4 APPLICATION OF COMPUTERS IN WRS

Processing of input data, and the design and operation of WRS require the use of powerful computer systems. At the beginning of the sixties data processing was the main example of the application of computers in water management (e.g. hydrological data processing, computation of statistical characteristics, data sorting, etc.). Lately, problems of reservoir storage e.g. release operations, complex hydraulic problems, etc. have been computed, and computers were used to simulate hydrological models for surface flow determination, to optimize hydroelectric power plants in power systems and to simulate models of WRS.

Computers are of fundamental importance for water management, with increasing application of the systems approach, e.g. mathematical models of water resource management in basins and, for the whole country, long-term planning, predictions, etc.

The use of computers necessitates knowledge of their function and the principles of programming, and particularly, a profound understanding of the professional problems involved in a proper analysis and formulation of tasks.

Theoretical knowledge should be accompanied by experience gained from direct contact with computer centres.

Anyone who intends to use computers has to adapt to their way of "thinking" as this adaptation is most effective in the man-machine system.

This chapter is intended as an introductory survey of the main types of computers, their facilities and devices and their use in WRS, as an explanation of some aspect of other chapters and as an introduction to specialized literature.

4.1 CHARACTERISTIC PROPERTIES OF COMPUTERS

The computation facilities include devices for easier and quicker computation, for mechanization and automation of computation and data processing. They include not only the general-purpose digital computers but also some tables, diagrams, nomographs, analog computers, single-purpose devices, etc. The methods of using these devices are sometimes called computation techniques (including computer software). The following analysis deals with computers.

A computer performs automatically a sequence of arithmetical and logical operations to obtain the required results. According to their principal physical function, computers are categorized into analogue, digital and hybrid computers. In WRS, digital computers are the most frequently used.

Analog computers

Electronic analog computers (Plander, 1969) represent the physical processes in models, and actual systems by analogous mathematical relations. The electronic computer units reflect the physical variables and numerical information by some analogous variable, e.g. by electrical force (voltage). One computing unit is necessary for each mathematical operation that takes place in an analog computer. Operational voltage is limited and this limit cannot be exceeded. Therefore the computer uses "fixed-point" methods of arithmetic.

There are two types of units:

– linear units (summator, inverter, integrator, coefficient potenciometer, etc.),
– non-linear units (multiplier, generator of functions, comparator, etc.).

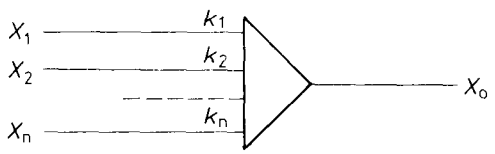

Fig. 4.1 Outline of the summator

Figure 4.1. shows a diagram of a summator, which produces a negative sum of input values multiplied by digital constants ($x_0 = -\sum_{i=1}^{n} k_i x_i$).

To set a task for an analog computer, a mathematical model of the system is first set up. The problem is formulated using physical laws so as to describe the system behaviour by mathematical relationships. By connecting the computer units, circuits are formed that carry out the computation of the problems.

Programming analog computer involves the preparation of the problem for the computer, drawing a block diagram and a scheme for the interconnection of computing units, the design of the experiments with the computer and the control of these experiments. The program, therefore, consists of formalized information on the total set of computing units and their connection, and on the dependent and independent variables which are physical realizations of variables in the problem analysed.

Some characteristic properties of the analog computer are as follows:

– dependent variables are expressed and processed in continuous form (analog computers are particularly suitable for problems of differential and integral calculus),
– programming is relatively easy,
– the computer provides a possibility for experiments on the model by variation of coefficients by potentiometers,
– the accuracy of computation is limited by the quality of computing units (0.1 to 0.01% of the full range of the maximum voltage),

– mathematical operations are carried out directly, possibilities for logical decisions and storage are limited,
– the computation is parallel, i.e. in all computer units at the same time.

Hybrid computers

Hybrid computers represent a purposeful combination of analog and digital computers. The advantages of both types of computers are utilized by employing the digital and analog sections with analog-digital and digital-analog converters. Hybrid computers were used in some optimization problems and it should, therefore, be possible to use them for WRS problems.

Digital computers

Digital computers model the algorithms by a sequence of operations that lead to the solution of the problems of the given type. These operations involve mathematical operations with numbers with a finite quantity of digits. Using the two basic physical states, not only numbers can be expressed but also alphameric and special information, logical constants and variables, etc. The possibilities of the digital computer are greater than those of the analog computer; digital computers can process any information (also non-mathematical) if the adequate algorithm is known.

Some characteristic properties of digital computers are as follows:
– the processing of information is performed in discrete steps sequentially,
– the computers are limited to the rudimentary arithmetical operations of addition, subtraction, multiplication and division; more complicated operations are performed by methods of numerical mathematics;
– the computers have the ability to carry out the logical operations of comparison, and of data storage on a large scale;
– they can use floating-point arithmetic;
– the accuracy of the computers depends only on the number of bits used for coding numerical information.

In modelling WRS, it is mainly digital computers that are used (further only "computers").

Computers can be classified in different ways e.g. according to "generation". Computers of a certain generation are characterized by certain types of construction elements. Computers of the first generation used vacuum tubes (Ural, Eniac), the second generation used transistors (Minsk, ZPA, NCR-Elliot, etc.) and the third generation uses integrated circuits (IBM 370, Siemens 4004, etc.). The principle and basic parts of computer structure are shown in Fig. 4.2. The input device serves for the input of the program and processed data into the main storage; at the same

time information is transformed into a form that can be used in the computer. The usual types of input devices are:

– the punched card reader, where the carrier of information is a punched card with 80 or 90 columns,

– the paper tape reader, where the program or data are punched in 5, 7 or 8 punching positions,

– the console printer-keyboard, used for manual control of the computer.

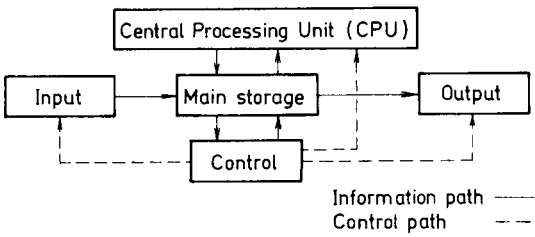

Fig. 4.2 Digital computer organization

The output devices provide the output of processed information from the computer. In general, the following output devices are used:

– a lineprinter (printer) for printing alpha characters, digits and special characters,

– a console printer-keyboard,

– a paper tape punch,

– a card punch,

– a display unit with vector graphics.

