Escrever um programa paramétrico ou um programa macro:
duas coisas semelhantes, mas tão longe!
A diferença substancial entre os dois programas é encontrada na definição das variáveis. O programa paramétrico define os valores das variáveis que irá usar, o programa macro define uma estrutura lógica de variáveis, que terão um valor apenas quando a macro será usada (chamada).

Este é um exemplo de programa paramétrico

```
O1000(PARA)
------
#100=1
#101=23
WHILE[#100GE#101]DO1
----
executar algo
----
#100=#100+1
END1
----
```

Notamos que as variáveis são definidas com um determinado valor.

Este é um exemplo de programa macro

```
O9010(MACRO)
(CHAMADA G190 X Y Z Q K D F A)
(ORIGEM DE USINAGEM X+ Y-)
#122=ROUND[#7] (PARTE FIXA)
IF[#7EQ#122]THENGOTO5
#123=#7-#122(RAIO)
#123=#123*10(FATOR MOLTIPL.)
#7=#7-[[#123*2]+1](FRESA REAL)
N5#100=FUP[#25/#7](NUMERO PASS)
#101=FIX[#25/#100](PASS)
#102=#101-[#7/2](1° Y)
#105=[#7/2]+5(SEGURANÇA X)
#106=#24+#105(POSICIONAMENTO X)
#26=#26+[#17*#6](Z+SOVRAM)
IF[#1EQ1]THEN#26=#26+0.1(ACABAMENTO)
----
(REGRAS PARA A ROTAÇÃO TRIGONOMÉTRICAS)
(X'=X*COS(A)-Y*SIN(A))
(Y'=X*SIN(A)+Y*COS(A))
#120=#106(X)
#121=0(Y)
#120=#120*COS[#4]-#121*SIN[#4](XROT)
#121=#120*SIN[#4]+Y*COS[#4](YROT)
----
```

O programa usa certas variáveis, nota-se que estas não tenham sido inicializadas. Na verdade, a macro precisa se preocupar em fazer o trabalho para o qual foi criada; os valores das variáveis serão recebidos a partir do exterior através da chamada.

Podemos portanto dizer que a diferença entre os dois programas é relativo ao controle das variáveis em uso e sua definição. Dito isto, construções lógicas ou matemáticas são as mesmas, bem como as funções ou operações.

Aqui vamos falar sobre programas macro, mas é claro que o argumento será referido também aos programas paramétricos.

Preparando-se para escrever um programa, você deve sempre perguntar-se se é melhor criar uma macro ou um programa paramétrico.

Em geral, podemos dizer que macros são escritas para ajudar os operadores CNC para realizar suas tarefas melhor, para ser mais produtivo, mais rápido, tornando o programa menos tedioso e menos propenso a erros. A macro é escrita, se houver uma repetitividade ou para automatizar certas operações. Por outro lado, se estamos na presença de um único problema, que talvez nunca irá acontecer, é preferível realizar um programa paramétrico, para resolver o problema, o que é a utilização de variáveis no programa em código G. Tais programas são mais simples de implementar, mais rápido para realizar, pode ser repetitivo, mas oferecem a solução para problemas específicos (devido ao fato de que as variáveis sofrem uma inicialização anterior).

Às vezes é melhor criar programas paramétricos para acelerar a solução do problema que depois serão transformados em macro!

Se você teve a chance de ler os programas macro escritos por pessoas diferentes, você percebeu que existem diferentes maneiras de escrever.
Na verdade, os manuais Fanuc não falam sobre estruturação de um programa, mas fornecem uma lista de operações.
Então cada um é livre de estruturar o programa como acredita.
Eu acho que todos nós devemos usar uma estrutura lógica comum e universal, então, vou propor um método que na minha opinião e experiência atende esse requisito.

Deveria estar claro que um programa macro é um programa que pode aceitar certos argumentos (valores associados às variáveis locais), e realiza uma função específica relacionadas com as possíveis operações realizadas com uma CNC. Você pode escrever programas em G-código , programas para ajudar o operador (IMPUT-OUT FILE) e tudo o que você quiser.

Como escrever uma macro?
A criação de um programa macro é dividido em fases distintas:

- Problema ->> macro?
- Escritura macro
- Experimentação e mudanças
- Implementação

Problema: Macro?
Realiza-se uma macro somente após uma consideração cuidadosa de todas as alternativas possíveis para a solução do nosso problema. O programa macro deve nos ajudar e não complicar nossa vida, deve ser simples, fácil de usar e eficaz na sua tarefa. Você vai escrever uma macro se existe uma certa repetitividade ou automação (porque escrever um tal programa ao usar apenas uma vez?), pois a escrita destes programas leva tempo e os recursos não são indiferentes.

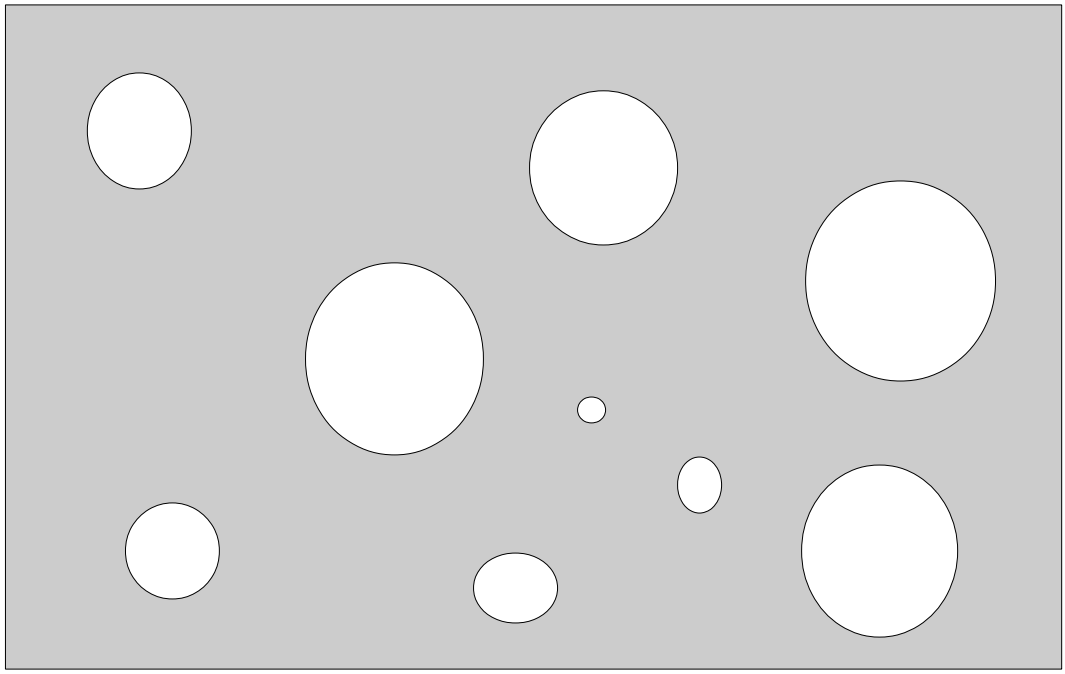
Escrever macros.
E' a parte que iremos discutir hoje: a tradução do problema em código paramétrico. Para escrever uma macro devemos ser bons programadores CNC, ter conhecimentos gerais de matemática e computação também. Mas não se preocupe muito sobre estes últimos aspectos: se você não os possui pode sempre obter ajuda de pessoas competentes. Sua tarefa é encontrar uma solução lógica, (eu quero fazer assim!), como implementá-la pode ser sugerido (você tem que fazer assim!).

