

Introdução à Programação

Programação C

Comandos de Repetição

Prof. Roberto M. de Faria/UASC/UFCG

Conteúdo

- Comandos de Atribuição
- Operadores de Incremento/Decremento
- Comandos de Repetição
- Comando **for**
- Comando **while**
- Comando **do-while**

Comandos de Atribuição

- **Operadores de Atribuição** (**+=**, **-=**, ***=**, **/=** e **%=**) permitem a abreviação de expressões de atribuição
- A expressão em C
`var = var + 3;`
- Poderá se abreviada usando o **operador de atribuição de adição**
`var += 3;`
- Forma do comando de atribuição
`variável = variável operador_aritmético operando;`
- Poderá se reescrita com o operador de atribuição
`variável operador_aritmético= operando;`

Comandos de Atribuição

- Exemplos de outros operadores de atribuição:

$d -= 4 \quad \leftrightarrow \quad d = d - 4$

$e *= 5 \quad \leftrightarrow \quad e = e * 5$

$f /= 3 \quad \leftrightarrow \quad f = f / 3$

$g \% = 9 \quad \leftrightarrow \quad g = g \% 9$

Operadores de Incremento/Decremento

- Operador de Incremento (**++**)
 - Uso em substituição de **+= 1**
- Operador de Decremento (**--**)
 - Uso em substituição de **-= 1**
- Pré-incremento / Pré-decremento
 - Operador usado antes da variável (**++var** ou **--var**)
 - Variável alterada (incrementada ou decrementada) antes da avaliação da expressão
- Pós-incremento/Pós-decremento
 - Operador usado após da variável (**var++** ou **var--**)
 - Comando que contém a variável será executado antes da alteração da variável (incremento ou decremento)

Operadores de Incremento/Decremento

- Exemplo:

```
var = 5;  
printf("%d", --var); // Imprimirá 4  
var = 5;  
printf("%d", var--); // Imprimirá 5
```

- Em qualquer dos casos, **var** assume o valor 4
- Quando a variável não está numa expressão o pré-incremento e pós-incremento apresentam o mesmo efeito:

```
var = 5;  
++var; // var igual a 6  
printf("%d", var); // Imprime 6  
var = 5;  
var++; // var igual a 6  
printf("%d", var); // Imprime 6
```

Comandos de Repetição

- Comandos de repetição, ou **laços**, permitem que um comando ou **bloco** (**{comandos}**) sejam repetidos várias vezes, sem que sejam reescritos várias vezes
- As instruções que são repetidas são chamadas de **corpo do comando de repetição** ou **corpo do laço**.
- Na linguagem C, existem três diferentes comandos de repetição (laços): o comando **for**, o comando **while**, e o comando **do-while**
- O comando **for** é usado comumente como um comando de repetição cuja execução depende de uma contagem
- Os comandos **while** e **do-while** são usados comumente como **laços condicionais**, ou seja, dependem de uma condição ser verdadeira para continuarem a ser executados

Comando **for**

- O comando **for** ,ou laço **for**, é usado quando é necessário repetir comandos em um programa, ou função, se pode ser conhecido previamente quantas vezes os comandos deverão ser repetidos
- Chama-se a variável que é usada para a contagem das iterações do laço, de ***variável de controle***
- Forma geral do comando **for**:

```
for (expr-1; expr-2; expr-3)  
    comando; ou { . . . }
```


Comando **for**

- Onde:
 - **expr-1** → *inicializações*: uma ou mais expressões, separadas por vírgula, que serão executadas uma única vez, no início do **for** – comumente a inicialização da variável de controle;
 - **expr-2** → *teste*: condição – comumente teste da variável de controle;
 - **expr-3** → *atualizações*: uma ou mais expressões separadas por vírgula, que serão executadas antes do próximo teste – comumente incremento ou decremento da variável de controle.
- O corpo do **for** deve ser deslocado à direita para identificá-lo mais facilmente
- Qualquer expressão do comando **for** pode estar ausente.
- Outro ponto importante: se a **expr-2** (teste) estiver ausente o seu valor será considerado *verdadeiro* (diferente de zero)

