

Rm 217 Brouse Copy

NUMBER 22 & 23

Pascal Users Group

Pascal News

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

SEPTEMBER, 1981

Two for one ...



Or one for two?

Return to:

Pascal Users Group
P.O. Box 4406
Allentown, Pa. 18104-4406

Return postage guaranteed
Address Correction requested



ATTN: ROOM 217 BROUSE COPY [81]
UNIV. OF MINNESOTA
UCC : 227EX

MAR 24 1982

POLICY: PASCAL NEWS

(15-Sep-80)

* Pascal News is the official but informal publication of the User's Group.

* Pascal News contains all we (the editors) know about Pascal; we use it as the vehicle to answer all inquiries because our physical energy and resources for answering individual requests are finite. As PUG grows, we unfortunately succumb to the reality of:

1. Having to insist that people who need to know "about Pascal" join PUG and read Pascal News - that is why we spend time to produce it!

2. Refusing to return phone calls or answer letters full of questions - we will pass the questions on to the readership of Pascal News. Please understand what the collective effect of individual inquiries has at the "concentrators" (our phones and mailboxes). We are trying honestly to say: "We cannot promise more that we can do."

* Pascal News is produced 3 or 4 times during a year; usually in March, June, September, and December.

* ALL THE NEWS THAT'S FIT, WE PRINT. Please send material (brevity is a virtue) for Pascal News single-spaced and camera-ready (use dark ribbon and 18.5 cm lines!)

* Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.

* Pascal News is divided into flexible sections:

POLICY - explains the way we do things (ALL-PURPOSE COUPON, etc.)

EDITOR'S CONTRIBUTION - passes along the opinion and point of view of the editor together with changes in the mechanics of PUG operation, etc.

HERE AND THERE WITH PASCAL - presents news from people, conference announcements and reports, new books and articles (including reviews), notices of Pascal in the news, history, membership rosters, etc.

APPLICATIONS - presents and documents source programs written in Pascal for various algorithms, and software tools for a Pascal environment; news of significant applications programs. Also critiques regarding program/algorithm certification, performance, standards conformance, style, output convenience, and general design.

ARTICLES - contains formal, submitted contributions (such as Pascal philosophy, use of Pascal as a teaching tool, use of Pascal at different computer installations, how to promote Pascal, etc.).

OPEN FORUM FOR MEMBERS - contains short, informal correspondence among members which is of interest to the readership of Pascal News.

IMPLEMENTATION NOTES - reports news of Pascal implementations: contacts for maintainers, implementors, distributors, and documentors of various implementations as well as where to send bug reports. Qualitative and quantitative descriptions and comparisons of various implementations are publicized. Sections contain information about Portable Pascals, Pascal Variants, Feature-Implementation Notes, and Machine-Dependent Implementations.

----- ALL-PURPOSE COUPON ----- (15-Dec-81)

Pascal Users Group
P.O. Box 4406
Allentown, Pa. 18104-4406 USA

****Note****

- We will not accept purchase orders.
- Make checks payable to: "Pascal Users Group", drawn on a U.S. bank in U.S. dollars.
- Note the discounts below, for multi-year subscription and renewal.
- The U. S. Postal Service does not forward Pascal News.

- | | | USA | UK | Europe | Aust. |
|--|-------------|-------|------|--------|--------|
| [] Enter me as a new member for: | [] 1 year | \$10. | #6. | DM20. | A\$8. |
| [] Renew my subscription for: | [] 2 years | \$18. | #10. | DM45. | A\$15. |
| | [] 3 years | \$25. | #15. | DM50. | A\$20. |
| [] Send Back Issue(s) | _____ | | | | |
| [] My new address/phone is listed below | | | | | |
| [] Enclosed please find a contribution, idea, article or opinion which is submitted for publication in the <u>Pascal News</u> . | | | | | |
| [] Comments: | _____ | | | | |

ENCLOSED PLEASE FIND:
CHECK no. _____

NAME _____

ADDRESS _____

PHONE _____

COMPUTER _____

DATE _____

JOINING PASCAL USERS GROUP?

