The History of Programming Languages

Jan 11 and 13, 2010
The Lambda Calculus

\[ \lambda m. \lambda n. \lambda z. \lambda s. m(nz)s \]
Plankalkül

P₁ max₃ (V₀[:8.0], V₁[:8.0], V₂[:8.0]) => R₀[:8.0]
max(V₀[:8.0], V₁[:8.0]) => Z₁[:8.0]
max(Z₁[:8.0], V₂[:8.0]) => R₀[:8.0]
END

P₂ max (V₀[:8.0], V₁[:8.0]) => R₀[:8.0]
V₀[:8.0] => Z₁[:8.0]
(Z₁[:8.0] < V₁[:8.0]) -> V₁[:8.0] => Z₁[:8.0]
Z₁[:8.0] => R₀[:8.0]
END

UNIVAC Short Code

\[ X_3 = \frac{X_1 + Y_1}{X_1 \times Y_1} \]

03 09 X1 07 Y1 02 04 X1 Y1
07 Y1 02 04 X1 Y1
0000 X3 03 09 X1

REAL SUM6, SUM7, SUM8, DIF6, DIF7, DIF8, SUMINF
*
OPEN(6, FILE='PRN')
SUM6 = .9*(1.-0.1**6)/0.9
SUM7 = .9*(1.-0.1**7)/0.9
SUM8 = .9*(1.-0.1**8)/0.9
SUMINF = 0.9/(1.0-0.1)
DIF6 = SUMINF - SUM6
DIF7 = SUMINF - SUM7
DIF8 = SUMINF - SUM8
WRITE(6,*) 'INFINITE SUM = ', SUMINF
WRITE(6,*) 'SUM6 = ', SUM6, ' INFINITE SUM - SUM6 = ', DIF6
WRITE(6,*) 'SUM7 = ', SUM7, ' INFINITE SUM - SUM7 = ', DIF7
WRITE(6,*) 'SUM8 = ', SUM8, ' INFINITE SUM - SUM8 = ', DIF8
STOP
END

http://www.engin.umd.umich.edu/CIS/course.des/cis400/fortran/Fortran_Example2.html
PROGRAM TPK
    REAL A(0:10)
    READ (5,*) A
    DO 10 I = 10, 0, -1
      Y = FUN(A(I))
      IF ( Y . LT. 400) THEN
        WRITE(6,9) I,Y
      ELSE
        WRITE (6,5) I
      ENDIF
    9 FORMAT(I10. F12.6)
    ELSE
        WRITE (6,5) I
    5 FORMAT(I10,' TOO LARGE')
    ENDIF
  10 CONTINUE
END
REAL FUNCTION FUN(T)
REAL T
FUN = SQRT(ABS(T)) + 5.0*T**3
END

http://www.nsc.liu.se/ boein/f77to90/a7.html
ID DIVISION.
PROGRAM-ID.  ACCEPT1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-FIRST-NUMBER PIC 9(3).
01 WS-SECOND-NUMBER PIC 9(3).
01 WS-TOTAL PIC ZZZ9.
*
PROCEDURE DIVISION.
0000-MAINLINE.
DISPLAY 'ENTER A NUMBER: '.
ACCEPT WS-FIRST-NUMBER.
*
DISPLAY 'ANOTHER NUMBER: '.
ACCEPT WS-SECOND-NUMBER.
*
COMPUTE WS-TOTAL = WS-FIRST-NUMBER + WS-SECOND-NUMBER.
DISPLAY 'THE TOTAL IS: ', WS-TOTAL.
STOP RUN.
ALGOL
begin

integer N;
Read Int(N);

begin
    real array Data[1:N];
    real sum, avg;
    integer i;
    sum := 0;

    for i := 1 step 1 until N do
        begin real val;
            Read Real(val);
            Data[i] := if val < 0 then -val else val
        end;

    for i := 1 step 1 until N do
        sum := sum + Data[i];
    avg := sum / N;
    Print Real(avg)
end
end

http://www.engin.umd.umich.edu/CIS/course.des/cis400/algol/average.html
(defun convert ()
  (format t "Enter Fahrenheit ")
  (LET (fahr)
    (SETQ fahr (read fahr))
    (APPEND '(celsius is) (*(- fahr 32)(/ 5 9)) ))
  )

http://www.engin.umd.umich.edu/CIS/course.des/cis400/lisp/convert.html
APL

http://www.rexswain.com/aplinfo.html

X[∆X+.≠’ ’;]

life←{↑1 ow.^3 4=+/,-1 0 1.0.e-1 0 1.0.Φ=ω}

&TRIM = 1
WORDPAT = BREAK(&LCASE &UCASE) SPAN(&LCASE &UCASE "','-") . WORD
COUNT = ARRAY('3:9',0)
READ LINE = INPUT :F(DONE)
NEXTW LINE WORDPAT = INPUT :F(READ)
COUNT<SIZE(WORD)> = COUNT<SIZE(WORD)>+ 1 : (NEXTW)
DONE OUTPUT = "WORD LENGTH NUMBER OF OCCURRENCES"
I = 2
PRINT I = I + 1
OUTPUT = LPAD(I,5) LPAD(COUNT<I>,20) :S(PRINT)

