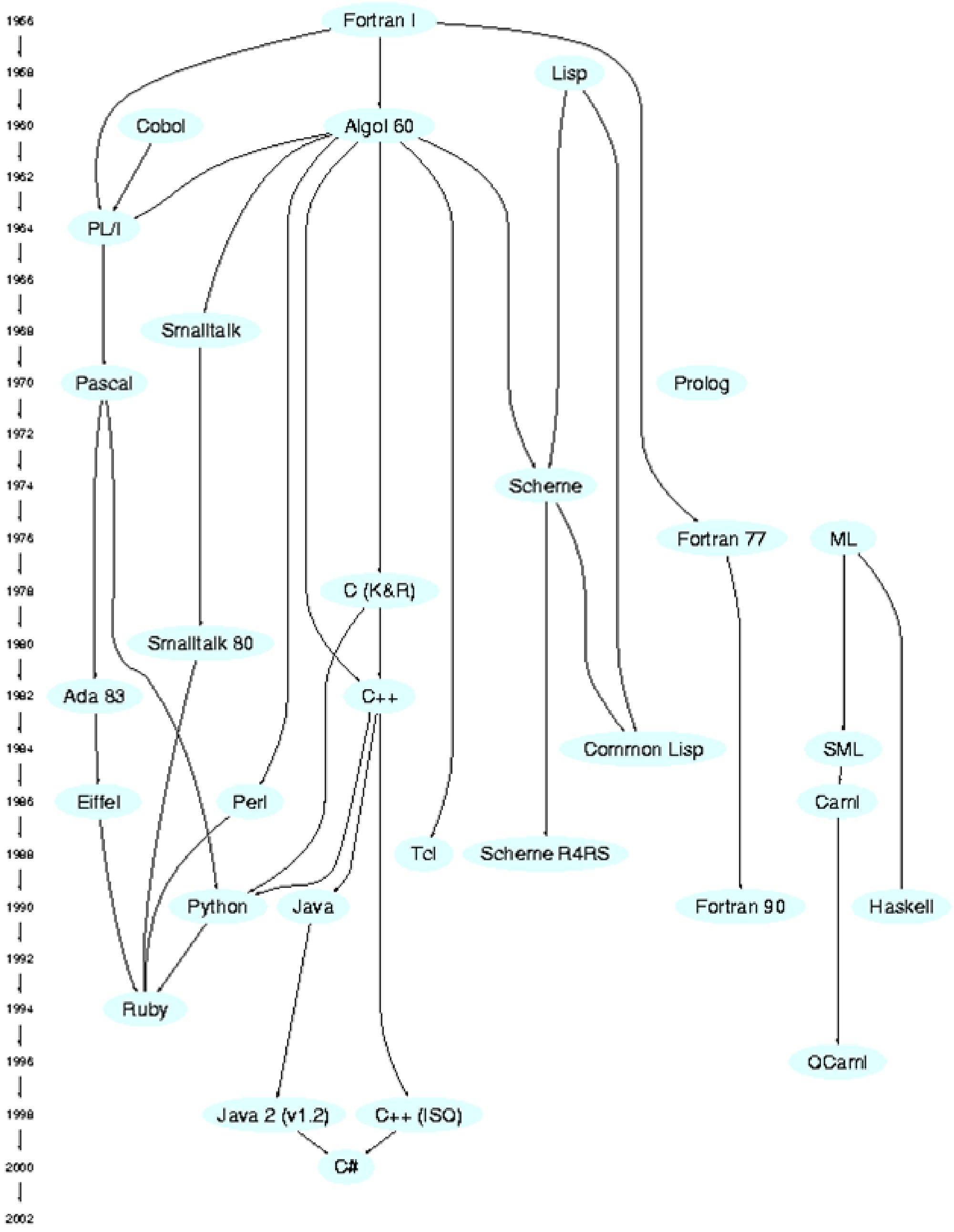
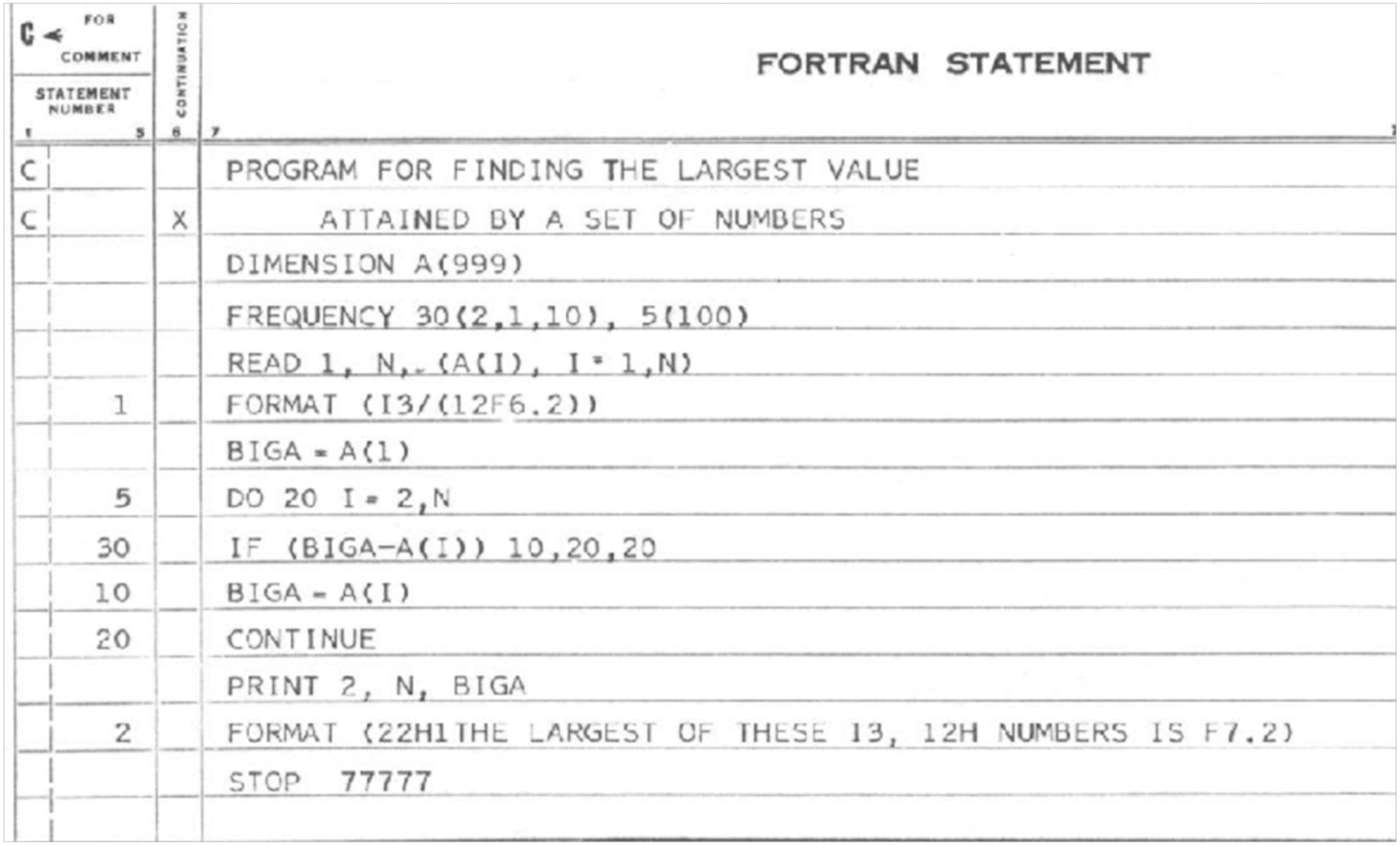
Lecture 17 — History of PL's

