# Comando de Repetição: while

Ronaldo F. Hashimoto e Carlos H. Morimoto

## Introdução

Essa aula introduz o comando `while`, que permite repetir instruções enquanto uma condição for verdadeira. Para utilizar o comando corretamente, você precisa se lembrar de inicializar as variáveis de controle antes do comando, certificar-se que a condição do `while` se mantem verdadeira pelo número correto de iterações, e por fim garantir que a condição se torne falsa para terminar o looping.

Ao final dessa aula você deverá saber:

- Utilizar comandos de repetição na resolução de problemas computacionais.
- Definir condições iniciais e de parada para o comando `while`.
- Simular o processamento do comando `while`.

## Sintaxe

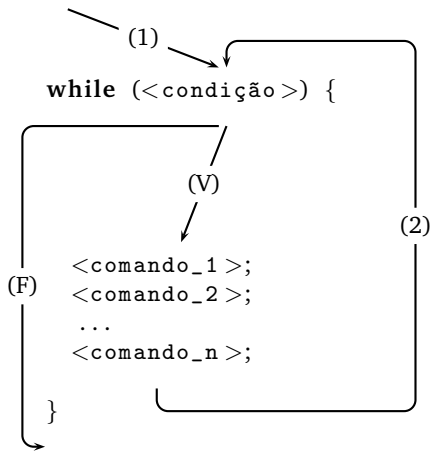
A sintaxe do comando de repetição do comando `while` pode ser vista ao lado.

A <condição> é uma expressão relacional que tem como resultado um valor **verdadeiro** ou **falso** (veja aula sobre **fundamentos**). A seqüência de comandos <comando_1>, <comando_2>, ..., <comando_n> pode conter comandos de atribuição, impressão de mensagens na tela ou leitura de números inteiros pelo teclado, entre outros.

```
while (<condição>) {
  <comando_1>;
  <comando_2>;
  ...
  <comando_n>;
}
```

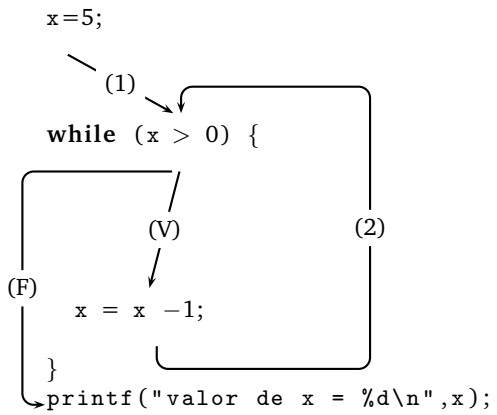
## Descrição

Basicamente, este comando de repetição tem o significado: enquanto a <condição> for **verdadeira**, a seqüência de comandos <comando_1>, <comando_2>, ..., <comando_n> é executada.




Vamos analisar o "fluxo" do programa usando o comando de repetição **while**. Primeiramente, quando a execução do programa chega no **while** (seta marcada com `(1)`) a <condição> é testada. Se "de cara" a <condição> é **falsa**, o fluxo do programa ignora a seqüência de comandos e segue a seta marcada com `(F)`. Agora, se a <condição> é **verdadeira**, então o fluxo do programa segue a seta marcada com `(V)` e executa a seqüência de comandos dentro do **while**; após executado o último comando (<comando_n>), o fluxo do programa segue a seta marcada com `(2)` e volta a testar a <condição>. Se a <condição> é **verdadeira**, então o fluxo do programa segue a seta marcada com `(V)` repetindo a seqüência de comandos dentro do **while**. Se <condição> é **falsa**, o fluxo do programa ignora a seqüência de comandos e segue a seta marcada com `(F)`.

Por exemplo, seja `x` uma variável inteira. O segmento de programa abaixo simplesmente subtrai 1 de `x`, 5 vezes (note que o comando `"x = x-1;"` é repetido 5 vezes).




