9 Dicas de Programação

Ronaldo F. Hashimoto e Leliane N. de Barros

Este texto contém algumas dicas de programação para resolução de exercícios do curso de Introdução à Programação.

Ao final dessa aula você deverá saber:

- Definir o conceito de padrão em computação
- Utilizar os padrões computacionais: **padrão sequência númerica lida**, **padrão sequência numérica gerada**, **padrão sequência numérica selecionada** e **padrão sequência numérica alternadamente selecionada**.

9.1 Sequências

A solução da maioria dos exercícios de Introdução à Programação envolve gerar e/ou ler pelo teclado uma sequência numérica. Exemplos:

1. Uma sequência de números inteiros diferentes de zero, terminada por um zero: `2, -3, 7, 1, 2, 0`.
2. A sequência dos números inteiros de `1` a `n`: `1,2,...,n`.
3. Uma sequência com `n>0` números inteiros: para `n=5`, a sequência `-2, 3, 0, -2, 7`.

Note que uma sequência numérica pode ser **gerada** pelo seu programa ou **lida pelo teclado** (digitada pelo usuário). Nos Exemplos 1 e 3 acima, as sequências podem ser lidas pelo teclado; enquanto que no Exemplo 2, a sequência pode ser gerada.

Considere agora os seguintes exercícios de Introdução à Computação:

1. **Exercício:** Dada uma sequência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma.

   Neste exercício a sequência numérica é uma "sequência de números inteiros diferentes de zero que termina com o número zero" (por exemplo: `2, -3, 7, 1, 2, 0`) que é fornecida pelo usuário, ou seja, o seu programa deve ler esta sequência, número a número, pelo teclado. Note que o número zero não faz parte da sequência; ele somente indica o seu término.

   Dizemos neste caso que a sequência é lida pelo teclado.

2. **Exercício:** Dado um número inteiro `n>0`, determinar a soma dos dígitos de `n`. Por exemplo, a soma dos dígitos de 63453 é 21.

   Neste exercício, a sequência numérica é composta pelos dígitos de `n`. Neste caso, o seu programa deve ler pelo teclado o número `n` e a partir dele gerar cada número da sequência (um dígito de `n`) e acumulá-lo em uma soma.

3. **Exercício:** Dado um inteiro `n>0`, calcular a soma dos divisores positivos de `n`.

   Note que neste exercício, a sequência numérica é composta pelos divisores positivos de `n`. Neste caso, o seu programa deve ler `n` pelo teclado, gerar cada número da sequência e acumulá-lo em uma soma.

Em resumo,

> para resolver um problema de Introdução à Computação, você tem que ter a habilidade de identificar que sequência númerica seu programa tem que gerar ou ler pelo teclado.

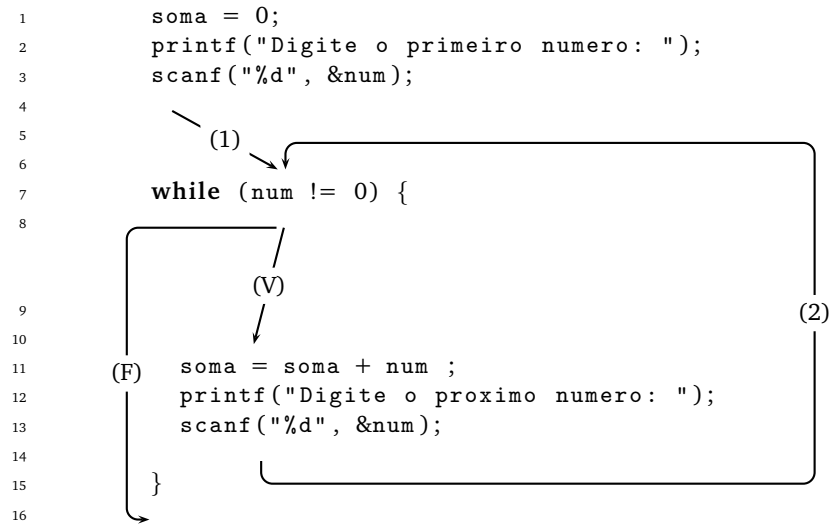
Mais exemplos:

