ICW nota 1851

february 1988

ASPECTEN VAN INFORMATIEVERWERKING
60
User's guide of the HANDY Instruction System

ing. J.B.H.M. van Gils

nota
ASPECTEN van INFORMATIEVERWERKING 60
instituut voor cultuurtechniek en waterhuishouding, wageningen

ASPECTEN van INFORMATIEVERWERKING

60

De nota's handelende over Aspecten van Informatieverwerking bevatten inlichtingen over de ontwikkeling van de informatieverwerking binnen het Instituut. Naast meer concluderende en toelichtende beschouwingen wordt aandacht besteed aan het gebruik van programma's, programmapakketten en apparatuur. Tevens worden inlichtingen gegeven over praktijkervaring met en toepassing van informatieverwerking

CONTENTS

LAYOUT OF THIS REPORT
The reader will find the text on the right hand page.
Table, schemes, and examples are placed on the left hand pages.
The explanations to table, schemes and examples are given at the opposite page.

Table 1. Types of one line instructions provided by HANDY.

closing notation to the question	expected one_line_ _instruction(s)	closing notation in instructions belonging to the question
? [Y/(N)]:	Y(ES) or y(es)	--
	N(O) or n(o) or empty	|
? [byte]:	integer value between	|
	-127 and +127 or empty	| <RETURN>
? [integer]:	integer value or empty	| or
? [real]:	real value or empty	| <space>/text<RETURN>
? [filename]:	filename or empty	|
? [yy/mm/dd hh:mm:ss]:	date/time value in	|
	max. format or empty	--
? [string]:	string of printable characters to be used as table text or empty	<RETURN>
: or empty	characters string in a block of instruction lines or empty	<space>/...<RETURN>

INTRODUCTION

The HANDY Instruction System (HIS) is a subsystem of a number of programs to process the instructions to be read.
Normally a programmed conversation on the screen will guide the user. Moreover this system enables testing, storing and automatic input of instructions.
The user will meet a more naturally reacting process than is suggested by the somewhat abstract and formal description below. It is recommended to use this guide during a terminal session of a program run.
Programs equipped with HIS are provided with modules from a HANDY library (Van Gils, 1984) installed on a VAX computer working under VMS. Most programs developed by the author are equipped with HIS.
Conversation and messages of HANDY modules are in english language.
The examples are taken from program ROCROP as described in Van Der Valk and Van Gils, 1983.

TYPE AND FORMAT OF INSTRUCTIONS

Instructions are detailed directives for the main program processing (e.g. calculation, data processing). The location in the sequence of instructions provides the interpretation by the program. In conversational mode questions are displayed indicating which interpretation will be made.
Instructions are asked by the program in readable characters in a suitable format. They are transmitted to the program when an end of record (= end of line) appears and are interpreted per line.
There are different types of interpretations programmed so there are different types of instructions as listed in table 1.

Explanation to table 1.
Pressing a key representing a nonprintable character is indicated by <key_name>.
A <RETURN> closes a line (end_of_record).
A question requiring one or more lines is placed in the lines before.
A question requiring a line in a block of instructions lines is only <space>: or the question is not put down.
A question requiring a line in a text block is placed before the line; the text block in the instructions is closed by an empty line.
After a closing <space>/ comment may appear in the line; comment is not used by the program; text cannot be closed by <space>/ because the /-sign may be part of the text.
Empty instructions consist of no or more spaces; a text line having spaces is considered non empty. Empty instructions are interpreted in the way defined in the text belonging to the question.
In some programs a date/time value without surrounding delimiters is required; because there are /-signs in this notation the closing notation of instructions is set to the string <space>/ ; in some cases a single /-sign will do.

```
┌──> reading detailed processing instructions
│        instruction input
│        data input
.          .
.          .
.          .
│      processing by stored instructions
└──<  input of control instructions
       = which group of processing instructions will be replaced or stop
```

Scheme 1. Structure of total program processing up till now.

```
------------------------------------------------------------------
|      .                                                         |
|      .                                                         |
|      .                                                         |
|                                                                |
|   PROGRAM CONTROL                                              |
|   ---------------                                              |
|        do you continue  program ROCROP? [Y/(N)]: Y             |
|    with another       instruction file? [Y/(N)]: n             |
|    with another               listfile? [Y/(N)]: n             |
|    with another         growing season? [Y/(N)]: n             |
|    with   other        planting system? [Y/(N)]: Y             |
|    with   other       plant properties? [Y/(N)]: Y             |
|    with   other                 prices? [Y/(N)]: n             |
|    with   other      radiation    data? [Y/(N)]: n             |
|    with   other      radiation factors? [Y/(N)]: n             |
|    with another               headline? [Y/(N)]: n             |
|                                                                |
|                                                                |
|   PLANTING SYSTEM                                              |
|   ---------------                                              |
|      .                                                         |
|      .                                                         |
|      .                                                         |
------------------------------------------------------------------
```

Example 1. Journal of a program control after a processing is done.

