UNIPAC

Universidade Presidente Antônio Carlos

Bacharelado em Ciência da Computação

Introdução a Programação

Material de Apoio

Parte V – Estruturas de Repetição

Prof. Nairon Neri Silva

naironsilva@unipac.br

1º sem / 2020

Material cedido pela professora Lívia

Estruturas de Repetição

- Existem situações que o processamento de um valor uma única vez não é suficiente para resolver o problema. Quando isto ocorre, deve-se utilizar uma *estrutura de repetição*.

- Estruturas de repetição permitem que uma ou mais instruções sejam executadas um número definido de vezes, ou enquanto uma determinada condição não é alcançada. Os algoritmos passam a ter iterações.

Estruturas de Repetição

- Exemplo: verificar se o aluno foi aprovado ou reprovado de acordo com sua nota final. Resolver este problema para 50 alunos de uma turma.
- Percebemos que ler as notas de 50 alunos para verificar se o mesmo foi aprovado ou reprovado, não é uma tarefa viável se feita de modo sequencial, não utilizando-se de estruturas de repetição.

Estruturas de Repetição

- Para resolver de modo eficiente, um trecho do algoritmo (ler a nota e verificar se aprovado ou não) pode ser repetido uma determinada quantidade de vezes, ou seja, ser realizado um retrocesso no algoritmo. No exemplo acima, corresponderia a repetir o mesmo trecho 50 vezes, sem, no entanto, ter de escrevê-lo 50 vezes.

Estruturas de Repetição

- Os trechos do algoritmo que são repetidos denominam-se laços ou *loops* de repetição. O número de repetição pode ser indeterminado, porém *necessariamente finito*.

- Cuidado fundamental que o programador deve ter é o de certificar-se que a condição, em algum momento, deve ser falsa, para que o algoritmo não entre em laço infinito.

Estruturas de Repetição

- As estruturas de repetição das linguagens de programação são de dois tipos
  - Condicional
    - Repetem até satisfazer a condição de repetição
    - São usadas quando não se sabe previamente quantas vezes deve-se executar as instruções do bloco de repetição
  - Contada
    - Repetem um número contado (pré-definido) de vezes
    - São usadas quando se sabe previamente quantas vezes deve-se executar as instruções do bloco de repetição

Estruturas de Repetição

- As estruturas de repetição condicional podem executar seus testes:
  - No Início
    - while
  - No Fim
    - do ... while
- As estruturas de repetição contada utilizam uma variável para controlar a quantidade de repetições
    - for

Estruturas de Repetição - WHILE

- Consiste em uma estrutura de controle do fluxo de execução que permite diversas vezes um mesmo trecho do algoritmo, porém, sempre verificando antes de cada execução se é “permitido” executar o mesmo trecho.
- Para realizar a repetição com teste no início, utilizamos a estrutura ***while,*** que permite que um bloco ou uma ação primitiva seja repetida enquanto uma determinada <condição> for verdadeira.

Estruturas de Repetição – WHILE

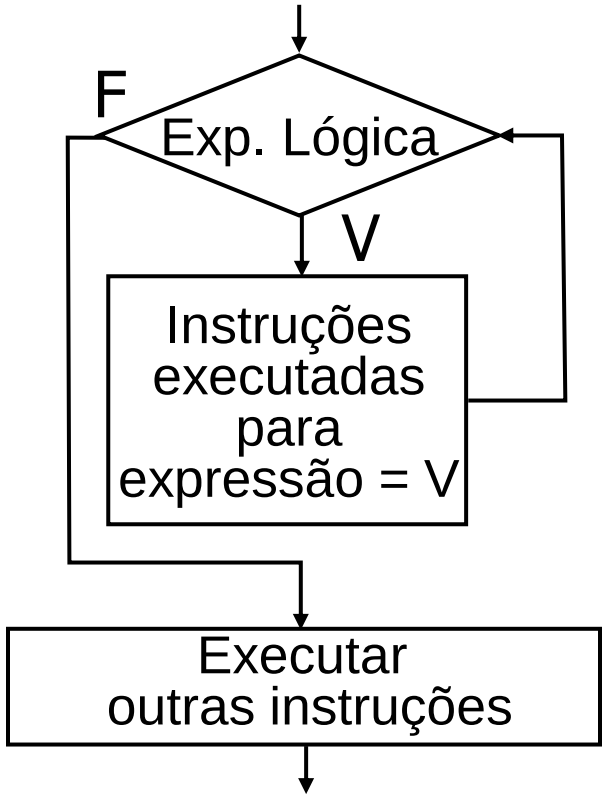
- Sintaxe:

```
while (<condição>)
 {
    <comando(s)>
 }
```

- Onde:
  - <condição> é uma expressão lógica ou relacional.
  - <comando(s)> são quaisquer comandos válidos, que serão executados enquanto a condição for verdadeira.

