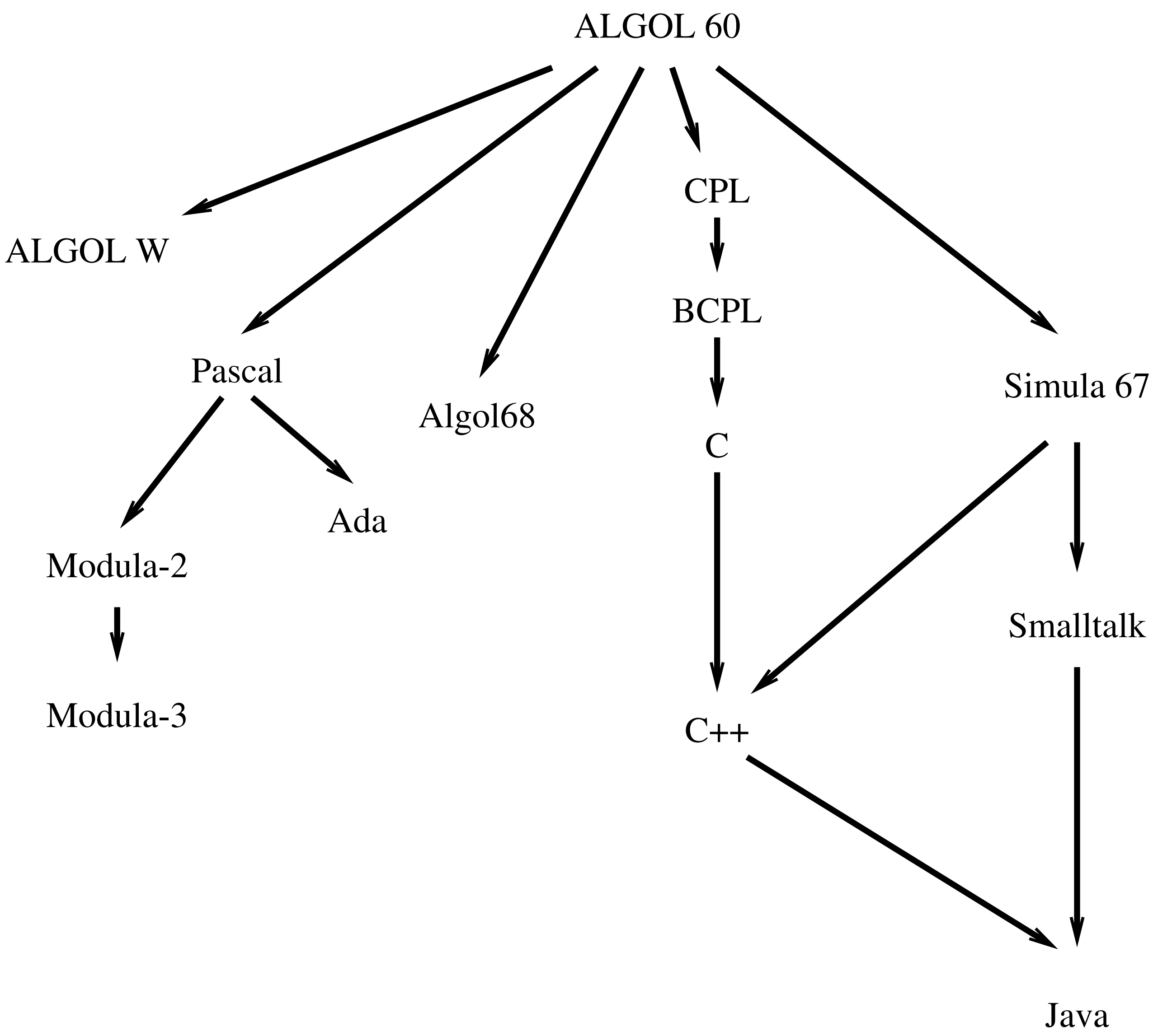
# ALGOL Influence



# ALGOL History

- Designed by a committee in 1958
- ALGOL 60 (1960), Revised (Rome 1962)
- Eclipsed by FORTRAN because
  - No I/O statements.
  - No separate compilation.
  - No library.
  - Not supported by IBM.
- But had a major effect on language design.