END

http://www.engin.umd.umich.edu/CIS/course.des/cis400/snobol/word.html
10 INPUT "ENTER TWO NUMBERS SEPARATED BY A COMMA:
20 LET S = N1 + N2
30 LET D = N1 - N2
40 LET P = N1 * N2
50 LET Q = N1 / N2
60 PRINT "THE SUM IS ", S
70 PRINT "THE DIFFERENCE IS ", D
80 PRINT "THE PRODUCT IS ", P
90 PRINT "THE QUOTIENT IS ", Q
100 END
FORWARD 100 ; draws a square with sides 100 units long
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90

http://en.wikipedia.org/wiki/Logo_(programming_language)
Forth

0 CONSTANT ${
: ->$$ CELLS + CELL+ @ COUNT ; ( addr ix -- 'strings )
: }$ CREATE ( addr*u u -- ) DUP , 0 ?DO , LOOP
DOES> ( ix -- c-addr u ) DUP @ 1- ROT - ->$$ ;
: }s$ CREATE ( addr*u u -- ) DUP 3 / , 0 ?DO , LOOP
DOES> ( ix -- c-addr u ) DUP @ 1- ROT - 3 * 3 CHOOSE + ->$$ ;
: }r$ CREATE ( addr*u u -- ) DUP , 0 ?DO , LOOP
DOES> ( -- c-addr u ) DUP @ CHOOSE ->$$ ;
S" phrases.forth" INCLUDED
' filler >BODY @ CONSTANT #phrases
' intros >BODY @ CONSTANT #intros
: Split-At-Char ( addr1 n1 char -- addr2+n2 n1-n2 addr2 n2 )
LOCALS| ch |
ch SKIP
2DUP ch SCAN TUCK 2>R - 2R> 2SWAP ;
: CR' CR 0 linecount ! ;
: SPACE' linecount @ IF SPACE 1 linecount +! THEN ;
: TYPE' DUP linecount +! TYPE ; ( char -- )
: -FITS? linecount @ + RMARGIN > ; ( #chars -- TRUE=fits-on-this-line )
: ANOTHER? DUP ; ( #chars -- TRUE=string-not-empty )

http://www.forth.com/starting-forth/sf12/wordgame.forth
prime
where
  prime = 2 fby (n whenever isprime(n));
  n = 3 fby n+2;
  isprime(n) = not(divs) asa divs or prime*prime > N
  where
    N is current n;
    divs = N mod prime eq 0;
  end;
end

http://en.wikipedia.org/wiki/Lucid_(programming_language)
gcd(A,B,GCD) :- A = B, GCD = A.
gcd(A,B,GCD) :- A > B, NA is A - B, gcd(NA,B,GCD).

fib(0,1).
fib(1,1).
fib(N,F) :- N > 1, N1 is N - 1, N2 is N - 2,
    fib(N1,F1), fib(N2,F2), F is F1 + F2.

ack(0,N,A) :- A is N + 1.
ack(M1,0,A) :- M > 0, M is M - 1, ack(M,1,A).
ack(M1,N1,A) :- M1 > 0, N1 > 0, M is M - 1, N is N - 1,
    ack(M1,N,A1), ack(M,A1,A).

http://cs.wwc.edu/KU/PR/Prolog.html
PL/I

BBBLE: PROCEDURE(ARRAY,N); /* BUBBLE SORT*/
DECLARE (I,J) FIXED BIN(15);
DECLARE S BIT(1);    /* SWITCH */
DECLARE Y FIXED BIN(15); /* TEMPO */
DO I = N-1 BY -1 TO 1;
    S = '1'B;
    DO J = 1 TO I;
        IF X(J)>X(J+1) THEN DO;
            S = '0'B;
            Y = X(J);
            X(J) = X(J+1);
            X(J+1) = Y;
        END;
    END;
    IF S THEN RETURN;
END;
RETURN;
END SRT;

http://www.engin.umd.umich.edu/CIS/course.des/cis400/pl1/pl1bubble.html
program ArithFunc;

const
  Sentinel = 0.0;

var
  X: Real;

begin
  writeln('After each line enter a real number or 0.0 to stop');
  writeln;
  writeln('X', 'Trunc(x)' :16, 'Round(X)' :10, 'Abs(X)' :10,
    'Sqr(X)' :10, 'Sqrt(Abs(X))' :15);
  readln(X);
  while X <> Sentinel do
  begin
    writeln(Trunc(X) :17, Round(X) :10, Abs(X) :10:2,
      Sqr(X) :10:2, Sqrt(Abs(X)) :10:2);
    readln(X);
  end
end.