- **The major strands of high-level programming languages — static vs. dynamic typing, imperative vs. functional — first showed up quite early in the history of computers. But they have been manifested in a great variety of forms. We give a brief, and possibly biased, overview of the main developments.**
- **Topics for today:**
  - **Selective history of programming languages, by example**

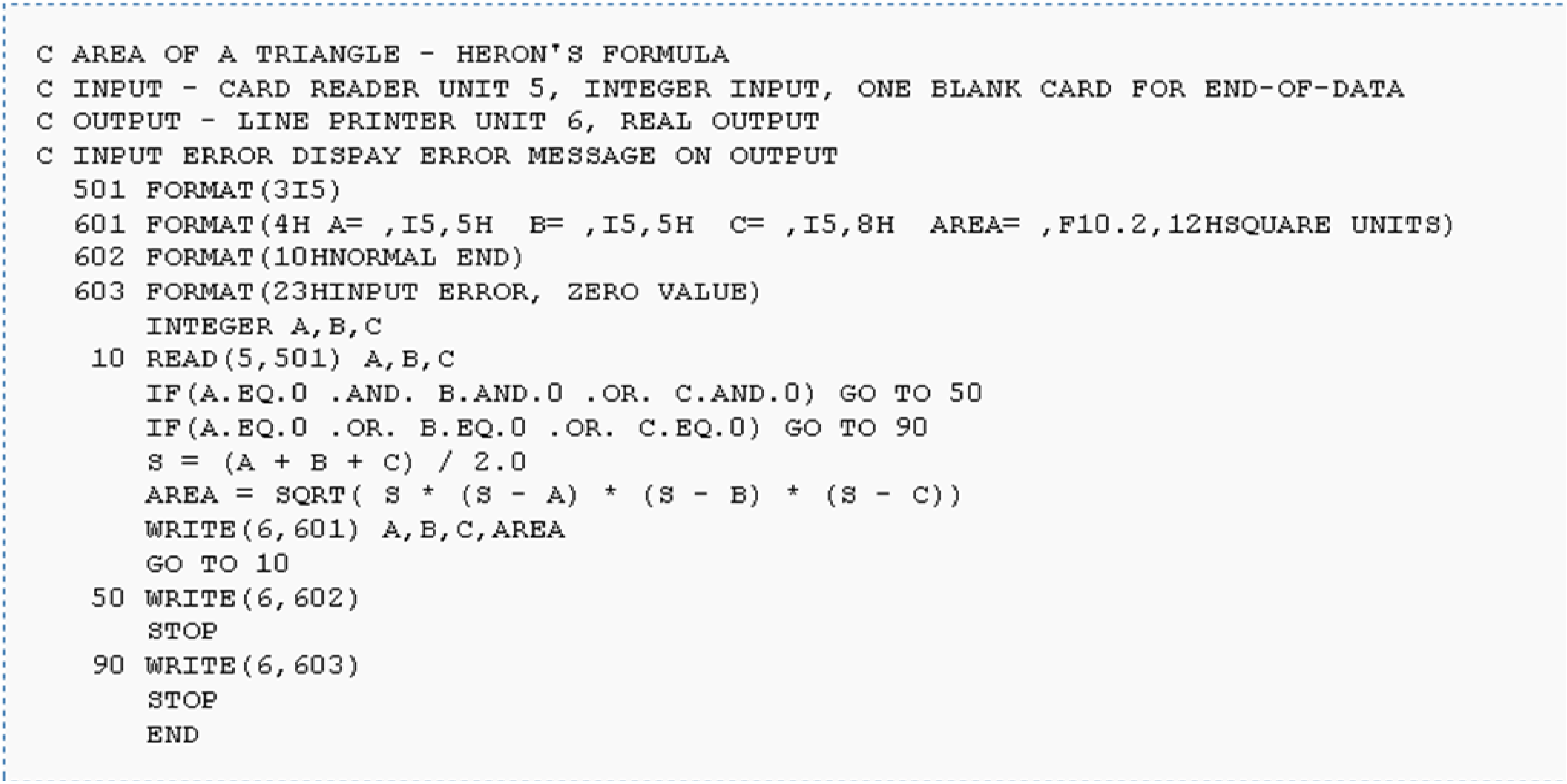

1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
Fortran I
Lisp
Cobol
Algol 60
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
C++
Common Lisp
SML
Eiffel
Perl
Caml
Tcl
Scheme R4RS
Python
Java
Fortran 90
Haskell
Ruby
OCaml
Java 2 (v1.2)
C++ (ISO)
C#