1. **Exercício:** Dado um número inteiro `n>=0`, determinar o seu fatorial.
   Neste exercício, a sequência numérica é composta pelos números inteiros `1,2,3,...,n`.
2. **Exercício:** Dados dois inteiros `x` e `n>0`, calcular a x^n.
   Note que neste exercício, a sequência numérica é a sequência composta por `n` números: $x, x, \ldots, x$.

9.2 Geração e/ou Leitura de uma Sequência Numérica

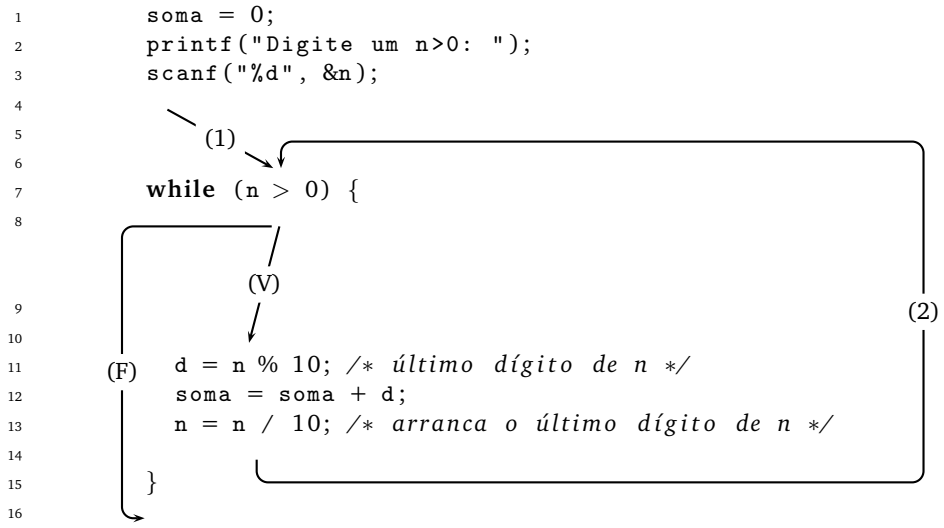
Para gerar e/ou ler uma sequência numérica o seu programa deve usar um comando de repetição. Exemplos:

1. **Exercício:** Dada uma sequência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma. Neste caso, a sequência deve ser **lida do teclado** usando o comando `scanf` (na linha 13) dentro de uma repetição `while` (linha 7):



   É utilizado o comando de repetição `while` para ler uma sequência de números inteiros pelo teclado e a cada número lido, este número é acumulado em uma variável soma.

2. **Exercício:** Dado um inteiro `n>0`, calcular a soma dos dígitos de `n`. Neste caso, a sequência deve ser **gerada** pelo programa (na linha 11) dentro de uma repetição `while` (linha 7):



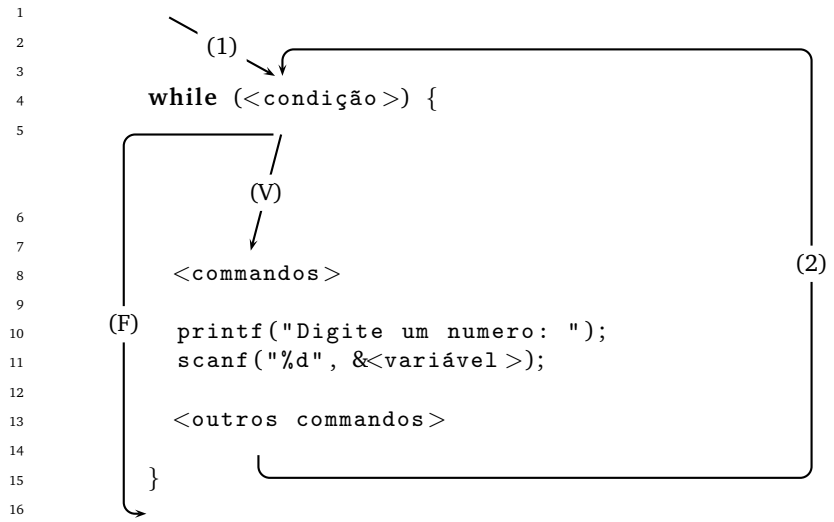
É utilizado o comando de repetição `while` para gerar a sequência dos dígitos de `n` e a cada dígito "descascado", este número é acumulado em uma variável soma.