Comando **for**

- Representação de um laço infinito com o comando **for**:

```
for ( ; ; )
```

```
    comando; ou { . . . }
```

- Por exemplo, será impressa uma coluna de números de 1 a 10:

```
for (contador = 1; contador <= 10; contador++)
```

```
    printf("%d\n", contador);
```

- O que imprimirá este laço?

```
for (controle = 20; controle >= 1; controle -= 3) {
```

```
    printf("Eu gosto de programar ");
```

```
    printf("na linguagem C!\n");
```

```
}
```

Exemplo de Comando `for`

```
int vezes; conta_maiusc = 0; conta_minusc = 0; conta_outro = 0;

printf("\nConta os tipos de 10 caracteres lidos\n");
printf("como minúsculo, maiúsculo ou outro:\n\n");
for (vezes = 1; vezes <= 10; vezes++) { // Uso só operador "+"

    printf("Digite um caractere: ");
    carac = getchar();
    if (carac >= 'A' && carac <= 'Z')
        ++conta_maiusc;
    else if (carac >= 'a' && carac <= 'z')
        ++conta_minusc;
    else
        ++conta_outro;
}

printf("Maiúsculas = %d, Minusculas = %d e Outros = %d\n",
       conta_maiusc, conta_minusc, conta_outro);
```

Exercícios

- 1) Faça um programa que calcule, classifique, e imprima o IMC de 5 pessoas. Pesquise na internet sobre o IMC e sua classificação
- 2) Modifique o programa anterior para que ele execute para um número determinado de pessoas
- 3) Faça um programa que calcule e mostre o fatorial de um numero natural
- 4) Faça um programa que some os números naturais até 20

Exercícios

- 5) Modifique o programa anterior para que some os números naturais até N
- 6) Faça um programa que some os números naturais no intervalo de M a N , inclusive
- 7) Faça um programa que calcule e mostre a soma de 10 números inteiros quaisquer
- 8) Modifique o programa anterior para que calcule e mostre a soma de N números inteiros

Exercícios

9) Faça um programa para desenhar as seguintes figuras, dado o número de linhas:

a) *	b) *	c) *****	d) *****
**	**	***	***
***	***	**	**
****	****	*	*

10) Faça um programa para desenhar e movimentar na tela, da esquerda para a direita, o caminhonete a seguir:

```

          _____ \
          |||
+-----+-----+-----0
  O
  |   _____   |   _____ \
O O =+-+ (O) +--+--+-----++ (O) +--+
  
```

Comando de Repetição **while**

- O comando **while** é usada como um ***laço condicional***
- Ele é usado para repetir uma **ação** quando, previamente, não se sabe quantas vezes a **ação** será repetida – parar a repetição depende de um evento futuro e em momento imprevisível
- A forma geral do comando **while** é:
while (condição)
comando; ou {...}
- O ***corpo do while***, que consiste de um **comando** ou de qualquer quantidade de comandos dentro de um **bloco**, é executado, repetidamente, enquanto a **condição** for ***verdadeira***

Comando de Repetição **while**

- O modo como funciona, é que primeiro, a **condição** é avaliada
- Se a **condição** é logicamente **verdadeira**, o **corpo do while** é executado
- Até aí, o comando **while** é igual a um comando **if**.
- No entanto, nesse ponto a **condição** é avaliada novamente
- Se é ainda **verdadeira**, o **corpo do while** é executado novamente e assim por diante
- Então, em algum momento, algo no **corpo do while** tem que mudar para assim, a **condição** tornar-se **falsa** e a repetição ser interrompida – a **condição** deve, eventualmente, tornar-se **falsa** para evitar um **laço infinito**

