Inclusões e Subprogramas

Nos capítulos anteriores vimos como construir um desenho em função de parâmetros; aqui trataremos de recursos que permitem isolar a definição destes parâmetros do programa que monta o desenho, separando claramente duas classes de arquivos: uma de dados para desenho de um determinado projeto, e a outra de programas de desenho, aplicáveis a qualquer número de projetos diferentes.

Inclusão de arquivos - INCLUI

A forma mais simples de montagem de um programa DP independente dos parâmetros usados é o comando INCLUI. Este comando desvia a execução de um arquivo DP para outro, retornando no final do arquivo incluído para o arquivo original. Fisicamente, é como se um arquivo inteiro especificado pelo usuário fosse incluído em uma única linha de programa, com o comando INCLUI. A sintaxe do comando é:

```
INCLUI 'arquivo' [ OPCIONAL ]
```

O 'arquivo' especificado entre apóstrofes segue as convenções do MS-DOS; se não tiver tipo, será adotado .DP. O processamento será interrompido se o arquivo incluído não existir, a menos que a palavra OPCIONAL seja fornecida.

Vamos a partir do EXEMPLO2.DP fazer o EXEMPLO3.DP. Agora, em vez de um paralelepípedo vamos desenhar dois. O primeiro terá medidas 400x150x100 e o segundo 200x200x200. O segundo paralelepípedo terá o título "SEGUNDO". O nome do primeiro arquivo de desenho será EXEMPL3A e o segundo EXEMPL3B:

```
DEFINE ESCALA 50
DESENHO 'EXEMPL3A'
N COMPRIM= 400
N LARGURA= 150
N ALTURA= 100
A TITULO= 'PARALELEPIPEDO'
INCLUI 'PARALEL'
FIM
DESENHO 'EXEMPL3B'
N COMPRIM= 200
N LARGURA= 200
N ALTURA= 200
A TITULO= 'SEGUNDO'
INCLUI 'PARALEL'
```

```
FIM
```

No EXEMPLO3.DP não há qualquer programação de desenho, apenas a indicação do nome do desenho e dos parâmetros necessários para a sua montagem. Este arquivo funciona assim como uma entrada de dados para um programa que processa desenhos.

O comando INCLUI foi usado duas vezes neste exemplo. Em cada uma delas, o arquivo PARALEL.DP foi inteiramente lido e processado, e no final do arquivo, o controle voltou ao EXEMPLO3.DP.

Faltou mostrar o conteúdo do arquivo PARALEL.DP. Literalmente, é um pedaço do EXEMPLO2.DP:

```
$PARALEL.DP:Desenho de um paralelepipedo de lados
$COMPRIM, LARGURA e ALTURA e titulo
$TITULO
$
1 0,0
3 1 @%COMPRIM,%LARGURA
2 X3,Y1
4 X1,Y3
POLIGONAL 1 2 3 4 1
$
$Vista frontal
$
8 1 @ 0,-1.3*%_ESCALA
6 8 @ %COMPRIM,-%ALTURA
5 X8,Y6
7 X6,Y8
POLIGONAL 5 6 7 8 5
$
$Vista lateral
$
9 2 @ 2.5*%_ESCALA,0
11 9 @ %ALTURA,%LARGURA
10 X11,Y9
12 X9,Y11
```

```
POLIGONAL 9 10 11 12 9
$
$Nos auxiliares
$
13 5 @ -1*%_ESCALA,0
14 4 @ 0,1*%_ESCALA
15 (X1 + X3)/2,(Y1 + Y3)/2
$
$Cotagens e titulo
$
COTAGEM VERTICAL 5 8 13
COTAGEM VERTICAL 1 4 13
COTAGEM HORIZONTAL 4 3 14
COTAGEM HORIZONTAL 12 11 14
TEXTO 15 CENTRADO '%TITULO'
```

Observe que o arquivo PARALEL.DP não tem o comando FIM, que termina o desenho. Isto aconteceu pois escolheu-se (arbitrariamente) colocar o FIM do desenho no arquivo DP original.

