Adriano A. Santos
Departamento de Engenharia Mecânica
Politécnico do Porto

Programação gráfica

1.ª Parte

INTRODUÇÃO

O desenvolvimento de um programa passa pela definição de uma série de instruções utilizadas para, de forma condicional, executá-lo ou repeti-lo um determinado número de vezes. Para isso, poderão ser utilizadas várias representações de código que passam pela escrita de texto estruturado, pela representação gráfica de elementos lógicos, por blocos função ou por estruturas que encerram no seu interior parcelas de código como é o caso das estruturas do LabVIEW.

ESTRUTURAS "*ENQUANTO QUE*" E "*PARA*"

As estruturas de programação a utilizar no LabVIEW estão localizadas no painel *Function* do *Block Diagram*. Na Figura 1 apresentam-se as diversas estruturas disponibilizadas pelo LabVIEW.

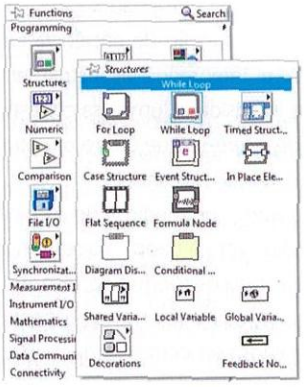

Figura 1. Painel Functions, Structures (Fonte: Ruchitha G.).

No painel *Functions* podemos encontrar estruturas do tipo ciclo (*For, While*), condicionais e de eventos (*Case, Event*, entre outros), que iremos analisar ao longo desta secção.

CICLO "*ENQUANTO QUE*" (*WHILE LOOP*)

O ciclo *While Loop*, conjuntamente com o ciclo *For Loop*, integram-se nas estruturas iterativas, também chamadas de estruturas de ciclo. Assim sendo, o ciclo *While* repetirá, consecutivamente, o código inscrito no seu interior até que seja verificada uma determinada condição. Esta será avaliada em cada iteração (Figura 2).

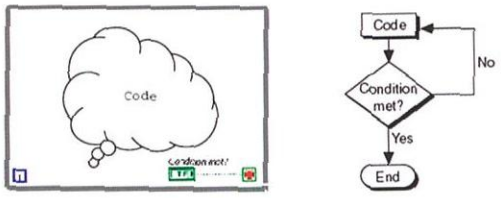

Figura 2. Estrutura While Loop e fluxograma de funcionamento (Fonte: NI a).

A estrutura *While* dispõe de um terminal de iteração, quadrado azul com o símbolo "*i*", considerado como um terminal de saída uma vez que contém o número de iterações concluídas. A tagem de iterações para o ciclo *While* começa sempre em e é sempre executada pelo menos uma vez. O terminal co cional ou dependente, quadrado verde, considerado como terminal de entrada, condicionará a execução do ciclo em ção da receção de um valor verdadeiro ou falso. As cond de funcionamento são definidas utilizando-se o botão d do rato, definindo-se a paragem do ciclo de acordo com o booleano do terminal: Verdadeiro (*Stop if True*) ou Falso (*C nue if True*) (Figura 3).

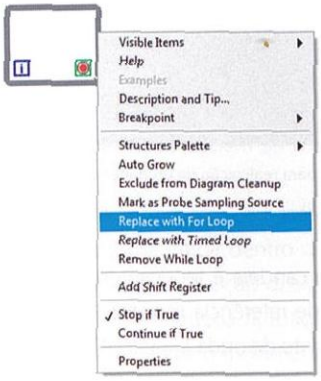

Figura 3. Menu de contexto da estrutura While Loop (Fonte: NI b).

Por outro lado, como na maioria dos programas, o me contexto é utilizado para definir ações ou funções espe criar constantes, controlos ou indicadores. Assim, analisar de forma sucinta o menu de contexto do ciclo deparam com algumas opções que se poderão tornar relevantes truturação e na organização do programa. Destas pode chamar à atenção para as seguintes opções:

- ***Visible Item*** – Permite ocultar a etiqueta de identific
- ***Description and Tip*** – Permite adicionar comentáric
- ***Breakpoint*** – Permite estabelecer um ponto de pa para se depurar o VI;
- ***Replace with ...*** – Permite alterar o ciclo para *For Loo Loop* ou para qualquer outra função utilizando-se a de estruturas (*Structures Palette*);
- ***Remove Loop*** – Permite apagar a estrutura, mas sen nar o código inscrito no seu interior.

