ALGOL 60

Jodok Huber

28.01.2013

ALGOL 60

- **ALGO**rithmic **L**anguage 19**60**
- developed by a committee of European and American computer scientists between 1958 - 1963
- gave rise to many other programming languages: BCPL, B, Pascal, Simula and C.

The Predecessor

Because of the lack of freedom in developing FORTRAN(property of IBM) the scientific computing community of the late 1950s developed a new language: ALGOL 58

The main goal was a syntax close to standard mathematical notations, so that the language could also be used for describing computing processes in written publications.

Its successor ALGOL 60 was then created to be used as a full-fledged, complete programming language for scientific computation.

Actual use

- became the standard Language for publishing Algorithms
- popular for teaching in University
- widely ignored by industry
- more success in Europe than in America

main Problems:

- absence of standard input/output facilities
- lack of interest in the language by large computer vendors

The Successor

- At first ALGOL W was intended to become the next ALGOL.
- But the committee decided on a more complex and advanced design, rather than a cleaned simplified ALGOL 60.
- The successor ALGOL 68 became a substantially different programming language.

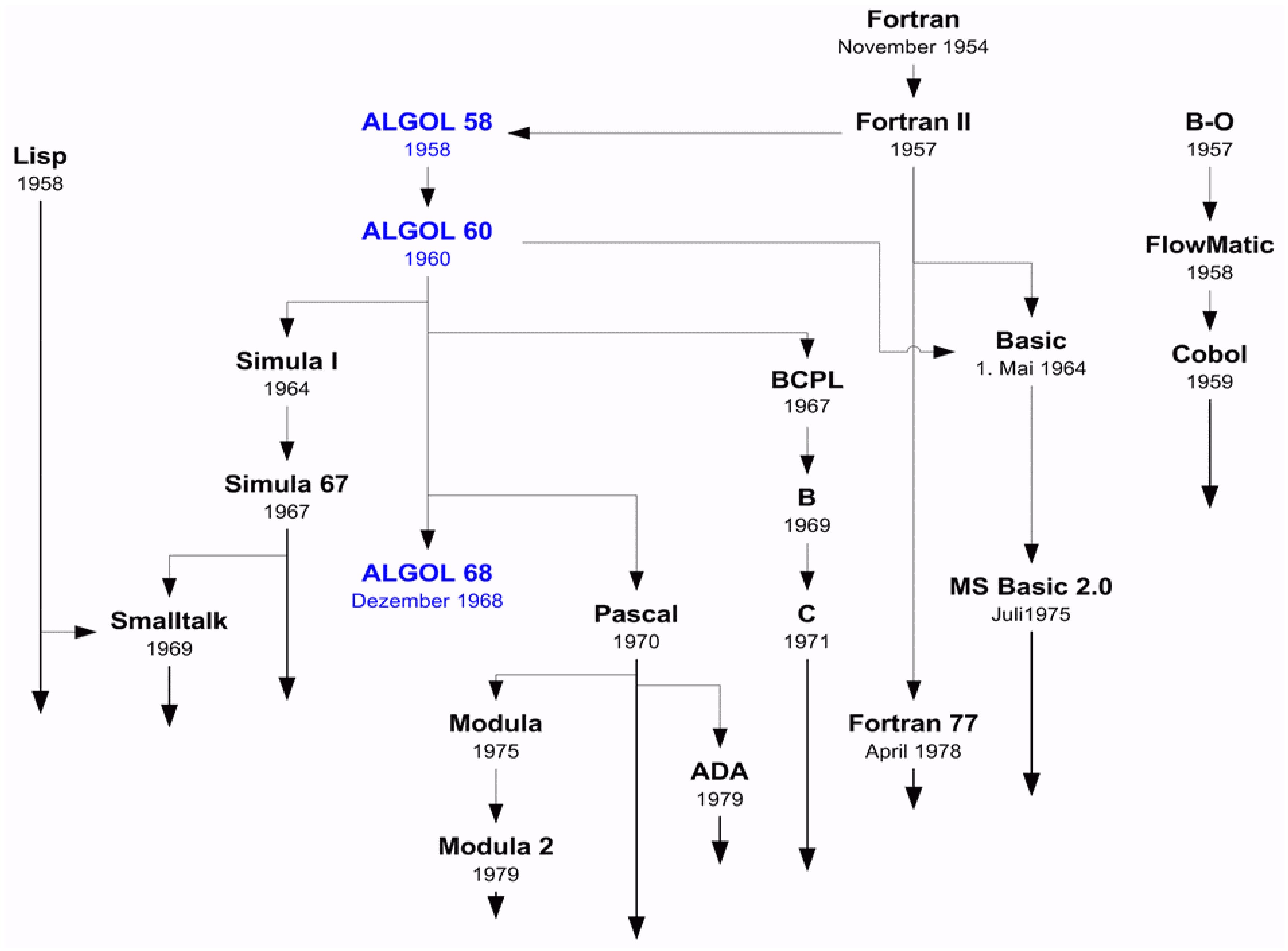
Innovations

ALGOL 60 introduced many new features into the history of Programming languages:

- With ALGOL 60 the Backus-Naur-Form became standard for describing programming languages.
- It already used runtime checking.

Innovations

- Keywords were reserved.
- ALGOL 60 uses a block-structure and was the first language implementing nested function definitions with lexical scope.
- The possibility to define procedures and functions and the ability to call them recursively was introduced.
- The If-then construct was extended to If-then-else.
- Flexible Loops were introduced.
- Dynamic declaration of arrays became possible.


Fortran
November 1954
Fortran II
1957
B-O
1957
ALGOL 58
1958
Lisp
1958
ALGOL 60
1960
FlowMatic
1958
Basic
1. Mai 1964
Cobol
1959
Simula I
1964
BCPL
1967
Simula 67
1967
B
1969
ALGOL 68
Dezember 1968
MS Basic 2.0
Juli1975
Smalltalk
1969
Pascal
1970
C
1971
Modula
1975
Fortran 77
April 1978
ADA
1979
Modula 2
1979

The program

ALGOL 60 is a block-structured language: A sequence of declarations followed by a sequence of statements and enclosed between begin and end.

```
begin
integer a, b, c;
a := 1;
b := a + 1;
c := a + b
end
```

Datatypes

