50X1-HUM

INFORMATION REPORT INFORMATION REPORT

CENTRAL INTELLIGENCE AGENCY

S-E-C-R-E-T

50X1-HUM

COUNTRY USSR

REPORT

SUBJECT Publication on Automatic Programing at the Computing Center of the Soviet Academy of Sciences ARSST

DATE DISTR. 26 September 1961

NO. PAGES 1

REFERENCES RD

DATE OF INFO.

PLACE & DATE ACQ.

50X1-HUM

THIS IS

1. a 28-page, English-language document, by A.P. Yershov, entitled The Works of the Computing Center of the Academy of Sciences of the USSR in the Field of Automatic Programing. Moscow. 1958. The publication contains a general discussion of computer programing, programing for the STRELA-3, and problems arising in programing.

50X1-HUM

50X1-HUM

S-E-C-R-E-T

STATE	ARMY	NAVY	AIR	NSA	OCR	NIC	
x	x	x	x	x	x	x	

(Note: Washington distribution indicated by "X"; Field distribution by "#".)

Academy of Sciences
of the USSR
Computing Centre

FOR OFFICIAL USE ONLY

THE WORKS OF THE COMPUTING CENTRE OF THE ACADEMY OF SCIENCES OF [illegible] IN THE FIELD OF AUTOMATIC PROGRAMMING

by
A.P. ERSHOV

M O S C O W
1 9 5 8

THE WORKS OF THE COMPUTING CENTRE OF THE ACADEMY OF SCIENCES OF THE USSR IN THE FIELD OF AUTOMATIC PROGRAMMING

by Ershov A.P.,
chief of the theoretical programming department
of the Computing Centre of the AS of the USSR

1. Preliminary researches

The real beginning of the automatic programming development in the Computing Centre (like as in the Soviet Union) is the year 1954. Just this year the first preliminary works appeared which contain the main ideas and their realization about which I am going to say at first.

The main necessity, which rised in programming from the very beginning, is the separation of the programming process into two main stages. On the first stage the programmer writes a detailed, but sufficiently lookable, computing plan of problem solving. On the second stage the direct programming (coding in your terminology) is carried out, which presents itself the writing of this detailed computing plan on the computer's language.

Thus, in the course of programming, the necessity of construction of the intermediate language rises out. This language is the language, by means of which the results of the first stage of programming (the detailed computing plan of problem solving) are written down.

At the beginning in the Soviet Union as well as in any other countries such an intermediate language was the flow-charts language, which was used still by John von Neumann and Goldstine.

But in the years 1952-53 prof.A.A.Ljapunov developed a new programming method, which was first described on his lectures in the Moscow University. This method had a great influence on the development of the programming in the USSR, is perfectly acknowledged nowadays and is known as the operator programming method.

The main conception of the operator method is the consideration of the program as a complex operator which acts with initial datas and consists of elementary operators of various types. The main types of elementary operators, acting with the initial datas, are arithmetical operators which fulfill the direct datas transformation and logical operators which determine the next direction of computations.

The conception of the depending of the operator on some integer parameter (for instance the depending of the adress of the indexed number on the values of these indexes) was also very important.

Besides this, the control operators were considered. These operators do not act with the initial datas but deal with the program itself. I shall mention at first the re-adressing operators which exchange the depending on the parameter adresses while the parameter's value is exchanging. The restoration operator establishes the depending on the

parameter operator to its initial view according to the initial value of the parameter.

The detailed computing plan's record written as the sequence of the operators of various types is called the operator scheme of program (or the logical scheme of program).

This conception was briefly described at first in the book /2/ (pp.193-206).

Ljapunov's operator method was very important for the development of the automatic programming because of the four main reasons:

1. For the first time the program's components (elementary operators) were formally classified according to their functional appointment, independable on the sense of the problem programmed. This permited to formalize the logical scheme program's language easily and make the language universal for writing problems to be solved.

