Programação de Computadores I – BCC701

Conteúdo Teórico em Linguagem Python

Módulo 3 – Entrada e Saída

2020/1

Elaborado pela Comissão de Unificação da Disciplina BCC701,
com a colaboração de professores e estagiários docentes
http://www.decom.ufop.br/bcc701/

DECOM – Departamento de Computação
ICEB – Instituto de Ciências Exatas e Biológicas
UFOP – Universidade Federal de Ouro Preto

Sumário

MÓDULO

Entrada e Saída

Normalmente, em um programa de computador, é necessário interagir com o usuário. Em algum momento é necessária a inserção de dados, comumente definidos como **entradas** do usuário, que posteriormente serão manipulados por um conjunto de operações definidas no programa. Depois, com os dados obtidos através das operações executadas, o resultado poderá ser apresentado como um retorno ou uma **saída** para o usuário do programa. O fluxograma da Figura 1 ilustra o cenário descrito com as principais etapas de um programa que fornece instruções de entrada e saída.

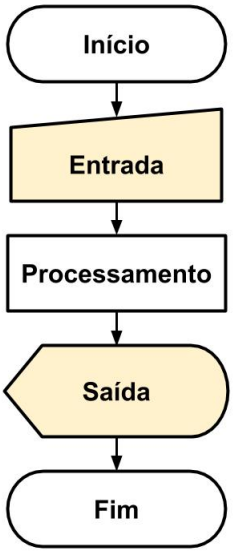

Figura 1: Fluxograma de exemplo para entrada e saída.

Na linguagem de programação **Python**, são fornecidos mecanismos para a **entrada** e **saída** através do *terminal*, a função **input()** realiza a entrada e a função **print()** realiza a saída, e são apresentadas a seguir. Por uma questão didática, começaremos pela **saída** e em seguida a **entrada**.

3.1 Saída: função print

A **saída** de dados em **Python**, como descrito anteriormente, através de sua interface padrão, é feita por meio da função **print**.

Abaixo descrevemos a sintaxe da função **print**:

`print(`*mensagem*`)`

onde:

- **print**: nome da função responsável por exibir mensagens como saída para o usuário.
- *mensagem*: argumento de entrada para a função. É uma **string** (**valor textual**) que será exibida como mensagem no terminal. Normalmente é uma **string** gerada a partir de informações de variados tipos, conforme veremos a seguir.

O exemplo seguinte apresenta o uso da função **print** gerando como saída para o usuário a frase `"Olá Mundo!"`.

```
1 frase = 'Olá Mundo!'
2 print(frase)
```

3.1.1 Strings de formatação

Em muitos cenários, um conjunto heterogêneo de dados poderá ser apresentado como saída ao usuário. Neste sentido, a linguagem **Python** oferece vários recursos para a formatação e posteriormente a apresentação destes dados. Um destes recursos consiste no uso de uma forma especial de **literais string** onde é possível intercalar valores de expressões no meio do texto da **string**.

Uma **string de formatação** (ou simplesmente **f-string**) é escrita colocando-se a letra **f** imediatamente antes do delimitador que começa a **string** (aspas ou apóstrofo). Por exemplo:

`f'Olá, mundo!'`

Em uma **f-string**, os caracteres **abre chave** (**{**) e **fecha chave** (**}**) são especiais, e são usados para intercalar valores de expressões na **string**. Basta escrever a expressão desejada entre chaves dentro da **f-string**, e o **Python** se encarregará de avaliar a expressão e inserir o seu valor no meio dela, onde o programador colocou as chaves. Por exemplo:

`f'A soma de 2 e 3 é {2+3}, certo?'`

Nesta **f-string** a expressão entre chaves (**2+3**) é avaliada e o seu resultado (**5**) é inserido na **string**, substituindo a expressão e as chaves. O texto gerado é:

`'A soma de 2 e 3 é 5, certo?'`

Podemos usar strings de formatação como argumentos para a função **print** para produzirmos saídas de dados formatadas em nossos programas.