The control unit ("control") selects the instructions of the program sequentially from the main storage and controls their operation in the computer. It is the function of this equipment to interpret orders and to synchronize and direct the operation of all the units that make up the computer. Controls are required inside a computer system to (a) tell the input devices what data to enter into main storage and when to enter it, (b) tell the main storage where to place these data, (c) tell the CPU what operations to perform, where the data are to be found, and where to place the results, (d) tell them what file devices to access and what data to access, and (e) tell them what output devices the final results are to be written on.

The central processing unit performs arithmetical and logical operations in accordance with the algorithm of instructions from the control. The processor manipulations are performed one operation at a time with intermediate results being put back into main storage.

Storage is used to input, hold and output information about the program, input data, intermediate results and final results. There are two types of storage:

– main storage (primary, operational storage) provides the system with directly addressable fast-access storage data. Both data and programs must be loaded into main storage from input devices before they can be processed. The main storage consists of memory locations (cells) that can be handled directly; they can be identified by numbers called addresses. A unit of information held in a cell ("word") is formed by an ordered sequence of characters processed as a whole. The storage capacity is given in words or bytes (e.g. in the IBM system, 1 byte = 8 bits, 1 word – 32 bits = 4 bytes) that can be held in this storage. This capacity is expressed in K-1024 units, e.g. a computer (IBM) with the main storage capacity 32 K has 32 768 storage locations (bytes).

– external storage (auxiliary storage) is large-scale capacity storage with longer access time than main storage. Main storage takes from external storage all the information necessary for further processing and stores in it the intermediate results that are not necessary for the time being.

The most commonly used external storage devices include magnetic discs, magnetic tapes, magnetic drums and punched cards.

Computer devices and their construction elements are often called "hardware". The term "software" is used for programs or data not forming part a computer but used for its operation. The growing demands for capacity and the complexity of hardware require adequate software; nowadays the costs of both these parts are approximately equal. The software includes the system reference library programmes, subroutine packages (e.g. Scientific Subroutine Package), operating system and system control programs, a supervisory program, application programs, diagnostic and other service programs.

Parameters of Computer Hardware Used for WRS

For the computation of WRS problems computers of the JSEP series (Joint System of Electronic Processors) and some computers imported from western countries were used. The JSEP electronic computers are third-generation computers. They are produced in series of seven main compatible types, some of which have been used for WRS problems, i.e. EC (Electronic Computer) 1010 (made in Hungary), EC 1020 (the Soviet Union, Bulgaria), EC 1021, 1025 (Czechoslovakia), EC 1030 (the Soviet Union), EC 1040 (GDR), EC 1050 (the Soviet Union). The capacities and speed of CPU increase with higher number; the main advantages are the possibility to connect optimal devices (storage devices, printer, etc. in different computer system configuration) and compatibility of programs.

In Table 4.1. a brief survey of the basic parameters of JSEP computers is given:

For WRS problems, the most commonly used third-generation computers im-

ported from western countries in the sixties included: IBM 360 (USA) and Siemens 4004 (Germany), ICL System 4 and ICL 1900 (both U.K.).

The basic parameters of IBM 360/40 are as follows:

CPU capacity	64 K
type of main storage	ferrit
access time	1.25 μs
external storage	magnetic tapes magnetic discs
cards read/punch	reading 1 000 cards/min punching 200 cards/min
printer	1 100 lines/min
arithmetical speed	7.5 μs (addition in fixed-point arithmetic)

Most computer programs for WRS problems in case studies of the General Water Plan used National Elliott 4120, Elliott 503 and IBM 360/40 computers.

Table 4.1 The main parameters of JSEP computers (the usual computer configuration)

Type	EC 1020	EC 1030	EC 1040	EC 1050[4])
CPU capacity KB	64 – 256	128 – 512	256 – 1024	128 – 1024
CPU cycle time μs	2	1.25	1.35	1
Punch card reader[1])	EC 6012	EC 6012	EC 6012	EC 6012
Paper tape reader[1])	EC 6022	EC 6022	EC 7902	EC 6022
Card punch[2])	EC 7010	EC 7010	EC 7010	EC 7010
Paper tape punch[2])	EC 7022	EC 7022	EC 7902	EC 7022
Printer[3])	EC 7030	EC 7030	EC 7031	EC 7030

[1]) Speed of reading EC 6012 500 punched cards/min., EC 6022 1500 characters/sec.

[2]) Speed of punching EC 7010 100 punched cards/min., EC 7022 150 characters/sec., EC 7902 100 characters/sec.

[3]) Speed of printing EC 7030 890 lines/min., EC 7031 900 lines/min., Number of symbols per line EC 7030 128; EC 7031 120.

[4]) U.S.S.R.

4.2 USE OF COMPUTERS IN MODELLING

4.2.1 Task Algorithmization

In computer operation there are some basic differences between the way a human being works and the way a computer works. A human being uses his ability and experience to change or complete the incorrect or missing rules determining data processing and computation. Therefore, only rough rules for data processing are necessary; he uses them and, if necessary, he corrects and complements them. Every detail of a task for a computer, however, has to be prepared and logically treated. Every situation that might occur during computation must be taken into account in advance. Therefore the task is divided up into relatively simple steps that can be expressed by the program used in the computer. Before programming of the computer two preliminary stages are necessary:

– formulation of the problem – i.e. a clear definition of the task and its objectives, and the form, content and type of output data,

– analysis of the problem, i.e. investigation of the logical structure of the problem; this analysis includes the exact rules governing the processing of the input data in order to obtain the output data.

The decision to investigate the behaviour of WRS by, for example, a simulation model is often based on the results of these two stages. The final stage of analysis, i.e. algorithmizing the problem, is necessary before we start programming. The processing of data is defined in sufficient detail, i.e. the algorithm of the problem is developed. There are many references (e.g. Markov, 1954) to more detailed investigation of the theory of algorithms. Markov defined an algorithm as an exact description defining the computation process that starts with the variable input data and ends with desired results.

The algorithm must be:

determinative; it must be exact, precise and comprehensive and reproducible by anybody; there should be no doubts as to the further steps of data processing,

generic; it can process variable input data, i.e. more tasks of the same type can be performed by it,

resultant; it must give the required result after a finite number of operations.

This formulation explains rather than defines the concept of an algorithm. The mathematical notion of an algorithm is sometimes taken as a principle (axiom) that cannot be transformed into simpler ideas.

In WRS analysis, the concept of an algorithm is defined as a generic, accurately determined description of procedures for a single task or for a group of analogous tasks. These procedures consist of elementary steps that should be sequentially carried out to produce the output information; the choice of the elementary steps of the process is arbitrary. There is no need for an exact knowledge of the theory of

algorithms that gives precision to this intuitive concept and formalizes the proof of theorems describing the properties of algorithms. A water resource engineer who uses algorithms in practice can do without this theory since he uses its results without being aware of it. However, if he does know it, he can be sure that no mistakes or incorrect statements appear in his algorithms.