http://www.engin.umd.umich.edu/CIS/course.des/cis400/pascal/arithmetic.html
package body ArrayCalc is
  function sum return integer is
    temp: integer;
    -- Body of function sum
    begin
      temp := 0;
      for i in 1..v.sz loop
        temp := temp + v.val(i);
      end loop;
      v.sz:=0;
      return temp;
    end sum;
  procedure setval(arg:in integer) is
    begin
      v.sz:= v.sz+1;
      v.val(v.sz):=arg;
    end setval; end;

http://www.engin.umich.umd.edu/CIS/course.des/cis400/ada/array_summation.html
// Routine to compute a checksum of a
// named file, simplified from a compiler example.
GET "libhdr"

LET start() = VALOF
$( LET args = VEC 50
  LET instream = 0
  LET outstream = 0
  LET sum = 314159

  IF rdargs("FROM/A,TO/K", args, 50) = 0 DO
    $( writes("Bad arguments for CHECKSUM*n")
    RESULTIS 20
  )$

  instream := findinput(args!0)
  IF instream = 0 DO $( writef("can’t open %s*n", args!0)
    RESULTIS 20
  )$

  selectinput(instream)

http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl?_alpha=b
printn(n, b) {
    extrn putchar;
    auto a;

    if(a=n/b) /* assignment, not test for equality */
    printn(a, b); /* recursive */
    putchar(n%b + '0');
}

http://en.wikipedia.org/wiki/B_(programming_language)
BEGIN INTEGER X, N, SUM, MAX;

IF LASTITEM THEN OUTTEXT("NULL LIST") ELSE
BEGIN SUM:=MAX:=ININT;
N:=1;

WHILE LASTITEM DC
BEGIN X:=ININT;
N:=N+1;
IF X > MAX THEN MAX:=X;
SUM:=SUM+X;
END;
OUTTEXT("LIST LENGTH = "); OUTINT (N, 6);
OUTTEXT(",
HIGHEST = "); OUTINT (MAX, 6);
OUTTEXT(",
AVERAGE = "); OUTFIX (SUM/N, 2,, 8);
END;
OUTIMAGE;
END

http://www.engin.umd.umich.edu/CIS/course.des/cis400/simula/f1.html
Smalltalk

<table>
<thead>
<tr>
<th>scfk</th>
</tr>
</thead>
</table>
s := Prompter prompt: 'Enter line'
default: ' '.
1 to: 26 do [:i | f at: i put: 0].
1 to: s size do: [:i |
c := (s at: i) asLowerCase.
c isLetter
ifTrue: [
k := c asciiValue - &a asciiValue + 1
]
].
^f

http://www.engin.umd.umich.edu/CIS/course.des/cis400/smalltalk/freq.html
Eternal Flame by Bob Kanefsky
(Parody on God lives on Terra by Julia Ecklar)

I was taught assembler
in my second year of school.
It’s kinda like construction work – with a toothpick for a tool.
So when I made my senior year,
I threw my code away,
And learned the way to program
that I still prefer today.
Now, some folks on the Internet put their faith in C++. They swear that it’s so powerful, it’s what God used for us. And maybe it lets mortals dredge their objects from the C. But I think that explains why only God can make a tree.
Humor

For God wrote in Lisp code
When he filled the leaves with green.
The fractal flowers and recursive roots:
The most lovely hack I’ve seen.
And when I ponder snowflakes,
ever finding two the same,
I know God likes a language
with its own four-letter name.
Now, I’ve used a SUN under Unix, so I’ve seen what C can hold.
I’ve surfed for Perls, found what Fortran’s for,
Got that Java stuff down cold.
Though the chance that I’d write COBOL code
is a SNOBOL’s chance in Hell.
And I basically hate hieroglyphs, so I won’t use APL.
Humor

Now, God must know all these languages, and a few I haven’t named.

But the Lord made sure, when each sparrow falls, that its flesh will be reclaimed.

And the Lord could not count grains of sand with a 32-bit word.

Who knows where we would go to if Lisp weren’t what he preferred?
And God wrote in Lisp code
Every creature great and small.
Don’t search the disk drive for man.c,
When the listing’s on the wall.
And when I watch the lightning burn
Unbelievers to a crisp,
I know God had six days to work,
So he wrote it all in Lisp.

Yes, God had a deadline.
So he wrote it all in Lisp.