ON LIBRARY ROUTINES IN THE STATISTICAL DATA PRODUCTION[1]

By SVEIN NORDBOTTEN and THOR AASTORP, Oslo

1. Routine libraries

Early in the history of program controlled computer application, users learned that programming efforts might be saved if programs and subroutines were written to satisfy certain criteria as to general use, and the concept of routine library was conceived.

There are two rather different types of libraries[2], the special library and the general library. The special library contains all tailor-made routines for recurring problems of exactly the same specification. In the statistical data production tailor-made programs have been extensively used owing to the peculiar applications occurring in a statistical bureau. This tendency makes cooperation between bureaus difficult as to special program exchange.

The cooperation has to be concentrated to the general libraries, which, however, might be very useful for the later elaboration of programs for the special libraries within each statistical bureau.

The definition of a general library routine is that it should at least be applicable to the same problem with formats or dimensions varying within given limits. The routines within the general library can be grouped into three classes according to how they work:

Class I: Routines for production runs of special applications the format of which are controlled by parameters.

Class II: Routines associated with a set of subroutines each identified by an assigned parameter value, and working on a varying length sequence of parameters by which the related subroutines either are compiled before or currently called during the production processing.

Class III: Routines comprising more or less complete alternative production routines for a special application and which by means of a parameter set defining format, etc. select and generate the optimum production routine for the problem with the specified format.

There exist programs which really may be classified in two classes. The principle difference between the classes is, however, important and may be made more clear by means of examples. A program for computing the square root of all integers up to a specified integer, is a typical program within the first class. The programs in this class can be made very efficient by machine oriented coding. They are, however, either by leaving few format dimension open for parametric specification, very much

[1] This paper is based on notes prepared for a meeting in Stockholm 12th—13th October 1961 of representatives from statistical bureaus using IBM 1401/7070 machines. — [2] According to the special terminology used in connection with data processing.

related to the routines in the special library and not within the scope of this paper, or by leaving many format dimension open for specification, are slow or inefficient for recurring runs.

The remaining classes represent two different approaches to solve the difficulties in making previous programming efforts useful.

The second class contains compiler and interpretive programs the aim of which are to reduce the programming by doing once and for all as much as possible of the detail coding. The class II programs are utilized in producing both general class I programs and special programs. The included subroutines are often problem oriented, and the specification parameter values take form of a problem oriented language which set up for solution of a problem is called a source program. IBM-compilers for 7070 are the Autocoder, Fortran and Cobol processors. The main characteristics of this type of compilers are their wide range of possible applications and the value of their associated languages as means in the efforts for standardization in problem formulation and documentation of solutions. Their general features may, however, cause a loss in operating efficiency of the resulting production programs and if these are to be used frequently the loss in efficiency may outweigh the gain in programming. The compiler programs are thus fitted for jobs which should be runned a few times or in a situation when machine capacity is no limiting factor.

Belonging to the second class are also the interpreters, where the translation of source program is performed step by step during the production run. Examples of interpretive programs are the programs for Simulation of the IBM 650 on the IBM 7070 and Simulation of the IBM 7070 on the IBM 704 and IBM 7090. Because translation has to be done every time an interpretive system is applied, the operating speed is usually low and interpreters are used mainly for a job occurring only once, in program testing and in the period transferring jobs from one computer to another.

There is another type of interpreters which is very important. That is the monitor programs the aim of which is to integrate several programmed processes in one continuous process avoiding manual managing of the system as far as possible.

The third class of programs contains the generators. The idea of a generator is that several applications are very frequently performed and deserve special attention. The general point of view represented by class II programs is sacrified in the generators. The aim of the generator program is both to reduce programming effort to a minimum and to achieve maximum operating efficiency for all formats of a certain problem. As the research in systems of administrative data processing applications has advanced, it has been found that these applications basically consist of standard processes even though the sequencing may vary. This research has increased the interest in generators and for the IBM 1401/7070 systems generators for Sorting, Merging and Report programs are now offered the user.

2. General programs for the statistical data production

Preparation of a large number of similar, but special programs and the integration of the processes of the different stages to an integrated, automatic

process, are common objectives for the efforts in the data processing field of the statistical bureaus. Recent reports from the US Bureau of the Census indicate that even working in the large scale of this agency, the processes are yet divided in separate runs requiring manual interference and implying risks of introducing human failures.

Two approaches have been tried to obtain the mentioned objectives. Messrs. F. Yates and H. R. Simpson have worked out an interpreter for a small machine able to take care of the different stages of statistical data production in one single run. Besides being restricted by the size of their machine, an interpretive scheme must be slow and of course require a new source program for each application. To our opinion this solution may be well fitted for application to small survey processing only to be done once.

