

4 Comando de Repetição: while

Ronaldo F. Hashimoto e Carlos H. Morimoto

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`.

4.1 Sintaxe

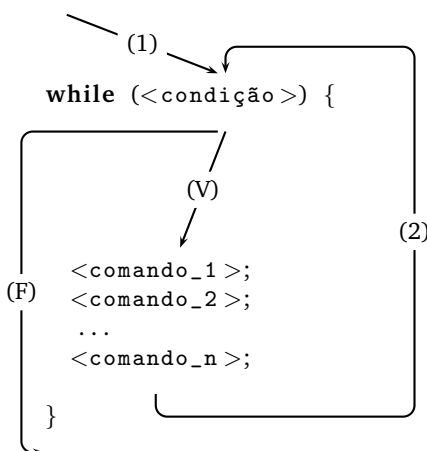
A sintaxe do comando de repetição do comando `while` pode ser vista ao lado.

A `<condição>` é uma expressão relacional que tem como resultado um valor **verdadeiro** ou **falso** (veja aula sobre **fundamentos**). A sequência 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 >;  
}
```

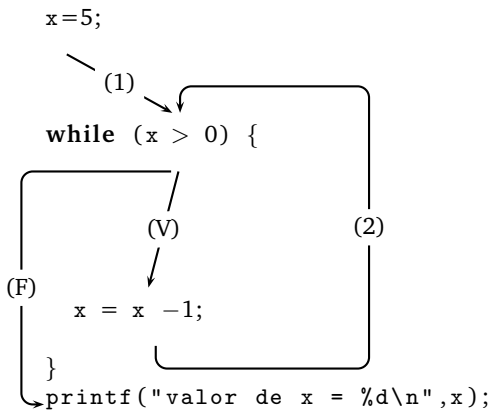
4.2 Descrição

Basicamente, este comando de repetição tem o significado: enquanto a `<condição>` for **verdadeira**, a sequência 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 sequência 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 sequência 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 sequência de comandos dentro do **while**. Se `<condição>` é **falsa**, o fluxo do programa ignora a sequência 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”.

4.3 Exemplos Comentados

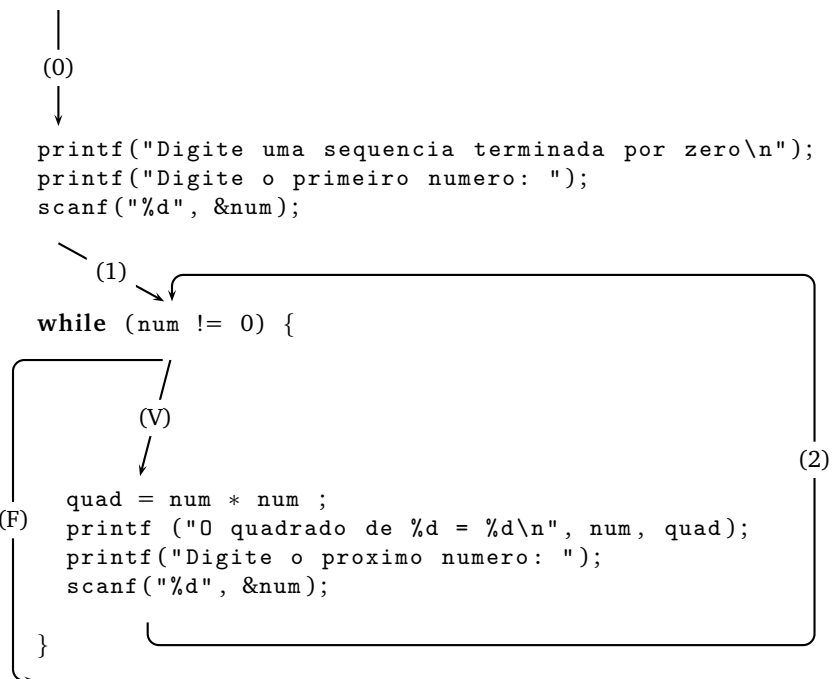
4.3.1 Exemplo 1

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

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 sequência 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 sequência na variável `num`
4. fim



O funcionamento do programa pode ser entendido também pelo diagrama ao lado.

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:

```

1 # include <stdio.h>
2 # include <stdlib.h>
3
4 int main () {
5     /* declaracoes */
6     int num; /* variavel utilizada para leitura da sequencia */
7     int quad; /* variavel que armazena o quadrado de um numero */
8
9     /* programa */
10    printf("Digite uma sequencia terminada por zero\n");
11    printf("Digite o primeiro numero: ");
12    scanf("%d", &num);
13
14    while (num != 0) {
15        /* os simbolos '!=' significam diferente */
16        quad = num * num ;
17        printf ("O quadrado de %d = %d\n", num, quad);
18        printf("Digite o proximo numero: ");
19        scanf("%d", &num);
20    }
21
22    /* fim do programa */
23    return 0;
24 }

```

4.3.2 Exemplo 2

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

Solução:

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

2 3 - 4 5 0

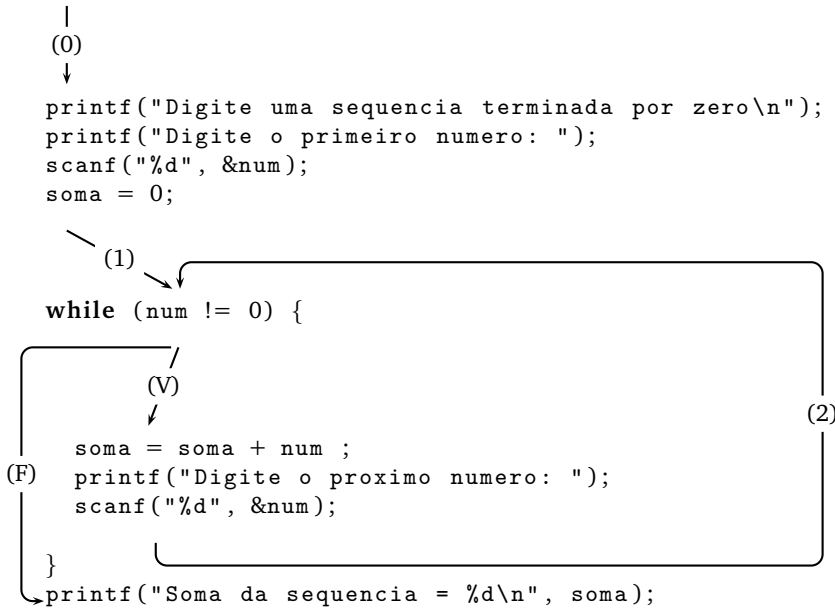
a saída de seu programa deve ser 6 (ou seja, $2 + 3 - 4 + 5$).

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 sequência, ser somada com mais esse número, até o final da sequência. Assim, para o exemplo acima, o valor de soma torna-se 2 após processar o primeiro elemento da sequência (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 sequência 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 sequência 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:

```

1  # include <stdio.h>
2  # include <stdlib.h>
3  int main () {
4      /* declaracoes */
5      int num; /* variavel utilizada para leitura da sequencia */
6      int soma; /* variavel que armazena a soma da sequencia */
7
8      /* programa */
9      printf("Digite uma sequencia terminada por zero\n");
10     printf("Digite o primeiro numero: ");
11     scanf("%d", &num);
12
13     while (num != 0) {
14         soma = soma + num ;
15         printf("Digite o proximo numero: ");
16         scanf("%d", &num);
17     }
18
19     printf("Soma da sequencia = %d\n", soma);
20
21     /* fim do programa */
22     return 0;
23 }

```

4.3.3 Exercícios Recomendados

1. (exercício 4 da lista) Dados números inteiros n e k , com $k \geq 0$, 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 $n \geq 0$, 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>.