A opção *Add Shift Register* (adicionar terminais) permit cionar, a cada um dos lados verticais da estrutura, um terminais. Os terminais adicionados permitem passar pe valores das interações anteriores transferindo-os para xima interação. O terminal do lado direito do ciclo, co uma seta que aponta para cima, é responsável pelo arr mento dos dados da interação concluída (Figura 4). O L transferirá os dados interligados ao lado direito do re para a próxima iteração. Após a execução do ciclo, o t do lado direito do ciclo devolve o último valor armazer *Shift Register*.

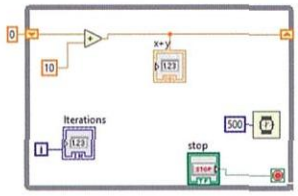

Figura 4. Shift Register num ciclo *While* (Fonte: Microcontrollers).

Por sua vez, o *Feedback Node*, situado no lado esquerdo da borda da estrutura do ciclo, permitirá definir o valor inicial do nó. Este terminal contém uma seta que aponta para baixo e transmite o valor lido à interação (Figura 4).

A entrada e saída de dados no *While Loop* é realizada através de "*túneis*" que são criados automaticamente nas laterais quando são cruzados pela cablagem, ligações. Estes são representados por um quadrado.

CICLO PARA (*FOR LOOP*)

O ciclo *For Loop* repete o código, inscrito no seu interior, o número de vezes definido inicialmente e que não poderá ser alterado após se iniciar a sua execução. Esta estrutura, à semelhança do *While Loop*, dispõe de um terminal de iteração, quadrado azul com o símbolo "*i*" considerado como um terminal de saída, contém o número de iterações concluídas. A contagem de iterações para o ciclo *For* começa, também ele, em zero (Figura 5).

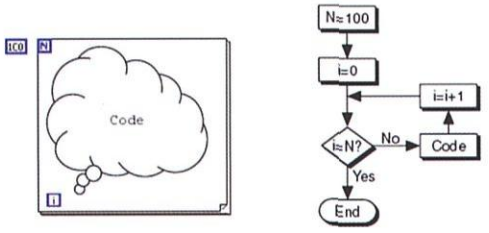

Figura 5. Estrutura *For Loop* e fluxograma de funcionamento (Fonte: NI a).

O terminal de contagem, quadrado azul colocado no canto superior esquerdo, é considerado como um terminal de entrada e permite fixar o número de vezes de repetições do ciclo. É representado pela letra "*N*" a que se associo um valor que indicará quantas vezes o subdiagrama será repetido.

A este ciclo são associados os mesmos elementos de controlo, *Sift Register* e *Feedback Node*, do mesmo modo e de acordo com a explicação dada no ciclo *While Loop*. Os menus de contexto do ciclo *For* são muito semelhantes aos menus apresentados para o ciclo *While Loop*. A entrada e saída de dados é realizada através de "*túneis*" que são criados automaticamente nas laterais da estrutura do ciclo. Na Figura 6 apresenta-se a estrutura do ciclo *For* associado à função *Timing*. Esta funcionalidade permite temporizar o tempo de execução quer do ciclo *For Loop* quer do ciclo *While Loop*.

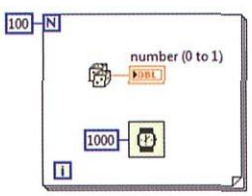

Figura 6. Estrutura *For Loop* com temporização (Fonte: NI c).

ESTRUTURAS COM CASOS E EVENTOS

Estas estruturas são caraterizadas por conterem múltiplos subdiagramas ou casos sendo, no entanto, visível somente um de cada vez.

Estrutura caso (*Case Structure*)

Esta estrutura é utilizada para executar apenas um caso de cada vez pelo que, dependendo do valor de entrada, é selecionado o subdiagrama a ser executado. A estrutura *Case* é similar a declarações de seleção ou comandos *If ... Then ... Else* das linguagens de programação baseadas em texto. Esta é composta por um identificador e seletor.

