PROGRAMSKI JEZIK PASCAL I

MATJAŽ GAMS (1),
IVAN BRATKO (2,1),
VLADIMIR BATAGELJ (3),
ROBERT REINHARDI (1),
MARK MARTINEC (1),
MARJAN ŠPEGEL (1),
PETER TANCIG (1)

UDK: 519.682.8

(1) INSTITUT „JOŽEF STEFAN", LJUBLJANA
(2) FAKULTETA ZA ELEKTROTEHNIKO, UNIVERZA EDVARDA KARDELJA, LJUBLJANA
(3) FAKULTETA ZA NAROSLOVJE IN TEHNOLOGIJO, UNIVERZA EDVARDA KARDELJA, LJUBLJANA

V članku je opisano mesto programskega jezika Pascala glede na ostale višje programske jezike. Opisani so značilni predstavniki najbolj pomembnih skupin programskih jezikov in posebej algoritmični jeziki, kamor uvrščamo tudi Pascal. Sledi kratek opis zgodovine Pascala in zaključna primerjava z ostalimi jeziki. Skozi celoten članek je osnovno merilo enostavnost oz. zapletenost pri programiranju s stališča človeka - uporabnika.

P r o g r a m m i n g L a n g u a g e P a s c a l I (comparision with other programming languages). The programming language Pascal is compared to other high level programming languages. Programming languages are devided into most important groups and one representative of each group is very shortly described. Special care is devoted to "algorithmic" languages. Finally some conclusions are drawn. Throughout the paper the emphasis is on the human engineering side.

1. Uvod

Programski jezik Pascal se v svetu čedalje bolj uveljavlja, čeprav ni miljenček nobene velike računalniške firme, tako kot npr. IBM podpira PL/1. V zadnjih letih pa se v literaturi poleg hval pojavlja precej kritik Pascala. Marsikdaj so te kritike upravičene. Zato je smiselno zbrati te kritike in jih objektivno oceniti /25/. Poleg teh pogosto drobnjakarskih kritik pa je potrebno oceniti Pascal tudi v širšem kontekstu programskih jezikov. Prava uveljavitev Pascala šele prihaja /1/, njegova usoda, kot enega najboljših predstavnikov skupine "algoritmičnih" (tudi postopkovnih, proceduralnih) jezikov, pa je v daljni bodočnosti negotova.

Tudi drugi razlogi so botrovali nastanku tega članka, npr. zmotne trditve v literaturi, da je Pascal primeren predvsem za učenje, ali marsikatera neutemeljena kritika v strokovni literaturi. Po drugi strani pa v nekaterih okoljih prevladuje pretirano navdušenje glede Pascala. Pomembno je dejstvo, da Pascal (upravičeno) prevladuje v slovenskem šolstvu /2/.

Radi bi se zahvalili vsem, ki so sodelovali pri oopravljanju članka, predvsem pa : Janezu Žerovniku, Maretu Bohancu, Henriku Krnecu, Damjanu Bojadžijevu in Igorju Mozetiču.

2. Programski jeziki

Na kratko se bomo ustavili pri značilnih skupinah visokih programskih jezikov (glej sliko 1). Osnovno merilo za razdelitev je način izražanja oz. oblikovanja ideje v delujočo kodo.

2.1. Skupine visokih programskih jezikov

SKUPINA	PREDSTAVNIK	SKRAJNI DOMET(*)
algoritmični j.	FORTRAN	PL/1
	COBOL	PL/1
	ALGOL-60	ALGOL-68
	PASCAL	ADA ?
	BASIC	
	C	
matrični j.	APL	(izvedenke?)
funkcijski j.	LISP	-"-
logični j.	PROLOG	-"-
vzorčni j.	SNOBOL	-"-
objektni j.	SMALLTALK	-"-

Slika 1. Skupine programskih jezikov.

(*) Tudi nenor, izvedenka, strannost, angleško "black hole".

Pri ocenjevanju uspešnosti programskih jezikov, ki smo jih označili s "SKRAJNI DOMET", so mnenja deljena. Npr. Hoare v /3/ očita tovrstnim jezikom kopico pomanjkljivosti, nekateri drugi avtorji pa jih zagovarjajo. O tej problematiki bomo še govorili v nadaljevanju članka.