Experimentação e mudanças.
Depois de escrever você deve experimentar. Geralmente uma macro não funciona na primeira vez, devemos corrigi-la, experimentar novamente, re-ajustar, testar novamente ... enquanto a macro pode ser considerada correta.

Implementação.
Quando a macro estiver concluída você terá um programa para colocar no CNC pronto para ser usado. A macro será executada em todas as CNC para qual foi escrita. Se este for o seu trabalho, talvez pode conseguir um dinheiro.

Para aprender a escrever macro é melhor começar a partir de problemas simples e bem conhecidos. A melhor maneira de entender como realiz-los é através de exemplos estruturados e compreensíveis.

Nós programadores estamos familiarizados com o corte interno de um círculo: a nossa macro terá como base este conceito.



Problema:
temos que realizar placas perfuradas, como mostrado em figura. As cavidades circulares têm sempre diferentes medidas e estão posicionadas em lugares diferentes com profundidade variadas. A cada duas ou três semanas nós temos que fazer várias destas placas.

Escrever o código G para um processo semelhante é simples. Há porém alguns aspectos que levam à criação de uma macro que pode fazer o corte interno do círculo, isto é, criar cavidades de tamanhos diferentes.

- A escrita é repetitiva, pois trata-se de escrever o mesmo código para cada círculo.
- A possibilidade de erro é alta, dada a repetitividade.
- É muito chato escrever a mesma operação o tempo todo.
- Embora a escrita seja simples, implica um certo tempo que poderia ser usado melhor.

Bem, nós percebemos que estamos enfrentando a possibilidade de criar uma macro, pois há critérios principais: repetibilidade e automação.

Por isto, vamos construir um programa capaz de realizar este trabalho de jeito automático, rápido e seguro.

(Em geral, refiro-me a uma fresadora (CNC). O conceito é válido mesmo para torno).

Este é o esquema que iremos utilizar para dar uma boa estrutura aos nossos programas macro.

- Comentários
- Controle de variáveis
- Resgate de dados modais
- Operações matemáticas e lógicas

- Recuperação de dados modais

Este esquema servirá como verificação final da estrutura do nosso programa, e não representa a seqüência lógica de implementação (veja abaixo).

Tente ser claro sobre o que você quer começar, como você pode começar e quais os problemas que precisam ser respondidos. E' este o passo mais difícil: dar um rosto ao seu programa. Não se preocupe em definir todos os aspectos agora, você vai ver que durante a implementação vai mudar de idéia com freqüência.

Reflexões!

* Como realiza-se o corte interno de um círculo?
A resposta é a solução para nosso problema! G03 I/J (em concordância)

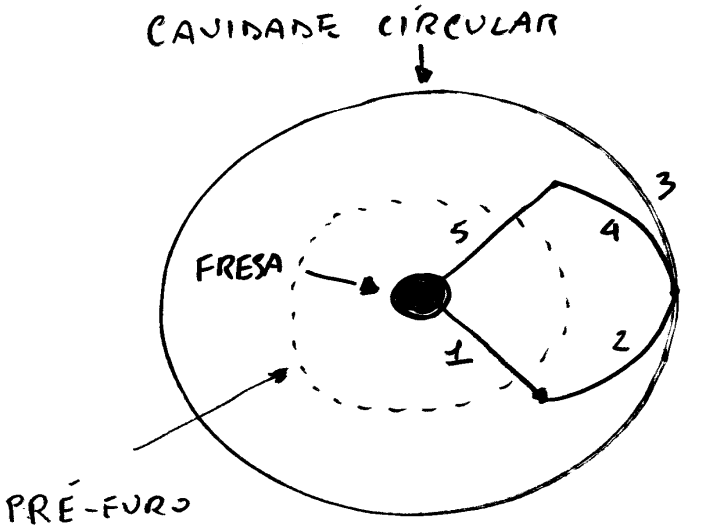
* Considerações!

- Considerar pré-furação?
- Cavidade cega ou passante?
- Utilizar compensação raio ou não?
- Controle diâmetros (ferramenta, cavidade, furo de entrada)
- Verificação passadas horizontais?
- Mais ..

Esses poucos pensamentos antes de escrever a macro para ter ideias claras.

Então, nossa idéia é a seguinte:
escrever um programa capaz de fazer o corte interno de um círculo, considerando uma ferramenta qualquer, um furo de entrada (não realizado pela macro). Vamos trabalhar com passadas horizontais e verticais, e vamos fazer uso de compensação raio.

Um programa macro não inventa nada de novo, mas cria uma estrutura diferente para se adaptar ás muitas necessidades. A figura é o ponto de partida da nossa macro



1 APROXIMAÇÃO
2 ATAQUE CIRCULAR
3 CORTE INTERNO CÍRCULO
4 SAÍDA CIRCULAR
5 RETORNO

Então começamos!

A macro deve sempre começar com comentários que explicam o uso.
Os comentários não servem para aqueles que utilizam a macro, mas para você, para lembrar o que escreveu .... depois de vários meses é normal não se lembrar das coisas escritas .... comentários ajudará a lembrar.

Vou usar a notação G (macro modal): a macro vai ser chamada com o código G200 (o código será associado com um parâmetro para o número do programa que você escolher O9010->6050, O9011->6051...).

A definição dos argumentos depende das escolhas objetivas (valores e dados que descrevem a realização da tarefa) e subjetivas (medidas para facilitar certas operações ou a introdução de funcionalidades adicionais).
As primeirass são essenciais as últimas são complementares.

Lembre-se que você pode implementar mudanças para o seu programa! Então não se preocupe se você esquecer alguma coisa ou se você deseja alterar sua lista.

Emergiu das considerações anteriormente mencionadas um primeiro esboço dos quais argumentos irá usar a macro:

Z Profundidade cavidade
D Diâmetro cavidade
B Diâmetro broca (pré-furação)
O Diâmetro fresa
R Início cavidade em Z
Q Remoção
F Avanço
A Cavidade cega/passante

Explicamos a escolha!

Uma cavidade circular é definida por: profundidade (início e fim), diâmetro, pode ser cego ou passante.
O processamento de uma cavidade é feito com certas remoções, com ferramentas escolhida, um valor justo de F.
Além disso, consideramos um buraco (não realizado pela macro) para a entrada da ferramenta.

