**Texto base**

**12**

**Exercícios**

Gilberto Alves Pereira

***Resumo***

*Sabemos que para absorver os conceitos envolvendo matemática ou lógica necessitam de prática. Sem dúvida alguma o exercício da manipulação das ferramentas envolvendo a lógica de programação é de fundamental importância no aprendizado dessa disciplina. Dedicamos este capítulo na discussão de 5 exercícios resolvidos, nos quais utilizamos as ferramentas aprendidas neste curso. Ao final é mostrado um deles convertido para Python.*

**1.1. Introdução**

Variáveis, comandos de atribuição, precedência de operadores aritméticos e lógicos, estruturas de seleção e repetição, contadores, acumuladores, vetores, funções, parâmetros, retorno de funções, algoritmos. Existem muitas ferramentas à nossa disposição para desenvolvermos nossos algoritmos. Vamos usá-las em nossos exercícios a seguir.

**Exemplo 1-** Escrever um fluxograma de uma função que retorna a soma um vetor de pesos. Criar também o programa que cria o vetor e chama a função.

|  | | **Programa Principal** | | | | | | --- | --- | --- | --- | --- | | **X** | **peso** | | | **Tela** | | **[0]** | **[1]** | **[2]** | | 60 | 10 | 20 | 30 | A soma é 60 | |  |  |  |  |  |  | **Função somar** | | | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | | **soma** | **Explicação soma** | **i** | **i < n** | **v** | | | **n** | | **[0]** | **[1]** | **[2]** | | 0 | soma = 0 | 0 | 0 < 3 V | 10 | 20 | 30 | 3 | | 10 | soma = soma + v[0]  soma = 0 + 10 | 1 | 1 < 3 V |  |  |  |  | | 30 | soma = soma + v[1]  soma = 10 + 20 | 2 | 2 < 3 V |  |  |  |  | | 60 | soma = soma + v[2]  soma = 30 + 30 | 3 | 3 < 3 F |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | | **Valor a retornar:** 60 | | | | | | | |   Note que os dados do vetor **peso** (do programa) são passados para o vetor **v** (da função somar). O segundo parâmetro da função é o tamanho do vetor (3 elementos neste exemplo).  A cada iteração da função somar o comando **soma = soma+v[i]** acumula um valor do vetor v à variável soma. Quando **i = 0** então **v[i]** tem o valor **10**, quando **i = 1** então **v[i]** tem o valor **20** e assim sucessivamente até o último elemento do vetor **v**. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Figura 12.1. Simulação, Fluxograma e Função que retorna a soma dos valores de um vetor de pesos. Fonte Autor.**

**Exemplo 2 -**Escrever um fluxograma que lê um vetor de inteiros com 10 elementos. Elaborar também uma função que calcula e retorna a média dos elementos pares de um vetor passado por parâmetro. Exibir a média.

|  |  |
| --- | --- |

**Figura 12.2. Fluxograma e Função que retorna a média dos elementos pares de um vetor passado por parâmetro. Fonte Autor.**

Observe que nesse exemplo foram passados como parâmetros para a função um vetor e o tamanho do vetor (n). Embora existam em muitas linguagens de programação funções que devolvem o tamanho do vetor (listas em Python), a utilização de um parâmetro com a quantidade de elementos é muito interessante pois quando usamos essas funções que devolvem o tamanho do vetor isso nos obriga a varrer todo o vetor e muitas vezes estamos utilizando apenas um pedaço do vetor, não ele por completo. Essa alternativa de passar a quantidade de elementos usados do vetor por parâmetro é muito válida.

**Exemplo 3 -** Escrever um fluxograma que lê um vetor de caracteres com 10 elementos. Elaborar também uma função ***contaletrasm(vet,letra)*** que calcula e retorna o número de letras do vetor ***vet*** que são maiores que ***letra*** (também passado por parâmetro). Exibir o retorno da função ***contaletrasm***.

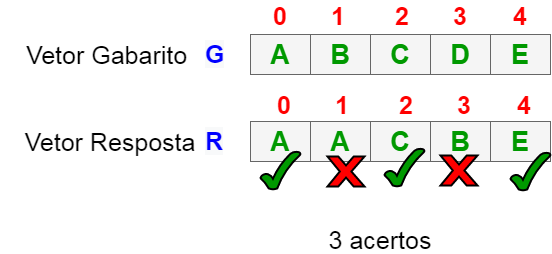
|  |  |
| --- | --- |

**Figura 12.3. Fluxograma e Função que recebe como parâmetro um vetor e uma letra e retorna o número de vezes que a letra é encontrada no vetor. Fonte Autor**

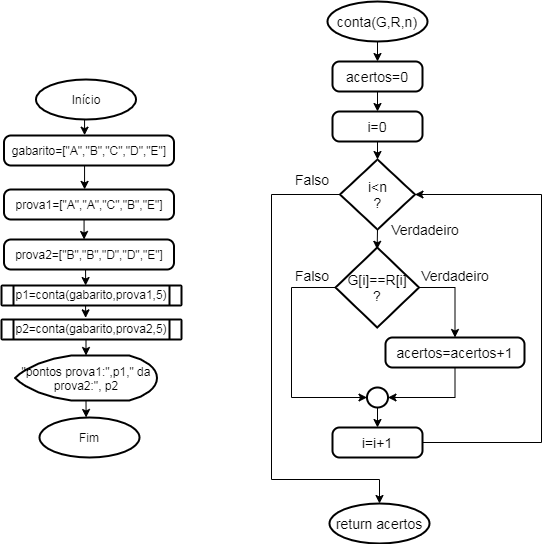
Veja que nesse exemplo utilizamos a função len() para obter o tamanho do vetor.

