TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Pascal en portabiliteit : een vergelijking van een aantal Pascalcompilers met de standaard

Citation for published version (APA):
Woude, van der, M., & Rietjens, M. J. M. (1986). *Pascal en portabiliteit : een vergelijking van een aantal Pascalcompilers met de standaard*. (Computing centre note; Vol. 29). Technische Universiteit Eindhoven.

Document status and date:
Gepubliceerd: 01/01/1986

Document Version:
Uitgevers PDF, ook bekend als Version of Record

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 16. Nov. 2023

THE-RC 65554

Eindhoven University of Technology
Computing Centre Note 29

Pascal en portabiliteit.
Een vergelijking van een aantal Pascal-compilers met de standaard.

Ir. M. van der Woude, M.J.M. Rietjens

maart 1986.

Inhoudsopgave: bladzijde

1. Inleiding.

Pascal is een hogere programmeertaal, ontwikkeld door N. Wirth. De eerste veel gebruikte definitie van Pascal werd gepubliceerd in het 'User Manual and Report' [1]. Dit Report is, hoewel goed leesbaar, niet geschikt als definitie van een standaard voor Pascal. Later werd op basis van dit Report een Pascal-standaard gepubliceerd, de ISO-standaard 7185 [2].

Deze ISO-standaard bevat 2 levels, level 0 en level 1. Level 0 komt inhoudelijk overeen met de definitie van Pascal, zoals die oorspronkelijk door Wirth geschreven is. Door ANSI is ISO-7185 level 0 als standaard overgenomen. ISO-7185 bevat als uitbreiding ten opzichte van level 0 een definitie van array parameters met variabele lengte (conformant array parameters). Het is waarschijnlijk dat in de uiteindelijke ANSI-standaard deze definitie niet zal worden overgenomen.

Tot de niet in ISO-7185 level 0 vastgelegde veel gebruikte faciliteiten behoren:

- OTHERWISE in het CASE-statement (soms ook ELSE).
- Diverse file handling procedures zoals close, open en procedures voor exceptie-afhandeling bij I/O-operaties.
- Random access file procedures zoals seek.
- Array parameters met variabele lengte.
- Separaat compileerbare modules, c.q. libraries.
- String handling procedures en functies.
- Functies of procedures die elapsed tijd, processor-tijd en datum aangeven.
- Default widths bij het schrijven van expressies van het type real, integer en boolean.
- Compiler-opties, zoals (het aantal levels) include.

De bovengenoemde onderwerpen zijn vaak voor diverse compilers verschillend geïmplementeerd. Het gevolg is dat Pascal-programma's meestal slecht overdraagbaar zijn.

In de volgende paragrafen wordt een aantal Pascal-compilers in het kort en dus onvolledig besproken en met de standaard en elkaar vergeleken.

2. Pascal op het Burroughs Large System.

De Burroughs Pascal Compiler staat beschreven in het user manual [3]. Deze compiler bevat een nagenoeg volledige implementatie van de ISO-level 0 standaard. De in level 1 voorgestelde faciliteit wat betreft 'conformant arrays' is niet geïmplementeerd. In plaats daarvan zijn 'array schemata' gekomen die waarschijnlijk in de uiteindelijke ANSI-standaard komen. De voornaamste uitbreidingen ten opzichte van de standaard zijn:

- Underscore (_) als karakter in identifiers.
- OTHERWISE in CASE-statement.
- Extra file handling procedures.
- Random access file I/O.
- Gebruik van separaat compileerbare modules in de vorm van libraries.
- String-procedures en variabele lengte string type.
- Tijdfuncties.
- Include nesting tot 5 niveaus diep.

Voor een volledig overzicht van uitbreidingen en systeemafhankelijkheden wordt verwezen naar [3]. Dit manual geeft een duidelijke definitie van Burroughs Pascal, maar het is niet geschikt als leerboek. Duidelijk is aangegeven waar Burroughs Pascal afwijkt van de standaard. De compiler produceert in één stap een executeerbare codefile. In [10] wordt het gebruik van Pascal op de B7900 toegelicht.