Um olhar sobre a escolha dos argumentos.
Z, R, F, Q têm o mesmo significado de um programa NC, assim você pode facilmente lembrar (é por isso que a lista dos argumentos macro possui esta ordem).
I, D, A, B foram escolhidos livremente.

Em seguida, a macro será chamada assim

G200 Z Q F R I D A B

sabemos que há uma correspondência entre argumento e variável, apresentamos novamente

#26=Z #18=R #9=F #17=Q #4=I #7=D #1=A #2=B

Nós definimos um programa macro que aceita determinados argumentos.
Quando iremos escrever G200 ... em um programa G-código é como escrever:

"Faça o corte interno di uma cavidade circular de profundidade Z, de diâmetro D, utilizando uma fresa I, que entra em um furo B, a altura inicial vai ser R.
O bolso pode ser cego ou passante, e o progresso será realizado com incrementos de corte Q com velocidade F seja em horizontal que em vertical."

Parece um bom começo!

(Para não perder o fio da conversa, ocasionalmente mostraremos os progressos alcançados com a escrita).

```
O9010 (CORTE INTERNO CIRCULO)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  AVANCO)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
```

Uma macro é um programa que aceita argumentos, então a primeira coisa a fazer é controlá-los. Verificar os argumentos significa:

- verificar a existência
- verificar o valor

** Verificar a existência*

Pode acontecer de esquecer alguns argumentos na chamada de uma macro

```
G200 Z R (F) A B I G Q
```

no exemplo esquecemos o argumento F (velocidade):
O que acontece quando você executa a macro?

Lembre-se que esquecer um argumento quer dizer não inicializar uma variável, isto é, deixá-la sem valor (assim são todas as variáveis antes de entrar na macro, isto é, antes de serem inicialisadas).

Não podemos fazer suposições genéricas, "o CNC irá usar o último F usado", pois depende de como a macro foi escrita. O comportamento de um programa macro segue as regras já discutidas; uma operação pode ser executada sem problemas aparentes, outra não, ou será executada de forma totalmente anormal.
Devemos sempre verificar os argumentos e implementar medidas adequadas.
Para o controle usa-se a #0.
#0 é uma variável nula, não é 0, é não inicializada. Ela pode ser lida, mas não aceita valor.
Se, em seguida, comparamos nossos argumentos com esta variável saberemos imediatamente se foi inicializada, quer dizer se o argumento foi ou não esquecido.

```
exemplo     IF[#26EQ#0]THEN #3000=1(FALTA Z)
```

Este é o exemplo clássico de controle presença de uma variável.
Se #26 (Z) é igual a #0 (não inicializada), então será gerado o alarme seguinte:

```
alarme macro 3001 "FALTA Z".
```

o CNC vai parar permitindo ao operador corrigir o erro.
O raciocínio será realizado para todas as variáveis!

Agora devemos dizer que nem todas as variáveis devem estar obrigatoriamente presentes.

Às vezes, para simplificar o trabalho do programador (e o nosso) podemos tomar uns valores padrão, ou seja, valores específicos para usar quando eles não estão claramente especificados em uma chamada. Por exemplo, podemos assumir por defeito que as cavidades sejam passantes, definindo o argumento A (cavidade cego/passante) apenas no caso cego. Isso facilitaria a tarefa do operador que deve lembrar um menor número de argumentos.

A macro pode ser chamada sem esse argumento

G200 Z R F Q I D B (A = default, padrão)

Também podemos usar como furo de entrada um valor padrão de 50% do diâmetro da cavidade, obliterando o parâmetro (B)

G200 Z R F Q I D (A e B default)

O valor padrão quando bem utilizado pode ajudar o operador de uma forma surpreendente.
Mas tenha muito cuidado, se você abusar corre o risco de causar mais danos do que boas ações ... avaliar com muito cuidado quais argumentos para uso padrão!

Estas variáveis serão, naturalmente, controladas de formas diferentes.
Possuindo valores padrão deveremos verificar a existência e atribuir valores específicos.

IF[#1EQ#0]THEN #3000=10(FALTA A)

A seguinte redação é portanto errada, para as escolhas feitas!

```
(#1=0 CAVIDADE CEGA)
(#1=1 CAVIDADE PASSANTE)
IF[#1EQ#0]THEN #1=1
```

Os comentários dizem-nos que o valor de #1 pode ser 0 ou 1.
0 em caso de CAVIDADE cega (ajustado pelo operador), 1 cavidade passante definido pelo programa como padrão.

A linha IF[#1EQ#0]THEN #1=1 significa que, se #1 não foi definida (que não está listada na chamada macro), então seu valor será definido como 1, indicando uma cavidade passante.

Aqui também temos de fazer um discurso sobre a escolha de valores a serem atribuídos às variáveis padrão. Em eletrônica e informática usa-se o sistema binário. Esta maneira de pensar e contar usa a notação 0 1 (binário) ou se preferir ON OFF. São, então, associadas as letras aos números da seguinte forma 0=OFF, 1=ON. Você está livre para usar os valores que quiser.

```
(#1=1887 CAVIDADE CEGA)
(#1=7384 CAVIDADE PASSANTE)
IF[#1EQ#0]THEN #1=7384
```

Como sempre lembre-se de ser simples: números como os escritos acima não são facilmente lembrados (o operador terá que escrever 1887 para realizar uma cavidade cega).

É preciso abrir uma parêntese sobre as diferenças entre os programas macro e programas paramétricos gerais.
As variáveis utilizadas pelos programas macro são locais, são variáveis inicializadas no momento da sua utilização. Essas variáveis são nulas, não têm nenhum valor, exceto no momento apropriado e voltam a ser nulas depois do encerramento da macro.
Em um programa NC usa-se normalmente variáveis comuns. Estas, dependendo do número podem ter um valor (500-999), ou ser nulas (100-199).
As últimas uma vez inicializadas não perdem o valor até o desligamento.

Esta diferença deve ser considerada pourque atribuir um valor padrão como 0 ou 1 a uma variável comum pode não ser uma idéia positiva.
Se assumirmos que uma variável tem o valor 0 (não imposta por nós, mas já presente) e nós tomá-lo como padrão, isso não garante o bom funcionamento do programa: quem definiu o valor 0 para a variável?
É claro que uma situação como esta deve ser evitada. Programas NC paramétricos devem sempre definir o valor de uma variável e nunca fazer suposições falsas!

Retomamos nossa discussão sobre a presença de variáveis, avaliar o comportamento de cada uma para definir as ações apropriadas.