Arquivos de inclusão podem INCluir outros arquivos.

Biblioteca de Inclusão

Não é boa prática de programação misturar informações temporárias de projeto com informações permanentes, tais como programas DP. O ideal é que programas DP sejam colocados em uma pasta separada. Por exemplo, se o programa PARALEL.DP estiver na pasta \DP\INCLUI, para inclui-lo faremos:

```
INCLUI'\DP\INCLUI\PARALEL'
```

para facilitar a separação de arquivos em bibliotecas (onde uma pasta independente pode ser considerada uma biblioteca), existe o comando:

```
DEFINE BIBINC 'pasta(s)'
```

Este comando define o nome de uma ou mais pastas de pesquisa de inclusão, separados por ponto e vírgula.

Suponha os comandos:

```
DEFINE BIBINC '\DP\INCLUI;\DP\SOLIDOS'
INCLUI 'PARALEL'
```

O DP fará as seguintes operações:

Incluirá o arquivo PARALEL.DP da pasta atual. Se não achar:

Incluirá o arquivo PARALEL.DP da pasta \DP\INCLUI. Se não achar:

Incluirá o arquivo PARALEL.DP da pasta \DP\SOLIDOS. Se não achar, emitirá mensagem de erro.

Definições tais como BIBINC podem ser permanentemente definidas através do uso do arquivo de critérios do DP.

Arquivo de Critérios

Toda vez que um programa é executado, o DP aciona internamente o seguinte comando, antes de qualquer outro:

```
INCLUI '%_SUPORTE\DP\INSTAL'
```

O comando acima não é listado. Se o arquivo INSTAL.DP na pasta %_SUPORTE\DP\ for encontrado, será automaticamente incluído em todo processamento. Este arquivo normalmente é usado para declaração de parâmetros com valores diferentes do default do sistema. Por exemplo, poderia conter a definição anterior de biblioteca de inclusão:

```
DEFINE BIBINC '\DP\INCLUI;\DP\SOLIDOS'
```

Como resultado, o projetista deixa de se preocupar com a localização física dos programas, precisando conhecer apenas o seu nome. A TQS distribui o DP com o seguinte arquivo INSTAL.DP:

```
$------------------ %_SUPORTE\DP\INSTAL.DP ---------------------
$
DEFINE BIBDP '%_SUPORTE\DP\DPS' $ biblioteca DPS
DEFINE BIBBLO '%_SUPORTE\BLOCOS\GERAIS' $ biblioteca de blocos
$------------------ %_SUPORTE\DP\INSTAL.DP ---------------------
```

Note que após a variável '%_SUPORTE' não temos %% como descrito no capítulo Atribuição de variáveis

alfanuméricas (pág. 61) pois o próximo caractere (\) não é um caractere válido para nome de variável. Assim sendo, o DP ignora o restante do texto para definir o nome da variável.

Subprogramas DPS - Comando DP

O comando INCLUI é uma forma simples de criação de programas, mas tem o inconveniente de não isolar as variáveis do programa incluído das do programa principal. Se mais de um programa for incluído para a geração de um mesmo desenho, o risco aumenta, e problemas podem aparecer sem que o projetista perceba.

Subprogramas são um tipo especial de inclusão, onde os parâmetros de desenho são passados para o arquivo de inclusão e as variáveis do arquivo incluído não interferem de modo algum com outras variáveis definidas (incluindo nós). Estas variáveis são chamadas de variáveis locais.

Subprogramas são codificados em arquivos com o tipo .DPS. O tipo diferente de arquivo tem por objetivo separar os arquivos DP, normalmente de dados, dos arquivos DPS, usados para codificação de programas.

Antes de entrarmos no conceito de subprogramas, vamos examinar o que são variáveis locais e globais.

Variáveis locais e globais

Variáveis locais a um programa são conhecidas exclusivamente dentro deste programa. Variáveis locais são criadas:

Dentro da seção DESENHO. Duas seções diferentes de desenho não interferem uma com a outra;

Dentro de subprogramas DPS.

