Universidade Positivo

Algoritmos de Programação

Aula 12

Prof.ª Mariane Cassenote

mariane.cassenote@up.edu.br

Conteúdo

- Estrutura de repetição **`enquanto/while`**
- Estrutura de repetição **`faca…enquanto/do…while`**

Ao final do componente curricular será possível utilizar estruturas de repetição simples e aninhadas para resolver problemas algorítmicos.

Estruturas de Repetição

Estruturas de Repetição

Para se resolver um problema algorítmico com estruturas de repetição deve-se:

- reconhecer o **padrão repetitivo** (ações que se repetem)
- definir o **controle** da repetição (contador ou decisão do usuário)
- estabelecer o critério de **parada** (condição)

O número de repetições pode ser previamente conhecido ou estar associado à ocorrência de uma condição específica que é testada no decorrer do processamento

EDELWEISS, Nina. Algoritmos e programação com exemplos em Pascal e C. Porto Alegre Bookman 2014 1 recurso on-line (Livros didáticos UFRGS 23).

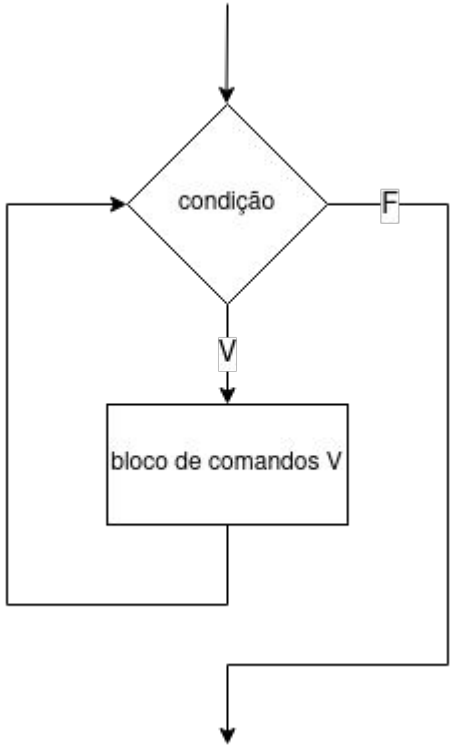
Estruturas de Repetição

`enquanto`

`while`

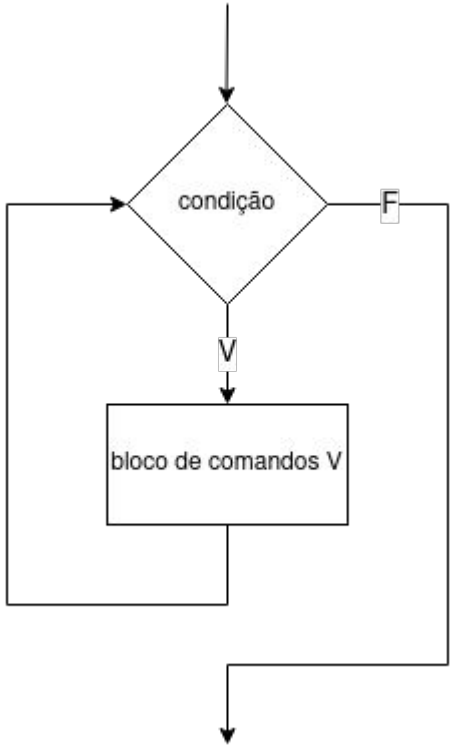
Estruturas de Repetição – enquanto...faça / while

- **Primeiro testa a condição**, depois executa o bloco de comandos
- A execução de um bloco de comandos é repetida **enquanto** a avaliação da condição tiver resultado **VERDADEIRO**
- Quando a condição passa a ser FALSA, a repetição é **encerrada**



Estruturas de Repetição – enquanto...faça / while

- O bloco de comandos **nunca será executado** se a condição for **FALSA** logo de início
- Se a condição for VERDADEIRA, é necessário que alguma **alteração** ocorra durante a execução do bloco de comandos para que em algum momento a condição seja avaliada como FALSA. Caso contrário, se tem um **laço (loop) infinito**



Cuidado com os loops infinitos!

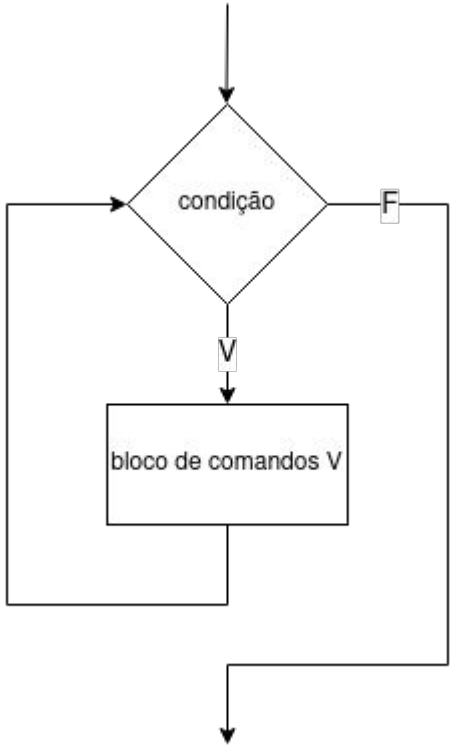
Estruturas de Repetição – enquanto...faça / while