3. Pascal op de VAX onder VMS.

De VAX/VMS Pascal-compiler staat beschreven in [4]. In dit manual staat geen zogenaamde compliance statement, zodat niet eenvoudig is na te gaan in hoeverre de standaard volledig is geïmplementeerd. VMS-Pascal bevat wel conformant arrays, zoals in level 2 van de ISO-standaard is voorgesteld. De voornaamste uitbreidingen van VAX/VMS Pascal ten opzichte van de standaard zijn:

- Underscore (_) en $ als karakter in identifiers.
- OTHERWISE in CASE-statement.
- Extra file handling procedures.
- Random access file I/O.
- Separaat compileerbare modules.
- String-functies en variabele lengte string type.
- Tijdfuncties.

Voor een volledig overzicht van uitbreidingen en systeemafhankelijkheden wordt verwezen naar [4]. Dit manual geeft een duidelijke definitie van VAX/VMS Pascal, maar het is niet bedoeld als leerboek. De uitbreidingen ten opzichte van de standaard zijn in een afwijkende kleur gedrukt. De VAX/VMS Pascal compiler produceert een object-file waarna door de linker een executeerbare file wordt gemaakt.

4. Pascal onder ULTRIX-32.

Op het ULTRIX-32-systeem (VAX-UNIX) is Berkeley Pascal aanwezig [5]. On line informatie is te verkrijgen met het commando "man pc" of "man pi".

De beschrijving heeft de vorm van een handleiding met een aantal voorbeelden. Voor taaldefinities wordt in [5] verwezen naar Wirth's User Manual and Report [1]. Deze compiler laat dus geen variabele lengte array parameters toe. Verder bevat de beschrijving een overzicht van systeemafhankelijkheden, beperkingen en uitbreidingen. De voornaamste beperkingen ten opzichte van de standaard zijn:

- Kleine en grote letters zijn buiten strings niet synoniem, zoals de standaard voorschrijft. Reserved words moeten bij Berkeley Pascal in kleine letters.
- Array subscripts moeten in absolute waarde kleiner zijn dan 32768.
- Variabelen (records en arrays) mogen maximaal 64 Kbyte groot zijn.
- Geen controle op arithmetische overflow.

Verder is er een aantal uitbreidingen, echter niet zoveel als bij Burroughs en VMS-Pascal.

- Extra file handling procedures.
- Separaat compileerbare modules.
- Tijdfuncties.
- Include nesting tot 10 niveaus diep.

In tegenstelling tot bij Burroughs en VMS-Pascal heeft deze compiler niet de faciliteiten voor:

- Underscore (_) als karakter in identifiers.
- OTHERWISE in CASE-statements.
- Random access file I/O.
- String-procedures en strings van variabele lengte.

Voor de uitbreidingen en restricties wordt verder verwezen naar [5]. Dit manual is niet te beschouwen als een eenduidige definitie van Berkeley Pascal, noch als leerboek. De compiler produceert in één stap een executeerbare file.

5. Pascal op de PC.

Er is een viertal Pascal-versies voor de IBM/PC en compatibles bekend, te weten: MS-Pascal, Pascal/MT+, TURBO-Pascal en UCSD-Pascal. Hoewel UCSD-Pascal zeer gebruikersvriendelijk is, wordt het hier niet verder besproken omdat UCSD-Pascal voorzover bekend niet onder MS-DOS draait en omdat de beschikbaarheid van UCSD-Pascal in de toekomst onzeker is.
De overige compilers zijn alle beschikbaar onder MS-DOS en zullen in het kort besproken worden.

5.1. TURBO-Pascal.

TURBO-Pascal is een door Borland International Inc. geleverd pakket [7]. Het bevat behalve een op ISO-standaard level 0 gebaseerde Pascal-compiler ook een op Wordstar gebaseerde editor die in het pakket geïntegreerd is.
Het pakket neemt weinig ruimte in op diskette, het is snel en levert snelle en compacte code af. Er is echter ook een aantal beperkingen ten aanzien van de standaard waarvan de voornaamste zijn:

- Ontbreken van de standaard-procedures get en put.
- Functies en procedures mogen niet als parameter van een functie of procedure optreden.
- Alleen geschikt voor kleine programma's (< 64 Kbyte).
- Geen variabele arrays als parameter.
- 16 bits integers, geen arithmetische overflow-detectie.
- Milde fout aanpak, bijvoorbeeld integer overflow wordt niet gedetecteerd.