Em resumo,

> para gerar e/ou ler pelo teclado uma sequência númerica, seu programa deve utilizar um comando de repetição.

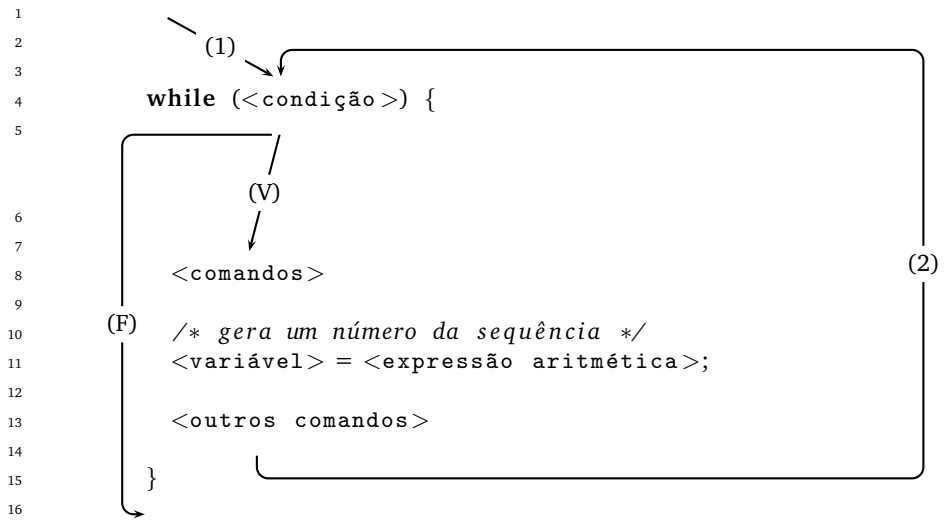
9.3 Padrão de Programação

Um padrão de programação é um trecho de código que tem uma utilidade bem definida e pode ser reutilizado. Por exemplo, o seguinte trecho de código é um padrão para ler uma sequência númerica pelo teclado:



Toda vez que você precisar ler uma sequência numérica pelo teclado, você vai utilizar um trecho de código como o apresentado anteriormente: uma repetição (`while` na linha 4) e um comando de leitura (`scanf` na linha 11) dentro desta repetição. Vamos chamar este padrão de **padrão sequência númerica lida**.

O padrão de programação para gerar uma sequência numérica é dado a seguir:



Toda vez que você precisar gerar uma sequência numérica, você vai utilizar um trecho de código como o apresentado anteriormente: uma repetição (`while` na linha 4) e um comando de atribuição que armazena em uma variável o resultado de uma expressão aritmética (`scanf` na linha 11) dentro desta repetição. Vamos chamar este padrão de **padrão sequência numérica gerada**.

Em resumo,

> apresentamos dois padrões de programação, um para ler uma sequência numérica pelo teclado e outro para gerar uma sequência numérica.