Z, a profundidade da cavidade depende do projeto, é um argumento que não pode ser considerado um padrão, então

```
IF[#26EQ#0]THEN #3000=1(FALTA Z)
```

R, o início da cavidade (em Z) pode assumir valores arbitrários, é um argumento que não pode ser considerado um padrão, então

```
IF[#18EQ#0]THEN #3000=2(FALTA R)
```

F, no que se refere à velocidade pode-se argumentar que pode ser constante (pois temos que trabalhar o mesmo material).
Sendo esta uma macro para uso geral, portanto, aplicável a qualquer material, consideramos F não como objecto de padrão

```
IF[#9EQ#0]THEN #3000=3(FALTA F)
```

Q, a remoção é uma escolha que cabe ao operador/programador, é um argumento que não pode ser considerado um padrão, então

```
IF[#17EQ#0]THEN #3000=4(FALTA Q)
```

Mesmo para este argumento se poderia implementar um valor padrão.
Você pode definir como padrão por exemplo 2mm e quando for necessário introduzir um novo valor. Você é livre para agir como você acredita, mas cuidadosamente analisadas as vantagens e desvantagens.

I, o diâmetro da fresa é decidido pelo operador, é um argumento que não pode ser considerado um padrão, então

```
IF[#7EQ#0]THEN #3000=5(FALTA I)
```

D, o diâmetro da cavidade depende do projeto, é um argumento que não pode ser considerado um padrão, então

```
IF[#7EQ#0]THEN #3000=6(FALTA D)
```

B, o diâmetro orifício de entrada é uma escolha do operador, é um argumento que não pode ser considerado um padrão, então

```
IF[#2EQ#0]THEN #3000=7(FALTA B)
```

A, cavidade cega ou passante. Para ajudar o operador vamos assumir um valor padrão (1=passante). O operador deve digitar 0 se quer fazer uma cavidade cega.

```
IF[#1EQ#0]THEN #1=1
```

O comentário relativo a #3000 pode chegar a 26 caracteres, incluindo espaços, para que você possa adicionar comentários como mais explicativa

```
IF[#26EQ#0]THEN #3000=1(INSERIR Z CAVIDADE)
```

Isto é válido sobretudo para os argumentos gerais que usam letras com

significado diferente de um programa NC

IF[#7EQ#0]THEN #3000=3(FALTA DIAMETRO FRESA)

Algumas pessoas preferem estruturar de forma diferente o tratamento de alarmes, colocando-os no final do programa. Neste caso, você deve usar um GOTO para realizar a operação desejada

```
IF[#26EQ#0]THEN GOTO 9500
----
programa
----
```

no final do programa teremos
N9500 #3000=1(FALTA Z)

Quem usa esse tipo de escrita não quer colocar no início do programa os comentários. Usar um método ou outro é contudo sua escolha.

Vamos recapitular

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
```

** VerifIcar o valor*

Definida a presença dos argumentos vamos prosseguir com a verificação dos valores que eles podem tomar. Essa verificação é importante para não se encontrar blocos inesperados do programa. A avaliação abrange dois aspectos:

- valores absolutos
- dependência de outras variáveis.

Temos que verificar que os valores associados às variáveis têm um sentido físico e matemático.

Z pode assumir qualquer valor? Claro, não sabemos onde está a origem de usinagem, então Z pode ser negativo, 0 ou positivo. É claro que, se escrevermos Z5000 (mm), certamente teremos cometido um erro, mas não pudendo limitar o valor

de Z, é para o operador ser cuidadoso.

(E' possivel criar programas de controle erros).

R, pela mesma razão, pode ter qualquer valor. Além disso, o valor de R não pode ser inferior ao de Z. Portanto, há uma relação entre as duas variáveis (a ordem de controle das variáveis é muito importante, vamos falar sobre isto mais para frente).

IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)

Se o valor de R (#18) é menor ou igual a Z (#26) há um problema de incompatibilidade de valores: como pode ser R menor ou igual a Z?
Um deles está errado!

F, o valor de F pode ser limitado ao valor máximo de F máquina.
Isso seria um impedimento se a macro foi publicada as CNC mais rápido, por isso, o único limite que colocamos é que deve ser positivo.

IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)

Q, o valor Q indica a remoção. Sendo uma quantidade física o seu valor é sempre positivo. O zero não é aceitável!
Devemos decidir se aceitar valores negativos ou não! O discurso não é simples de resolver! Suponha de transformar os valores negativos (Q-3) de uma forma positiva. Agora, o valor de Q-3 é uma declaração do representante ou um erro? Quero dizer, se você escrever Q-3, é isto mesmo que queria escrever ou quis escrever por exemplo Q13? Foi erro ou não?.
Estamos em uma situação anômala, onde o sinal (-) não garante a clareza da redação! E' talvez aconselhável, por isso, causar um alarme ... você vai ver que depois de várias tentativas, o operador vai entender como inicializar a variável. (A regra básica é: não fazer suposições!)

IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)

Se o valor de Q (#17) é negativo ou 0, um alarme será exibido.

I, diâmetro da fresa. A fresa é naturalmente um objeto concreto, por isso não pode ser negativo ou zero

IF[#4LE0]THEN #3000=9(VALOR I ERRADO)

Na realidade as coisas não são tão simples, porque você pode trabalhar com qualquer valor do raio da ferramenta, zero positivo ou negativo. Não quero aprofundar o assunto aqui, eu acho seja a escolha melhor para operadores/programadores neste contexto.

Um aspecto a considerar é a relação entre a ferramenta, o diâmetro final da cavidade, e também a relação de entre diâmetro da ferramenta e diâmetro de perfuração.
A fresa não pode ser superior aos dois: não iremos trabalhar. Deve também ser ligeiramente menor, a fim de usar correção de raio ferramenta.

Neste discurso queremos marcar um aspecto importante sobre o controle das variáveis, ou seja, a ordem de inspeção.
Neste caso, você deve primeiro verificar o valor do diâmetro da cavidade, então o valor do diâmetro do buraco e, finalmente o da fresa , relacionando este último com os outros.
Comparar o valor diâmetro da fresa com o furo sem primeiro ter verificado este último pode levar a erros.

Vamos supor o diâmetro do furo de entrada de 30mm, o da cavidade 20mm; valores ainda não controlados.
Suponha que você use uma fresa de 25mm. Esta é menor que o diâmetro perfurado, o

que torna possível ao processo, mas, pelo contrário, o mesmo é maior do diâmetro da cavidade.
Estamos em uma condição anormal que deve ser primeiramente analisada.
E' preciso efectuar os controlos na ordem mais apropriada:

D, diâmetro da cavidade depende do projeto, não é escolhido por nós, pode assumir qualquer valor diferente do 0 ou negativo

```
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
```

B, furo de entrada, que é o mesmo discurso com a atenção que este último deve ser menor que o diâmetro da cavidade

```
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)

IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
```

Se o valor do furo de entrada (#2) é maior ou igual ao diâmetro da cavidade vai ser gerado um alarme.

Agora você pode verificar o valor da fresa em relação ao furo.