O primeiro comando, a atribuição `x = 5;` (a variável `x` recebe o valor `5`), é executado antes do **while** (condição inicial da variável que controla o **while**). Depois o fluxo do programa segue a seta marcada com `(1)` e testa a condição `(x>0)` do **while**. Se ela é **verdadeira**, executa os comandos dentro do **while** (que nesse caso contém apenas a atribuição $x = x - 1$;) seguindo a seta marcada com `(V)`. Depois o fluxo do programa segue a seta marcada com `(2)` e a condição `(x>0)` é testada novamente (agora `x` tem um valor decrescido de um). Dessa forma, o comando $x = x - 1$; é executado enquanto a condição do **while** é **verdadeira**. Somente quando a condição for **falsa**, o **while** termina, seguindo a seta marcada com `(F)` e a instrução seguinte é executada (no caso, o `printf`).

**NOTA:** para que o seu programa termine, você precisa garantir que a <`condição`> do **while** seja alterada de alguma forma, ou seja, você precisa garantir que a condição de parada seja alcançada. Caso contrário, o programa entra em "looping infinito".

# Exemplos Comentados

## Exemplo 1

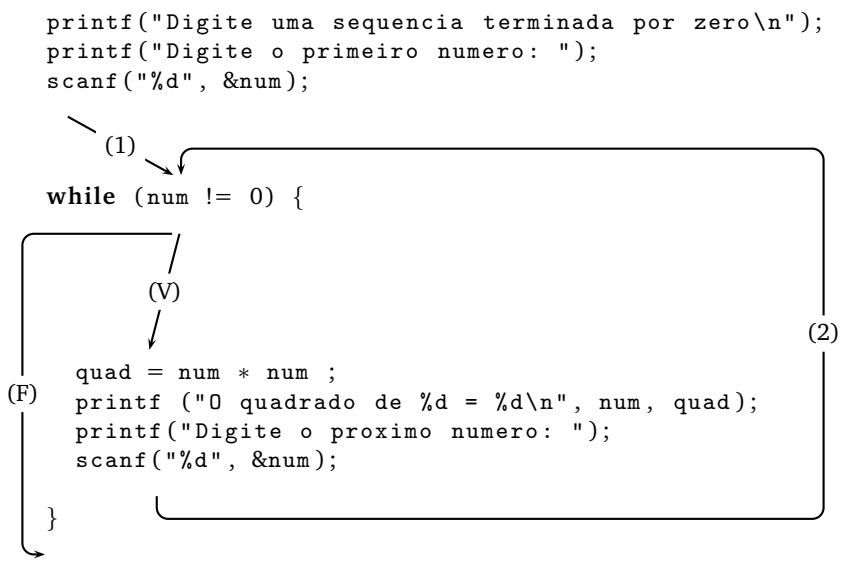
Dada uma seqüência de números inteiros diferentes de zero, terminada por um zero, imprima o quadrado de cada número da seqüência.

**Solução:**

Uma solução possível pode ser descrita de modo informal como:

1. imprima uma mensagem para o usuário saber o que fazer
2. leia pelo teclado o primeiro número da seqüência na variável `num`
3. enquanto `num` for diferente de zero faça:
   (a) calcule `quadrado` = `num` * `num`
   (b) imprima na tela o valor de `quadrado`
   (c) leia pelo teclado o próximo número da seqüência na variável `num`
4. fim

O funcionamento do programa pode ser entendido também pelo diagrama abaixo:




Em geral, é mais simples desenhar o diagrama e, quando você estiver certo de que ele funciona, sua "tradução" para a linguagem C é simples, basta copiar o esqueleto de um programa em C visto anteriormente, e preencher as lacunas. O programa em C ficaria:

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

int main () {
  /* declaracoes */
  int num; /* variavel utilizada para leitura da sequencia */
  int quad; /* variavel que armazena o quadrado de um numero */

  /* programa */
  printf("Digite uma sequencia terminada por zero\n");
  printf("Digite o primeiro numero: ");
  scanf("%d", &num);

  while (num != 0) {
    /* os simbolos '!=' significam diferente */
    quad = num * num ;
    printf ("O quadrado de %d = %d\n", num, quad);
    printf("Digite o proximo numero: ");
    scanf("%d", &num);
  }