Fortran I


C ← FOR COMMENT
STATEMENT NUMBER
CONTINUATION
FORTRAN STATEMENT
1
5
6
7

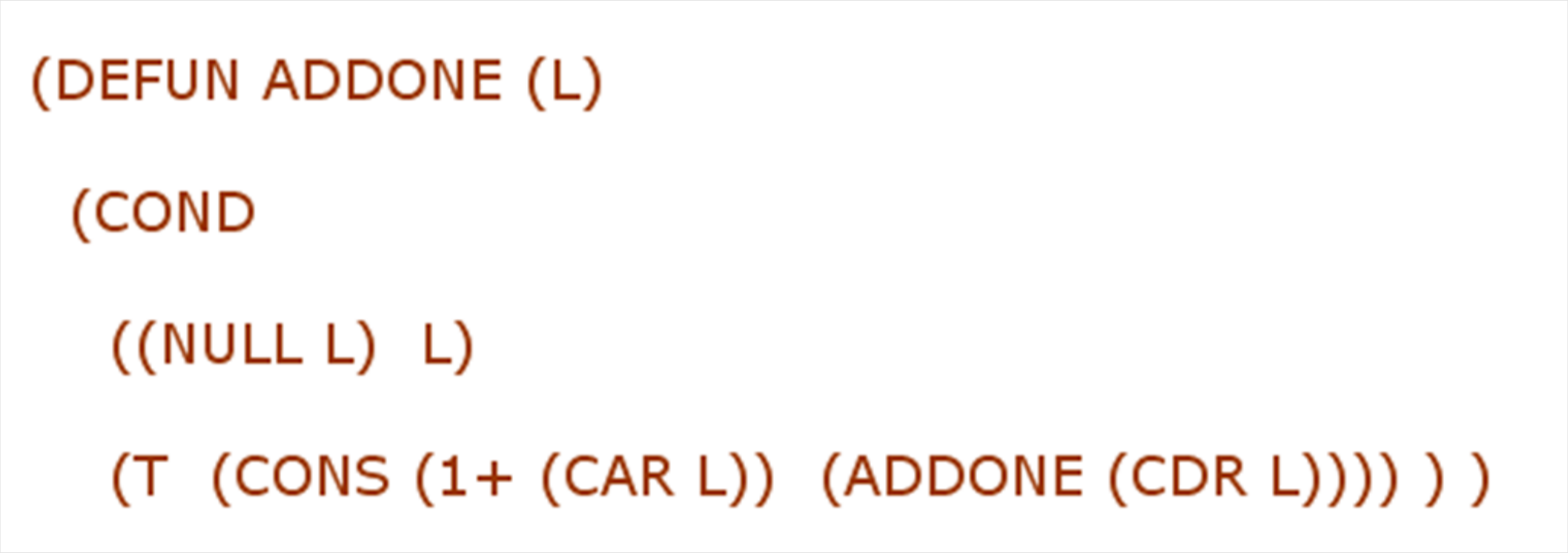
```
C     PROGRAM FOR FINDING THE LARGEST VALUE
C    X     ATTAINED BY A SET OF NUMBERS
      DIMENSION A(999)
      FREQUENCY 30(2,1,10), 5(100)
      READ 1, N, (A(I), I = 1,N)
    1 FORMAT (I3/(12F6.2))
      BIGA = A(1)
    5 DO 20 I = 2,N
   30 IF (BIGA-A(I)) 10,20,20
   10 BIGA = A(I)
   20 CONTINUE
      PRINT 2, N, BIGA
    2 FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)
      STOP  77777
```

