



Professor
Maromo

LINGUAGEM DE PROGRAMAÇÃO

Material 004



GitHub
maromo71

C



Agenda



Estruturas de Controle

- Laço (for, while, do..while),
- Comando break,
- Comando continue

Material: LP_004



Comando for



O comando **for** é uma estrutura de repetição em C que permite executar um bloco de comandos várias vezes. Ele é especialmente útil quando você sabe, antecipadamente, quantas vezes deseja que um bloco de código seja executado.

Sintaxe

```
for (inicialização; condição; atualização) {  
    // bloco de comandos  
}
```

- **Inicialização:** É executada uma vez no início do loop. Geralmente, é usada para definir a variável de controle do loop.
- **Condição:** É uma expressão booleana que é avaliada antes de cada iteração do loop. Se a condição for verdadeira, o bloco de comandos é executado. Se for falsa, o loop termina.
- **Atualização:** É executada após cada iteração do loop e é geralmente usada para atualizar a variável de controle.



Exemplo:

Vamos considerar um exemplo simples em que queremos imprimir os números de 1 a 5:

```
#include <stdio.h>

int main() {
    int i;
    for (i = 1; i <= 5; i++) {
        printf("%d\n", i);
    }
    return 0;
}
```

- A variável *i* é inicializada com o valor 1.
- Antes de cada iteração, verifica-se se *i* é menor ou igual a 5.
- Após cada iteração, *i* é incrementado em 1 (isso é feito pelo *i++*).
- O loop continuará enquanto *i* for menor ou igual a 5, e imprimirá os números de 1 a 5.

Cuidado: laço infinito

O loop infinito é representado pela estrutura:
for (inicialização;; incremento) comandos;

Como a condição de término não é especificada, o **loop é considerado infinito** e continuará executando indefinidamente. Para encerrar um loop desse tipo, utilizamos o comando **break**.

Ao invocar o **break**, o loop é interrompido imediatamente e a execução do programa prossegue com as instruções subsequentes.

Exemplo: laço infinito

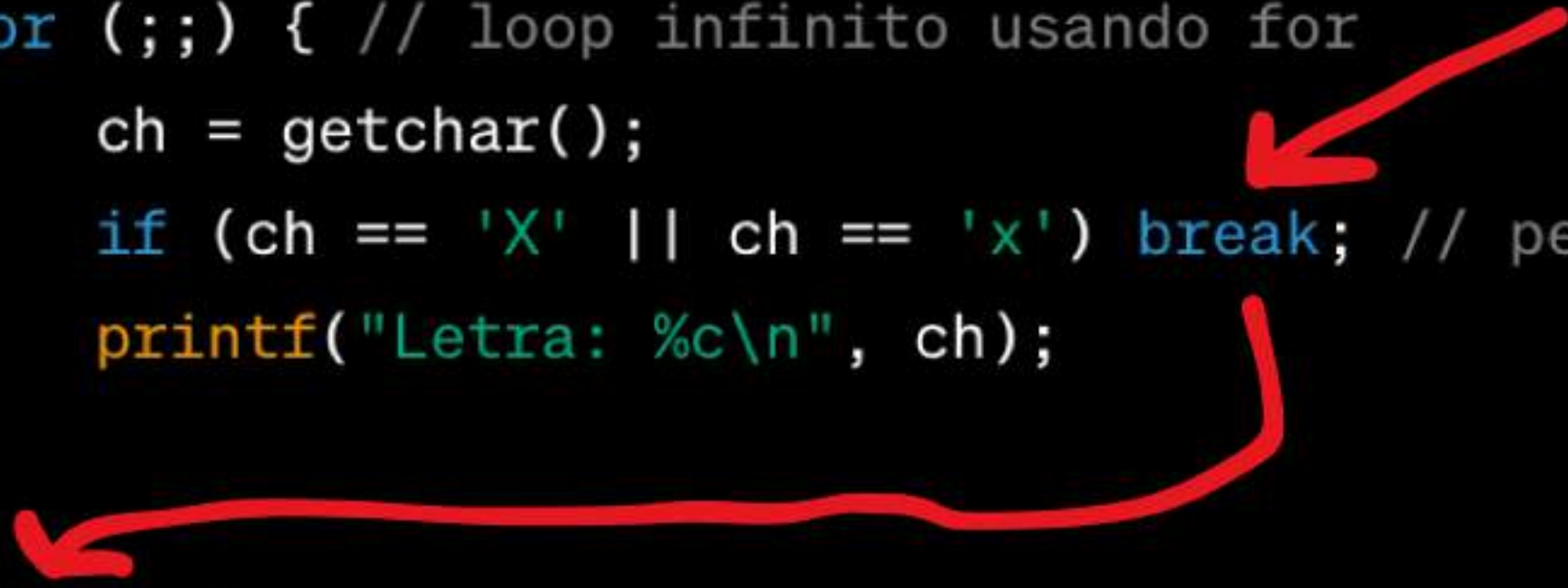
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    char ch;

    printf("Pressione 'X' para sair do loop.\n");

    for (;;) { // loop infinito usando for
        ch = getchar();
        if (ch == 'X' || ch == 'x') break; // permite que tanto 'X' maiúsculo quanto 'x' minúsculo saiam do loop
        printf("Letra: %c\n", ch);
    }

    return 0;
}
```



Laço sem conteúdo

Um loop sem corpo é caracterizado pela ausência de instruções dentro de sua declaração. Sua sintaxe padrão é (observe o ponto e vírgula!):

```
for (inicialização; condição; incremento);
```

Essa estrutura é frequentemente utilizada para criar pausas ou tempos de espera no programa.

Exemplo - Laço sem conteúdo

Um método comum para **gerar um tempo de espera em C**, especialmente em sistemas que não possuem funções específicas de delay, é usar um loop vazio para consumir tempo. No entanto, é importante observar que esse método é **bastante rudimentar** e o tempo real de espera pode variar dependendo da máquina e do compilador.