Daarnaast bevat TURBO-Pascal een aantal uitbreidingen:

- Underscore (_) als karakter identifiers.
- ELSE in CASE-statement.
- Extra file handling procedures.
- Random access file I/O.
- String-procedures en variabele lengte strings.
- Include nesting 1 niveau diep.
- Uitgebreide interface met MS-DOS inclusief PC-graphics.

In tegenstelling tot bij Burroughs en VMS-Pascal is er bij TURBO-Pascal:

- Geen separaat compileerbare modules; er is echter wel overlay en program chaining mogelijk.

- Geen tijdfuncties. Er bestaat echter wel de mogelijkheid om deze te maken in TURBO-Pascal.

Het TURBO-Pascal-systeem is in de eerste plaats bestemd voor interactief gebruik. Dat wil zeggen dat een compileer-opdracht gegeven wordt, nadat het TURBO-systeem en de te compileren source in het geheugen is geladen. Door bepaalde opties te zetten wordt een codefile (.COM of .CHN file) op disk gecreëerd, default wordt de code alleen in het primaire geheugen geschreven, wat erg snel gaat (in één opdracht).
Het TURBO-Pascal-system versie 3.01 kan inclusief bijgeleverde voorbeelden en hulp-files ruimschoots op een diskette van 360 Kbyte (double-sided, double density). Voor het minimale gebruik behoeft alleen de TURBO.COM-file op de actieve drive te staan. Deze is 40 Kbyte groot.

5.2. Microsoft Pascal.

Microsoft Pascal versie 3.20 is gebaseerd op de ISO-7185 level 0 standaard met een aantal uitbreidingen. Het compileren met MS-Pascal gebeurt in 2 of 3 stappen (PAS1, PAS2 en soms PAS3), gevolgd door een link stap. Aan de compiler is, in tegenstelling tot TURBO-Pascal, geen editor verbonden. Het pakket neemt in zijn totaliteit meer dan een floppy van 360 Kbyte in beslag. Het compileerproces gaat gepaard met veel disk I/O en is daarom aanzienlijk trager dan bij TURBO-Pascal. Het pakket past aanzienlijk beter bij de standaard dan TURBO-Pascal. Als voornaamste beperking geldt hier dat het pakket in feite ook slechts geschikt is voor kleine programmamodules (< 64 Kbyte), ook al wordt in het manual [6] een aantal wegen besproken om boven deze grens uit te gaan. Verder gelden nog als beperkingen:

- Functies en procedures niet als parameters overdraagbaar naar andere modules.
- 16 bits integers. Er is echter een type INTEGER4 van 32 bits.
- Milde foute aanpak, bijvoorbeeld integer overflow wordt alleen gedetecteerd als de optie $mathck aan staat.

Als voornaamste uitbreidingen noemen we:

- Underscore (_) mag in identifiers.
- OTHERWISE in CASE-statements.
- Extra file handling procedures.
- Random access file I/O.
- Separaat compileerbare modules,overlay mogelijk.
- Array parameters met variabele dimensie in de vorm van super-arrays.
- String-procedures en variabele lengte strings.
- Tijdroutines.
- Include nesting 1 niveau diep.
- Behoorlijke interface naar MS-DOS en procedures voor bit en byte manipulatie. In tegenstelling tot bij TURBO-Pascal is program chaining niet mogelijk.

Het werken met MS-Pascal.

Het creëren van een executie-file met behulp van deze compiler bestaat uit een aantal stappen (commando's), te weten:

- PAS1

  PAS1 genereert, indien er geen fouten gedetecteerd worden, twee intermediate files die als invoer dienen voor PAS2.

- PAS2

  Met behulp van dit commando wordt de object-file aangemaakt. Tevens worden de in PAS1 gegenereerde intermediate files verwijderd.

- PAS3

  Deze stap is alleen van toepassing als in PAS1 aangegeven wordt dat er een code listing file gewenst is. In dit geval genereert PAS2 intermediate files die als invoer dienen voor PAS3.

- LINK

  Met behulp van dit commando wordt de executie-file gemaakt. Aan het programma kunnen libraries worden gekoppeld. De standaard libraries die nodig zijn om tot een executie-file te komen, zijn PASCAL.LIB en MATH.LIB.

  Om het programma uit te voeren hoeft slechts de naam van de executie-file opgegeven te worden.