2. The language of logical schemes of programs was easy to algebraize. The latter was very important for the direct input of the program's logical scheme into the computer.

3. The logical scheme of program as an intermediate result of the programmer's work, successfully separates the programming process. The composition of the logical scheme accords to the "intellectual" work of the programmer while the most labour and mechanical work falls on the stage of the transition from the logical scheme to the object program. Thus, the stage of the programming process, which is the

easiest to automatize, was separated.

4. The partition of the whole problem on the sequence of separate operators permits the programmer to divide the programming process on programming of separate elementary operators. This also makes the coding more easy.

Still in 1953 prof.Ljapunov supposed that some parts of the second stage of programming could be automatized.

The first works in the field of automatic programming in the Soviet Union were carried out in the Mathematical Institute of the AS of the USSR and in the Computing Centre of the AS of the USSR, which at that time was a division of the Institute of Exact Mechanics and Computing Technique of the AS of the USSR.

In summer 1954 in the Computing Center V.M.Kurochkin composed a program for the BESM computer, which permited to fulfill the symbolic programming. L.M.Korolev composed the first version of the program for BESM, which carried out the programming of arithmetical formulas.

G.S.Bagrinovskaja from the Mathematical Institute described a project of the program, fulfilling the programming of readressing operators. In summer 1954 E.Z.Lubimsky and S.S.Kaminin constructed the programming program for the STRELA computer (PP-1), which performed the programming of readressing, restoring and logical operators.

All these works were only experimental but it was a necessary stage on the way to the design of big programming programs.

2. The first programming programs

The first project of the programming program (PP-2), which fully automatize the transition from the logical scheme to the object program, was described by E.Z.Lubimsky and S.S.Kaminin on the prof.Ljapunov's seminar in october 1954 /3/.

Source information for this PP consists of the logical scheme of the program and the table of the storage distribution. The logical scheme may include arithmetical and logical operators, readressing and restoring operators and so called non-standard operators, which presents some peaces of the object program written in commands with symbolic adresses.

In February 1955 a group of the Mathematical Institute stuff-workers constructed a working PP for the STRELA computer on the basis of this project. This PP was described in a series of articles /4/.

In December 1954 a project for another PP was carried out. Some versions of the PP were developed on the basis of this project; the final one was constructed to March 1956. This PP is described in my book /5/.

I want to point out the main differences of the PP for BESM from the PP-2 for STRELA.

a) In the PP for BESM the hierarchy of arithmetical operations when programming arithmetical formulas is taken into consideration.

b) In PP for BESM the logical operators used are of another type than those in the PP-2.

c) There are no readressing and restoring operators in the source information for the PP for BESM, but some special notations, pointing out the necessity of the cyclic repetition of some operator's group, take place.

d) In the PP for BESM the storage is distributed automatically.

The first public communications about the first programming programs were made by the authors in their reports on the All-Union Conference "The ways of the development of the Soviet mathematical computing machinery" held in Moscow on the 12-17 of March, 1956.

3.The programming program for the STRELA-3 computer

The main result of the Computing Centre's works in the field of automatic programming is nowadays the programming program for the STRELA-3 computer. This computer is installed in the Computing Centre.

In autumn 1956 the work on the construction of the programming program for the STRELA-3 (PPS) began. It has been finished in autumn 1957 and performed by six stuff-workers of the theoretical programming department.

When the requirements for the source information were developed we were intended to realize the following aims:

a) The representation of the source information has to be close to the mathematical formulation of the problem.

b) The size of the auxiliary and technical work, not connected with the mathematical formulation, has to be reduced to minimum.

c) The source information must give the full information about the structure of the object program.

d) The source information has to be maximally compactable and lookable.

The source information prepared by the programmer consists of five parts:

1. The program's scheme (the main part of information)
2. The carried away operators
3. Information about variables
4. Information about storage blocks
5. Blocks.

Some parts of the source information may be absent.

