Introdução à Programação na Linguagem F

Jaime Ramos, Luís Cruz-Filipe

DMIST, Dezembro de 2006

Capítulo 8

Formatação de I/O

Objectivos

Formatação de saída

A formatação de saída consiste numa lista de códigos, separados por vírgulas e delimitados por parênteses. Existem cinco categorias para estes códigos: `f`, `es`, `a`, `l`, e `i`.

Os códigos da categoria `f` (*floating point*) utilizam-se para formatar a apresentação de números reais em notação de vírgula flutuante. A sua estrutura é da forma `f`n`.`d em que n define o número total de símbolos que serão usados na escrita do número real (incluindo o ponto decimal e o eventual sinal `-`). Por exemplo, o código `f6.2` indica que se pretende escrever um número real usando, no máximo, 6 símbolos e com duas casas decimais. Assim, o maior número que se pode escrever com este código é `999.99` e o menor é `-99.99`. No caso de o número não necessitar de todos os símbolos, as posições mais à esquerda são ocupadas por espaços em branco. No caso de o número necessitar de mais símbolos aparecem no ecrã asteriscos em vez do número. Os três comandos seguintes

```
print *, sqrt(2.0)
print "(f6.2)", sqrt(2.0)
print "(f7.3)", sqrt(2.0)
```

produzem o seguinte resultado no ecrã.

```
␣1.4142135
␣␣1.41
␣␣1.414
```

Os códigos da categoria `es` (*engineer and science*) utilizam-se para formatar a apresentação de números reais em notação científica. A estrutura deste código é da forma `es`n`.`d com o mesmo significado que anteriormente. Por exemplo, o código `es8.2` indica que se pretende escrever um número real em notação científica usando, no máximo, 8 símbolos e com duas casas decimais. Os três comandos seguintes

```
print *, sqrt(20000.0)
print "(es8.2)", sqrt(20000.0)
print "(es9.3)", sqrt(20000.0)
```

produzem o seguinte resultado no ecrã.

```
␣141.42136
1.41E+02
1.414E+02
```

Os códigos da categoria `i` (*integer*) utilizam-se para formatar a apresentação de números inteiros. A sua estrutura é da forma `i`n em que n define o número total de símbolos que serão usados na escrita do número inteiro. Por exemplo, o código `i4` indica que se pretende escrever um número inteiro usando, no máximo, 4 símbolos. Neste caso, o maior número inteiro que se pode escrever com este código é `9999` e o menor é `-999`. Por exemplo, os dois comandos seguintes

```
print *,123, 43, 12
print "(i4,i4,i4)",123,43,12
```

produzem o seguinte resultado no ecrã

```
␣123␣43␣12
␣123␣␣43␣␣12
```

Os códigos da categoria `a` (*alphanumeric*) utilizam-se para formatar a apresentação de sequências de símbolos, isto é, de tipo `character`. Por exemplo,

```
print "(a)","1+1=2"
```

produz o seguinte resultado no ecrã.

```
1+1=2
```

Escrita em ficheiros

É possível escrever informação em ficheiros. Não se pretende aqui descrever exaustivamente todas as primitivas de escrita em ficheiros, mas apenas as instruções básicas.

O primeiro passo consiste em *abrir* o ficheiro para escrita. O comando `open` tem a seguinte sintaxe:

`open(unit=`*ficheiro*`, file=`*nome_ficheiro*`, status=`*modo*`, access=`*acesso*`, action=`*acção*`)`

em que:

- *ficheiro* é um número inteiro (por exemplo 2) que referencia um ficheiro externo;
- *nome_ficheiro* é nome físico do ficheiro entre aspas (por exemplo, `"result.txt"`);
- *modo* é uma string que especifica o modo de abertura do ficheiro. Pode ser `"new"`, `"old"`, `"replace"` ou `"scratch"`. O modo `"old"` refere-se a um ficheiro que tem que existir; o modo `"new"` refere-se a um ficheiro que não pode existir (vai ser criado); no modo `"replace"` se o ficheiro não existir, é criado, se existir é apagado e criado um novo ficheiro vazio com o mesmo nome. Recomenda-se a utilização deste último modo quando se desconhece o estado do ficheiro.
- *acesso* é uma string que especifica o tipo de acesso ao ficheiro. Pode ser `"direct"` ou `"sequential"`. Recomenda-se a utilização do modo `"sequential"`.
- *acção* é uma string que especifica o tipo de acção a realizar: `"read"`, `"write"` ou `"readwrite"`.

Por exemplo, suponha-se que se pretende criar um novo ficheiro `result.txt` para escrever informação. O comando é o seguinte

```
open(unit=2, file="result.txt", status="replace", access="sequential", action="write")
```

Uma vez aberto o ficheiro para escrita, pode *escrever-se* informação através do comando `write`. A sintaxe do comando é a seguinte:

`write(unit=`*ficheiro*`, fmt=`*especificação*`)` *dados*

em que:

- *ficheiro* é um número inteiro que referencia um ficheiro externo (já aberto);
- *especificação* é uma lista de códigos de formatação (como descritos na secção anterior) a ser aplicada a *dados*;
- *dados* é a informação que se pretende escrever.

Por exemplo, após ter aberto o ficheiro `result.txt` como descrito acima, o comando

```
write(unit=2,fmt="(a,i2)") "1+1="1+1
```

escreve no referido ficheiro a seguinte informação:

```
1+1=␣2
```

No fim, é necessário *fechar* o ficheiro. Para tal, utiliza-se o comando `close`, cuja sintaxe é a seguinte:

`close(unit=`*ficheiro*`)`

em que *ficheiro* tem o mesmo significado que acima. Por exemplo, para fechar o ficheiro `result.txt` usar-se-ia o comando seguinte.

```
close(unit=2)
```