The second approach is represented by Messrs. A. S. Douglas and A. J. Mitchell by a compiler with a source language well oriented towards the statistical terminology. This compiler seems to be very general and should in principle be able to solve many of the problems of statistical bureaus in one process including editing, selection, tabulation and analytical computation. The produced production program will probably be much faster than a corresponding interpretive, but in both cases a complete new source program must be written for each application. The scheme should therefore be adequate for frequently recurring applications on small data masses.

In general, the work of the statistical bureaus is comprising a large number of extensive jobs which often are recurring. Our proposal is a third alternative approach based on a brick-system of general routines integrated and controlled by a monitor program.

3. Monitor controlled statistical processing

The statistical data processing can be considered as comprising the following functions which may be performed in different sequences:

1. Administration
2. Data conversion and transcription
3. Sorting
4. Editing
5. File work
6. Tabulation
7. Analytical computations

We propose to elaborate general programs which together with already existing will cover these functions by means of compilers and/or generators producing special optimum production programs which are integrated to a single process by means of a monitor program. Some of these programs will be useful also for other kinds of users and it is reasonable to leave the construction of these to the manufacturers, while others are particularly useful for the statistical bureaus and must probably be worked out by ourselves. The rest of this section will be used for describing the ideas of the system more in detail.

3.1 The monitor program

The monitor program takes care of all coordination simulating the work of machine operation scheduling staff and represents the function of administration. The operation of the monitor is controlled by a set of control cards which the monitor is interpreting be-

tween the operation of two programs. Controlled by these specifications, the monitor has to be able to leave the control for further operations to any specified program on a library tape, it must be able, if necessary, to modify by specification all input and output allocations made in the individual programs in such a way that tapehandling may be reduced to a minimum, it must be able to do the housekeeping operations specified between the operation of two library programs such as printing messages, storage clearing, failure indicating, etc.

A very important part of the monitor system is the library tape. The library tape contains the necessary general programs, class I, class II or/and class III programs, covering all the necessary processes and conveniently labelled for reference in monitor control cards.

When the called program is a compiler, the monitor should be able to store the resulting production program on the library tape as an extra provisional program with specified label. A generator, on the other hand, will usually store the generated program directly in the internal store and leave the control to it immediately.

Another component of the system is the monitor control specifications supplying monitor and general programs with the necessary information. These specifications may be read by a cardreader.

A typical statistical process sequence consists of data input, sorting, editing and tabulation. Assume that the library tape besides the monitor program, contains a general input-program, a sort generator, a compiler for working out a production program according to a special editing, and a generator for tabulating programs, which are labelled from L_1 to L_4, respectively.

The work will proceed in this sequence specified by the monitor control cards:

1. Control is transferred to the editing compiler which itself reads in source program. The monitor is copying the resulting production program on the library tape with the specified label L_5.
2. Control is transferred to the general input-program which after reading the necessary parameters performs the initial card-to-tape process on the data to be processed.
3. Control is transferred to the sort generator which reads its parameters and leaves the generated program in the store, and then transfers control to it. The parameters read by the generator direct the sorting program to fetch the input automatically from the output station of the previous process.
4. Control is transferred to the compiler produced editing program labelled L_5 which works on the sorted output of the process no. 3.
5. Control is transferred again to the sort generator and the edited data are re-sorted.
6. Control is transferred to the generator of tabulation programs which generates a production program and leaves control to it to perform the final processing.
7. Control may again be transferred to sort and tabulation generators if necessary for further tabulation.

Monitor programs for the IBM 7090 systems already exist and IBM is also working on monitors for IBM 7070. The use of monitor programs is said to in-

crease the capacity of an 7090 system with 50—100 per cent, saving operating staff and increasing the quality of the final results by elimination of handling failures.

3.2 Input-output program

In addition to the usual editing features of card-to-tape, tape-to-card and tape-to-printer programs, a general program for statistical processing should also include the possibility of labelling *each item within a record.* Usually the item represents the observation of a characteristic which is identified by the position of the item within the record. When the record is merged in a file with other types of records from the same statistical unit but collected in other relations, the resulting record may be of varying length from one unit to another requiring labels for each item. It may be efficient to label the items already during conversion.

As the statistical masses are extensive and the conversion formats varying, it is thought that it will be worth while working out a generator for input-output programs.

3.3 Sorting programs

Sorting is a general function in data processing. The reason for sorting in the statistical data production is due to the necessity to merge one set of data with others and because it is a more efficient alternative than complete internal selection connected to the extensive tabulations.

Existing generators for sorting such as IBM 7070 Sort 90 seems to be powerful, and our attention should be concentrated at other functions.

3.4 Editing programs

The editing function is supposed to be particular for the statistical data production. An editing program must take care of both control of collected data and correction of wrong items. The control as well as the correction process are by principle more or less similar from one statistics to another and may perhaps be standardized.