Comandos de Repetição

- Exemplo com o uso do `while`:

```
int inteiro, invertido;

printf("\nPrograma que recebe um inteiro positivo\n"
       "e mostra-o com seus dígitos invertidos:\n\n");

printf("Digite um número inteiro positivo: ");
scanf("%d", &inteiro);

invertido = 0;

while (inteiro > 0) {
    invertido = invertido * 10 + inteiro % 10;
    inteiro = inteiro / 10;
}

printf("\nNúmero com os dígitos invertidos:\n\n");
printf("%d\n\n", invertido);
```

Comandos de Repetição

- Exemplo com `while`:

```
int parcela, soma = 0;
printf("Programa que soma inteiros positivos e ");
printf("para quando encontra um negativo ou 0:\n");
printf("Informe um inteiro a somar: ");
scanf("%d", &parcela);
while (parcela > 0) {
    soma += parcela;
    printf("Informe um inteiro a somar: ");
    scanf("%d", &parcela);
}
printf("\nA soma dos inteiros positivos é %d\n\n", soma);
return 0;
```

Exercícios

- 1) Modifique o programa exemplo com `while` para somar valores e parar quando aparecer um múltiplo de 5.
- 2) Modifique o programa anterior para parar quando aparecer um múltiplo de N
- 3) Faça um programa que calcule o MDC de dois inteiros positivos (Exemplo do método: MDC de 32 e 18)

Dividendo	Divisor	Resto
32	18	14
18	14	4
14	4	2
4	2	0

Diagram illustrating the Euclidean algorithm for finding the GCD (MDC) of 32 and 18. The table shows the sequence of divisions. Arrows indicate the flow of the algorithm: from the Divisor of one row to the Dividendo of the next, and from the Resto of one row to the Divisor of the next. The final row shows the remainder as 0, indicating the end of the process.

Arrows indicate the flow of the algorithm:

- From Divisor (18) to Dividendo (18)
- From Resto (14) to Divisor (14)
- From Divisor (14) to Dividendo (14)
- From Resto (4) to Divisor (4)
- From Divisor (4) to Dividendo (4)
- From Resto (2) to Divisor (2)
- From Divisor (2) to Dividendo (2)
- From Resto (0) to Divisor (0)

The final result is the MDC (2), indicated by the green circle around the 2 in the Divisor column.

Para quando igual a zero

Exercícios

- 4) Modifique o programa anterior para que no mesmo **printf** que mostra o MDC, mostrar também os valores sobre os quais foi calculado o MDC
- 5) Faça um programa para calcular as raízes de várias equações do 2º. grau e parar quando o coeficiente **a** for igual a zero
- 6) Faça um programa que receba um inteiro positivo e mostre os seus dígitos separados por espaços em branco

Exercícios

7) Modifique o programa anterior para que mostre a soma de dígitos de vários números recebidos e pare quando encontrar um número par

8) Um “Número de Armstrong” de n dígitos é aquele que é igual a soma das potências de seus dígitos com expoente n

Exemplo: $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27$

Mostre os números de Armstrong menores que M

Exercícios

- 9) Faça um programa para mostrar se um número positivo é ou não primo, lembrando que 1 é divisor de qualquer número e que os divisores de um número, com exceção dele mesmo, encontra-se entre 1 e a metade desse número
- 10) Faça um programa que gere um número aleatório entre 1 e 100 e aceite palpites do usuário até que o mesmo acerte esse número
- A cada palpite, o programa informa se o palpite foi maior, menor ou se o usuário acertou o número

Comando de Repetição **do-while**

- O comando **do-while** é usado como um *laço condicional*
- Similar à estrutura do **while**, mas o teste da condição para repetição ocorre após a execução do *corpo do do-while*
- O *corpo do do do-while* é executado pelo menos uma vez
- Formato:

do

comando; ou { . . . }

while (condição) ;

