Algoritmos e Programação II

https://evandro-crr.github.io/alg2

Evandro Chagas Ribeiro da Rosa

Revisão de Programação I

- **Expressão**: combinação de literais, variáveis e operadores que geram um valor. Ex.:
  - `2 + 5 * (x + 2)`
  - `pi * r * r`
  - `10 * (1.0 / 2.0)`
- **Instrução**: linha de código que realiza uma ação. Ex.:
  - `int a = 10;`
  - `x = 2 * a + 1;`
  - `if (...) ...`
  - `for (int i ...) ...`

Expressão

Convertendo expressão algébrica para código

Expressão Algébrica	C++ Equivalente
$6x$	`6 * x`
$4xy$	`4 * x * y`
$\frac{a+b}{c}$	`(a + b) / c`
$3\frac{x}{2}$	`3 * (x / 2)`
$3bc + 4$	`3 * b * c + 4`
$\frac{2x+3}{4a-1}$	`(2 * x + 3) / (4 * a - 1)`

Expressões com Tipos Diferentes

Ranking dos tipos

- `long double`
- `double`
- `float`
- `unsigned long`
- `long`
- `unsigned int`
- `int`

- Em uma operação com tipos diferentes, o valor de menor ranking será promovido antes da operação.
- Em uma atribuição, o valor final será convertido para o tipo da variável sendo atribuída.

Expressões com Tipos Diferentes

```
int x, y = 4;
float z = 2.7;
x = y * z;
```

Qual o valor de `x`?

Instruções de desvio condicional

```
if (teste) {
  instrução;
  //...
}
```

```
if (teste) {
  instrução;
  //...
} else {
  instrução;
  //...
}
```

- `teste` é uma expressão, normalmente usando operadores relacionais:
  - `x > y` maior que
  - `x < y` menor que
  - `x >= y` maior ou igual
  - `x <= y` menor ou igual
  - `x == y` igual
  - `x != y` diferente

Instrução `switch-case`

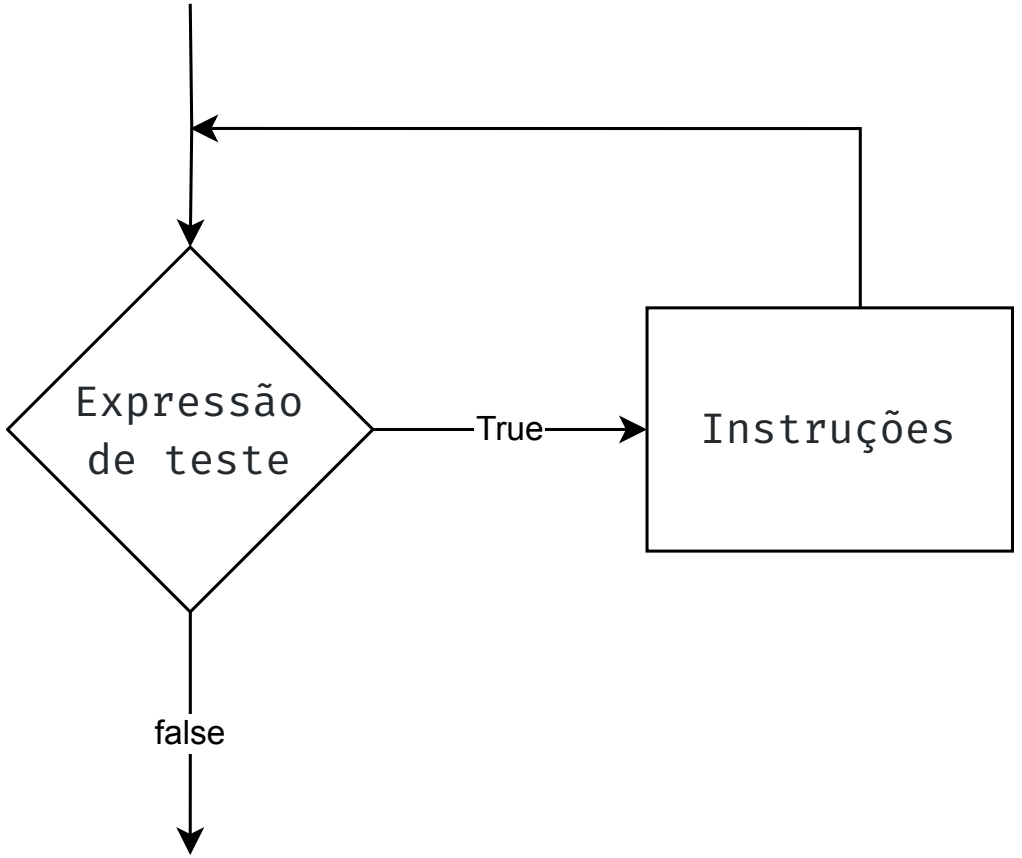
```
switch (teste) {
case expressão_constante:
    instrução;
    //...
    break;
case expressão_constante:
    instrução;
    //...
    break;
default:
    instrução;
    //...
}
```

- `teste` é uma expressão que resulta em um valor `int`.
- `expressão_constante` é uma expressão que deve ser definida em tempo de compilação, ou seja, o valor não pode ser definido dinamicamente durante a execução.

