LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

AULA 4

Prof. Sandro de Araujo

CONVERSA INICIAL

Esta aula tem como base os livros *Lógica de programação e estruturas de dados*, *Lógica de programação algorítmica* e *Treinamento em Linguagem C* (vide a seção **Referências** para mais informações sobre cada obra). Em caso de dúvidas ou se desejar aprofundar seus conhecimentos, consulte as obras disponibilizadas em nossa Biblioteca Virtual Pearson.

A aula apresenta a seguinte estrutura de conteúdo:

1. Estrutura de repetição;
2. Estrutura de repetição *while* (enquanto);
3. Estrutura de repetição *do-while* (repita-ate);
4. Estrutura de repetição *for* (para);
5. Estrutura de repetição aninhada.

O objetivo desta aula é conhecer os principais conceitos e aplicações das estruturas de repetições *while* (enquanto), *do-while* (repita-ate) e *for* (para), bem como representá-las em fluxograma, pseudocódigo e Linguagem C para resolver problemas computacionais.

TEMA 1 – ESTRUTURA DE REPETIÇÃO

As estruturas de repetição também são conhecidas como *laços (loops).* Elas são a execução de um conjunto de ações uma vez, várias vezes ou nenhuma vez, dependendo de uma condição verdadeira ou falsa, resultado booleano da avaliação de uma expressão. Essa condição é chamada de *expressão de controle* ou *condição de parada* e está associada a um bloco de instruções (Puga; Rissetti, 2016).

Qualquer estrutura de repetição contém quatro elementos fundamentais:

1. Inicialização – determina a condição inicial da repetição.
2. Condição – é a expressão booleana que, após cada leitura do corpo, avalia e determina se uma nova leitura deverá ser feita ou se a estrutura de repetição pode ser encerrada.
3. Corpo – formado por todas as instruções que serão executadas repetidamente.
4. Iteração – é a repetição de um conjunto de instruções, juntamente com a condição de terminação do laço.

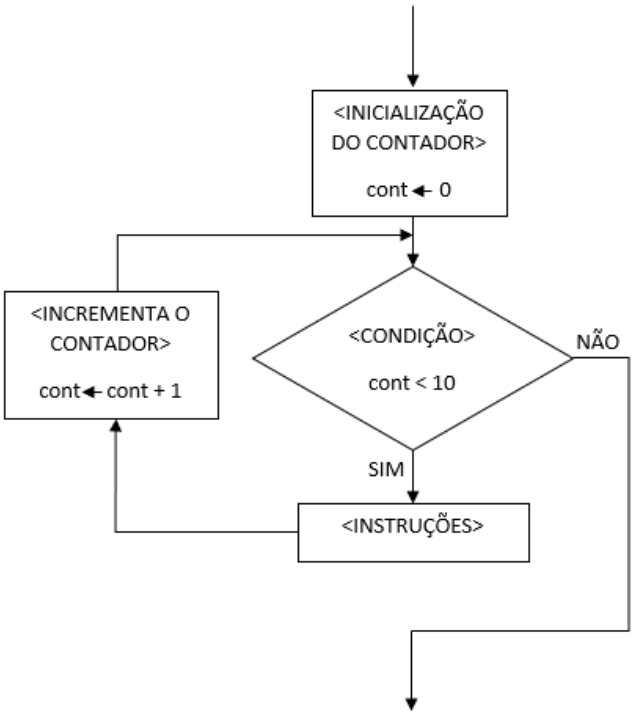
As estruturas de repetição são utilizadas para executar repetidamente uma instrução ou bloco de instrução enquanto determinada condição estiver sendo satisfeita. Sua aplicação pode ser associada a determinadas situações nas quais temos que repetir o programa ou parte dele várias vezes (Guedes, 2014; Puga; Rissetti, 2016).

Essas estruturas nem sempre possuem recursos para fazer o controle do número de vezes em que o laço deverá ser executado, necessitando de uma variável de controle, preferencialmente do tipo inteiro, funcionando como um contador e garantindo que o número de repetições seja finito (Puga; Rissetti, 2016).

A variável contadora começa com um valor inicial e a condição para executar a iteração é o que define a quantidade de repetições do laço. No final de cada iteração, o valor da variável contadora é incrementado em um número definido pelo programador.

O incremento é um recurso que serve pera definir qual valor será somado à variável, por exemplo de 1 em 1 (padrão), de 2 em 2, de 3 em 3, entre outros (Puga; Rissetti, 2016). A Figura 1 mostra graficamente como funciona.

