Introdução ao fortran 90 - Aula 1

Alexandre Diehl

Departamento de Física – UFPel

Histórico

1954-1957: The IBM Mathematical Formula Translating System

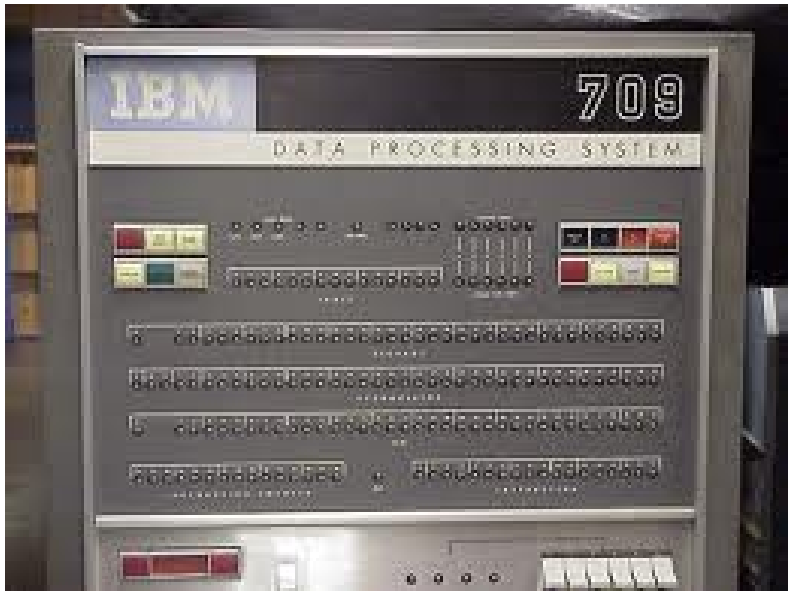
John W. Backus da IBM:

Fortran I: alternativa à linguagem assembler para a programação do IBM 704.

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77



→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

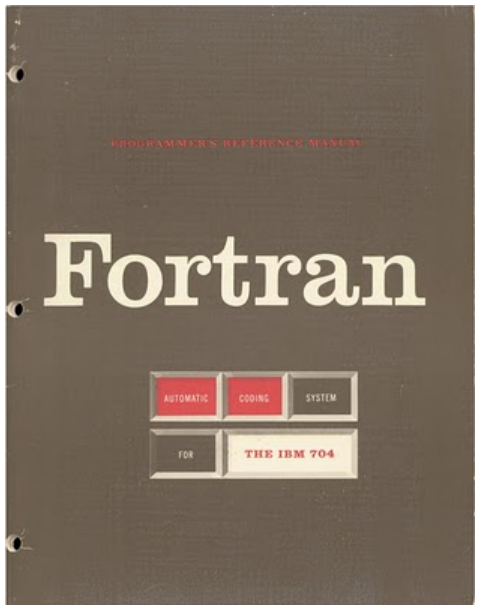
- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