Membership is open to anyone: Particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Please enclose the proper prepayment (check payable to "Pascal User's Group"); we will not bill you. Please do not send us purchase orders; we cannot endure the paper work! When you join PUG any time within a year: January 1 to December 31, you will receive all issues of Pascal News for that year. We produce Pascal News as a means toward the end of promoting Pascal and communicating news of events surrounding Pascal to persons interested in Pascal. We are simply interested in the news ourselves and prefer to share it through Pascal News. We desire to minimize paperwork, because we have other work to do.

American Region (North and South America) Join through PUG(USA).

European Region (Europe, North Africa, Western Asia): Join through PUG(EUR) Pascal Users Group, c/o Grado Computer Systems & Software, Weissenburgerstrasse 25, D-8000, Munchen 80, Germany.

United Kingdom Region: join through PUG(UK) : Pascal Users Group, c/o Shetlandtel, Walls, Shetland, ZE2 9PF, United Kingdom.

Australasian Region (Australia, East Asia - incl. India & Japan): PUG(AUS). Pascal Users Group, c/o Arthur Sale, Department of Information Science, University of Tasmania, Box 252C GPO, Hobart, Tasmania 7001, Australia. International telephone: 61-02-202374

RENEWING?

Please renew early (before November) and please write us a line or two to tell us what you are doing with Pascal, and tell us what you think of PUG and Pascal News. Renewing for more than one year saves us time.

ORDERING BACK ISSUES OR EXTRA ISSUES?

Our unusual policy of automatically sending all issues of Pascal News to anyone who joins within a year means that we eliminate many requests for backissues ahead of time, and we don't have to reprint important information in every issue--especially about Pascal implementations!

Issues 1 .. 8 (January, 1974 - May 1977) are out of print.

Issues 9 .. 12, 13 .. 16, & 17 .. 20 are available from PUG(USA) all for \$15.00 a set, and from PUG(AUS) all for \$A15.00 a set.

Extra single copies of new issues (current academic year) are: \$5.00 each - PUG(USA); and \$A5.00 each - PUG(AUS).

SENDING MATERIAL FOR PUBLICATION?

Your experiences with Pascal (teaching and otherwise), ideas, letters, opinions, notices, news, articles, conference announcements, reports, implementation information, applications, etc. are welcome. Please send material single-spaced and in camera-ready (use a dark ribbon and lines 18.5 cm. wide) form.

All letters will be printed unless they contain a request to the contrary.

	Pascal News #22 & 23	September 1981	Index
0	POLICY, COUPONS, INDEX, ETC.		
1	EDITORS CONTRIBUTION		
3	HERE AND THERE WITH Pascal		
3	Summary of Implementations (for PN 15..18)		(G. Marshall)
4	APPLICATIONS		
4	The FMI Compiler (code)		A. Tanenbaum
38	Options -- Control Statement Option Settings		S. Leonard
39	Treeprint -- Prints Trees on a Character Printer		Freed & Carosso
44	Compress & Recall -- Text compression using Huffman codes		T. Stone
50	ARTICLES		
50	"The Performance of three CP/M based Translators"		Johnson & Sidebottom
54	"A Geographer Teaches Pascal -- Reflections on the Experience"		J. Pitzl
56	"An Extension That Solves Four Problems"		J. Yavner
61	OPEN FORUM FOR MEMBERS		
68	IMPLEMENTATION NOTES		
81	ONE PURPOSE COUPON, POLICY		

APPLICATION FOR LICENSE TO USE VALIDATION SUITE FOR PASCAL

Name and address of requestor: _____
(Company name if requestor is a company): _____
Phone Number: _____
Name and address to which information should be addressed (write "as above" if the same) _____
Signature of requestor: _____
Date: _____

In making this application, which should be signed by a responsible person in the case of a company, the requestor agrees that:

- a) The Validation Suite is recognized as being the copyrighted, proprietary property of R. A. Freak and A. H. J. Sale, and
- b) The requestor will not distribute or otherwise make available machine-readable copies of the Validation Suite, modified or unmodified, to any third party without written permission of the copyright holders.