Figura 1 – Incrementação da variável cont (contador)



Ao examinar cuidadosamente a Figura 1, veremos, primeiro, a inicialização é executada, que é a sentença cont = 0. Isso modifica o valor da

variável cont para 0. Então, o teste é executado. Como 0 < 10 é verdadeiro, o laço continua. Assim, o corpo da repetição é executado. Depois disso, o incremento é executado, que é a sentença cont + 1, que altera o valor da variável cont para 1.

Como resultado do contador teremos:

```
INÍCIO DO LAÇO
        cont = 0
        cont = 1
        cont = 2
        cont = 3
        ...
        cont = 9
TERMINOU DE CONTAR!!!
```

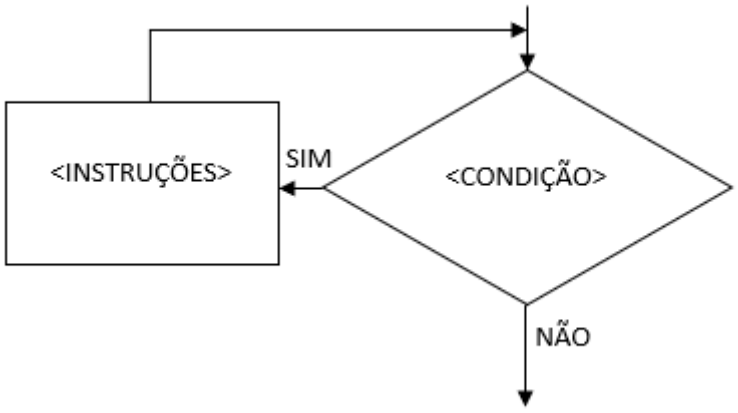
Esse processo de contagem vai de 0 até que a variável cont receba o número 9, totalizando 10 repetições. Depois disso, o contador é incrementado para 10 e o teste é encerrado, 10 < 10 é falso e o laço não continua. Com esse recurso conseguimos que as instruções sejam repetidas dez (10) vezes.

TEMA 2 – ESTRUTURA DE REPETIÇÃO *WHILE* (ENQUANTO)

Na estrutura de repetição *while* (enquanto), a execução de uma ou mais instruções de um bloco ou laço depende de uma condição de controle verificada no início ou na entrada do laço. Enquanto o resultado da condição for verdadeiro, o bloco de instruções é executado; caso contrário, ocorre o desvio para a primeira linha após esse bloco.

Vejamos a seguir a sintaxe da estrutura de repetição *while* em fluxograma, pseudocódigo e Linguagem C. A Figura 2 mostra a sintaxe da estrutura de *while*.

Figura 2 – Estrutura de repetição *while*



O exemplo a seguir mostra a sintaxe da estrutura de repetição *while* na representação algorítmica em pseudocódigo:

```
enquanto <EXPRESSÃO BOOLEANA> faca
        <INSTRUÇÕES>
fimenquanto
```

A sintaxe da estrutura de repetição *while* na linguagem de programação C se dá conforme mostrado no exemplo a seguir:

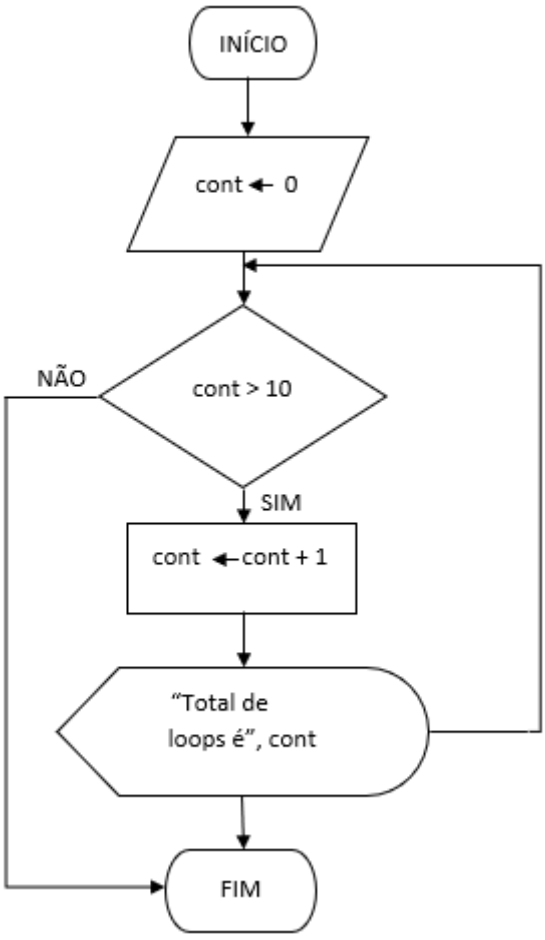
```
while (<condição>)
        {
                <comandos>
        }
```

A seguir, apresentam-se dois exemplos de estrutura de repetição *while* (enquanto) em fluxograma, pseudocódigo e Linguagem C.