Complex WRS are often treated by teams of experts (task-force-groups) of different professions. The algorithm to be solved, i.e. the result of their work, is transformed by programming into a language that is comprehensible to the computer (i.e. a program is developed). It is not convenient to express the algorithm in terms used by experts in their professional languages; that must be done in a common language comprehensible both to the members of the team and to programmers. This form reduces possible errors due to misunderstanding between the members of the team, and communication with the programmer is made easier.

4.2.2 Flowcharts

Complex problems require communication between the people who take part in solving problems of different levels. The decisions resulting from the problem formulation and analysis should be correctly, briefly and unambiguously communicated to those who are to implement them. Usually, a graphic aid is used. Graphic representation and standard symbols aid the imagination, and understanding and facilitate intelligibility. They help orientation in complex problems, and sometimes they are indispensable for effective processing of information.

Graphic communication must:

- accurately and unambiguously represent complex logical problems,
- allow easy modification arising from changed conditions,
- be independent of computer type,
- provide accurately defined documentation.

The logical structure of a problem can be expressed in many ways. The most widely used form is the *flowchart*. For their own internal purposes and for dealing with customers, each of the major computer producers has, over the years, developed, adopted and published flowcharting conventions. Many committees were appointed to standardize these conventions. In the United States, this resulted in the ANSI standard, 1970 (American National Standard Institute). The 1970 revision extended the standard to match more closely that of the ISO (International Standard Organisation). The Czechoslovak standard ČSN 369 030 is similar to both these standards. The ANSI standard 1970, defines a set of graphic outlines, termed symbols. The flowchart symbols for information processing cover two major situations:

(a) representation of systems without indicating the nature of the component algorithms. It describes the system structure and relationships between activities. Other terms are procedure chart, run diagram.

(b) Representation of algorithms, especially those to be executed by computers. It schematizes the logical structure of the computer program for data processing. It focuses on the sequence of data transformation needed to produce the output data from the input data. Other terms are flow diagram, logic chart, block diagram.

(c) Representation of data diagram that indicates the flow of data in information processing.

The flowchart consists of several types of graphic outlines in which the operations are written in words or symbols or are indicated as a group of operations. The size of the outlines is not specified, and it is determined by the size of the content. The shape of the outlines should be maintained, i.e. the ratio of the width to the height and the general geometrical configuration. To indicate the sequence of operations or data, flowlines connect the outlines. To increase comprehensibility, horizontal and vertical flowlines with two preferred directions (top-to-bottom or left-to-right) are used in the positioning of outlines, and, if unambiguous, the open arrowhead can be omitted.

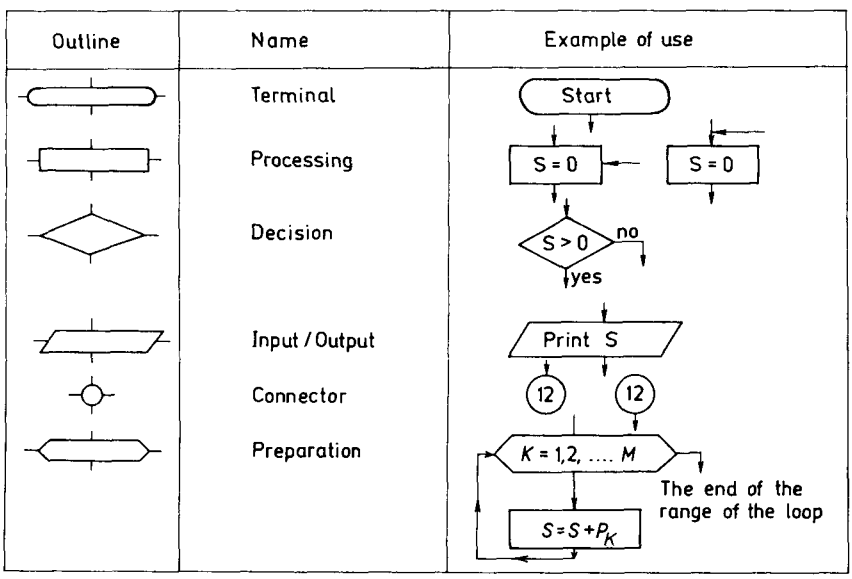

Fig. 4.3 Some outlines of the flow-charts

Figure 4.3. illustrates the outlines most frequently used in WRS modelling; they are sufficient for an understanding of the principles of flow charts:

Terminal serves to indicate a terminal point in the program – the beginning, and/or some break (start, stop, halt, delay, pause) in the usual line of flow. The terminal outlines in the first two situations are applied not only for the whole program but also for some self-contained portions, e.g. subroutines.

Processing is the general purpose outline that represents the performance of any operation (or group of operations) that is used for data transformation, data movement, or logical operations. The flowlines can enter this outline from any side, but it has only a single output.

Decision indicates comparison, decision, testing or switching operations, which determine or select among a variety of alternative flows (sequences of operations). The number of flowlines leaving a decision outline must always be greater than one (usually two or three), and these flowlines must be assigned (e.g. "yes", "no").

Input/output indicates an input or output operation, i.e. input of information for processing, recording or storing the processed information (output). It is defined for use irrespective of media format, equipment and timing. Some specialized outlines may be substituted for this outline, or the type of input/output medium or equipment may be described in words in this outline.

Connector describes the transition to another part of the flowchart (outconnector) or from it (inconnector). There are always at least two connectors, the related connectors being assigned the same symbol (letter, number, etc.).

Preparation indicates operations on the program itself. They are usually control, initialization, clean-up, or overhead operations not directly concerned with producing the output data, e.g. setting the value of a program switch.

The ANSI (and ČSN 369 030) specifies outlines for data-carrying media (document, magnetic tape, punched card, punched paper tape), for peripheral equipment (disc storage, display, manual input, off-line storage, on-line storage, communication link, etc.) for file processing (merge, sort, extract, etc.) and standard conventions for striping, cross-reference, multiple outlines, etc.

In computer processing, data are represented by *identifiers*, i.e. symbolic names. An identifier is a single alphabetical character or a string of alphameric characters; the initial character of the string must be alphabetical, e.g. A, Avarage, S10 (e.g. 2B is not identifier). Symbolic names represent the contents, i.e. concrete values at some addressed locations. For example, the statement MIN = 0, written in the process outline, means that there is a zero in the location MIN. If the following process outline is B = MIN it means that the content of the location called B is identical with the content of the location called MIN and the initial content of the location MIN is maintained. The content of MIN is changed by the input of a further value into MIN. If a set of data with the same characteristics is processed an array is formed. The variables which the array comprises are called *subscripted variables*. The monthly flows for a period of 50 years can be put into the array P_I for $I = 1, 2, \ldots 600$ or into an array $P_{I,K}$ for $I = 1, 2, \ldots 12$ and $K = 1, 2, \ldots 50$.