Fortran IV



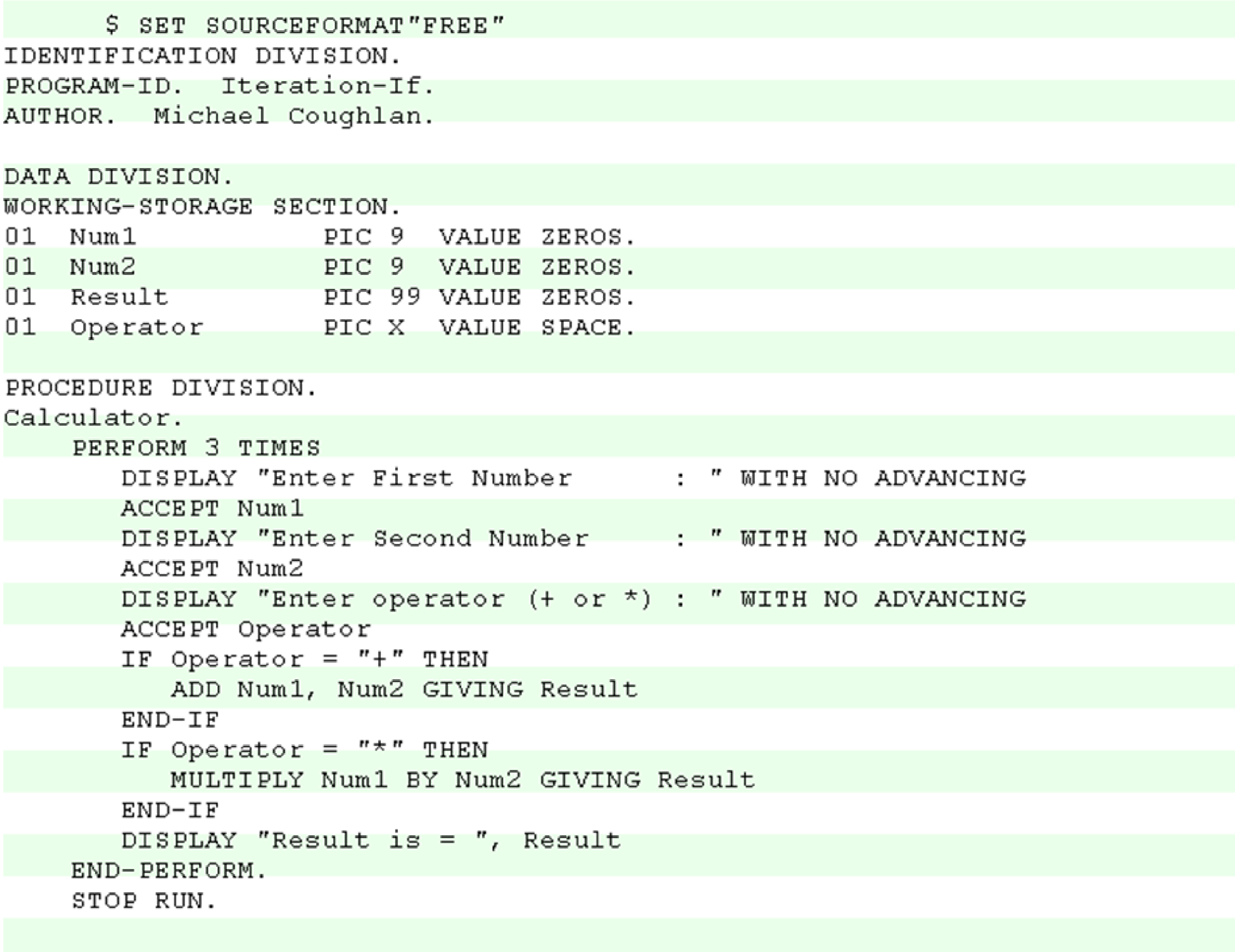
```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT, ONE BLANK CARD FOR END-OF-DATA
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPAY ERROR MESSAGE ON OUTPUT
  501 FORMAT(3I5)
  601 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,12HSQUARE UNITS)
  602 FORMAT(10HNORMAL END)
  603 FORMAT(23HINPUT ERROR, ZERO VALUE)
      INTEGER A,B,C
   10 READ(5,501) A,B,C
      IF(A.EQ.0 .AND. B.AND.0 .OR. C.AND.0) GO TO 50
      IF(A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) GO TO 90
      S = (A + B + C) / 2.0
      AREA = SQRT( S * (S - A) * (S - B) * (S - C))
      WRITE(6,601) A,B,C,AREA
      GO TO 10
   50 WRITE(6,602)
      STOP
   90 WRITE(6,603)
      STOP
      END
```

Lisp



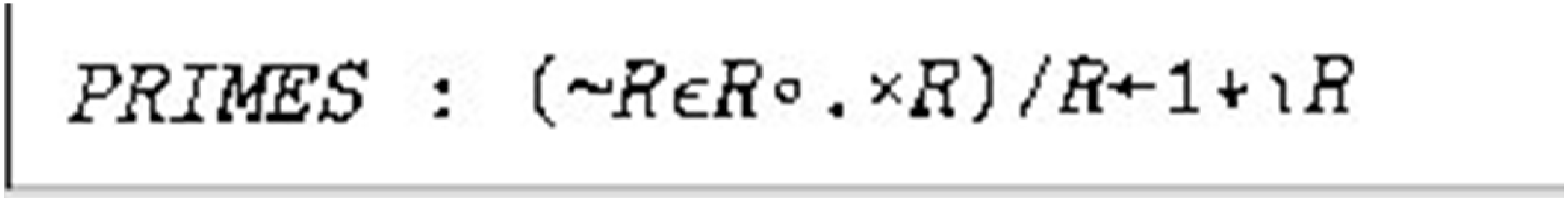
```
(DEFUN ADDONE (L)

  (COND

    ((NULL L)  L)

    (T  (CONS (1+ (CAR L))  (ADDONE (CDR L)))) ) )
```