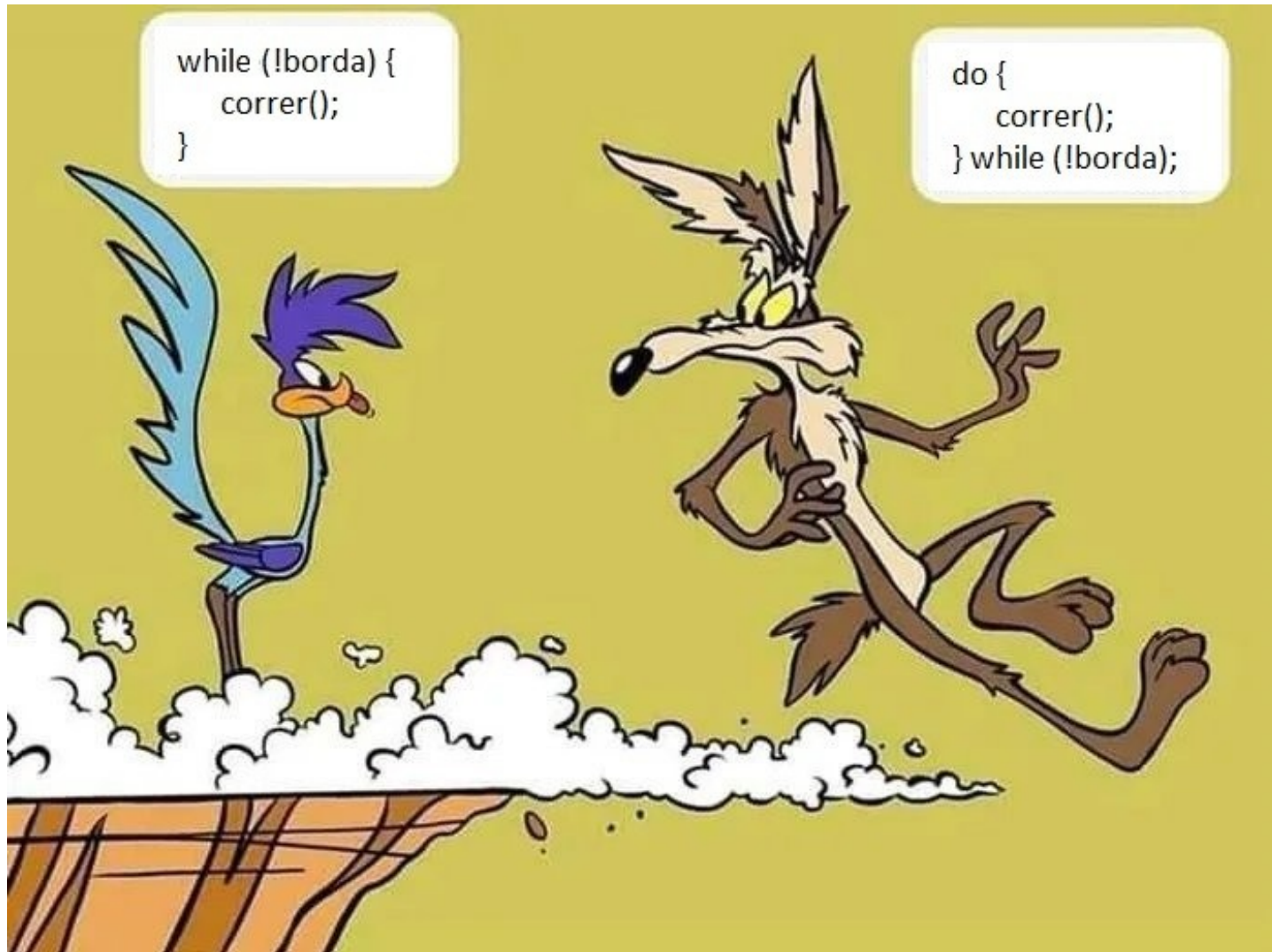
Comando de Repetição do-while

- Exemplo:

```
// Uso do comando de repetição do-while
// Mostra os inteiros de 1 a 10
#include <stdio.h>

int main() {
    int contador = 1;
    do {
        printf("%d ", contador);
    } while (++contador <= 10);
    return 0;
}
```


