Linguagem de Programação I

Aula 6.3 – Estruturas de Repetição - DO - WHILE

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Estruturas de Repetição

Executando comandos múltiplas vezes

Por que precisamos de repetição?

▪ Sem repetição (inviável)

```
printf("1 ");
printf("2 ");
printf("3 ");
...
printf("100");
```

▪ Com repetição (elegante)

```
for(int i=1; i<=100; i++)
  printf("%d ", i);
```

▪ Definição

Permitem que um bloco de comandos seja executado **repetidamente** até que uma **condição de interrupção** seja satisfeita. A condição é uma expressão lógica. [Schildt e Mayer 1997]

Tipos em C:

1. `for`
2. `while`
3. `do-while`

Comparando as Estruturas de Repetição

▪ `for`

Nº iterações **conhecido**.

```
for(i=0; i<N; i++) { cmds; }
```

▪ `while`

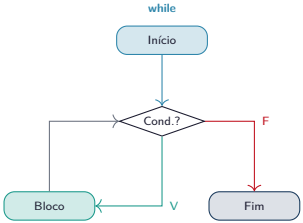
Nº iterações **desconhecido**; condição verificada **antes**.

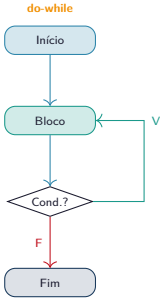
```
while(cond) { cmds; }
```

▪ `do-while`

Executa ao menos **uma vez**; condição verificada **depois**.

```
do { cmds; } while(cond);
```





Estrutura DO-WHILE

Executa primeiro, testa depois

Estrutura `do-while`

Como funciona

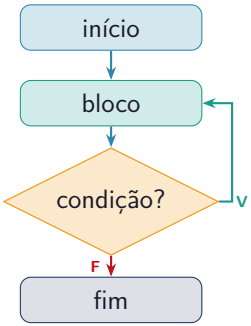
Executa o bloco de comandos **primeiro** e só depois verifica a condição. Se verdadeira, repete. Garante **ao menos uma execução**, independente da condição.

Sintaxe

```
do {
  instrucoes;
} while (condicao);
```

Quando usar?

Ideal para **menus** e **validação de entrada** — sempre precisamos executar ao menos uma vez antes de perguntar se o usuário quer continuar.



DO-WHILE — Contagem Simples

main.c

```c
#include <stdio.h>
void main(){
int i=0;
do {
    printf("%d ",i);
    i++;
}while (i <= 10);
}
```

```
0 1 2 3 4 5 6 7 8 9 10
```

Loop Infinito com DO-WHILE

▪ Loop Infinito

Assim como no `while`, é possível criar um loop infinito garantindo que a condição seja **sempre verdadeira** (qualquer valor $\neq 0$).

▪ Sintaxe

```
do {
  instrucoes;
} while (1); // sempre V
```

▪ Saída com `break`

Use `break` dentro do bloco para sair do loop quando uma condição específica for atingida.

```
if (condicao) break;
```

```c
#include <stdio.h>
void main(){
int i=0;
do {
    printf("%d ",i);
    i++;
}while (1);
}
```

DO-WHILE com `break`

```c
#include <stdio.h>
void main(){
int i=0;
char resp;
do {
    printf("%d ",i);
    i++;
    printf("Deseja parar? S/N: ");
    scanf("%c",&resp);
    getchar();
    if(resp=='S') break;
}while (1);
}
```

```
0 Deseja parar? S/N: n
1 Deseja parar? S/N: n
2 Deseja parar? S/N: n
3 Deseja parar? S/N: n
4 Deseja parar? S/N: S
```

Exercícios

Pratique com DO-WHILE

Exercícios — DO-WHILE

▪ Resolva usando `do-while`

1. Leia uma quantidade **desconhecida** de números e conte quantos estão nos intervalos: $[0, 25]$, $[26, 50]$, $[51, 75]$ e $[76, 100]$. Encerre ao ler um número **negativo**.

2. Implemente um sistema de **login e senha**:
   Login = `"ifbaBSI"` Senha = `"alunoIFBA"`
   Exiba **"Acesso Negado"** se errar e **"Acesso Permitido"** se acertar. O usuário pode tentar **indefinidamente** até acertar.

3. Leia *N* **valores positivos**, encontre e exiba o **maior** e o **menor**. Encerre ao ler um valor **negativo**.

Referências I

SCHILDT, H.; MAYER, R. *C completo e total*. [S.l.]: Pearson University, 1997. ISBN 9788534605953.