1.1 Exemplo 1:

Considere um algoritmo que vai imprimir na tela o número de vezes que foi executado o *loop* usando a estrutura de repetição *while* (enquanto).

1.1.1 Fluxograma



1.1.2 Pseudocódigo

```
algoritmo "Exemplo1"
var
   cont : inteiro

inicio
      cont ← 0

          enquanto cont < 10 faca

                    escreval (“Total de loops é: ”, cont)
                    cont ← cont + 1

          fimenquanto
fimalgoritmo
```

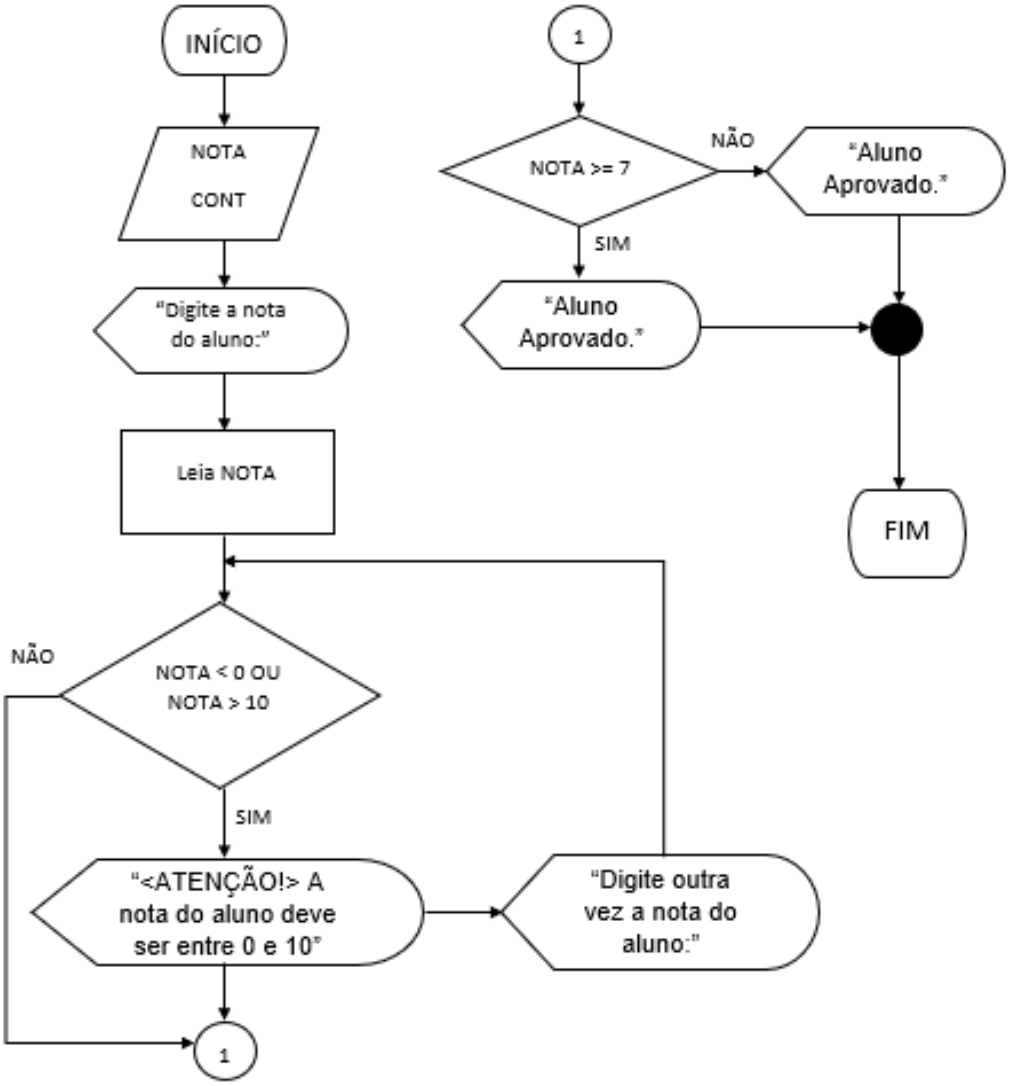
1.1.3 Linguagem C

```
#include <stdio.h>
#include <conio.h>

int main()
{
    int cont = 0; //declarando e inicializando a variável de controle

    while (cont < 10) // Testando a condição
    {
          printf("Total de loops é: %d ", cont); //Executando um comando dentro do
laço

          cont++; //atualizando a variável de controle
    }

    system("pause");
    return(0);
}
```