9.3.1 Exercícios que Usam Estes Padrões

1. Dado um número inteiro positivo `n`, calcular a soma dos `n` primeiros números naturais.
2. Dados um inteiro `x` e um inteiro não-negativo `n`, calcular x^n.
3. Dado um inteiro não-negativo `n`, determinar `n!`
4. Um matemático italiano da idade média conseguiu modelar o ritmo de crescimento da população de coelhos através de uma sequência de números naturais que passou a ser conhecida como sequência de Fibonacci. O `n`-ésimo número da sequência de Fibonacci F_n é dado pela seguinte fórmula de recorrência:

$$\begin{cases} F_1 = 1 \\ F_2 = 1 \\ F_i = F_{i-1} + F_{i-2} \quad para \quad i \geq 3. \end{cases}$$

   Faça um programa que, dado `n>0`, calcula F_n.

5. Qualquer número natural de quatro algarismos pode ser dividido em duas dezenas formadas pelos seus dois primeiros e dois últimos dígitos.

   Exemplos:

   - `1297`: `12` e `97`.
   - `5314`: `53` e `14`.

   Escreva um programa que imprime todos os milhares (4 algarismos) cuja raiz quadrada seja a soma das dezenas formadas pela divisão acima.

   Exemplo: raiz de `9801` = `99` = `98` + `01`.

   Portanto `9801` é um dos números a ser impresso.

9.4 Uso dos Comandos de Seleção

Por enquanto, falamos somente do uso do comandos de repetição. E os comandos de seleção (`if` ou `if-else`), como podem ser usados? Para ilustrar a forma de como estes comandos podem ser utilizados, considere o seguinte problema:

Exercício: Dizemos que um inteiro positivo `n` é perfeito se for igual à soma de seus divisores positivos diferentes de `n`. Exemplo: 6 é perfeito, pois 1+2+3 = 6. O problema é: dado um inteiro positivo `n`, verificar se `n` é perfeito.

Você é capaz de detectar qual é a sequência numérica que está envolvida neste exercício?

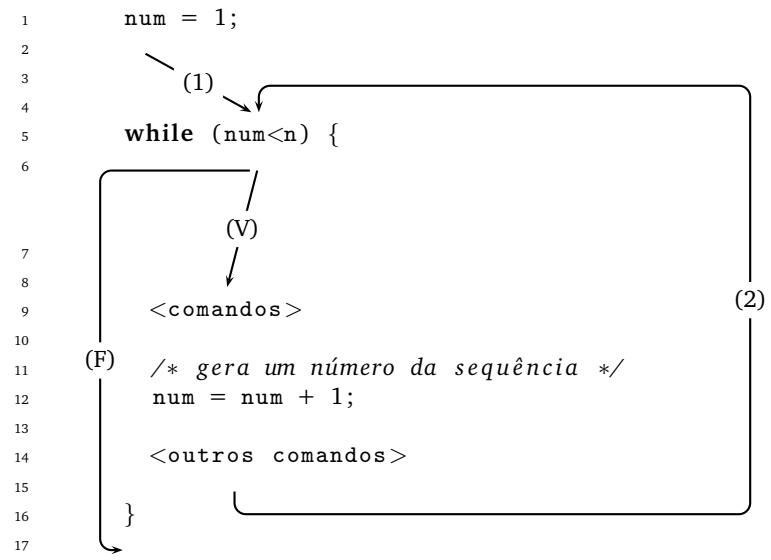
É a sequência dos divisores positivos de `n`. No exemplo acima, a sequência é: `1`, `2` e `3`. Além disso, esta sequência deve ser gerada pelo se programa. Dessa forma, vamos ter que usar o padrão **sequência numérica gerada**.

Agora, vem a pergunta: como gerar a sequência dos divisores positivos de `n`? Você pode notar que não é possível gerar diretamente esta sequência usando o padrão **sequência numérica gerada**!