Variáveis globais são variáveis conhecidas em qualquer lugar, dentro de uma seção de DESENHO ou subprograma. Assim, uma variável global que assume um valor dentro de um desenho, pode ser usada no desenho seguinte.

Um caso especial de variáveis globais são as variáveis do sistema, começadas pelo sinal "_". Estas variáveis podem ser usadas a qualquer momento em qualquer lugar.

Variáveis globais podem ser declaradas no arquivo de critérios, estando então disponíveis permanentemente.

Declaração de variáveis locais

Variáveis locais são criadas na primeira vez que são usadas. O DP permite também, para efeito de melhor organizar um programa, que as variáveis locais sejam declaradas separadamente numa seção de programa, com a sintaxe:

```
LOCAIS
```

```
(...declaração de variáveis...)
FIM
```

A declaração de uma variável é idêntica a um comando de atribuição:

```
NUM nome [[=] valor ]
ALF nome [[=] 'texto' ]
COO nome [[=] coord ]
```

Variáveis declaradas podem receber um valor de inicialização. Variáveis não inicializadas recebem o valor nulo correspondente ao tipo. Um exemplo de codificação:

```
LOCAIS
N COMPRIM
N LARGURA
N ALTURA
A TITULO = 'PARALELEPIPEDO'
FIM
```

Declaração de variáveis globais

Variáveis declaradas fora da seção DESENHO são automaticamente globais. Dentro da seção DESENHO e de subprogramas, variáveis globais podem ser declaradas na seção GLOBAIS:

```
GLOBAIS
(...declaração de variáveis...)
FIM
```

A declaração de variáveis globais é idêntica a de locais. Existe um pequeno artifício quanto a inicialização de variáveis globais: a inicialização só é executada se a variável não estiver definida; no caso da declaração de uma variável que já existe, ela é ignorada.

Seja por exemplo o arquivo .DPS com os seguintes comandos:

```
GLOBAIS
ATEX= 'PRIMEIRA VEZ'
FIM
```

```
TEXTO0,0 '%TEX'
ATEX= 'SEGUNDA VEZ'
```

Supondo a variável TEX indefinida, na primeira vez que este arquivo for executado, o texto 'PRIMEIRA VEZ' será colocado no desenho; na segunda vez, o texto será 'SEGUNDA VEZ', pois a variável TEX já definida não será reinicializada.

Chamada de um subprograma

A chamada de um subprograma em um arquivo DPS é parecida com a INClusão de um arquivo, a menos do fato que os parâmetros do subprograma são definidos durante a chamada. A sintaxe é:

```
DP 'nome'[coord] [ANG angulo] (
parametro1=valor1, parametro2=valor2, ...)
```

Neste comando:

`'nome'` É o nome do subprograma com tipo .DPS, entre apóstrofes. O tipo não precisa ser fornecido;

`coord` Coordenadas opcionais de inserção do desenho gerado pelo subprograma;

`ANG` ânguloÂngulo opcional, que girará todos os elementos de desenho gerados pelo subprograma.

Os parâmetros do subprograma são passados após a abertura de parênteses, terminando com o fechamento de parênteses. Os parâmetros podem ser declarados na mesma linha, separados (ou não) por vírgulas ou em mais de uma linha, sem sinal de continuação. Subprogramas podem por sua vez chamar outros subprogramas.

Vamos criar o EXEMPLO4.DP a partir do EXEMPLO3.DP, mas agora, chamando o subprograma PARADP.DPS em vez de incluir o arquivo. O EXEMPLO4.DP terá seguinte forma:

```
DEFINE ESCALA 50
DESENHO 'EXEMPL4A'
DP 'PARADP' (
N COMPRIM = 400
N LARGURA = 150
N ALTURA = 100
```

```
  A TITULO = 'PARALELEPIPEDO' )
  FIM
```

```
DESENHO 'EXEMPL4B'
DP 'PARADP' (
N COMPRIM = 200
N LARGURA = 200
N ALTURA = 200
A TITULO = 'SEGUNDO' )
FIM
```

A mudança em relação ao exemplo anterior foi aparentemente pequena, apenas na sintaxe. O subprograma, no entanto, tem as suas próprias variáveis locais, e não corre o risco de interferir com outros subprogramas.

