Fluxo de Controle

Algoritmos e Programação de Computadores

Guilherme N. Ramos
gnramos@unb.br

2017/1

Fluxo de Controle

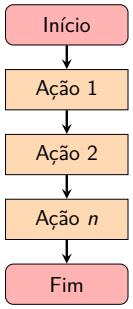
O *fluxo de controle* de um algoritmo é a ordem em que são executadas suas instruções.

Todo algoritmo computacional é baseado em 3 conceitos:

- instruções sequenciais;
- bifurcação;
- repetição.

Sequencial

Em um algoritmo, as instruções são executadas de forma sequencial, uma após a outra, na mesma ordem em que se encontram na representação do algoritmo.




```
1 Algoritmo TrocaPneuFurado
2 Início
3     AfrouxarPorcas(PNEU_FURADO)
4     SuspenderCarro()
5     RetirarPorcas(PNEU_FURADO)
6     Retirar(PNEU_FURADO)
7     Posicionar(PNEU_ESTEPE)
8     ColocarPorcas(PNEU_ESTEPE)
9     AbaixarCarro()
10    ApertarPorcas(PNEU_ESTEPE)
11 Fim
```

Esta execução ordenada é uma condição essencial para funcionamento correto da maioria dos algoritmos.

Sequencial

00-divisao.c

```c
#include <stdio.h> /* Biblioteca de E/S */

int main() {
    float numerador, denominador; /* variáveis */

    printf("Digite o numerador: ");
    scanf("%f", &numerador);   /* Lê o numerador */
    printf("Digite o denominador: ");
    scanf("%f", &denominador); /* Lê o denominador */

    /* Só agora é possível computar o resultado */
    printf("A divisão é: %f\n", numerador/denominador);

    return 0;
}
```

Sequencial

01-horas.py

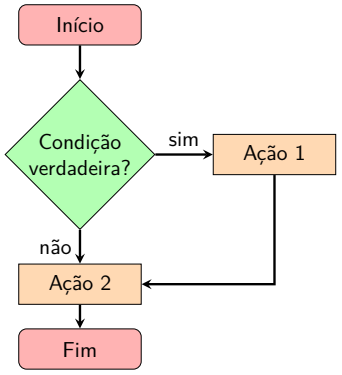
```
1 DIAS_NO_ANO = 365
2 HORAS_NO_DIA = 24
3
4 idade_em_anos = int(input('Olá! Quantos anos você tem? '))
5
6 # Agora que se tem a idade em anos, é possível calcular
7 # quantos dias já foram vividos (aproximadamente).
8 idade_em_dias = idade_em_anos * DIAS_NO_ANO
9
10 # E agora que se sabe quantos foram os dias, pode-se calcular
11 # quantas horas foram vividas (aproximadamente).
12 horas_vividas = idade_em_dias * HORAS_NO_DIA
13
14 # Só agora é possível mostrar o resultado.
15 print('Sabia que já viveu', horas_vividas, 'horas?')
```


Bifurcação

O *fluxo de controle* de um algoritmo é sequencial, mas a sequência a ser executada pode ser alterada com uma estrutura de bifurcação.

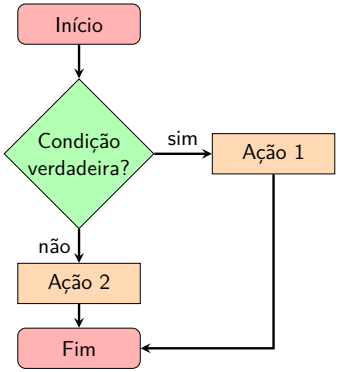
Desta forma, um conjunto de instruções é executado apenas se certa condição for verdadeira.