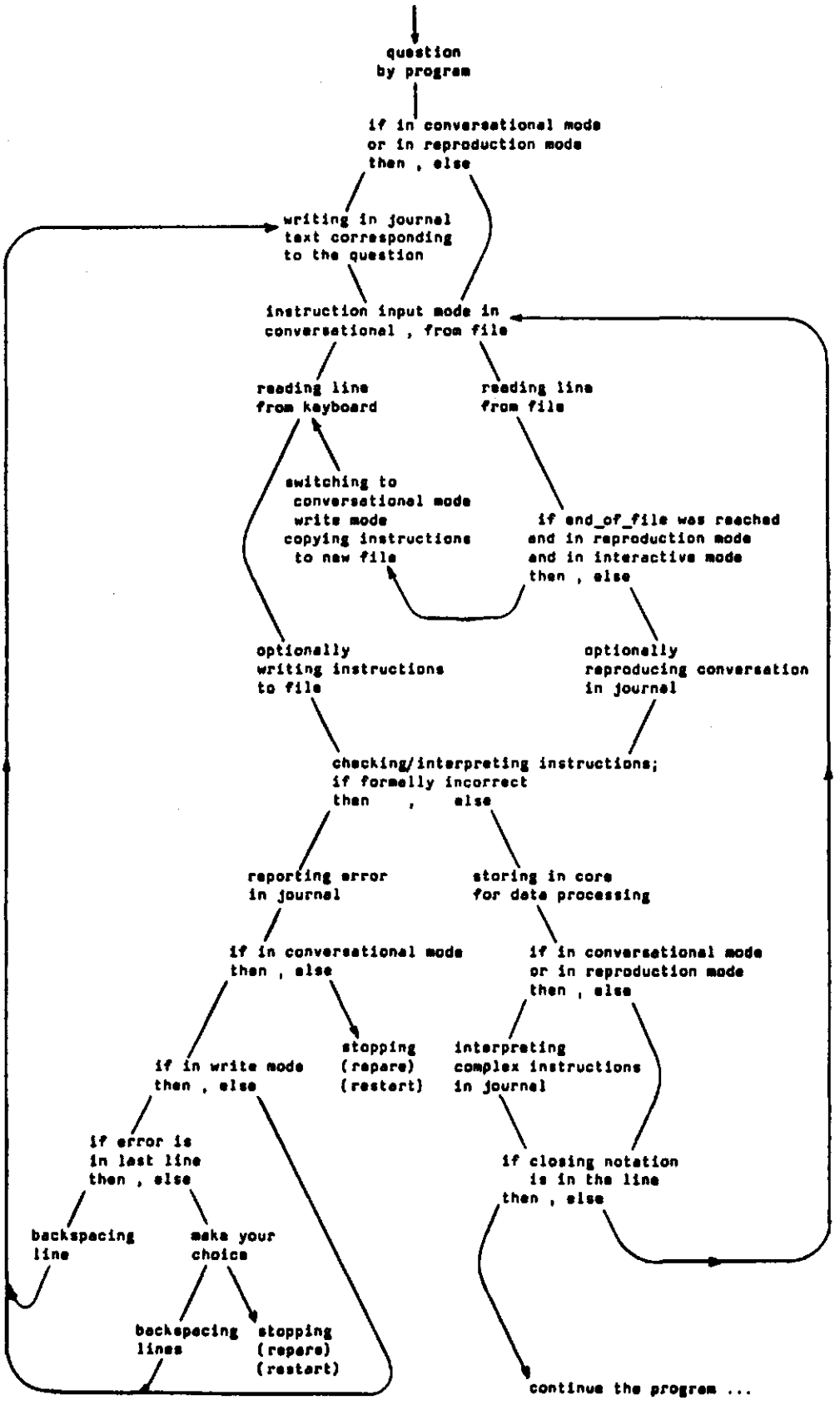
BLOCKS OF INSTRUCTION LINES

Some questions ask for a block of data. Mostly the block is stored as a single file, if wanted it is read from the terminal; there are no other possibilities of mixing other datatypes in the instructions.
Blocks of instructions are composed in a more complex format. The format is defined in the conversation when no pre-knowledge is required. Incorrect statements written in correct format may be interpreted by the program in a surprising manner; in cases that may happen, the program can output its interpretation.
Blocks of instructions consist of values between separators, these are clusters of one or more characters. The separating clusters define the meaning that the program gives to the value. Some constant values are written as strings surrounded by delimitors (single quote, ^-sign); /-signs, that are part of constants, don't close instructions.

PROCESSING BY MAIN PROGRAM

Uptill now all programs equipped with HIS have the same structure (scheme 1). After answering the last question processing starts immediately; only parts of the instructions will be replaced before the next processing is done.

Explanation to scheme 1:
Before processing by instructions starts, all instructions and other data are read, checked, interpreted and stored.
In conversation the groups of instructions are indicated. After processing a group of instructions is read controlling wich groups of instructions are replaced before the next processing starts (example 1).
Some programs don't allow a rerun with the same instructions.



Scheme 2. Processing of instruction lines corresponding to a question.

INPUT FROM FILE <--> CONVERSATION <--> STORAGE IN FILE
Instructions may be read in conversation (from terminal) or from an instruction file (Scheme 2).