A little about terminology. A variable is a letter symbol which is contained in the source information. It may denote some mathematical variable or constant or, in general, any machine word. A storage block means any group of locations which has to be placed in the storage one after another.

First about the 3-th and 5-th parts. In the 3-th part all the variables, which require some additional information, are written down. Such an additional information may be:

a) A constant, if the variable represents a constant. Four types of constants may exist: decimal flow-point constant, binary flow-point constant, command with symbolic adresses and command with real adresses.

b) An adress, if you wish to prescribe the location of the variable in the storage.

The 5-th part includes any datas placed into blocks and introduced into the storage by the object program itself.

The program's scheme. In the program's scheme all the operators of the object program are written down one after another. The order of placing of the operators in the scheme corresponds to their placing in the object program. The scheme may include operators of the folowing types:

Name of the operator	Symbol denoting the operator's type
1. Arithmetical	А
2. Logical	Л
3. Restoring	В
4. Non-standard	Н
5. Readressing	П
6. Doubl-counting	ДС

As a rule, every operator is represented by a letter denoting the operator's type, after which (in parenthesis) the information about the operator follows. Only for the arithmetical operators the letter symbol and parenthesis are omitted.

It is possible to require in the program's scheme the syclic repetition of some sequence of operators. For this purpose such a sequence is put into a figure brackets: { is an opening bracket of the sycle (a loop in your terminology) and } is a closing bracket of the sycle.

Beside this, special symbols may appear in the source information; they denote the command, which transmit an information between computer's parts, and transfers.

Every operator may posess a number. The number is put by a letter denoting the operator's type as a subscript.

Every operator can be carried away from the program's scheme into the second part of the source information. All the non-standard operators and some complicated arithmetical ones are carried away into the second part. In this case on the place of the operator carried away only a letter denoting the type with the number is left in the scheme.

The main cause, which makes the source information lookable and obvious, is the successful solution of the problem of the recording of arithmetical formulas and the use of the obvious symbolism.

Arithmetical operators. Arithmetical operator fulfills the evaluation by the help of the sequence of formulas as a

$$F(x_1,\ldots,x_n) \rightarrow y,$$

where F is a superposition of operations from some fixed list of operations and $x_1,\ldots,x_n$ are variables and constants. In particular, this list contains all the operations

of the STRELA-3. y denotes the result of the evaluating of th formula. The constants in formulas are represented either by letters or by numbers. The variables may have any number of letter subscripts. In this case every subscript must be a parameter of some loop.

I shall give (with some abbreviations) the list of operations used in the arithmetical operators.

The list of the operations

Name	Symbol
operations with two arguments	
addition	+
substraction	-
addition by modulo 2 (digit by digit)	. +
logical addition (digit by digit)	v
multiplication	x or ·
logical multiplication(digit by digit)	^
operations with one argument	
integer part	E
fractional part	D
absolut value	Mod
signum	sign
sinus	sin
cosinus	cos
arcsinus	arcsin
arccosinus	arccos
arctangent	arctg
exponent	exp

Name	Symbol
natural logarithm	ln
binary to decimal conversion	Dec
decimal to binary conversion	Bin
logical negation (digit by digit)	¬

powers:

x^{-1}, x^2, x^3, $x^{\frac{1}{2}}$, $x^{-\frac{1}{2}}$, $x^{\frac{p}{2^q}}$ $(1 \leq p \leq 511, 0 \leq q \leq 7)$

The usual hierarchy of operations useful in mathematics is used for the determination of the order of fulfilling of the operations. There is only one difference between the marked and non-marked multiplication (x and ·), which is obvious from th. 'ollowing example:

$$\underline{\text{tg}\,\underline{\cos 2x^2}} + \underline{\sin\underline{(a+b)(a-b)}} + \underline{\sin\underline{(a+b)}\times(a-b)} - \underline{\text{Mod}\,\underline{y^{-1}}}\,\underline{\text{sign}\,\underline{(x+y)}}\,\underline{\text{tg}\,\underline{a^2}}.$$

In this example the terms, which are arguments of the corresponding operations, are underlined. Besides this, the following record is allowed: $\sin^m x$, which denotes $(\sin x)^m$.