Uma forma de fazer isso é gerar uma sequência numérica que contenha a sequência dos divisores de `n` e que seja facilmente obtida pelo padrão **sequência numérica gerada**. Esta sequência poderia ser a sequência dos números inteiros de `1` a `n-1`:

$$1, 2, \ldots, n-1$$

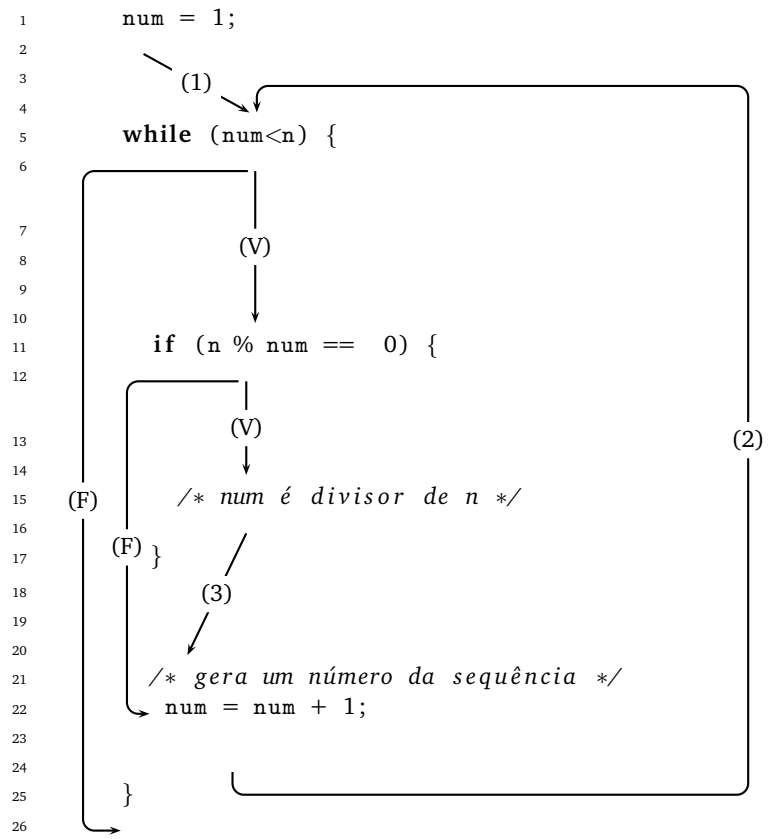
que pode ser facilmente gerada usando o seguinte trecho de código:



Observe bem este trecho de programa. A variável `num` guarda cada número da sequência gerada. Se o conteúdo da variável `num` fosse imprimida dentro do laço, seria gerada a sequência de inteiros de `1` a `n-1`.

Note a condição do `while`. Quando `num` fica igual a `n`, a condição fica falsa e o fluxo do programa sai da repetição (seguindo a seta marcada com `(F)`), garantindo que dentro do laço nunca o conteúdo da variável `num` fica igual ao contéudo da variável `n`. Note também que imediatamente depois que o fluxo do programa sai do laço, o contéudo da variável `num` é sempre igual ao conteúdo da variável `n`, pois caso contrário, o fluxo do programa continuaria dentro do laço.

Depois usando um comando de seleção `if`, "selecionar" os números da sequência gerada que são divisores de `n` da seguinte forma:



Observe então que a repetição (`while` da linha 5) é usada para gerar a sequência de inteiros de `1` a `n-1` e o comando de seleção (`if` da linha 11) é usado para selecionar os números que são divisores de `n`.

