– Typeset by GMNI & FoilTEX **–**

LENGUAJE FORTRAN: ORIGEN Y EVOLUCIÓN

F. Navarrina, I. Colominas, H. Gómez, J. París, M. Casteleiro

GMNI — GRUPO DE MÉTODOS NUMÉRICOS EN INGENIERÍA

Departamento de Métodos Matemáticos y de Representación
Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos
Universidad de A Coruña, España

e-mail: fnavarrina@udc.es
página web: http://caminos.udc.es/gmni

ÍNDICE

- Principales paradigmas de programación
- FORTRAN
- Lenguajes compilados
- FORTRAN 77
- Fortran 90/95/2003
- Fortran . . . ¿3000?

PRINCIPALES PARADIGMAS (en Ingeniería):

▶ Lenguaje Máquina:	Hardware	($<$ 1940 aprox.)
▶ Ensamblador:	Software	(1940–1950 aprox.)
▶ FORTRAN / II / IV:	Compiladores	(1950–1960 aprox.)
▶ BASIC:	Intérpretes	(1960–1970 aprox.)
▶ PASCAL:	Prog. Estructurada	(1970–1980 aprox.)
▶ FORTRAN 77:	(influencia del Pascal)	(1977)
▶ Lenguaje C:	Portabilidad, S.O. UNIX	(1980–1990 aprox.)
▶ Lenguaje C++:	Prog. orientada a objeto	(1990–2000 aprox.)
▶ Fortran 90/95/2003:	(influencia del C)	(1990,1995,2003)
▶ Java:	Internet	($>$ 2000 aprox.)

FORTRAN (I)

FORTRAN / II / IV

- Lenguaje de **ALTO NIVEL**. (*)
- Diseñado por John Backus para IBM en 1953, como alternativa al Lenguaje Máquina de un ordenador IBM 704.
- El primer compilador fue distribuido en 1957.

John Backus (1924–2007).
Creador del FORTRAN (1953) en IBM.
(Fuente: IBM)