Instruções de repetição `while`

```
while(teste) {
    instrução;
    //...
}
```

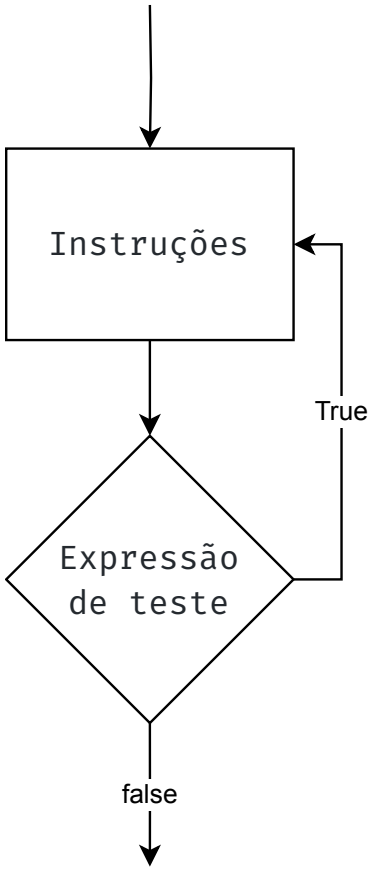
- As instruções dentro do *corpo* do `while` executam repetidamente até que a expressão `teste` resulte em falso.



Instruções de repetição `do-while`

```
do {
    instrução;
    //...
} while(teste);
```

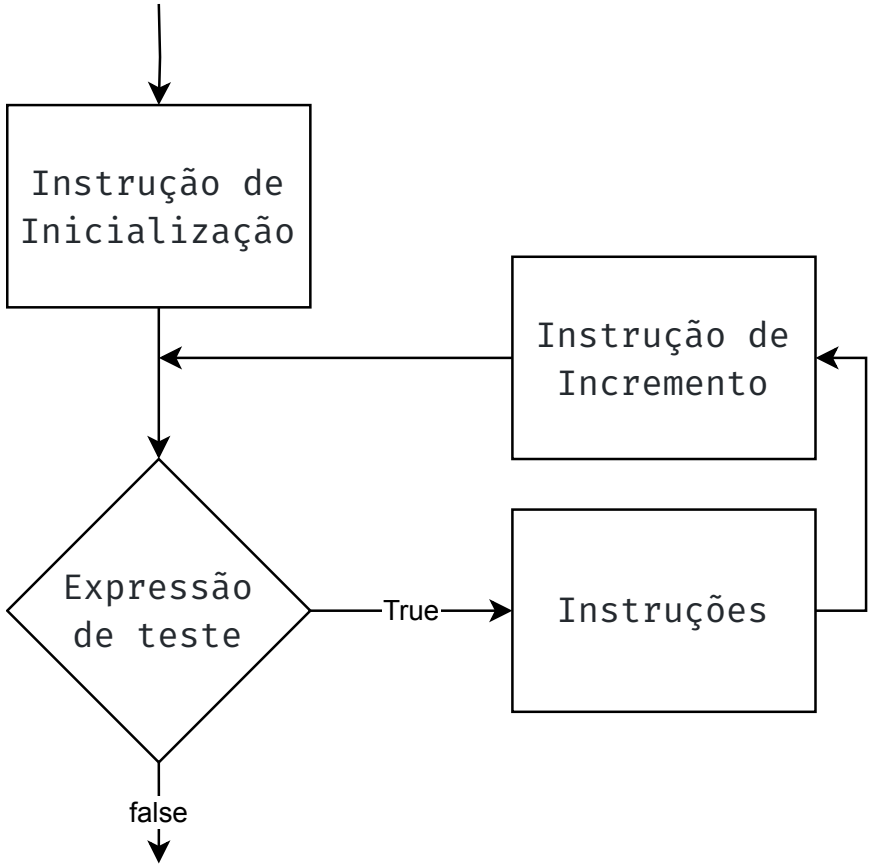
- As instruções dentro do *corpo* do `do` são executadas, e então a expressão `teste` é avaliada. Se resultar em verdadeiro, o corpo do `do` é executado novamente.



Instruções de repetição `for`

```
for (inicialização; teste; incremento) {
    instrução;
    //...
}
```

- As instruções do *corpo* do `for` são executadas repetidamente enquanto a expressão de teste for verdadeira. A variável do teste é atualizada a cada iteração.



Algoritmos e Programação II

https://evandro-crr.github.io/alg2

Evandro Chagas Ribeiro da Rosa