Vsako skupino jezikov smo poskusili okarakterizirati samo z enim pridevnikom, ki skuša opozoriti na stil oz. način programiranja, ki ga jezik najbolj podpira. To je bilo tudi osnovno merilo za razdelitev jezikov v skupine. Stili programiranja so opisani hkrati z opisom skupin. Skupine so v veliki meri privzete po /4/. Pri vsaki

skupini je ocenjen najbolj značilen jezik in primerjava s Pascalom. Velja naslednje:
-vsaka skupina jezikov je za določeno področje bolj primerna kot druga skupina.
-V Pascalu lahko z nekaterimi spremembami (najčešče s spremembo stila programiranja in naborom podprogramov v knjižnjicah) dokaj uspešno konkuriramo nekaterim jezikom.

2.2. Opis najbolj značilnih nealgoritmičnih jezikov

APL (A Programming Language) temelji na konverzacijski interakciji preko posebnega terminala. Nastal je okoli šestdesetih let. Je funkcijski jezik z obsežnim naborom uporabnih podprogramov. APL ima goreče nasprotnike in zagovornike. Zagovorniki trdijo, da so bistveno bolj produktivni, kadar uporabljajo ta jezik /5/. Nasprotniki trdijo, da je jezik popolnoma nepregleden. Po mnenju nekaterih strokovnjakov je mogoče podobno produktivnost doseči v večini algoritmičnih jezikov, če uporabljamo knjižnjice z ustreznim naborom podprogramov. APL na prvi pogled ločimo od ostalih jezikov po množici nenavadnih znakov. Poglejmo si primer enostavnega stavka v APLju:

```
MID ← (A + B)÷2 ◇ FM ← FUN MID
```

LISP (LISt Processor) je najbolj široko uporabljan jezik umetne inteligence. Nastajal je okoli leta 1960 /6/. Poglejmo si primer stavka v LISPu, tj. telo rekurzivne procedure, ki računa faktoriela:

```
((COND ((ZEROP N) 1)
       (T (TIMES N (FACT (DIFFERENCE (N 1))))))
)
```

LISP podpira funkcijsko programiranje /7/, ki pa ga po našem mnenju lahko v veliki meri uporabljamo tudi v Pascalu. Funkcijsko programiranje je stil oz. način programiranja tako kot npr. strukturirano ali objektno programiranje. Zelo preprosto novedano je funkcijsko programiranje tako, da prevladujejo funkcije, oziroma vrednotenje in ne prireditveni stavki kot v običajnem postopkovnem programiranju. Poglejmo si na preprostem primeru stil funkcijskega programiranja:

```
IF NaRobuDosega(letalo)
  THEN SproziOpozorilo(letalo)
  ELSE IF ZnotrajDosega(letalo) THEN
    IF Nenajavljeno(letalo)
      THEN SprozIAlarm(letalo)
      ELSE IF NeznanoLeteceVozilo(letalo)
        THEN SproziPojzvedbo(stan)
        ELSE continue;
```

Na tak način lahko programiramo precej podobno kot v funkcijskih jezikih, kar pa Pascal ni. Prav tako Pascal nima množice ugodnih lastnosti LISPa, npr. udobnega procesiranja seznamov.

PROLOG je novejši jezik umetne inteligence. Omogoča nam tudi nedeklarativno in logično programiranje /8,9/, ki je v Pascalu često težko izvedljivo. Na PROLOG lahko gledamo tudi kot na produkcijski sistem /10,11/, kar je ena izmed alternativ bodočnosti programskih jezikov /1/. V Pascalu je smiselno uporabiti principe produkcijskih sistemov zlasti pri strukturiranju podprogramov /11/. Poglejmo si primer deklarativnega in nedeklarativnega načina programiranja na podprogramih za združevanje dveh seznamov in za ugotavljanje vsebovanosti elementa v seznamu:

```
PROLOG
conc([],X,X).
conc([X|Tail],Y,[X|Z]) :- conc(Tail,Y,Z).

Pascal
procedure Conc(x,y: tipSeznam; var z:tipSeznam)
BEGIN
  IF x = nil THEN z := y
    ELSE
     BEGIN
       Conc(x↑.next, y, z);
       DodajVSeznam(x↑.vsebina, z)
     END
END;(*Conc*)

PROLOG
member(X,Z) :- conc(L1,[X|L2],Z).

Pascal
FUNCTION Member(x: tipElement; z: tipSeznam) :
         boolean;
BEGIN
  IF z = nil THEN Member := false
    ELSE IF x = z↑.vsebina THEN Member := true
      ELSE Member := Member(x,z↑.next)
END;(*Member*)
```