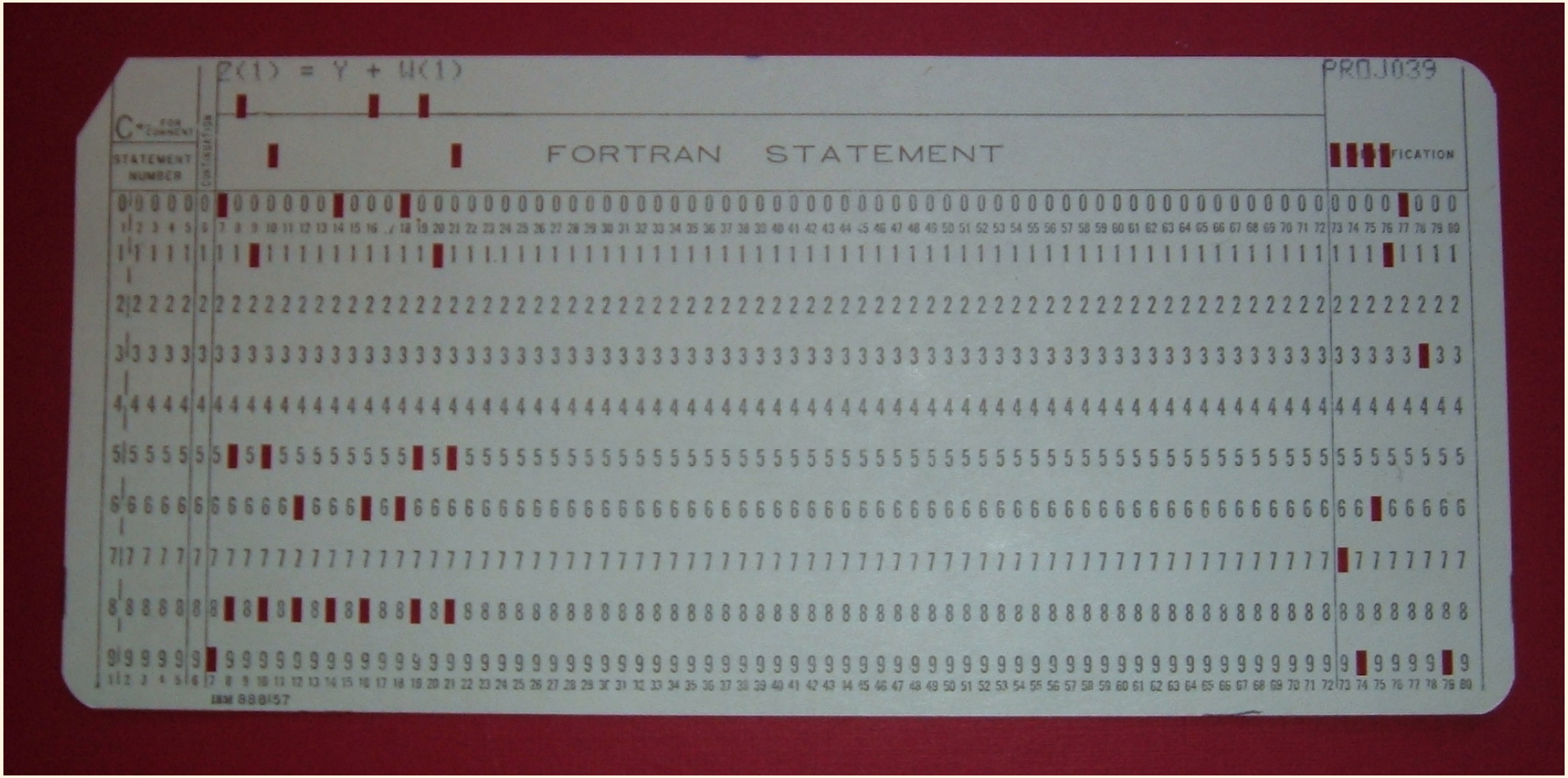
(*) Muy alejado del lenguaje máquina.

FORTRAN (II)

- **FORTRAN** proviene de **FORmula TRANslator** (“traductor de fórmulas”).
- La codificación de las fórmulas es sencilla.
- La sintaxis viene condicionada por el soporte (tarjetas perforadas).

En efecto . . .

- Origen: **Tarjetas Perforadas** (”punch cards”)



Tarjeta Perforada. (Fuente: <http://commons.wikimedia.org/wiki/Image:FortranCardPROJ039.agr.jpg>)

- Origen: **Perforadora de Tarjetas**

Perforadora de Tarjetas. (Fuente: <http://www.chilton-computing.org.uk/acl/technology/atlas/p013.htm>)

- Origen: **Programa FORTRAN en tarjetas perforadas**

Programa FORTRAN en tarjetas perforadas. (Fuente: <http://www.staff.ncl.ac.uk/roger.broughton/museum/iomedia/pc.htm>)

FORTRAN (IIId)

- Origen: **Lectora de Tarjetas**

Programa FORTRAN en tarjetas perforadas. (Fuente: <http://www.staff.ncl.ac.uk/roger.broughton/museum/iomedia/pc.htm>)

FORTRAN (IV)

- La gestión de memoria es rígida pero predecible. (*)
- Las instrucciones de control son muy primitivas. Básicamente...
  - **GOTO** incondicional
  - **IF aritmético**
  - **DO – CONTINUE**
  - **CALL**
- Modelo de implementación: **PROGRAMA PRINCIPAL + SUBRUTINAS**
  - **SUBRUTINA** = **subprograma**
    - **CALL** transfiere el control a un subprograma
    - que forma parte del **programa ejecutable** o
    - cuya localización conoce el **programa ejecutable** (ejemplo: librerías del sistema)
  - La transfencia de argumentos se realiza **POR REFERENCIA**.

