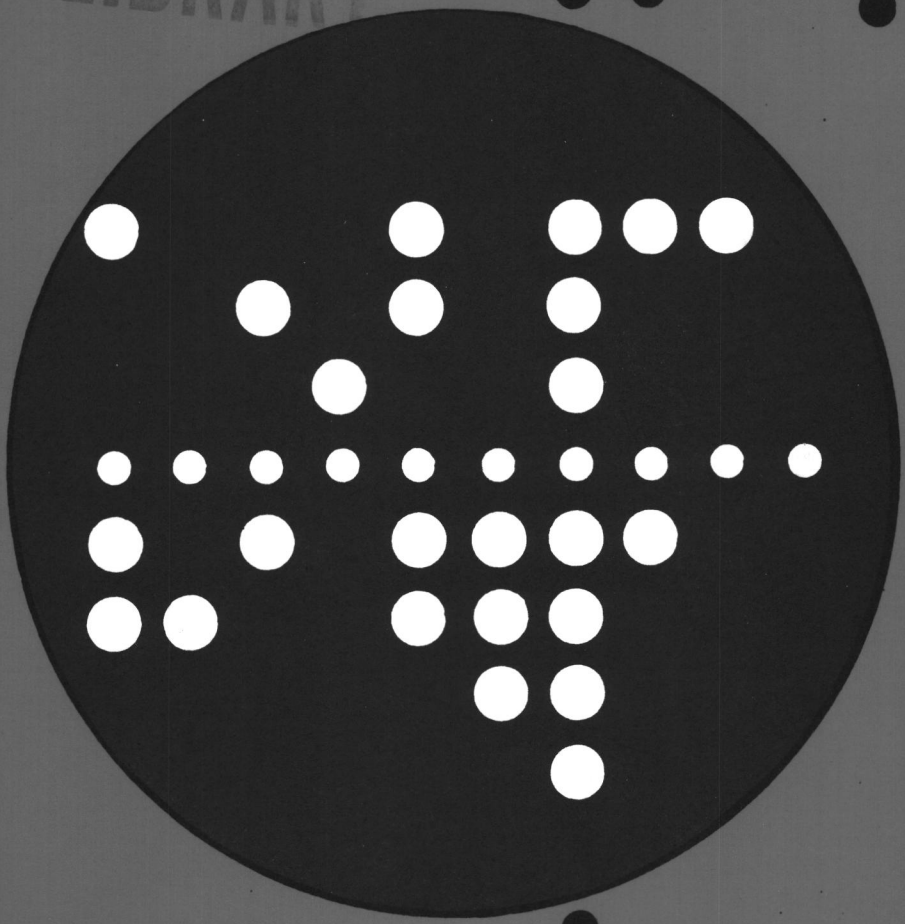
Commission of the European Communities

Joint Research Centre - Ispra

LIBRARY

Computing Centre Newsletter



November 1976 • No 6

REF: XV/6

Contents

Editorial note	2
Introduction to data base management systems	3
Introduction to STAIRS	7
The COREA system	11
Statistics on computer utilization – October	16
Utilization by objectives and accounts – October	17
Table of equivalent time, summary per month and cumulative	18
SHELTRAN, an example of application	19

Note of the Editor

The present Newsletter will be published monthly except for August and December.

The Newsletter will include:

- Developments, changes, uses of installations
- Announcements, news and abstracts on initiatives and accomplishments.

The Editor thanks in advance those who will want to contribute to the Newsletter by sending articles in English or French to one of the following persons of the Editorial Board.

Note de la Rédaction

Le présent Bulletin sera publié mensuellement excepté durant les mois d'août et décembre.

Le Bulletin traitera des:

- Développements, changements et emploi des installations
- Avis, nouvelles et résumés concernant les initiatives et les réalisations.

La Rédaction remercie d'avance ceux qui voudront bien contribuer au Bulletin en envoyant des articles en anglais ou français à l'un des membres du Comité de Rédaction.

Editorial Board / Comité de Rédaction

S.R. Gabbai, D.G. Ispra
H. de Wolde, C.C. Ispra
C. Pigni, C.C. Ispra
J. Pire, C.C. Ispra

Editor : Sylvia R. Gabbai
Layout : Paul De Hoe
Graphical and Printing Workshop, JRC Ispra

Acknowledgement should be given for their technical support to Mr. E. Eiselt, Mrs. M.G. Giaretta, Mrs. M. Van Andel, Mr. G. Clivio, A. Margnini, G. Zurlo

Introduction to Data Base Management Systems

A. Borella, S. Capobianchi

Reasons for Choosing a DBMS

The necessity of collecting an ever greater quantity of information relating to the various fields of activity of the Centre (technical scientific, financial, personnel administration, etc.) to be placed at the disposal of the various users for processing in different ways in accordance with specific requirements, has led the Informatics personnel to study and test some data processing systems for the construction and management of data banks (DBMS, Data Base Management System).

By data bank is meant a set of data stored just once and interconnected by logical and physical relationships. Such relationships, which form part of the bank, are defined and managed in a way independent of the application and of the data stored.

With the traditional information filing methods, the need to process a specific datum together with complementary data from different sources required redundancy of data and considerable programming effort. Additionally the lack of correlation between data from different sources failed to take full advantage of the data themselves.

Present integrated data management techniques (DBMS) have solved most of the difficulties of traditional methods.
Briefly, the most interesting characteristics of DBMS are:

a) the simple, safe, uniform and documented management of large amounts of data and their logical relationships,
b) the elimination of files directed to specific uses, by the use of an integrated system in which data are not repeated but stored just once, so that the data supplied can be up-dated and accurately controlled;
c) the availability of the data for interactive processing using a standardised and easy to apply Teleprocessing system; this allows a wider and more intensive use of the TP in the processing and inquiry of the data stored in a more sophisticated way;
d) the creation of data structured in accordance with physical and logical hierarchies with the possibility of modifying the logic structure on the basis of possible new requirements that necessitate a different view of the structure;

e) the maximum possible independence of the programs from the physical structure of the data storage in mass memories. The program can, in fact, use the logical structure of the data instead of the physical; this makes it possible to add (with caution) new types of information to the structures, in a way transparent to the already existing applications programs which are not directly concerned with the new data. Naturally this transparency can be used in all the applications programs which process data existing from the start in the physical structure; such programs need only know the data that concern them and take no account of the others.

f) The simultaneous access to the bank by more than one user. Access to data of a confidential nature can be protected by appropriate keys and consequently prevented for non-authorised users.