Exemplo 1:

Faça um programa que pergunta ao usuário o seu nome, e exibe uma saudação de boas-vindas, incluindo o nome na saudação.

```
1 nome = input('Qual é o seu nome? ')
2 print(f'Bom dia {nome}, seja bem vindo(a)!')
```

Exemplo 2:

Escreva um programa que exibe a média aritmética de dois números disponíveis em duas variáveis.

```
1 x = 5.6
2 y = 8.8
3 media = (x + y) / 2
4 print(f'A média aritmética de {x} e {y} é {media}')
```

3.1.2 Especificando a formatação

O **Python** permite a indicação de como os valores das expressões devem ser formatados ao serem inseridos em uma **f-string**. Para tanto basta colocar, imediatamente após a expressão, antes de fechar as chaves, o caractere dois pontos ("**:**") seguido da indicação de formatação desejada.

Uma **especificação de formatação simplificada** (*existem opções avançadas*) consiste em indicar:

1. (*opcional*) a **largura** do campo (quantidade mínima de caracteres a ser usada);
2. (*opcional*) o caractere "**.**" seguido da **precisão** para números reais (quantidade de casas decimais);
3. o **tipo** do dado:
   - "**s**" para **string** (sequência de caracteres);
   - "**d**" para **decimal** (valor numérico inteiro em notação decimal);
   - "**f**" para **float** (valor numérico real).

Exemplo 3:

Para demonstrar o uso do operador de formatação observe o bloco de códigos abaixo.

```
1 cidade     =  'Ouro Preto'
2 celsius    =  32
3 fahrenheit =  89.60
4 print(f'A temperatura em {cidade}:')
5 print(f'{celsius} Celsius')
6 print(f'{fahrenheit:.2f} Fahrenheit')
```

Neste código foram definidas e inicializadas três variáveis, nas linhas 1 a 3, que armazenam valores de tipos distintos de dados. Na linha 4, a função **print**, como visto anteriormente, tem como parâmetro uma **f-string** correspondente à mensagem a ser exibida. Observe a ocorrência da variável **cidade** entre chaves – o seu valor será inserido na mensagem. Na linha 5, a função **print** também é usado com uma **f-string** na qual é inserido o valor da variável **celsius**. Já na linha 6, o valor da variável **fahrenheit** é inserido em uma **f-string** usando duas casas decimais.

Exemplo 4:

O programa a seguir ilustra as indicações de tipo e tamanho do campo na formatação do valor inserido em uma **f-string**.

```
1  x = 1
2  y = 10
3  print('Sem formatação:')
4  print(f'x = {x}')
5  print(f'y = {y}')
6  print('Formatação para número inteiro:')
7  print(f'x = {x:d}')
8  print(f'y = {y:d}')
9  print('Formatação para número inteiro com largura mínima igual a 10:')
10 print(f'x = {x:10d}')
11 print(f'y = {y:10d}')
```

Neste programa estão declaradas duas variáveis **x** e **y**, inicializadas com valores do tipo inteiro. Observe que a especificação de formatação "**:10d**" reserva um espaço mínimo de dez caracteres para a inserção do valor da expressão. O espaço não utilizado é preenchido com espaço em branco à esquerda do número.

Exemplo 5:

O código a seguir mostra como o número de casas decimais pode ser ajustado na formatação de números reais.

```
1  x = 1.1111111111
2  y = 1.88888888
3  z = 2.0
4  print('Sem formatação:')
5  print(f'x = {x}')
6  print(f'y = {y}')
7  print(f'z = {z}')
8  print('Formatação para número real:')
9  print(f'x = {x:f}')
10 print(f'y = {y:f}')
11 print(f'z = {z:f}')
12 print('Formatação para número real com largura mínima igual a 10:')
13 print(f'x = {x:10f}')
14 print(f'y = {y:10f}')
15 print(f'z = {z:10f}')
16 print('Formatação para número real com largura mínima igual a 10 e
       precisão igual a 3:')
17 print(f'x = {x:10.3f}')
18 print(f'y = {y:10.3f}')
19 print(f'z = {z:10.3f}')
```

Este programa faz uso de três variáveis **x**, **y** e **z** para ilustrar a formatação de números reais. A especificação de formatação "**:10.3f**" reserva um espaço mínimo de largura **dez** para inserir um número real, que será formatado com **três** casas decimais.

Exemplo 6:

Para ilustrar a formatação **strings** usamos o seguinte programa.

```
1  oi = 'Olá Mundo'
2  tchau = 'Adeus mundo cruel'
3  print('Sem formatação:')
4  print(f'{oi}!!!')
5  print(f'{tchau}!!!')
6  print('Formatação para string:')
7  print(f'{oi:s}!!!')
8  print(f'{tchau:s}!!!')
9  print('Formatação para string com largura mínima igual a 25:')
10 print(f'{oi:25s}!!!')
11 print(f'{tchau:25s}!!!')
```

A especificação de formatação "**:25s**" usada no exemplo insere a **string** em um espaço de largura mínima de **25** caracteres.