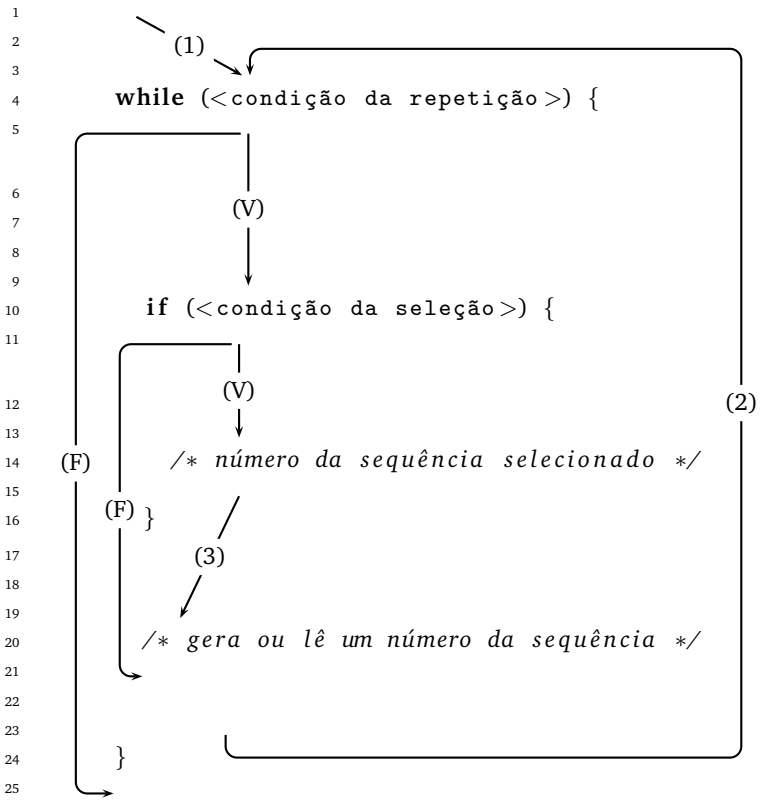
Agora, precisamos acumular em uma variável a soma dos divisores de `n` e depois (fora do laço) testar se a soma final é igual a `n`. Desta forma, teríamos:

```
1       num = 1;
2       soma = 0;
3       while (num<n) {
4         if (n % num ==  0) {
5           /* num é divisor de n */
6           soma = soma + num;
7         }
8         /* gera um número da sequência */
9         num = num + 1;
10      }
11      if (soma == n) {
12        printf ("%d eh um numero perfeito\n", n);
13      }
14      else {
15        printf ("%d nao eh um numero perfeito\n", n);
16      }
```

Fica como tarefa para você escrever o programa completo que seja solução para este exercício!

9.5 Padrão Sequência Numérica Selecionada

Note que podemos então ter um novo padrão de programação que chamamos de **padrão sequência numérica selecionada**:



9.5.1 Exercícios que Usam este Padrão

1. Dado um número inteiro positivo `n`, imprimir os `n` primeiros naturais ímpares. Exemplo: Para n=4 a saída deverá ser 1,3,5,7.

2. Uma loja de discos anota diariamente durante o mês de março a quantidade de discos vendidos. Determinar em que dia desse mês ocorreu a maior venda e qual foi a quantidade de discos vendida nesse dia.

3. Dados `n>0` e uma sequência de `n` números inteiros, determinar a soma dos números pares.

4. Dados `n>0` e dois números inteiros positivos `i` e `j` diferentes de `0`, imprimir em ordem crescente os `n` primeiros naturais que são múltiplos de `i` ou de `j` e ou de ambos.

5. Dados dois números inteiros positivos, determinar o máximo divisor comum entre eles usando o algoritmo de Euclides.

   Exemplo:

	1	1	1	2	
24	15	9	6	3	$= mdc(24,15)$
9	6	3	0		

6. Dizemos que um número `i` é congruente módulo `m` a `j` se `i % m = j % m`.

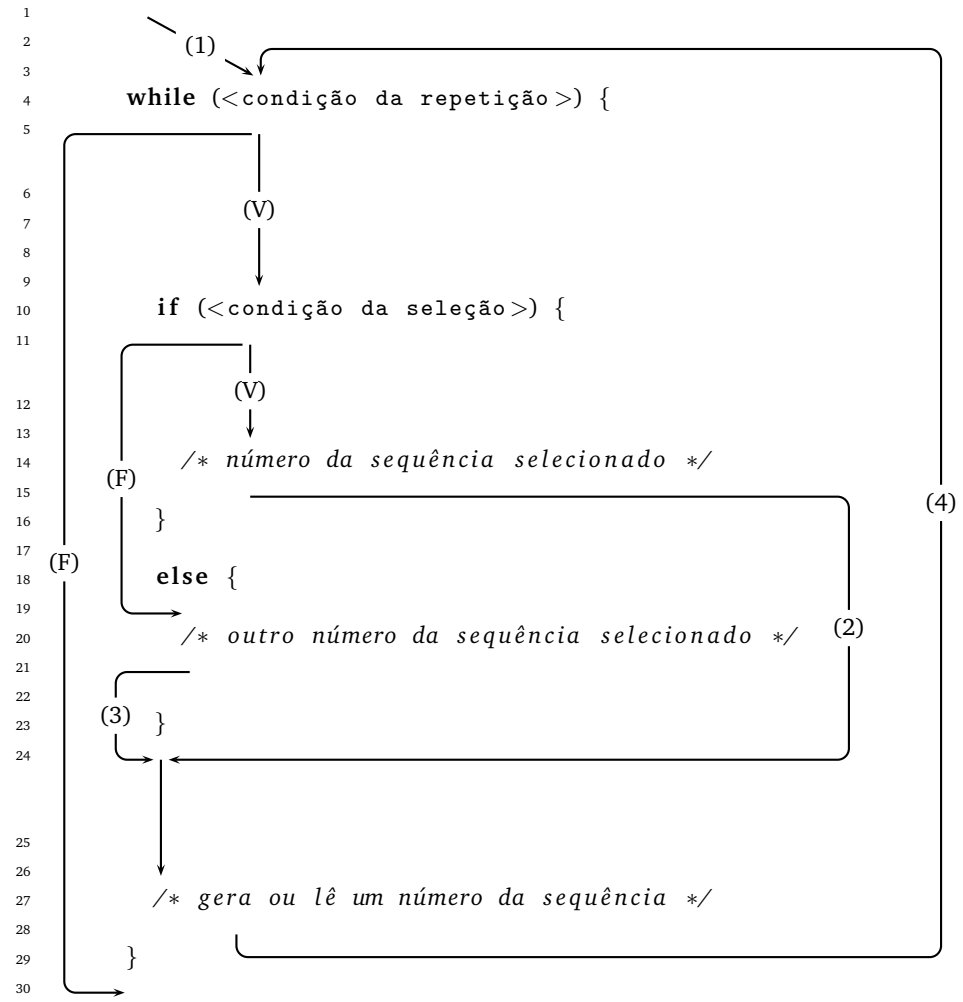
   Exemplo: `35` é congruente módulo `4` a `39`, pois `35 % 4 = 3 = 39 % 4`.

   Dados inteiros positivos `n`, `j` e `m`, imprimir os `n` primeiros naturais congruentes a `j` módulo `m`.

9.6 Padrão Sequência Numérica Alternadamente Selecionada

Note que poderíamos ter situações em que fossem necessários selecionar números de uma sequência alternadamente. Por exemplo, selecionar da sequência de `1` a `n` os números pares e os números ímpares (ou os números são divisores de `3` e os que não são) para detectar quantos são pares e qual é a soma dos ímpares (não sei para que saber estas informações, mas é um bom exemplo de uma sequência alternadamente selecionada).

Este padrão usaria então um comando de seleção `if-else` dentro de uma repetição `while` da seguinte forma:



9.6.1 Exercícios que Usam este Padrão

1. Dados `n>0` e uma sequência de `n` números inteiros positivos, determinar a soma dos números pares e a quantidade dos números ímpares.

2. Dado um inteiro `n>0`, determinar a quantidade de divisores positivos e pares de `n` e calcular a soma dos divisores positivos de `n`.