Our experience shows that all the transformations of formulas, which are carried out by the preparation of source information, are thus: the omitting of radixes and lines, denoting the division, and replacing them by appropriate powers.

Logical operators. One logical operator realizes the varifying of one of five standard logical relations. The standard logical relations are following:

$$a < b$$

$$a \leqslant b$$

$$|a| < |b|$$

$$|a| \leqslant |b|$$

$$a = b ,$$

where a and b are variables or constants.

In the program's scheme logical operators are represented as

$$\prod \left(a \sim b \begin{array}{|c} N_1 \\ \hline N_2 \end{array} \right)$$

where N_1 and N_2 are operators' numbers, $a \sim b$ is one of the five logical relations. This operator acts in such a way: if the relation $a \sim b$ is true then we transfer to the operator N_1, otherwise we transfer to the operator N_2. If one of the operators' numbers is omitted, this means, that we transfer to the operator next after the logical operator.

Non-standard operators. A non-standard operator is an any part of object program, written in commands with symbolic adresses. There are many rules of writing down the non-standard operators but they are of no interest, and I shall omit them.

As it was mentioned above, the non-standard operator is always written down in the second part of the source information (carried-away operators). In the program's scheme the

non-standard operator is represented by a letter H with the number. In the second part a "head" is placed before the non-standard operators, which has a form

N	n		50

where N is the non-standard operator's number, n is a number of words the operatorcontains.

The loops. A loop realizes the repeated fulfillment of some sequence of operators for all given values of the parameter of the loop.

In the program's scheme the loops are represented in such a form

$$\left\{ A \right\},$$

where $\{$ and $\}$ are the opening and closing brackets of the loop, A is the sequence of the operators repeated. The loop's parameter i and its initial value i_M, if it is not equal to zero, are placed under the opening bracket :

$$\underset{i=i_M}{\{} \quad \text{or} \quad \underset{i}{\{} \qquad (\text{if } i_M = 0)$$

If the number of repetitions of the loop is determined by the final value of the parameter i_K then the latter is placed over the opening bracket :

$$\overset{i_K}{\underset{i=i_M}{\{}} \quad \text{or} \quad \overset{i_K}{\underset{i}{\{}}$$

In such a case the loop will be repeated from the value i_H up to the value i_K includingly by the step of the size 1 ($i_H \leqslant i_K$).

i_H and i_K are either variables or non-negative integer constants. If i_H and i_K are variables, then it is supposed that they are evaluated by the object program before running of the loop as non-negative integer normalized numbers.

It is also possible to set the number of the loop's repetitions by means of one of five logical relations a ~ b.

The readressing operators. A readressing operator is recording in the program's scheme as

$$\Pi\ (N,\ n,\ h),$$

where N is an operator's number; n is either a number of a command of a non-standard operator or a variable; h is either a variable or an integer constant.

The readressing operator is fulfilled thus:

a) If n is a number of a command of the non-standard operator N, then the readressing constant h is added to the command.

b) If n is a variable and h is a constant, then h is added to all the adresses of the variable n in the operator N.

c) If n is a variable and h is also a variable, then it is supposed that the value of the h needed is evaluated in the third adress of the location for h as an integer number.

The restoring operators. A restoring operator is recorded in the program's scheme as

$$B(N)$$

where N is an operator's number.

If there are some variable commands in the operator N which are transformed by readressing operators, the restoring operator will transfer the initial values of these variable commands into their places in the operator N .

The double-counting operator. While composing the program with the help of the PPS we may prescribe the control of the computation by means of double computations of separate peaces of the program. Therefore the double-counting operators' symbols are arranged in the program's scheme. The computer will compute every peace of the program between two such symbols (following one after another) and accumulate all the contents of the internal storage. If the sums are equal, then the whole content of the storage is recorded on the magnetic tape, and the computation is carried on further. If the control sums are not equal,the computer stops and, after a new start, repeats the computation once more comparing after that three sums already.