- <u>real</u>
- <u>integer</u>
- <u>Boolean</u>
- <u>string</u>

Arrays

Arrays can be defined as follows:

```
integer array A[start:end];
```

Comments

```
comment This is a comment;
```

goto

- with goto it is possible to jump to a designational expression

```
M:x:=x + 1;
```

```
goto M;
```

Loops

```
for variable := list of elements do statement;
```

the elements in the list can be:

- arithmetic expressions
- a step b until c

  a(start value),b(step size) und c(limit) are arithmetic expressions
- d while e

  d is a arithmetic and e is a Boolean expression

Blocks

- The parts of the code in which Statements are allowed, blocks are allowed too.

```
begin
declarations
statements and/or other blocks
end
```

Blocks

- Variables are only visible in its own block and in the blocks beneath.
- When declaring a variable that already exists globally, the new variable will override in its block and all blocks beneath.

- Gotos can only jump to designational expressions in its own or in overlying blocks.

Procedures

<ins>`procedure`</ins> *name* `(`*list of parameters*`)` *;*
<ins>`value`</ins> *list of parameters that have to be called by value ;*
list of parameters including their types ;
Statements and/or a block

Call by value
The local variable in the procedure is initialized by the calling argument. Modification of this local variable would not affect anything external to the procedure.

Call by name
The actual argument effectively replaces every occurrence of the variable in the body of the procedure before the body is executed.

Procedures

Procedures can also return values if a type is writen in front of the procedure. In the body a return value has to be assigned to the procedure.

Example:

```
integer procedure sum(a,b);
integer a, b;
sum := a + b;
```

Sieve of Eratosthenes Example

```
begin
integer array candidates[0:1000];
integer i,j,k;
for i := 0 step 1 until 1000 do candidates[i] := 1;
candidates[0] := 0;
candidates[1] := 0;
i := 0;
for i := i while i<1000 do
        begin
        for i := i while i<1000 and candidates[i] = 0 do i := i+1;
        if i<1000 then
                begin
                j := 2;
                k := j*i;
                for k := k while k < 1000 do
                        begin
                        candidates[k] := 0;
                        j := j + 1;
                        k := j*i;
                        end;
                i := i+1;
                end;
        end;
end
```

Thank you for your attention.