PROLOGov stavek "member" preberemo takole: Element "X" je član seznama "Z", kadar lahko združimo nek seznam "L1" in seznam, ki je sestavljen iz elementa "X" in seznama "L2", v seznam "Z".

V Pascalu smo dokaj podobno kot v Prologu rešili nalogo z združevanjem dveh seznamov. Pri nedeklarativni verziji iskanja vsebovanosti pa vidimo, da tak način programiranja v Pascalu ni mogoč.

SNOBOL (String Oriented Symbolic Language) je vzorčni jezik/12/. Nastajal je koncem šestdesetih let. Novejše variante SNOBOLa so SPITBOL, SITBOL in FASBOL. SNOBOL omogoča asociativno in vzorčno programiranje. Poglejmo si princip vzorčnega programiranja v SNOBOLu:

```
 U 'Z' BREAK('+,-') = 'FIRST' : S(SEM) F(TJA)
```

Zgornji stavek preberemo takole: če se v spremenljivki "U" (tipa niz) pojavi vzorec, ki se začne s črko "Z" in mu sledi poljubno število znakov do znaka "+" ali "-", celoten vzorec zamenjamo z nizom "FIRST" in izvajanje nadaljujemo na oznaki "SEM". Če vzorca v spremenljivki "U" ne najdemo, se izvajanje prenese na "TJA".

S Pascalskim prevajalnikom, ki dopušča spremenljivo dolžino polj (dinamična polja), npr. kot v novem ISO standardu/22/, lahko dokaj uspešno konkuriramo SNOBOLu.

SMALLTALK je objektni jezik ali bolje rečeno sistem. SMALLTALK je del sistema, ki je nastal kot raziskovalni projekt osebnih računalnikov (Dynabook) /13/. Te raziskave imajo velik vpliv na razvoj najmodernejših osebnih računalnikov. Za vse te sisteme je značilno, da so močno povezani s celotnim programskim okoljem, ki je v Pascalu nomankljivo. Kaj je to objektno programiranje? Poglejmo si najprej razlike med vrednostmi in objekti. Vrednosti so abstrakcije, zato jih ne moremo spreminjati ali jim nastaviti vsebine - so časovno nespremenljive. Objekti so časovno spremenljivi, imajo stanje, ki odgovarja realnemu svetu. Objektom lahko spreminjamo vsebino, jih kreiramo ali brišemo, ali jim damo ime. Uporabnik vedno vidi objekte le "od zunaj", vsi objekti so zani enaki (ni različnih tipov). Uporabnik pošlje objektu sporočilo in objekt izvrši opravilo sam ali

tako, da drugim objektom pošlje sporočila. Poleg aktivnosti in komunikacije je pomembna lastnost dedovanje oz. deljivost. Na primer, zemlja in pomaranče sta okrogli in imata skupni vse lastnosti, ki pripadajo lastnosti "okroglost".

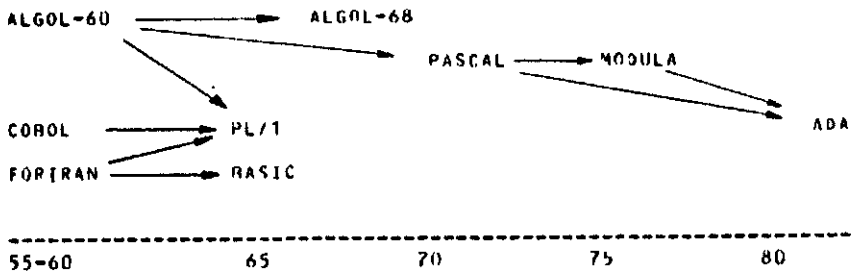
Stil objektnega programiranja je za Pascal precej neprimeren, čeprav ima velika večina jezikov vsaj nekaj objektnega. Po /14/ je programiranje objektno usmerjena matematika in matematika vrednostno usmerjeno programiranje. Pascalski zapis (record) ima nekaj lastnosti objektov, vendar ne podpira objektnega programiranja. Okviri (frame) v LISPu so veliko bližje pojmu objekta.