## ALGOL Innovations

- Block structure.
- Explicit type declarations for variables.
- Scope rules for local variables.
- Dynamic (not static) lifetimes for variables.
- Nested if-then-else expressions and statements.
- Call-by-value and call-by-name arguments.
- Recursive subroutines.
- Arrays with dynamic bounds.
- Use of BNF syntax description.

# ALGOL Example

```
comment  The following procedure
         will transpose a square matrix;

procedure transpose(A, m)
  value m;
  real array A; integer m;
begin integer i, k;
  for i := 1 step 1 until m do
    for k := i+1 step 1 until m do
      begin real w;
            w := A[i,k];
            a[i,k] := a[k,i];
            a[k,i] := w
      end
end transpose
```

# Call by name

```
real procedure sum(E, i, low, high)
  value low, high; real E;
  integer i, low, high;
begin sum := 0.0;
  for i := low step 1 until high do
    sum := sum + E;
end
...
integer j;
real array A[1:10];
real result;
for j := 1 step 1 until 10 do A[j] := j;

result :=  sum(A[j], j, 1, 10)
```

# Call by name

The call:

```
sum(A[j], j, 1, 10)
```

is equivalent to:

```
begin sum := 0.0;
  for j := 1 step 1 until 10 do
    sum := sum + A[j];
end
```

# Nested Procedures

```
integer procedure f(n);
  integer n;
begin
  integer a;

  integer procedure g(m);
    integer m;
  begin g := n + a + m;
        a := a+1
  end;

  if n=0 then a := 0;

  f := g(a)
end
```

## Trouble spots 1

11 different possible answers to:

```
begin
  integer a;

  integer procedure f(x, y);
      value x, y;
      integer x, y;
    a := f := x+1;

  integer procedure g(x);
      integer x;
    x := g := a+2;



  a := 2;
  outreal(1, a + f(a, g(a))/g(a))
end
```

# Trouble spots 2

- The types and mode of calling of procedure arguments could not be specified.
- The goto statement could cause a jump out to a label outside the current procedure.
- Labels were numeric (as in FORTRAN) and could be muddled with integers when passed in function arguments.
- Automatic type conversions were not fully specified, for example `x := x/y` was not properly defined when `x` and `y` were integers. (Is it allowed, and if so was `x` rounded or truncated.
- Own variables were a disaster.
- No precision specified for real numbers.

# ALGOL W

- Designed by Niklaus Wirth (about 1968).
- More pragmatic with repect to efficiency.
- One of the first languages to have structures in the form of records with named fields.
- Allowed pointers to records.
- It had four modes of calling procedure arguments.

# ALGOL W Arguments

```
PROCEDURE SUB1(INTEGER A;
               INTEGER VALUE B;
               INTEGER RESULT C;
               INTEGER VALUE RESULT D);
BEGIN
      B:= 7;
      A:=A+B;
      C:=B+D;
      D:=5
END
```

# ALGOL W Arguments

```
PROCEDURE SUB1(INTEGER A;
               INTEGER B;
               INTEGER C;
               INTEGER D);
BEGIN
      INTEGER BB;
      INTEGER CC;
      INTEGER DD;
      BB:=B;
      DD:=D;

      BB:= 7;
      A:=A+BB;
      CC:=BB+DD;
      DD:=5;

      C:=CC;
      D:=DD
END
```

# ALGOL W Records

```
RECORD CARD (INTEGER VAL;
             INTEGER SUIT);
RECORD HAND (REFERENCE(CARD)
             REFERENCE(HAND) NEXT);

REFERENCE(HAND) NORTH;

NORTH := HAND(CARD(1,1),
         HAND(CARD(12,1),NULL));

... VAL(CARD(NEXT(NORTH))) ...
```

# ALGOL W Problems

What does `X(Y)` mean?

It depends on the types of `X` and `Y`.

It could be:

- An array subscription.
- A procedure call.
- A field selection.
- A record constructor application.

Unfortunately, since formal procedures do not have their argument types specified, the types of variable are not always known.