Os parâmetros passados ao subprograma devem ser especialmente declarados dentro do subprograma. Se forem passados parâmetros a menos, valores default serão assumidos.

Declaração de Parâmetros

A declaração de parâmetros dentro do subprograma é feita na seção PARAMETROS. A seção PARAMETROS deve ser declarada antes de qualquer outra seção de programa. O seu objetivo é de permitir que o DP verifique se foram declarados parâmetros que não pertençam ao subprograma.

A sintaxe desta declaração é:

```
  PARAMETROS
  (...declaração de variáveis...)
  FIM
```

A sintaxe de declaração de cada parâmetro é idêntica à das outras seções de declaração, a menos da definição opcional de COMENTÁRIO:

```
  NUM nome [[=] valor ] [ COM 'texto']
  ALF nome [[=] 'texto' ] [ COM 'texto']
  COO nome [[=] coord ] [ COM 'texto']
```

O comentário é um texto opcional que descreve sucintamente (até 40 caracteres) o objetivo do parâmetro. Comentários são desprezados no processamento normal do DP, mas são usados pelo gerador de programas para perguntar ao projetista o valor de cada variável de forma interativa. A digitação de dados será analisado no capítulo Digitação de dados de desenho.

Um ponto importante é que variáveis podem ser inicializadas na seção de parâmetros. Se parâmetros deixarem de ser declarados na chamada de um subprograma, automaticamente serão inicializados com o valor declarado nesta seção.

Parâmetros só podem passar do programa principal para um subprograma, não o contrário. A única forma de um programa usar um valor calculado no subprograma é através da utilização de variáveis globais.

O programa PARALEL.DPS citado no exemplo anterior será parecido com o PARALEL.DP, a menos da seção de PARÂMETROS no início e do comando FIM no fim do arquivo:

```
$ PARADP.DPS - Desenho de um paralelepipedo de lados
$ COMPRIM, LARGURA e ALTURA e título TITULO
$
PARAMETROS
N COMPRIM COM 'Comprimento em cm'
N LARGURA COM 'Largura em cm'
N ALTURA COM 'Altura em cm'
A TITULO COM 'Titulo do paralelepipedo'
FIM
1 0,0
3 1 @ %COMPRIM,%LARGURA
2 X3,Y1
4 X1,Y3
POLIGONAL 12341
$
$Vista frontal
$
8 1 @ 0,-1.3*%_ESCALA
6 8 @ %COMPRIM,-%ALTURA
5 X8,Y6
7 X6,Y8
POLIGONAL 56785
```

```
$
$Vista lateral
$
9 2 @ 2.5*%_ESCALA,0
11 9 @ %ALTURA,%LARGURA
10 X11,Y9
12 X9,Y11
POLIGONAL 91011129
$
$Nos auxiliares
$
13 5 @ -1*%_ESCALA,0
14 4 @ 0,1*%_ESCALA
15 (X1 + X3)/2,(Y1 + Y3)/2
$
$Cotagens e titulo
$
COTAGEM VERTICAL 5 8 13
COTAGEM VERTICAL 1 4 13
COTAGEM HORIZONTAL 4 3 14
COTAGEM HORIZONTAL 12 11 14
TEXTO 15 CENTRADO '%TITULO'
FIM
```