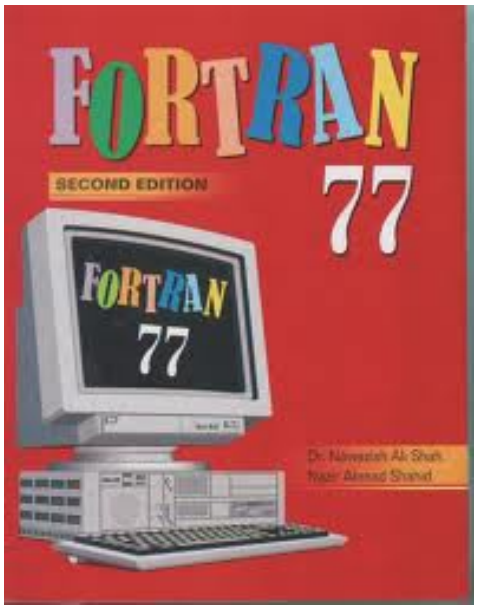


→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

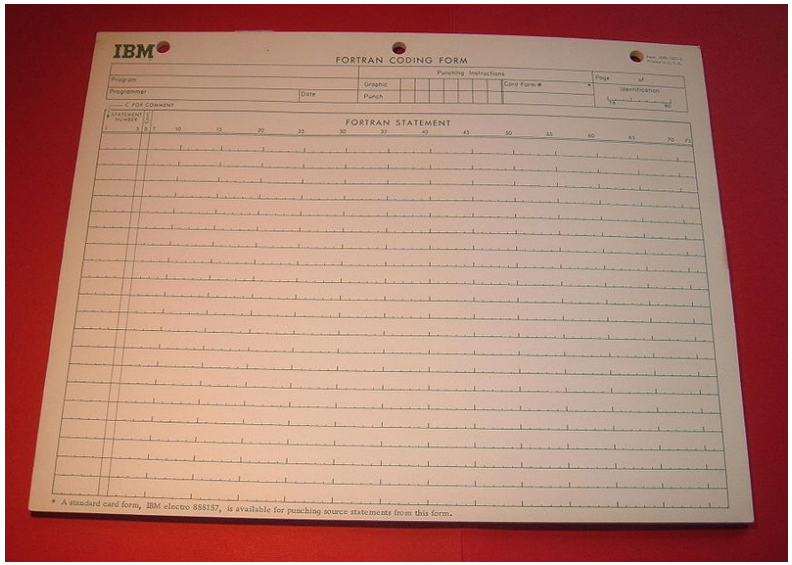


→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77



→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

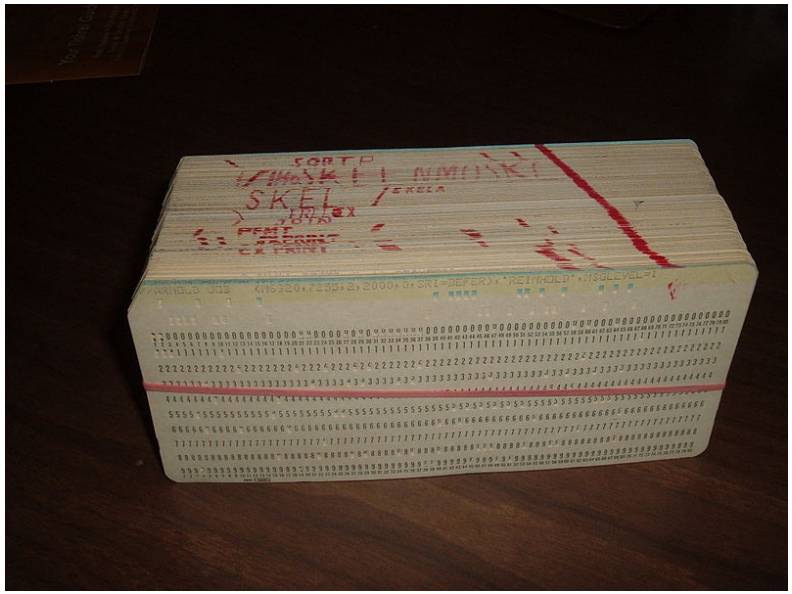
- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77