Estruturas de Repetição - WHILE

- Semântica:
  - Faz o teste no início do laço. Se o resultado for V as instruções do laço e/são executadas. Volta-se para o início do laço e testa-se novamente a sua condição. Isto é repetido enquanto a condição testada for V.



Estruturas de Repetição - WHILE

- Observações importantes:
  - Não se sabe de antemão quantas vezes o bloco de repetição será executado. Isto é, ele pode ser executado várias vezes ou nenhuma vez.
  - Testa a condição antes de entrar na estrutura de repetição.
  - Repete a execução do bloco de instruções toda vez que condição for verdadeira.
  - A execução do bloco é finalizada quando a condição for falsa.
  - Pode-se ter qualquer comando dentro de uma estrutura de repetição, inclusive, outra(s) estrutura(s) de repetição.

Estruturas de Repetição - WHILE

Algoritmo para verificar se um aluno foi aprovado ou reprovado, de acordo com a sua média (4 notas). Turma de 50 alunos.

```
int main ()
{
     float n1, n2, n3, n4, media;
     int cont;     //contador
     cont = 0;
     while (cont < 50)
        {
         scanf("%d%d%d%d", &n1, &n2, &n3, &n4);
         media = (n1+n2+n3+n4)/4;
         if (media >= 6)
           printf("Aluno aprovado!");
         else
           printf("Aluno reprovado!");
         cont = cont + 1;   //incrementando o contador
        }
     return 0;
 }
```

Estruturas de Repetição - WHILE

```
int main ()
{
      float n1, n2, n3, n4, media;
      int cont;     //contador
      cont = 0;
      while (cont < 50)
        {
          scanf(“%d%d%d%d”, &n1, &n2, &n3, &n4);
          media = (n1+n2+n3+n4)/4
          if (media >= 6)
             printf("Aluno aprovado!")
          else
             printf("Aluno reprovado!")
        }
      return 0;
}
```

// ERRO: LOOP INFINITO – A VARIÁVEL CONT SEMPRE TERÁ VALOR IGUAL A 0 (ZERO), OU SEJA, A CONDIÇÃO (CONT<50) SEMPRE SERÁ VERDADEIRA!

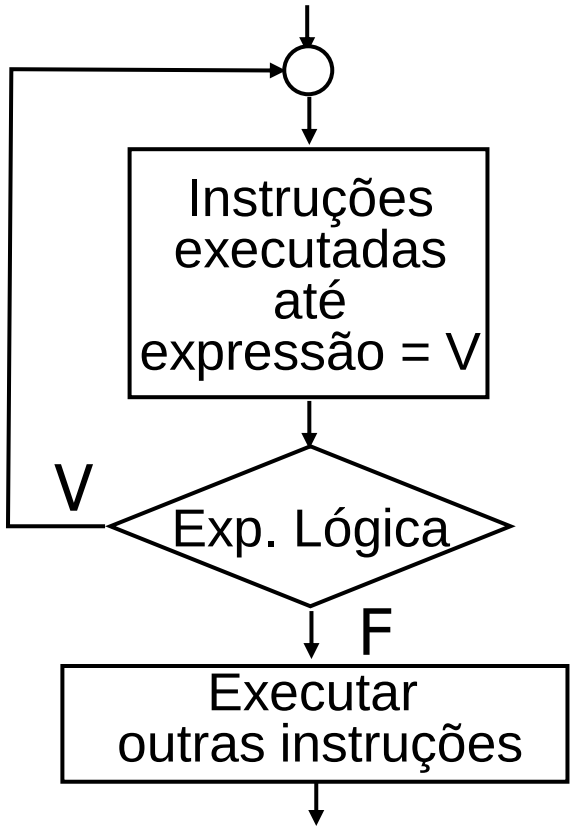
Estruturas de Repetição – DO .. WHILE

- Para realizar a repetição com teste no final, utilizamos a estrutura do .. while que permite que um bloco ou ação primitiva seja repetido enquanto uma determinada condição seja verdadeira.
- Sintaxe:

```
do
 {
    <comando(s)>
 }while (<condição>)
```

Estruturas de Repetição – DO .. WHILE

- Semântica:
  - Efetua um teste lógico no fim do laço, garantindo que pelo menos uma vez as instruções deste são executadas.



Estruturas de Repetição – DO .. WHILE

- Observações importantes:
  - Não se sabe de antemão quantas vezes o bloco de repetição será executado. Todavia é garantido que ele será executado pelo menos uma vez.
  - Testa a condição depois de entrar na estrutura de repetição.
  - Repete a execução do bloco de instruções toda vez que a condição for V.
  - A execução do bloco é finalizada quando a condição for F.