Special symbols for transfer and transmission of information between computer's parts. The following notation, which denotes the transmission of information between the different computer's parts, is used:

$$A \xrightarrow{n} B$$

where A is the symbolic adress of the beginning of the place, from which they are going to transfer the informatio B is the symbolic adress of the beginning of the place, where they are going to transfer, and n is a number of words in the information transferred.

The unconditional transfer is denoted by

$$\left[\begin{matrix} N \\ N \end{matrix}\right.,$$

where N is a number of an operator to which we shall transfer.

The conditional transfer is denoted by

$$\left[\begin{matrix} N_1 \\ N_2 \end{matrix}\right.$$

If the signal ω for the preceeding command is equal to 1, we shall transfer to the operator N_1. Otherwise we shall transfer to the operator N_2.

For transfer to subroutines the following notation is used:

$$\left\{\begin{matrix} N_1 \\ N_2 \\ N_1 \end{matrix}\right.,$$

which means: to fulfill the subroutine beginning from the operator N_1 and up to the operator N_2 exclusively.

Information about storage blocks. This information, at first, shows how many blocks there have to be in the storage and how long they should be. If it is necessary, it is possible to point out the position of some blocks in the storage. Such an information about every block has the following form

M_k , λ words, $\nu = \ldots$

where k is the block's number, λ is its length and ν is the adress of the first location of the block (if it is given).

Then the information about the variables concerned to this block is written down.

If the adress of a variable a or $a_{ij\ldots}$ depends on parameters $i,j,\ldots$, it is recorded in such a form

$$a(\text{or } a_{ij\ldots}) = H(\text{or } K) + \Delta + h_1 i + h_2 j + \ldots$$

where H denotes the beginning of the block (its first location) and K denotes the end of the block (its last location). Number Δ is a "shift" in our terminology, h_1, h_2,... are the steps of readressing according to parameters $i,j,\ldots$. Δ and h may be either integer constants or variables.

If the adress of the variable a from the block does not depend on single parameter then the information about its place in the block is recorded by

$$a = H(\text{or } K) + \Delta .$$

If some variable appears in the program's scheme with different sets of equal number of subscripts then the dependness of the adress of the variable on the subscripts is mentioned in the information only once. For example, if the variables $a_{isk}, a_{kli}, a_{ksl}$ appear in the program's scheme, then in the information about blocks only one row is written down

$$a_{ijk} = H(\text{or } K) + \quad + h_1 i + h_2 j + h_3 k.$$

That is all about source information.

I shall briefly describe the structure of PPS. The PPS as a whole consists of 30 separate blocks recorded on the magnetic tape. Every block works only once during the programming program's running. The information transformed is placed in the internal storage. Every block has the length of about 240 locations. A part of the storage (about 130 locations) is intended for some working materials, which the PPS works out and overgives from one block to the other. Thus the length of the simultaneously transformed information may arise up to 1500 words. The blocks of the PPS change one after another automatically. The PPS punches on punch-cards the ready object program and some additional information. Besides this, it is possible (if you wish) to punch some intermediate information.

PPS is able to discovery automatically many formal mistakes in the source information.

In conclusion of this part let us consider an example of preparation of the source information.

Suppose we have to integrate a parabolic partial differential equation

$$\frac{\partial z}{\partial t} = 0{,}75 \sqrt{x(1-x)(t^2 x+z)}\, \frac{\partial^2 z}{\partial x^2}$$

$$z(x,0) = 0;\ z(0,t) = 0;\ z(1,t) = t$$

from $t = 0$ to $t = T$, $M = h^{-1}$ and τ are given (h is the step by x, τ is the step by t).