2.3. Opis značilnih algoritmičnih jezikov

Poglejmo si razvoj algoritmičnih jezikov:

BASIC (Beginner's ALL-purpose Symbolic Instruction Code) so razvili leta 1965 kot enostaven jezik za učenje programiranja in kot stopničko do FORTRANa in ALGOLa. BASIC ima kljub svoji relativni skromnosti pomembno vlogo v računalništvu kot jezik za pogovorno delo (interaktivnost) in v zadnjih letih zaradi svoje majhnosti in preprostosti kot jezik za domače računalnike.

PL/1 (Programming Language One) je nastajal v letih 1960 - 1965. Prvi prevajalnik je postal dostopen po letu 66. V PL/1 so upoštevane izkušnje FORTRANa, ALGOLa, COBOLa, itd., torej je jezik s širokim področjem uporabe in tudi primerno močnimi konstrukti. V PL/1 je vse, kar se da razumeti, tudi dovoljeno, zato poskuša razumeti tudi napake. PL/1 podpira IBM in to je verjetno tudi eden od vzrokov za precejšnjo razširjenost tega jezika.



Slika 2. Razvoj algoritmičnih jezikov.

Na sliki 2 vidimo razvoj algoritmičnih jezikov. Slika je prevzeta pretežno po /15/.

FORTRAN (FORmula TRANslation). FORTRAN je nastajal okoli leta 1955 kot program za prevajanje matematičnih postopkov v računalniške programe. Prvotno zelo enostavni verziji so se kmalu pridružile izpeljanke: FORTRAN II, III in IV. FORTRAN je že okoli leta 1960 postal zelo razširjen jezik. Leta 1966 sta bila sprejeta dva standarda za FORTRAN, ki približno odgovarjata FORTRANu II in FORTRANu IV. V zadnjih letih so se pojavili še FORTRAN V, FORTRAN 77 in cela vrsta "strukturiranih" FORTRANov. FORTRAN se uporablja predvsem za reševanje znanstvenih in tehničnih problemov.

ALGOL (ALGOritmic Language) je nastajal okoli leta 1960. To je bil prvi jezik, ki je bil tudi formalno opisan. Iz tega poskusa je zrasla teorija formalnih jezikov in prevajalnikov. ISO standard za ALGOL 60 je bil izdan leta 1965. ALGOL je močno vplival na nadaljni razvoj programskih jezikov, vendar je ostal pretežno evropski jezik. V Ameriki so ga izpodrivali jeziki, za katerimi je stala ameriška računalniška industrija. Kot izvedenka ALGOLa 60 se je pojavil ALGOL 68, vendar se ni uveljavil. Uspešnejše izvedenke so bile ALGOL W, SIMULA in PASCAL.

COBOL (COmmon Business Oriented Language) je nastal kot jezik za poslovne obdelave okoli leta 1960. Standardiziran je bil leta 1968. Za COBOL je značilna težnja, da bi bil čim bolj "naraven" - podoben naravnemu jeziku (angleščini). Ta lastnost naj bi omogočala precejšnjo samodokumentiranost programov in tudi preverljivost s strani neprogramerjev, npr. vodstvenih delavcev. COBOL in FORTRAN sta še vedno najbolj uporabljana programska jezika.

MODULA je nastala kot naslednik Pascala. Kmalu jo je nadomestila precej bolj uspešna MODULA-2 /16,17/. MODULA-2 je večnamenski jezik, ki je predvsem namenjen za implementacijo sistemov na mikroračunalnikih. Jezik je definiral N. Wirth, implementiran je bil leta 1980.

