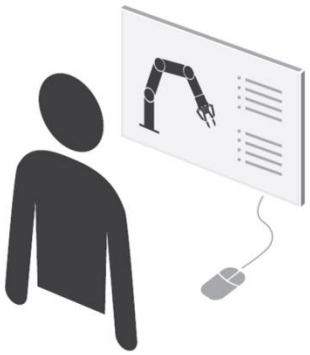
Introdução a Linguagem *RAPID*



Estrutura de um programa

Um programa é identificado como um módulo, com sua denominação definida pelo programador e com um ou diversos procedimentos contendo as instruções. A definição de modulo é feita pela instrução MODULE e finalizada com **ENDMODULE.**

MODULE *nome_modulo*

instruções

.....

.....

ENDMODULE

Dentro de um módulo deve existir ***um programa principal,*** definido como ***PROC MAIN***, podendo conter ou não a inserção de outros *PROC*, tratados como *subprogramas.*

PROC MAIN ( )

instruções

.....

.....

ENDPROC

Programação de Robôs

Uma SUB-ROTINA, corresponde a *subprograma qualquer (**Proc**)*

As sub-rotinas funcionam como programas secundários que podem ser requisitados pelo programa principal apenas referenciando o seu nome.

PROC *nome_prog* ( )

instruções

.....

.....

ENDPROC

Programação de Robôs

Exemplos..

```
PROC GARRA ( )
Instruções
.....
END PROC
```

```
PROC VENTOSA ( )
Instruções
.....
ENDPROC
```

```
PROC  main ( )
      ! programa para paletização com 2 sub-rotinas

      GARRA;    ! chamada de sub-rotina GARRA
      VENTOSA; ! chamada de sub-rotina VENTOSA
ENDPROC
```

Características de sintaxe..

Inserção de Comentários no programa

- ✓ Comentários podem ser inseridos no programa, desde que antecedidos do sinal de exclamação "!"
- ✓ Podem constar em uma linha independente, ou após uma instrução qualquer.

Finalização de uma instrução

Uma instrução (comando, declaração, etc.), é concluída com o caracter ; (*ponto e vírgula*)

Declaração de Variáveis

As variáveis de programa devem ser declaradas no **MÓDULO**, atribuindo-se um *nome* e o respectivo *tipo*. A declaração é feita com a palavra **VAR**, no formato.

VAR *tipoVar* *nomeVar* ;

Os nomes de variáveis podem ser definidos pelo usuário, e os tipos corresponderem, respectivamente, entre outros, para:

- *Num* – numérico
- *Bool* – bolena
- *String* – texto
- *Robtarget* - posição

Declaração de Variáveis

Atribuição de valores

As variáveis precisam ser inicializadas, isto é, assumirem um valor inicial, que deve ser definido. A atribuição é feita acrescentando-se após o nome o símbolo "**:=**" e declarando a valor assumido.

exemplos;

Reg **:=**1;

VAR string *nome1* **: =** "pedro";

VAR num *vetor* {10} **:=** **[**1,2,3,8 ,9,7,6,5,4,3**]**;

Variáveis

Alteração de valores

Algumas funções especiais permitem alterar ou incrementar o valor de determinada variável inicializada.

Exemplos, para uma variável r*eg1*:

CLEAR	*reg1*	→	cancela o registro da variável
INCR	*reg1*	→	incrementa de 1 unidade
DECR	*reg1*	→	diminui de 1 unidade
ADD	*reg1*, *x*	→	adiciona um valor (*x*) a variável, reg1

Programação de Robôs

Controle no fluxo de programas

Desvio Condicional - "IF"

O comando ***IF*** tem duas modalidades, uma forma compacta, que admite apenas um comando em resposta a condição de teste e outra completa, com desvios de instruções em função da um resultado verdadeiro ou falso da condição testada.

forma compacta

IF <condição> comando

ENDIF

---

forma completa

IF <condição> THEN

..instruções...

.. ELSE

...instruções...

ENDIF

Controle no fluxo de programas

Comando WHILE

Permite a execução de uma série de instruções enquanto determinada expressão for verdadeira

WHILE <*exp*> **DO**

.....

instruções... execução para número indefinido de vezes

ENDWHILE

Para <*exp*> correspondente a uma expressão qualquer

Comando "FOR"

Permite a execução de uma série de instruções por um número de vezes (de *ni* a *nf*) controladas através do valor de uma variável (*i*)..