```
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)

IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
```

Você está se perguntando por que analisar assim cuidadosamente o relacionamento com o furo de entrada?
Esta necessidade surge para avaliar não só a possibilidade de entrada de uma ferramenta, mas também o cálculo do passe horizontal que a fresa tem que fazer em caso de alta relação entre o diâmetro da cavidade e o diâmetro da ferramenta.

Se criarmos uma cavidade de 200 mm de diâmetro com uma ferramenta de 20mm e um diâmetro de 60mm de furo, é claro que você não pode realizar uma única passada horizontal, mas a ferramenta irá executar os movimentos no chão, antes de aumentos em Z. A macro então terá que permitir essa possibilidade, além de um loop vertical também um horizontal. Isto nos permite usar uma ferramenta não-ideal (relação diâmetro-cavidade/diâmetro-fresa elevado). Não, é um discurso acadêmico, mas um requisito de negócio real. Quantas vezes não temos a ferramenta certa e precisa usar uma subdimensionada? A macro leva isso em conta, permitindo que você escolha qualquer ferramenta e automaticamente calcula qualquer passada horizontal para cobrir toda a superfície.

Feitas as verificações necessárias, podemos prosseguir com a escrita.

Resumo.

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
```

```
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
```

O que temos feito até agora?
Nós escolhemos um método para a chamada (G), definimos um programa (O9010), comentamos sobre o uso, escolhemos os argumentos definindo o campo de ação.

E' importante fazer as verificações necessárias para não encontrar erros durante a execução do programa que talvez você não sabe de onde são gerados.
Isso não significa evitar qualquer possível erro, o que pode acontecer de qualquer maneira, mas por-se em condição de deixar o pé direito. Em caso de erros sabemos que não dependem de nós, mas por factores externos: iremos analisa-los e defini-los!

** Salvar os dados modais*

O controle dos argumentos é uma grande ajuda, mesmo que às vezes chato, mas só isso não é suficiente para garantir a aplicação adequada. Há um aspecto que poucas pessoas pensam, não avaliados: o estado da máquina.
Os dados modais indicam o estado da máquina, quer dizer as funções modais ativas, funções que definem o comportamento global.

G17, G40, G00, G54, G80 ...

Como a macro se relaciona com os estados modais? Ela muda o comportamento?
A macro é uma sub-rotina e funciona como todos os programas. O problema é que sendo um programa particular, se não for realiza-da com perfeição poderia causar caos no operador.
Suponha que você criou uma macro que irá alterar o sistema de referência, por exemplo, do sistema atual ao G54.1 P200
Um operador que está se preparando para usar a macro não espera mudanças como essa: por que a macro vai mudar a referência? Por que não deixa o meu sistema ativo? Você compreende que é uma situação anômala: a macro não deve alterar o estado modal da entrada, quando acontecer (o que é possível) deve primeiro salvar o estado e depois restaurá-lo na saída.
(É claro que há casos em que o estado modal pode ser mudado, não o consideramos aqui).

O conceito é o seguinte:

- salvar o estado modal
- realizar todas as operações necessárias
- restaurar o estado modal

Desta forma, o operador estará sempre na condição prévia á chamada.
Bem, mas o que devemos salvar?

Os dados modais se encontram nas variáveis do sistema de #4001 a #4130.
Dependendo do nosso programa, vamos salvar os dados que acreditamos possam mudar.
O que a nossa macro faz? Executar repetições.
Inútil salvar o plano (G17) ou uma função como S, M, não é essencial. Por que

salvar os códigos como G20 ou G21 se o programa não intervém nestes?

Decidimos salvar os seguintes dados

#4001, estado G00, G01, G02..
#4003, estado G90, G91
#5003, valor atual de Z na coordenada peça ativa

Vamos salvar a #4001, pois usinando, certamente mudaremos o estado. O objetivo deste resgate é preventivo porque a macro poderia ser chamada em sucessão.

Salvamos #4003 pela mesma razão.

O valor de #5003 será usado como valor de retorno para Z. Não sabendo ainda como a nossa macro vai acabar, este valor vai nos dar a certeza de um posicionamento correto. (partir e voltar para o valor de Z100 nos fornece um valor válido).

Salvar dados modais significa colocar esses dados em variáveis específicas escolhidas por você, possivelmente para não usar, assim que você pode restaurar.

```
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
```

Nosso programa de controle das variáveis, presença, valor e dados modais, foi escrito. Agora vamos nos concentrar em escrever o código para a usinagem.

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
```

E' extremamente importante comentar as operações realizadas; não se preocupe com a duração do programa. Uma vez definida a macro, iremos por no CNC o programa sem comentários.

É um grande trabalho certo? Mas depois da primeira macro você verá que tudo tornará mais fácil!

Prosseguimos então!

Você deve ter uma vaga idéia (melhor se já definida) qual será a sua macro.

A fresa entrando em un furo vai cortar o interno da circunferencia, fazendo passadas de lado se necessário, tudo vai ser repetido em Z até alcançar o valor desejado. Devemos considerar que o bolso pode ser cego ou passante, fazendo uso de compensação de raio com ataque e saída circulares.
No final da usinagem iremos restaurar os dados guardados modais.

Enquanto estava escrevendo atirei a seguinte observação. A usinagem de um perfil pode ser feita para fazer desbaste ou acabamento: por que não usar esse conceito em nosso programa? Podemos introduzir este elemento novo?
Precisa uma nova variável K, indicando a possibilidade de uma operação de desbaste ou de acabamento.
Vamos assumir o seguinte comportamento:

K=0 acabamento
0<K<=1 desbaste, com valor de default =0.2 (sobre-metal)

Em seguida, as alterações

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
(#6=K  K=0.2 DESBASTE/DEFAULT)
(#6=K  K=0 ACABAMENTO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#6EQ#0]THEN #6=0.2(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#6LT0]THEN #3000=13(VALOR K ERRADO)
IF[#6GT1]THEN #3000=16(VALOR K ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
```

```
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
```

O que estamos tentando mostrar não é um programa perfeito.
Nosso objetivo é mais educacional do que real,
então eu vou deixar para você a tarefa de avaliar as melhores opções práticas.

Por exemplo, podemos definir

```
K=0 acabamento
K=1 desbaste
```

obteríamos

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
(#6=K  K=1 DESBASTE/DEFAULT)
(#6=K  K=0 ACABAMENTO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#6EQ#0]THEN #6=1(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#6LT0]THEN #3000=13(VALOR K ERRADO)
IF[#6GT1]THEN #3000=16(VALOR K ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
```

É claro que vamos agir em conformidade, alterando as características do programa.

É sempre possível alterar a estrutura de um programa, você vai perceber durante o julgamento ... mesmo que às vezes isso significa mudar nossos planos.

Podemos decidir se trabalhar com diâmetros ou raios, é sua escolha.
Eu decido para trabalhar com os raios e logo em seguida imposto as variáveis relativas e o que for necessário.

