Giorgio Spugnesi

# Literate Programming

Seminario di Cultura Digitale
Pisa, 25 luglio 2012

Knuth, Donald. *Literate Programming* in "The Computer Journal" (1984) 27 (2) pp. 97-111.

- Programmazione come *work of literature*

- Il sistema WEB

**Donald Knuth** (1938), Professor Emeritus in the Art of Computer Programming – Stanford University

The Art of Computer Programming (7 voll. iniziato nel 1962, il vol. 5 è previsto per il 2020)

TEX (1978) e The TeXbook (1984)

METAFONT (1979) e The METAFONTbook (1986)

# Programmazione Letteraria

- La programmazione è una forma d'arte e, in particolare, una forma di letteratura
- La programmazione è comunicazione tra esseri umani in merito alla risoluzione di problemi
- Il programma deve convincere chi lo legge della propria correttezza
- I programmi letterari sono migliori ed hanno meno bug

# Il sistema WEB

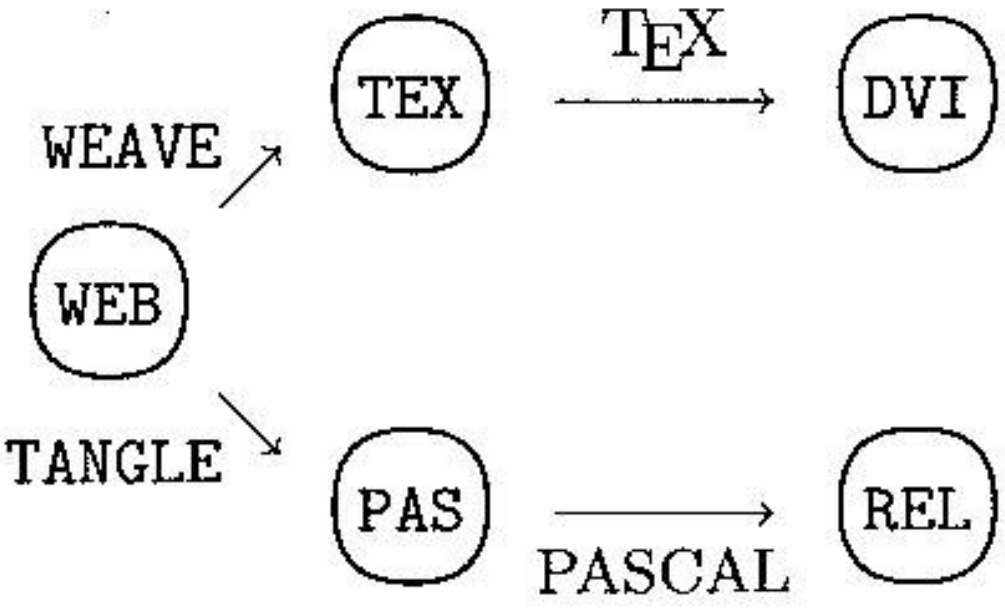


**Figure 1.** Dual usage of a WEB file.

# Il codice WEB

```
@ This program has no input, because we want
to keep it rather simple.  The result of the
program will be to produce a list of the
first thousand prime numbers, and this list
will appear on the |output| file.

Since there is no input, we declare the value
|m=1000| as a compile-time constant. The
program itself is capable of generating the
first |m| prime numbers for any positive |m|,
as long as the computer's finite limitations
are not exceeded.

\[The program text below specifies the
``expanded meaning'' of `\X2:Program to print
$\ldots$ numbers\X'; notice that it involves
the top-level descriptions of three other
sections. When those top-level descriptions
are replaced by their expanded meanings, a
syntactically correct \PASCAL\ program will
be obtained.\]

@<Program to print...@>=
program print_primes(output);
const @!m=1000;
@<Other constants of the program@>@;
var @<Variables of the program@>@;
begin @<Print the first |m| prime numbers@>;
end.
```

**Figure 2b.** The WEB code that generated §2.

# Il risultato del WEAVE

**2.** This program has no input, because we want to keep it rather simple. The result of the program will be to produce a list of the first thousand prime numbers, and this list will appear on the *output* file.

Since there is no input, we declare the value $m = 1000$ as a compile-time constant. The program itself is capable of generating the first $m$ prime numbers for any positive $m$, as long as the computer's finite limitations are not exceeded.

⟦The program text below specifies the "expanded meaning" of '⟨ Program to print ... numbers 2 ⟩'; notice that it involves the top-level descriptions of three other sections. When those top-level descriptions are replaced by their expanded meanings, a syntactically correct PASCAL program will be obtained.⟧

```
⟨ Program to print the first thousand prime
      numbers 2 ⟩ ≡
program print_primes(output);
  const m = 1000;
    ⟨ Other constants of the program 5 ⟩
  var ⟨ Variables of the program 4 ⟩
    begin ⟨ Print the first m prime numbers 3 ⟩;
    end.
```

This code is used in section 1.

# Il risultato del TANGLE

```
{1:}{2:}PROGRAM PRINTPRIMES(OUTPUT);
CONST M=1000;{5:}RR=50;CC=4;WW=10;{:5}{19:}
ORDMAX=30;{:19}VAR{4:}
P:ARRAY[1..M]OF INTEGER;{:4}{7:}
PAGENUMBER:INTEGER;PAGEOFFSET:INTEGER;
ROWOFFSET:INTEGER;C:0..CC;{:7}{12:}J:INTEGER;
K:0..M;{:12}{15:}JPRIME:BOOLEAN;{:15}{17:}
ORD:2..ORDMAX;SQUARE:INTEGER;{:17}{23:}
N:2..ORDMAX;{:23}{24:}
MULT:ARRAY[2..ORDMAX]OF INTEGER;{:24}
BEGIN{3:}{11:}{16:}J:=1;K:=1;P[1]:=2;{:16}
{18:}ORD:=2;SQUARE:=9;{:18};
WHILE K<M DO BEGIN{14:}REPEAT J:=J+2;{20:}
IF J=SQUARE THEN BEGIN ORD:=ORD+1;{21:}
SQUARE:=P[ORD]*P[ORD];{:21}{25:}
MULT[ORD-1]:=J;{:25};END{:20};{22:}N:=2;
JPRIME:=TRUE;
WHILE(N<ORD)AND JPRIME DO BEGIN{26:}
WHILE MULT[N]<J DO MULT[N]:=MULT[N]+P[N]+P[N]
;IF MULT[N]=J THEN JPRIME:=FALSE{:26};N:=N+1;
END{:22};UNTIL JPRIME{:14};K:=K+1;P[K]:=J;
END{:11};{8:}BEGIN PAGENUMBER:=1;
PAGEOFFSET:=1;
WHILE PAGEOFFSET<=M DO BEGIN{9:}
```

## Vantaggi

- Portabilità
- Personalizzazione (.CH)
- Adattabilità a tutti i linguaggi
- Generazione “automatica” di documentazione sempre aggiornata
- Programmazione come “flusso di coscienza” (ordine logico mentale)
- Minor numero di errori e, soprattutto, debug più veloce

## Limiti

- E' necessaria una certa capacità letteraria che non tutti gli informatici posseggono
- Richiede tempi di sviluppo più lunghi rispetto agli standard aziendali
- Non viene presa in considerazione la componente delle interfacce