- ***Identificador*** – está localizado na parte superior da estrutura indicando os distintos casos. A seleção de um destes casos permite visualizar e editar o subdiagrama a executar. O seletor de caso apresenta no centro o nome do valor do seletor para além de setas de incremento e decremento. Com a ajuda das setas laterais e da seta que aponta para baixo podemos selecionar um dos casos apresentados no menu. Utilizando-se o menu de contexto podemos adicionar novos casos, eliminar, duplicar, entre outros, ou mesmo reordená-los recorrendo à opção *Rearrange Case*.
- ***Terminal selector*** – está localizado no lado esquerdo da estrutura associado ao símbolo "?". É utilizado para selecionar, em função da condição de chegada, o subdiagrama a executar. Poderão ser utilizados valores inteiros, booleanos, *string* ou um qualquer valor numérico de acordo com o exposto em seguida:
  - ***Booleano***: atuará como uma sentença *If ... Then ... Else* numa linguagem de texto tradicional. O seletor booleano só poderá assumir os valores Verdadeiro ou Falso (*True* ou *False*) (Figura 7). Executar-se-á um dado código caso a condição seja verdadeira e um outro código caso seja falsa.
  - ***Numérico***: a condição provocará a execução do código associado ao menu *Case* que possua o número igual ao número do seletor. Quando se pretende executar o mesmo código num dado intervalo de valores, lista de valores, escrevemos no menu do *Case* os valores pretendidos separados por vírgulas ("*2, 3, 4*") ou define-se um dado intervalo, que poderá ser fechado ou aberto, como por exemplo "*-10..-1*" ou "*3..*", utilizando-se sempre números inteiros (Figura 7). Utilizando *enum* ou *ring* podemos escrever, diretamente, no seletor o número do item que se pretende executar.
  - ***String***: esta *Case Structure* diferencia-se das anteriores pelo facto de no menu do identificador se utilizar um texto limitado por comas ("") (Figura 7). A condição provocará a execução do menu *case* que possua o texto igual ao do seletor.

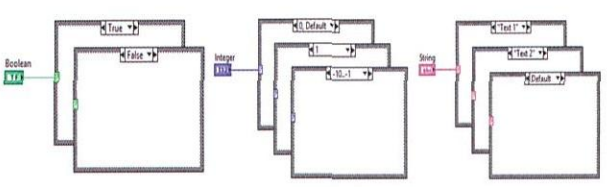

Figura 7. Ativação de uma estrutura Case (Fonte: Wiki).

A utilização de uma *string* ou de um valor numérico na ligação ao seletor obriga à definição de um *case* que deverá ser executado por defeito. Esta definição pode ser realizada escrevendo-se no identificador uma coma seguida de *Default* ou selecionando *Make This The Default Case* no menu de contexto. Assim, dependendo do tipo de dado que se utiliza na ligação ao seletor (booleano, *string* ou numérico) este assumirá a cor verde, azul ou magenta, respetivamente (Figura 7).

Estrutura de eventos *(Event Structure)*

A estrutura de eventos é similar à estrutura de caso. Esta permite executar uma ou outra parte do código em função dos diversos eventos associados ao seletor. Estes podem ser *clics* ou movimentos do rato, ações de maximizar ou minimizar janelas, ações sobre teclas, entre outros.

Estruturas de sequência (*Flat* e *Stacked Sequence*)

As estruturas de sequência são constituídas por um ou mais subdiagramas ou quadros que são executados sequencialmente. Em cada quadro de uma estrutura de sequência, como nos restantes diagramas de blocos, a dependência de dados determina a ordem de execução dos nós, isto é, só são executados quando estão disponíveis todos os dados. Assim sendo, quando é necessário executar um dado diagrama antes de outro deveremos utilizar uma estrutura de sequência. Com estruturas de sequência poderemos quebrar o paradigma de fluxo de dados da esquerda para a direita sempre que se usar uma variável local de sequência.

Devemos ainda considerar que os túneis de saída de uma estrutura de sequência podem ter apenas uma fonte de dados, ao contrário das estruturas *Case*. A saída pode ser emitida a partir de qualquer quadro. Tal como acontece com as estruturas *Case*, os dados nos túneis de entrada estão disponíveis para todos os quadros na estrutura de sequência plana ou sequência empilhada.

Existem dois tipos de estruturas de sequência - a estrutura de Sequência Plana (*Flat Sequence*) e a estrutura de sequência Empilhada (*Stacked Sequence*). As estruturas de sequência planas e empilhadas são apresentadas na Figura 8.