Simples



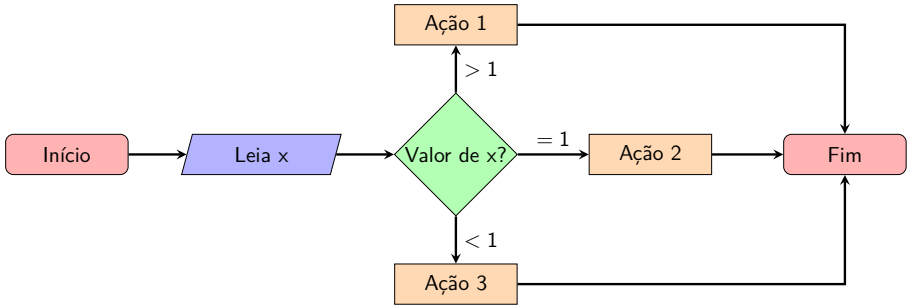

```
1 Algoritmo PreparaCafé
2 Início
3     Traga(XÍCARA)
4     Sirva(CAFÉ)
5     Se Deseja(AÇÚCAR) Então
6         Coloque(AÇÚCAR)
7         Mexa()
8     FimSe
9 Fim
```


Dupla




```
1 Algoritmo EspelhoEspelhoMeu
2 Início
3     EXISTE_MAIS_BELA = Pergunte(ESPELHO)
4     Se EXISTE_MAIS_BELA Então
5         Escreva("Sim, minha rainha!")
6         Escreva("Branca de Neve é a mais bela.")
7     Senão
8         Escreva("És a mais bela de todas as mulheres!")
9         /* SuspiroAliviado() */
10    FimSe
11 Fim
```

Múltipla



Múltipla

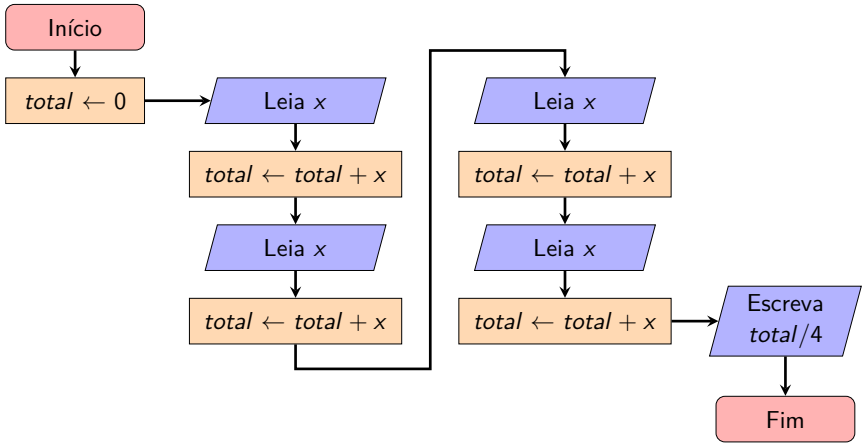
```
1 Algoritmo AjustaTemperatura
2 Variáveis
3     temperatura : real
4 Início
5     temperatura ← Leia(TERMÔMETRO)
6     Conforme temperatura Faça
7         Caso temperatura < 20
8             Vista(CASACO)
9             Tome(CHÁ)
10        Caso 20 ≤ temperatura < 25
11            Vista(BLUSA)
12        Caso 25 ≤ temperatura < 30
13            Tire(CALÇA)
14            Vista(BERMUDA)
15        OutrosCasos
16            Ligue(AR_CONDICIONADO)
17            Tome(SUCO)
18    FimConforme
19 Fim
```


Bifurcação

O custo de executar instruções sequenciais é proporcional a quantidade de instruções (cada instrução é executada, no máximo, uma vez).

Repetição

Certos problemas exigem a execução de **um** mesmo conjunto de instruções [*sequenciais*] repetidas vezes... um procedimento que ~~pode~~ deve ser automatizado!



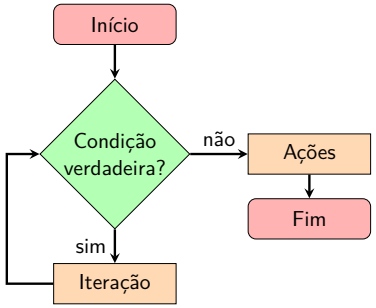
Repetição

Um *laço de repetição* contém um conjunto de instruções que são executadas (sequencialmente) enquanto determinada condição for verdadeira. Cada execução é chamada de *iteração*.

```
1  Algoritmo MédiaAritmética
2  Variáveis
3      x, total : real
4      contador : inteiro                // Variável auxiliar!
5  Início
6      total ← 0
7      contador ← 0
8      Enquanto contador < 4 Faça        // Laço de repetição
9          Leia(x)
10         total ← total + x
11         ++contador                    // Atualiza a informação
12     FimEnquanto
13     Escreva("Média = ", total/contador)
14 Fim
```