**Exemplo 4 -** Criar uma função que recebe como parâmetro 2 vetores G gabarito e R resposta relativos a questões de uma prova e retorna a quantidade de acertos.

Nesse exemplo cada elemento do vetor representa a resposta de uma questão. A contagem consiste na seguinte verificação: cada posição dos vetores corresponde a uma pergunta. Portanto a posição 0 corresponde à primeira pergunta. O valor dessa posição no vetor Gabarito corresponde a resposta correta e o valor dessa posição no vetor resposta corresponde a resposta do aluno. Caso seus conteúdos sejam iguais, a resposta está correta (Figura 12.4). O Fluxograma deve fazer esses teste para todas as posições do vetor e a cada acerto efetuar a contagem (Figura 12.5).



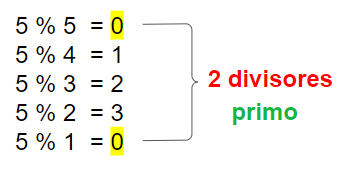
**Figura 12.4. Funcionamento da contagem das respostas. Fonte Autor.**



**Figura 12.5. Fluxograma do programa e função que conta os acertos de uma prova teste. Fonte Autor.**

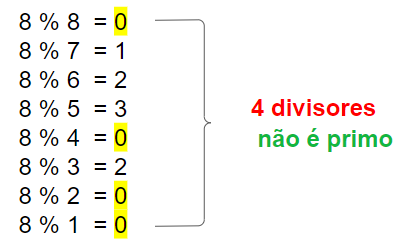
**Exemplo 5 -** Desenhar um Fluxograma de uma função que receba um número inteiro e retorne 1 caso o número seja primo e zero caso contrário

Primeiramente vamos entender como podemos calcular os divisores de um número e verificar se o número é primo. Veja a figura 12.6. Para verificarmos quais os divisores de 5 vamos calcular o resto da divisão de 5 por todos os números inferiores ou iguais a 5 até o número 1. Verificamos que teremos 5 divisões a serem feitas para os números 5,4,3,2 e 1. Percebemos que de todas essas divisões apenas as divisões pelo próprio número (5) e por 1 são exatas. Essa é a definição de número primo. Todo número primo é divisível apenas por ele mesmo e por 1 (todos os números são divisíveis por 1). Todos os números primos possuem apenas 2 divisores, logo 5 é primo porque possui e divisores.



**Figura 12.6. Cálculo dos divisores de 5 Fonte Autor.**

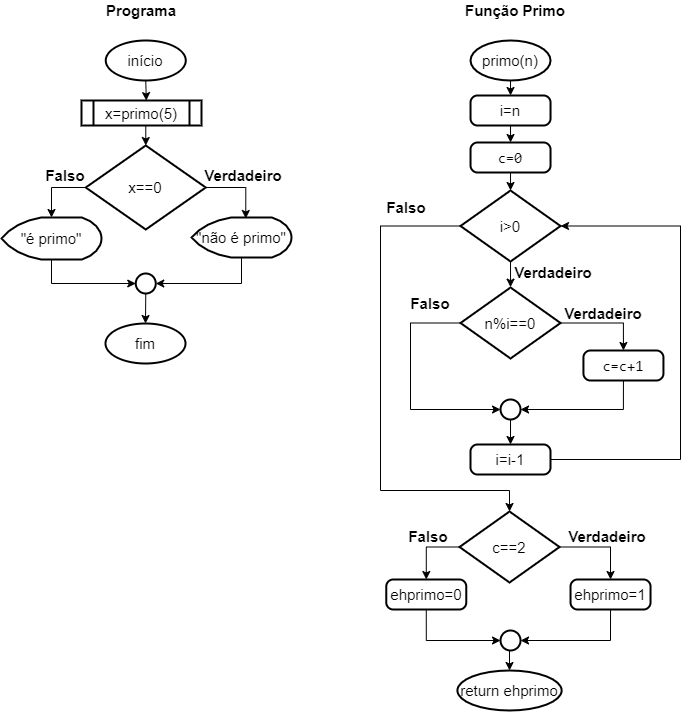
Vejamos o mesmo cálculo para o número 8. Figura 12.7. Nesse caso percebemos que 8 possui 4 números que dividem 8 de forma exata (resto 0) 4 divisores. Como ele tem mais de 2 divisores portanto ele não é primo. Esse será o algoritmo que iremos utilizar. Vamos verificar para todos os números iguais ou inferiores ao número que queremos efetuar o teste se é primo, quantos dividem esse número de forma exata. Caso seja 2 é primo. Caso o número de divisores seja maior que 2 o número não é primo.