In return, the copyright holders grant full permission to use the programs and documentation contained in the Validation Suite for the purpose of compiler validation, acceptance tests, benchmarking, preparation of comparative reports and similar purposes, and to make available the listings of the results of compilation and execution of the programs to third parties in the course of the above activities. In such documents, reference shall be made to the original copyright notice and its source.

Distribution Charge: \$50.00

Make checks payable to ANPA/RI in US dollars drawn on a US bank.
Remittance must accompany application.

Source Code Delivery Medium Specification:

- 800 bpi, 9-track, NRZI, odd parity, 600' magnetic tape
 1600 bpi, 9-track, PE, odd parity, 600' magnetic tape

ANSI-STANDARD

a) Select Character Code Set:

- ASCII EBCDIC

b) Each logical record is an 80 character card image. Select block size in logical records per block.

- 40 20 10

Special DEC System Alternates:

- RSX-IAS PIP Format (requires ANSI MAGtape RSX SYSGEN)
 DOS-RSTS FLX Format

Mail Request to:
ANPA/RI
P.O. Box 598
Easton, Pa. 18042
USA
Attn: R. J. Cichelli

Office Use Only

Signed _____
Date _____

Richard J. Cichelli
On behalf of A.H.J. Sale and R.A.Freak

Editor's Contribution

GOOFED AGAIN

Yes as all you loyal Pennsylvanians have noticed in the last issue of PN we managed to mess up the zip code of Allentown PA, and of course the USPS has come down on us like a ton of bricks! Please note that the zip is 18014 not 18170. It has been corrected in the new APC.

THE NEW APC

Speaking of the new APC we have simplified it some more, and added current prices for the UK and Europe, and have modified the reverse side of the coupon to reflect the new foreign editors and their current addresses.

THE LATEST EUROPEAN SOLUTION

Speaking of the European editors, we have two new ones! One for the UK, and one for the Continent. Nick Hushes will be handling all business for Britain, and Hellmut Heher will be in charge of the European Region. Please see the APC for their addresses.

ON CALLING

Please restrict yourself to written correspondence when dealing with PUG. This is strictly a scholarly function. None of the editors (including myself) gets paid. All have a real job that pays their bills, and they owe their office hours to their employer. All PUG work is done on their own time. So please write to the appropriate regional editor. It leaves a documentary trail that can be followed and handled as fast as we can. Honest!

COMBINED ISSUE

This is of course a combined issue. We are doing this to catch up and to beat the postal system and their high rates. If this upsets anyone we are sorry. We are doing our best.

ON BEING THE EDITOR

Anyone who is interested in being the new editor of PN should write to me at the main address (APC).

STANDARDS

Good news from the standard front! 7185.1 was approved by the international committee. More next issue from Jim Miner the Standards Editor.

Here and There With Pascal

Summary of Implementations

THIS ISSUE

The highlight of this issue is the long awaited (from last issue at least!) of Andrew Tanenbaum's EM1 compiler. I think it is really great. Tell us what you think! In the Here and There section Gress Marshall has summarized the past few issues (15 .. 19) implementation notes. Thanx. In addition to the EM1 compiler, the Applications section includes an improved version of the subroutine "options", as well as a tree printing routine, and a set of routines to compress and expand text using Huffman codes. Good work! And finally the articles section has some fine contributions. Many people have asked (on the phone ... see above) about how the various CP/M compilers stack up. Now we have an answer. Also there is an article of the experiences of a novice teaching Pascal. From a geography teacher no less! And finally a probins article by Jonathan Ymuner concerning problems with Pascal and some proposals for their solution.

Hope you like it.