FOR *i* FROM *ni* **TO** *nf* **DO**
...instrução
.....
instrução..
ENDFOR

para ***i*** - *variável de controle*

ni - valor inicial da variável *"i"*

nf - valor final da variável *"i"*

Desvio condicional "TEST"

Desvio do programa em função do valor (***vn***) assumido pela variável de controle **(*NomeVar*)**

TEST *NomeVar*

CASE ***v1***:

...instruções...

CASE *v2*:

...instruções...

CASE *v3:*

.....

DEFAULT:

!... valor não previsto para *vn*

ENDTEST

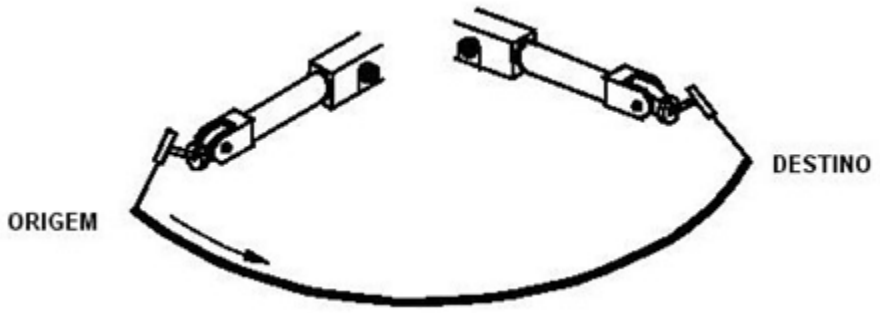
onde *v1, v2.. vn* correspondem a valores assumidos pela variável de controle (*NomeVar*)

Trajetórias (Interpolação entre Posições)

Uma trajetória é sempre definida entre posições previamente conhecidas (***origem e destino***), executada por interpolação entre pontos intermediários em diferentes modalidades.

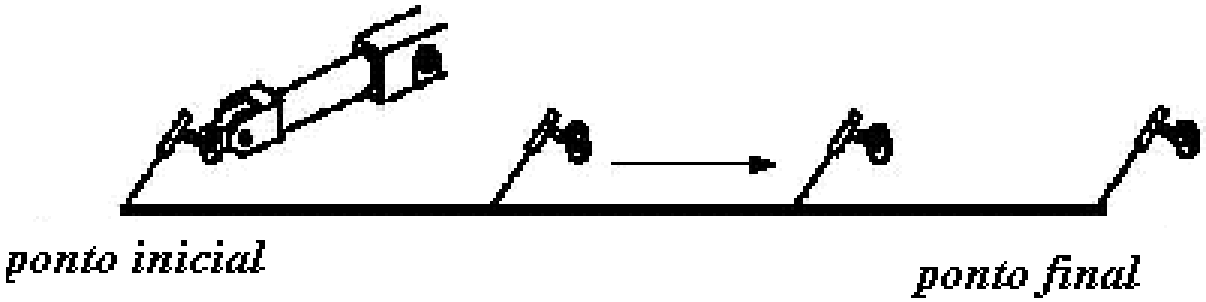
Interpolação definidas por movimento livre - "JUNTAS"

Opção para movimentação fácil e rápida entre um ponto e outro quando a precisão da trajetória não é importante. Todos os eixos se movem com velocidade constante.



Interpolação “LINEAR”

Em uma interpolação linear, o TCP (referência da extremidade do manipulador) se desloca seguindo uma linha reta entre ponto inicial e destino.



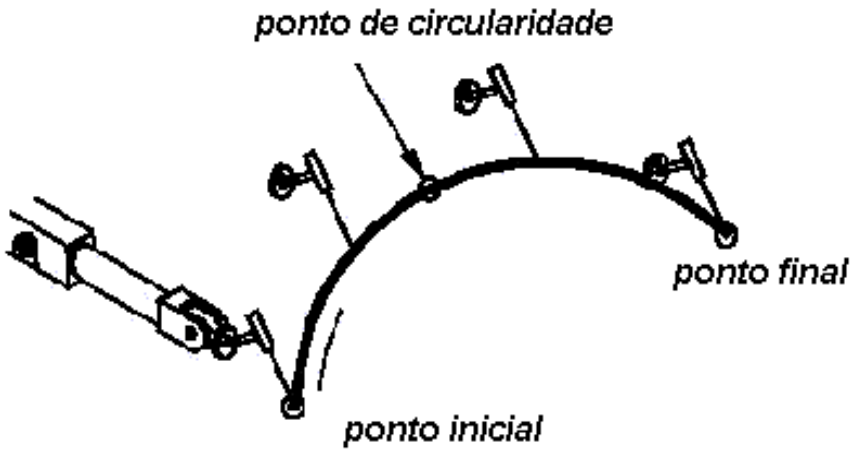
Interpolação "CIRCULAR"

