SNOBOL

SNOBOL ("StriNg Oriented and symBOlic Language") is a series of programming languages developed between 1962 and 1967 at AT&T Bell Laboratories by David J. Farber, Ralph E. Griswold and Ivan P. Polonsky, culminating in SNOBOL4. It was one of a number of text-string-oriented languages developed during the 1950s and 1960s; others included COMIT and TRAC.

SNOBOL4 stands apart from most programming languages of its era by having patterns as a first-class data type (*i.e.* a data type whose values can be manipulated in all ways permitted to any other data type in the programming language) and by providing operators for pattern concatenation and alternation. SNOBOL4 patterns are a type of object and admit various manipulations, much like later object-oriented languages such as JavaScript whose patterns are known as regular expressions. In addition SNOBOL4 strings generated during execution can be treated as programs and either interpreted or compiled and executed (as in the eval function of other languages).

SNOBOL4 was quite widely taught in larger US universities in the late 1960s and early 1970s and was widely used in the 1970s and 1980s as a text manipulation language in the humanities.

In the 1980s and 1990s its use faded as newer languages such as AWK and Perl made string manipulation by means of regular expressions fashionable. SNOBOL4 patterns subsume BNF grammars, which are equivalent to context-free grammars and more powerful than regular expressions.[2] The "regular expressions" in current versions of AWK and Perl are in fact extensions of regular expressions in the traditional sense, but regular expressions, unlike SNOBOL4 patterns, are not recursive, which gives a distinct computational advantage to SNOBOL4 patterns.[3] (Recursive expressions did appear in Perl 5.10, though, released in December 2007.[4][5])

The later SL5 (1977)[6] and Icon (1978) languages were designed by Griswold to combine the backtracking of SNOBOL4 pattern matching with more standard ALGOL-like structuring.

SNOBOL

Paradigm	Imperative, procedural, unstructured
Designed by	David J. Farber, Ralph E. Griswold and Ivan P. Polonsky
Developer	David J. Farber, Ralph E. Griswold, Ivan P. Polonsky, and Bell Labs
First appeared	1962
Stable release	SNOBOL4 / 1967
Website	https://www.regressive.org/snobol4/
Major implementations	
SNOBOL, SPITBOL	
Influenced by	
COMIT	
Influenced	
AWK, SL5, Icon, bs, Lua[1]	

Development

SNOBOL1

The initial SNOBOL language was created as a tool to be used by its authors to work with the symbolic manipulation of polynomials. It was written in assembly language for the IBM 7090. It had a simple syntax, only one datatype, the string, no functions, and no declarations and very little error control. However,

despite its simplicity and its "personal" nature its use began to spread to other groups. As a result, the authors decided to extend it and tidy it up.

SNOBOL2

SNOBOL2 did exist but it was a short-lived intermediate development version without user-defined functions and was never released.

SNOBOL3

SNOBOL was rewritten to add functions, both standard and user-defined, and the result was released as SNOBOL3. SNOBOL3 became quite popular and was rewritten for other computers than the IBM 7090 by other programmers. As a result, several incompatible dialects arose.

SNOBOL4

As SNOBOL3 became more popular, the authors received more and more requests for extensions to the language. They also began to receive complaints about incompatibility and bugs in versions that they hadn't written. To address this and to take advantage of the new computers being introduced in the late 1960s, the decision was taken to develop SNOBOL4 with many extra datatypes and features but based on a virtual machine to allow improved portability across computers.[7] The SNOBOL4 language translator was still written in assembly language. However the macro features of the assembler were used to define the virtual machine instructions of the **S**NOBOL **I**mplementation **L**anguage, the SIL. This very much improved the portability of the language by making it relatively easy to port the virtual machine which hosted the translator by recreating its virtual instructions on any machine which included a macro assembler or indeed a high level language.[8]

The machine-independent language SIL arose as a generalization of string manipulation macros by Douglas McIlroy, which were used extensively in the initial SNOBOL implementation. In 1969, McIlroy influenced the language again by insisting on addition of the table type to SNOBOL4.[9][10]

SNOBOL4 features

SNOBOL is distinctive in format and programming style, which are radically different from contemporary procedural languages such as Fortran and ALGOL.

SNOBOL4 supports a number of built-in data types, such as integers and limited precision real numbers, strings, patterns, arrays, and tables (associative arrays), and also allows the programmer to define additional data types and new functions. SNOBOL4's programmer-defined data type facility was advanced at the time —it is similar to the records of the earlier COBOL and the later Pascal programming languages.