Rick

ALL	#15:101	Pascal I (Derived from Pascal S)	
BESM-6	#15:107		
Burroughs B5700	#15:107		
Burroughs B6700/B7700 (MCP)	#19:113		
CDC 6000	#19:115		
CDC 6000	#15:108		
Cyber 70 and 170	#15:108		
DEC PDP-11	#19:115	UCSD Pascal	
DEC PDP-11	#15:111		
DEC PDP-11	#15:112	UCSD Pascal	
DEC PDP-11	#15:124		
DEC PDP-11 (RSTS)	#15:100	Pascal S	
DEC PDP-11 (RSX-11M/IAS)	#17:86		
DEC PDP-11 (RSX-11M/RT-11)	#15:101	Concurrent Pascal	
DEC PDP-11 (Unix)	#15:111		
DEC PDP-11 (Unix)	#15:100	Pascal E	
DEC PDP-11 (Unix)	#15:103	Modula	
DEC PDP-15	#15:124		
DEC VAX	#17:89		
DEC VAX (Unix)	#19:115		
DG Eclipse	#17:106		
DG Eclipse (AOS)	#15:110	RDOS, DOS)	
DG Eclipse (AOS)	#15:109		
DG Eclipse (RDOS)	#15:108		
DG Nova (AOS)	#15:110	RDOS, DOS)	
Digico Micro 16E	#15:113		
Facom 230-45S	#15:112		
General Electric GEC4082	#15:113		
Golem B (GOBOS)	#17:104		
HP 1000	#19:116		
Honeywell 6000 (GCOS III)	#15:113		
Honeywell Level 6	#15:113		
IBM 3033	#19:120		
IBM 360/370	#15:114		
IBM 360/370	#15:115		
IBM 370	#17:104		
IBM 370	#19:117		
IBM 370	#15:124		
IBM 370	#17:102		
IBM 370/303x/43xx	#19:117		
IBM Series 1	#19:116		
IBM Series 1	#15:114		
ICL 1900	#15:116		
Intel 8080/8085	#15:119		
Intel 8080/8085	#15:118		
Intel 8080/8085	#15:119		
Intel 8080/8085	#17:102		
Intel 8080/8085	#15:117		
Intel 8080/8085 (CP/M)	#17:105		
Intel 8080/8085 (TRS-80)	#15:100		
Intel 8080/8085 (Northstar)	#15:100		
Intel 8086	#15:119		
Intel 8086	#15:103		
MOS Tech 6502 (Apple)	#15:107		
Modcomp II and IV	#15:120		
		Motorola 6800	#15:120
		Motorola 6800	#19:120
		Motorola 6800	#19:121
		Motorola 6800	#17:102
		Motorola 6800 (Flex)	#15:123
		Motorola 68000	#19:121
		Motorola 6809	#15:103
		Motorola 6809 (MDOS09)	#17:102
		Nord 10 and 100 (Sintran III)	#15:121
		Perkin-Elmer 3220	#15:122
		Perkin-Elmer 7/16	#15:121
		RCA 1802	#17:103
		RCA 1802	#15:122
		Siemens 7.748	#15:124
		Sperry-Univac V77	#15:124
		Texas Instruments 990	#17:101
		Texas Instruments 9900	#15:124
		Zilog Z-80	#15:124
		Zilog Z-80	#19:123
		Zilog Z-80	#15:124
		Zilog Z-80	#17:88
		Zilog Z-80	#17:104
		Zilog Z-80 (CP/M)	#17:103
		Zilog Z-80 (TRS-80)	#15:124
		Zilog Z-80 (TRS-80)	#19:124
		Zilog Z80	#15:118
		Zilog Z80	#15:119
		Zilog Z8000	#15:119

Applications

EM1 COMPILER

```
1 #include ".../local.h"
2 #include ".../em1.h"
3
4 (c) copyright 1980 by the Vrije Universiteit, Amsterdam, The Netherlands. Explicit permission is hereby granted to universities to use or duplicate this program for educational or research purposes. All other use or duplication by universities, and all use or duplication by other organizations is expressly prohibited unless written permission has been obtained from the Vrije Universiteit. Requests for such permissions may be sent to
5
6 Dr. Andrew S. Tanenbaum
7 Wiskundig Seminarium
8 Vrije Universiteit
9 Postbox 7161
10 1007 MC Amsterdam
11 The Netherlands
12
13 Organizations wishing to modify part of this software for subsequent sale must explicitly apply for permission. The exact arrangements will be worked out on a case by case basis, but at a minimum will require the organization to include the following notice in all software and documentation based on our work:
14
15 This product is based on the Pascal system developed by Andrew S. Tanenbaum, Johan V. Stevenson and Hans van Steyvers of the Vrije Universiteit, Amsterdam, The Netherlands.
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