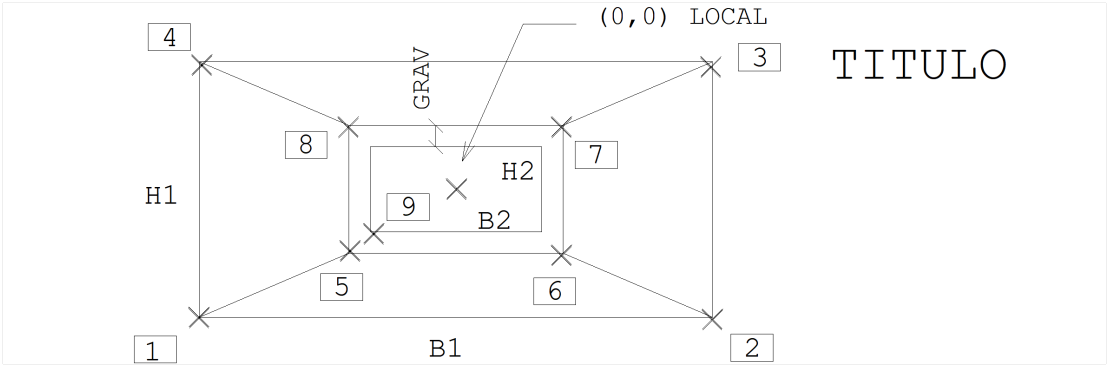
O comando FIM neste caso delimita o fim de um subprograma DPS, e não o fim do desenho.

Sistema de coordenadas DPS

O DP permite que se especifique coordenadas e ângulo de rotação na chamada do subprograma. Estes valores definem um ponto de inserção do desenho gerado pelo subprograma, em relação ao seu próprio sistema local.

Tudo se passa como se o desenho gerado pelo subprograma fosse um bloco de desenho com base (0,0), e a chamada do subprograma (comando DP) o comando de inserção de blocos. Se as coordenadas de inserção e ângulo de rotação não forem definidos, o sistema local do subprograma coincidirá com o sistema de coordenadas do programa principal.

Como exemplo, faremos o desenho de uma sapata para ser inserida em um desenho de formas em qualquer escala e ângulo. O ponto de inserção da sapata será o seu centro, que obrigatoriamente terá coordenada (0,0). A figura abaixo mostra o esquema da sapata:



Os parâmetros que definem a sapata são: B1/H1, as dimensões externas, B2/H2 as dimensões do pilar sobre a sapata, e GRAV a largura da gravata. A sapata receberá também um título, colocado no lado direito superior. Adotaremos o default para a gravata e permitiremos a definição da escala como parâmetro; o subprograma SAPFOR.DPS será:

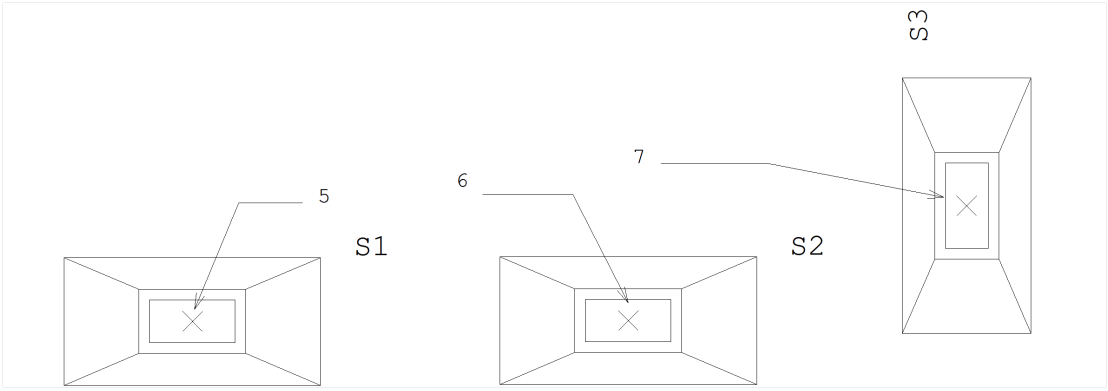
```
$
$SAPFOR.DPSForma de concreto para sapatas
$
PARAMETROS
ATITULOCOM 'Titulo'
NB1COM 'Comprimento da Sapata (cm)'
NH1COM 'Largura da Sapata (cm)'
NB2COM 'Comprimento do Pilar (cm)'
NH2COM 'Largura do Pilar (cm)'
NGRAV = 5COM 'Gravata (cm)'
NESCALA = 50COM 'Escala 1:x'
FIM

DEFINE ESCALA %ESCALA
1-%B1/2,-%H1/2
21 @%B1,0
32 @0,%H1
43 @-%B1,0
POL12341
5-((%B2/2) + %GRAV),-((%H2/2) + %GRAV)
```

```
65 @%B2 + (2*%GRAV),0
76 @0,%H2 + (2*%GRAV)
87 @- (%B2 + (2*%GRAV)),0
POL56785
POL15
POL26
POL37
POL48
95 @%GRAV,%GRAV
POL9 REL %B2,0 @ 0,%H2 @ -%B2,0 @ 0,-%H2
$
$Titulo a direita do no' 3
$
TEXTO3 @0.5*%ESCALA,0 HTEX 0.4 '%TITULO'
FIM
```