Explanation to scheme 2:
At the moment of reading an instruction line the processing modes in the program have been set:
- instruction input mode: conversational or from file;
- reproduction mode or non-reproduction mode;
- write mode or non-write mode when in conversational mode;

In an instruction line it is determined:
- whether an end_of_file is reached;
- wheter and where a closing notation appears;

Together with the question in the program it is defined:
- whether instructions can comprise more lines;
- which notation closes the instructions corresponding to the question;
- how instructions are interpreted.

After a program stop the (incorrect) instructions remain unaltered, the errors can be repared in the instruction file and the program restarted. This also applies to a manual stop. The procedure yields a number of versions of instruction files and output files.

In the conversation the program puts the questions. In principle conversation is self explaining and "downward scrolling". Some more complex instructions need pre-knowledge (e.g. instructions for the calculator and selector in HANDY); the interpretation of these instructions by the program is shown in the journal. In case of an error message the question is repeated.

```
$ RUN 'ROCROP'ROCROP

     bulb production of tulips

  Filenames in the instructions:
      an empty type of an instruction file defaults to .INS
       "   "      "   "  "        input file      "      " .DAT
       "   "      "   "  "       output file     "      " .LIS
       "   "   name defaults to SYS$INPUT or SYS$OUTPUT
  A date is notated in the format YY/MM/DD


  INSTRUCTIONS
  ------------
        instructions from a file? [filename]: <RETURN>
        instructions   to a file? [filename]: ROCROP<RETURN>
     .
     .
     .
```

Example 2. Starting a program in conversation and with storage of instructions.

Instructions read in conversation can be stored in a file, corrected and reused (example 2).

Explanation to example 2:

The directory holding the executable program is the string value of DCL symbol ROCROP. When the symbol is mentioned between single quotes, the value of the symbol is substituted.
An empty filename stated in conversational mode causes terminal input (SYS$INPUT) or terminal output (SYS$OUTPUT).

During conversation stored instructions belonging to questions that require one line, contain the text of the question as comment after space and /-sign.

COMPOSING AND TESTING INSTRUCTIONS

When working in conversation (and with a test case) and when storing the instructions, the program can be stopped manually. This is useful in case a logically incorrect instruction is stated.

The part of instruction lines starting with the incorrect instruction can be removed by means of an editor. After that the program can be restarted with the shortened instructions. The instructions read from file are reproduced in the journal together with the questions and program interpretations as given in conversation. When reaching the end of the instruction file during the simulation of conversation, the program switches to conversation and storage of instructions on file. This procedure may be repeated until processing is going well.
After an error message caused by instructions read from file the program stops. The instructions can be corrected by means of an editor, after which the program is restarted.

```
$ ROCROPINS:=ROCROP
$ RUN 'ROCROP'ROCROP


INSTRUCTIONS
------------
with reproduction of instructions? [Y/(N)]: <RETURN>
    .
    .
    .
```

Example 3. Starting a program reading instructions from file without simulation of conversation.

INSTRUCTION INPUT IN A DCL COMMAND PROCEDURE
During automatic processing by execution of a command procedure conversation cannot be done so the instructions must come from an existing file.
At the start the program first tries to read the instruction file "HANDYINIT.TMP" (see Van Gils, 1984), in which only the name of the (stated) instruction file is expected. If this fails the program tries to read the name of the (stated) instruction file in a special DCL symbol. Mostly this symbol is given the program name (example 3). When the read action is successful the symbol is deleted; when the read action is not successful the program switches to conversation.

Explanation to example 3:
The symbol ROCROPINS is created and gets the value "ROCROP". Program ROCROP reads and deletes this symbol and switches to reading instructions from file ROCROP.INS .
The symbol named ROCROP is occupied (directory name) so the symbol holding the filename is named ROCROPINS.

In the journal file the answers on the questions don't come behind the question but in the next line; a non beauty of DCL. All messages are coming in the journal so the journal file should be inspected by the user.
The only legal program stop by a program is caused by the first instruction in the group of control instructions ("Do you continue program ...").
This stop returns the value TRUE in the DCL symbol "SUCCESSFUL", the other stops return the value FALSE. In a procedure this symbol value could be used to control the execution.

PROCESSING IN BATCH
A batch job is a DCL command procedure being executed independently of the terminal; this is a possibility to execute processing with tested instructions; the journal messages are very important, the conversation of tested instructions is not needed and may be suppressed. The time limit of the central processor in a batch job must be stated; the value can be estimated from the capacity reported in the journal after a processing is done.
Processing in bath can be done with the DCL symbol BATCH defined by command procedure 'HANDY'LOGIN and command procedure 'HANDY'BATCH (Van Gils, 1984).

REFERENCES
Gils, J.B.H.M. van. 1984. Aspecten van informatieverwerking 48.
HANDY '84 utilities user's guide. Nota ICW 1536. 31+1 pp.
Valk, G.G.M. van der, and J.B.H.M. van Gils. 1983. Bulb production of tulip crops, a preliminary model. ICW-nota 1486: 66 p.