(*) En principio (siempre que el programa esté bien hecho) cuando comienza la ejecución de un programa debe haber memoria suficiente para la realización de todos los cálculos, por lo que todo debería funcionar correctamente (sin detenciones por falta de recursos, "cuelgues" del sistema, etc.)

FORTRAN (V)

- Ejemplo: Programa **FORTRAN IV** para calcular factoriales

```
C     PROGRAMA PARA CALCULAR FACTORIALES
      READ(5,100) N
  100 FORMAT(I5)
      NFAC=1
  500 CONTINUE
      IF (N) 1000, 1000, 600
  600 CONTINUE
      NFAC=NFAC*N
      N=N-1
      GOTO 500
 1000 WRITE(6,110) NFAC
  110 FORMAT(I10)
      STOP
      END
```

LENGUAJES COMPILADOS

▶ FORTRAN ES EL PROTOTIPO DE LENGUAJE COMPILADO

- El programador escribe el **programa fuente** (uno o varios archivos *.f, *.for).
- Un programa (**compilador FORTRAN**) se encarga de **COMPILAR** (*)
  el **programa fuente** (uno o varios archivos *.f, *.for) y
  crear los correspondientes **programas objeto** (archivos *.o, *.obj)
- Otro programa (**linker**) se encarga de **LINKAR** (**)
  el/los **programa/s objeto** (archivos *.o, *.obj) y
  crear el correspondiente **programa ejecutable** (archivo *.exe o sin extensión).
- Una vez creado, el **programa ejecutable** (archivo *.exe o sin extensión)
  puede utilizarse cuantas veces sea necesario.

(*) Traducirlo/s a lenguaje máquina.

(**) Dimensionar la memoria necesaria y
unir (“link”) todas las partes.

Lenguajes Compilados (II)

Programa Fuente (*.f, *.for)

⇓ **COMPILACIÓN** [compilador]

Programa Objeto (*.o, *.obj)

⇓ **LINKADO** [linker]

Datos → **Programa Ejecutable (*.exe, sin ext.)**

⇓ **EJECUCIÓN** [loader o debugger (*)]

Resultados

(*) Debugger (depurador): loader que permite ejecutar paso a paso un programa y que tiene herramientas que permiten localizar dónde se producen los errores.

FORTRAN 77 (I)

FORTRAN 77

- El **FORTRAN** se acomoda a las nuevas tendencias...
  - el lenguaje existe como standard ANSI a partir de 1966,
  - se evita el uso del **GOTO** incondicional,
  - se cambia profundamente el estilo de programación para que la lógica de los programas sea más evidente, y
  - se introducen muchas extensiones (no standard) en los compiladores (*)
- **FORTRAN 77** incorpora nuevas instrucciones de control más sofisticadas como...
  - **DO – ENDDO** (**)
  - **DO WHILE – ENDDO**
  - **IF – THEN– ELSE – ENDIF**

(*) Finalmente muchas de estas extensiones han acabado formando parte del standard.

(**) ENDDO fue una extensión (no standard) que acabó formando parte del standard.

FORTRAN 77 (II)

- Ejemplo: Programa **FORTRAN 77** para calcular factoriales

```
C     Programa para calcular factoriales

      read(5,*) n

      nfac=1

      do while (n.gt.0)
        nfac=nfac*n
        n=n-1
      enddo

      write(6,*) nfac

      stop
      end
```

