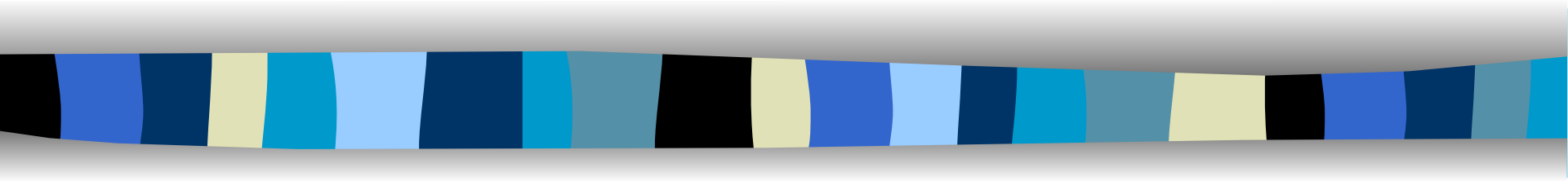
# Estruturas de Controle



**João Frederico Roldan Viana**

jfredrv@gmail.com

(85)99231.2777

# Agenda

- Estruturas de Controle
  - Estruturas de Repetição
    - for
    - while
    - do-while
  - Comandos de Desvio
    - break
    - continue
    - return

# Estruturas de Repetição

- *for*
  - Permite que trechos de programa sejam repetidos um certo número de vezes controlado pelo programa.
  - Sintaxe:

    *for (expressão1; expressão2; expressão3)*
    *bloco_de_comandos;*

  - Expressão1: inicialização da variável de controle.
  - Expressão2: teste que controla o fim do laço ou repetição.
  - Expressão3: incremento ou decremento da variável de controle.

# Estruturas de Repetição

- *for*
  - Funcionamento:
    - Passo1: Uma variável de controle, geralmente um contador, recebe um valor inicial.
    - Passo 2: O valor da variável de controle é comparada com o valor final que ela deve alcançar. Caso a condição de término tenha sido atingida (expressão2 = 0), o laço é interrompido. Caso contrário (expressão2 $\neq$ 0), vai para o passo3.
    - Passo3: O trecho de programa que pertence ao laço é executado.
    - Passo4: A variável de controle é incrementada ou decrementada e volta ao passo 2.

# Estruturas de Repetição

- *for*

**Exemplo**

```
fatorial = 1;
for (i = numero; i > 1; i--) {
    fatorial = fatorial * i;
}
```

# Estruturas de Repetição

- *for*
  - Variações:
    - Com mais de uma variável:

```
for (x = 0, y = 0; x + y < 10; x++, y++)
```

    - Sem uma expressão:

```
for (i = 0; i != -1; )
```

```
for ( ; i < 100; i++)
```

    - Laço Infinito:

```
for ( ; ; )
```

      Obs: No laço infinito, o programa pára quando o comando *break* é executado.

# Estruturas de Repetição

- *while*
  - Permite que trechos de programa sejam repetidos um certo número de vezes controlado pelo programa.
  - Sintaxe:

    *while (expressão)*

    *bloco_de_comandos;*

  - Funcionamento:
    - Passo1: A expressão é avalidada.
    - Passo2: Caso o resultado seja verdadeiro ($\neq 0$), o bloco de comandos é executado e volta ao passo 1. Caso contrário (= 0), o laço é terminado.

# Estruturas de Repetição

- *while*

**Exemplo**

```
fatorial = 1;
i = numero;
while (i > 1) {
    fatorial = fatorial * i;
    i--;
}
```

# Estruturas de Repetição

- *do-while*
  - Permite que trechos de programa sejam repetidos um certo número de vezes controlado pelo programa, porém, neste caso, o bloco de comandos é executado pelo menos uma vez, já que a expressão de teste fica após a execução dos comandos.
  - Sintaxe:

```
do
    bloco_de_comandos;
while (expressão);
```

# Estruturas de Repetição

- *do-while*
  - Funcionamento:
    - Passo1: Executa o bloco de comandos.
    - Passo2: Avalia a expressão.
    - Passo3: Caso o resultado da expressão seja verdadeiro ($\neq 0$), volta ao passo 1. Caso contrário ($= 0$), interrompe o *do-while*.

# Estruturas de Repetição

- *do-while*

**Exemplo**

```
i = 1;
do {
    printf (“Numero %d\n”, i);
    i++;
} while (i <= 100);
```

# Comandos de Desvio

- *break*
  - Pode ser usado tanto para terminar um teste *case* dentro de um comando *switch* quanto interromper a execução de um laço.
  - Quando o comando é utilizado dentro de um comando *for*, por exemplo, o laço é imediatamente interrompido e o programa continua a execução no comando seguinte ao comando *for*.

# Comandos de Desvio

- *break*

**Exemplo**

```
i = 1;
for (i = 0; i < 100; i++) {
    scanf (“%d”, &num);
    if (num < 0)
        break;
    vetor[i] = num;
}
```

# Comandos de Desvio

- *continue*
  - O comando continue é parecido com o comando break. A diferença é que o comando continue simplesmente interrompe a execução da iteração corrente passando para a próxima iteração do laço, se houver uma.

# Comandos de Desvio

- *continue*

**Exemplo**

```
i = 1;
for (i = 0; i < 100; i++) {
    scanf (“%d”, &num);
    if (num < 0)
        continue;
    vetor[i] = num;
}
```

# Comandos de Desvio

- *return*
  - Usado para interromper a execução de uma função e retornar um valor ao programa que chamou esta função.
  - Caso haja algum valor associado ao comando *return* este é devolvido para a função, caso contrário um valor qualquer é retornado.
  - Uma função declarada como do tipo *void* não pode ter um comando *return*.
  - Sintaxe:

    *return expressão;*

    Obs: Expressão é opcional