1.2 Exemplo 2:

Considere um algoritmo que vai pegar um número real, verificar se o usuário digitou um número entre 0 e 10, e imprimir na tela uma mensagem de *aprovado*, caso o aluno tenha nota igual ou acima de 7, ou mensagem de *reprovado*, caso o aluno tenha nota abaixo de 7.

1.2.1 Fluxograma



1.2.2 Pseudocódigo

```
algoritmo "Exemplo2"

var
nota: real
inicio
escreval("Digite a nota do Aluno: ")
leia(nota)
enquanto (nota < 0) ou (nota > 10) faca
   escreval("<ATENÇÃO!> A nota do aluno deve ser entre 0 e 10")
   escreval("Digite outra vez a nota do aluno: ")
   leia(nota)
fimenquanto
se (nota >= 7) entao
   escreval("Aluno aprovado!")
senao
   escreval("Aluno reprovado!")
fimse
fimalgoritmo
```

1.2.3 Linguagem C

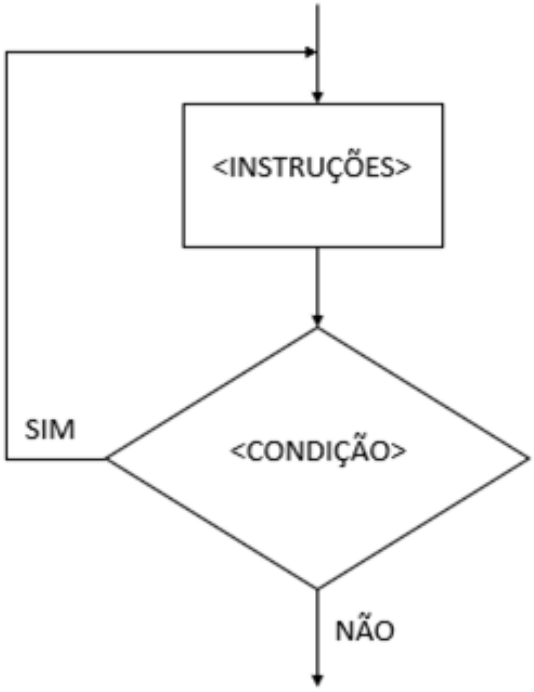
```
#include <stdio.h>
#include <conio.h>
int main()
{
    float nota, cont;

    printf("Digite a nota do Aluno: \n");
    scanf("%f", &nota);

while((nota<0.0) || (nota>10.0))
    {
        printf("<ATENÇÃO!> A nota do aluno deve ser entre 0 e 10\n");
        printf("<Digite outra vez a nota do aluno: \n");
        scanf("%f", &cont);
        nota = cont;
    }
if (nota >= 7.0)
          printf("Aluno Aprovado:\n");
else
          printf("Aluno Reprovado:\n");

    system("pause");
    return(0);
}
```

TEMA 3 – ESTRUTURA DE REPETIÇÃO *DO-WHILE* (REPITA-ATÉ)

A estrutura de repetição *do-while* (repita-até) é bastante similar ao laço *while*. Essa estrutura é utilizada em simulações em que se faz necessário executar o corpo do laço uma vez e depois avaliar a expressão de teste e criar um ciclo repetido (Mizrahi, 2008). Ou seja, temos a garantia de que o laço será executado uma vez.

Vejamos a seguir a sintaxe da estrutura de repetição *while* em fluxograma, pseudocódigo e Linguagem C. A Figura 3 mostra a sintaxe da estrutura de *do-while* (repita-até).