Representation of an algorithm by a flowchart is usually performed in three stages:

– *initial operations*; these include initialization of data (e.g. by setting zeros for some variables), preparation of constants, printing of headings, etc.

– *main operations*; these perform the transformation of input data into output information. They include the input/output operations, arithmetical and logical operations, etc.

– *final operations*; arrangement of results in the desired form and their output.

The simulation model of WRS simulates the behaviour of the system at each time step – the computation is repeated in a cycle. The flowcharts are drawn for one step only with repetition drawn in the cycle (loop). Inside this loop other loops can be "nested" (i.e. they cannot exceed the range of the main loop). A group of statements in the cycle is to be executed a stated number of times while a control variable is incremented each time through the loop as long as some condition is satisfied. An example illustrating the use of a flowchart and the use of identifiers for variables is given in Fig. 4.4.

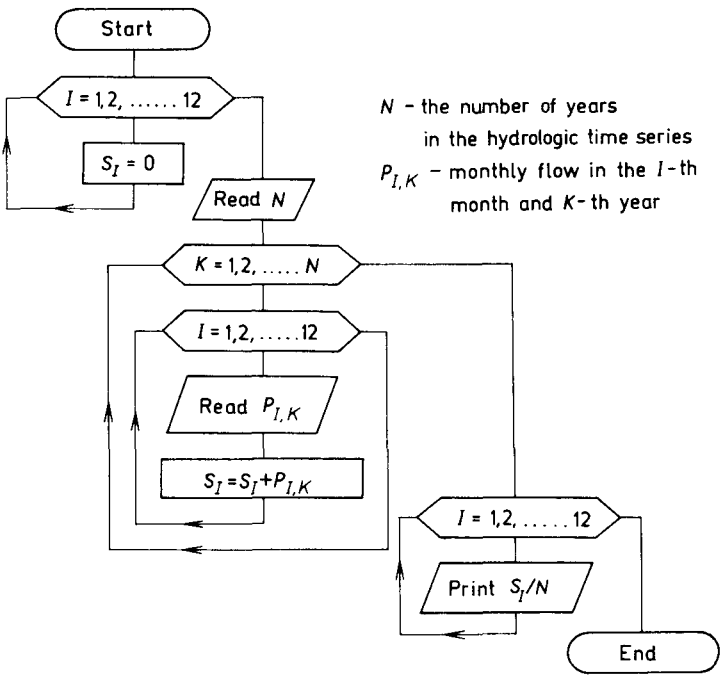

Fig. 4.4 Flow-chart of computation of average monthly flows

Two further types of graphic form of treating information-handling problems are *decision tables* and *flowcharts of operators*. The decision table is a tabular presentation of conditions (factors to consider in decision making), actions (steps to be taken when a certain combination of conditions exists) and rules (specific combinations of conditions and actions to be taken under those conditions). They are advantageous for problems with complex logical dependencies. Their use can eliminate

the irrelevant combinations from a complete set of possible combinations of conditions. In flowcharting it is difficult to determine if the whole problem has been covered, if all logical elements have been defined and analysed, and if all combinations of conditions have been exhausted. The decision tables form a transition stage for converting decision alternatives into computer programs. There are three methods for this conversion: manual coding from tables, processers that convert tables into source languages for input into existing compilers, and compilers that translate tables directly into a machine language.

According to some references, programming time is reduced by as much as 60% if decision tables rather then flowcharting are used. The use of decision tables might therefore be regarded as a progressive technique for programming WRS models.

The flowcharts of operators were developed by the mathematician Lyapunov in 1962. He distinguished between computation and program flowcharts of operators. The latter were developed from the former by adding the operators specific to the computer.

4.2.3 Languages of Automatic Programming

Once the problem has been algorithmized (by a team of experts) and the algorithm has been expressed in one of the stated ways, it is necessary to describe the algorithm in a form that can be used in the computer. The transformation of a method of problem solving from a natural into a computer language is called *programming*. Programming is a precise specification of a sequence of well-defined operations that execute the chosen method of solution; the resulting specification is called a program.

The computer can process information if it is expressed in a computer code that is obtained by *machine language programming*. The computer code is composed of a set of instructions, for operations within the capability of a given computer. The machine language contains these instructions and the rules for forming the sequence to represent the algorithm. For example, the addition operation in the computer EC 1030 is 00011010.

The first stage of machine language programming in bits, was followed by representation in a decadic form. An example of an instruction in the machine language is 10 00 0201 0202. The separate parts of this instruction mean: operation code, auxiliary code, two addresses (memory location reference). The programs in this form have a great number of numbers with many digits and many errors occur in programming. Correction of errors and changes in the program are difficult. Each computer has its own code and programs are not compatible. A change of computer means learning a new machine language. Machine language programming has, however, some advantages; it (or some higher machine-oriented language) is used

for programs or parts of programs if the time reduction in computation or a reduction in the main memory requirements is emphasized.

For these reasons, the most tedious part of programming was transferred to the computer. Programming languages closer to natural languages have been developed and have done away with the disadvantage of machine language programming. The first stage was the development of *symbolic machine languages*. Instead of absolute addresses, relocatable addresses are used, and for the operation code a mnemotechnic operation code is used. Translation into the computer code is performed by a *compiler*.

Further development brought *autocodes*, i.e. languages with a simple structure, the statements of which correspond to several instructions in the machine language. Their main disadvantage is a dependence on a particular type of computer.

A further advance is the system of *automatic programming* that consists of (a) the computer, (b) a source language program e.g. FORTRAN, ALGOL, PL/I, (c) a compiler that translates from the source language into the computer code, and (d) the product of translation, i.e. the program in the computer code.

The languages of automated programming do not depend on a computer type; they have the character of universal languages. Automatic programming facilitates coding of very complex algorithms and relieves the programmer of much routine work. The programs are well arranged, and easy to understand, which reduces the number of errors in their assembly. Most errors in machine language originated in coding operations. This can be done without errors by the computer itself. The main advantage, are the greater ease and shorter time of learning of the language and the absence of any tie to a particular computer type.

