiti sve

dotad poznate programske paradigme, pa se jezici počinju razlikovati jedino po sintaksi. Njihov razvitak nije završen, ali se težište razvitka prebacuje na projektiranje velikih programskih sustava i kolaboracijskog rada.

2.2. Programski jezici

Kao što se fizikalni zakoni izražavaju formulom, matematičkim jezikom, tako se algoritmi (postupci ili procedure) nekog kompjutorskog problema rješavaju programskim jezikom. Taj jezik, prije svega mora sadržavati spremišta podataka, zovemo ih varijablama, nad kojima program obavlja jednostavne operacije, često puta povezane u složene.

Za matematičara, na primjer, izraz X=X+1 je neistina, pa prema tome besmislica. Za informatičara, programera, to je osnovna naredba s dva operatora (pridružbom i zbrojem) i jednom promjenljivicom, varijablom X. Njeno značenje je: nova vrijednost X varijable poprima staru vrijednost uvećanu za 1. Kao što se više istovrsnih podataka povezuje u polje (za matematičara je to vektor ili matrica), tako se i niz operacija povezuje u petlju (engl. *loop*), s mogućnosti ispitivanja uvjeta (ako-onda-inače, prekini, izađi, nastavi i sl.). Izgradnja složenijih oblika programa, kao i načina korištenja računala (s tekstualnog na grafičko sučelje, s tipkovnice na miša), pokrenula je razvitak načina ili 'filozofije' programiranja: od proceduralnog ili top-down programiranja, preko funkcionalnog do objektnog.

2.2.1. Proceduralni jezici

Programi pisani u proceduralnim jezicima sastoje se od mnogo međusobno povezanih struktura podataka i naredbi koje ih mijenjaju. Glavna poteškoća ovakvog pristupa je da kako programi postaju složeniji tako raste kompleksnost hijerarhijskog ustroja podataka. Naredbe napisane od strane jednog programera mogu činiti ispravne promjene podataka koje on razumije, ali drugi programer može dodati novu informaciju za koju vjeruje da će se obraditi na prikladan način. No, ako se to ne dogodi, podaci ne poprimaju predviđene moguće vrijednosti zamišljene od strane obaju autora, pa se program kod izvođenja ruši.

2.2.2. Funkcionalno programiranje

Zbog te ranjivosti hijerarhijskih struktura podataka i poteškoća timske suradnje na izgradnji složenog programskog projekta nastali su tzv. funkcionalni, neproceduralni jezici (npr. LISP) i logičko programiranje (npr. PROLOG). Naredbe postaju elegantne, rekurzivne postavke jednostavne i moćne, broj naredbi za složene apstraktne modele se smanjuje. No, (sve ima svoj 'ali') novi pristup briše mogućnost opisa vrijednosti koja se mijenja, ono jedino i čvrsto što je programer naučio, dovodeći ga u svijet novih pravila. Umjesto brojanja podataka u listi, na primjer, programer sad raspolaže nizom naredbi koje se odnose na 'idući član u listi', a postupak se ponavlja s 'preostalim elementima' u listi. Nema više eksplicitne informacije o podatku u listi, njegovom mjestu u strukturi, niti varijabli-brojilu koje pokazuje trenutačno mjesto u procesu traženja ili promjene podataka. Funkcionalni jezici izjednačavaju tako pristup podacima i naredbama (gubi se njihova prvotna razlika), nudeći pouzdaniji software na račun stanovite krutosti i formalizma pisanja programa. Programiranje u čistim funkcionalnim jezicima postaje stoga teže, pa se zadržalo samo u znanstvenom i istraživačkom okolišu. No, razvijene ideje našle su svoje mjesto u snažnim matematičkim alatima (npr. WRI Mathematica-i) i novom pristupu programiranja, tzv. objektnom programiranju (OOP – object oriented programming).

2.2.3. Objektno programiranje

Sada se u apstraktno zamišljenoj klasi i njezinoj realizaciji nazvanoj objekt, povezuju i podaci i naredbe koje nad tim podacima djeluju, tvoreći zasebnu cjelinu. Sve se te zasebne cjeline mogu povezivati s različitim razinama otvorenosti za promjene i izmjene podataka, kao i multiplicirati (klonirati, nasljeđivati svojstva i sl.). Tako je nastao jedan poseban svijet programskih 'lego-kockica', jednostavnijih, a moćnijih, od negdašnjih 'biblioteka' programa. Usporedo s nastajanjem novih knjižnica, ovog puta klasa (a ne potprograma, kao nizova naredbi) koje uključuju i podatke i funkcije koje nad tim podacima djeluju, razvija se i arhitektura izgradnje većih cjelina iz osnovnih. Pokazuje se da se mnogi problemi dadu svesti na samo desetak osnovnih, temeljnih. Ti kanonski primjerci nazvani

su *uzorcima* (engl. *patterns*). Težište negdašnjeg programiranja u posljednjih 10 godina pomaklo se na projektiranje uzoraka (engl. *design patterns*) – elementi OOP-a (Object Oriented Programming) jednom napisani koriste se s lakoćom u tisućama primjena, treba 'samo' uočiti kojem uzorku naša primjena odgovara.