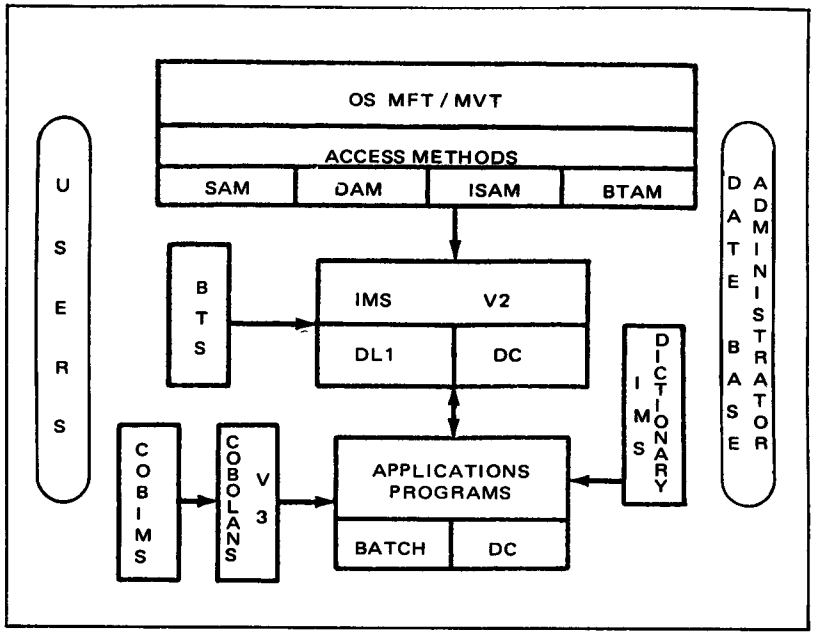
All these characteristics are so useful that it is difficult to imagine a future development of informatics in the management of large volumes of integrated data that prescinds the use of DBMS. It is clear that they entail a greater amount of work in the initial phase and require a delicate transition period in passing from a traditional environment (based on classical files) to a DB environment, but it is also clear that an 'a priori' rejection of DB methodology would be quite contrary to the line of development of informatics.

It must, however, be remembered that while DBMS techniques are extremely helpful in the study and solution of particular applications they do not remove the necessity for specific programming for everything which does not concern the management of data. Its use requires an accurate analysis of the problem to be solved which takes into account the requirements of the correlations between the data. If such analysis is not based on adequate technical preparation, the resulting structure could, from an efficiency point of view, greatly degrade the performance of the system giving rise to unacceptable times and costs.

Description of the DMBS Installed at Ispra: IBM's IMS

Having analysed the DBMSs at present offered by software manufacturers from both a cost/performance and a reliability and maintenance point of view, it was decided to try IBM's IMS (Information Management System).

From an experimental phase on several data banks (reliability data base, medical service, etc.) we then passed on to an operative phase (personnel and financial DBs).
The IMS installed at the Ispra Computer Centre was introduced into the environment shown in Fig. 1.



The basic IMS software is composed of two parts:

- the first, called DL1, for the management of the files in the data bases,
- the second, called DC (data communication), for the management of application programs, interactive or non-interactive.

Further complementary software has gradually been installed to produce an "environment" more suitable for users.

The use of a DBMS raises two kinds of problems:

• to bring about a centralised coordination of the DBMS, indispensable for the maintenance, development and safety of an IMS, DB/DC environment. In particular, this centralised DB/DC environment brings out some aspects relating to control and availability:

- of the data, their specifications and physical and logical connections;
- of transactions (inquiries and answers in an interactive environment);
- of the authorisations to use the resources (data, terminals).

All these functions in reality devolve onto a "logical" person, the Data Base Administrator, who is responsible for the physical safety of the DBs and their definitions as well as for the distribution of the resources controled by the DBMS and the enforcement of the regulations concerning the use of the system.

In order to help the DBA in his activities, an IBM product-program, the data dictionary, has been installed which, by means of the creation, updating and inquiry of various dictionaries (of the data, data and program descriptive blocks, user's libraries, formats), automatically supplies the DBA with documentary information on the state of the system.

• To make the system easy to use while safeguarding the integrity of the DBs for each user. It is, in fact, the task of the user to analyse his particular problem, check with the DBA whether the necessary information already exists on the operative DBs and define the new information to be added and the logical interconnection; once this phase has been achieved, thy programmer can devote himself exclusively to the programming and tests without having to concern himself about the structure of the data. He can, in fact, use the necessary data independent of their location and connection in the bank.

The IMS allows the use of programming languages such as COBOL, PL1 and Assembler by means of a direct interface via "Call".

To facilitate the task of the programmers further, two auxiliary product-programs have been installed, namely:

- COBIMS, (COBOL IMS), a COBOL pre-compiler possessing the statements necessary for the definition and call of the IMS function, and for the management of messages and errors while the programs are running. Not only does it generate the parts of the COBOL program necessary for the above functions, but introduces and verifies the definitions of the different components of the data bases, having as a secondary but no less important effect that of standardising the definition modules.
- the BTS (Batch Terminal System) with its 3270 formatting feature, is a program for simulating the Data Communication typical functions. The BTS allows interactive programs to be run in batch mode.