```
57 f: size of reals in words (2)
58 i: controls the number of bits in integer sets (16)
59 l: insert code to keep track of source lines (-)
60 o: optimize (+)
61 p: size of pointers in words (1)
62 r: check subranges (+)
63 s: accept only standard pascal programs (-)
64 t: trace procedure entry and exit (-)
65 u: treat '-' as letter (-)
66
67 [.....]
68 #ifdef STANDARD
69 label 9999;
70 #endif
71
72 const
73
74 (powers of two)
75 t1 = 128;
76 t2 = 255;
77 t3 = 256;
78 t4 = 16384;
79 t5 = 32767;
80
81 (EM-1 sizes)
82 bytebits = 8;
83 wordbits = 16;
84 wmb = 15; (wordbits-1)
85 minint = -t5w;
86 maxint = t5w;
87 maxintstring = '0000032767';
88 maxlongstring = '2147483647';
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
```

```
113 p-option. Floating point numbers in EM-1 currently have size 4, but this might change in the future to 8. The default can be overwritten by the f-option. The routines involved with alignment are 'even', 'address' and 'arraysize'.
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
```

```
169 type
170 (scalar types)
171 symbol (comma,semicolon,colon,colon2,notary,bracket,ident,
172 intout,charact,realout,longout,stringout,alicut,minary,
173 plusy,percent,arrow,arrayy,recordy,setsy,filesy,
174 packedy,progy,labely,consty,typesy,varsy,procsy,
175 funcy,beginy,gotosy,ifsy,whiley,repeaty,foray,
176 withy,casemy,becomes,staray,divy,mody,alseny,
177 anday,oray,exory,notay,asy,asy,asy,
178 leay,iny,andy,elasy,untily,ofay,dasy,
179 downtoy,coy,thensy,rbrack,rparent,period
180 ); (the order is important)
181
182 chartype (lower,upper,digit,layout,tabch,
183 quotech,quotech,odotech,perioch,leasch,
184 greaterch,lparentch,lbracoch,
185 rparentch,rbrackch,rbrackch,comach,semich,arowch,
186 plusch,minch,alch,star,equal,
187 ); (also symbols)
188
189 others
190 );
191
192 standpf (pread,preadln,write,partials,pput,pgot,
193 preat,prewrite,pnes,dispose,ppack,punpack,
194 pmack,prelease,ppage,phalt,
195 ); (all procedures)
196
197 eof,feofn,fabs,fsg,ford,forb,fpred,fauc,fodd,
198 fsumo,fround,fsin,foos,fexp,fsgt,fin,fiactan
199 ); (all functions)
200
201 ); (the order is important)
202
203 libnames (IN, OPL, CLS, WPL,
204 OFN, GETX, END, RDC, END, NML, NLS
205 ); (see input files)
206
207 CRF, FUF, WFI, NSI, WAC, WSC, WRS, WSS, WNB,
208 WNB, WVR, WVR, WML, WSL, WNF, WRI, WSI, WLM, PAC,
209 ); (see output files, order important)
210
211 ABS, RND, SIN, COS, EXP, SQRT, LOG, ATN
212 ); (floating point)
213
214 ABI, ABL, BCP, BVS, NEWI, SAV, EST, IBI, IBI,
215 ASS, OTO, PAC, WFP, DIS, ARZ, MDI, MDL
216 ); (miscellaneous)
217
218
219
220
221
222
223
224
```

PASCAL NEWS #22 & #23

SEPTEMBER, 1981

Page 4

PASCAL NEWS #22 & #23

SEPTEMBER, 1981

Page 5

225 rrange= 0..rval; 281 end;

226 bytes 0..lbit;

228 (pointer types)

229 sp= "structure; 284

230 ip= "identifier; 285 fname:ip; (one deeper)

231 l= "labl; 286 (first name: root of tree)

232 bps= "blockinfo; 287 case occur:where of 287

233 sp= "nameinfo; 288 blok:(); 288

234 289 rec: (); 289

235 (set types)

236 set= set of symbol; 290 end;

237 setofid= set of idclass; 291 blockinfo:record (all info of the current procedure)

238 format= set of structform; 292 set:pp; (pointer to blockinfo of surrounding proc)

239 sflagset= set of structflag; 293 lcin:integer; (data location counter (from begin of proc))

240 iflagset= set of idclass; 294 libno:integer; (number of last local label)

241 295 forwardcount:integer; (number of not yet specified forward procs)

242 (array types)