The evaluation formulas are following

$$z_{0,n+1} = 0$$

$$z_{m,n+1} = z_{m,n} + \tau \cdot 0{,}75\sqrt{x(1-x)(t^2x+z_{m,n})}\,\frac{z_{m+1,n}-2z_{mn}+z_{m-1,n}}{h^2}$$

$$m = 1,2,\ldots,M-1$$

$$z_{M,n+1} = (n+1)\tau$$

$$n = 0,1,2,\ldots$$

Let us pick out one block of the storage, containing 1000 number for storing all the values of z for each layer of t. The new z we shall place on the places of the old ones with delay for one step, because the old values of z will be needed for the evaluation of the new ones. Every time three points z_m, z_m^- and z_m^+ will be used, which correspond to $z_{m,n}$, $z_{m-1,n}$, $z_{m+1,n}$.

The source information is thus:

1. The program's scheme

$$\rightarrow M \rightarrow \tau \text{ ДС(5) } \left\{ 0 \rightarrow z_m \right\}_{m=0}^{M} \left\{ n\tau \rightarrow t \left\{ \Pi(m=0\square)_{N_1} \right\}_{m}^{M} \right\}_{n}$$

$$0 \rightarrow r\square_{N_2}^{N_2} \Pi_{N_1}(m=M\square_{N_3})\,(n+1)\tau \rightarrow z_m\square_{N_4}^{N_4} \sqcup_{N_3} mM^{-1} \rightarrow x$$

$$z_m + 0{,}75(x(1-x)(t^2x+z_m))^{\frac{1}{2}}(z_m^+ - 2z_m + z_m^-)M^2 \rightarrow r$$

$$\sqcup_{N_4} s \rightarrow z_m^- \sqcup_{N_2} r \rightarrow s \Big\} \text{ ДС} \Big\} \left\{ \text{Dec } z_i \rightarrow z_i \right\}_{i=1}^{M} \Big\} a \xrightarrow{1000} \text{STOP.}$$

The logical operators here are picking out the cases $m = 0$ and $m = M$, which are evaluated separately. The sign $\lrcorner$ is used to show the number of the next operator, if the latter does not posess a letter denoting the operator's type.

2. Carried away operators are absent.

3. Information about variables is absent.

4. Information about blocks of storage is following

M_1, 1000 words

$z_m = H + 1 \cdot m$

$z_m^- = H - 1 + 1 \cdot m$

$z_m^+ = H + 1 + 1 \cdot m$

$a = H$.

5. Blocks of storage are absent.

4. Some problems of the automatic programming.

The problems I am going to say about are only a little part of the great number of problems of the automatic programming, but they characterize our interests clear enough. I will consider two types of problems:

A) The problems of the source information and

B) The problems of construction of new programming algorithms.

A). The problems of the source information. This name unificates all the problems, which solution simplifyes the

construction of the source information and approaches its form to the mathematical form of the problem, using all the richness of the modern mathematical symbolism.

The natural stage in this direction is the expansion of the symbolism, admissible in the source information, by adding some widespread mathematical terminology. The latter may include some sorts of mathematical operators, such as summation, the evaluation of multiplication, differential operators, and so on, or some functionals, such as $\min f(x)$ $\max f(x)$, $\int_a^b f(x)dx$ and so on, some widespread special functions. And, at last, we may permit to include into the source information not only real scalar numbers but more complicated such as complex numbers, vectors or matrixes and so on.

The method of subschemes (already used in some programming programs) may be used for decoding of new symbols which take place in the source information. The essence of this method is following. A special block is added to the PP which may "understand" new symbols in the source information. For every symbol of that sort, which denotes some mathematical algorithm D, this block constructs a subscheme which represents the record of the algorithm. This record is the algorithm D in terms of standard operators (arithmetical, logical and so on). The subscheme constructed substitutes the symbol, denoting the algorithm D, into the source information. After this the program's scheme transformed,

which already does not contain special symbols, is transformed by PP in a general way.