ADA (Ada Avgusta Lovelace - pionirka računalništva) je jezik, katerega izdelavo je naročil DOD (Department of Defense), to je ameriško obrambno ministrstvo. ADA naj bi bil "jezik vseh jezikov", torej naj ni bil zelo široko uporabljan, predvsem pri numeričnih nalogah, sistemskem programiranju in paralelnem izvrševanju programov v realnem času. Pri snovanju jezika je sodelovala množica strokovnjakov, ki so skušali v ADI združiti dobre lastnosti vrste jezikov, od Pascala do MODULE ali Concurrent PASCALa, Euclida ali CLUja. ADA ima zelo močno izražen koncept modulov. Kljub dolgoletnim naporom (od leta 75 dalje) še danes ni delujočega prevajalnika, ki bi implementiral vse opisane lastnosti jezika. Zadnja leta naj se pojavljajo podmnožice ADE, npr. JANUS /18,19/.

2.4. Ocena algoritmičnih jezikov

FORTRAN, COBOL in BASIC so splošno razširjeni jeziki in ne kaže, da bodo v bližnji bodočnosti v večji meri odstopili mesto drugim, čeprav boljšim jezikom. Ta prehod bo kvečjemu postopen. FORTRAN je primeren zlasti za numerične naloge za znanstvene raziskave. COBOL se uporablja predvsem za poslovne aplikacije, BASIC pa za učenje in enostavne poslovne aplikacije. Za vse tri jezike je značilno, da se pojavljajo čedalje novejše izpeljanke teh jezikov, ki v veliki meri privzemajo konstrukte, ki podpirajo

strukturirano programiranje kot v Pascalu. Ti jeziki so precej enakovredni Pascalu, čeprav večina programerjev, ki so dalj časa programirali v katerem koli omenjenih treh jezikov in Pascalu, raiši programira v Pascalu. Pascal nima tako obsežnega nabora standardnih podprogramov kot FORTRAN, vendar je v standardu določena možnost klicanja podprogramov v FORTRANu. COBOL ima pred Pascalom to prednost, da ima datoteke z direktnim dosegom, indeksno sekvencialne datoteke in lepe možnosti formatiranja zapisov (I/O). Glavna prednost BASICA je njegova poenostavljenost in interaktivna usmerjenost. Zato navedenih pomanikljivosti je Pascal brez dodatkov glede na omenjene jezike manj primeren za nekatere poslovne aplikacije in za manj vešče uporabnike.

Prvi jezik, ki je bil narejen kot močnejši brat sorodnega jezika, pa je doživel dokaj žalostno usodo, je bil ALGOL-68. Danes se malo uporablja, čeprav se pojavljajo znaki oživljanja. Ta jezik je nastal kot izpeljanka ALGOLa-60 tako, da je bil bistveno sposobnejši in svobodnejši v izražanju. To pa je vodilo v zelo obsežne in počasne prevajalnike, v nepreglednost in težko čitljivost programov. Odkrivanje napak je bilo otežkočeno in produktivnost je padla.

PL/1 je nastal kot naslednik FORTRANa, COBOLa in ALGOLa 60 tako, da je pobral večino konstruktov iz teh jezikov in še kaj. Je močnejši od Pascala. Računalniški strokovnjaki ga v precejšnji meri ocenjujejo kot stranpot. Težko je v nekaj besedah povedati, kje so snovalci tega jezika pogrešili. Morda je razlog preobširnost jezika, ki se kaže kot obsežna množica med seboj dostikrat tujih konstruktov in prevelika svoboda izražanja. Mogoče so pogrešili implementatorji jezika, ki so napisali zelo obsežen in pogosto neučinkovit prevajalnik. Kljub vsemu se PL/1 precej uporablja.

Podobno situacijo srečujemo tudi pri ADI. ADA ima relativno malo popolnoma novih konceptov glede na ostale algoritmične jezike. Predvsem MODULa-2 ji je močno podobna, sem na lahko prištejemo še USCD Pascal za mikroračunalnike, Pascal PLUS za diskretno simulacijo in Concurrent Pascal za aplikacije v realnem času. Te verzije Pascala so usmerjene na podobna področja kot ADA, standardni Pascal pa lahko obogatimo s knjižnjicami podprogramov v zbirnem jeziku. ADA je vseeno močnejša od omenjenih jezikov, vendar je ta njena moč tudi dvorezen meč, saj je v tako močnih jezikih pogosto neprijetno programirati in se jih je težja naučiti. Poleg tega je prevajalnik vedno obsežen in ga je težko implementirati. Debate o ADI so dokaj ostre in deljene, pravo sodbo pa bo dal šele čas. Pascal je po svojih sposobnostih podmnožica ADE. Podobna razmerja smo srečali npr. že v odnosu COBOL - PL/1, vendar nas zgodovina uči, da močnejši jezik redkokdaj zasenči svoje predhodnike /3,15,20/, še zlasti, kadar nima veliko novih konceptov. Primerna podmnožica ADE bi bila podobna MODULI-2 in bi prav gotovo pomenila korak naprej.