All SNOBOL command lines are of the form

label subject pattern **=** *object* **:** *transfer*

Each of the five elements is optional. In general, the *subject* is matched against the *pattern*. If the *object* is present, any matched portion is replaced by the *object* via rules for replacement. The *transfer* can be an absolute branch or a conditional branch dependent upon the success or failure of the subject evaluation, the pattern evaluation, the pattern match, the object evaluation or the final assignment. It can also be a transfer to code created and compiled by the program itself during a run.

A SNOBOL pattern can be very simple or extremely complex. A simple pattern is just a text string (e.g. "ABCD"), but a complex pattern may be a large structure describing, for example, the complete grammar of a computer language. It is possible to implement a language interpreter in SNOBOL almost directly from a Backus–Naur form expression of it, with few changes. Creating a macro assembler and an interpreter for a completely theoretical piece of hardware could take as little as a few hundred lines, with a new instruction being added with a single line.

Complex SNOBOL patterns can do things that would be impractical or impossible using the more primitive regular expressions used in most other pattern-matching languages. Some of this power derives from the so-called "SPITBOL extensions" (which have since been incorporated in basically all modern implementations of the original SNOBOL 4 language too), although it is possible to achieve the same power without them. Part of this power comes from the side effects that it is possible to produce during the pattern matching operation, including saving numerous intermediate/tentative matching results and the ability to invoke user-written functions during the pattern match which can perform nearly any desired processing, and then influence the ongoing direction the interrupted pattern match takes, or even to indeed change the pattern itself during the matching operation. Patterns can be saved like any other first-class data item, and can be concatenated, used within other patterns, and used to create very complex and sophisticated pattern expressions. It is possible to write, for example, a SNOBOL4 pattern which matches "a complete name and international postal mailing address", which is well beyond anything that is practical to even attempt using regular expressions.

SNOBOL4 pattern-matching uses a backtracking algorithm similar to that used in the logic programming language Prolog, which provides pattern-like constructs via DCGs. This algorithm makes it easier to use SNOBOL as a logic programming language than is the case for most languages.

SNOBOL stores variables, strings and data structures in a single garbage-collected heap.

Example programs

The "Hello, World!" program might be as follows...

```
          OUTPUT = "Hello, World!"
END
```

A simple program to ask for a user's name and then use it in an output sentence...

```
          OUTPUT = "What is your name?"
          Username = INPUT
          OUTPUT = "Thank you, " Username
END
```

To choose between three possible outputs...

```
          OUTPUT = "What is your name?"
          Username = INPUT
          Username "J"                                    :S(LOVE)
          Username "K"                                    :S(HATE)
MEH       OUTPUT = "Hi, " Username                        :(END)
LOVE      OUTPUT = "How nice to meet you, " Username      :(END)
HATE      OUTPUT = "Oh. It's you, " Username
END
```

To continue requesting input until no more is forthcoming...

```
          OUTPUT = "This program will ask you for personal names"
          OUTPUT = "until you press return without giving it one"
          NameCount = 0                                          :(GETINPUT)
AGAIN     NameCount = NameCount + 1
          OUTPUT = "Name " NameCount ": " PersonalName
GETINPUT  OUTPUT = "Please give me name " NameCount + 1
          PersonalName = INPUT
          PersonalName LEN(1)                                    :S(AGAIN)
          OUTPUT = "Finished. " NameCount " names requested."
END
```

Implementations

The classic implementation was on the PDP-10; it has been used to study compilers, formal grammars, and artificial intelligence, especially machine translation and machine comprehension of natural languages. The original implementation was on an IBM 7090 at Bell Labs, Holmdel, N.J. SNOBOL4 was specifically designed for portability; the first implementation was started on an IBM 7094 in 1966 but completed on an IBM 360 in 1967. It was rapidly ported to many other platforms.

It is normally implemented as an interpreter because of the difficulty in implementing some of its very high-level features, but there is a compiler, the SPITBOL compiler, which provides nearly all the facilities that the interpreter provides.

The classic implementation on the PDP-10 was quite slow, and in 1972 James Gimpel of Bell Labs, Holmdel, N.J. designed a native implementation of SNOBOL4 for the PDP-10 that he named SITBOL. He used the design as the basis of a graduate class in string processing that he taught that year at Stevens Institute of Technology (which is why it was named SITBOL). Students were given sections to implement (in PDP-10 assembler) and the entire semester was focused on implementing SITBOL. It was over 80% complete by the end of the semester and was subsequently completed by Professor Gimpel and several students over the summer. SITBOL was a full-featured, high-performance SNOBOL4 interpreter.

The Gnat Ada Compiler comes with a package (GNAT.Spitbol) that implements all of the Spitbol string manipulation semantics. This can be called from within an Ada program.