```
(CONVERTENDO VARIAVEIS)
#2=#2/2.0(RAIO FURO ENTRADA)
#10082=[#4/2.0+#6](RAIO FRESA = RAIO FRESA + SOBRE-METAL)
#7=#7/2.0(RAIO CAVIDADE)
#8=500.0(AVANCO EM Z)
```

A #8 é o avanço de cada passada vertical.
A #10082 é o raio do corretor número 82 que será usado para esta usinagem.
Qualquer ferramenta usará este corretor. É uma escolha pessoal e isto é mostrado para fins educacionais. Você pode escolher o offset atual e usa-lo como favorito.

Devemos agora determinar se a macro vai fazer movimentos horizontais.

Estes controles preventivos refletem a realização da macro, mas devem ser feitos antes do ciclo de usinagem para o correto funcionamento.

```
(ACABAMENTO/UMA PASSADA)
IF[#6EQ0]THEN#2=#7-2.0
```

Se fizermos uma operação de acabamento (a cavidade já foi desbastada) ou uma única passada, vamos definir um valor de furo ligeiramente inferior ao da cavidade, isso irá garantir que executaremos um acabamento pois não há material para passadas horizontal a ser removido.
Às vezes recorremos a truques como este para obter os resultados desejados!

```
(NUM PASSADAS HORIZ)
#10=FUP[[#7-#2]/[#4*2.0]]
```

A diferença entre o raio da cavidade e o raio do furo (#7-#2) nos dá o material a ser removido, se você dividir pelo diâmetro da fresa (#4*2) obtemos o número de passadas a serem feitas sobre o plano para cortar o círculo .
Em geral, este número não é um número inteiro, em seguida, aplicar a função FUP que aproxima por excesso para garantir o número certo de passadas inteiras.

```
(AUMENTO HORIZ.)
#11=[#7-#2]/#10
```

Agora, determinadas as passadas podemos calcular o aumento horizontal adequado dividindo o material que sobra (#7-#2) para o número de passadas encontradas.

Finalmente, nós incluímos um controle de avanço vertical
em relaçao ao tamanho fresa/furo

```
(CONTROLE FRESA/FURO DE ENTRADA)
IF[#4GE#2]THEN#8=#8/10
```

O ciclo de usinagem que vamos escrever é o seguinte:

```
WHILE[]DO1        (CICLO VERTICAL)
  WHILE[]DO2      (CICLO HORIZONTAL)
      ---
      ---
  END2            (FINE CICLO HORIZONTAL)
END1              (FINE CICLO VERTICAL)
```

São dois ciclos WHILE aninhados, iremos realizar a remoção em direção horizontal e vertical.

Vamos recapitular novamente.

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
(#6=K  K=0.2 DESBASTE/DEFAULT)
(#6=K  K=0 ACABAMENTO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#6EQ#0]THEN #6=0.2(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#6LT0]THEN #3000=13(VALOR K ERRADO)
IF[#6GT1]THEN #3000=16(VALOR K ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
(CONVERTENDO VARIAVEIS)
#2=#2/2.0(RAIO FURO ENTRADA)
#10082=[#4/2.0+#6](RAIO FRESA = RAIO FRESA + SOBRE-METAL)
#7=#7/2.0(RAIO CAVIDADE)
#8=500.0(AVANCO EM Z)
(ACABAMENTO/UMA PASSADA)
IF[#6EQ0]THEN#2=#7-2.0
(NUM PASSADAS HORIZ)
#10=FUP[[#7-#2]/[#4*2.0]]
(AUMENTO HORIZ.)
#11=[#7-#2]/#10
(CONTROLE FRESA/FURO DE ENTRADA)
IF[#4GE#2]THEN#8=#8/10
```

Nosso programa está tomando forma!

Nossa macro não usa argumentos que se referem aos eixos X e Y, o que significa que será aplicada no ponto da chamada: o operador deve primeiramente posicionar a ferramenta no centro da cavidade, em seguida, chamar a macro. Esta decisão foi tomada para não ter muitos argumentos. Se você acha que deve introduzi-los pode, G200 X Y Z R F Q I D A B K onde X e Y representam o centro da cavidade.

```
(ABORDAGEM)
G90G0Z[#18+5.0]
G01F1000Z#18
```

As linhas representam a abordagem em Z (estamos no centro do bolso): primeiro, a uma distância de 5 mm (G0) e, em seguida, na altura Z de #18 (G01).

Agora vamos escreveremos o ciclo de trabalho verdadeiro (o corte do círculo).

```
(CICLO VERTICAL #5003=#26)
WHILE[#5003NE#26]DO1
(CONTACTOR CICLO HORIZ.)
#13=1.0
(CONTROLE PROXIMA PROFUNDIDADE)
#14=#5003-#17
(CALCULO FALTA DE Z)
#15=ABS[#26-#5003]
(SE PASSANTE DESVIA)
IF[#1NE0]GOTO20
(SE NO FUNDO)
IF[#15EQ#17]THEN#8=#8/10
(ALEM DO FUNDO)
N20IF[#14LE#26]THEN#17=#15
G91G01F#8Z-#17
(CICLO HORIZ.)
WHILE[#13LE#10]DO2
(CALCULO PONTOS ENTRADA CIRCULAR)
(RAIO RELATIVO)
#7=#2+[#11*#13]
(RAGGIO DE SAIDA)
#20=[#7+#4]/2.0
#21=[#7-#20]
#22=#21*SIN[45]
#23=#20*SIN[45]
#19=#20-#23
#12=#22+#23
(EXECUTAR)
G01G41D32X[#12+#19]Y#22F#9
G03X-#19Y#23R#20
G03I-#12J-#12
G03X-#23Y#19R#20
G01G40X-#22Y-[#12+#19]F[#9*3]
(AUMENTO CONTACTOR HORIZ)
#13=#13+1.0
(FINAL CICLO HORIZ.)
END2
(FINAL CICLO VERTIC.)
END1
```

Em negrito você pode ver a estrutura apresentada antes do WHILE loop aninhado. Explicamos!

```
(CICLO VERTICAL #5003=#26)
WHILE[#5003NE#26]DO1
```

O ciclo vertical depende da profundidade da cavidade, acabará ao alcançar do valor final definido pela #26 (Z). Para conseguir isto, usamos a #5003 o que representa o valor atual Z da coordenada peça ativa.

WHILE[#5003NE#26]DO1 significa continuar até que o valor atual de Z é diferente do valor de Z definido por variavéis.

O problema mais comum ao executar repetições em Z é verificar que o número de remoções não exceda o valor desejado. Em suma, você precisa verificar para não ficar demasiado (especialmente nas cavidades cegas).

```
(CONTACTOR CICLO HORIZ.)
#13=1.0
```

Vamos definir uma variável (contactor horizontal) para a remoção no plano.
O valor 1 é o mais adequado pois precisa executar pelo menos uma passada.
O contator, então, irá crescendo.