→ Padrão fixo: formatação do código baseada no cartão perfurado

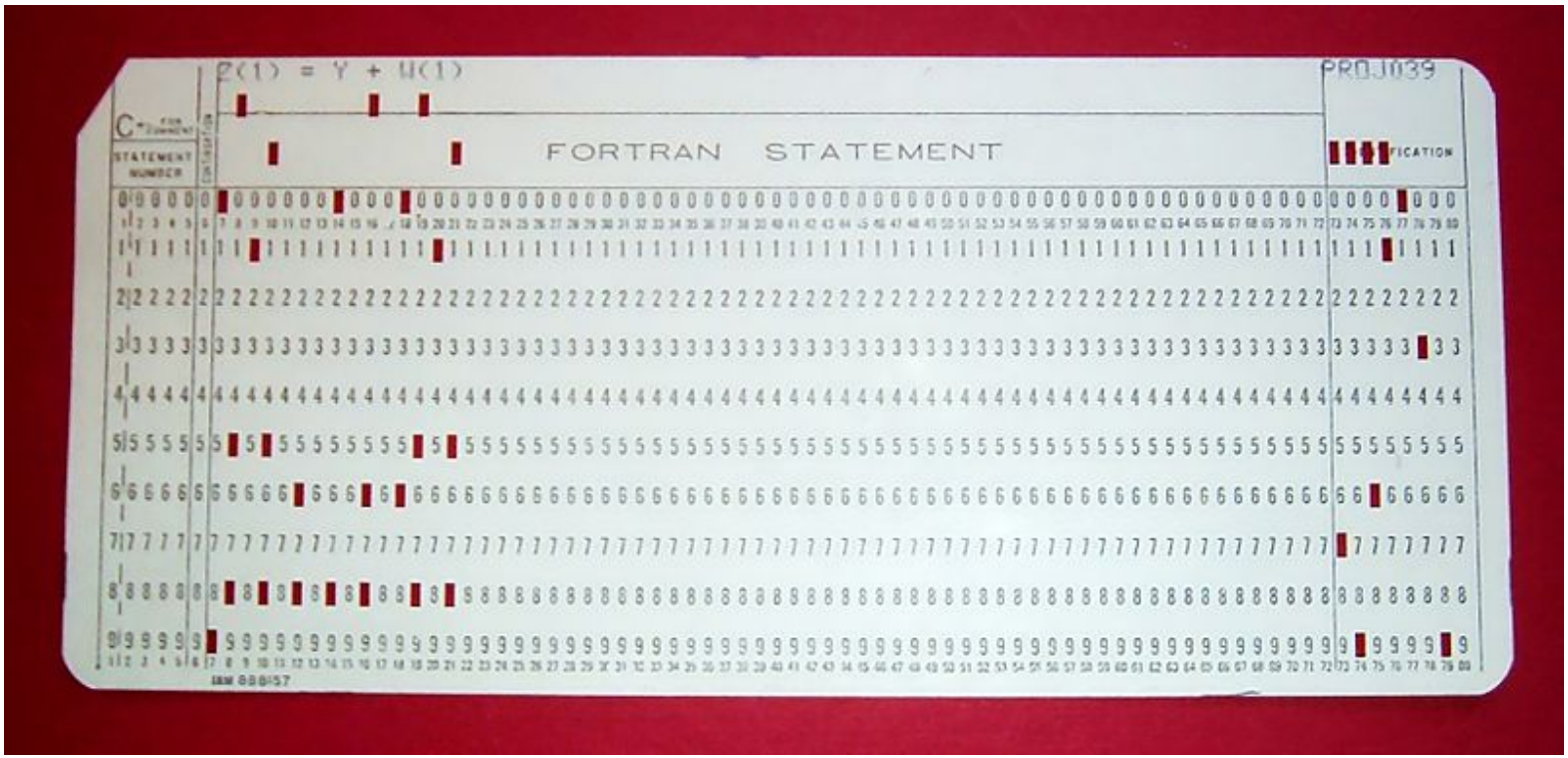
Histórico

1957-1978: Atualizações do FORTRAN disponíveis

- 1958: FORTRAN II e III
- 1961: FORTRAN IV
- 1963: mais de 40 compiladores FORTRAN a disposição
- 1966: FORTRAN 66 como padrão
- 1978: FORTRAN 77

→ Padrão fixo: formatação do código baseada no cartão perfurado

1957-1978: Estrutura de um programa em FORTRAN



1957-1978: Estrutura de um programa em FORTRAN

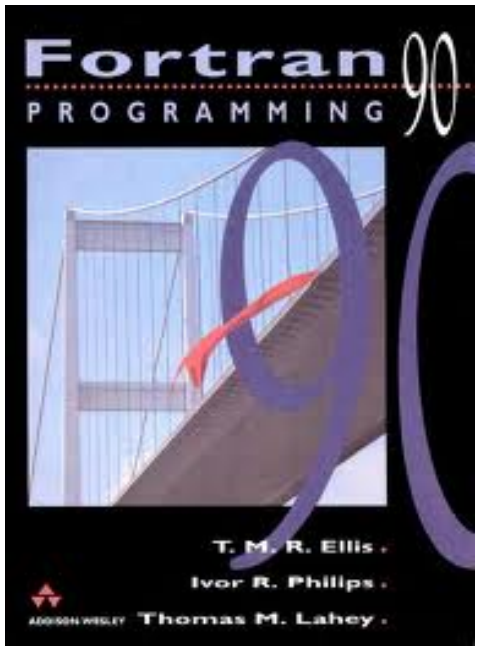
```
C FORTRAN IV program to
C find sum of integers 1-100
      INTEGER I, SUM
      SUM=0
      DO 1 I=1, 100
    1 SUM = SUM + I
      WRITE (6,2) SUM
    2 FORMAT(1X,I5)
      STOP
      END
```