The file editor for the Michigan Terminal System (MTS) provided pattern matching based on SNOBOL4 patterns.[11]

Several implementations are currently available. Macro SNOBOL4 in C written by Phil Budne is a free, open source implementation, capable of running on almost any platform.[12] Catspaw, Inc provided a commercial implementation of the SNOBOL4 language for many different computer platforms, including DOS, Macintosh, Sun, RS/6000, and others, and these implementations are now available free from Catspaw. Minnesota SNOBOL4, by Viktors Berstis, the closest PC implementation to the original IBM mainframe version (even including Fortran-like FORMAT statement support) is also free.[13]

Although SNOBOL itself has no structured programming features, a SNOBOL preprocessor called Snostorm was designed and implemented during the 1970s by Fred G. Swartz for use under the Michigan Terminal System (MTS) at the University of Michigan.[14] Snostorm was used at the eight to fifteen sites that ran MTS. It was also available at University College London (UCL) between 1982 and 1984.

Snocone by Andrew Koenig adds block-structured constructs to the SNOBOL4 language. Snocone is a self-contained programming language, rather than a proper superset of SNOBOL4.[15]

The SPITBOL implementation also introduced a number of features which, while not using traditional structured programming keywords, nevertheless can be used to provide many of the equivalent capabilities normally thought of as "structured programming", most notably nested if/then/else type constructs. These features have since been added to most recent SNOBOL4 implementations. After many years as a commercial product, in April 2009 SPITBOL was released as free software under the GNU General Public License.

Naming

According to Dave Farber,[16] he, Griswold and Polonsky "finally arrived at the name Symbolic EXpression Interpreter SEXI."

> All went well until one day I was submitting a batch job to assemble the system and as normal on my JOB card — the first card in the deck, I, in BTL standards, punched my job and my name — SEXI Farber.
>
> One of the Comp Center girls looked at it and said, "That's what you think" in a humorous way.
>
> That made it clear that we needed another name!! We sat and talked and drank coffee and shot rubber bands and after much too much time someone said — most likely Ralph — "We don't have a Snowball's chance in hell of finding a name". All of us yelled at once, "WE GOT IT — SNOBOL" in the spirit of all the BOL languages. We then stretched our mind to find what it stood for.

Common backronyms of "SNOBOL" are 'String Oriented Symbolic Language'[17] or (as a quasi-initialism) 'StriNg Oriented symBOlic Language'.[18]

See also

- Icon (programming language)
- Snowball (programming language)
- Snostorm
- SPITBOL
- Unicon (programming language)

References

1. Ierusalimschy, Roberto; de Figueiredo, Luiz Henrique; Celes, Waldemar (2007), "The Evolution of Lua" (https://www.lua.org/doc/hopl.pdf) (PDF), *HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*: 26, doi:10.1145/1238844.1238846 (https://doi.org/10.1145%2F1238844.1238846)
2. Gimpel, J. F. (February 1973). "A theory of discrete patterns and their implementation in SNOBOL4" (https://doi.org/10.1145%2F361952.361960). *Communications of the ACM*. **16** (2): 91-100. doi:10.1145/361952.361960 (https://doi.org/10.1145%2F361952.361960). S2CID 17059429 (https://api.semanticscholar.org/CorpusID:17059429).
3. "Dr. Dobb's: Programs That Transform Their Own Source Code; or: the Snobol Foot Joke" (http://www.drdobbs.com/architecture-and-design/programs-that-transform-their-own-source/228701469). Dobbscodetalk.com. Retrieved 2011-12-04.