Survey of Programming Languages

In specialist literature more than 120 programming languages have been cited. They can be classified into several groups:

– languages for mathematical, numerical, scientific and engineering problems (ALGOL, FORTRAN, GPL, MAP, BASIC, etc.),
– automatic business data processing (COBOL),
– processing of files and strings (IPL-V, COMIT, etc.),
– formal algebraic operations (FORMAC, ALTRAN, etc.),
– general purpose languages (PL/I, FORMULA ALGOL, etc.),
– languages used in specialized disciplines (e.g. languages for programming machine tool design, civil engineering design IGES, etc.),
– design and assembling of compilers,
– simulation languages (GPPS, DYNAMO, SIMULA 67, etc.),
– languages for other purposes, e.g. query information retrieval systems (473 Query etc.).

Structure of Programming Languages

Automatic programming languages as synthetic languages are distinguished from natural languages by their simplicity. However, they have their own syntax and semantics. Syntax is the theory of making expressions from the units of the language. It determines precise rules for the formation of sequences of characters, symbols or units that are acceptable in the language. Semantics gives the feasible symbols (character strings, etc.) a certain meaning.

The simplest elements of the programming languages are the basic symbols, i.e. letters, digits, operators (e.g. multiplication sign), delimiters (e.g. brackets), special symbols and basic word symbols. By means of the syntactic rules the more complex units of the language, i.e. words, are formed. Word consist of identifiers and numbers. Each language forms compound symbols, i.e. key words that have a special purpose and can be used only in the way the language permits. Expressions are formed from words, operators, delimiters and special symbols; the execution of an expression yields the value of the expression. Assignment and descriptive sentences (simple sentences) are formed from expressions, words and basic symbols. Sentences are grouped into compound sentences (block of statements etc.) and the compound sentences together form the subroutines ("procedures" in ALGOL). A program is the highest-level unit of the language structure.

The best-known automatic programming languages are ALGOL, FORTRAN, PL/I and COBOL.

The programming language ALGOL-60 (Algorithmic language) was developed in 1960 especially for programming algorithms of scientific and engineering problems. After it was perfected in 1968 (ALGOL-68) it became a universal language that can also be used for data processing. The following forms of it can be distinguished:

- reference ALGOL – an exact definition of the language,
- publication ALGOL – the form in publications,
- computer-oriented ALGOL – this reflects the characteristic deviations for the particular computer used.

FORTRAN (Formula Translation) was developed in 1954 for coding algorithms of scientific and engineering problems and some problems of data processing by IBM computers. Modifications FORTRAN I to FORTRAN IV have gradually been developed.

PL/I (Programming Language I) was developed as a general purpose universal language that can be used for engineering, mathematical and scientific computations and for business data processing. It uses elements of ALGOL, FORTRAN and COBOL.

COBOL (Common Business-Oriented Language) was developed in 1959 for business data processing.

Most programs for WRS problems were written in ALGOL and FORTRAN, and some in PL/I. To illustrate the basic ideas of program structure and the possibilities provided by the automatic programming languages, a simple example is presented. The program written below is a modification of ALGOL for the NCR-ELLIOTT 4120 computer[1]). In the commentary to the program lines the main elements of the ALGOL language are described.

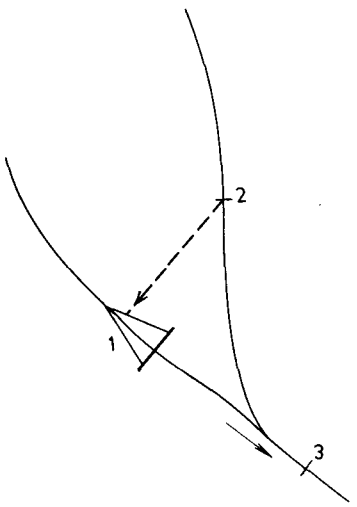

Fig. 4.5 Reservoir on tributary with diversion canal

EXAMPLE: Fig. 4.5 displays a schematic representation of a system with different elements, e.g. reservoir at site 1 and diversion channel from site 2 to site 1. The purpose of the reservoir is the river flow regulation at site 3 in order to obtain the draft *Q1* in June, July and August (e.g. reservoir operation with recreation constraints) and to obtain the draft *Q2* in other months. To keep the program relatively simple (the aim is to clarify the principles of programming of WRS) some simplifying assumptions have been introduced; e.g. all the losses of water in the reservoir were accounted for by some portion of the active storage, i.e. the calculated input value *WZA* is used (e.g. $WZA = 90\%$ of the active storage). The simulation model is applied with the time step $\Delta t = 1$ month; the hydrological data are represented by the series of monthly flows in the 40-year period.

For a chosen combination of input parameters *WZA*, *Q1* and *Q2*, the program computes the flows controlled by the reservoir operation (these flows are recorded), and the number of months when the required drafts *Q1* and *Q2* cannot be satisfied.

For a description of the program the lines have been given numbers.

[1]) The ALGOL language or FORTRAN is the form preferred for teaching basic programming processes. A modification for a computer was necessary to describe program inputs and outputs.

Text of the program:

```
(1) RESERVOIR;
(2) "COMMENT" PROGRAMMING ILLUSTRATION;
(3) "BEGIN" "REAL" WZA, Q1,Q2, WZ, QN, WK, P;
(4)         "INTEGER" K, I, T;
(5)         "ARRAY" P1, P2 [1 : 12, 1 : 40];
(6)         "PROCEDURE" HDATA (A); "ARRAY" A;
(7)         "FOR" K: = 1 "STEP" 1 "UNTIL" 40 "DO"
(8)         "FOR" I: = 1 "STEP" 1 "UNTIL" 12 "DO" "READ" A [I, K];
(9)         PUNCH (4); SAMELINE; ALIGNED (3,3): DIGITS (3);
(10)        HDATA (P1);
(11)        HDATA (P2);
(12) L1:    "READ" WZA, Q1, Q2;
(13)        WZA: = WZA/2.6298;
(14)        WZ: = WZA; T: = 0;
(15)        "FOR" K: = 1 "STEP" 1 "UNTIL" 40 "DO"
(16)        "FOR" I: = 1 "STEP" 1 "UNTIL" 12 "DO"
(17)        "BEGIN" P: = P1 [I, K] + P2 [I, K];
(18)                "IF" I > 7 "AND" I < 11 "THEN" QN: = Q1
                    "ELSE" QN: = Q2;
(19)                WK: = WZ + P - QN;
(20)                "IF" WK > WZA "THEN" WK: = WZA;
(21)                "IF" WK < 0 "THEN"
(22)                "BEGIN" T: = T + 1; "PRINT" - WK, K, I; WK = 0;
(23)                "END";
(24)                "PRINT" PUNCH (1), WZ - WK;
(25)                WZ: = WK;
(26)        "END";
(27)        "PRINT" 'WZA=', WZA * 2.6298, 'MIL.M3',
(28)                "L'Q1 = ', Q1, 'M3/S',
(29)                "L'Q2 = ', Q2, 'M3/S',
(30)                "L' NUMBER OF MONTHS WITH DEFICITS=',
                    T"F";
(31)        "GOTO" L1;
(32)        "END";
```

