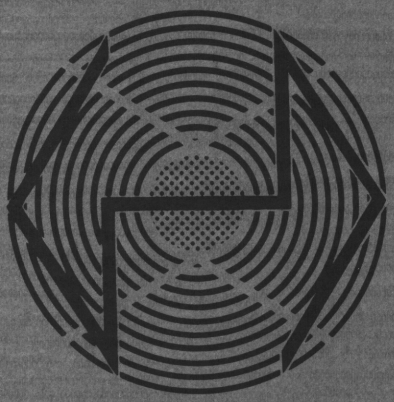


A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 · DK-2000 COPENHAGEN F

RCSL No: 31-D228
Edition: February 1973
Author: Tove Ann Aris

Title: HEAPSORT

Keywords: RC 4000 Software, Magnetic Tape Sorting, Users Manual.

Abstract: Heapsort sorts the records of a number of files on a number of magnetic tapes. The files may have MTS labels or no labels. The records may have fixed or variable length. Heapsort is an Algol 6 program. 10 pages.

CONTENTS

		PAGE
1.	Syntax of call	2
1.1	Syntax of data	2
1.2	Comments to call	2
1.3	Comments to message, sorting spec. and criterions	2
1.4	Examples of call	3
1.5	Sorting specification	3
1.6	Sorting criterions	5
2.	Current output	6
2.1	Log output	6
2.2	Error messages	6
2.2.1	Errors during call and input of criterions and spec.	7
2.2.2	Errors during input	7
2.2.3	Errors during merging	7
3.	Environmental requirements	8
3.1	Structure of heapsort	8
3.2	Program call and specification	9
3.3	Generating strings	9
3.4	Merging strings	9
3.5	Final merging	9
3.6	Restart	9

1. Syntax of Call

heapsort {<source>}$_0^1$ {<input spec.>}$_1^{10}$ {<output spec.>}$_1^2$

<input spec. >::= in.<mtname>.<fileno>
<output spec.>::= out.<mtname>.<fileno>

1.1 Syntax of Data

{message <text string without <NL> >}$_0^1$

sorting specification

sorting criterions

1.2 Comments to Call

No source implies that reading of data continues from current input (usually next line).

Files are corresponding to RC 4000 software numbered from 0, but file 0 will usually contain a VOL label and should therefore not be used.

All input files will be read. If two output tapes are specified and output only needs one tape, then only one tape will be used. The log will show state of affairs, see 2.1.

1.3 Comment to Message, Sorting Specification and Criterions.

Message: the text string will be listed on current out.

Bytes are numbered from 1 corresponding to field variable values.

The specification must contain block and either fix or var, other parameters are initialized, see Sec. 1.4.

The program starts with initialization, e.g. maxblock:=512, see Sec. 3.1.

1.4 Examples of Call

```
heapsort in.mt12345.1 out.mt23456.1
message sorting
fix.30 block.100.200
real.4
```

```
heapsort in.mt123.2 in.mt123.6 out.mt1234.1
mts var.40 block.10.10 test
word.6 word.10
```

1.5 Sorting Specification

`block.<integer1>.<integer2>`

<integer1> = maxblockin in reals
<integer2> = maxblockout in reals
2 <= maxblockin <= maxblock
2 <= maxblockout <= maxblock

e.g. block.90.100

`var.<integer1>` <integer1> = max record length in bytes
`var.<integer1>.<integer2>`
<integer2> = byte in which actual record length is stored
if the record has been output by outvar (i.e. record length is stored in byte 2) then .<integer2> may be omitted.
2 <= <integer1> <= 4*min(maxblockin,maxblockout)
<integer1> must be even.
<integer1> >= <integer2>
<integer1> >= max. sorting byte no.

e.g. var.40
var.40.8

`fix.<integer>` <integer> = record length in bytes
2 <= <integer> <= 4*min(maxblockin,maxblockout)
<integer> must be even
<integer> >= max. sorting byte no.

e.g. fix.50

`tapes.<integer>` <integer> = no. of tape stations wanted
3 <= <integer> <= 10
tapes <= heaps

`parity.<integer>` <integer> = 0,2,4,6
if the parameter mts is used then only 0 and 2 is allowed

final.<integer>	if system 2 and <integer> = 1 then inputstation=outputstation. if system 3 and final<>0 then <integer> = device number of input and output station. This parameter makes only sense in case the installation has one station of a special type and wants as well input as output on this special station.
mts	fileno must by > 0
test	implies listing of all blocklength from input tapes, if var then also all record length.
destroy	input tapes are not unloaded and will be used during merging as working tapes
error	program will continue sorting of data with skipped blocks.

Initial values:

tapes =3	=> 3 tape stations required
parity=0	=> parity is odd
final=0	=> no special station required
mts=false	=> standard files
test=false	=> no listing of input block length and record length
error=false	=> program break after input of last file in case of any error
destroy=false	=> input tapes are unloaded after use

1.6 Sorting Criterions

byte.<integer>	<integer> is byte no. <integer> <= 4*maxblock
byte.<integer>.d	decreasing sorting

e.g. byte.5
byte.8.d

word.<integer>	<integer> is byte no. of backmost byte <integer> <= 4*maxblock <integer> must be even
word.<integer>.d	decreasing sorting

e.g. word.6
word.8.d

long.<integer>	see word
long.<integer>.d	
real.<integer>	see word
real.<integer>.d	

2. Current Output.

During sorting heapsort lists information to enable the user to check a possible error.