Neste ponto, realizamos uma série de controles para definir:

- o próximo valor de Z, #14=#5003-#17
- a distância restante até o fim do processamento,, #15=ABS[#26-#5003]
- executar açoes relativamente á cavidade, cego/passante

```
* (SE PASSANTE DESVIA)
  IF[#1NE0]GOTO20

* (SE NO FUNDO)
  IF[#15EQ#17]THEN#8=20.0

* (SE ALEM DO FONDO)
  N20IF[#14LE#26]THEN#17=#15
```

Escreveremos algo como:

```
(CONTROLE PROXIMA PROFUNDIDADE)
#14=#5003-#17
(CALCULO, FALTA DE Z)
#15=ABS[#26-#5003]
(SE PASSANTE, DESVIA)
IF[#1NE0]GOTO20
(SE NO FUNDO)
IF[#15EQ#17]THEN#8=#8/10
(ALEM DO FUNDO)
N20IF[#14LE#26]THEN#17=#15
```

Usamos duas variáveis para determinar o valor de Z para a próxima profundidade (#14=#5003-#17) e o valor de Z que falta (#15=ABS[#26-#5003]) para chegar ao final do processamento.
No caso de uma cavidade passante precisa uma verificação entre as duas variáveis para determinar se passamos o valor final (N20 IF[#14LE#26]THEN#17=#15), no caso iremos definir automaticamente a próxima passada igual ao valor de #15 (o que falta).
No caso de cavidade cega, vamos derrubar a fresa com velocidade adequada antes da usinagem (o valor #8/10 é indicativo, pois se quer marcar o conceito).

Movemos nossa ferramenta então

```
G91G01F # 8z-# 17
(A descida é feita em incremental, mas você pode trabalhar em absoluto)
```

Estamos em um bom ponto!

Ter descido, tornando estas verificações é ter descrito o ciclo vertical, que deve ser repetido.

Agora vamos descrever o loop horizontal que vai usar as #10 e #11 anteriormente encontradas

#10, número de passadas horizontal
#11, remoção horizontal

Decidi considerar o uso do corector de raio e fazer a entrada e saida em interpolação circular .... então vamos calcular tudo.

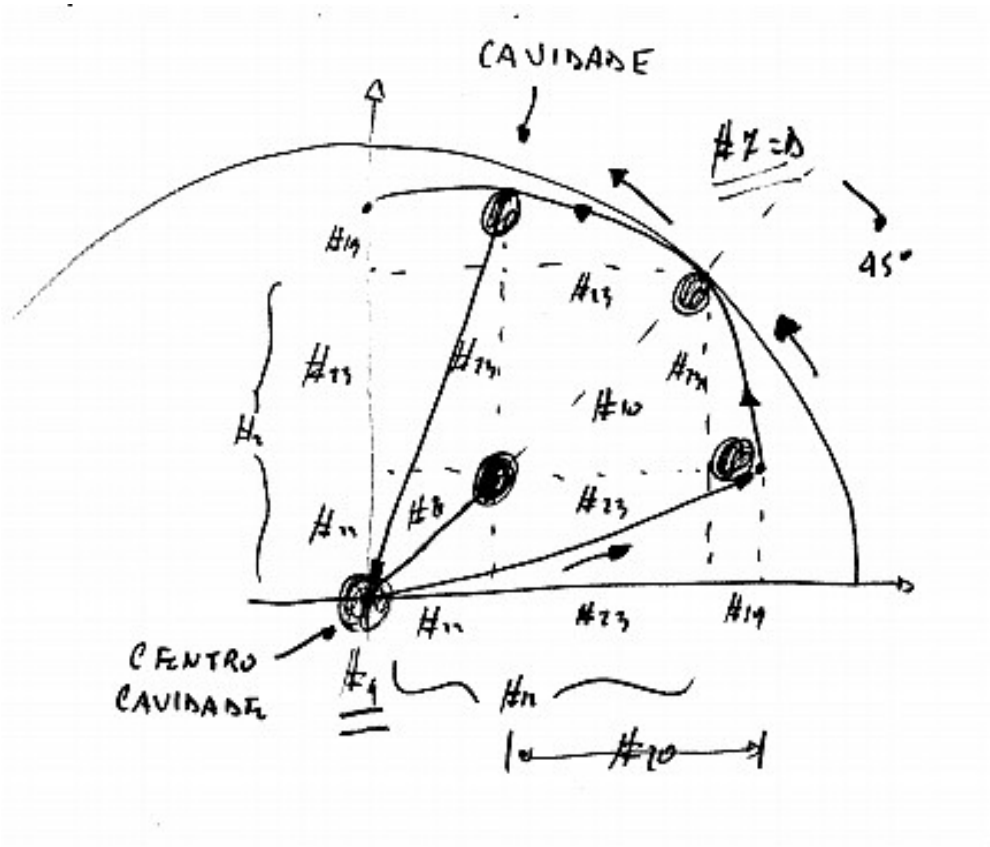
```
(CICLO HORIZ)
WHILE[#13LE#10]DO2
```

o ciclo será executado até que #13 será inferior ou igual a #10, isto é até o número calculado de passadas.

Para cada execução do ciclo será calculado o relativo raio de ataque e escape (ver figura)

```
(CALCULO PONTOS ATAQUE E DESTAQUE CIRCULAR)
(RAIO RELATIVO)
#7=#2+[#11*#13]
(RAIO DE ESACPE)
#20=[#7+#4]/2.0
#21=[#7-#20]
#22=#21*SIN[45]
#23=#20*SIN[45]
#19=#20-#23
#12=#22+#23
```

Para isso, use trigonometria.
É claro que você pode escolher qualquer ângulo.



Nós temos todos os dados que nos interessam por isso é só aplicá-los para a função G03 que todos nós conhecemos muito bem.

```
(EXECUTAR)
G41D82X[#12+#19]Y#22F#9            (ativação compensação do raio)
G03X-#19Y#23R#20                   (ataque circular)
G03I-#12J-#12                      (execução da cavidade)
G03X-#23Y#19R#20                   (Destacamento circular)
G01G40X-#22Y-[#12+#19]F[#9*3]      (desativação compensação do raio)
```

O ciclo será repetido de acordo com o número de vezes presente em #10, por isso vamos aumentar o contator para repetir.

```
#13=#13+1.0(AUMENTO CONTACTOR HORIZ.)
END2 (END CICLO ORIZZ.)
```

O que acontece no final do loop horizontal?
A ferramenta retorna para o centro, o programa irá analisar se prosseguir com outro ciclo vertical (assim repetindo o horizontal) ou acabar.

Devemos, portanto, fechar o ciclo vertical.

```
(FINAL CICLO VERTICAL).
END2
```

Não é mau, o que acha?

O que resta a ser feito?
Se a tarefa da macro estiver concluída (processamento final), é restaurar os dados modais e sair da macro.

```
(RESTAURAR DADOS MODAIS)
G90G0Z#5
G#3G#16
M99
```

Em G90 vai a ferramenta para uma altitude segura (#5),
Então nós colocamos os dados encontrados na entrada modal #3 (G90..) e #16 (G00..) em seguida, retornar ao programa que chamou a macro com o M99.