Fortran 90/95/2003

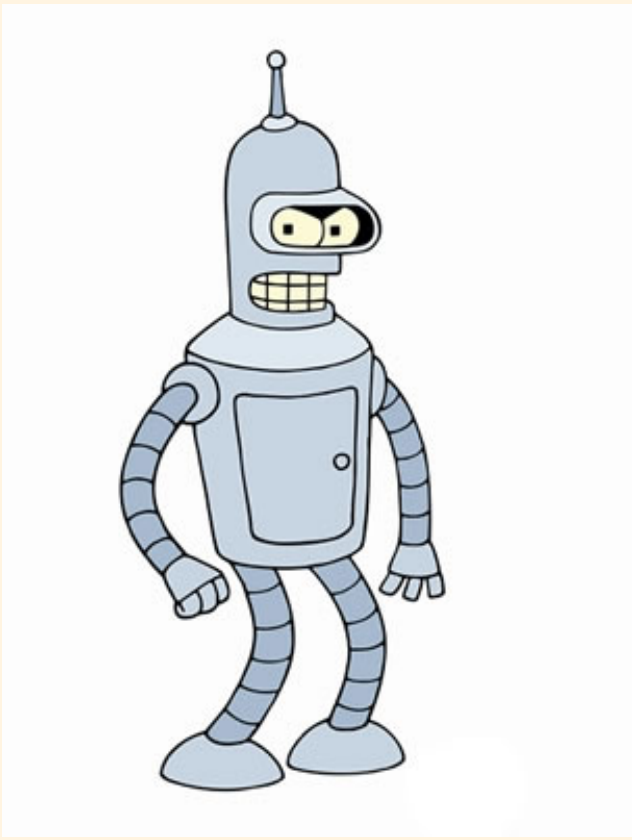
- El **Fortran** se acomoda a las nuevas tendencias (una vez más)...
  - El lenguaje existe como standard ANSI a partir de 1992.
  - Se incorporan conceptos propios de otros lenguajes y estilos de programación:
    - Declaración **ALLOCATABLE** e instrucción **ALLOCATE**. (*)
    - Archivos fuente tipo **Free-form** (más de 80 carácteres por línea).
    - Punteros, recursividad, operaciones vectoriales, prog. orientada a objetos, etc.
  - Se procura eliminar inconsistencias internas y causas de errores frecuentes:
    - En Fortran 90 se declaran **obsoletas** algunas instrucciones, aunque se permite su uso. En Fortran 95 se prohíbe su utilización. (**)
  - El nuevo lenguaje es más potente y versátil que sus precedentes, pero también
    - menos intuitivo y más difícil de aprender, [$\Longrightarrow$ no gusta a programadores noveles]
    - menos atractivo que otros lenguajes modernos. [$\Longrightarrow$ no gusta a programadores profesionales]

(*) Es posible dimensionar memoria adicional en tiempo de ejecución.

(**) Algunos programas antiguos no pueden ser recompilados directamente.

Fortran . . . ¿3000? (Ia)