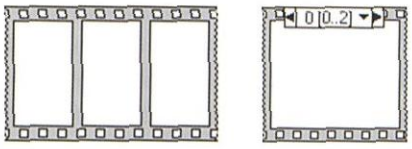

Figura 8. Estrutura de Sequência Plana (*Flat Sequence*) e Empilhada (*Stacked Sequence*) (Fonte: NI d).

Sequências planas

A estrutura de Sequência Plana (*Flat Sequence*) apresentam uma estrutura que faz recordar um fotograma de uma película. Os quadros são executados sequencialmente uns atrás dos outros, da esquerda para a direita, e quando todos os valores dos dados ligados a um quadro se encontrarem disponíveis. Os quadros podem ser interligados, através dos túneis, pelo que os dados podem deixar cada quadro quando a execução do quadro termina. Isso significa que a entrada de um quadro pode depender da saída de outro (Figura 9).

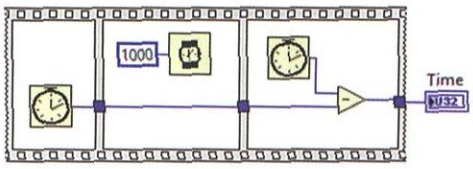

Figura 9. Passagem de dados na Sequência Plana.

Sequências empilhadas

Estruturas de sequência empilhadas (*Stacked Sequence*) garantem a ordem de execução e proíbem operações paralelas. Por exemplo, tarefas assíncronas que usam dispositivos de E/S, como PXI, GPIB, portas série e dispositivos DAQ, podem ser executadas simultaneamente com outras operações se as estruturas de sequência não as impedirem. Estas estruturas são identificadas por um menu alocado na parte superior indicando a numeração do quadro visualizado bem como o número total de quadros que contem.

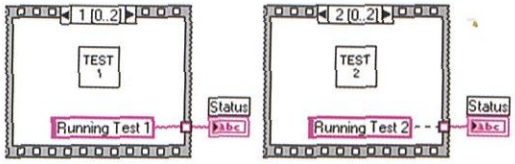

Figura 10. Sequência empilhada (*Stacked Sequence*).

Ao clicar-se com o botão direito do rato na borda da estrutura podemos adicionar e excluir quadros ou criar sequências locais responsáveis pela passagem de dados entre quadros. Usa-se a estrutura de sequência empilhada para garantir que um subdiagrama é executado antes ou depois de um outro subdiagrama. Nos diagramas apresentados na Figura 10 apresenta-se um VI de teste com um indicador de *status* que exibe o nome do teste em andamento.

REFERÊNCIAS

- Martínez, J. Molina – *Programación gráfica aplicada*. In Canales, António Ruiz y Martínez, José Molina – *Automatización y Telecontrol de Sistemas de Riego*. Barcelona. Editora Marcombo, 2010. ISBN 9788426716347. Cap. 18.
- Ruchitha Geebu, "*Labview Structures*", *online*. Acedido a 6 agosto 2021. Disponível em https://mindmajix.com/labview/structures.
- National Instruments (NI a), "*Estruturas de execução no LabVIEW*", *online*. Acedido a 6 agosto 2021. Disponível em www.ni.com/getting-started/labview-basics/pt/execution-structures.
- National Instruments (NI b), "*LabVIEW Idea Exchange*", *online*. Acedido a 6 agosto 2021. Disponível em https://forums.ni.com/t5/LabVIEW-Idea-Exchange/Add-quot-Replace-with-For-Loop-While-Loop-quot-methods-to-VI/idi-p/2598641?profile.language=en.
- Microcontroller Lab, "*Labview Tuorials – Shift registers in labview exemples*", *online*. Acedido a 6 agosto 2021. Disponível em https://microcontrollerslab.com/shift-registers-labview-examples/.
- National Instruments (NI c), "*6. Execution Structures*", *online*. Acedido a 7 agosto 2021. Disponível em www.ni.com/academic/students/learnlabview/i/execstructures.htm.
- LabVIEW Wiki, "*Case Structure*", *online*. Acedido a 7 agosto 2021. Disponível em https://labviewwiki.org/wiki/Case_structure.
- National Instruments (NI d), "*Sequence Structures: Executing Sections of Code Sequentially*", *online*. Acedido a 8 agosto 2021. Disponível em https://zone.ni.com/reference/en-XX/help/371361R-01/lvconcepts/sequence_structures_concepts/.