  The BTS accepts the card reader as input instead of the keyboard and the printer as output instead of the video screen. It thus simulates the characteristics of the normal terminals, such as the IBM 3270, supplying print-outs that represent the contents of the screen in its input and output conditions.

  Moreover, the BTS provides analytical and synthetic lists to check the correctness and optimisation of the programs. Thus, this product-program allows conversational programs to be developed without interference with the normal run of work already in production.

Introduction to STAIRS

S. Perschke, G. Fattori

Introduction

Recently the IBM program product STAIRS (Storage and Information Retrieval Systems) was implemented on our computer. STAIRS is a package for the creation, maintenance and on-line inquiry of a particular class of data bases in which the essential part of the information is recorded in natural language. However, it permits the structuring of the information contained in a "document", which is the basic unit of data, and the handling of formatted data.

The version implemented at Ispra uses the data communication facility of IMS (IMS/DC) as time sharing monitor while the data base creation and maintenance programs operate in batch mode.

STAIRS is to be used within the Ispra Establishment on the one hand for experimental work in automatic documentation and on the other hand for the creation of some of the data banks within the program of the JRC.

Data Base Structure

The structure and inter-relation of the different files which compose the data base is shown below.

Dictionary

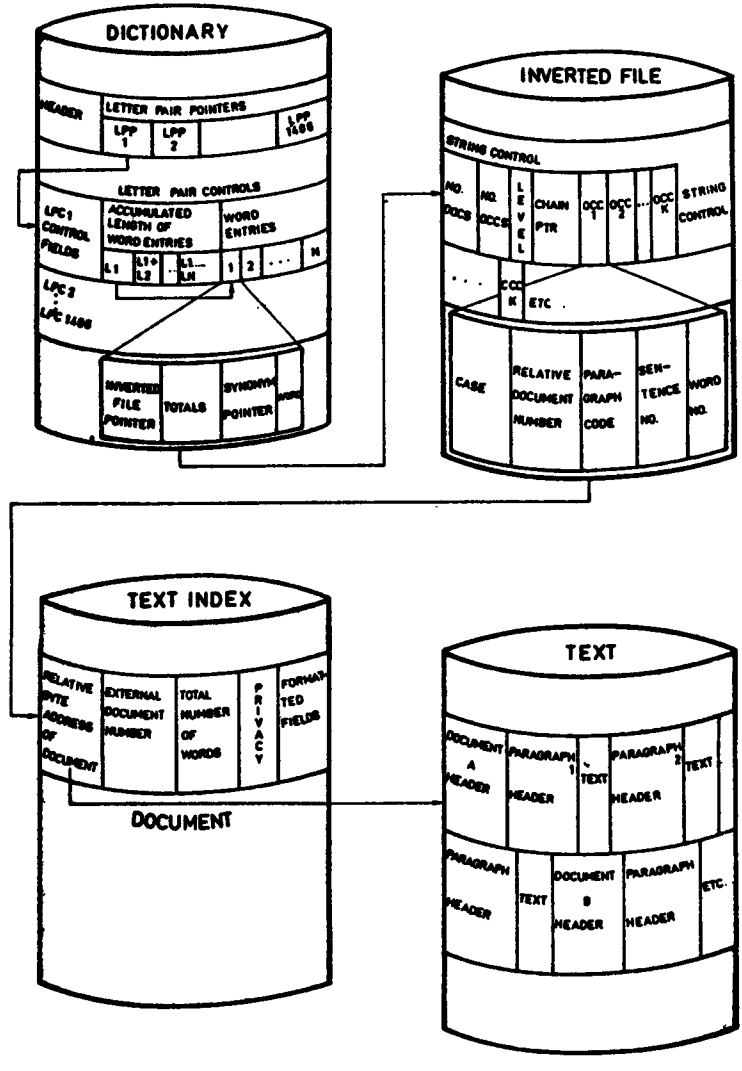
This is an indexed sequential data set whose access keys are the keywords extracted from the source data. The dictionary is the main access path to the data at inquiry time, and the search program includes facilities to permit access to terms whose exact spelling is not known. Each record of the dictionary contains the total number of occurrences of the term, the number of documents in which it is contained and the address of the associated inverted file entry.

Inverted file

This is a direct access file which contains a vector of pointers to the documents in which a word occurs along with positional information (paragraph, sentence, word number).

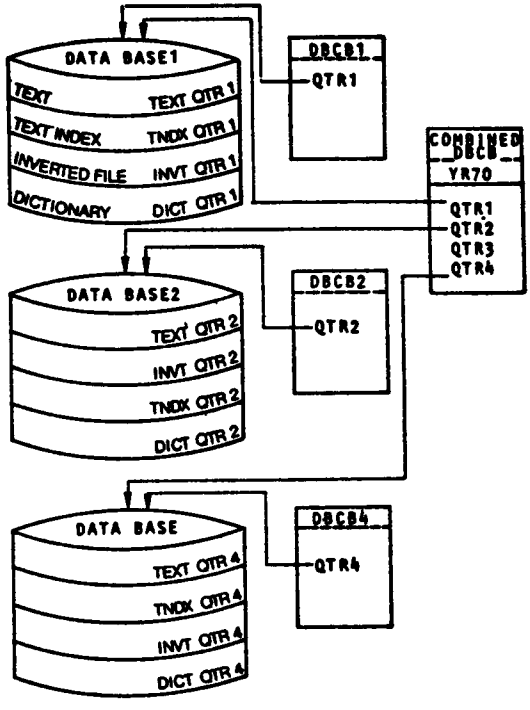
Text index

This is a direct access data set which contains mainly the formatted fields and information about the document (privacy, etc.).