COBOL



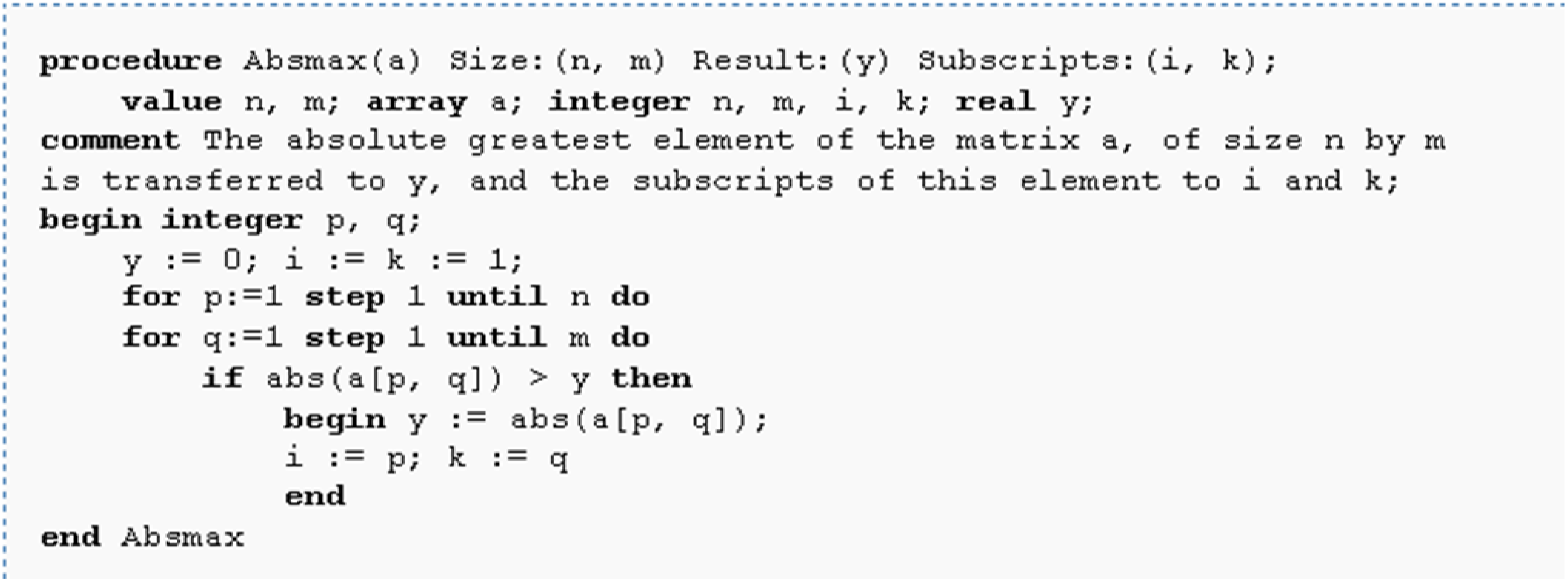
```
      $ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  Iteration-If.
AUTHOR.  Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  Num1               PIC 9  VALUE ZEROS.
01  Num2               PIC 9  VALUE ZEROS.
01  Result             PIC 99 VALUE ZEROS.
01  Operator           PIC X  VALUE SPACE.

PROCEDURE DIVISION.
Calculator.
    PERFORM 3 TIMES
       DISPLAY "Enter First Number      : " WITH NO ADVANCING
       ACCEPT Num1
       DISPLAY "Enter Second Number     : " WITH NO ADVANCING
       ACCEPT Num2
       DISPLAY "Enter operator (+ or *) : " WITH NO ADVANCING
       ACCEPT Operator
       IF Operator = "+" THEN
          ADD Num1, Num2 GIVING Result
       END-IF
       IF Operator = "*" THEN
          MULTIPLY Num1 BY Num2 GIVING Result
       END-IF
       DISPLAY "Result is = ", Result
    END-PERFORM.
    STOP RUN.
```