There will, however, be marked differences between the editing in a population census and in a foreign trade statistics. In some applications tolerance intervals must be determined by statistical methods and correction performed by estimation procedures, in other the control criteria are prefixed and corrections based on Monte Carlo generations from certain empirical distribution.

The editing processes are therefore both similar in most applications and at the same time different as to the most convenient solution. Perhaps a compiler with a source language directly oriented towards editing problems will be the most suitable solution.

3.5 File work programs

As indicated above, the collected data represented as records must often be merged in larger files and later records or parts of records with specified labels have to be retrieved. This file work is also a very usual part of most business applications, and generators for file programs as for instance IBM 7070 Merge 91 are worked out.

Again the special requirements of statistical data production must be stressed. During the merging process it should for instance be possible to give the items of the new data records specified labels and during the retrieval process it must

be possible to retrieve either records or only items or both according to specifications referring to a complete or incomplete set of labels. The needs indicate that special attention should be given to the file work requirements of the statistical data production when producing general programs for this function.

3.6 Tabulation programs

Aggregating records in tabular forms or reports seems also to be a function common for very many applications within both business and statistical data production and several generators have been designed, e. g. the IBM 7070 Report Generator Program.

Usually it is desirable that a tabulation program is able to perform both individual and collective arithmetic operations in connection with the tabulation process. In our monitor scheme, it may be most efficient to do this kind of operations separately by the analytical computation programs, and concentrating efforts in the generator for tabulation on the pure tabulation functions, i. e. internal selection and accumulation. This will greatly simplify the design of a generator.

On the other hand, the generator ought to be designed to do selection of items based on item labels, the pair of labels and items arbitrary scattered in the records. So far, this seems to be a special requirement to tabulation programs from the statistical data producer.

3.7 Programs for analytical computations

The last brick in the proposed monitor system will be a compiler producing programs for arithmetic and analytical computations. The language of this compiler must be oriented towards the statistical terminology with powerful instruction which besides the elementary arithmetic operations should include instructions for direct computation of variance and standard deviation of items with specified labels, correlation and regression coefficient for specified labels, etc. In contrary to most other compilers, this one must as before mentioned, possess the ability to produce programs which can identify the items both by their relative position within the record and by their labels in the case they are packed and their relative position is not fixed.

The monitor controlled statistical processing outlined in this section is by no means planned in detail and much work has yet to be done as to the requirements for each program. Some preliminary studies and work have been done in the Central Bureau of Statistics of Norway on small generators for an IBM 1401 without any tape-equipment, and our experience so far seems to indicate that the monitor controlled system may be a useful approach for making statistical data production both integrated and efficient.

4. General programs and program generators

The ordering of IBM 1401 for the Central Bureau of Statistics of Norway was made assuming that this system should replace some of the conventional punch-card machines. We were fully aware that we had to rely upon an extensive use of general programs, because we did not have the programming capacity to make special programs for all problems. We therefore investigated the standard programs available for 1401, but found that they did not quite satisfy our re-

quirements. On that account we decided to develope general programs and program generators which were more suited for our purposes.

We started developing the programs by stating the basic requirements, which were:

i) The programs should be flexible, i. e. they should cover many different problems
ii) The programs generated should be economic and efficient regarding machine-time and storage positions
iii) The general programs should be easy to use.

On further evaluation of these requirements we strongly felt that the last two were the most important by far. Accordingly we were willing to sacrifice in flexibility in order to gain efficiency and ease in use, the more so as the flexibility was very restricted because of very limited storage capacity in our 1401.

Four months after the installation of the 1401 system, we were inclined to say that, owing to the general programs and program generators, this first period had been much easier and more successful than anticipated.

The rest of this section gives a short description of a general program and three frequently-used program generators which are developed by the Central Bureau of Statistics. Although these programs are written for a 1401 system using card input, card and printed output and comparatively few storage positions (4000), it is hoped that they may be of interest to other 1401 users as well.

4.1 List Program Generator

A problem often encountered, is to list the informations contained in the cards. This problem of preparing a listed report of the information in the cards is a conversion problem, i. e. card-to-printer, and requires a programmer's effort if not having the assistance of a program generator.

The List Program Generator produces list programs with a minimum of time and effort. The user writes specifications for the problem on a form designed for this purpose. The specifications are later punched on control cards which are used as parameter input to the List Program Generator.

The list programs which are generated, will produce listed reports with a speed of 533 lines per minute. The cards to be listed can consist of up to 4 different card types. One or two columns in the cards may be used in identifying the card types. The user must give the following specifications in the control cards: The number of card types, which columns that are used for identifications and what punched codes that identifies the different card types. Any punching that is valid in 1401, may be used as identifier.