DICTIONARY
HEADER
LETTER PAIR POINTERS
LPP 1
LPP 2
LPP 1406
LETTER PAIR CONTROLS
LPC 1 CONTROL FIELDS
ACCUMULATED LENGTH OF WORD ENTRIES
WORD ENTRIES
L1
L1+ L2
L1— LN
1
2
N
LPC 2
LPC 1406
INVERTED FILE POINTER
TOTALS
SYNONYM POINTER
INVERTED FILE
STRING CONTROL
NO. DOCS
NO. OCCS
LEVEL
CHAIN PTR
OCC 1
OCC 2
OCC K
STRING CONTROL
CCC K
ETC
CASE
RELATIVE DOCUMENT NUMBER
PARA-GRAPH CODE
SEN-TENCE NO.
WORD NO.
TEXT INDEX
RELATIVE BYTE ADDRESS OF DOCUMENT
EXTERNAL DOCUMENT NUMBER
TOTAL NUMBER OF WORDS
PRIVACY
FORMATTED FIELDS
DOCUMENT
TEXT
DOCUMENT A HEADER
PARAGRAPH 1 HEADER
TEXT
PARAGRAPH 2 HEADER
TEXT
PARAGRAPH HEADER
TEXT
DOCUMENT B HEADER
PARAGRAPH HEADER
ETC.

Text

This is a direct access data set which contains those data which are to be displayed or printed after search.
One of the interesting features of the STAIRS concept is the possibility of combining up to 16 (homogeneous) da bases which, in connection with the very articulated protection and privacy mechanism, can be used to control access to the single datum.



Document Structuring and Search Mechanisms

A document which is the unit of data to be retrieved is subdivided into:

- paragraphs
- sentences
- formatted fields.

Names can be associated with the paragraphs (or paragraph groups) and with the formatted fields.

As already stated above, the principal path of access to a data base is via dictionary and inverted file. This mode of inquiry is called "SEARCH".

Search terms or statements can be combined with each other:

a) with Boolean operators (AND, OR, NOT),
b) with positional operators (SAME - same paragraph, WITH - same sentence, ADJ - same word order).

The range of the search can be limited through the indication of particular paragraph in which a search term must or must not be located.
The access to the "formatted" field, for which STAIRS provides the "SELECT" mode, is to be considered a secondary path of access to the data base, because it implies a sequential scan of the entire TEXT INDEX data set, which might degrade the response time considerably.
It is therefore advisable to retrieve a subset of the data base with the search mode, before one enters the select mode.
The advantage of the select mode over the search mode is the availability, in addition to the Boolean operators, of relational operators (EQ - equal, NE - not equal, GT - greater than, LT - less than, WL - within limits, OL - outside limits, etc.) which makes it particularly suitable for numerical values. Data in the formatted fields can also be modified on-line during a search session.

Conclusion

In comparison with so-called Data Base Management Systems (like IMS, TOTAL, etc.), STAIRS is a package, i.e. a set of programs with well-defined functions and a certain number of options among which one can choose when a new data base is being defined and generated.

The great advantage of such a solution is that the effort of designing and creating a data base is minimized, because it involves virtually no programming, but one is limited in the possibilities of structuring the data and of defining access and transaction mechanisms.

It is therefore only through an examination of the information which is to constitute the data base, and of the use which is to be made of it that one can decide whether the use of this package is feasible for some application or whether one should embark on the effort and expense necessary to create a data bank using the services offered by a generalized DBMS.

The COREA System

G. Crestoni*), G. Gaggero, A.A. Pollicini

Introduction

The main purpose of the COREA system fits in the spirit of data acquisition and manipulation by terminal, avoiding use of punched cards, as stated in the article of Mr. Pire, published in the Newsletter No. 4.

It is planned that the COREA system will come into operation next January.

The next issue of the Newsletter will report some examples of application.

General Description

The COREA system has been developed to provide the users of the JRC computing installation with a simple and flexible tool for using a library of application programs from remote terminals.

The system works under the local conversational extension of HASP-2, TELEUR.

Outline of Facilities

The system provides the user with the following basic facilities:

- creation and editing of private data-files (hereafter called "Input-tasks") to be used as input data to library programs;
- submission of jobs asking for execution of a library program, using an input-task as case data and storing results into an user data-file (hereafter called "Output-task");
- selective inspection of the content of an Output-task.

The editing facilities can operate at line, character string or word level.

A special facility allows formatted data to be entered as list of items which is then edited according to an user specified format.

---

*) *PRAXIS CALCOLO Spa, Milano*

COREA Files

The COREA system operates, from the user point of view, on four logical files:

- the Input-Task File, (ITF)
- the Output-Task File, (OTF)
- the Library-Procedure File, (LPF)
- the Work File, (WF)

While the ITF, OTF, and LPF files are permanent files, the WF file is temporary, i.e. it exists only for the duration of a COREA session.

The Input-Task File

The ITF file contains all the user Input-tasks.
An Input-task is a named set of text-lines.
Lines are sequentially numbered from 1 to 7.200, which is the maximum number of lines in a task.
Each text-line contains 80 alphanumeric characters and, for this reason, it may also be referred to as a "card".

The Output-Task File

The OTF file contains all the user Output-tasks.
An Output-task is a named set of text-lines.
Lines are sequentially numbered from 1 to the maximum number of lines in an Output-task.
Each text-line can contain a maximum of 133 alphanumeric characters.

The Library-Procedure File

The LPF file contains all the JCL procedures for executing the COREA library programs.
From the user point of view the LPF is a read-only file. However, the user is allowed to make temporary changes to a procedure for an individual job submission.

The Work File

The WF file can contain different things during a COREA session.
It can be empty or contain either:

- a set of text-lines which have been entered but not yet used to update an input-task; or
- a library procedure which has been called for.

COREA Language

The user can enter "commands" and "text-lines".
An easy-to-use and extensive set of commands is available.

Command syntax:

Command-keyword [parameter - list]

Where:

Command-keyword	consists of a "flag character" immediately followed by one of the words which form the COREA command dictionary.
Parameter-list	is a list of parameters in accordance with the individual command syntax.