Em pseudocódigo:

```
enquanto (<condição>) {
    <bloco de comandos>
}
```

Em linguagem C:

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



Estruturas de Repetição – enquanto...faça / while

Construir um programa que apresente a soma dos cem primeiros números naturais (1+2+3+ ... +98+99+100).

Estruturas de Repetição – enquanto...faça / while

```c
// EM LINGUAGEM C – Apresente a soma dos cem primeiros números naturais
#include <stdio.h>

int main() {
    int i, soma;

    i = 1;
    soma = 0;

    while (i <= 100) {
        soma = soma + i;
        i = i + 1;
    }

    printf("soma: %d", soma);

    return 0;
}
```

```
> ./main
soma: 5050>
```

Estruturas de Repetição – enquanto...faça / while

```
// EM PSEUDOCÓDIGO – Apresente a soma dos cem primeiros números naturais
programa {
    funcao inicio() {

        inteiro i, soma

        i = 1
        soma = 0

        enquanto (i <= 100) {
            soma = soma + i
            i = i + 1
        }

        escreva("soma:", soma);

    }
}
```

```
soma:5050
Programa finalizado.
```

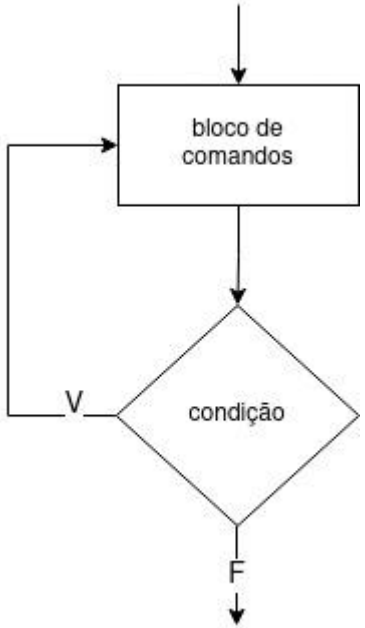
Estruturas de Repetição

`faca...enquanto`

`do...while`

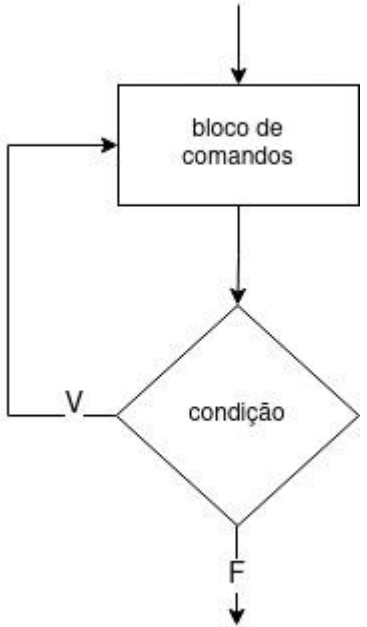
Estruturas de Repetição – faca...enquanto / do...while

- **Primeiro executa o bloco de comandos**, depois testa a condição
- **O bloco de comandos é executado no mínimo uma vez**, independentemente da avaliação da condição
- A execução de um bloco de comandos é repetida **enquanto** a avaliação da condição tiver resultado **VERDADEIRO**
- Quando a avaliação da condição se torna FALSO, a repetição é encerrada



Estruturas de Repetição – faca...enquanto / do...while

- O bloco de comandos será executado mesmo se a condição for **FALSA logo de início**
- Se a condição for VERDADEIRA, é necessário que alguma **alteração** ocorra durante a execução do bloco de comandos para que em algum momento a condição seja avaliada como FALSA. Caso contrário, se tem um **laço (loop) infinito**



Cuidado com os loops infinitos!

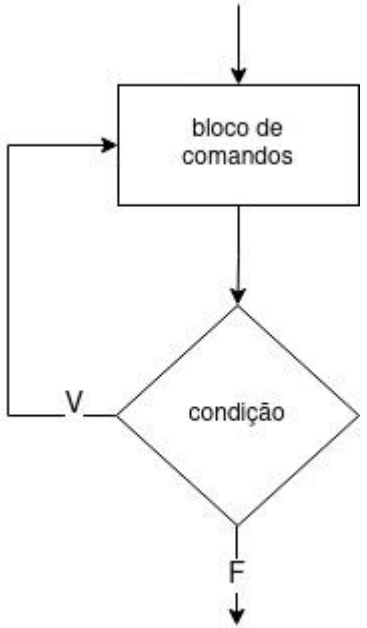
Estruturas de Repetição – faca...enquanto / do...while

Em pseudocódigo:

```
faca {
    <bloco de comandos>
} enquanto (<condição>)
```

Em linguagem C:

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



Estruturas de Repetição – faca...enquanto / do...while

Construir um programa que leia valores inteiros do teclado até que o produto (multiplicação) desses valores ultrapasse 100.