A interpolação circular é definida por três posições que descrevem um segmento circular.

PONTO *Inicial*

PONTO de *Circularidade*

PONTO *Final*



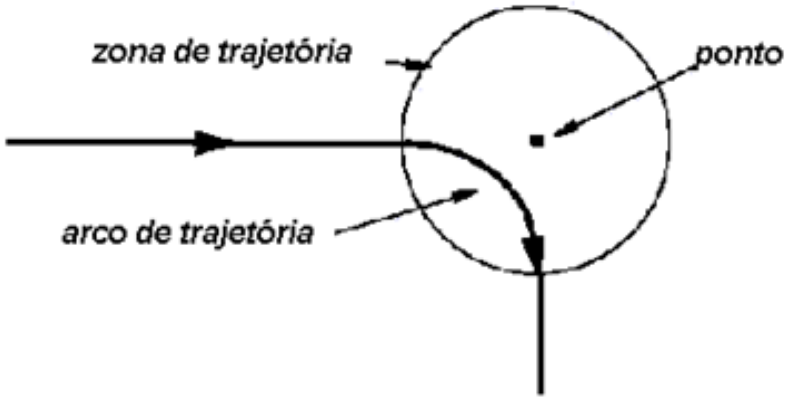
MODOS DE INTERPOLAÇÃO NAS MUDANÇAS DE TRAJETÓRIA

A interpolação pode ser do tipo **ponto a ponto** ou ***fly-by***.

- **PONTO A PONTO**: o ponto de destino impõe uma parada, até que todas as velocidades sejam zeradas.

- ***FLY-BY***: modalidade em que há *continuidade do movimento* passando pelas posições programadas em alta velocidade sem utilizar-se de desnecessárias paradas.

FLY-BY gera uma trajetória parabólica passando pelo ponto programado, não atingindo o mesmo. O início e o fim dessa transição são definidos pela região (**Zona)** em torno da posição específica.



Instruções para diferentes MOVIMENTOS

MOVE “?”

Movimento Controlado - *Circular*

MOVEC – movimento (TCP) circularmente para um dado destino; orientação não muda em relação ao círculo.

MOVEC *CirPoint, ToPoint ,Speed ,Zone, Tool;*

Obs: A ferramenta é reorientada a uma velocidade constante da posição inicial a final. A reorientação é realizada relativamente à trajetória circular, em determinada velocidade (*Speed*) com certa ferramenta (*Tool*), passando por determinada região (*Zone*).

Programação de Robôs

Movimento "*não controlado*" – *LIVRE*

MOVEJ – todos os eixos atingem o destino (posição) ao mesmo tempo.

MOVEJ *ToPoint, SPEED, ZONE, TOOL;*

Movimento "controlado" - Linear

MOVEL – movimento linear do TCP.

MOVEL *ToPoint, SPEED, ZONE, TOOL;*

Movimento relativo (sistema referência base)

OFFS – comando usado para adicionar um *offset* para uma posição, do robô. Os valores de deslocamento (em x, y, z) são em unidades correntes (mm).

OFFS **(POINT** ***X****offset* ***Y****offset* ***Z****offset* **)**

POINT – ponto destino (*robtarget*)

Xoffset *deslocamento em x*

Yoffset *deslocamento em y*

Zoffset *deslocamento em z*

Movimento Relativo (sistema referência ferramenta)

RELTool – comando para um movimento relativo no sistema de referência da ferramenta. Os valores de deslocamento são em unidades correntes, milímetros para deslocamentos lineares e graus para as rotações.