Any of the following characters $. /) can be used as flag character, but the character which is used as flag in the first command of a session is recognized as flag for the duration of that session.
The command-keyword and the parameter-list must be separated by at least one space.
The parameter-list can consist of one or more parameters. In the latter case, they are separated by a comma.
A parameter can have a subparameter, in which case the subparameter is enclosed in parantheses and must immediately follow the parameter.

The COREA command-dictionary contains the following words:

ALTER	FORMAT	LIST	PROGRAM	UNFORMAT
COPY	HELP	MODIFY	REPLACE	WRITE
DELETE	INSERT	NAME	STOP	
EXECUTE	JUSTIFY	OUTPUT	TASK	

All command-words may be abbreviated by typing the first character only. However all the characters typed are checked for correctness.

Text-line syntax:

character-string | item-list

Where :

character-string	is any string of alphanumeric characters. It can consist of up to 80 characters.
item-list	is a list of items in accordance with the rules for building a formatted text-line.

The first character of a text-line cannot be the character chosen as flag-character for the session. This restriction is imposed by the necessity to distinguish between command and text-lines.

The alternative forms "character-string" and "item-list" can be used for building unformatted and formatted text-lines respectively.

Commands Classification

Depending on the operation to be performed and the file(s) concerned, the commands can be grouped into five classes.

Furthermore, there exist three system commands, which provide analysis facilities to locate and remove system bugs.

Class of Commands for General Purposes

NAME	To open the system and to allow user to access Corea-files.
TASK	To identify a user Input or Output task to be activated for further operations.
HELP	To guide the user in learning system use and error recovery.
STOP	To close the system.

Class of Commands to Introduce text-lines into the WF

FORMAT	To declare a Fortran-format to build formatted text-lines.
JUSTIFY	To declare a tabulation to build formatted text-lines.
UNFORMAT	To introduce unformatted character strings.
MODIFY	To modify an incorrect text-line in the WF.

Class of Commands to Operate on Input-Tasks

INSERT	To insert text-lines into the active input-task.
REPLACE	To replace text-lines of the active input-task.
DELETE	To delete text-lines from the active input-task.
ALTER	To substitute a string of characters within a text-line.
COPY	To copy all or part of an input-task.
LIST	To list all or part of the active input-task.

Class of Commands for Job Submission

PROGRAM	To invoke a procedure to execute a library program.
EXECUTE	To perform job submission from the terminal.

Class of Commands to Operate on Output-Tasks

OUTPUT	To analyse the content of the active output-task.
WRITE	To print the active output-task.

- A course for potential users of the COREA system will be held on January 25th.
- A User's Manual will be made available to all interested people by the time of the course.

Statistics of computing installation utilization

Report of computing installation exploitation for the month of October

	YEAR 1976	YEAR 1975
Number of working days	21.50 d	23 d
Work hours from 8.00 to 24.00or	16.00 h	9.25 h
Duration of scheduled maintenance	29.81 h	19.50 h
Duration of unexpected maintenance	2.25 h	9.25 h
Total maintenance time	32.06 h	28.75 h
Total exploitation time	311.11 h	197.25 h
CPU time in problem mode	119.64 h	76.68 h
Teleprocessing:		
CPU time	2.38 h	0.89 h
I/O number	443,000	573,000
Equivalent time	3.10 h	4.90 h
Elapsed time	183.00 h	87.40 h
Batch processing:		
Number of jobs	10,194	8,291
Number of cards read	2867,000	2869,000
Number of cards punched	200,000	241,000
Number of lines printed	26160,000	21504,000
Number of pages printed	574,000	494,000

BATCH PROCESSING DISTRIBUTION BY CLASS

	A	1	2	3	4	5	D	TOTAL
Number of jobs	1322	3222	1460	2218	509	181	591	9503
Elapsed time (hrs)	23	121	98	189	82	42	57	612
CPU time (hrs)	0.7	11	21	36	29	12	6.5	116.2
Equivalent time (hrs)	9	31	43	91	41	22	30	267
Turn around time (hrs)	0.3	0.5	0.6	0.7	1.3	1.9	0.9	0.6

PERCENTAGE OF JOBS FINISHED IN LESS THAN

TIME	15'	30'	1^h	2^h	4^h	8^h	1^D	2^D	3^D	6^D
% year 1975	22.7	38.8	55.6	69.6	79.4	84.5	94.9	96.2	97.9	100
% year 1976	46.4	65.5	80.5	91.8	98.1	99.3	99.5	99.6	100	

Utilization of the computer center by the objectives and appropriation accounts for the month of October

		IBM 370/165 equivalent time in hours
120	General Infrastructure	59.3130
130	Scientific and Technical Support	0.7348
143	ESSOR Reactor	7.0843
145	Medium Activity Laboratory	0.0194
146	Central Bureau for Nuclear Measurements (CBNM)	–
191	Technical Support to Commission Activities	3.1564
193	Technical Support to Power Stations	
211	Waste Disposal	1.6620
213	Materials Science and Basic Research on Materials	2.5879
214	Hydrogen	0.8581
221	Reactor Safety	50.4739
222	Applied Informatics	50.9031
223	Information Analysis Services	22.4851
230	European Informatics Network	2.3795
251	Standards and Reference Materials	7.6376
252	Protection of the Environment	11.1125
253	Remote Sensing of Earth's Resources	18.0220
254	New Technologies	–
412	Fissile Materials Control	0.7664
	TOTAL	243.6443
190	Services to external Users	30.7908
	TOTAL	274.4351

EQUIVALENT TIME TABLE FOR ALL JOBS OF THE ADMINISTRATION – MONTHLY AND CUMULATIVE STATISTICS

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1975	64	55	62	73	62	61	94	52	51	59	74	70
accumulation	64	119	181	254	316	377	471	523	574	633	707	777
Year 1976	84	82	101	77	57	64	73	54	61	59		
accumulation	84	166	267	344	401	465	538	592	653	712		