To make a long story short, I am speaking about the unification of the PP programming method with the compiling programs' method. Though the principal possibility of such an unifications has been discussed long ago, there is little done in this direction.

The problems of constructing of new programming algorithms

From the broad field of problems of that kind I shall consider only two, perhaps, the most important from the practical view.

1. The analysis and transformations of the program's schemes. The term "analysis of the program's scheme" denotes the construction of algorithms, able to get the various information from the program's scheme about the interconnections between the operators forming the scheme.

Before now the programming programs were constructed in such a way, that every operator from the scheme was programmed as possible independently on the other ones. For example, the economy of commands during the programming of arithmetical operators is performed only in one operator. By the programming of loops it requires from all variable commands, which depend on the loop's parameter, to be placed in the scheme between the first operator and the last one of the loop. The independent programming of various operators allows us to simplify the programming algorithms but escapes some opportunities of the improvement of the program constructed.

I shall give an example of the improvement of the programming algorithms requiring an analysis of the program's scheme.

Let us suppose that in some arithmetical operators $A_1, \ldots, A_n$ we meet one and the same term $F(x_1,\ldots,x_k)$. It is obvious that, in some cases, $F(x_1,\ldots,x_k)$ may be programmed in one operator only, while in the others the term $F(x_1,\ldots,x_k)$ is replaced by the result's symbol of the term's evaluation. This may be done in the case if there is an operator A_i among $A_1,\ldots,A_n$, which is fulfilled always before the others and if $x_1,\ldots,x_k$ do not appear as the results in the computation between the work of the operator A_i and A_j (A_j is one from $A_1,\ldots,A_n$). It is clear, that it is impossible to discovery the possibility of such a generalized economy of commands without the analysis of the program's scheme.

The problem of the analysis of the program's scheme may be formulated in the particular case more exactly in such a way:

To construct an algorithm, which for any arbitrary chosen two operators A_i and A_j from the scheme, determines, which one of the possibilities takes place

a) A_i is fulfilled always before A_j ;

b) A_j is fulfilled always before A_i ;

c) A_i and A_j are not simultaneous, that is, if one is fulfilled, the other is not;

d) A_i and A_j can be fulfilled in any relative order.

The construction of such an algorithm would be very important for programming.

In connection with the inclusion of such notations as the summation term Σ and the multiplication term Π into the source information, there appears an interesting problem of the program's schemes' transformation.

Let us consider as an example the problem of evaluating the series

$$u = \sum_{k=0}^{n} f(k)$$

with some given accuracy. It is obvious, that if we substitute this notation directly by the subscheme

$$0 \Rightarrow u \left\{ u + f(k) \Rightarrow u \right\}_{k}^{n},$$

the object program would be very bad. If we shall compute according to this program, we have to evaluate $f(k)$ for every new k anew, and this is extremely improfitable in most cases. In fact, we usually reduce the direct evaluation of $f(k)$ to the evaluation of $f(k)$ by use of some recurrent relations.

For instanse, the series

$$u = \sum_{k=0}^{n} \frac{x^k}{n!} .$$

is always evaluated according to such a scheme

$$1 \Rightarrow t, 0 \Rightarrow u \left\{ u + t \Rightarrow u,\ \frac{t \cdot x}{k} \Rightarrow t \right\}_{k=1}^{n} .$$

It seems possible to us to solve the following problem. For any arbitrary function $f(k)$ of an integer argument, represented by a combination of the exponent, power and factorial functions, to find an algorithm, which replaces the close notation $f(k)$ by the recurrent relations. The realization of such an algorithm in PP would permit the programmer not to fulfill the preliminary work of finding the recurrent relations but to replace this work to the computer. The programmer has only to write down the common member of the series or the multiplication in a close form.

In connection with the programming of loops the following problem also concerning to the transformation of the program's scheme is interesting. In many problems (particularly by the evaluation of tables) the following prescription appears:

"Evaluate the table of the values of the function

$$z = F(p,l,m,n)$$

(where p is a parameter) for all sets of the values of the integer arguments, for which

$$l_0 \le l \le l_K,\ m_0 \le m \le m_K,\ n_0 \le n \le n_K."$$

(The number of arguments is of no importance and can be arbitrary).