2.1. Log Output.

Below output corresponds to the calls in Sec. 1.4.

```
sorting

heapsort in.mt12345.1 out.mt23456.1
block.100.200 fix.30 tapes.3
real.4

open mt12345 file 1
eof on mt12345 file 1

no. of input records = 100

open mt23456 file 1
eof on mt23456 file 1
no. of output records = 100
```

```
heapsort in.mt123.2 in.mt123.6 out.mt1234.1
block.10.10 var.40.2 tapes.3 mts test
word.6 word.10

open mt123 file 2
eof on mt123 file 2

open mt123 file 6

inrec6   38:       12   10   16
inrec6   38:       18   20
inrec6   22:       22
inrec6   40:       24   16
inrec6   38:       18   20
inrec6   26:       12   14
eof on mt123 file 6

no. of input records = 12

open mt1234 file 1
eof on mt1234 file 1
no. of output records = 12
```

NB: in above example file no. 2 is empty.

2.2 Error Messages.

Any error message in the program call will break the program before any sorting takes place.

In case the parameter error is given in the specification the sorting will continue and only break after finished output. In any case all input will be read and checked.

If number of input records<>number of output records or in case of mts and error in eof label writing the program will break after finished output.

All breaks will terminate with the message

```
heapsort      0 line 338-341
```

After program run the fpcall

```
if ok.no
```

will check whether everything was ok.

2.2.1 Error During Call and Input of Criterions and Specification.

This call will give below error messages:

```
heapsort  in.mt1234.1 mt256.2
message too bad
var.30.6 mst destroy
byte.8 long.10.d integer.10



***heapsort alarm, the following
call parameters are unintelligible:
 mt256.2
***heapsort alarm, output parameters missing

too bad
***heapsort alarm, error in specification
mst
***heapsort alarm, error in sortcrit.
integer.10
***heapsort alarm, inconsistency in specification
block length missing
heapsort      0  line  338-341
```

2.2.2 Error During Input.

During reading from the input tape the following error messages may occur:

parity error block dd, block skipped

blocklength error block dd, block skipped

word defect block dd, block skipped

record length error block dd
maxindiv=dd, record length=dd, bytes skipped=dd

record length error block dd
firstkey=dd, record length=dd, bytes skipped=dd

2.3.3 Error During Merging.

Read error during merging is so serious the the program will break via stderror.

Write error will cause call of phase restart see Sec. 3.6

3. Environmental Requirements

Heapsort is an Algol 6 program for sorting of a number of files on a number of tapes.

Heapsort uses the external algol procedure heapsortpr, RCSL 31-D229.

3.1 Structure of Heapsort.

Heapsort is divided in phases with different functions and each phase is coded as a block, the phases are as follows:

3.2 Program call and input of sorting criterions and specification
3.3 Generating strings
3.4 Merging of strings
3.5 Final merging
3.6 Restart

Only the phases necessarily according to the present data structure will be used.

The program starts with the following initialization which may be changed if needed:

```
begin
comment heapsort 1972.12.07;
boolean system3;
integer sortud,coresegments,cdcpar,
        maxinput,maxoutput,maxkrit,firstsortfile,maxblock;

maxblock:=512;      comment max. block length, if changed to >1024 then
                            change langind:=z.bf extract 12 etc.;
sortud:=maxblock;   comment blocklength for sortzone;
firstsortfile:=1;   comment first file used on sorttapes;
coresegments:=16;   comment program segments in core during sorting;
maxinput:=10;       comment max. number of input files;
maxoutput:=2;       comment max. number of output tapes;
maxkrit:=30;        comment max number of sorting kriteries;
system3:=true;      comment true->sorttape names=<::>
                            false->sorttape names=<:srt81:>,<:srt82:>...;
```

3.2 Program Call and Specification.

All information is checked and the corresponding internal parameters and tabels are set. If the information is unacceptable the program will list the unacceptable information and break. After accept all information is listed.

3.3 Generating Strings.

The user specified input files are read, the sorted strings are output on working tapes. (from file 1 as the tapes are supposed to have a name label).

Under system 2 the working tapes will be named srt81, srt82, .. under system 3 the system will generate some names.

On current output will be listed opening and closing of files, e.g.

```
open mt5062 file 4
eof on mt5062 file 4
```

3.4 Merging Strings

If only one working tape is used then this phase will not be used.

If the parameter destroy is used then the first merging output will be on the input tape.

3.5 Final Merging

The final merging is a traditional merging as every tape contains one string.

Opening and closing will be listed as for 3.3.

3.6 Restart

This phase will automatically be activated in case of write error in 3.3 or 3.5.

The defect tape will be unloaded and the program will ask for another.

e.g.

```
hard write error
mount new tape
remount mt123456
```

After mounting the merging will continue from the correct place.