EQUIVALENT TIME TABLE FOR THE JOBS OF ALL THE OBJECTIVES – MONTHLY AND CUMULATIVE STATISTICS

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1975	178	171	168	166	142	166	228	137	152	170	190	176
accumulation	178	349	517	683	825	991	1219	1356	1508	1678	1868	2044
Year 1976	206	237	270	241	229	248	249	223	233	244		
accumulation	206	443	713	954	1183	1431	1680	1903	2136	2380		

EQUIVALENT TIME TABLE FOR THE JOBS OF THE EXTERNAL USERS – MONTHLY AND CUMULATIVE STATISTICS

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1975	16	28	24	28	32	31	26	15	18	19	12	18
accumulation	16	44	68	96	128	159	185	200	218	237	249	267
Year 1976	18	19	28	16	25	32	14	11	27	31		
accumulation	18	37	65	81	106	138	152	163	190	221		

EQUIVALENT TIME TABLE FOR ALL JOBS OF ALL USERS – MONTHLY AND CUMULATIVE STATISTICS

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1975	214	216	208	215	190	222	266	166	181	202	219	208
accumulation	214	430	638	853	1043	1265	1531	1697	1878	2080	2299	2507
Year 1976	233	271	313	280	277	281	260	245	273	287		
accumulation	233	504	817	1097	1374	1655	1915	2160	2433	2720		

SHELTRAN : An Example of Application

A.A. Pollicini

Introduction

Structured Programming in a FORTRAN Environment

There is a basic need for structured programming techniques: the possibility of building program segments by means of **concatenation, selection** and **iteration.**

The FORTRAN control statements allow users with too freedom in implementing control patterns. Indeed there are some negative restrictions:

- there are no closed structures (1 entry, 1 normal exit),
- the logical IF has not a frame that intrinsically includes the alternative way;

and some harmful extensions:

- the DO structure may allow an unlimited number of exits,
- all type of GO TO statements may permit branchs anywhere in the program;

furthermore there are not logical parentheses like **begin end** to enclose blocks of statements.

The Control Structures Simulation Approach

The simplest way to go round these difficulties is the definition of a set of programming recommendations which show how simulate inexisting control structures by means of existing control statements.

Although it is not the goal of this presentation, the simulation of structured pattern is a very interesting technique.

Nevertheless it is well known that recommendations are rarely successful because it is necessary to enforce oneself a certain discipline to follow them.

The Precompiler Approach

The introduction of a precompilation step within a job, implies some drawbacks concerning time overhead, indirect path when interpreting compiler diagnostics, increase of rules to be observed etc.

But in this context, because the lack of structuring means in FORTRAN Language, it can be an useful technique as proved by the proliferation of structured FORTRAN precompilers.

Indeed a list of 69 such tools has been established.

One of them, the SHELTRAN precompiler, is available for use at the Ispra Computing Centre since November 1975.

A brief description of the language and an example of use is presented hereafter.

Description of the SHELTRAN Language

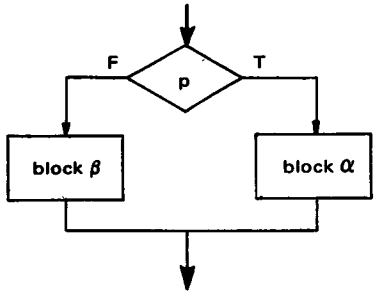
Control Structures

To inhibit the unrestraint use of branchs, the statement label is illegal in SHELTRAN, except for the FORMAT statements.

This fact implies that also the FORTRAN statements which refer to statement label (e.g. control statements) become illegal.

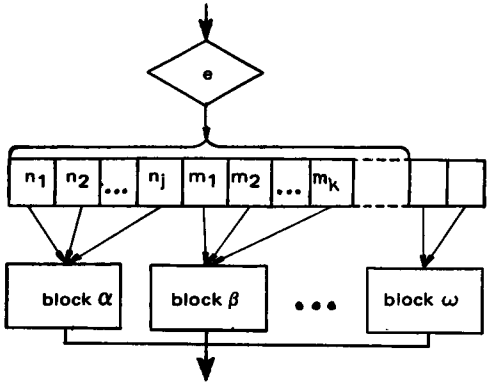
Of course, a set of suitable closed control structures is provided.
Notice that statement labels must be less than 10000.

SELECT-CASE-OTHER-CSELECT



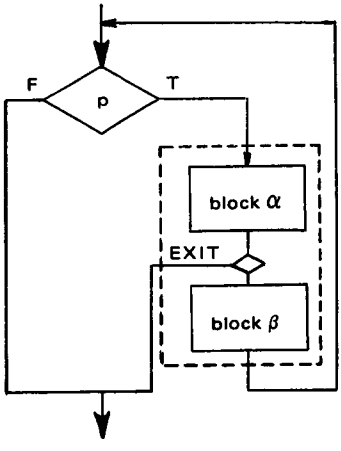
IF p
THEN
block α
[ELSE
block β]
CIF

IF-THEN-ELSE-CIF



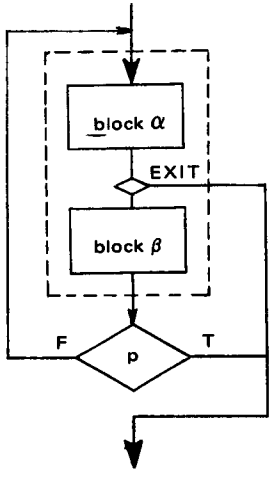
SELECT expression
CASE $n_1[, n_2, \ldots, n_j]$
block α
[CASE $m_1[, m_2, \ldots, m_k]$
block β
....
....]
OTHER
[block ω]
CSELECT

WHILE-XWHILE-CWHILE



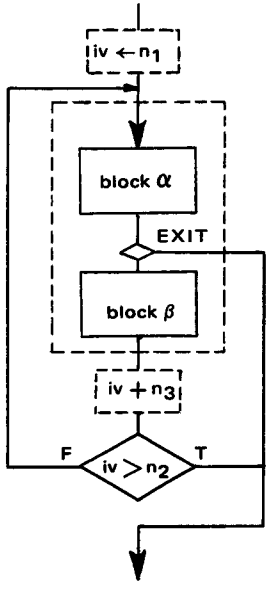

```
WHILE     p
      block α
[ XWHILE]
      block β
 CWHILE
```

REPEAT-XREPEAT-UNTIL




```
REPEAT
      block α
[ XREPEAT ]
      block β
 UNTIL  p
```

FOR-XFOR-CFOR



FOR iv = n_1, n_2 [, n_3]
block α
[XFOR]
block β
CFOR

The Procedure Concept

The SHELTRAN Language introduces into the FORTRAN environment, the facility of procedure definition, well known in other programming languages.

