K.Kuutti, H.Putkonen, I.Tervonen
University of Oulu
Institute of Data Processing Science
Linnanmaa, 90570 OULU

SAMPO PROGRAMMING LANGUAGE

K.Kuutti, H.Putkonen, I.Tervonen
University of Oulu
Institute of Data Processing Science

1. THE BACKGROUND OF SAMPO LANGUAGE

It is no easy task to teach novices to understand and program a computer. So we have been interested in, how the mastery of basic programming concepts could be easily achieved.

In winter 1982-83 we became acquainted with LOGO programming language, which seemed very different from other programming languages.

1.1 The LOGO programming language

The LOGO language is developed by prof. S. Papert at the Massachusett's Institute of Technology (MIT) for teaching purposes. The most remarkable features of the language are (/4/):

- it is a interpretative programming language (like BASICs in microcomputers): commands and programs are executed immediately, without any compiling, linking, etc.

- it supports effectively interactive graphics: the cursor ("turtle") can be moved on the screen by using commands, and the moving turtle draws lines. All the basic structures of the language can be used in picture building.

- the language is expandable: any series of commands can be named with own name and the new command behaves like the older ones.

LOGO was very easy to learn and use. It became apparent that if we could use the LOGO-approach, some of our problems in concept mastering could

be solved. However before using that approach we had to understand, why the LOGO-language was so easy to learn and use.

1.2 About the psychology of learning

We sought the solution from the psychology of learning. The cognitive approach offered the most usable schemas. Especially the thoughts of culturical-historical branch of cognitive psychology seemed to be relevant.

The following description of some aspects in the learning process is based on the references /1/,/3/,/5/. The description is very simplified and coarse.

> In the cognitive science there is the widely accepted view that an active action of man is always based on a conceptual model about the action's object.
>
> The accuracy of this model determines the success of the action. The model developes in reflexing the reality better and better. This development can be called learning.
>
> The process, in which one learns to master a new phenomenon (or, in which one developes a fixed and accurate conceptual model of that phenomenon) can be divided into three successive phases:
>
> > 1) When the phenomenon is fully new and there is not much knowledge about it's behaving and structure, one tries to link and understand it with already familiair phenomena. Thus the new conceptual model is always based on the older ones.
> >
> > 2) When the phenomenon is still new and one interacts with it one must first "plan" the action using the conceptual model. The success or the failure of the action strenghtens valid traits of the model and eliminates invalid ones. A fast feedback from the action supports the moving into the next phase.
> >
> > 3) When one has developed the conceptual model so accurate that the action is always successful (one masters the phenomenon and its premises), there is no need anymore for beforehand planning. So it "dies down" and the action is automatized.

1.3 The explanation of LOGO-properties

When the LOGO-language was examined using this schema, the following properties were found:

> 1) LOGO supports the initial phase of the conceptual model forming, the linking with already familiair phenomena: the "turtle-graphics" is fully understandable with everyday experience.
>
> 2) LOGO supports the conceptual model forming phase by the interactive practice: all the basic structures of the language can

be used in turtle-graphics, where the success or failure of the action can be seen immediately.

3) LOGO supports the automatization process of actions because the feed-back about actions is fast due to the interpreter.

4) When the action is ready to be automatized (the premises and consequencies of the action are mastered) LOGO supports the "fixing" of the action - a new command can be made from it.

These properties seemed to explain the effectiveness of the LOGO-approach and so we tried to find the way to utilize it. Because we wanted to experiment with real novices, we needed a programming language based on finnish. We also wanted to transport the good features of LOGO into the teaching of other ADP-related concepts.So we made the decision to design a new programming language, called SAMPO.

2. THE SAMPO PROGRAMMING LANGUAGE

2.1 The development goals

The general development goals were formulated during the the following way:

1. The useful features of the LOGO-language must be preserved.

2. The SAMPO language must include support for some new and difficult learnable thing, because the found principles must be tested.

3. The names of commands must be easily changed for possibility to examine the efficiency of learning with different command sets.

According to these principles we have reserved in SAMPO the essential properties of LOGO: interpretation, interactive graphics and extensibility. However the basic structure of languages is different: SAMPO is based on the use and manipulation of the stacks - LOGO concentrates on the manipulation of the lists.

One reason of this dissimilarity is the development goal number 2. The teaching of the stack manipulation was selected one goal of the language, because we didn't know any tool with what the stack manipulation could be teached efficiently.

There is also an other dissimilarity: LOGO uses the workspace approach when saving data and programs, but SAMPO is based on more conventional file approach. Also this choice has an instructional goal: "file" is one of the most difficult concepts for the novice, and it is impossible to teach it by workspace approach.

2.2 The main properties of SAMPO language

Like in the LOGO language also in the SAMPO language the most essential property from the learning/teaching point of view is the using of turtle-graphics in becoming acquainted with the microcomputer and the basic structures of the programming. In directing of the turtle we start with simple direction commands (words), which we can incrementally make more complicated using control structures.