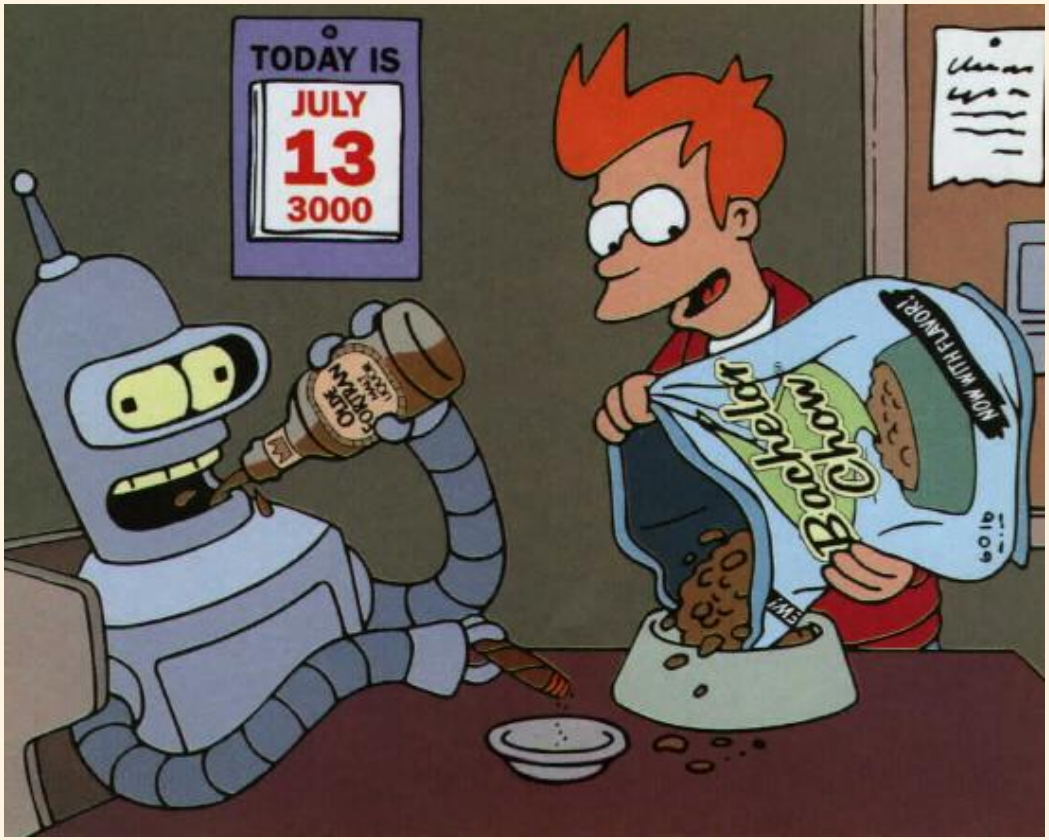
- **¿Perdurará el lenguaje Fortran?**

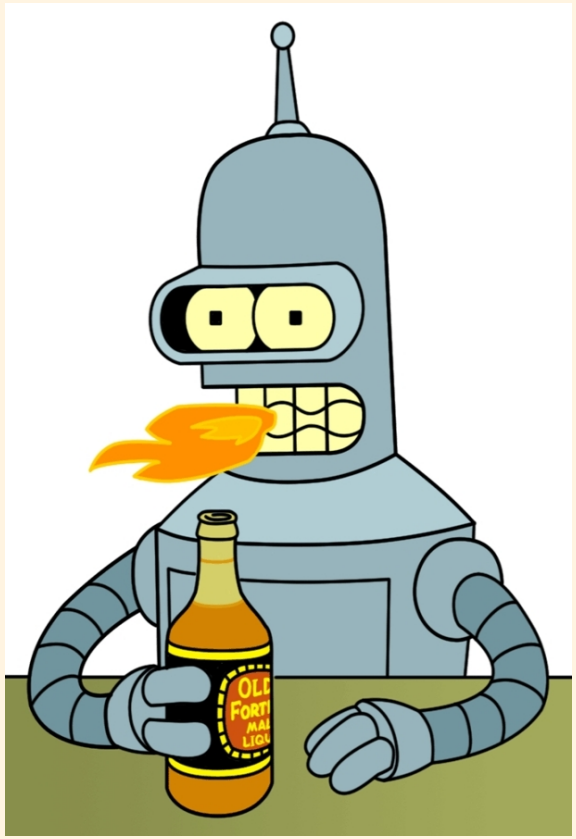


Cartel de FUTURAMA (izda.) y Robot BENDER (derecha).
[Matt Groening y David X. Cohen, FOX Broadcasting Company]

Fortran . . . ¿3000? (Ib)

- **¿Perdurará el lenguaje Fortran?**





BENDER con FRY (izda.) y BENDER (derecha) bebiendo.
[Matt Groening y David X. Cohen, FOX Broadcasting Company]

Fortran . . . ¿3000? (Ic)

- **¿Perdurará el lenguaje Fortran?**

Caja (izda.) y botella (derecha) de OLDE FORTRAN.
[Matt Groening y David X. Cohen, FOX Broadcasting Company]