A SHELTRAN procedure is a set of statements which can be called for execution from several points of the program unit (MAIN, SUBROUTINE or FUNCTION) which defines it.

For this reason, the procedure has no parameters and can access all variables and arrays defined in the program unit that contains it.

A procedure is defined as follows:

```
PROC    name
        statements
CPROC
```

and can be referred to by the keyword PERFORM:

```
PERFORM    name
```

Editing Facilities

The SHELTRAN precompiler includes three output facilities:

- **Note.** That is a kind of comment card, whose content will be printed on the same line of the next instruction, starting from column 82.
  A Note card must have N in column 1.
- **Eject.** This command causes the next instruction to be printed on a new page.
  An Eject card must have E in column 1 and may contain a comment which will be printed as header of the new page.
- **Indentation.** The precompiler may edit the source statements, with automatic indentation of nested control structures.

A detailed description of the Language can be found in:

G.A. Croes, F. Deckers
Aspects of structured programming in FORTRAN
Informatie jaargang 17 n. 3 1975 (pp. 121, 131).

A copy of this paper, as well as other documentation can be requested at the EUROCOPI secretary.

The Application Example

The Problem

Generate all sequences of N characters, chosen from an alphabet of NS symbols, such that no two immediately adjacent subsequences are equal.

This example is a generalization of an algorithm presented by Niklaus Wirth in his book "Systematic Programming" and developed in several steps, using the PASCAL Programming Language.

The Design Phase

The initial framework is a global and abstract definition of the solving program, in which the logic layout of the control flow is already expressed by means of the keywords of the SHELTRAN structures, but the operations are described by synthetical phrases which point out logical and self-contained functions.

Such a design language is also called pseudo-code.

DEFINITION OF THE PROGRAM "GENSQ"

```
Get the alphabet.
Get the length of the sequence.
IF    the length is within the dimension limit,
  THEN
    Set to zero the counter of the generated sequences.
    Set a counter to define a null sequence.
    Declare valid the null sequence.
    REPEAT
        IF    the sequence is valid,
          THEN
            IF    the sequence has reached the fixed length,
              THEN
                Write the sequence.
                Increase by 1 the counter of the generated sequences.
                Change the sequence.
              ELSE
                Extend the sequence appending one character.
            CIF
          ELSE
            Change the sequence.
        CIF
        Declare valid a priori the new sequence.
        Check the sequence for validity.
    UNTIL  all valid sequences have been generated.
    Write the total number of valid sequence of the fixed length.
  ELSE
  Write a message to point out the input inconsistence.
CIF
END PROGRAM.
```

This global definition of the program contains the formulation of a set of functions to be performed. While many of them can be directly coded into one or more FORTRAN statements, there are three functions which imply some complex operations on the actual sequence.

A more detailed definition of these functions will be given below and represents the first refinement of the problem definition.

```
Function EXTEND
   Append one character to the right of the actual sequence.
   Assign the initial symbol of the alphabet to the new character.
END function.


Function CHANGE
   WHILE    the rightmost character of the sequence is equal to the
                terminal symbol of the alphabet,
      IF    the character under consideration is in the leftmost position,
        THEN
           Point out that no more sequence can be generated.
           XWHILE
        ELSE
           Decrease by one the sequence length, ignoring the
                rightmost character.
      CIF
   CWHILE
   IF    valid sequences can still be generated,
     THEN
        Look for what alphabet symbol is contained in the rightmost
             character.
        Replace the actual symbol with its successor within the alphabet.
   CIF
END function.
```

```
Function CHECK
   Take into account a null subfield.
   Set the boundary of the largest subfield.
   WHILE    sequence is potentially valid and subfields are shorter than
                the boundary,
        Increase by one the subfield length.
        Point to the rightmost position of the subfields.
        REPEAT
             Locate the homologous characters of the two subfields
                  to be verified.
             Declare a priori that the two subfields are different.
             IF   the characters under consideration are equal,
               THEN
                  Declare the two subfields potentially equal.
             CIF
             Point to the contiguous position of the subfields.
        UNTIL subfields are different or subfields have reached
                  the boundary.
   CWHILE
END function.
```

This first refinement level of the functions, for the problem is small, can directly be coded into FORTRAN statements.
But in practice, more levels can be needed before coding the program definition in source language.

The Coding Phase

At this point, coding the program in SHELTRAN Language is an easy task because the program structure is already supplied (according to the syntax of the language) by the program design in pseudo-code.