Estruturas de Repetição – DO .. WHILE

```
int main ()
{
   int x;
   do {
        printf("Digite o valor de x ou 0 para sair: ");
        scanf("%d", &x);
        if (x > 0){
             printf("Positivo \n");
        }
        else if (x < 0){
             printf("Negativo\n");
        }
   }while (x != 0);
   return 0;
}
```

Estruturas de Repetição – FOR

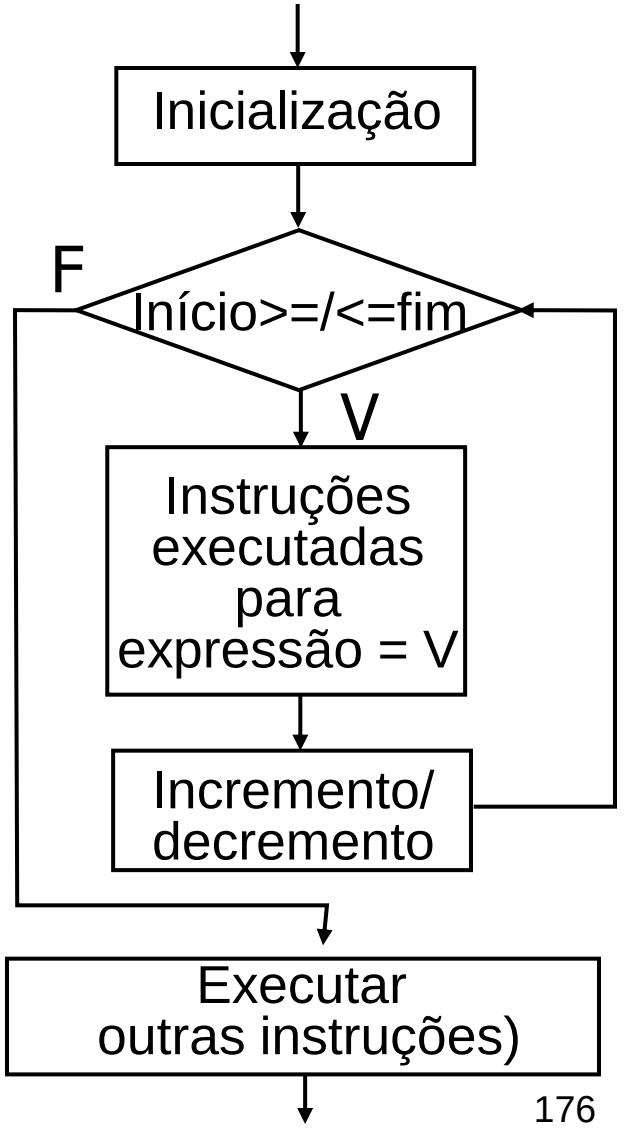
- A estrutura `for` repete a execução do bloco um número definido de vezes, pois ela possui limites fixos.

- Sintaxe:

```
for (inicialização; condição; incremento)
{
   <comando(s)>
}
```

Estruturas de Repetição – FOR

- Semântica:
  - Repete as instruções enquanto a variável contador não atingir o valor final. Ressalta-se que a variável contador é previamente inicializada e incrementada ou decrementada de uma constante a cada repetição.



Estruturas de Repetição – FOR

- Observações importantes:
  - Sabe-se de antemão quantas vezes o bloco de repetição será executado. Isto é, repete enquanto o valor final não atingir o valor final da variável de controle.
  - A variável de controle deve ser um número inteiro e não deve ser modificada dentro do bloco.
  - A estrutura **for** é mais completa que as anteriores, pois ela incorpora a inicialização, incremento e teste de valor final da variável de controle. É preferencialmente utilizada em situações em que sabe-se previamente o número de repetições a serem feitas. Este número de repetições pode ser uma constante ou estar em uma variável.

Estruturas de Repetição – FOR

```
int main()
{
   int i, numero;
  for (i = 1; i < 5; i++)
   {
        scanf("%d",&numero);
        if (numero > 0)
              printf("Positivo");

   }
  return 0;
}
```

Estruturas de Repetição – FOR

```
int main()
{
   int i, n;
   scanf("%d",&n);
   for(i=1; i<n; i++)
       printf("%d",i);
   return 0;
}
```

Algoritmo que lê um número N e escreve todos os números naturais de 1 a N.

Exemplo

```
int main()
{
   int r, n1, n2;
   n1 = 1;
   while (n1 <= 10)
     {
       for(n2=1; n2<=10; n2++)
         {
          r = n1 + n2;
           printf("%d + %d = %d\n",n1,n2,r)
         }
        n1++;
      }
    return 0;
}
```