APL



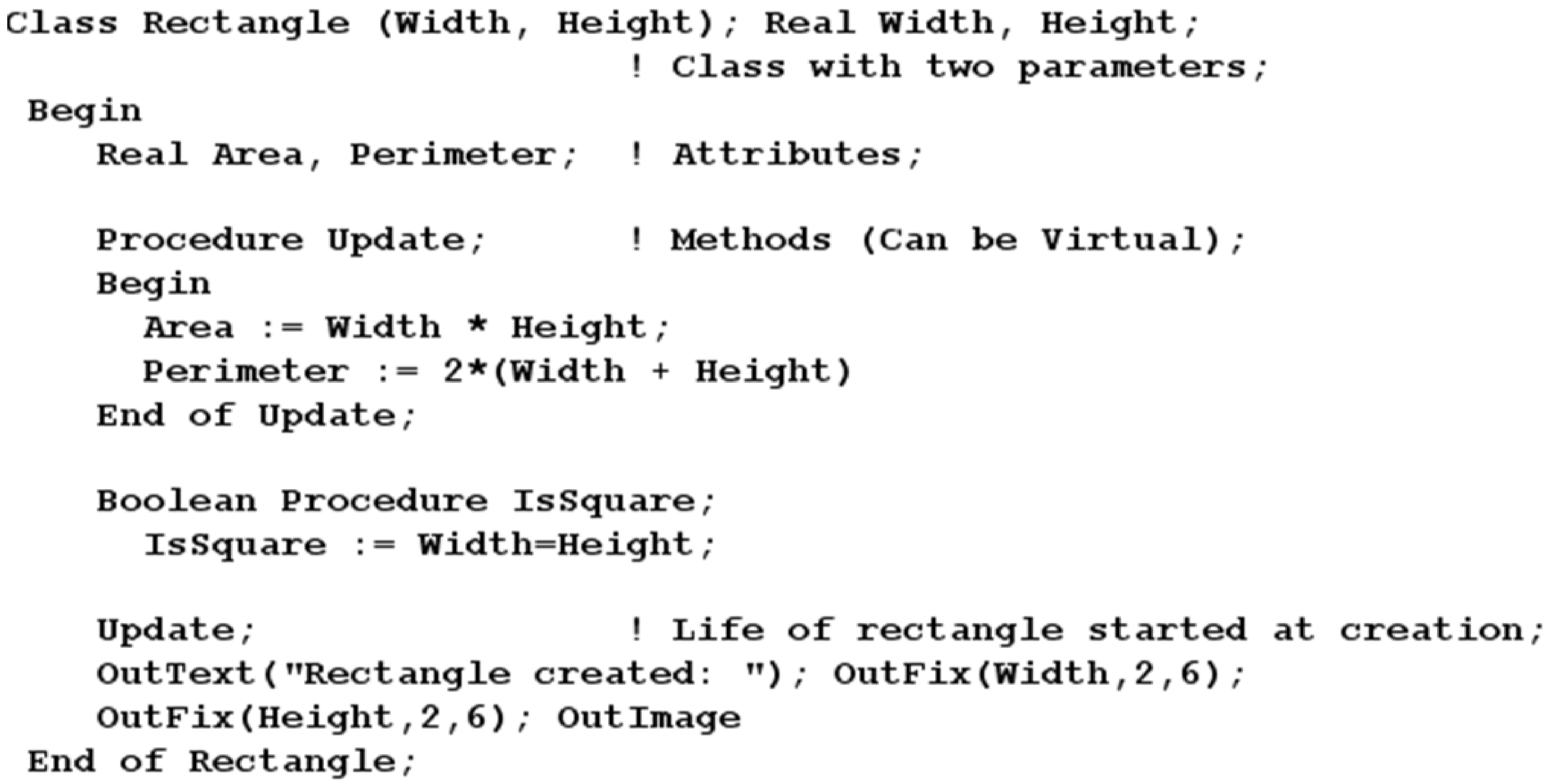
```
PRIMES : (~R∊R∘.×R)/R←1↓ιR
```

Algol



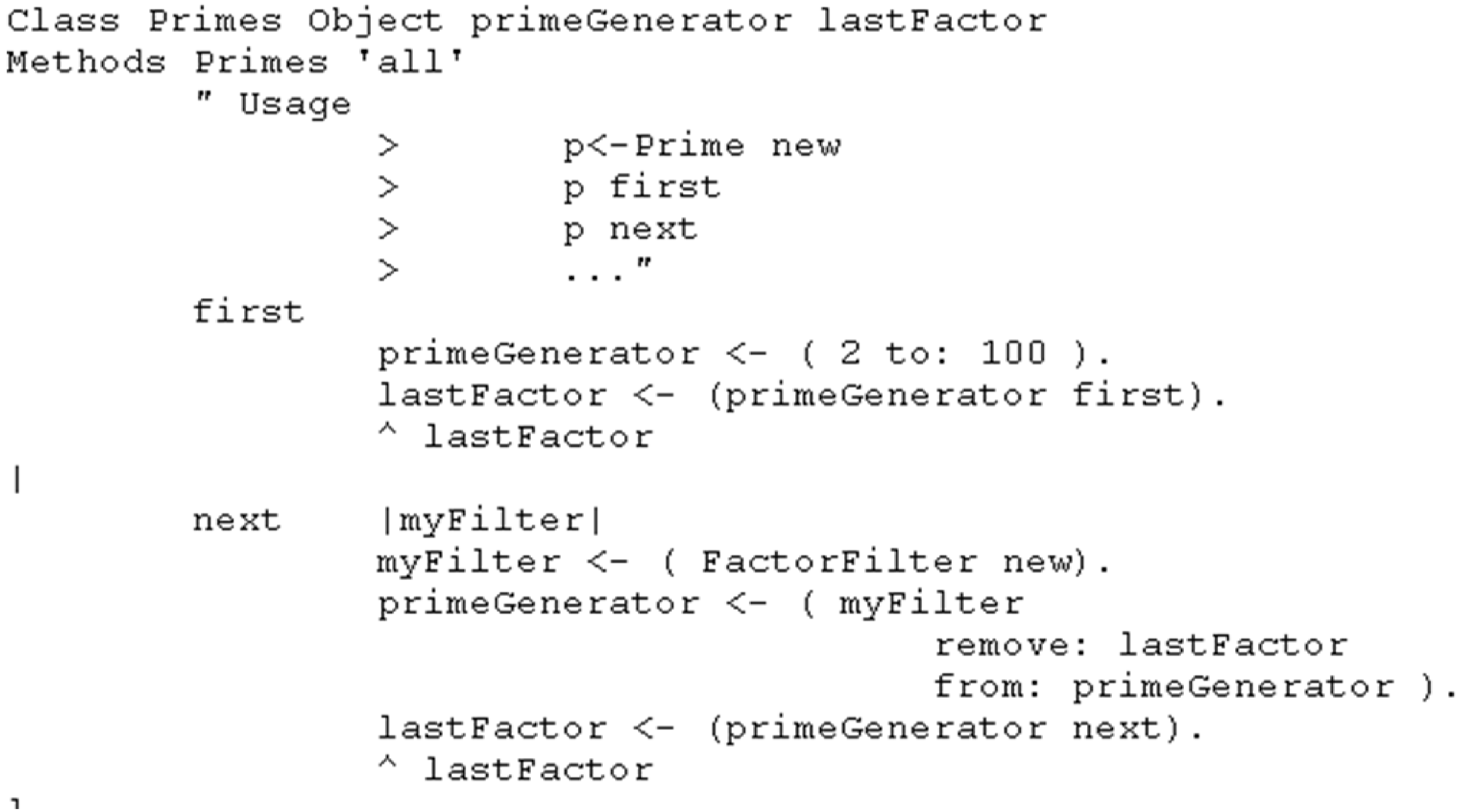
```
procedure Absmax(a) Size:(n, m) Result:(y) Subscripts:(i, k);
    value n, m; array a; integer n, m, i, k; real y;
comment The absolute greatest element of the matrix a, of size n by m
is transferred to y, and the subscripts of this element to i and k;
begin integer p, q;
    y := 0; i := k := 1;
    for p:=1 step 1 until n do
    for q:=1 step 1 until m do
        if abs(a[p, q]) > y then
            begin y := abs(a[p, q]);
            i := p; k := q
            end
end Absmax
```