Figura 3 – Estrutura de repetição *do-while*



No pseudocódigo, o *do-while* é o repita-até. O exemplo a seguir mostra a sintaxe da estrutura de repetição *while* na representação algorítmica em pseudocódigo:

```
repita
   <INSTRUÇÕES
ate <EXPRESSÃO BOOLEANA>
```

Para a linguagem de programação C a sintaxe da estrutura de repetição *do-while* começa com a palavra-chave ***do*** seguida de um bloco de uma ou mais instruções entre chaves e terminadas pela palavra-chave ***while***, seguida de uma

expressão de teste entre parênteses terminada por ponto e vírgula. Observe o exemplo a seguir:

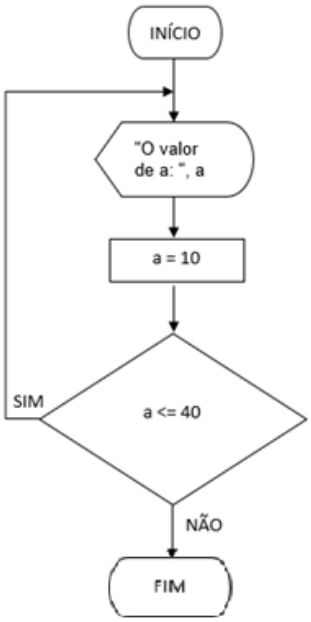
```
do{
        <comandos>
   } while (<condição>);
```

A seguir apresentam-se dois exemplos de estrutura de repetição *do-while* (repita-até) em fluxograma, pseudocódigo e Linguagem C.

3.1 Exemplo 1:

Considere um algoritmo que vai contar de 10 até 40 usando a estrutura de repetição *do-while*.

3.1.1 Fluxograma



3.1.2 Pseudocódigo

```
algoritmo "Exemplo1"
var
   a : inteiro

inicio
      a <- 10
      repita
           escreval ("O valor de a: ", a)
           a <- a+1
         ate ( a <= 40 )
fimalgoritmo
```

Atenção!

- Em alguns programas usados para compilar pseudocódigos a seta ← é substituída pelo **<-** para atribuir um valor à variável.
- Em programas como o **visualg**, para funcionar a condição do algoritmo acima deve-se substituir o **ate (a <= 40)** para **ate (a > 40)**. O laço só se repetirá se a condição for falsa. A lógica das estruturas pode mudar de acordo com a sintaxe da linguagem de programação que você escolher para programar seus códigos.

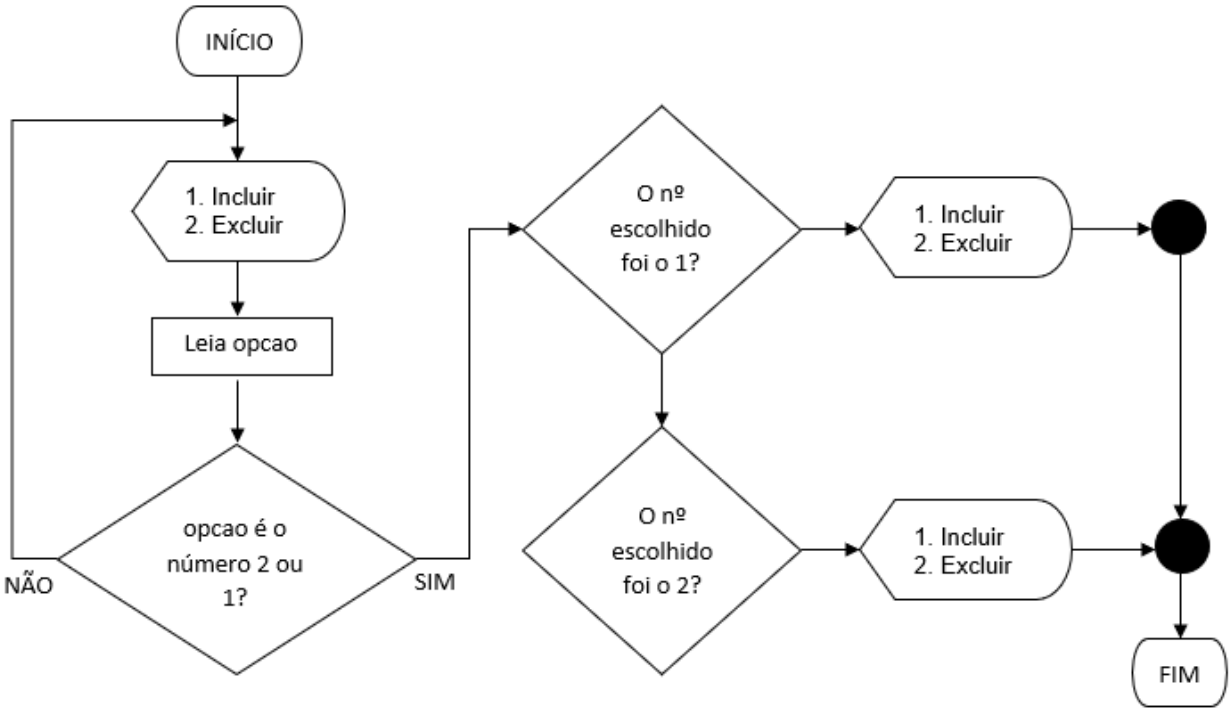
3.1.3 Linguagem C

```
#include <stdio.h>

int main () {
  int a = 10;

  do {
    printf("O valor de a: %d\n", a);
    a = a + 1;
  } while( a <= 40 );

  return 0;
}
```