Estruturas de Repetição – faca...enquanto / do...while

```c
// EM LINGUAGEM C – Ler valores até que o produto deles seja maior que 100
#include <stdio.h>

int main() {
     int valor, produto;

     produto = 1;

     do {
          printf("Digite um valor: ");
          scanf("%d", &valor);

          produto = produto * valor;

          printf("Produto parcial: %d \n \n", produto);

     } while (produto <= 100);

     printf("O produto e: %d", produto);
     return 0;
}
```

```
> ./main
Digite um valor: 105
Produto parcial: 105

O produto e: 105>
```

```
> ./main
Digite um valor: 4
Produto parcial: 4

Digite um valor: 7
Produto parcial: 28

Digite um valor: 1
Produto parcial: 28

Digite um valor: 2
Produto parcial: 56

Digite um valor: 6
Produto parcial: 336

O produto e: 336>
```

Estruturas de Repetição – faca...enquanto / do...while

```c
// EM LINGUAGEM C – Ler valores até que o produto deles seja maior que 100
#include <stdio.h>

int main() {
     int valor, produto;

     produto = 1;

     do {
          printf("Digite um valor: ")
          scanf("%d", &valor);

          produto = produto * valor;

          printf("Produto parcial: %d \n \n", produto);

     } while (produto <= 100);

     printf("O produto e: %d", produto);
     return 0;
}
```

O que aconteceria se inicializássemos “produto” como 0 (zero)?

Loop infinito!

```
> ./main
Digite um valor: 105
Produto parcial: 105

O produto e: 105>
```

```
> ./main
Digite um valor: 4
Produto parcial: 4

Digite um valor: 7
Produto parcial: 28

Digite um valor: 1
Produto parcial: 28

Digite um valor: 2
Produto parcial: 56

Digite um valor: 6
Produto parcial: 336

O produto e: 336>
```

Estruturas de Repetição – faca...enquanto / do...while

```
{EM PSEUDOCÓDIGO – Ler valores até que o produto deles seja maior que 100}
programa {
     funcao inicio() {

          inteiro valor, produto

          produto = 1

          faca {

               escreva("Digite um valor: ")
               leia(valor)

               produto = produto * valor

               escreva("Produto parcial:", produto, "\n")

          } enquanto (produto < 100)

          escreva("O produto e: ", produto)

     }
}
```

```
Digite um valor: 105
Produto parcial:105
O produto e: 105
Programa finalizado.
```

```
Digite um valor: -9
Produto parcial:-9
Digite um valor: 6
Produto parcial:-54
Digite um valor: 2
Produto parcial:-108
Digite um valor: -8
Produto parcial:864
O produto e: 864
Programa finalizado.
```

Estruturas de Repetição – Síntese

Em pseudocódigo:

```
enquanto (<condição>) {
    <bloco de comandos>
}
```

Em linguagem C:

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

Em pseudocódigo:

```
faca {
    <bloco de comandos>
} enquanto (<condição>)
```

Em linguagem C:

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

Estruturas de Repetição

Ao analisarmos o que ocorre nos laços de repetição estudados até agora, percebemos que:

- Ocorre a **inicialização de uma variável** envolvida na condição que controla o número de repetições
- A **condição que envolve essa variável é testada** (antes ou depois da execução do bloco de comandos da repetição, dependendo da estrutura utilizada)
- Dentro do laço de repetição ocorre uma **atualização no valor dessa variável** de forma a prever o encerramento do laço em algum momento (evitando loop infinito)

Para praticar

1. Elaborar um programa que apresente todos os valores numéricos inteiros ímpares situados no intervalo de 0 a 20.
2. Escrever um algoritmo que leia uma quantidade desconhecida de números e conte quantos deles estão nos seguintes intervalos: [0,25], [26,50], [51,75] e [76,100]. A entrada de dados deve terminar quando for lido um número negativo.
3. Escreva um programa que leia uma sequência de números inteiros positivos e negativos, e imprima a soma dos números positivos e a soma dos números negativos.

Para praticar

4. Escreva um programa que leia números inteiros positivos até que o número 0 seja digitado e imprima a média desses números.

5. Escreva um programa que leia dois números inteiros e calcule o máximo divisor comum (MDC) desses números.

6. Escreva um programa que leia uma sequência de números inteiros positivos e imprima o menor e o maior número lido

Universidade Positivo

Algoritmos de Programação

Aula 12

Prof.ª Mariane Cassenote

mariane.cassenote@up.edu.br