243 alpha= packed array[irange] of char; 297 lchain:ip; (first label: header of chain)

244 fctype= packed array[frange] of char; 298 end;

246 (record types)

247 error:record 300 structure:record

248 error:integer; (error number) 301 size:integer; (size of structure in bytes)

249 error:integer; (identifier parameter if required) 302 sflag:flagset; (flag bits)

250 num:integer; (numeric parameter if required) 303 case form:structform of 303

251 colno:integer; (column number) 304 scalar: (scalno:integer; (number of range descriptor)

252 line:integer; (line number) 305 fconst:ip; (names of constants)

253 lin:integer; (relative to start of (included) file) 306 ;)

254 orig:integer; (idem, but before preprocessing) 307 subrange:(ain,am:integer; (lower and upper bound)

255 fnam:ftype; (source file name) 308 rangetype:ip; (type of bounds)

256 end; 309 subno:integer (number of subr descriptor)

258 position:record (the addr info of certain variable) 310 ;)

259 ad:integer; (for locals it is the byte offset) 311 pointer: (altype:ap; (type of pointed object)

260 lv:integer; (the level of the beast) 312 power: (alast:ap; (type of set elements)

261 #finder SEGMENTS 313 files: (fitype:ap; (type of file elements)

262 sg:sgrange (only relevant for globals (lv=0)) 314 arrays:array; (type of array elements)

263 #endif 315 ;)

264 end; 316 inttype:ap; (type of array index)

266 (records of type attr are used to remember qualities of 317 arpos:position (position of array descriptor)

267 expression parts to delay the loading of them. 318 ;)

268 Reasons to delay the loading of one word constants: 319 records: (fstfid:ip; (points to first field)

269 - bound checking 320 tagap:ap; (points to tag if present)

270 - set building 321 ;)

271 Reasons to delay the loading of direct accessible objects: 322 variant: (varval:integer; (tag value for this variant)

272 - efficient handling of read/write 323 mtrvar:ap; (next equilevel variant)

273 - efficient handling of the with statement. 324 subtag:ap; (points to tag for sub-case)

274 ;)

275 attr:record 325 ;)

276 exp: (type of expression) 326 tag: (fstvar:ap; (first variant of case)

277 packbit:boolean; (access method) 327 tfid:ap; (type of tag)

278 attrkind; (access method) 328 end;

279 pos:position; (eg, lv and ad) 329

280 (if almost then the value is stored in ad) 330

331 identifier:record 331

332 idtype:ap; (type of identifier)

333 name:alpha; (name of identifier)

334 link:linkip; (see enterid,searchid)

335 next:ip; (used to make several chains)

336 iflag:iflagset; (several flag bits)

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

367

368

369

370

371

372

373

374

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

337 case klass:idclass of

338 types: ();

339 konst: (value:integer); (for integers the value is

340 computed and stored in this field.

341 For strings and reals an assembler constant is

342 defined labeled '.1', '.2', ...

343 This '.1' number is then stored in value.

344 For reals value may be negated to indicate that

345 the opposite of the assembler constant is needed.)

346 vars: (vpos:position); (position of var)

347 field: (ffoffset:integer); (offset to begin of record)

348 oarrbnd: (); (idtype points to array)

349 proc,func: (name pf:kindofpf of (identification)

350 standard:(key:standpf); (identification)

351 formal,actual,forward,extra: (lv gives declaration level.

352 (pfpos:position); (lv gives instruction segment of this proc and

353 ad is relevant for formal pf's and for