The list programs can list up to 30 fields from the data cards. These fields are specified in the control cards by the column number of the first and last column. Furthermore the user must inform the generator where each field is going to be printed, i. e. the rightmost print position, and if the field shall be printed with or without left-hand zero-elimination. Any editing of the printed results beyond that can not take place.

Title and heading can be printed on the two first lines of the report if the information regarding this are given in the control cards. More than one card can be printed on the same line provided these cards belong to different card

types, i. e. only one card of each type can be printed on one line.

This program generator does not include any provision for tabulating, except optional counting of cards and printing out of these totals on up to three different control levels.

4.2 Report Program Generator

The Report Program Generator has been developed to produce programs that write and/or punch reports of varying format from cards. The need for writing a tailor-made program for each report is reduced to the requirement of supplying only a set of specifications. The user writes down the information relevant to the report on a special designed form. The specifications have to be written down according to some rules and conventions, these are, however, so simple that no detailed knowledge of the 1401 or programming are required from the users. The specifications are punched on control cards that constitute the input to the generator, which in turn generates the report program.

The report programs allow a maximum number of 30 accumulators on each control level. Fields from the cards may either be transferred to, added or subtracted into the accumulators which are numbered from A 1 to A 30. The accumulators may also be used for cross adding or — subtracting of numbers within cards, in these cases one has to use as many accumulators as factors in the cross summation. An accumulator may also be used as a counter by adding a 1 into it. There is no practical limit of the size of each accumulator, but the number of positions in all the accumulators on one level must not exceed 119.

Totals may be printed and/or punched out on four levels. One may freely choose which totals to print and/or punch and the order of these totals. If one summary card is not sufficient for all totals on one level, two cards can be punched.

The report programs provide for processing of max. 5 different types of cards in the same run. In order to identify the different card types, the following information should be given on the control cards:

i) location of the identification fields, max. 3 fields with up to 5 columns each. The identification fields must be located in the same place in all types of cards within one run,
ii) identification codes for each type of card, one for every identification field that is used,
iii) one identifier for every identification code.

The contents of the identification fields are compared with the identifiers to see if it is equal, unequal, greater or less. The identification codes specify what the results of the comparisons should be for each card type.

Furthermore, the generator includes provision for printing of title and heading, sorting of data — and summary cards, carriage control and programmed halts.

Although this generator can not produce list programs, such programs may, however, be simulated by the setting of one of the sense switches. The effect of this switch setting is that every card will be »listed», i. e. minor totals will be printed for every card. In the same way one may choose whether the cards should be sequence controlled or not by the setting of another sense switch.

There may occur programmed halts

both in the processing of the control cards and the data cards. A programmed halt on the control cards signals an error in these cards or out of sequence, while a programmed halt on the data cards indicates either undefined card type or out of sequence.

4.3 Reproducing Program Generator

By means of programs produced by this generator, all reproducing, X-gang-punching and reproducing with gang-punching can be carried out. This program generator may in fact accomplish all what normally is done on a conventional reproducer.

All informations concerning the reproducing and/or gangpunching are punched in control cards.

4.4 Condensing Program

When a program is tested and found to be correct, this Condensing Program can be used in order to pack the program so that the deck of program cards will consist of less cards.

The Condensing Program works, in contrast to IBM's Condensing Routine, after the Post Mortem principle. This means that the program to be condensed is first read into 1401 and stored, then the Condensing Program is run in and takes care of punching out the condensed program cards.

When using this program one must follow these two rules:

i) the programs to be condensed can not have instructions stored in Read, Punch or Print Area and
ii) the first instruction to be executed must have the same address in all programs, namely 190.

The condensed program will get a program number and a continuous card number punched in the cards.

5. Summary

In each computing center there usually exist two types of routine libraries of which the general library is of common interest for other data processing centers. The routines in this library are 1) general production routines, 2) program producing routines of compiler and interpreter type, and 3) program generating routines.

In order to reduce programming efforts and machine running time in statistical data production two approaches have been tried, i. e. construction of a statistical oriented interpreter and a statistical oriented compiler. A third approach is proposed in the paper. This approach is based on the monitor principle controlling and integrating the work of suited routines designed as program bricks for the different operations of the statistical data production.

The last part of the paper reports on several such general program bricks designed for an IBM 1401 to satisfy the particular requirements of the statistical data production.

References

DOUGLAS, A. S. & MITCHELL, A. J., Autostat: a language for statistical data processing. The Computer journal 3 (1960): 2, s. 61—66.

YATES, F. & SIMPSON, H. R., A general program for the analysis of surveys. The Computer journal 3 (1960): 3, s. 136—140.

KUNGL. BOKTR. STHLM 1962