Enquanto-Faça

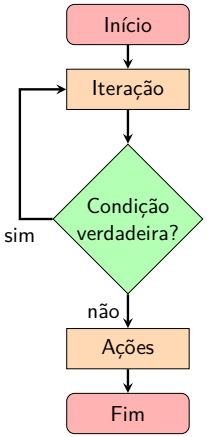
Avalia a condição e, se verdadeira, executa o laço.




```
1  Algoritmo PreparaCaféDireito
2  Início
3      Traga(XÍCARA)
4      Sirva(CAFÉ)
5      Enquanto Quer(AÇÚCAR)
   Faça
6          Coloque(AÇÚCAR)
7          Mexa()
8          // Qual a instrução
9          // mais importante?
10     FimEnquanto
11 Fim
```


Faça-Enquanto

Avalia a condição de parada *após a execução do laço*.




```
1  Algoritmo LanchaDireito
2  Variáveis
3    resp : caractere
4  Início
5    Escreva("Boa tarde! Eis seu chá.")
6    Sirva(CHA)
7    Faça
8      Sirva(BISCOITO)
9      Escreva("Quer mais um biscoito?")
10     Leia(resp)
11   Enquanto(resp = 's' Ou resp = 'S')
12 Fim
```


Atenção!

...e viva o `Ctrl+c`!

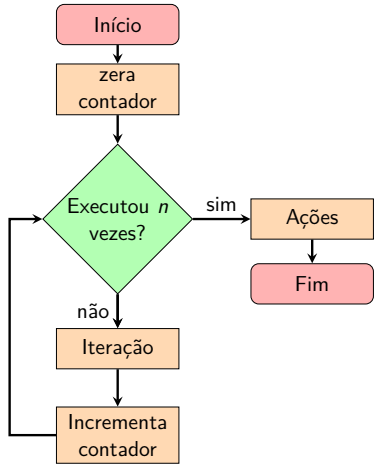
Perigo! Perigo!

É preciso ter muito cuidado ao usar laços de repetição:

1. implementar um laço infinito;
2. execução do um número *correto* de vezes;
3. volte a instrução 1.

Para-Até-Faça

Sabendo *quantas são as iterações*, pode-se simplesmente definir um laço que seja executado tantas vezes.




```
1 Algoritmo ContaGotasDeRemédio
2 Variáveis
3     i : inteiro
4 Início
5     Para i ← 1 Até 10 Faça
6         Pingue(REMÉDIO)
7         i ← i + 1;
8     FimPara
9 Fim
```


Para-Até-Faça

Os laços de repetição são "equivalentes"...

- Inicialização das condições de parada.
- Teste das condições de parada.
- Atualização da condições de parada.

```
1 for(/* 1 */; /* 2 */; /* 3 */) {
2     /* Instruções */
3 }
```

```
1 /* 1 */
2 while(/* 2 */) {
3     /* Instruções */
4
5     /* 3 */
6 }
```


Para-Até-Faça

Os laços de repetição são "equivalentes"...

- Inicialização das condições de parada.
- Teste das condições de parada.
- Atualização da condições de parada.

```
1 for x in range(a, b):
2     # Instruções
```

```
1 x = a
2 while(x < b):
3     # Instruções
4
5     x = x + 1
```


Repetições

O custo de executar instruções sequenciais é proporcional a quantidade de instruções, e do valor das variáveis de controle.

Repetições

Karaokê

06-incomodam

Um elefante incomoda muita gente, N+1 elefantes incomodam quanto mais? Implemente o código que recebe a quantidade de elefantes e escreva a letra da música.

Repetições

Método da Bissecção

```
1 Faça
2     r ← (a + b)/2
3     Se r*r < n Então
4         a ← r
5     Senão
6         b ← r
7     FimSe
8 Enquanto r*r ≠ n /*?*/
```