Simula67



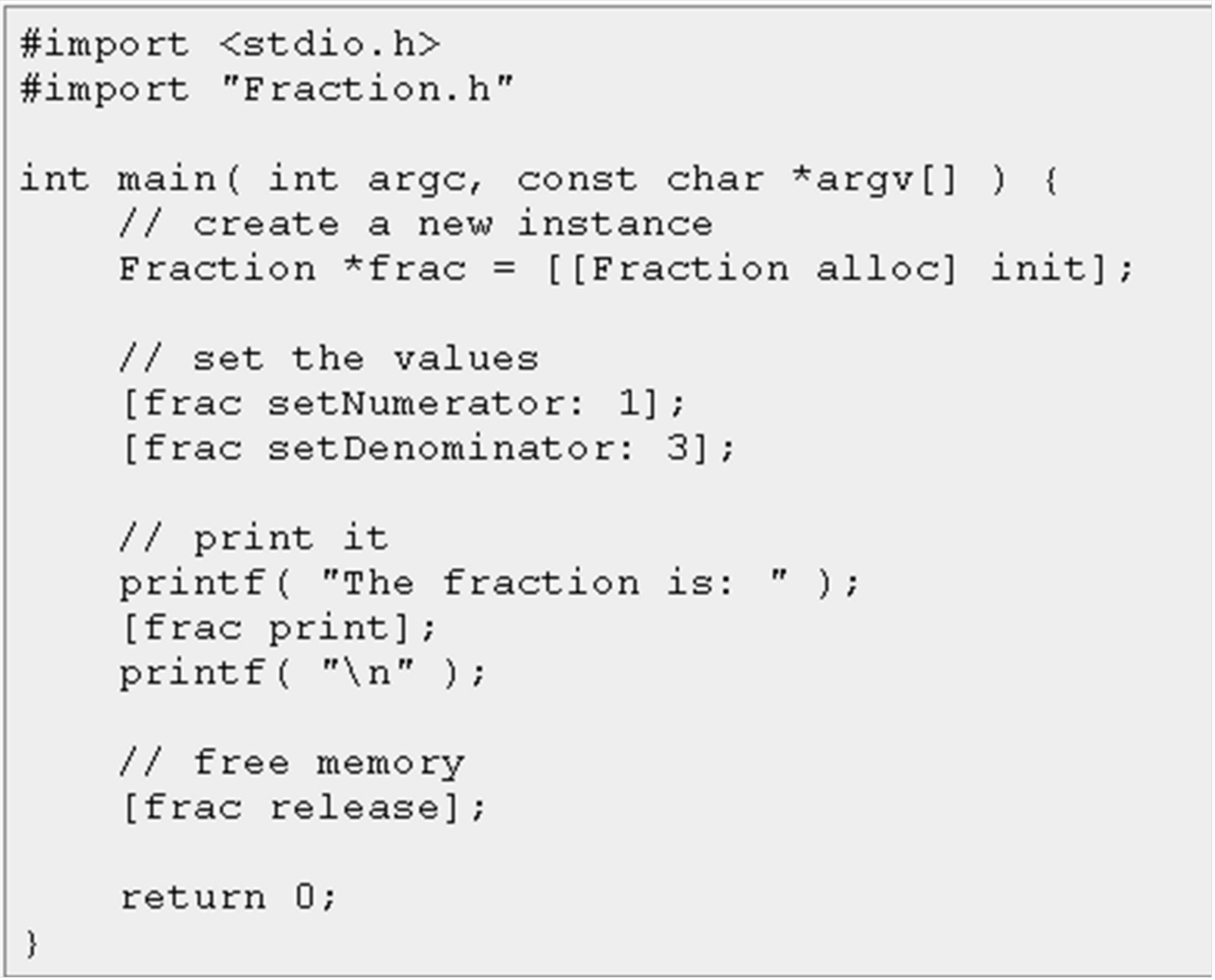
```
Class Rectangle (Width, Height); Real Width, Height;
                           ! Class with two parameters;
 Begin
    Real Area, Perimeter;  ! Attributes;

    Procedure Update;      ! Methods (Can be Virtual);
    Begin
      Area := Width * Height;
      Perimeter := 2*(Width + Height)
    End of Update;

    Boolean Procedure IsSquare;
      IsSquare := Width=Height;

    Update;                ! Life of rectangle started at creation;
    OutText("Rectangle created: "); OutFix(Width,2,6);
    OutFix(Height,2,6); OutImage
 End of Rectangle;
```

Smalltalk



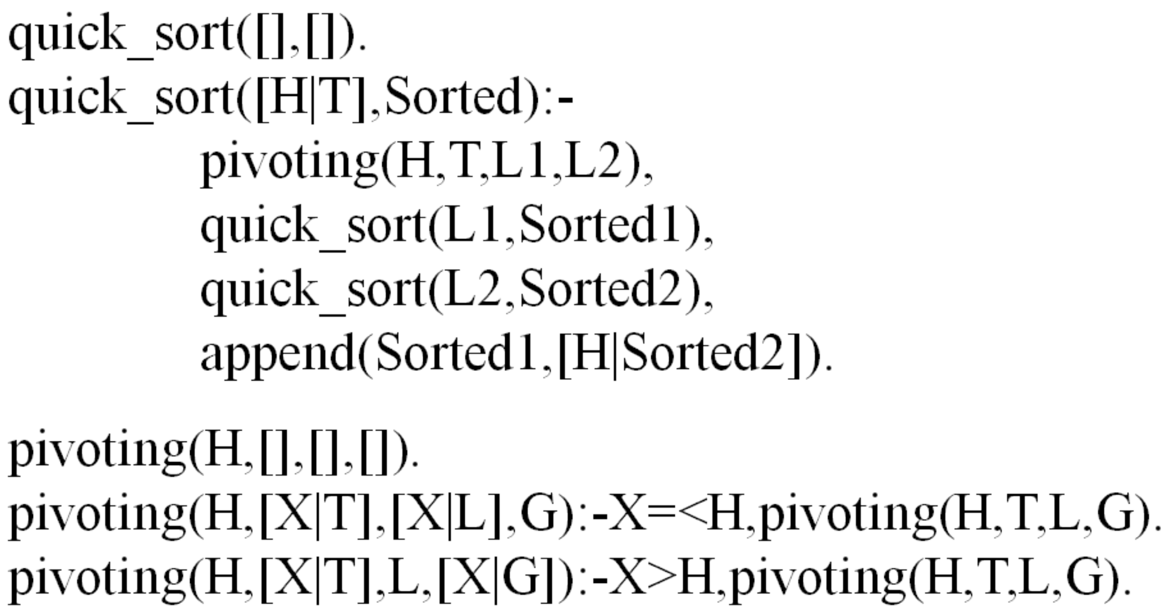
```
Class Primes Object primeGenerator lastFactor
Methods Primes 'all'
        " Usage
                >       p<-Prime new
                >       p first
                >       p next
                >       ..."
        first
                primeGenerator <- ( 2 to: 100 ).
                lastFactor <- (primeGenerator first).
                ^ lastFactor
|
        next    |myFilter|
                myFilter <- ( FactorFilter new).
                primeGenerator <- ( myFilter
                                        remove: lastFactor
                                        from: primeGenerator ).
                lastFactor <- (primeGenerator next).
                ^ lastFactor
]
```