2.3. Programiranje procesnih računala

Posebna kategorija programiranja odnosi se na programiranje mikrokontrolera – uređaja koji su preplavili industriju i našli primjenu u svih područjima ljudske djelatnosti. Zbog ogromne primjene cijena im je izuzetno niska (oko 2 US $ po komadu), a razvitak konstantan – svakih 6 mjeseci novi tip računala, nova komponenta - veća integracija, veće mogućnosti i manja cijena. U tom svijetu procesnih računala posebno mjesto danas imaju familije PIC-eva (Peripheral Interface Controller). Informatičar treba upoznati strukturu kontrolera na razini block-dijagrama (nije potrebno, a nije ni moguće upoznati tehnologiju izvedbe), kako bi ga znao programirati. Programiranje se ostvaruje putem posebnih programatora, ili preko osobnog računala, i to u najnižem programskom jeziku (assembler-u), koji je na razini programskog kôda komponente, ili u nekoj varijanti C programskog jezika posebno razvijenoj za tu primjenu.

Kako redovito pojedinačna rješenja teže integraciji u složene sustave, tako se i u svijetu procesnih računala i njihovog software-a ugrađenog u njih (engl. *embeded systems*) događaju pomaci – sve više se postavljaju zahtjevi na rješenjima povezanih sustava (*distributed systems*), što pak uključuje znanja računalnih mreža.

3. Računalne mreže

Računala mogu međusobno razgovarati. To njihovo važno svojstvo uključuje hw/sw potporu. Ne ulazeći u hardware-ske probleme prijenosa informacije (telefonske žice, modemi, kabliranje, switch-evi, router-i, optika i sl.) zanimaju nas temeljna informatička znanja potrebna studentu. Na to pitanje nije lako, ni jednostavno odgovoriti. S jedne strane to su znanja o komunikaciji, a s druge strane tu su nove tehnike programiranja. Jer mrežu ne čini samo povezanost dva računala, nego povezanost računala jedne tvrtke, banke, sveučilišta, grada, države i na koncu povezanost računala cijelog svijeta. Mreža svih mreža - Internet povezuje danas više od 500 milijuna računala i barem isto toliko ljudi, a 'hrani' se svakodnevno s otprilike 90 milijuna dokumenata. Njezin najpoznatiji servis WWW (World Wide WEB - svjetska paučina) samim imenom dovoljno govori. Ako je današnja procjena uskladištene informacije na Internetu oko 8000 TByte-a ($8\ 10^{15}$ okteta, znakova), onda svako naše bavljenje računalom mora počinjati i završavati s pitanjem – a gdje je tu Internet?

Internet je razvio mrežne servise i usluge (ftp, telnet, rpc), tehnologije komunikacije (client-server, peer-to-peer), programske alate (XML jezik, DOM i SAX parseri, XSLT) i serverske jezike (JSP, ASP, PHP i druge). Informacija koja se sprema i prenosi nije više vezana za jednu vrstu uređaja (računalo) nego za niz komunikacijskih uređaja i formata (moby, PDA-personal digital assistent, kamere i sl.).

Pogrešno je misliti kako je za objavljivanje informacije na mreži dovoljno poznavati neki komercijalni program tipa 'Dreamweaver' ili 'Front-Page', te da nije nužno spuštati se na razinu jezika s oznakama (XML, HTML). Danas se sve više (a u budućnosti će to biti isključivo) traže dinamičke stranice - informacije koja se generiraju na temelju korisnikovog upita. Autor takvih stranica mora dobro poznavati jezik oznaka, protokole komunikacije (TCP/IP, HTTP, SOAP) i jezik u kojem to treba napisati. Kako se informacija na serverskoj strani najčešće nalazi u kompjutorskoj bazi podataka, nužno je poznavanje rada i programiranja kompjutorskih baza.