It will suffice to substitute each statement in natural language with SHELTRAN statements to obtain the source program listed below, which can be processed by the SHELTRAN precompiler.

```
VERSION 1 (APR 74 )                    SHELTRAN                    PAGE  1

TARGET          THE SHELTRAN TRANSLATOR IS A PROPERTY OF SIPM BV THE HAGUE
STM.NO

      C     THIS  PROGRAM IS A SOLUTION OF THE PROBLEM..
      C - FIND ALL SEQUENCES OF LENGTH  N  WITH NO EQUAL ADJACENT SUB-SEQUENCE
      C     ACCORDING TO THE ALGORITHMS DESCRIBED BY  NIKLAUS WIRTH  IN..
      C - SYSTEMATIC PROGRAMMING   PRENTICE-HALL,ENGLEWOOD CLIFFS,1973(PP.142-
      C
   2        DIMENSION ALF(4),S(8)
   3        LOGICAL GOOD,END
   4   500  FORMAT(14,6X,4A1)
   5   600  FORMAT(8X,4HS = ,8A1)
   6   601  FORMAT(//10X,33HTOTAL NUMBER OF VALID SEQUENCES =,I4)
   7   602  FORMAT(10X,58HINPUT PARAMETER  N  EXCEEDS THE SEQUENCE SIZE, FIXED
          1 AT 8.)
   8        LR=5
   9        LW=6
  10        READ(LR,500) NS,(ALF(I),I=1,NS)
  11        READ(LR,500) N
  12        IF  N.LE.8
  14        THEN
  14           M=0
  15           NTOT=0
  16           GOOD=.TRUE.
  17           END=.FALSE.
  18           REPEAT
  19              IF  GOOD
  21              THEN
  21                 IF  M.EQ.N
  23                 THEN
  23                    WRITE(LW,600) (S(I),I=1,N)
  24                    NTOT=NTOT+1
  25                    PERFORM  CHANGE
  29                 ELSE
  31                    PERFORM EXTEND
  35                 CIF
  36              ELSE
  38                 PERFORM CHANGE
  41              CIF
  42              GOOD=.TRUE.
  43              PERFORM CHECK
  47           UNTIL  END
  49           WRITE(LW,601) NTOT
  51        ELSE
  53           WRITE(LW,602)
  54        CIF
  55        STOP

VERSION 1 (APR 74 )                    SHELTRAN                    PAGE  2

TARGET          FUNCTION..EXTEND THE SEQUENCE APPENDING 1 CHARACTER.
STM.NO

  56        PROC  EXTEND
  57           M=M+1
  58           S(M)=ALF(I)
  59        CPROC
```

```
VERSION 1 (APR 74 )                    SHELTRAN                          PAGE  3
TARGET          FUNCTION..SHORTEN(IF NEEDED) AND REPLACE THE RIGHTMOST CHARACTER.
STM.NO

   60           PROC CHANGE
   61              WHILE   S(M).EQ.ALF(NS)
   64                  IF   M.EQ.1
   66                  THEN
   66                      END=.TRUE.
   67                      XWHILE
   68                  ELSE
   69                      M=M-1
   70                  CIF
   71              CWHILE
   73              IF   .NOT.END
   75              THEN
   75                  K=1
   76                  WHILE   S(M).NE.ALF(K)
   79                      K=K+1
   80                  CWHILE
   82                  S(M)=ALF(K+1)
   83              CIF
   84           CPROC
```

```
VERSION 1 (APR 74 )                    SHELTRAN                          PAGE  4
TARGET          FUNCTION..CHECK THE SEQUENCE FOR VALIDITY.
STM.NO

   85           PROC  CHECK
   86              L=0
   87              MH=M/2
   88              WHILE   GOOD.AND.L.LT.MH
   91                  L=L+1
   92                  I=0
   93                  REPEAT
   94                      K1=M-I
   95                      K2=M-L-I
   96                      GOOD=.TRUE.
   97                      IF   S(K1).EQ.S(K2)
   99                      THEN
   99                          GOOD=.FALSE.
  100                      CIF
  101                      I=I+1
  102                  UNTIL   GOOD.OR.I.EQ.L
  104              CWHILE
  107           CPROC
  108           END

END OF SEGMENT

  OPTIONS IN EFFECT - LINECOUNT=55,LINEWIDTH=131,  SOURCE,  FORTRAN  NUMBERING

   0 ERRORS              107 TARGET STATEMENTS
```

Execution and Results

Users are provided with two procedures that invoke the precompiler as first step:

- Precompilation and compilation (FORTRAN H)

```
// EXEC   SHPHC
//PRC.SYSIN DD  *
        {source in SHELTRAN language
/*
```

- Precompilation, compilation (FORTRAN H), Link-edit and execution

```
//  EXEC SHPHCLG
//PRC.SYSIN DD  *
        {source in SHELTRAN language
/*
//GO.SYSIN  DD  *
        { data
/*
```

Both procedures must be executed in class 3 because of memory requirement of the compiler FORTRAN H.

The program described here above has been executed with the following data:

N	= 4	length of the sequences
NS	= 3	number of alphabet symbols
X	Y Z	chosen symbols.

The output obtained is reported below:

```
S = XYXZ
S = XYZX
S = XYZY
S = XZXY
S = XZYX
S = XZYZ
S = YXYZ
S = YXZX
S = YXZY
S = YZXY
S = YZXZ
S = YZYX
S = ZXYX
S = ZXYZ
S = ZXZY
S = ZYXY
S = ZYXZ
S = ZYZX

TOTAL NUMBER OF VALID SEQUENCES =  18
```

☆ ☆

Nous informons nos lecteurs que, comme prévu, Computing Centre Newsletter ne paraitra pas en Décembre. A tous, la Rédaction présente ses meilleurs voeux pour Noël et le Nouvel An!

We inform our readers that, as formerly planned, the December issue of Computing Centre Newsletter will not be available. The Editorial Board whishes to all, a very Merry Christmas and a Happy New Year!

☆ ☆

Les personnes intéressées et désireuses de recevoir régulièrement "Computing Centre Newsletter" sont priées de remplir le bulletin suivant et de l'envoyer à:

Mme R. Porta
Bibliothèque des Programmes
Bât. 36, Tel. 760

Nom .....

Adresse .....

.....

Tel. .....

The Persons interested in receiving regularly the "Computing Centre Newsletter" are requested to fill out the following form and to send it to:

Mrs R. Porta
Program Library
Building 36, Tel. 760

Name .....

Address .....

.....

Tel. .....