Objective C



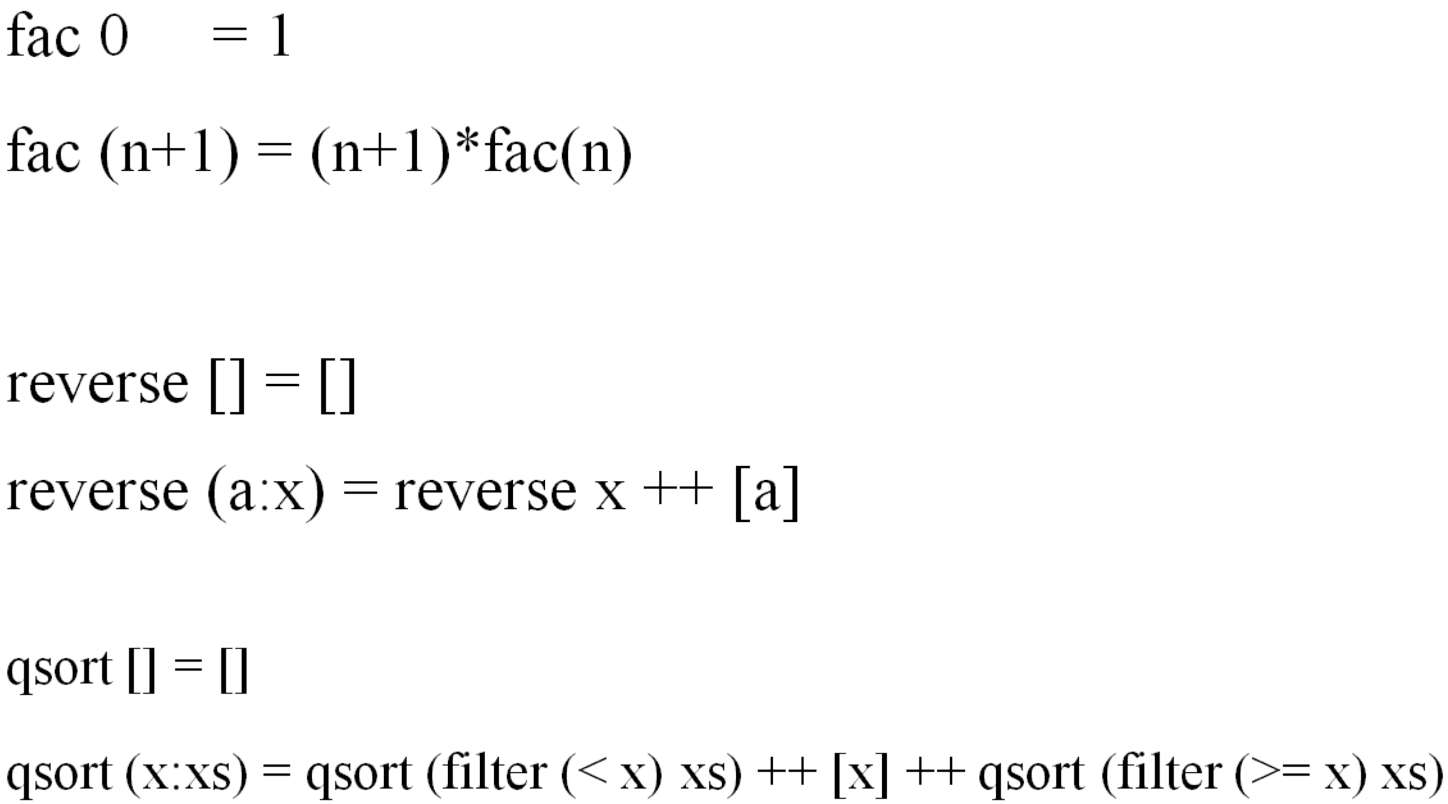
```
#import <stdio.h>
#import "Fraction.h"

int main( int argc, const char *argv[] ) {
    // create a new instance
    Fraction *frac = [[Fraction alloc] init];

    // set the values
    [frac setNumerator: 1];
    [frac setDenominator: 3];

    // print it
    printf( "The fraction is: " );
    [frac print];
    printf( "\n" );

    // free memory
    [frac release];

    return 0;
}
```

Prolog



```
quick_sort([],[]).
quick_sort([H|T],Sorted):-
            pivoting(H,T,L1,L2),
            quick_sort(L1,Sorted1),
            quick_sort(L2,Sorted2),
            append(Sorted1,[H|Sorted2]).

pivoting(H,[],[],[]).
pivoting(H,[X|T],[X|L],G):-X=<H,pivoting(H,T,L,G).
pivoting(H,[X|T],L,[X|G]):-X>H,pivoting(H,T,L,G).
```

Haskell



```
fac 0     = 1

fac (n+1) = (n+1)*fac(n)


reverse [] = []

reverse (a:x) = reverse x ++ [a]


qsort [] = []

qsort (x:xs) = qsort (filter (< x) xs) ++ [x] ++ qsort (filter (>= x) xs)
```

Scala

```
/** Print prime numbers less than 100, very inefficiently */
object primes extends Application {
 def isPrime(n: Int) = (2 until n) forall (n % _ != 0)
 for (i <- 1 to 100 if isPrime(i)) println(i)
}

/** Basic command line parsing. */
object Main {
 var verbose = false
 def main(args: Array[String]) {
   for (a <- args) a match {
    case "-h" | "-help" => println("Usage: scala Main [-help|-
verbose]")
    case "-v" | "-verbose" => verbose = true
    case x => println("Unknown option: '" + x + "'")
 }
 if (verbose) println("How are you today?")
}}
```