3.2 Exemplo 2:

Considere um algoritmo que vai criar um menu com duas opções de escolha, 1 para “Incluir”/2 para “Excluir”, e mostra na tela do usuário a opção escolhida. Enquanto o usuário digitar o número diferente de 1 e 2, o menu se repetirá usando a estrutura de repetição *do-while*.

3.2.1 Fluxograma



3.2.2 Pseudocódigo

```
algoritmo "Exemplo2"
var
   opcao: inteiro
inicio
      repita
         escreval ("1. Incluir: ")
                escreval ("2. Excluir: ")
       leia (opcao)
          ate ((opcao <> 1) e (opcao <> 2))
    escolha opcao
         caso 1
           escreval("Escolheu incluir")
         caso 2
           escreval("Escolheu excluir")
      fimescolha
fimalgoritmo
```

Seguindo a mesma explicação do Exemplo 1, para funcionar no **visualg** substitua a condição do algoritmo **ate ((opcao <> 1) e (opcao <> 2))** por **ate ((opcao = 1) ou (opcao = 2))**.

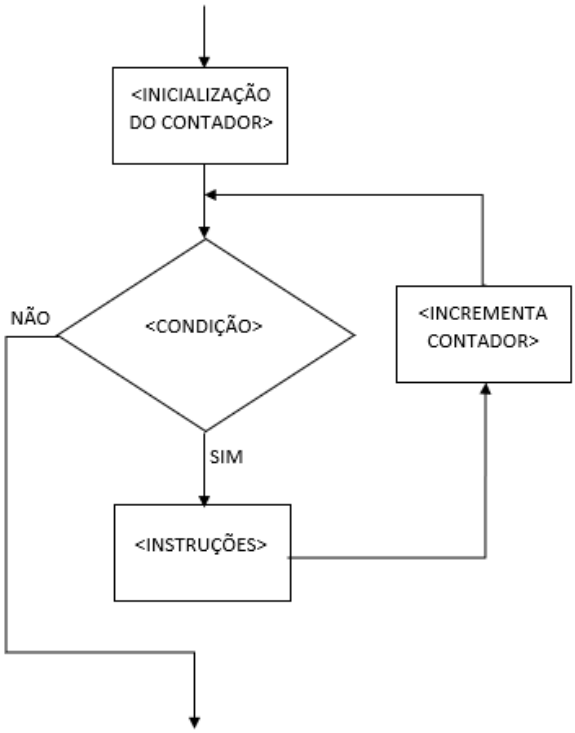
3.2.3 Linguagem C

```
#include <stdio.h>
#include <conio.h>

int main()
{
int opcao; // para exemplificar 1ª execução incondicional do do-while

do {
        printf("1. Incluir\n");
        printf("2. Excluir\n");
        scanf("%d", &opcao);

 } while ( (opcao != 1) && (opcao != 2) );

 opcao = opcao;

 switch(opcao) {
 case 1:
        printf("Escolheu incluir\n");
        break;
 case 2:
 printf("Escolheu excluir\n");
 break;
 }
   system("pause");
   return(0);
}
```

TEMA 4 – ESTRUTURA DE REPETIÇÃO *FOR* (PARA)

A estrutura de repetição *for* (para) é uma estrutura que apresenta no cabeçalho os elementos de inicialização, condição e iteração reunidos na forma e o corpo é organizado em seguida. O laço vai se repetir se o resultado da condição for verdadeiro, conforme mostrado pela representação algorítmica em fluxograma da Figura 4.