To explain the respective water management activities the identifiers used are listed, in the order they appear in the program (line (10) and following lines);

$P1_{I,K}$ $[m^3 s^{-1}]$ uncontrolled monthly flows at site 1, I-th month, K-th year,
$P2_{I,K}$ $[m^3 s^{-1}]$ uncontrolled monthly flows at site 2, I-th month, K-th year,

WZA [mil. m^3]	calculated input value of the active storage A_z (e.g. $WZA = 0.9\ A_z$)
$Q1$ [$m^3\ s^{-1}$]	controlled flows in the months June to August
$Q2$ [$m^3\ s^{-1}$]	controlled flows in the months September to May,
WZ [m^3]	the storage volume in the reservoir at the beginning of the month[1])
T	current number of months with water deficits
K	subscript of the annual loop
I	subscript of the monthly loop
P [$m^3\ s^{-1}$]	monthly reservoir inflows
QN [$m^3\ s^{-1}$]	draft value used in computation
WK [m^3]	the storage volume at the end of the month

Commentary on the program

(1) Name of the program

(2) The keyword "COMMENT" can be followed by any commentary, e.g. the goal of computation, the manner of program use, etc.; the compiler ignores the whole text between this keyword and the first semicolon.

(3) The key-word "BEGIN" marks the beginning of the program. This key-word is followed by a statement of the type, arrays and procedures (lines 3 to 8). The keyword "REAL" is followed by a list of real, floating-point variables.

(4) The keyword "INTEGER" is followed by a list of integer, fixed-point variables. Apart from the real and integer variables, logical variables ("BOOLEAN") can be used.

(5) *P1* and *P2* are the subscribed variables; a list of these is preceded by the key-word "ARRAY" and their identifiers are followed by the range for each dimension given by the bounds of subscripts – *P1* and *P2* are two-diemsional arrays.

(6) to (8) Description of the procedure for reading 40 × 12 values into the elements of the array $A_{I,K}$. Every procedure must have its identifier (in this case HDATA) that can be followed by a list of dummy arguments (A). These dummy arguments are used in statements of the procedure. If the procedure is used in the main program, it is called the procedure statement (identifier of this procedure is used), with dummy argument replaced by the actual parameters of the procedure. Procedures are used if the algorithm is repeated several times in different parts of the program or with different values.

(7) to (8) The "FOR" statement ("DO" statement in FORTRAN) specifies that a group of statements is to be executed a stated number of times while a control variable is incremented each time through the loop. The "FOR" statement has the following form: "FOR" ⟨variable⟩: = ⟨arithmetical expression⟩ "STEP" ⟨arithmetical expression⟩ "UNTIL" ⟨arithmetical expression⟩ "DO". The keyword "FOR" is fol-

[1]) The computation is done in units [$m^3 \cdot s^{-1}$], the conversion to [mil. m^3] is expressed in line (13).

lowed by the control variable. The symbol := assigns the control variable its initial value. The keyword "STEP" is followed by the step (increment) and "UNTIL" is followed by the upper bound, i.e. the highest value of the sequence of controlled variable values for which the loop is executed. The initial value, the step and the upper bound of the loop can be given by a constant, by a variable or by an arithmetical expression. The keyword "DO" can be followed by any statement (or by another "FOR" statement as in our example). The statement "READ" $A[I, K]$ executes reading of input data from a punched paper tape into the memory locations with the name $A_{I,K}$. The statement in the loop is first executed for the initial value of the control variable, then it is incremented by the value of the step that is repeated until the upper bound is reached (the upper bound is included). In our case the following values are read: $A_{1,1}$, $A_{2,1}$, ..., $A_{12,1}$, $A_{2,2}$, ..., $A_{12,40}$. (Another possible form with explicitly stated values of the control variable is: "FOR" K: = 2, 4, 6, 7 "DO" "READ" $P[K]$;)

(9) The statements in this line are computer oriented and they describe the form of print: PUNCH (4) specifies output on the lineprinter, SAMELINE, print on the same line, ALIGNED (3,3) specifies the form of print of real variables (sign or blank, three digits, decimal point, three digits), DIGITS (3) specifies the form of print of integer variables, i.e. the sign and three digits.

(10), (11) Calling procedures for reading of flows.

(12) L1 is the label for labelling the statement that follows. It is the target of the "GOTO" statement, i.e. the point to which the control is transferred if the "GOTO" statement is executed. The statement following the L1 label specifies reading of values WZA, $Q1$ and $Q2$ from the paper tape.

(13), (14) Assignment statements in the form ⟨variable⟩: = ⟨expression⟩.

(15), (16) "FOR" statements.

(17) to (26) A compound statement; its general form is "BEGIN", followed by several statement ending with semicolons and "END"; the statements between these keywords are executed in the order stated. (In out case for each pair of the specified control variables of the loop K and I).

(18) The conditional statement; the general form is "IF" ⟨boolean expression⟩ "THEN" ⟨unconditional statement⟩ "ELSE" ⟨statement⟩. If the condition is satisfied (boolean expression is true), i.e. $I > 7$ and $I < 11$, then the statement that immediately follows is executed, i.e. QN: = $Q1$; if it is not satisfied, then the statement QN: = $Q2$ is executed.

(19) The assignment statement.

(20) The "IF" statement: "IF" ⟨boolean expression⟩ "THEN" ⟨unconditional statement⟩. If the condition ($WK > WZA$) is satisfied, row (20) is executed, i.e. WK: = WZA; if is not not satisfied, the following statement is executed, i.e. the "IF" statement in rows (21) to (23).

(21) to (23) The "IF" statement; if the condition is satisfied the compound statement

in row (22) is executed. The keyword "END" in row (23) ends the compound statement.

(24) The statement for punching the flows controlled by the reservoir operation into the paper tape. The number in brackets is the number of the output channel (1 – paper tape punch, 3 – console keyboard, 4 – lineprinter).

(25) The assignment statement.

(26) The end of the compound statement.

(27) to (30) The print statement to output the values on the lineprinter the statement PUNCH (4) in row (9) is global, i.e. it is valid for the whole program with the exception of row (24) where a local statement requires punching of the paper tape). The inItruction to print character strings (i.e. the sequences of characters) is given by enclosing them by single inverted commas (e.g. 'STORAGE'); printing on a new line or a specific arrangement of output data is given by a specific character in double inverted commas (e.g."L" – for printing on a new line, "F" – new page, etc.).