De complete MS-Pascal compiler versie 3.20 past niet op een floppy van 360 Kbyte. Indien PAS3 wordt weggelaten, kan echter een voor eenvoudig gebruik voldoende configuratie worden gemaakt die wel op een floppy past. Een compilatie gevolgd door een run kan met behulp van de volgende commando-file worden gedaan:

file: A:CRPAS.BAT

inhoud:

```
B: PAS1 %1;
B: PAS2
B: LINK %1,, NUL.MAP, B:;
%1
```

aanroep:

```
CRPAS PROG
```

aannames:

```
MS-Pascal op driver B, actieve drive: A
Te compileren programma A:PROG.PAS
Executie-file wordt     A:PROG.EXE
```

5.3. Pascal/MT+.

Pascal/MT+ versie 3.3 [9] van de firma Digital Research is volgens het manual een volledige implementatie van ISO-standaard 7185, inclusief conformant arrays. Er zijn echter de volgende beperkingen:

- Identifiers beperkt tot 8 significante karakters.
- Beperkte runtime foutcontrole, geen overflow-indicatie.
- Alleen geschikt voor kleine programma-eenheden (< 64 Kbyte), er is echter wel overlay en chaining mogelijk.
- 16 bits integers. Er zijn echter 32 bits integers beschikbaar als type LONGINT.
- Functies en procedures niet als parameter overdraagbaar naar andere modules.

Als voornaamste uitbreidingen noemen we:

- Underscore (_) mag in een identifier.
- ELSE in CASE-statement.
- Extra file handling procedures.
- Random access file I/O.
- Separaat compileerbare modules.
- Conformant arrays.
- String-procedures en variabele lengte strings.

- Include nesting 1 niveau diep.
- Behoorlijke interface naar MS-DOS en procedures voor bit- en byte-manipulatie.

In tegenstelling tot de meeste andere compilers biedt PASCAL/MT+ geen tijdfuncties.

Pascal/MT+ versie 3.3 is een one-pas compiler. Na compilatie moeten aan het programma libraries gelinkt worden met behulp van de linker. Deze levert een executeerbare file af. Met behulp van de volgende commando-file kan een volledige compile run-stap worden gedaan:

file: B:MTCR.BAT

inhoud:

```
MT86 %1
LINKMT %1,PASLIB/S
%1
```

aanroep:

```
B: MTCR PROG
```

aannames:

```
MT+ Pascal op actieve drive A
Te compileren programma B: PROG.PAS
Executie-file wordt     B: PROG.EXE
```

De volledige Pascal/MT+ compiler beslaat ongeveer 300 Kbyte op diskette. Er zit geen editor bij zoals bij TURBO-Pascal.

5.4. Vergelijking TURBO-Pascal, MS-Pascal en Pascal/MT+.

In de volgende tabel is een vergelijking gemaakt van deze drie compilers voor het programma ERATOS.PAS, dat een implementatie is van de "zeef van Erathosthenes". (Zie ook [11] voor vergelijking met andere compilers.)

ERATOS.PAS:

```
PROGRAM Erathosthenes_sieve;
        const size = 8190;
        var flags:array [0..size] of boolean;
            i,prime,k,count,iter:integer;
        begin
            writeln('10 iterations');
            for iter := 1 to 10 do
            begin  count:=0;
                   for i:=0 to size do flags[i]:= true;
                   for i:=0 to size do if flags[i] then
                   begin
                        prime:=i+i+3;
                        k:=i+prime;
                        while k < size do
                        begin flags[k]:=false; k:=k+prime end;
                        count:=count+1
                   end;
            end;
            writeln(count,' primes');
        end.
```

compiler	compilatie-tijd (sec)	runtijd (sec)	grootte codefile (Kb)
TURBO-Pascal	4	17	12
MS-Pascal	112	16	28
Pascal/MT+	72	17	11

Tabel 5.1 Vergelijking compilatietijd, runtijd en codefile-grootte voor de source-file ERATOS.PAS.

De gegevens van tabel 5.1 zijn verkregen met behulp van een IBM-PC met 2 floppy-disks, zonder 8087 coprocessor. Alle compilers konden van dezelfde source-tekst gebruik maken.
Een andere test werd gedaan met het programma MULREG. MULREG voert een multiple regressie uit op 25 (onafhankelijke) variabele bij 100 waarnemingen. In dit programma dat uit 39 regels source-tekst bestaat, wordt een source-file van 400 regels ingevoegd.