Diferença entre o **while** e o **do-while**



Exercícios

- 1) Escreva um programa para implementar uma calculadora com as seguintes operações reais: soma, subtração, multiplicação, divisão, exponenciação e raiz quadrada. A operação será escolhida por um menu. Os operandos serão solicitados de acordo com a operação escolhida. Haverá no menu uma opção para fechar a calculadora. O programa deve utilizar um comando **do-while**.
- 2) Faça um programa que simule o lançamento de dois dados várias vezes. O programa mostrará o resultado dos dois dados a cada lançamento e encerrará quando a soma dos pontos dos dados for 12. Use a função **rand()** na simulação dos lançamentos dos dados e um **do-while** para a repetição dos lançamentos. Garanta, com o uso da função **srand()** que a cada execução do programa a sequência dos pontos dos dados seja diferente.

Exercícios

- 3) Faça um programa que usando um comando **do-while** implemente a brincadeira de adivinhar um número num intervalo de inteiros por dois jogadores. O computador escolherá o número a ser descoberto pelo usuário, usando a função **rand()**, mas antes, um dos jogadores escolherá os valores inicial e final do intervalo. Os palpites dos jogadores (jogador-1 e jogador-2) serão alternados e o computador dará as mensagens: “O jogador-i acertou” (i será 1 ou 2), “Palpite muito baixo” ou “Palpite muito alto”.
- 4) Estenda o programa anterior para serem realizadas várias jogadas. As vitórias de cada jogador serão contabilizadas. Vencerá o jogo o jogador que atingir 10 vitórias ou o que conseguir 3 vitórias consecutivas.

Exercícios

- 5) Reescreva o programa do MDC, que utiliza o algoritmo de Euclides, agora, utilizando o comando **do-while** em vez do comando **while**.
- 6) Faça um programa para contabilizar a arrecadação de fundos para uma vaquinha com finalidade filantrópica específica, com determinado limite de arrecadação. O programa solicitará o limite de arrecadação e em seguida solicitará as doações para a vaquinha. Quando o limite de arrecadação da vaquinha for atingido ou superado, as doações serão encerrados e o excedente, se houver, será rateado equitativamente pelos doadores. No final, será informado: o limite de arrecadação, o valor arrecadado, o número de doadores e quanto será devolvido a cada doador. Use um comando **do-while** para controlar a solicitação de doações.

Comandos **break** e **continue**

- Comando **break**
 - Saída imediata de uma estrutura **for**, **while**, **do-while** ou **switch**
 - Continuação da execução do programa a partir da primeira instrução após o comando de repetição ou após do **switch**
- Usos comuns da instrução **break**
 - Saída prematura de um laço
 - Desconsideração do restante de uma estrutura **switch**

Comandos **break** e **continue**

- Comando **continue**
 - Desconsideração das instruções restantes do corpo de um comando **for**, **while** ou **do-while**
 - Prosseguimento com a próxima iteração do *laço*
- **while** e **do-while**
 - Avaliação do teste de continuação do laço imediatamente após a execução da instrução **continue**
- **for**
 - Execução da(s) expressão(ões) de atualização(ões) e em seguida, a avaliação do teste do **for**

Comandos **break** e **continue**

```
// Uso do comando continue em um comando for
#include <stdio.h>

int main() {
    int x;
    for (x = 1; x <= 10; x++) {
        if (x == 5)
            continue; // Ignora os comandos restantes,
                       // se x igual a 5
        printf("%d ", x);
    }
    printf("\ncontinue foi usado para saltar a"
           "impressão do 5\n");
    return 0;
}
```