It is very easy to write the program's scheme in such a form:

$$\left\{ \begin{matrix} l_K \\ l=l_0 \end{matrix} \right\} \left\{ \begin{matrix} m_K \\ m=m_0 \end{matrix} \right\} \left\{ \begin{matrix} n_K \\ n=n_0 \end{matrix} \right\} \left\{ F(p,l,m,n) \to z,\ \text{punch } z \right\} \Big\} \Big\} \Big\}.$$

But if, for instanse, $F(p,l,m,n)$ can be represented in such a way:

$$F(p,l,m,n) = \varphi(l,\psi(m,\xi(n,\zeta(p)))),$$

then the scheme

$$\zeta(p) \to u\left\{\xi(n,u) \to v\left\{\psi(m,v) \to w\left\{\varphi(l,w) \to z,\ \text{punch } z\right\}\right\}\right\}$$

$$n = n_0,\ n_K \qquad m = m_0,\ m_K \qquad l = l_0,\ l_K$$

would be, of course, much more profitable. That is because, according to this scheme, only that part of the function F is being evaluated in every loop, which is essentially depending on the parameter of this loop. The evaluation of the left part of F is carried out into the external loops, which repeat more seldom. Therefore the program as a whole works faster.

In this case it is also probably possible to find an algorithm, which would perform the transformation of the program's scheme mentioned above.

2. The increasment of the velocity of the PP's work. The new PP obtain the more and more opportunities in order to the quality of the object programs by the expansion of the complication of the programming algorithms. But the increased complication and the size of PP have their shadow side, which turns out in the increasment of the programming time. When the running of PP offers much time, it reduces the

effect of using of the PP (in particular, when the computer works unstable). Therefore I would like to stress the fact, that it is necessary while constructing new algorithms of the programming, to aspire in every possible way the increasing of the velocity of the algorithm's work. One my work /6/ testifies some possibilities in this direction.

Reference

/1/ A.A.Ljapunov. "O logičeskikh skhemakh program" (On logical schemes of programs). Problemy kibernetiki, Vypusk 1 (Cybernetics problems, issue 1). Moskva, Fiziko-matematičeskoje izdatelstvo, 1958.

/2/ A.I.Kitov. "Elektronnye cifrovye mashyny" (Electronic digital computers), Moskva, Sovetskoe radio,1956.

/3/ S.S.Kaminin, E.Z.Ljubimskiy, M.R.Shura-Bura. "Avtomatizacija programmirovanija c pomoščju programmirujuščej programmy" (Automatization of programming by means of a programming program). Problemy kibernetiki, Vypusk 1.

/4/ E.Z.Ljubimskiy. "Arifmetičeskij Blok v PP-2" (The arithmatical block in the PP-2).
S.S.Kaminin. "Blok pereadresacii v programme PP-2" (The readressing block in the PP-2 program).
E.S.Lukhovickaja. "Blok obrabotki logičeskikh usloviy v PP-2" (The block of the treatment of logical conditions in the PP-2).
V.S.Shtarkman. "Blok ekonomii rabočikh jačeek v PP-2" (The block of the economy of temporary locations in PP-2).
Problemy kibernetiki, Vypusk 1.

/5/ A.P.Ershov. "Programmirujuščaja programma dlja bystrodejstvujuščej elektronnoj sčetnoj mashyny" (The programming program for the BESM). Moskva, Izdatelstvo Akademii Nauk, 1958.

/6/ A.P.Ershov. "O programmirovanii arifmetičeskikh operatorov" (On programming of arithmetical operators), Doklady Akademii Nauk SSSR, v.118, №3 (1958), 427-430.
There is a translation: Communications of the ACM, v.1. №8 (1958).