Using the control structures in conventional way one tries to make programs more compact and still understandable. In the SAMPO language we use control structures and input/output words to render possible the producing of the conventional programs.

The saving of the developed words and so the expanding of the SAMPO language is made by the block/file-mechanism. One writes first the definition of the word on the screen (working area) and saves it as a block on the file (on the discette), from where one can get it anytime.

Because the SAMPO language has been developed using the stack structure, it is natural that these stack characteristics and stack basic words are also provided to the SAMPO user. The user notices the stack structure most clearly in the calculating words (+, -, x, /, ↑) or in the postfix notation of the SAMPO words (first the arguments and then the word).

To enable the symbol manipulation we have in the SAMPO language also list words. With help of the combining of the list and stack structures we want to create base for the continued extensibility and developability of the SAMPO language.

2.3 SAMPO language and the other interpreters

To examine the properties of the SAMPO language we have compared APL, LOGO, FORTH and SAMPO languages from the 11 features point of view. All these are interpretative languages and so in the measuring of the effectivity at the same level. The expertation and the stack/list basis have anyway effect on the effectivity of the language in some processing area. So APL language fits best to the numeric calculating, LOGO language to the list manipulating and turtle graphics and FORTH language to the stack processing.

Enabling both the list and stack processing machanism in the SAMPO language we hope to remove the shortages, which we have found in LOGO and FORTH languages; the utilizing of files in LOGO language and the missing graphics in FORTH language.

The following array is the short summary of the previous languages.

Property	APL	LOGO	FORTH	SAMPO
Notation	normal	prefix	postfix	postfix
Code	minimal	normal	tight	tight
Files	yes	no	yes	yes
Matrix calc.	yes	no	no/poss.	no
Lists	no	yes	no	yes
Extensibility	simple	yes	yes	yes
Recursion	yes	yes	no	yes
Assembler	no	no	yes	no
Stack	no	no	yes	yes
Graphics	no	yes	no/poss.	yes
Language conv.*	no	no	yes	no

* means the changing the commands of the language

2.4 Extensibility

Although we have already explained some features of the extensibility we take a more closely look at it in the following section. The more closely explanation of the term extensibility is necessary because the simplification of the expanding is really important improvement in comparing to the conventional languages.

The extensibility has been accomplished in programming languages using abstraction mechanism. By building the abstraction levels one can hide unnecessary information from the user and save only the necessary information for the user - what the modul (word) makes, what inputs it needs and what output it produces.

The accomplishment of the abstraction levels has in the conventional programming languages been bound to the compiler/interpreter mechanism. The "extensibility" of the compilers is based on the subroutine libraries and the linker. The using of the interpreter like Basic doesn't support extensibility because the working area limits the usability of the subroutines. The user (programmer) has to decide to where in the working area the subroutines are loaded.

Another possibility could be of course to expand the compiler or the interpreter by building a new level of the language on the top of that compiler/interpreter. In this building we have to use some lower level language.

The extensibility and the abstraction levels of the SAMPO language are based on the creating of the new words. The new word is like subroutine and always at the higher level than its "subwords". The subwords can be either primitive words (85 altogether) or words made of them. We can expand the language also like expanding the compiler: writing the new primitive to the dictionary, which is an editable file.

2.5 Portability

With the term portability we mean how much conversion effort is needed to make SAMPO language work in new hardware environment or in new user language environment.

The conversion to the new hardware is limited by the some hardware dependent features like graphics and file processing. The part of these is anyway relatively small and according to the Gilb's metric /2/

$$\text{Portability} = 1 - \frac{\text{resources for conversion}}{\text{resources for creating the whole system}}$$

the conversion of the SAMPO language to the new hardware environment has been established to the 75 %.

The conversion of the SAMPO language to the new user language environment (English, Swedish,...) has been estimated to 95 % using above metric, because the dictionary is on the editable file. To make this conversion we must translate the words, sort them according to the length of the word, translate the error messages and write this all on the file.

3. CONCLUSIONS

We have not yet any scientific results, but the feedback from the users has been promising: the SAMPO language is a useful tool in teaching either basic structures of programming or more exotic features like stack manipulation and list processing. Especially remarkable has been, that the language has interested people, who take else not interest in microcomputers at all.

It were even better, if there were no barrier between the programming language and the command language of the operating system. Then one could really start with playing and end with using a computer.

REFERENCES:

/1/ Galperin: Johdatus psykologiaan. Kansankulttuuri 1979
/2/ Gilb: Software metrics. Studentlitteratur 1976
/3/ Leontjev: Toiminta, tietoisuus, persoonallisuus. Kansankulttuuri 1977
/4/ Papert: Mindstorms. Basic Books 1980
/5/ Vygotski: Ajattelu ja kieli. Weilin & Göös 1982