Figura 4 – Estrutura de repetição *for*



No pseudocódigo, o *for* é o *para-faca*. O exemplo a seguir mostra a sintaxe da estrutura de repetição *for* (para-faca) na representação algorítmica em pseudocódigo:

```
para <variável> de <início> ate <fim> faca
        <instruções>
fimpara
```

A inicialização da variável contadora é realizada implicitamente, com o valor <início> informado na declaração da estrutura *para*. A iteração é controlada pela condição que determina a quantidade de vezes que será repetida pelo valor <início> e <fim>. No final de cada iteração, o valor da variável contadora é incrementado em 1 (ou o valor declarado como <valor de incremento>).

Do mesmo modo, na linguagem de programação C a estrutura de repetição *for* repete uma sequência de comandos por um determinado número de vezes. A **inicialização**, o **teste** e a **iteração** aparecem entre parênteses após a palavra-chave ***for***, separadas por ponto e vírgula.

O exemplo a seguir mostra a sintaxe da estrutura de repetição *for* na linguagem de programação C.

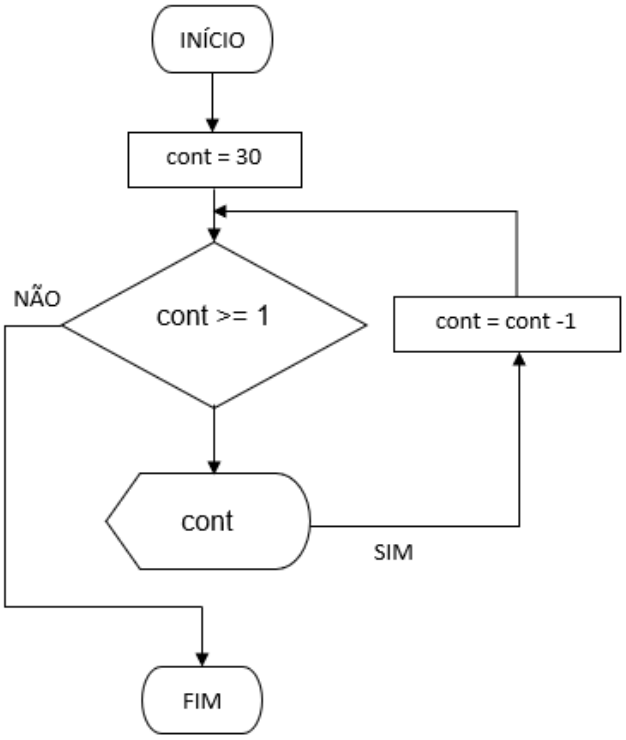
```
for (inicialização <valor inicial>; teste <condição>; interação <incremento>)
{
        <instruções>;
}
```

A seguir dois exemplos de estrutura de repetição *for* (para-faca) em fluxograma, pseudocódigo e Linguagem C:

4.1 Exemplo 1:

Considere um algoritmo que faz a contagem decrescente de 1 até 30 e imprime na tela o resultado.

4.1.1 Fluxograma



4.1.2 Pseudocódigo

```
algoritmo "Exemplo1"

var
cont: inteiro
inicio
        para cont de 30 ate 1 passo -1 faca
                escreval (cont)
        fimpara
fimalgoritmo
```

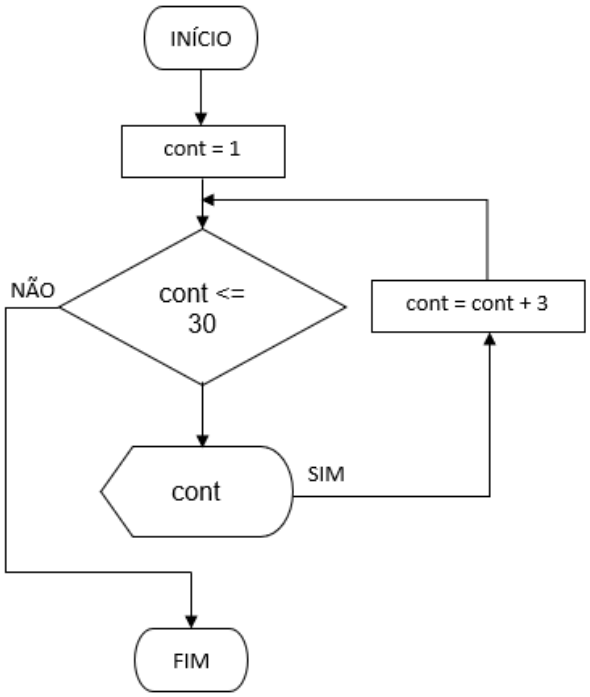
4.1.3 Linguagem C

```
#include <stdio.h>
#include <conio.h>
int main()
{
int cont;
        for (cont = 30; cont >= 1; cont--)
        {
                printf("%d\n", cont);
        }
  system("pause");
  return 0;
}
```

4.2 Exemplo 2:

Considere um algoritmo que faz a contagem crescente de 1 até 30, de 3 em 3, e imprime na tela o resultado.