FIGURE 114. FORTRAN IV program

Histórico

1991-hoje: uma nova versão para o Fortran

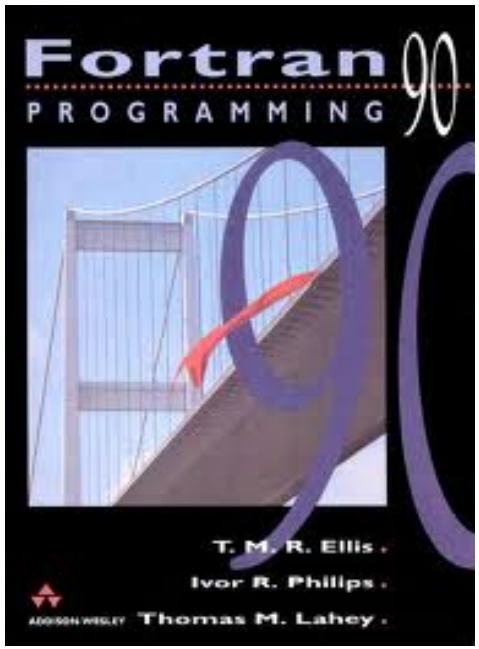
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



Histórico

1991-hoje: uma nova versão para o Fortran

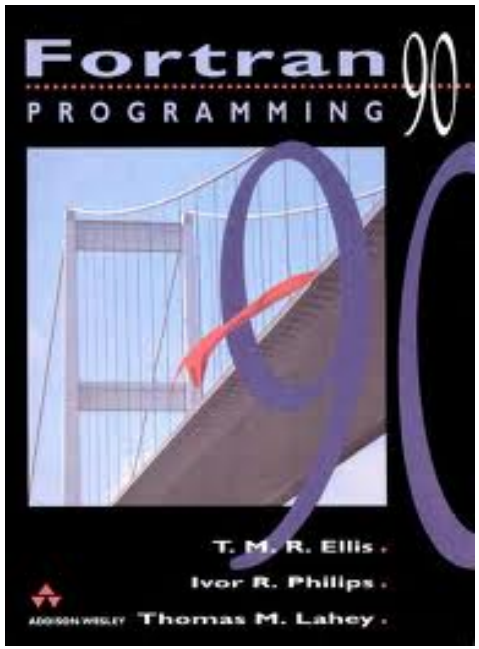
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



Histórico

1991-hoje: uma nova versão para o Fortran

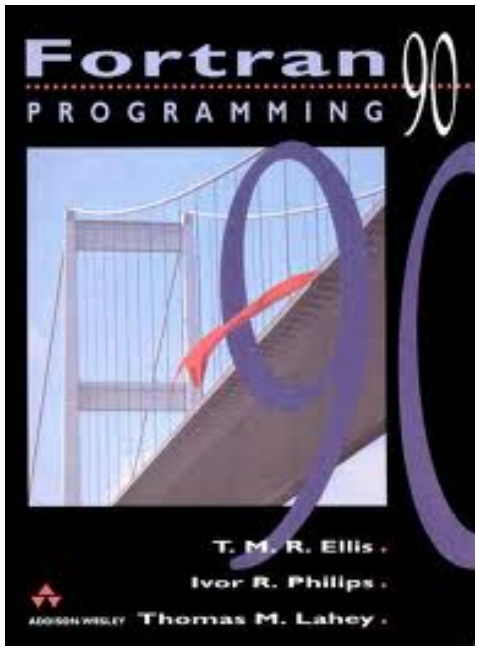
- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



Histórico

1991-hoje: uma nova versão para o Fortran

- 1992: Fortran 90
  Fim da formatação do cartão!
- 1997: Fortran 95
  Fortran de alta performance.
- 2004: Fortran 2003
  Na direção da programação orientada à objeto.
- 2010: Fortran 2008
  Atualização do Fortran 2003.



Variáveis e constantes: o que é permitido nos nomes

- nomes com até 31 caracteres: não exagere
  → use nomes que signifiquem algo
- uso de maiúsculas e minúsculas: f90 não é case sensitive
  → passos, Passos, pAssOs, PASSOS são a mesma coisa
- o primeiro caracter deve ser uma letra
- os outros caracteres podem ser letras, números ou _
  → const1, number_of_steps, hora1var são permitidos

Variáveis e constantes: o que não é permitido nos nomes

- o uso de espaços:
  → soma total é errado. Use soma_total
- o uso de hífen:
  → soma-total é errado. Use soma_total
- o uso de acentos, cedilha, pontuação (ponto, vírgula), !, &, % não são permitidos:
  → torção é errado. Use torcao ou torsion