3. Zgodovina in kratek opis Pascala

Pascal je algoritmični programski jezik algolskega tipa. Med neposredne predhodnike Pascala lahko štejemo Algol 60 in Algol W. Pascal je v celoti razvil prof. N. Wirth, tako idejno kot implementacijsko. Prvi prevajalnik je nastal leta 1969 /21/. Izšlo je že nekaj predlogov za standarizacijo jezika, vendar brez končnega rezultata /22/. Naštejmo nekaj dobrih lastnosti Pascala /15,23/:

- majhen in prenosljiv prevajalnik, zato se lahko uporablja na mini in zmogljivejših mikro računalnikih
- dobra, enostavna in razumljiva literatura o Pascalu
- majhen in udoben jezik z malo rezerviranimi besedami, malim številom sintaktičnih in semantičnih pravil in z malo izjemami, zatorej čista in učinkovita krmilna struktura, ki daje občutek zanesljivosti, uporabnik se lahko nauči in obdrži v spominu vse značilnosti jezika
- dobre metode strukturiranja podatkov
- dovolj močni kontrolni in podatkovni konstrukti, da omogočajo udobno in nitro programiranje
- omogoča dobro preglednost in razne stile programiranja (npr. s strukturiranjem ali brez, itd.)

Nameni načrtovalcev Pascala /15,23/:

- velika učinkovitost naj bi bila dosežena s čimveč kontrolami v času prevajanja in samo nujno potrebnimi v času izvajanja
- Pascal naj bi omogočal sistematično učenje programiranja
- dokazali bi radi, da se da jezik z bogatimi kontrolnimi strukturami in fleksibilnimi podatkovnimi tipi implementirati z učinkovitim in majhnim prevajalnikom
- tako jezik kot prevajalnik naj bi bila lahko čitljiva, učinkovita in zanesljiva
- Pascal naj bi bil splošno namenski jezik, torej naj bi omogočal udobno programiranje in aplikacije na čimveč področjih.

4. Zaključna ocena

Pascal (to velja tudi za njemu sorodne jezike, npr. za MODULO-2 ali nekatere inačice ADE) je po svojih lastnostih verjetno eden najboljših predstavnikov algoritmičnih splošno namenskih jezikov. Nealgoritmični jeziki so težje primerljivi s Pascalom, vendar se tudi tu Pascal dokaj dobro odreže. Skoraj na vsakem ožjem področju lahko najdemo jezike, ki so boljši kot Pascal in vsaka značilna skupina jezikov ima svoje načine izražanja, ki so v Pascalu neprijetni. Kljub temu pa lahko velik del za določena problemska okolja značilnih nalog v Pascalu rešimo na skoraj enakovreden način, pa naj primerjamo COBOL pri poslovnih aplikacijah ali SNOBOL pri procesiranju tekstov. Poleg tega je učinkovitost novejših Pascalskih prevajalnikov tako glede dolžine generirane kode programa kot hitrosti izvajanja kode taka kot pri novejših Fortranskih prevajalnikih /24/, torej običajno precej boljša kot pri BASICu, COBOLu ali PL/1 in še bistveno boljša kot pri LISPu, SNOBOLu ali PROLOGu. Velika slabost Pascala ostaja slabo programersko okolje. Ravno to pa je področje, kjer lahko v naslednjih letih pričakujemo največje korake naprej. Programer veliko časa porabi za testiranje programov. Jeziki (bolje rečeno programsko okolje), ki omogočajo ločeno prevajanje, uspešno testiranje (debugger) in imajo interpreter in prevajalnik, ter vse to integrirano in istočasno dostopno, so bistveno boljše orodje za razvijanje programov, kot pa jeziki brez teh lastnosti. Primerjajmo recimo udobnost PROLOGovega debuggerja ali LISPovo programsko okolje ali BASICovo interaktivnost in videli bomo, da ima tu Pascal še veliko neizkoriščenih možnosti. Daljna bodočnost pa bo verjetno prinesla kaj novega, verjetno nekaj izven skupine algoritmičnih jezikov. Mogoče bo ta jezik v marsičem podoben PROLOGu, temeljnemu jeziku japonske pete generacije računalnikov.