354 functions (no conflict)).

355 for functions: ad is the result address.

356 for formal pf's: ad is the address of the

357 descriptor)

358 pfno:integer; (unique pf number)

359 parhead:ip; (head of parameter list)

360 head:integer (1s when heading summed)

361 ;)

362)

363 end;

367 labl:record (chain of labels)

368 next:ip;

369 seen:boolean;

370 labval:integer; (label number given by the programmer)

371 labname:integer; (label number given by the compiler)

372 labdib:integer (save some only locally used, otherwise dibno of label information)

373 ;)

374 end;

376 var (the most frequent used externals are declared first)

377 sy:symbol; (last symbol)

378 sattr; (type,access method,position,value of expr)

379 (returned by inapp)

380 chchar: (last character)

381 chtype: (type of ch, used by inapp)

382 val:integer; (if last symbol is an constant)

383 ix:integer; (string length)

384 col:boolean; (true if current ch replaces a newline)

385 newstrng:boolean; (true for strings in " ")

386 idalpha; (if last symbol is an identifier)

387 (some counters)

388 line:integer; (line number on code file (1..n))

389 dibno:integer; (number of last global number)

390 lmax:integer; (deepest level of nesting of ls)

391 level:integer; (current static level)

393 ptrsize:integer;

394 realize:integer;

395 rbase:integer; (file header size)

396 arg:integer; (index in arg)

397 lastpfno:integer; (unique pf number counter)

398 oopt:integer; (C-type strings allowed if on)

399 dopt:integer; (longs allowed if on)

400 lop:integer; (number of bits in sets with base integer)

401 sop:integer; (standard option)

402 (pointers pointing to standard types)

403 realptr, intptr, textptr, emptyptr, boolptr:ip;

404 charptr, nilptr, stringptr, longptr:ip;

405 (flags)

406 give:line:boolean; (give source line number at next statement)

407 including:boolean; (no LHM's for included code)

408 eof:expected:boolean; (quit without error if true (nextch))

409 main:boolean; (complete programme or a module)

410 inttypedec:boolean; (true if nested in typedefinition)

411 flused:boolean; (true if floating point instructions are used)

412 seconddot:boolean; (indicates the second dot of '..')

413 (pointers)

414 fptr:ip; (head of chain of forward reference pointers)

415 progid:ip; (program identifier)

416 curproc:ip; (current proc/func ip (see casestatement))

417 top:ip; (pointer to the most recent name space)

418 lasttag:ip; (pointer to nameinfo of last searched ident)

419 (records)

420 bblockinfo; (all info to be stored at pfdeclaration)

421 error; (all info required for error messages)

422 fattr; (fstr for current file name)

423 (arrays)

424 source:ftype; (name of pascal source file)

425 strbuf:array[1..max] of char;

426 lop:array[boolean] of ip;

427 (flags:standard input, true:standard output)

428 rv:array[rrange] of alpha; (reserved words)

429 (reserved words)

430 frv:array[0..idmax] of integer; (indices in rv)

431 ;)

432 ray:array[rrange] of symbol; (symbol for reserved words)

433 ;)

434 ca:array[char] of chartype; (char type of a character)

435 ;)

436 ca:array[rrange] of symbol; (symbol for single character symbols)

437 ;)

438 lms:array[lidmax] of char; (mnemonics of pascal library routines)

439 ;)

440 opt:array['a'..'z'] of integer;

441 forwardopt:array['a'..'z'] of boolean;

442 ;)

443 undefip:array[idclass] of ip;

444 (used in searchid)

445 arg:array[0..maxarg] of

446 record name:alpha; ad:integer end;

447 (save here the external heading names)

448 (files)

449