(31) The "GOTO" statement; the form is "GOTO" ⟨label⟩; in our case the control is transmitted to the label L1 for further reading of input data.

(32) The end of the program.

The advantages of the ALGOL programming derive from the arrangement in blocks. A block is formed by a compound statement, in which the keyword "BEGIN" is followed immediately by a declaration. The arrangement of a block is: "BEGIN", declaration, statements of the block, "END";. The variables declared in this block are found only in this block or in blocks nested in this block (they are local in this block, the variables declared in the external block are global). If a complex algorithm is coded by an automatic programming language, subsystems can be distinguished. If these subsystems are programmed in a block form, the job can be divided among several programmers, which speeds up the programming work.

4.2.4 Debugging

Debugging is an activity that helps to isolate and correct mistakes in coding (in the chosen program language) and structure of the program. Debugging prevents malfunction of the program. The overall computation tests also include the computer hardware (which is in the charge of the technical personnel of the computer centre). The probability of malfunction of the computer is negligible, but errors are possible in the functioning of the input devices, for example, if the input media (punched cards, paper tapes) are worn out.

Programming requires accuracy and precision on the part of the programmer; even a highly experienced programmer cannot avoid mistakes. The transcription of the algorithm into the programming language is usually a source of syntactic errors (clerical errors, omision, etc.). The second group of more serious mistakes can

occur in the logical structuring of the program. If the definition of the problem is not complete or if it is not quite correct, some important relationships may be omitted.

Debugging is an integral part of the programmer's activities (see Fig. 4.6). These activities include main four steps: writing of the program in the source code, preparatory activities, and two debugging steps.

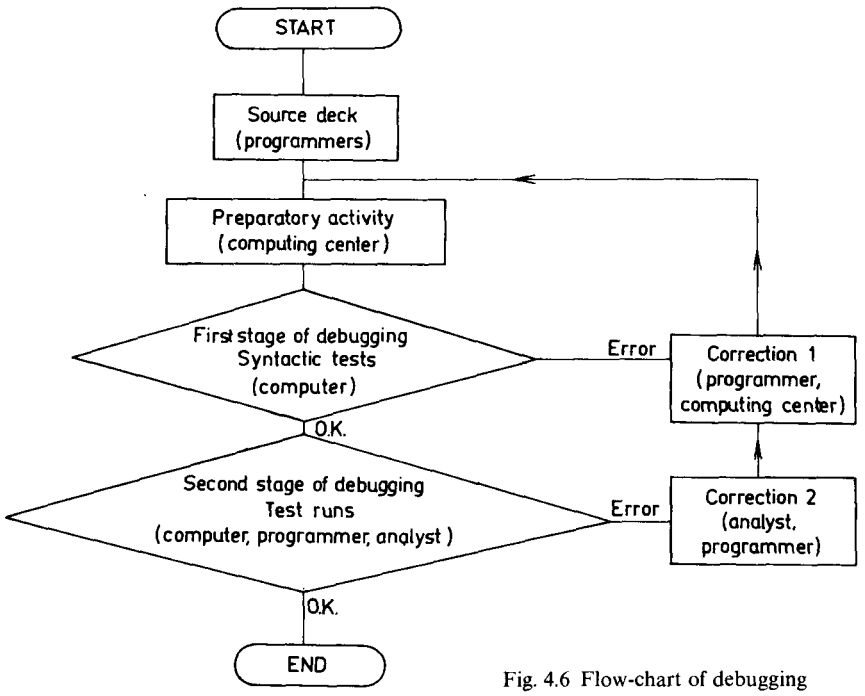

Fig. 4.6 Flow-chart of debugging

Source code

The result of the analytical modelling procedure is the flowchart (decision tables etc.) that serves the programmer in assembling the program (i.e. in producing the source deck).

Preparatory activities

The program and the input data must be transformed into a form suitable for the input device of the computer, e.g. it is punched into the punch cards or paper tape or written on the keyboard of the terminal device. Before execution of the program on the computer the operator or, in an automatic system, the supervisor (type of system program) prepares some system programs that make provisions for debugging.

First stage of debugging

The source code is transformed by the compiler into a computer code. Syntax errors in the source language are detected. In automatic programming languages the syntax tests are perfect and very detailed. Often the compiled program is printed on the lineprinter and the errors detected are numbered by the number of the probable error, or a short explanation is given. Using this information the programmer can correct the error on the input medium.

Compilation is often performed in two stages. In the first stage, the source program is compiled in the inner code, which is different from the computer code. In the inner code the standard unit, i.e. components of the standard software are added. The second stage links all the units that will be used in computation together, and the compilation is completed. Syntactic tests are a necessary by-product of these activities.

Second stage of debugging

If formal errors have been corrected, test computer runs are performed. First, possible mistakes that can occur during computation are corrected (e.g. overflow of the acceptable range of the number in fixed-point and floating-point representation, exceeding the range of subscripts defined in declaration, etc.). The search for the source of these errors is tedious, and diagnostic test programs are therefore included in the compiler, specifying the type of error, the variable name and the place in the program (number of block, number of line, etc.) where the error might occur. Then the tests and analysis of the output data follow. Special test data should be prepared for this purpose, based on the task analysis; a quick test of the correctness of the results is thus possible (e.g. by comparison with the results obtained by a different method of computation). If the computation results do not correspond to this prepared test output, the errors are detected and corrected as follows;

a) without the computer:
   - test of the program by another programmer searching for the coding error in the source program,
   - simulation of a computation run by a calculator using the test data and following the program instructions;

b) with the computer:
   - print of the intermediate results (in the simulation model e.g. the changes in important variables at the end of the time step),
   - program tracing, i.e. print out of the locations where the computation was executed,
   - memory dumps, i.e. a record of memory contents when execution stops.

The method of debugging corresponds to the complexity of the program. Test data are designed to activate the execution of all the branches in the program.

Debugging of programs for computers of the second generation was done in the presence of the programmer or the analyst. Debugging of programs for computers of the third generation is often done via the terminals and the programmer can correct programs in a time-sharing and interactive mode. Not only computer time but also the programmer's time is used more effectively.

Program optimization is concerned with changes to shorten the required computer time or reduce the number of the necessary memory locations. A program must be optimized if it is used many times.

4.2.5. Documentation in Programming

The work of programming does not end with debugging, testing and obtaining the first correct results. Documentation should be an integral part of each program. Often, in some isolated tasks, documentation is omitted due to lack of time, resulting in losses of computer time and programmer's time with future use of this program. Incorrect data use and false assumptions are associated with changes and new arrangements in the program.