RELTool **(Ponto, dx, dy,dz** \[Rx] \[Ry] \[Rz] **)**

Onde

dx, dy, dz - deslocamentos nas respectivas direções x, y e z dos eixos da ferramenta;

Rx, Ry, Rz - Rotações em torno dos respectivos eixos x, y e z da ferramenta.

ps. A função é aplicada associado a um comando de movimento:

MOVEJ Reltool.., MOVEL Reltool.. ou ***MOVEC Reltool ...***

Interação Homem/Máquina (via *Teach-pendant*)

Escrita de mensagem no display do *teach-pendant*.

- ✓ ***TPWrite*** "mensagem...";

Leitura de variáveis no teclado do *teach-pendant*.

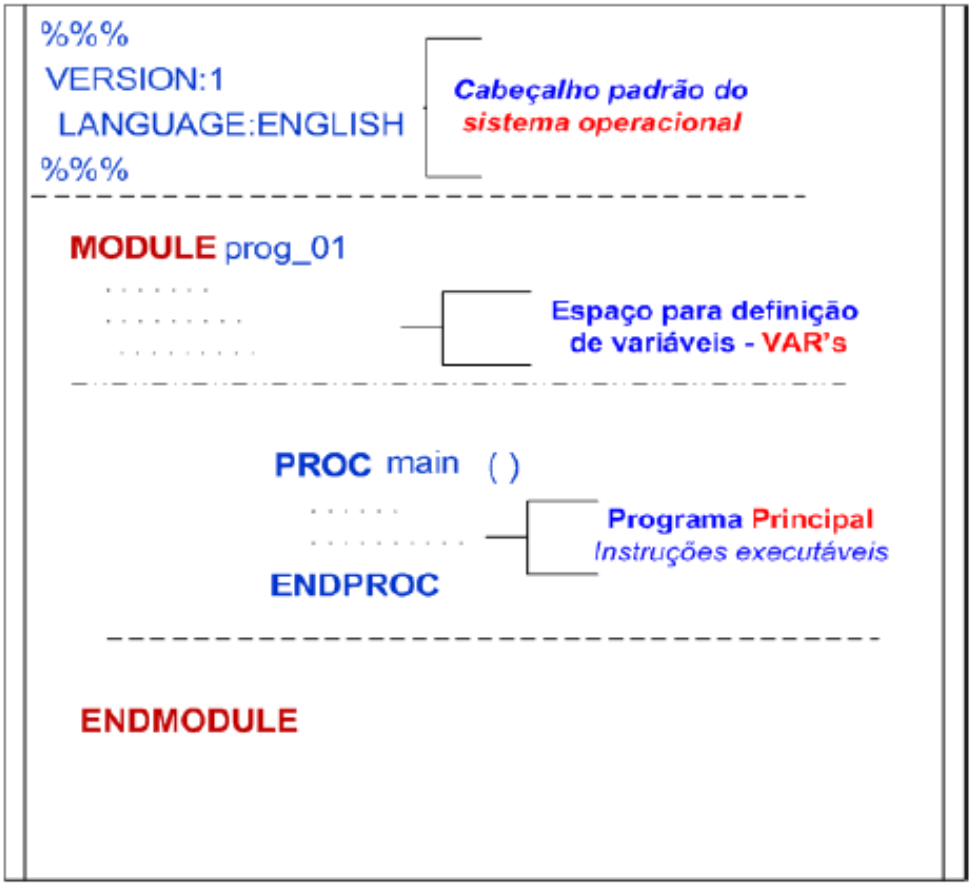
- ✓ ***TPRead*** *NomeVar*, " ";

TPReadNum *NomeVar*, " "**;** (ler a variável numérica "*NomeVar*").

Programando em *RAPID*

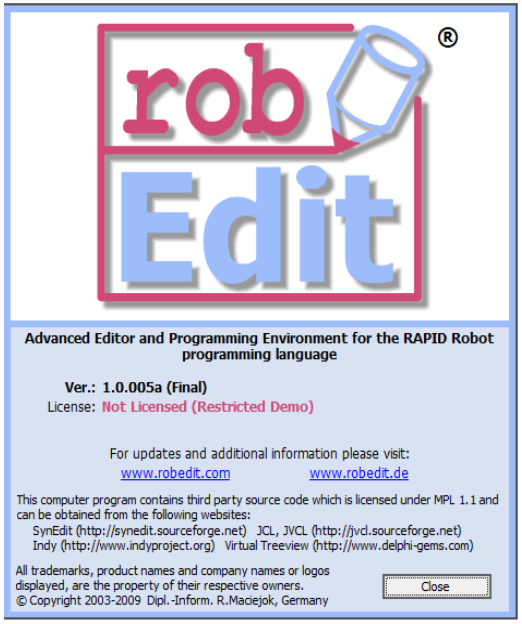
- Estrutura de Programa
- Exemplo de Editor
- Exemplo de Código