5. Literatura

1. A.I.Wasserman,S.Gutz : The Future of Programming, CACM, Vol. 25, Num. 3, str. 196 - 206, maj 1982
2. R.Reinhardt,V.Rajković,I.Lajovic,J.Vrečko : Kriteriji za izbiro programirnega jezika za pouk računalništva na srednji šoli, simpozij INFORMATICA 77, 7 107, 5 str., Bled, 1977
3. C.A.R.Hoare : The Emperor's Old Clothes, CACM, Vol. 24, Num. 2, str. 75 - 83, februar 1981
4. Editorial, ACM SIGPLAN Notices, Vol. 19, Num. 9, september 1982
5. On APL and Productivity, CACM Forum, CACM, Vol. 24, Num. 7, str. 478 - 479, julij 1981
6. J.McCarthy, etc.: LISP 1.5 Programmer's Manual, The M.I.T. Press, 1962
7. R.Herman : The Function of Faster Programming, New Scientist 25, str. 512 - 515, november 1982
8. I.Bratko,M.Gams : PROLOG : osnove in principi strukturiranja podatkov, Informatica 4, str. 40 - 47, 1980
9. R.Kowalski : Algorithm = Logic + Control, CACM, Vol. 22, Num. 7, str. 572 - 595, 1797
10. D.A.Waterman,F.Hayes-Roth : An Overview of Pattern-Directed Inference Systems, Academic Press, 1978
11. M.Gams : Pomen in vloga znanja v sistemih za interakcijo z uporabnikom, magisterska delo, junij 1982
12. W.D.Maurer : The Programmer's Introduction to SNOBOL, American Elsevier Publish. Comp., 1976
13. B.J.MacLennan : Values and Objects in Programming Languages, ACM SIGPLAN Notices, Vol. 17, Num. 12, str. 70 - 80, december 1982
14. T.Rentsch : Object Oriented Programming, ACM SIGPLAN Notices , Vol. 17, Num. 9, str. 51 - 58, september 1982
15. R.Cailliau : How to Avoid Getting SCHLONKED by Pascal, ACM SIGPLAN Notices, Vol. 17, Num. 12, str. 31 - 41, december 1982
16. R.E.Sumner, R.E.Gleaves : Modula-2 -- A Solution to Pascal's Problems, ACM SIGPLAN Notices, Vol. 17, Num. 9, str. 28 - 34, september 1982
17. R.Cailliau : A Letter to Editor, ACM SIGPLAN Notices, Vol. 17, Num. 12, str. 10 - 11, december 1982
18. A.P.Železnikar : Programiranje v ADI I, Informatica 3, str. 10 - 23, 1982
19. A.P.Železnikar : Programiranje v ADI II, Informatica 4, str. 19 - 30, 1982
20. H.F.Leddgard,A.Singer : Scaling Down Ada (Or Towards A Standard Ada Subset), CACM, Vol. 25, Num. 2, str. 121 - 125, februar 1982
21. N.Wirth : The Design of a Pascal Compiler, Software Practice and Experience, 1, str. 309 - 333, 1971
22. Second draft proposal ISO/DP 7185 - Specification for the Computer Programming Language - Pascal, Pascal News, num. 20, december 1980
23. K.Jensen,N.Wirth : Pascal,User Manual and Report, Springer Verlag, 1978
24. Benchmark test na Quicksortu, Special Software Limited, Informatica 3, str. 77, 1982
25. M.Gams,I.Bratko,V.Batagelj,R.Reinhardt,M.Martinec,M.Spegel,P.Tancig : Programski jezik Pascal II (podrobnejša analiza pomanjkljivosti Pascala), v pripravi