```
#include <stdio.h>

int main() {
    unsigned long int delay;

    printf("Iniciando...\n");

    for(delay = 0; delay < 100000000; delay++); // loop vazio para criar d

    printf("Finalizado!\n");

    return 0;
}
```



Comando
While

Comando while

O comando **while** é uma das estruturas de controle de repetição fundamental na linguagem C. Ele permite executar um bloco de comandos repetidamente enquanto uma condição especificada for verdadeira.

Sintaxe básica

```
while (condição) {  
    // bloco de comandos  
}
```

Condição: É uma expressão booleana que é avaliada antes de cada iteração do loop. Se a condição for verdadeira, o bloco de comandos dentro do loop é executado. Se for falsa, a execução do loop termina e o programa continua com qualquer comando que venha após o loop.

Comando while – Exemplo

Quando contador chegar em 6, sai fora do laço

```
#include <stdio.h>

int main() {
    int contador = 1;

    while (contador <= 5) {
        printf("%d\n", contador);
        contador++; // Incrementa o valor de contador
    }

    return 0;
}
```




Observações sobre o comando while

- É **crucial garantir que algo dentro do loop modifique a condição**, de modo que ela eventualmente se torne falsa; caso contrário, você pode acabar com um loop infinito.
- O loop **while** é **especialmente útil quando você não sabe antecipadamente quantas vezes o bloco de comandos precisa ser executado**, já que a decisão é baseada em uma condição que pode mudar dinamicamente durante a execução do programa.



Comando
do..while

Comando do..while

O comando **do..while** é uma estrutura de controle de repetição em C que é similar ao while, mas com uma diferença fundamental: no do..while, o **bloco de comandos é executado pelo menos uma vez**, independentemente da condição, pois a verificação da condição ocorre após a execução do bloco.

Sintaxe básica

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

Comando do..while – Exemplo

Quando contador chegar em 6, sai fora do laço

```
#include <stdio.h>

int main() {
    int contador = 1;

    do {
        printf("%d\n", contador);
        contador++; // Incrementa o valor de contador
    } while (contador <= 5);

    return 0;
}
```




Observações sobre o comando do..while

- O do..while é útil em situações **em que você quer garantir que o bloco de comandos seja executado pelo menos uma vez**, independentemente da condição.
- Assim como com o **loop while**, é **crucial garantir que algo dentro do loop modifique a condição**, evitando assim um loop infinito.

Escolha pelo número



```
int main(int argc, char *argv[]) {  
    int escolha;  
    do {  
        printf(" \n \nEscolha a fruta pelo número: \n");  
        printf(" \t(1)...Mamão \n");  
        printf(" \t(2)...Abacaxi \n");  
        printf(" \t(3)...Laranja \n \n");  
        scanf("%d", &escolha);  
    } while (escolha < 1 || escolha > 3);  
    printf(" \t \tVocê escolheu ");  
    switch (escolha) {  
        case 1:  
            printf("Mamão. \n");  
            break;  
        case 2:  
            printf("Abacaxi. \n");  
            break;  
        case 3:  
            printf("Laranja. \n");  
            break;  
    }  
    return 0;  
}
```


Resumo dos Laços

Laço	Sintaxe	Execução da Instrução	Utilização
while	<pre>while (condição) { comandos; }</pre>	Antes de cada iteração	Usado quando o número de iterações é desconhecido e depende de uma condição para ser verdadeira.
for	<pre>for (inicialização; condição; atualização) { comandos; }</pre>	Inicialização é feita uma vez no início; condição é verificada antes de cada iteração; atualização ocorre após cada iteração	Ideal quando o número de iterações é conhecido ou pode ser determinado. Permite inicialização, condição e atualização em uma única linha.
do..while	<pre>do { comandos; } while (condição);</pre>	Após cada iteração	Similar ao while, mas garante que o bloco de comandos seja executado pelo menos uma vez, independentemente da condição.



Exercícios - Aula 04

Exercício 1: Progressão Aritmética (PA)

Enunciado: Crie um programa que gere uma Progressão Aritmética (PA) com uma razão especificada pelo usuário. A série deve ter 10 termos e começar com o número 1.

Exercício 2: Soma de Números

Enunciado: Desenvolva um programa que leia uma série de números inteiros N (onde $0 \leq N < 200 \leq N < 20$). O programa deve calcular e apresentar a soma desses números. A entrada de um valor 0 deve encerrar a leitura e iniciar o cálculo da soma.

Exercício 3: Contagem de Números Pares

Enunciado: Escreva um programa que receba 10 números inteiros do usuário e conte quantos desses números são pares.



Exercícios - Aula 04

Exercício 4: Conversão de Temperatura

Enunciado: Elabore um programa que crie uma tabela de conversão de temperaturas de Fahrenheit para Celsius. A tabela deve incluir temperaturas de 0 a 100 graus Celsius, variando de 1 em 1 grau. Utilize a fórmula **$C = (F - 32) / 1.8$**

Exercício 5: Análise de Valores Inteiros

Enunciado: Construa um programa que faça o seguinte:

- Leia 5 valores inteiros.
- Identifique e exiba o maior valor.
- Identifique e exiba o menor valor.
- Calcule e exiba a média dos números inseridos.

Referências

DAMAS, L. M. D. Linguagem C. LTC, 2007.

HERBERT, S. C completo e total. 3a. ed. Pearson, 1997.