4.2.1 Fluxograma



A diferença está na atualização, que aumenta o valor de número em três unidades a cada repetição do bloco.

4.2.2 Pseudocódigo

```
algoritmo "Exemplo2"
var

cont: inteiro
inicio

para cont de 1 ate 30 passo 3 faca
   escreval (cont)
fimpara
fimalgoritmo
```

4.2.3 Linguagem C

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int cont;
    for (cont = 1; cont <= 30; cont += 3) {
    printf("%d \n", cont);
}
  system("pause");
  return 0;
}
```

TEMA 5 – ESTRUTURA DE REPETIÇÃO ANINHADA

A Linguagem C permite a utilização de uma estrutura dentro de outra estrutura de repetição. Quando um laço faz parte do corpo de outro laço, dizemos que o laço interno é um laço aninhado (Mizrahi, 2008).

A seguir a sintaxe das três estruturas de repetições (*while, do-while* e *for*) alinhadas.

5.1 Sintaxe while aninhada:

```
while (<condição>)
{
        while (<condição>)
        {
                <comandos>
```

```
        }
 <comandos>
}
```

5.2 Sintaxe do-while aninhada:

```
do
{
        <comandos>

        do{
                <comandos>
            } while (<condição>);

} while (<condição>);
```

5.3 Sintaxe *for* aninhada:

```
for (<valor inicial>; <condição>; <incremento>)
{
        for (<valor inicial>; <condição>; <incremento>)
        {
                <instruções>;
        }
        <instruções>;
}
```

A seguir um exemplo do aninhamento de cada estrutura de repetição na linguagem de programação C.

5.4 Exemplo com while:

```
#include <stdio.h>

int main( ){
        int linha, coluna;
        printf("\n");
        linha = 1;
while (linha < 8)
        {
                printf( "\t" );
                coluna = 1;
                while (coluna < linha)
                {
                        printf( "*" );
                        coluna += 1;
                }
```

```
        printf( "\n" );
        linha += 1;
        }
        system("pause");
return 0;
}
```

O exemplo anterior terá como saída:

```
*
**
***
****
*****
******
*******
********
```

5.5 Exemplo com do-while:

```
#include <stdio.h>

int main ()
{
                int i=0;
                do
                {
                        int j=0;
                        do
                        {
                                printf(" UNINTER ", i,j);
                                j++;
                        } while (j<2);
                        i++;
                        printf("\n");

                } while (i<2);
system("pause");
return 0;
}
```

O exemplo anterior terá como saída:

```
UNINTER  UNINTER
UNINTER  UNINTER
```

5.6 Exemplo com for:

```
#include <stdio.h>

int main ()
{
        int i, j;

        for(i=2; i<10; i++) {
                for(j=2; j <= (i/j); j++)
                if(!(i%j)) break;
        if (j > (i/j)) printf("%d Número primo \n", i);
        }
system (“pause”);
return 0;
}
```

O exemplo anterior terá como saída:

```
2 Número primo
3 Número primo
5 Número primo
7 Número primo
```

FINALIZANDO

Nesta aula aprendemos os principais conceitos que envolvem as estruturas repetição *while* (enquanto), *do-while* (repita-até) e *for* (para), e como utilizá-los nas representações algorítmicas – fluxograma, pseudocódigo e linguagem de programação C. Também tivemos uma introdução sobre as estruturas de repetições aninhadas, muito utilizadas nas construções de algoritmos quando precisamos testar algo dentro de um outro teste. Portanto, aproveite a disciplina e bons estudos.

REFERÊNCIAS

GUEDES, S. **Lógica de programação algorítmica**. São Paulo: Pearson, 2014.

MIZRAHI, V. V. **Treinamento em linguagem C**. 2. ed. São Paulo: Pearson, 2008.

PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados**. São Paulo: Pearson, 2016.