Linhas de comando: o que é permitido

- a linha pode começar na coluna 1
- até 132 caracteres por linha
- use o caracter & para indicar a continuação da linha
- use o caracter ! no início de um comentário
- mais de um comando por linha são separados por ;
  → a = 1; b = 2; c = 3
- use espaços entre nomes e comandos para melhor vizualização
  → a=1 é a mesma coisa que a = 1
- use recuos (formatação) para melhor vizualização

Estrutura básica de um programa em Fortran 90

```
PROGRAM meu_primeiro
   IMPLICIT NONE ! para evitar declarações implícitas

   Comandos de Declaração:REAL, INTEGER, PARAMETER

   Comandos de Execução: IF-ENDIF, DO-ENDDO, atribuição

END PROGRAM meu_primeiro
```

Tipos de dados no Fortran 90

- Numérico:

  ```
  REAL :: a
  INTEGER :: i
  ```

- Caracter:

  ```
  CHARACTER :: answer
  CHARACTER(LEN=10) :: nome
  ```

- Lógico ou booleno:

  ```
  LOGICAL :: overlap
  ```

Variáveis do tipo INTEGER: 4 bytes, 8 bytes e 16 bytes

→ são armazenadas apenas a parte inteira de um número

- INTEGER(KIND=4) ou INTEGER(4) ou INTEGER

  → números inteiros entre -2^{31} até $+2^{31}$
- INTEGER(KIND=8) ou INTEGER(8)

  → números inteiros entre -2^{63} até $+2^{63}$
- INTEGER(KIND=16) ou INTEGER(16)

  → números inteiros entre -2^{127} até $+2^{127}$

Variáveis do tipo REAL: precisão simples, dupla e quádrupla

→ também chamados de números com ponto flutuante

- REAL(KIND=4) ou REAL(4) ou REAL

  → 8 casas decimais: 3.40282347E+38 é o maior número
- REAL(KIND=8) ou REAL(8)

  → 16 casas decimais: 1.7976931348623157E+308 é o maior número
- REAL(KIND=16) ou REAL(16)

  → 35 casas decimais:

  1.189731495357231765085759326628007 02E+4932 é o maior número

Variáveis do tipo COMPLEX: precisão simples, dupla e quádrupla

→ número complexo $a + ib$ é representado pelo par (a,b)

- COMPLEX(KIND=4) ou COMPLEX(4) ou COMPLEX
- COMPLEX(KIND=8) ou COMPLEX(8)
- COMPLEX(KIND=16) ou COMPLEX(16)

Variáveis do tipo CHARACTER: 1 byte por caracter

→ também chamadas de strings

- Conjunto de caracteres contidos entre aspas ou apóstrofes

  → 'bom Dia', "bom Dia", 'meio-dia'

  → a palavra (ou string) é lida literalmente

- CHARACTER(LEN=???) o comprimento (LEN) da palavra é definido na declaração da variável

  → CHARACTER(LEN=10) :: word
  → CHARACTER(10) :: word

  indicam que a variável word tem no máximo 10 caracteres

Variáveis do tipo LOGICAL: também chamadas de boolenas

→ Variáveis lógicas que assumem os valores verdadeiro e falso

- LOGICAL :: overlap ! declara a variável overlap como lógica
- overlap = .true. ! atribui à variável overlap o valor verdadeiro
- Variáveis lógicas são usadas em estruturas condicionais

CONSTANTES no Fortran 90:

Declaradas no início do programa, pois não mudam durante a execução

- INTEGER, PARAMETER :: n = 1000
- REAL(8), PARAMETER :: pi = 3.14159274
- LOGICAL, PARAMETER :: overlap = .false.

Comandos de entrada e saída de dados: formato livre

- READ*, a
- PRINT*, b
- PRINT*, “a=”,a

READ (*,*) a

WRITE (*,*) a

WRITE (*,*)’a=’,a

O formato livre, embora prático, não permite grandes manipulações com os dados.

FORMATAÇÃO de dados

- Variável inteira: devemos prever a posição do sinal

  -12034 → I6 29883904 → I8

- Variável real: devemos prever a posição do sinal e do ponto decimal

  +10.2398 → F8.4 -0.1298345 → F10.7

- Variável real: formato exponencial

  +0.2283490E+03 → E14.7 -7.8723E-10 → E11.4