4. Contact details. "perlre" (http://perldoc.perl.org/5.10.0/perlre.html#Extended-Patterns). perldoc.perl.org. Retrieved 2011-12-04.
5. "Recursive Regex Tutorial" (http://www.rexegg.com/regex-recursion.html). Retrieved 2017-03-19.
6. Griswold, Ralph E.; Hanson, David R. (April 1977). "An Overview of SL5" (https://doi.org/10.1145%2F954654.954658). *ACM SIGPLAN Notices*. **12** (4): 40–50. doi:10.1145/954654.954658 (https://doi.org/10.1145%2F954654.954658). S2CID 38692673 (https://api.semanticscholar.org/CorpusID:38692673).
7. See Chapter 1 of *The Macro Implementation of SNOBOL4*
8. SNOBOL4 has been implemented using C to recreate the virtual machine instructions.
9. Griswold, Ralph (1978). "A history of the SNOBOL programming languages" (https://web.archive.org/web/20190302233559/http://pdfs.semanticscholar.org/a404/c09b14e2b03496604387f532fd33975179ec.pdf) (PDF). *ACM SIGPLAN Notices*. **13** (8): 275–308. doi:10.1145/960118.808393 (https://doi.org/10.1145%2F960118.808393). S2CID 5413577 (https://api.semanticscholar.org/CorpusID:5413577). Archived from the original (http://pdfs.semanticscholar.org/a404/c09b14e2b03496604387f532fd33975179ec.pdf) (PDF) on 2019-03-02.
10. Wexelblat, Richard L., ed. (2014) [1981]. *History of Programming Languages*. Academic Press. p. 784. ISBN 9781483266169.
11. Introduction to the MTS file editor (https://books.google.com/books?id=c9BWAAAAMAAJ), University of Michigan Computing Center, 1986.
12. "SNOBOL4.ORG -- SNOBOL4 Resources" (http://www.regressive.org/snobol4/).
13. "The MINNESOTA SNOBOL4 Programming Language" (http://www.berstis.com/snobol4.htm).
14. "SNOSTORM" (https://books.google.ca/books?id=WxVXAAAAMAAJ&pg=PA114&lpg=PA114&dq=snostorm+preprocessor&source=bl&ots=H5fdaCJj5n&sig=xvuW41x302-Je8xI1q8SbMSTM4Y&hl=en&sa=X&ei=E4f3U5jaOaLFigLOyYD4DQ&redir_esc=y#v=onepage&q=snostorm%20preprocessor&f=false), *MTS Volume 9: SNOBOL4 in MTS*, Computing Center, University of Michigan, June 1979, pages 99-120. Retrieved 1 September 2014.
15. "The Snocone Programming Language" (http://www.snobol4.com/report.htm), Andrew Koenig, USENIX (Portland, Oregon), June 1985. Retrieved 2 September 2014.
16. WORTH READING Wikipedia entry on SNOBOL — the TRUE story NOT Wikipedias (http://www.listbox.com/member/archive/247/2008/12/sort/time_rev/page/1/entry/0:180/20081226091150:1B4F85B0-D357-11DD-ABF7-AB09AB975BFC/) (Dave Farber, Interesting People mailing list, 26 December 2008)
17. *Computers and the humanities* **1**:158, 1967.
18. Belzer, Jack; Holzman, Albert G.; Kent, Allen, eds. (1979). "SNOBOL" (https://books.google.com/books?id=CEGXR7FeAWQC&dq=SNOBOL&pg=PA173). *Encyclopedia of Computer Science and Technology*. Vol. 13. CRC Press. p. 173. ISBN 0-8247-2263-9.

Further reading

- Emmer, Mark B. (1985). *SNOBOL4+: The SNOBOL4 Language for the Personal Computer User*. Prentice Hall. ISBN 0-13-815119-9.
- Gimpel, James F. (1976). *Algorithms in SNOBOL4*. Wiley. ISBN 0-471-30213-9. republished Salida, CO: Catspaw, 1986 (ISBN 0-939793-00-8).
- Griswold, Ralph E. (1972). *The Macro Implementation of SNOBOL4*. W.H. Freeman. ISBN 0-7167-0447-1.

- Griswold, Ralph E.; Poage, J.F.; Polonsky, I.P. (1968). *The SNOBOL4 Programming Language*. Prentice Hall. ISBN 0-13-815373-6.
- Griswold, Ralph E. (1975). *String and List Processing in SNOBOL4: Techniques and Applications*. Prentice Hall. ISBN 0-13-853010-6.
- Hockey, Susan M. (1985). *Snobol Programming for the Humanities*. Clarendon Press. ISBN 0-19-824676-5.

External links

- CSNOBOL4 (http://www.regressive.org/snobol4/csnobol4) is a free and open source BSD-licensed port of the original Bell Labs SNOBOL4 to systems with a C compiler, and includes SPITBOL and Blocks enhancements.
- Catspaw, Inc. offers implementations of and commercial support for SNOBOL4 (http://www.regressive.org/snobol4/)
- SNOBOL (https://curlie.org/Computers/Programming/Languages/Snobol) at Curlie
- Griswold, Ralph E. (25 July 1990). "Oral history interview with Ralph E. Griswold — discusses development of SNOBOL" (http://purl.umn.edu/107340). Minneapolis: Charles Babbage Institute, University of Minnesota.].
- "Charles Hall Collection on the SNOBOL Programming Language" (http://purl.umn.edu/40905). Minneapolis: Charles Babbage Institute, University of Minnesota.
- For a small brief taste of what SNOBOL4 is about try this online compiler (https://www.vintagebigblue.org/Compilerator/SNOBOL4/mvsSnobol4Compile.php)
- Try It Online (Snobol4/CSNOBOL) (https://tio.run/#snobol4) Online compiler