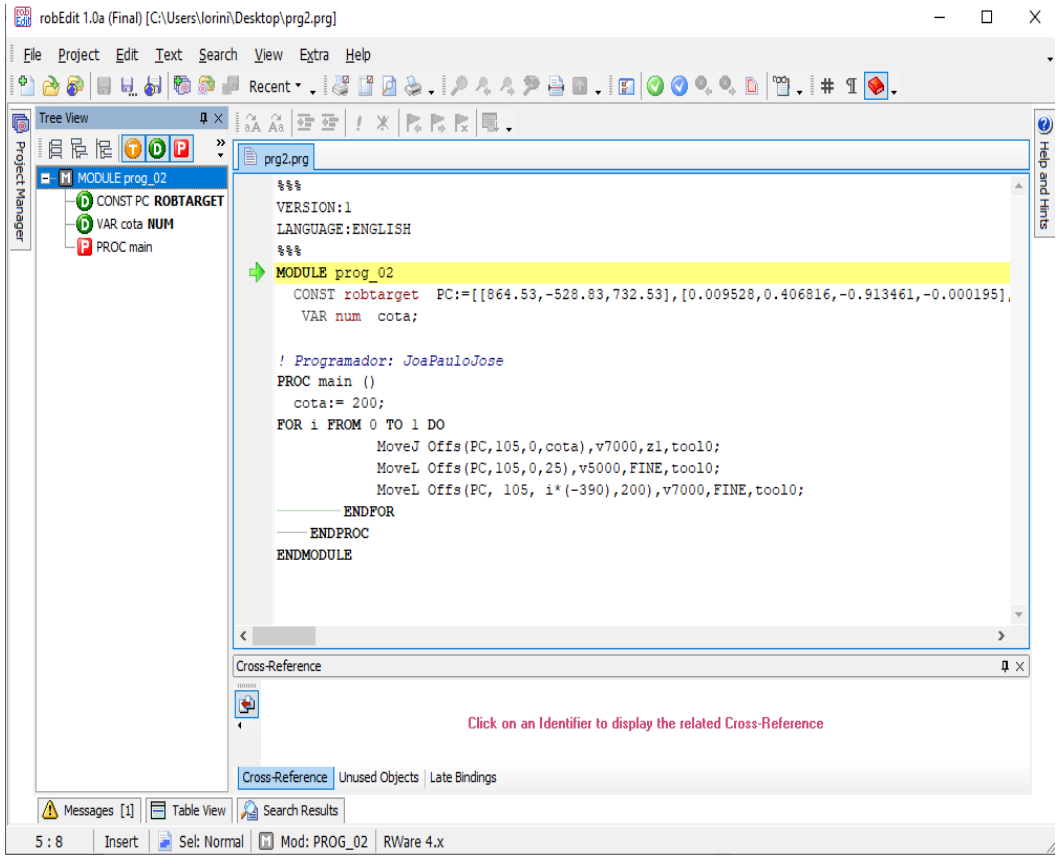
Exemplo de Estrutura de Programa



Programação de Robôs

Exemplo de Editor de Programa





UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Programação de Robôs

Exemplo de Código de Programa

```
%%%
VERSION:1
LANGUAGE:ENGLISH
%%%
MODULE prog_02
  CONST robtarget  PC:=[ [864.53,-528.83,732.53], [0.009528,0.406816,-0.913461,-0.000195], [-1,-1,-1,0],
      [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]  ] ;
   VAR num  cota;
! Programador: JoaPauloJose
   PROC main ()
   cota:= 200;
              FOR i FROM 0 TO 1 DO
                  MoveJ Offs(PC,105,0,cota),v7000,FINE,tool0;
                  MoveL Offs(PC,105,0,25),v5000,FINE,tool0;
                  MoveL Offs(PC, 105, i*(-390),200),v7000,FINE,tool0;
               ENDFOR
    ENDPROC
ENDMODULE
```