  /* fim do programa */
  return 0;
}
```

## Exemplo 2

Dada uma seqüência de números inteiros diferentes de zero, terminada por zero, calcular a somatória dos números da seqüência.

**Solução:**

Para melhor entender o problema, vamos ver um exemplo concreto de uma seqüência numérica. Para a seqüência:

$$2 \quad 3 \quad -4 \quad 5 \quad 0$$

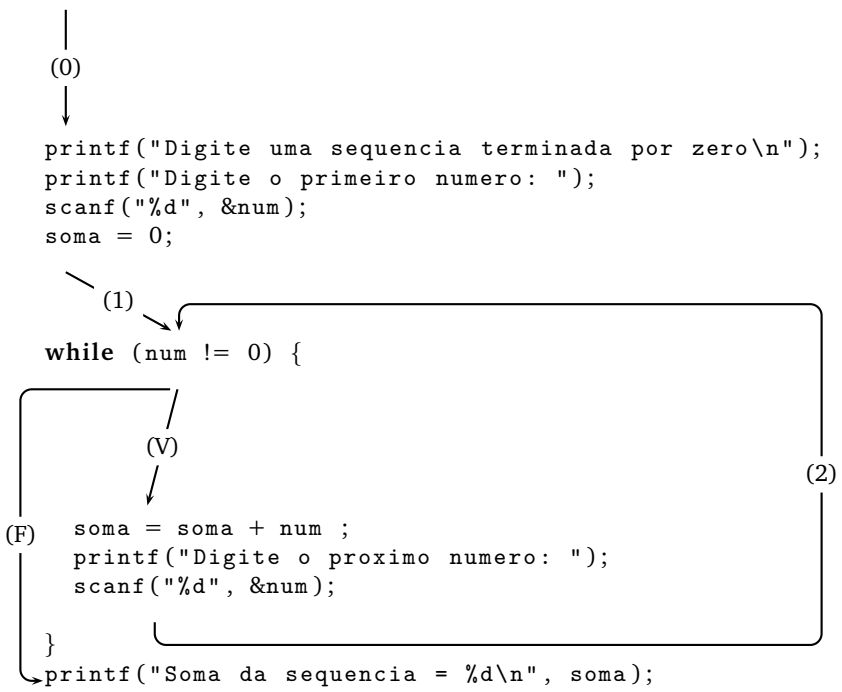
a saída de seu programa deve ser $6$ (ou seja, $2 + 3 - 4 + 5$).

Uma forma possível para resolver esse problema é imaginar uma variável que armazena as somas parciais. Essa variável deve iniciar com o valor zero, e para cada número da seqüência, ser somada com mais esse número, até o final da seqüência. Assim, para o exemplo acima, o valor de soma torna-se 2 após processar o primeiro elemento da seqüencia (soma-se o 2), 5 após o segundo (soma-se o 3), 1 após o terceiro (soma-se o 4), e assim até o final.

Uma solução possível pode ser descrita de modo informal como:

1. imprima uma mensagem para o usuário saber o que fazer
2. leia pelo teclado o primeiro número da seqüência na variável num
3. inicialize uma variável soma com zero
4. enquanto num for diferente de zero faça:
   (a) acumule na variável soma o número lido
   (b) leia pelo teclado o próximo número da seqüência na variável num
5. imprima na tela a soma final
6. fim

O funcionamento do programa pode ser entendido também pelo diagrama abaixo:




O programa completo ficaria:

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

int main () {
  /* declaracoes */
  int num;  /* variavel utilizada para leitura da sequencia */
  int soma; /* variavel que armazena a soma da sequencia */

  /* programa */
  printf("Digite uma sequencia terminada por zero\n");
  printf("Digite o primeiro numero: ");
  scanf("%d", &num);

  while (num != 0) {
    soma = soma + num ;
    printf("Digite o proximo numero: ");
    scanf("%d", &num);
  }

  printf("Soma da sequencia = %d\n", soma);

  /* fim do programa */
  return 0;
}
```

## Exercícios Recomendados

1. (exercício 4 da lista) Dados números inteiros $n$ e $k$, com $k >= 0$, determinar $n^k$ ($n$ elevado a $k$). Por exemplo, dados os números 3 e 4 o seu programa deve escrever o número 81.
2. (exercício 8 da lista) Dado um número inteiro $n >= 0$, calcular o fatorial de n (n!).

A solução para esses e outros exercícios você encontra na lista de exercícios em `http://www.ime.usp.br/~macmulti/exercicios/inteiros/index.html`.