De wijzigingen in de source-tekst voor de drie compilers betreffen (zie appendix):

- Definitie variabele lengtestring type.
- Specificatie van de include file.
- Openen en initialiseren van een disk-file.
- Close-statement.

In tabel 5.2, kolom 2 tot en met 4, zijn de resultaten van het programma MULREG dat op een IBM-PC met 2 floppy-disks zonder co-processor gedraaid is, weergegeven. In kolom 5 van tabel 5.2 zijn de run-tijden weergegeven voor het programma MULREG op een systeem met harddisk en 8087 co-processor.
N.B. Voor TURBO-Pascal is hierbij een apart geleverde 8087-versie gebruikt.
Ter vergelijking zijn in tabel 5.2 ook de compilatie, runtijden en codefile-groottes op B7900, VAX/VMS en VAX/ULTRIX gegeven.

compiler	compilatie-tijd (sec)	runtijd (sec)	grootte codefile (Kb)	runtijd (met 8087 co-processor) (sec)
TURBO-Pascal	17	112	22	52
MS-Pascal	420	178	63	36
Pascal/MT+	183	675	35	67
B7900 Pascal	2.0	1.4	48	
VAX/VMS-Pascal	44.9	6.7	9	
Berkeley-Pascal (VAX/Ultrix)	84.6	6.1	36	

Tabel 5.2 Compilatietijd, runtijd en codefile-grootte voor het programma MULREG.

Opmerking 1:
De getallen voor VAX/VMS in tabel 5.2 zijn verkregen met de /NOOPTIMIZE compiler directive. Zonder deze directive werd een foute executie-file gegenereerd.

Opmerking 2:
De compilatietijden in kolom 2 van tabel 5.2 zijn inclusief de link-tijd. Hiervoor is bij MS-Pascal, Pascal/MT+ en VAX/VMS-Pascal een apart commando nodig.
Opmerking 3:
In de appendix is een overzicht van de verschillen in de 6 gebruikte source-teksten gegeven. In alle gevallen behoefde de include-file niet gewijzigd te worden.

6. Conclusies.

6.1. Conclusies voor Pascal op de PC.

TURBO-Pascal is goedkoop en snel (compilatietijd ± 80/100 regels per seconde als de code alleen in primair geheugen wordt gezet), is handig te gebruiken (editor maakt deel uit van programmeeromgeving) en heeft een goede interface naar MS-DOS (graphics, inline machine code, door de gebruiker geschreven I/O-routines).
Het heeft echter nogal wat afwijkingen van de ISO-Pascal standaard:
- Geen put en get.
- Geen procedures en functies als parameter.

Verder zijn er, behalve het gebruik van overlays en chaining, geen voorzieningen voor separate compilatie.
Daardoor is TURBO-Pascal niet geschikt voor toepassingen met procedurebibliotheken. De portabiliteit van bestaande programma's naar TURBO zal problemen opleveren vanwege de reeds genoemde afwijkingen van de standaard en vanwege afwijkend interactief I/O.
MS-Pascal en Pascal/MT+ voldoen beter aan de standaard dan TURBO-Pascal. Beide zijn echter aanzienlijk minder gemakkelijk in het gebruik omdat er geen editor bij de programmeeromgeving zit, de compiler veel trager werkt en de compiler veel meer ruimte in beslag neemt.
Uit de uitgevoerde tests komt de indruk naar voren dat, wat compilatie betreft, Pascal/MT+ aanzienlijk sneller is dan MS-Pascal. Helaas is de runtijd van Pascal/MT+ programma's met veel floating point-operaties aanzienlijk langer dan die van overeenkomstige MS-Pascal-programma's (respectievelijk een factor 6 en een factor 2 zonder en met gebruik van de 8087-coprocessor). De Pascal/MT+ compiler produceert ongeveer de helft kleinere code-files dan de MS-Pascal-compiler.