The first part of the documentation, which is necessary for the user of the program, should include the following information:

- the name and purpose of the program,
- different possibilities of program utilization,
- the program language used,
- the type of computer that can be used for this program,
- the necessary system configuration (necessary for computation),
- instructions for input data preparation,
- a survey of output data,
- instructions for the operators of the computer system.

The second part of the documentation is important for the analysts and programmers and should contain a detailed description of the program. The form of this description includes the flowcharts (decision tables etc.) and the program printout on the lineprinter. A list and a description of the most important identifiers are helpful.

The flowchart has to correspond to the present state of the program and should be comprehensible to programmers who will later deal with this program. Each change should be documented, otherwise the flowchart is useless.

Often, changes in the program are required. The user requires new input and output data (sometimes due to inadequate system analysis) or he has obtained some new information on the problem. The computer centre may require interruption of the computation with continuation later. The analyst may find an improvement in the logical aspects of system description or ways of speeding up computation or of obtaining more accurate results. The changes are often necessary and advantageous; however, they must be properly documented.

4.3 PRINCIPLES OF A COMPUTER CENTRE PROJECT

The computer centre costs run into millions, and the time needed for an economic return on the costs should be short-term. Therefore, a project for a computer centre and its input data should be thoroughly analysed.

If a computer centre system is to be designed, previous experience with a hired computer in a service computer centre is very important. The danger of the importance of systems analysis of the computer centre project being understimated is thus reduced. Utilization of a service computer centre before the installation of an institute's or a firm's computer centre is therefore recommended. Service computer centres can be classified into two groups:

(a) *analytical computer centre* similar to a consulting engineering service, where the customer orders the solution of his problem from the data delivered. These data and the problem are analysed (with the aid of the computer), methods of computation and data processing are designed, and programs are coded, debugged and prepared for execution;

(b) a *service computer centre*, where all the work described above is done by the customer. The computer centre is in charge of the operation of the computer, punching and testing of the transformation of input data for the computer, and the customer hires these services and the computer time.

In the design of a computer centre the following stages can be distinguished:

In the *first stage* the problem is defined and the requirements formulated. The computer tasks are specified, the method of data processing is stated and the cost-benefit analysis performed. The type of computer need not be decided upon at this stage. A comprehensive analysis of the development and future activities is all-important. The computer will be used not only for water management and scientific problems but also for business data processing, for management information systems and possibly for the operation of WRS. It is a very important, time- and labour-consuming activity and the management has to assemble a qualified and competent working group for it.

In the *second stage*, a survey of the input and output information and the system charts for the principal tasks are set up (at a rough discriminating level). By using this information in the project of the new system of data processing and computation, the requirements for the computer and the peripherals are derived (input/output devices, capacity of the main computer storage, auxiliary storage, software, etc.).

The *third stage* involves the choice and purchase of the system comprising the computer, input/output and other devices. In this choice, devices that have been purchased already and their role in the future system are taken into account. The technical parameters of the system hardware in a certain class are approximately the same, and the costs do not differ very much, but the software (compilers for the

programming languages, subroutine packages, standard reference library, etc.) and the whole conception of the system (reliability of the system, prompt service by the manufacturer, flexibility in extension of the system, etc.) are often decisive. It is advantageous to obtain information from firms which deal with the same or similar problems or which own the same type of computer.

In the *fourth stage*, personnel are trained, especially the programmer and the service personnel and the programs are prepared, debugged, executed and tested in a service computer centre.

The *fifth stage* comprises the installation of the computer, tests of the system and its operation.

The activities need not always be performed in the stated order as some feedback loop may occur.

The professions represented in the computer centre include the following categories:

– *management of the centre*: director, secretary, clerical services, etc.

– *analytical and programmer's group*: the head of the group, analysts, departments for business data processing, scientific and technology tasks, econonomic problems, specialists, etc.

– *technical service group*: head of the group, service personnel,

– *operation group*: head of the group, heads of the work shifts, operators,

– *data preparation group*: personnel for punching data, typing in display stations, data control, etc.

The approximate proportions of these professions in the computer centre may be as follows: Director and heads of groups 5%, analysts 25%, programmers 37%, operators 25%, technical service personnel 8%.

In the choice of the personnel of the computer centre cooperation with psychologists is recommended. The work of the analysts is similar to that involved in management positions, but it is more diversified; a comprehensive evaluation of the analyst's psychological characteristics is therefore desirable. Programmers should possess the ability to combine information, a good insight into problems, a flexible way of thinking, etc. Operators have to be accurate and reliable. The personnel for data punching, typing, and testing should be able to concentrate, should possess good pattern and character recognition combined with a short-time but accurate memory capability. The results of the psychological tests should be supplemented by information on mental qualities, and the ability to cooperate in working groups, high motivation and a positive attitude to work, psychological stability, etc. are all important factors. The dynamic process of the transformation of these psychological qualities due to the environment and type of work needs to be taken into account.

4.4 PROSPECTS OF COMPUTER USE IN WATER RESOURCE SYSTEMS

Simulation models are often used for the design and operation of WRS. For this purpose a number of mathematical optimization models (e.g. linear or dynamic programming) have also been used. All these models use computers, not only for modelling, but also for processing hydrological input data and for the extraction of more information from the observations. If the data have been processed, corrected and transferred to a medium suitable for computer input, no further errors due to clerical oversight, reproduction failure, etc. occur.

Computers are used in hydraulic structure design computations and other fields related to water management.

At present, the development of interrogative computer systems, linked by terminals with the data base of big computers, seems to be the most progressive trend in WRS analysis and design. The computers, hardware and software can thus be used not only by programmers but by water resource engineers, planners and decision-makers and a high efficiency of work and extensive application of new ideas is possible. These users need not know in detail the algorithms of computation, programming and other activities that can be transferred to the computer system. The interrogative computer system can be used in an interactive way for the evaluation of different assumptions in project alternatives using more relevant information than could be used without this system.

The methods of business data processing by computers can be used in water balance evaluation. Some types of computers will be used for real-time control of releases from a system of reservoirs.

The information and control systems will include observation sites with remote control, data transmission, system monitoring and computer control. This system has to monitor the states of WRS, evaluate their effect on operation (even with a short-term forecast of flows) and, based on them, the system can offer a proposal for an optimal method of operation (WRS-real-time dispatching). The first stage of computer use in this system, i.e. formation of information subsystems, has been implemented in several WRS. Information retrieval systems (see Chapter 11) are a further aspect of computer use in water management with cost saving in the retrieval of information and a basic reduction in the time necessary to gain access to information about publications, reports, books, papers, etc. in water management.