4. Baze podataka

Baze podataka su prisutne u svijetu računala od njihovog početka. I kao što su se mijenjali programski jezici, tako su i kompjutorske baze doživljavale razvitak, kako u funkcionalnom smislu, tako i u razvoju jezika koji ih izgrađuju. Slično kao što danas svaki programer treba znati C i Java programski jezik, tako se i u bazama znade standard: SQL jezik. Iako ima mnogo konkretnih softverskih proizvoda koji poštuju taj standard, npr. IBM-ov DB2 te Microsoftov SQL Server, moguće

je istaknuti dva najrasprostranjenija proizvoda: Oracle i MySQL. Prvo je komercijalna baza koja je uspjela istisnuti mnogostruku konkurenciju i zagospodariti tržištem. Ne smije se zaboraviti da bilo koje bankarsko ili ekonomsko tržište ne može funkcionirati bez kompjutorskih baza, pa takva informacija ima veliku težinu. Treba znači poznavati Oracle. No, ako ste student i nemate sredstava kupiti takvo komercijalno rješenje (premda postoje i Oracle studentske inicijative), onda je MySQL kao 'open-source' (besplatno) rješenje izvanredno. Radeći u MySQL-u korisnik nauči 90% svih potrebnih stvari – od relacijskog modela, preko interaktivnog SQL jezika, do svih traženja i sortiranja koje se od baze očekuju. Jedino se transakcije i posebni slučajevi obnavljanja sadržaja u slučaju rušenja baze mogu dobro naučiti tek s Oracle-om. Dakako, Oracle ima i svu prateću sistemsku potporu za razvijanje aplikacija u najrazličitijem okolišu, dok se kod MySQL-a očekuje jednostavno programiranje, npr. u PHP-u i ODBC integracija s ostalim sustavima (npr. Micorosoftovim Access-om). Microsoft je kao najveći proizvođač software-a danas sve svoje proizvode orjentirao prema korisnikovoj lakoj upotrebi, što je dovelo do kompjutorske revolucije u poimanju "čovjek-stroj". Razvilo se novo područje kompjutorskog svijeta sadržanog u jednoj riječi – multimedija.

5. Multimedija

Multimedija predstavlja kombinaciju zvuka, grafike, animacije i videa. U računalnom svijetu ona je podskup hipermedije (engl. hypermedia), u kojoj se kombiniraju elementi multimedije s hipertekstom, u kojem se informacija povezuje svezama (engl. links). Tako se dolazi do interaktivnog formata pristupu informacije koju korisnik na brz i jednostavan način sâm izabire.

Dugo vremena se unos podataka u računalo (ne mislim ovdje na bušene kartice ili magnetske vrpce) odvijao preko naredbene linije – upiše se naredba, pošalje i stroj na nju odgovori. Danas su pred vama ikonice (svejedno da li je operacijski sustav Microsoftov ili je Linux) i kazalo miša. Klikom na ikonicu pokreće se program koji ona predstavlja. A taj program u sebi opet ima izborne ponude ili nove ikonice i korisnik većinu vremena uz računalo provodi klikanjem. I to na isti način za slušanje glazbe, crtanja slika ili gledanja video zapisa. S korisničke strane stvar je jednostavna (ili tako izgleda), ali s programerske strane stvari postaju složene. Bez objektnog programiranja tu se više nema što raditi, treba poznavati COM ili .NET objekte, treba proučiti ActiveX kontrole. Programiranje, kao i rezultati, postaju vizualni. Vizualni, multimedijski svijet, zahtjeva i iste takve alate. To je sasvim novo područje programiranja. Štoviše, u programsko rješenje ulaze konstrukcijske (design-erske) sposobnosti, estetski ukus i psihološka domišljatost programera i konstruktora. Novo grafičko sučelje (GUI) uvjetuje da proizvod bude slično grafički obrađen, a programski kôd prepun gotovih objekata i kontrola, zvukovnih i video efekata, animacija. Kao da ste iz hladnog laboratorija prešli u cirkusku arenu. To je današnjica. Na tržištu ostaju samo šareni, zabavni, makar i površni.

Ispuniti to nije lako. Tu su znanja objektnog programiranja i vizualnih alata (Visual C, Macromedia Flash, C#), algoritmi kompresije podataka (zvukovnih, grafičkih i video zapisa), različiti prijenosi (streaming) i kriptiranja. Znanje nužno zahtjeva poznavanje složenijih algoritama, ne samo onih koja se uče u prva dva semestra. I što je najvažnije, nužno je razumjeti da danas u kompjutorskom svijetu pojedinac ne može puno načiniti – potrebne su ekipe, potrebna već gotova rješenja objekata i kontrola (koja se, dakako, plaćaju), potrebni mediji i management da računalni proizvod uopće bude prihvaćen. Ako ste pritom i vrlo stručni, shvaćate vrlo brzo kako su novi kompjutorski izazovi tek na početku. Studente treba stoga od samog početka podučavati ekipnog radu u programiranju.

