Fundamentos de Programação

# Estruturas de REPETIÇÃO

## O Comando *While*

Aula 12

# Post It da Aula Passada

**Intuição de Repetição**

- Rotina
  - Tarefas que se repetem
- Parada
- Iteração

**O Comando *For***

- Estrutura genérica do comando
- Variável de controle
- Espaço de iteração
  - *range*
- Indentação

# O Comando *While*

O segundo comando de repetição que temos é o chamado *while*

Diferente do *for*, que naturalmente trabalha com um espaço de iteração definido, o *while* visa satisfazer um critério de parada

Na prática, **o critério de parada nada mais é que um teste condicional** (igual àqueles que vimos quando estudamos o *if*), ou seja:

- O resultado deve ser verdadeiro (*True*) ou falso (*False*)

# O Comando *While*

Sendo assim, **enquanto o resultado do teste for verdadeiro, o bloco de operações é executado**; já quando o teste for falso, a repetição da execução do bloco de comandos se encerra

A “cara” da estrutura de repetição *while* é:

**while** **teste condicional****:
    **comandos executados a cada iteração**

# O Comando *While*

**while** **teste condicional****:
    **comandos executados a cada iteração**


****teste condicional****: teste que deve sempre resultar em um dado lógico, ou seja, ou é verdadeiro (*True*) ou falso (*False*)
****comandos executados a cada iteração****: bloco que comandos que será executado para cada iteração

# O Comando *While*

**while** **teste condicional**:
    **comandos executados a cada iteração**

Podemos “ler” uma estrutura de repetição *while* da seguinte forma: **“enquanto o teste condicional for verdadeiro, execute os comandos do bloco de iteração, refazendo o teste condicional após cada iteração”**

Ou seja, no primeiro momento que o teste condicional for falso, a estrutura de repetição será encerrada

# O Comando *While*

Vamos considerar um exemplo simples para colocarmos o comando *while* em prática!

Receba um número inteiro pela entrada padrão. **Enquanto o valor da variável diferir de zero, decremente (se valor positivo) ou incremente (se valor negativo) uma unidade** na mesma, exiba o valor da variável a cada iteração

# O Comando *While*

“Receba um número inteiro pela entrada padrão”

```
var_num = int(input(“Digite um número inteiro:”))
```

“Enquanto o valor da variável diferir de zero”

```
var_num = int(input(“Digite um número inteiro:”))
while var_num != 0:
```

# O Comando *While*

“decremente (se valor positivo) ou incremente (se valor negativo) uma unidade na mesma”

```
var_num = int(input(“Digite um número inteiro:”))
while var_num != 0:
    if var_num > 0:
        var_num = var_num - 1
    else:
        var_num = var_num + 1
```

# O Comando *While*

“exiba o valor da variável a cada iteração”

```
var_num = int(input(“Digite um número inteiro:”))
while var_num != 0:
    if var_num > 0:
        var_num = var_num - 1
    else:
        var_num = var_num + 1
    print(var_num)
```

# O Comando *While*

O que acontece se o usuário digitar “3”?

```
var_num = int(input(“Digite um número inteiro:”))
while var_num != 0:
    if var_num > 0:
        var_num = var_num - 1
    else:
        var_num = var_num + 1
    print(var_num)
```

# O Comando *While*

- **var_num != 0? (3 != 0?)**
  - var_num = var_num - 1
  - print(var_num)
- **var_num != 0? (2 != 0?)**
  - var_num = var_num - 1
  - print(var_num)
- **var_num != 0? (1 != 0?)**
  - var_num = var_num - 1
  - print(var_num)
- **var_num != 0? (0 != 0?)**

# O Comando *While*

Algumas vezes podemos querer verificar o **teste condicional após a execução parcial** de uma iteração

Para fazermos isso, podemos **definir uma repetição garantidamente infinita** através do comando *while*:

**while** True**:**
    **comandos executados a cada iteração**

# O Comando *Break*

Nesse caso, **precisamos definir o teste condicional como parte do bloco de comandos** (geralmente por um condicional — *if*) e, para quando ele for satisfeito, utilizamos o comando *break* para forçar a parada da estrutura de repetição

```
while True:
    ...
    if **teste condicional**:
        break
    ...
```

# O Comando *Break*

Para este caso, vamos considerar o seguinte exemplo:

Requisite um número natural para o usuário através da entrada padrão e o armazene em uma variável. **Enquanto o valor digitado não for um número inteiro, continue requisitando um número ao usuário.** Quando os critérios forem satisfeitos, exiba o número aceito na tela

# Repetições Aninhadas

Naturalmente, como acontece para os condicionais, **podemos aninhar estruturas de repetição** (definir estruturas de repetição em uma previamente existente)

Todas as regras e conceitos apresentados até aqui valem da mesma forma para quando uma estrutura de repetição está aninhada em outra, porém temos que tomar muito cuidado com a...

**INDENTAÇÃO**

# Repetições Aninhadas

Vamos analisar o seguinte exemplo relacionado à definição de estruturas de repetição aninhadas:

**Faça um servidor** que requisite um dado qualquer a partir da entrada padrão. Para cada um dos dados fornecidos, **conte a quantidade de caracteres “a”** existentes. Por fim, exiba o resultado da contagem na tela a cada iteração

# Repetições Aninhadas

Como temos um servidor, podemos inferir que o **primeiro laço será infinito**! No contexto deste primeiro laço, devemos receber um dado da entrada padrão (*input*). Para contarmos a quantidade de caracteres “a”, podemos **aninhar um segundo laço ao laço infinito**, mas este iterando sobre cada um dos caracteres existentes na *string* provida pela função *input*. Para cada iteração desse segundo laço, **verificaremos (com um condicional) se o caractere corresponde a letra “a”**, se sim, somamos uma unidade a um contador previamente definido (antes de declarar o segundo laço). Após a conclusão do segundo laço, exibimos o valor do contador na tela através da função *print*

# Repetições Aninhadas

```
while True:
    var_str = input(“Digite algo: ”)
    contador = 0
    for caractere in var_str:
        if caractere == “a”:
            contador = contador + 1
    print(“O número de “a”s é:”, contador)
```

# Exercício #12

Considere o método iterativo de Newton para solucionar raízes quadradas:

$$y_{i+1} = (y_i + (a/y_i)) / 2$$

Faça um programa que receba da entrada padrão um **erro e entre 0 e 1** (será critério de parada da estrutura de repetição – y(i+1) - yi <= erro), defina **um chute inicial para a variável $y_i$** (sugiro utilizar a/2) e calcule uma aproximação da **raiz da quadrada ($y_{i+1}$) de um número a**, também recebido pela entrada padrão

Fundamentos de Programação

Aula 12

# Obrigado e ATÉ A PRÓXIMA!