Aqui é a nossa macro!

```
O9010 (CORTE INTERNO CIRCUlo)
(18/09/2011  REV 1.0)
(CHAMADA  G200 Z R F Q I D A B)
----
(Aqui todos os comentários que gostaria)
(FERRAMENTA CENTRO FURO)
----
(#26=Z FINAL Z CAVIDADE)
(#18=R INICIO Z CAVIDADE)
(#9=F  VELOCIDADE)
(#17=Q REMOCAO)
(#4=I  DIAMETRO FRESA)
(#7=D  DIAMETRO FURO)
(#1=A  FURO PASSANTE/CEGO)
(#2=B  DIAMETRO PRE-FURO)
(#6=K  K=0.2 DESBASTE/DEFAULT)
(#6=K  K=0 ACABAMENTO)
IF[#26EQ#0]THEN #3000=1(FALTA Z)
IF[#18EQ#0]THEN #3000=2(FALTA R)
IF[#9EQ#0]THEN #3000=3(FALTA F)
IF[#17EQ#0]THEN #3000=4(FALTA Q)
IF[#7EQ#0]THEN #3000=5(FALTA I)
```

```
IF[#7EQ#0]THEN #3000=6(FALTA D)
IF[#2EQ#0]THEN #3000=7(FALTA B)
IF[#1EQ#0]THEN #1=1(DEFAULT)
IF[#6EQ#0]THEN #6=0.2(DEFAULT)
IF[#18LE#26]THEN #3000=14(VALORES Z/R ERRADOS)
IF[#9LE0]THEN #3000=15(VELOCIDADE ERRADA)
IF[#17LE0]THEN #3000=8(VALOR Q ERRADO)
IF[#7LE0]THEN #3000=9(VALOR D ERRADO)
IF[#6LT0]THEN #3000=13(VALOR K ERRADO)
IF[#6GT1]THEN #3000=16(VALOR K ERRADO)
IF[#2LE0]THEN #3000=10(VALOR B ERRADO)
IF[#2GE#7]THEN #3000=11(VALOR B MAIOR DE D)
IF[#4LE0]THEN #3000=9(VALOR I ERRADO)
IF[#4GT#2]THEN #3000=12(I SUPERIOR B)
(SALVAR DADOS MODAIS)
#16=#4001(G00..)
#3=#4003(G90..)
#5=#5003(Z atual)
(CONVERTENDO VARIAVEIS)
#2=#2/2.0(RAIO FURO ENTRADA)
#10082=[#4/2.0+#6](RAIO FRESA = RAIO FRESA + SOBRE-METAL)
#7=#7/2.0(RAIO CAVIDADE)
#8=500.0(AVANCO EM Z)
(ACABAMENTO/UMA PASSADA)
IF[#6EQ0]THEN#2=#7-2.0
(NUM PASSADAS HORIZ)
#10=FUP[[#7-#2]/[#4*2.0]]
(AUMENTO HORIZ.)
#11=[#7-#2]/#10
(CONTROLE FRESA/FURO DE ENTRADA)
IF[#4GE#2]THEN#8=#8/10
(ABORDAGEM)
G90G0Z[#18+5.0]
G01F1000Z#18
(CICLO VERTICAL #5003=#26)
WHILE[#5003NE#26]DO1
(CONTACTOR CICLO HORIZ.)
#13=1.0
(CONTROLE PROXIMA PROFUNDIDADE)
#14=#5003-#17
(CALCULO FALTA DE Z)
#15=ABS[#26-#5003]
(SE PASSANTE DESVIA)
IF[#1NE0]GOTO20
(SE NO FUNDO)
IF[#15EQ#17]THEN#8=#8/10
(ALEM DO FUNDO)
N20IF[#14LE#26]THEN#17=#15
G91G01F#8Z-#17
(CICLO HORIZ.)
WHILE[#13LE#10]DO2
(CALCULO PONTOS ENTRADA CIRCULAR)
(RAIO RELATIVO)
#7=#2+[#11*#13]
(RAIO DE SAIDA)
#20=[#7+#4]/2.0
#21=[#7-#20]
#22=#21*SIN[45]
#23=#20*SIN[45]
#19=#20-#23
#12=#22+#23
(EXECUTAR)
G01G41D32X[#12+#19]Y#22F#9
G03X-#19Y#23R#20
```

```
G03I-#12J-#12
G03X-#23Y#19R#20
G01G40X-#22Y-[#12+#19]F[#9*3]
(AUMENTO CONTACTOR HORIZ)
#13=#13+1.0
(FINAL CICLO HORIZ.)
END2
(FINAL CICLO VERTIC.)
END1
(RESTAURAR DADOS MODAIS)
G90G0Z#5
G#3G#16
M99
```

Você apenas tem que experimentá-la (com cuidado) e depois usá-la. Como mencionado acima, você pode salvar uma cópia em algum lugar e colocar apenas a versão sem comentários no CNC.
Lembre-se também para proteger o seu programa com as funções apropriadas.

Nós estamos no fim, a nossa macro é definida: o que fazer?

Lembre-se que para experimentar e usar a macro você deve definir o parâmetro associado com o programa O9010 com o número 200.

Toda vez que fizer uma cavidade cilíndrica você pode usar o código G200 .. Este é um novo G-código da máquina como qualquer outro. Você não é obrigado a usar a macro em situações particulares, mas quando você quer.

A criação de um programa de macro pode ser feito de várias maneiras.
Por exemplo, você pode considerar o uso da interpolação em espiral
(o exemplo foi deliberadamente mantido simples).

Uso correto deste macro, como todo o resto, fica a critério do operador que deve ser inteligente o suficiente para compreender os valores reais ou não para ser aplicado (Refiro-me aos valores reais e hipotéticos a serem atribuídos aos argumentos).

Antes de escrver o programa certifique-se de conhecer o comportamento do CNC, nem todos funcionam da mesma maneira. Cada CNC é definida usando parâmetros que determinam o funcionamento e, embora, em geral, as diferenças são mínimas, são estes que podem ser desconcertante. Por exemplo, a eliminação de variáveis comuns pode ou não acontecer premindo o botão RESET, o "/" pode ou não estar ativo para macros ....

Por fim, assunto muito importante, sempre perguntar se a macro é apenas um conjunto de operações, ou está bem estruturado de acordo com os princípios listados no começo.

Tente escrever código G para a realização da figura pag 3.
Escolha os valores de profundidade, diâmetro e posicionamento que quiser.
Agora escreva o código utilizando a macro: o que acha?

O que podemos ainda dizer?

Escrever um programa macro é extrapolar um único problema para torná-lo generalizado.
Não inventamos nada de novo, mas escrevemos de forma diferente.
Leia atentamente (se o meu português permitir) e tentar avaliar onde podemos melhorar o programa.
Por agora vou parar esperarando seus comentários
Até