Voor zowel TURBO, MS-Pascal en Pascal/MT+ geldt dat programma-units groter dan 64 Kbyte niet of moeilijk gemaakt kunnen worden. Verder zijn al deze compilers behoorlijk compleet gedocumenteerd. De (file) I/O en andere uitbreidingen vertonen voor ieder van deze compilers verschillen.
Gezien de prijs verdient in die gevallen waarbij geen hoge eisen aan portabiliteit worden gesteld en geen gebruik wordt gemaakt van separate compilatie, en procedures als parameter, TURBO-Pascal de voorkeur. In de overige gevallen zal de keuze van geval tot geval bepaald moeten worden. Naast compilatietijd, runtijd en codefile-grootte zullen ook andere factoren een rol kunnen spelen, zoals de mogelijkheid tot gebruik van (graphics) libraries.

6.2. Conclusies ten aanzien van portabiliteit.

In tabel 6.1 is in een overzicht een aantal eigenschappen en beperkingen van Pascal-compilers opgenomen. Voor portabel programmeren in Pascal kan het volgende geadviseerd worden:

- Gebruik alleen constructies uit de ISO level 0 Pascal-standaard.
- Gebruik van procedures als parameters en vermijd put en get.
- Namen van identifiers niet langer dan 8 karakters.
- Gebruik kleine letters voor reserved words.
- Geen arrays en records groter dan 64 Kbyte gebruiken.

De programma's die volgens deze regels gemaakt zijn, zullen redelijk overdraagbaar zijn. Men moet echter voorbereid zijn om aanpassingen te maken op het gebied van:

- Interactieve I/O en ander file-gebruik.
- Verschillen in maximale grootte van reals en integers (bij kleinere systemen wordt niet op overflows gecheckt).

Voor wat Pascal onder MS-DOS betreft, zal men zich verder moeten beperken tot kleine programma-eenheden (kleiner dan 64 Kbyte).
In de praktijk is het vaak onvermijdelijk om van de door het beschikbare Pascal-systeem geboden uitbreidingen gebruik te maken. Helaas zijn de uitbreidingen op vele systemen verschillend geïmplementeerd, zodat door het gebruik van uitbreidingen de portabiliteit van programma's wordt beperkt.
Voor betere portabiliteit is het nodig dat de Pascal-standaard wordt uitgebreid met de in de inleiding genoemde faciliteiten. Zolang dit niet het geval is verdient het aanbeveling om de systeemafhankelijkheden in een liefst beperkt aantal procedures te concentreren.

	maximum aantal (significante) karakters per identifier	separate compilation-units	diepte include nesting	mogelijk-heden random access-files	maximum array-grootte	maximum module-grootte	voorzie-ningen dynamische arrays
Burroughs Pascal	72(72)	ja	5	seek			array-schemata
VAX/VMS Pascal	72(31)	ja	5	FIND, LOCATE			conformant schema's
Berkeley (UNIX) Pascal	160(160)	ja	10	nee	64 Kb	?	-
MS-Pascal	72(19)	modules/overlay	1	seek	64 Kb	64 Kb*	superarray
MT+ Pascal	72(8)	modules/overlay/ chain	1	seek	64 Kb	64 Kb	conformant arrays
TURBO-Pascal	127(127)	overlay/chain	1	seek	64 Kb	64 Kb	-
UCSD-Pascal	72(8)	ja	3	seek	16 Kb	?	-

* Door bijzondere voorzorgen kan men volgens het manual deze limiet passeren.

Tabel 6.1 Overzicht van de voornaamste beperkingen Pascal-systemen.

7. Referenties.

[1] Jensen, Kathleen & Niklaus Wirth
Pascal - User Manual and Report
Springer-Verlag, New York, 1975

[2] Specification for Computer Programming Language Pascal
British Standards Institution
Deze specificatie is volgens het voorwoord identiek met ISO 7185 (1982)
Bibliotheek Rekencentrum DEF 82 SPE

[3] B5000/B6000/B7000 Series Pascal Reference Manual
Burroughs publikatie nr. 5014558

[4] Programming in VAX-11 Pascal
Digital Equipment Corporation, ordernr. AA-L369B-TE

[5] Berkeley Pascal User's Manual
Version 3.0, July 1983
in: ULTRIX-32 Supplementary Documents. Volume II,
pp. (2-159)-(2-209), uitgave Digital Equipment Corporation,
ordernr.
AA-BG67A-TE

[6] Mircrosoft Pascal for the MS-DOS operating system
Volume 1, volume 2, uitgave Microsoft, 1984

[7] Turbo Pascal version 3.0 Reference Manual
1985, Borland International Inc.