**Figura 12.7. Cálculo dos divisores de 8 Fonte Autor.**

Na Figura 12.8 podemos observar os fluxogramas do programa e da função **primo(n)**. A esquerda, no fluxograma do programa, a função primo é chamada **x=primo(5)**. Perceba que o parâmetro da função é exatamente o número que queremos verificar se é primo. A variável x recebe o retorno dessa função. Se após a execução da função o valor recebido pela variável x for 0 significa que o número passado como parâmetro para a função não é primo. Caso contrário o número é primo. Ao final do programa foi colocada uma estrutura de seleção para exibir uma mensagem adequada para cada caso.

Vamos observar a função **primo(n)** agora. Nesta função usamos a variável **c** para contar a quantidade de divisores de **n**. Usamos também uma variável de controle **i** de um loop. Nesse loop, a variável de controle **i** começa valendo **n** e a cada passo é decrementada até atingir o valor 0 quando o loop é interrompido. A cada valor de **i** verifica-se se esse valor é divisor de n usando o teste: **n%i==0**. Se o teste for verdadeiro, **c** é incrementada indicando que encontramos mais um divisor de **n**. Ao final, o valor da variável **c** indica quantos divisores foram encontrados para **n**. É feito então um tratamento usando a estrutura de seleção de maneira que caso o número seja primo (isso ocorre quando **c** é igual a 2) o retorno será 1, caso contrário o retorno será 0.

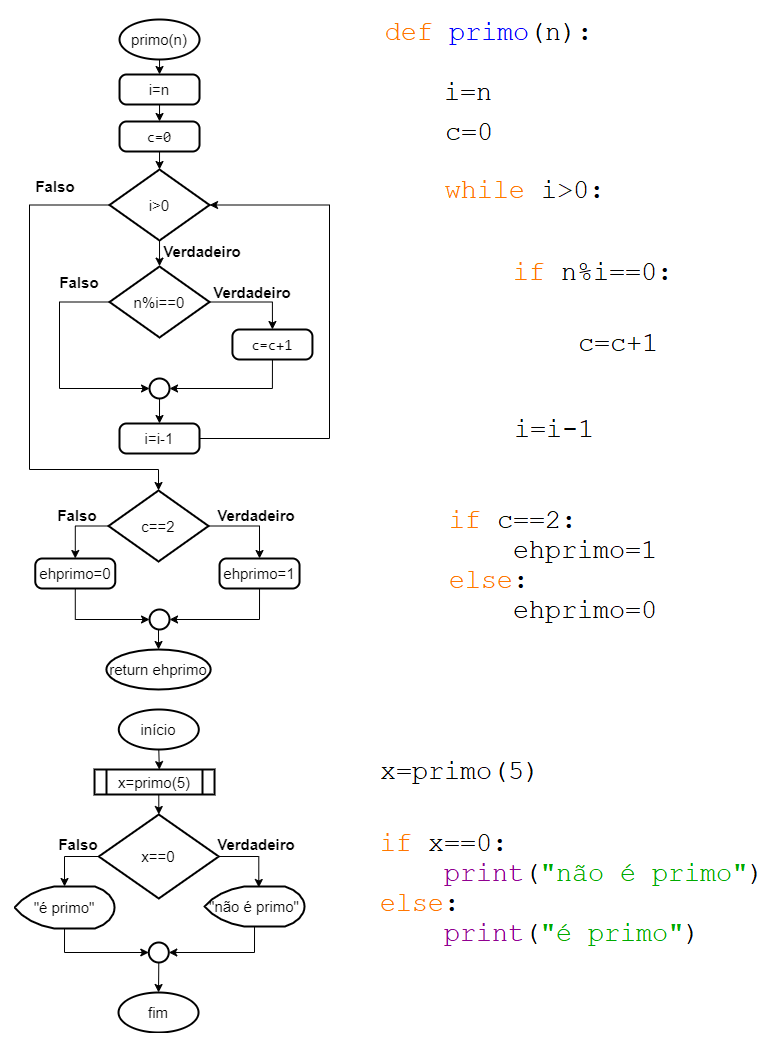


**Figura 12.8. Fluxograma do programa e função que conta os acertos de uma prova teste. Fonte Autor.**

**1.2. Comparativo entre fluxograma e Python**

Abaixo podemos identificar um fluxograma da função e programa do exemplo 5 - verificação se um número é primo e sua conversão para Python.

| **Fluxograma** | **Python** |
| --- | --- |

****

**Figura 12.9. Fluxograma e seu respectivo código em Python do exemplo 5 verificação de é primo. Fonte Autor.**

**1.3. Referências**

DIERBACH, C. “Introduction to Computer Science Using Python: A Computational Problem-Solving Focus”1st Edition, New York: Wiley, 2012.

Python Tutor - <http://www.pythontutor.com/visualize.html>