6. Informatičar

Nabrojena informatička znanja potrebna su bilo kojem studentu tehničke usmjerenosti i mogu se prenijeti čak samo s po jednim informatičkim predmetom u svakom semestru. No, ako je studentova usmjerenost na informatiku veća, onda bi u zadnje dvije godina studija trebao po svakom semestru slušati barem po jedan informatički predmet i barem jedan izborni, fakultativni. Oni obavezni kolegiji svakako bi ga trebali osposobiti za ekipni rad, što danas nužno znači projektiranje software-a, konkrentno, rad s programskim OOP uzorcima (eng. *patterns*). Osim toga trebao bi dublje poznavati jezične procesore i operacijske sustave, kako bi ih mogao što bolje koristiti, mijenjati i poboljašavati. U tomu mogu pomoći znanja iz dva područja: područje paralelnog programiranja i

područje zaštite sustava. U prvom području uči se paralelizam kompjutorskog rada na razini podataka, naredbi i strojeva (od komponenti do strojeva), a za drugo je nužno znanje kriptografije i operacijskih sustava zaštite. Danas je već uobičajeno da i svako osobno računalo spojeno na mrežu ima svoj *firewall* – zaštitni program koji ispituje sve ulazne i izlazne komunikacijske aktivnosti. Nije dovoljno znati koristiti, potrebno je znati to načiniti. Posebno je to važno u sustavima visokih kriterija zaštite, npr. bankarskim, policijskim ili vojnim sustavima.

Što se izbornih kolegija tiče bitno je da pokrivaju područja kojim se stečeno znanje produbljuje: npr. WEB programiranje, konstrukcija compiler-a, znanstveno računalstvo ili računalna matematika. Važno je pritom da nema preklapanja u već obrađenim područjima, da se ne događa ponavljanje gradiva ili onaj vječni, dugotrajni 'ab ovo' - uvijek isti uvod (od jajeta nadalje).

7. Zaključak

Izobrazba informatike u Hrvatskoj danas može se razlikovati od one koja se ostvaruje na Zapadu samo u naglascima onih područja za koje imamo tehnologijske mogućnosti prenošenja. To se u prvom redu odnosi na učenje software-a. Istina je da se taj prijenos svjetskog znanja informatike može događati pri različitim fakultetskim odjelima, makar je najprirodnije vezati ga uz matematiku. Dokaz za to su nam brojni 'Department of mathematics and computer science' u svijetu i njihovi rezultati. Pritom treba posve odvojeno razmatrati matematička područja od informatičkih, utopiti ih u matematiku isto je tako pogubno kao smatrati ih elektrotehnikom. Štoviše, iskustva pokazuju da informatičarima danas najviše nedostaju ona stručna znanja iz područja u kojima rade. Zato bi pravi informatičar bio onaj kojemu je fakultet osim matematičkog i informatičkog znanja pružio i primjenjena znanja one grane ljudske djelatnosti za koju se student opredijelio. Na višim godinama studija to bi po semestru bila barem po dva predmeta stručnog profila (npr. bankarstva, ekonomije, prometa i sl.) za koju granu se informatičar educira. Sama informatika je kao i matematika – lijepa i beskorisna. Budući informatičari imat će znanja koja će tu nesavršenost ispraviti. U kojoj mjeri, vidjet ćemo. Sve počinje od dobrih temelja.

LITERATURA

1. A.S. Tanenbaum, M. Van Steen, *Distributed systems: Principles & Paradigms*, Prentice Hall, 2002.
2. W. Zuser, S. Biffl, T. Grecherig, M. Kohn*, Software Engineering mit UML und dem Unified Process*, Pearson Studium, 2001.
3. N. Wirth, J. Gutknecht, *Project Oberon: The design of an Operating System and Compiler*, Addison-Wesley, 1992.
4. A.W.Appel, *Modern Compiler Implementation in Java*, Cambridge University Press, 1998.
5. A.S. Tanenbaum, *Structured Computer Organization,* 5th ed., Prentice-Hall, 1999.
6. Dale & Lewis, *Computer Science Illuminated,* Jones and Bartlett Publishers, 2002.
7. R. Elmasri, S. Navathe, *Fundamentals of Database Systems*; 3/E, The Benjamin/Cummings Publishing Company, Inc., 2000.
8. L. Welling, L. Thomson, *PHP and MySQL Web Development*, Sams, 2003.
9. D. W. Smith, *PIC in Practice: An Introduction to the PIC Microcontroller*, Butterworth-Heinemann, 2002.
10. C. Rey, B. Schneier, *Macromedia Flash MX: Training from the Source*, Peachpit Press, 2002.