[8] UCSD Pascal Reference for the UCSD p-System Version IV.0
Uitgave IBM

[9] Pascal MT+ language Reference Manual
Uitgave Digital Research

[10] RC-Informatie AG-80
Pascal op Burroughs Large Systems
Uitgave THE-Rekencentrum

[11] Gilbreath, J. and G. Gilbreath
Eratostheses Revisited
BYTE, January 1983, pp. 283-326

Appendix.

Enige niet-portabele Pascal-constructies.

In deze appendix worden voor de geteste compilers enige implementatie-afhankelijke eigenschappen getoond. De gepresenteerde gevallen zijn die welke tijdens het uitvoeren van de in dit rapport vermelde tests naar voren zijn gekomen. Uiteraard is deze lijst verre van compleet.

B7900.

```
{type declaration of variable length string of maximum 80 characters}
TYPE VLSTRING = STRING(80);
{include file with title "PROG/INCL" on the active family}
$ INCLUDE "PROG/INCL"
{open and initialize input file INP with title "inp/dat" and output file
 OUTP with title "outp/dat" both on disk USER1}
SETATTRIBUTE (INP, TITLE, 'inp/dat ON USER1');
RESET (INP);
SETATTRIBUTE (OUTP, TITLE, 'outp/dat ON USER1');
SETATTRIBUTE (OUTP, BLOCKSTRUCTURE, 0);
REWRITE (OUTP);
{close and save output file OUTP}
CLOSE (OUTP, SAVE);
```

VAX/ULTRIX.

```
{type declaration of fixed length string of 80 characters (no variable
 length allowed)}
type VLSTRING = packed array [1..80] of char;
{include file with title "prog.i"
 (the title has to end with ".i")}
# include "prog.i"
{open and initialize input file INP with title "inp.dat" and output file
 OUTP with title "outp.dat" both on active directory}
 reset (INP, 'inp.dat');
 rewrite (OUTP, 'outp.dat');
{close and save file OUTP}
{a file that is not removed is closed at  program completion}
```

Note: all reserved words need to be lowercase.

VAX/VMS.

```
{type declaration of variable length string of maximum 80 characters}
TYPE VLSTRING = VARYING[80] OF CHAR;
{include file with title "PROG.PAS"}
% include 'PROG.PAS'
{open and initialize
 input file INP with title "inp.dat"
 output file OUTP with title "outp.dat"
 both on active directory}
OPEN (INP, 'inp.dat', OLD);
RESET (INP);
OPEN (OUTP, 'outp.dat');
REWRITE (OUTP)
{close and save file OUTP}
close (OUTP, SAVE);
```

Pascal/MT+

```
{type declaration of variable length string of maximum 80 characters}
TYPE VLSTRING = STRING[80];
{include file with title "B:PROG.PAS"}
{$I B:PROG.PAS}
{open and initialize
 input file INP with title "b:inp.dat"
 output file OUTP with title "b:outp.dat"}
ASSIGN (INP, 'b:inp.dat');
RESET (INP);
ASSIGN (OUTP, 'b:outp.dat');
REWRITE (OUTP);
{close and save file OUTP
 (RESULT is variable of type INTEGER)}
CLOSE (OUTP, RESULT);
```

MS-Pascal.

```
{type declaration of variable length string of maximum 80 characters}
TYPE VLSTRING = LSTRING(80);
{include file with title: "PROG.PAS" on active disk}
$ include:'PROG.PAS'
{open and initialize
 input file INP with title "inp.dat"
 output file OUTP with title "outp.dat"
 both on active directory}
ASSIGN (INP, 'inp.dat');
RESET (INP);
ASSIGN (OUTP, 'outp.dat');
REWRITE (OUTP);
{close and save file OUTP}
CLOSE (OUTP);
```

TURBO-Pascal.

```
{type declaration of variable length string of maximum 80 characters}
TYPE VLSTRING = STRING[80];
{include file with title "B:PROG.PAS"}
{$I B:PROG.PAS}
{open and initialize
 input file INP with title "B:inp.dat"
 output file OUTP with title "B:outp.dat"}
ASSIGN (INP, 'B:inp.dat');
RESET (INP);
ASSIGN (OUTP, 'B:outp.dat');
REWRITE (OUTP);
{close and save file OUTP}
CLOSE (OUTP);
```