Linguagem de Programação

comandos de repetição e operadores de incremento

Prof. Francisco Glaubos

Tópicos abordados

- operadores aritméticos de atribuição, de incremento e decremento
- comandos de repetição e interrupção de laço

Expressões compactas

- Operadores aritméticos de atribuição
  - combinam em um único operador, uma operação aritmética e uma atribuição

Expressão	Forma compacta
$x = x + y$	$x \mathrel{+}= y$
$x = x - y$	$x \mathrel{-}= y$
$x = x * y$	$x \mathrel{*}= y$
$x = x / y$	$x \mathrel{/}= y$
$x = x \% y$	$x \mathrel{\%}= y$

Expressões compactas

- incremento e decremento
  - se uma expressão incrementa ou decrementa o valor de uma variável, podemos escrevê-la de uma forma mais compacta com os operadores ++ ou --
  - podem ser usados na forma prefixa ou posfixa:
    - ++variavel, --variavel
    - variavel++, variavel--

```
// Operadores de incremento e decremento.
int x=5, y=5;
++x;
y--;
printf("\n x=%d y=%d", x, y);
```

- Como esperado, a saída produzida pelo código será x=6 y=4.

Expressões compactas

- diferença entre prefixa e posfixa:
  - na forma prefixa, a variável é alterada e, depois, seu valor é usado.
  - na forma posfixa, o valor da variável é usado e, depois, ela é alterada.

```
int main(void) {
 int x=5, y=5, v, w;
 v = ++x;
 w = y--;
 printf("x=%d  y=%d  v=%d  w=%d \n",x,y,v,w);
 return 0; }
```

- Quais serão os valores de x,y,v e w ?

Expressões compactas

- diferença entre prefixa e posfixa:
  - na forma prefixa, a variável é alterada e, depois, seu valor é usado.
  - na forma posfixa, o valor da variável é usado e, depois, ela é alterada.

```
int main(void) {
 int x=5, y=5, v, w;
 v = ++x;
 w = y--;
 printf("x=%d  y=%d  v=%d  w=%d \n",x,y,v,w);
 return 0; }
```

- Quais serão os valores de x,y,v e w ? x=6 y=4 v=6 w=5

Exercício

Seja x=5 e considere a instrução y = x++ + ++x. Quais os valores das variáveis x e y após a execução dessa instrução? Por quê?

Exercício

Seja x=5 e considere a instrução y = x++ + ++x. Quais os valores das variáveis x e y após a execução dessa instrução? Por quê?

Resposta: x = 7 y = 12

Repetição com contador

- Em C, um laço de repetição pode ser controlado por um contador
  - contador: variável que contabiliza quantas vezes um comando é executado
  - sintaxe: *for(inicialização; condição; alteração) comando;*

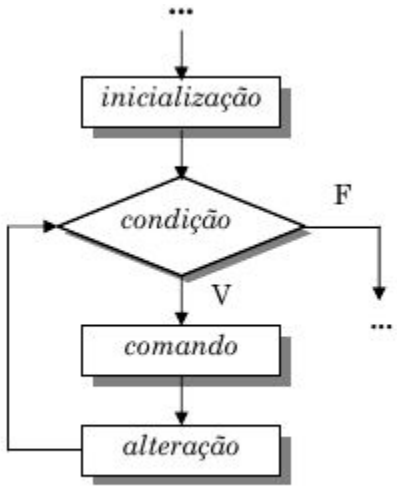


Figura 3.1 – *A estrutura de repetição com contador*

```
int main() {
 int c;
 for(c=1;c<=9;c++) printf("%d ", c);
 return 0;
}
```

A saída produzida pelo código será 1 2 3 4 5 6 7 8 9

Repetição com contador

O ***for*** em C é bem flexível:

```c
int max = 5;
/* Omite a primeira e a última expressão */
int c = 0;
for(; c < max;){
      c++;
      printf("c = %d\n",c);
}
```

```c
/* omitindo todas as expressões (loop infinito)*/
   int i = 0;
   for(;;)
   {
       i++;
       /* Usando break para escapar do loop*/
       if(i > max)
           break;
       printf("i = %d\n",i);
   }
```

For aninhados

```
int main()  {
  int n;// variable declaration
  printf("Enter the value of n :");
  scanf("%d", &n);
  // Displaying the n tables.
  for(int i=1;i<=n;i++) {  // outer loop
      for(int j=1;j<=10;j++) {  // inner loop
         printf("%d\t",(i*j)); // printing the
value.
      }
      printf("\n");
  }
  return 0; }
```

```
Enter the value of n :3
1  2  3  4  5  6  7  8  9  10
2  4  6  8  10  12  14  16  18  20
3  6  9  12  15  18  21  24  27  30
```

Exercício

- Escreva um programa em C para calcular o fatorial de um dado número.

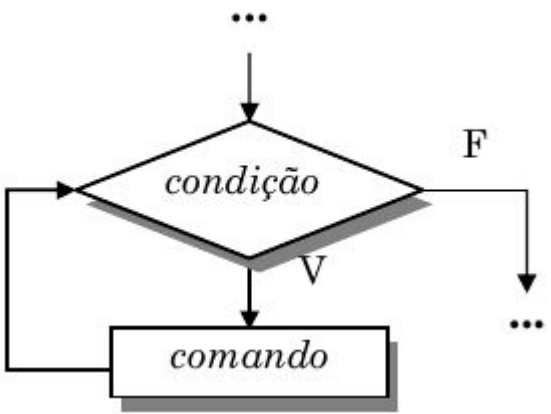
Exercício

- Dado um inteiro x, desenhe um padrão de tamanho x utilizando ‘*’. Por ex.: se x=4 o padrão é:

  ****
  ***
  **
  *

Repetição com pré-condição

- O C possui uma estrutura de repetição mais genérica que o **for**
- sintaxe: *while(condição) comando;*




```c
int main () {
   /* local variable definition */
   int a = 10;
   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }
   return 0; }
```

Do...while

- Testa a condição de continuação depois de o corpo do loop ser executado
- Sempre será executado pelo menos uma vez

```
do {
instrução
} while (condição) ;
```

Instruções Break e Continue

- são usadas para alterar o fluxo de controle em estruturas como while, for, do/while ou switch
- break : quando executada em uma estrutura, faz com que aconteça a saída imediata dessa estrutura
- continue : ignora (salta sobre) as instruções restantes no corpo da estrutura e realiza a próxima iteração do *loop*

```
/* Usando a instrução break em uma
estrutura for */
#include <stdio.h>
main( ) {
 int x;
 for (x = 1; x <= 10; x++) {
 if (x == 5)
   break; /* sai do loop somente se x ==
5 */
 printf ("%d ", x);  }
 printf ("\n Saiu do loop em x == %d\n",
x); return 0; }
```

Exercício

Escreva um programa em C que receba um valor inteiro X e mostre a quantidade de dígitos.

dica: analise o resultado de X/10

Exercício

- Crie um programa que leia um número inteiro de entrada X e o mostre na ordem inversa.

dica: analise o resultado de X%10

Exercício

- Crie um programa em C para contar a quantidade de ocorrências de um dígito específico em um número, ambos valores fornecidos como entrada