Vamos agora inserir uma sapata 120 x 60 com pilar de 40 x 20 sobre os nós 5 e 6 da forma com ângulo de zero graus e no nó 7 com ângulo de 90 (arquivo EXEMPLO5.DP):

```
DP'SAPFOR' 5(A TITULO ='S1',
N B1 = 120,N H1 = 60,
N B2 = 40,N H2 = 20)
DP'SAPFOR' 6(A TITULO ='S2',
N B1 = 120,N H1 = 60,
N B2 = 40,N H2 = 20)
DP'SAPFOR' 7 ANG 90(A TITULO ='S3',
N B1 = 120,N H1 = 60,
N B2 = 40,N H2 = 20)
```

O resultado será:



Uma observação interessante é que os nós 5, 6 e 7 definidos na forma, não têm nenhuma relação com os usados internamente no subprograma DPS. Como dissemos antes, os subprogramas tem as suas próprias variáveis locais, que não interferem com as de outros programas.

Subprogramas podem funcionar como se fossem blocos de desenho, com a diferença de que blocos podem ser apenas escalados, enquanto que subprogramas podem desenhar peças em quaisquer proporções, títulos variáveis, etc.

Subprogramas podem também efetivamente gerar blocos de desenho. Você pode definir a seção de blocos antes da chamada do subprograma ou mesmo dentro do próprio.

ORIGEM de um Subprograma

Todos os elementos gráficos de um subprograma são colocados em um sistema local, que pode ser transladado durante a chamada do subprograma. Na prática, o que ocorre é que a origem do sistema local de coordenadas é alterada durante a chamada de um subprograma; esta origem é a mesma definida pelo comando ORIGEM.

Por isto o comando ORIGEM não deve ser usado dentro de um subprograma para a mudança de sistema local; se necessário, pode-se chamar outro subprograma mudando o sistema durante a chamada.

Por outro lado, todas as coordenadas dentro de um subprograma estão dentro do sistema local. Havendo necessidade de transportar estas variáveis para fora do subprograma (por meio de variáveis globais), para passa-las ao sistema global é necessário antes "desligar" o sistema local, através do comando origem:

```
ORIGEM 0,0 ANG 0
```

Bibliotecas de subprogramas

O que dissemos anteriormente a respeito de bibliotecas de inclusão vale também para bibliotecas de subprogramas. Os subprogramas DPS devem ser isolados em um ou mais pastas (bibliotecas) diferentes, e a definição das pastas é feita através do comando:

```
DEFINE BIBDP 'pasta(s)'
```

Por exemplo, para tornar permanente a definição da biblioteca %_SUPORTE\DP\DPS colocaremos no arquivo %_SUPORTE\DP\INSTAL.DP:

```
DEFINE BIBDP'%_SUPORTE\DP\DPS'
```

Quando um subprograma é chamado, o DP verifica se está na pasta atual, e depois, em cada uma das pastas definidos em BIBDP.

Listagem do arquivo DPS

Por default, o DP não emite listagens no processamento de subprogramas; apenas erros de sintaxe quando encontrados são mostrados. No entanto, durante a fase de desenvolvimento de um subprograma pode ser interessante que seja listado para facilitar a eliminação de erros. Neste caso, o comando:

```
DEFINE LISTA
```

deve ser colocado no inicio do subprograma a listar. O comando

```
DEFINE NLISTA
```

tem efeito contrário, suprimido as listagens a partir do ponto onde for executado.