3.1.3 Inserção de caracteres especiais em uma string

Em algumas situações desejamos incluir **caracteres especiais**, geralmente que não tem uma representação gráfica, em um literal **string**. **Python** permite o uso de **sequências de escape** dentro de uma **string** para a inclusão destes caracteres.

Uma sequência de escape é formada pelo caractere "\" seguido de um código que identifica o caractere especial a ser incluído na **string**.

Algumas sequências de escape são listadas a seguir:

- \n: Representa o caractere de nova linha. Quando digitado no programa faz com que o cursar avance para a próxima linha naquela posição.
- \t: Representa o caractere de tabulação horizontal. Quando digitado no programa faz com que o cursor avance para a próxima parada de tabulação. Normalmente as paradas de tabulação ocorrem a cada oito posições.
- \": Representa o caractere aspas. Deve ser usado para inserir aspas em um literal **string** delimitado por aspas.
- \': Representa o caractere apóstrofo. Deve ser usado para inserir apóstrofo em um literal **string** delimitado por apóstrofos.
- \\: Representa o caractere barra inclinada para a esquerda. Deve ser usado para inserir a barra, pois a barra por si só já é especial, uma vez que é usada para começar uma sequência de escape.
- Se um literal **string** é prefixado com as letras "r" ou "R" este mecanismo de sequências de escape é desativado, e todos os caracteres são inseridos literalmente na **string**.
- Em um literal **f-string** os caracteres "{" e "}" são especiais e indicam a inserção do valor de uma expressão. Para inserirmos estes caracteres na **string** precisamos escrevê-los duas vezes em sequência na **string**: "{{" e "}}".

Exemplo 7:

O programa abaixo exibe várias mensagens com ocorrências de sequências de escape em literais string.

```
1 print('Ouro\tPreto\nMinas\tGerais')
2 print('\n')
3 print('Uma aspa \" no meio da string')
4 print('Um apóstrofo \' no meio da string')
5 print('Uma barra \\ no meio da string')
6 print(r'Aqui o caracter \ não é especial')
7 print(R'Nesta string \n não é nova linha')
8 print(f'Chaves {{ aqui }} em uma string de formatação')
```

3.2 Entrada: função input

Abaixo a sintaxe da função **input**:

input(*mensagem*)

onde:

- **input**: nome da função responsável pela entrada de dados informados pelo usuário;
- *mensagem*: argumento de entrada para a função. É uma **string** (**valor textual**) que será exibida como mensagem no terminal para orientar o usuário.

Abaixo, como exemplo, a função **input** imprime a mensagem `"Informe um nome: "` e aguarda a entrada de dados pelo usuário. Após a inserção dos dados é necessário um clique na tecla **ENTER** para indicar o término da entrada e o retorno para o programa.

```
1 input('Informe um nome: ')
```

É comum no desenvolvimento, que os dados fornecidos sejam armazenados em variáveis para posteriormente serem manipulados em algum ponto do programa. Em **Python** as entradas feitas através da função **input** podem ser atribuídas (usando o operador de atribuição "=") diretamente a uma variável.

Abaixo a sintaxe da função **input** em conjunto com a atribuição de dados a uma variável:

variável = **input**(*mensagem*)

onde:

- *variável*: nome da variável que armazenará o valor da entrada;
- =: operador de atribuição;
- **input**: nome da função responsável pela entrada de dados informados pelo usuário;
- *mensagem*: argumento de entrada para a função. É uma **string** (**valor textual**) que será exibida como mensagem no terminal para orientar o usuário.

Abaixo, um exemplo com a atribuição, primeiro a função **input** exibirá a mensagem na tela, aguardará até que o usuário digite algo e conclua com a tecla **ENTER**, e depois, o valor informado será atribuído à variável como um dado do tipo **string** (**str**), ou seja, uma sequência de caracteres digitada pelo usuário através do terminal.

```
1 nome = input('Informe um nome: ')
```

3.2.1 Conversão de tipos

O resultado da função **input** será sempre um valor **string**, independente do conteúdo informado pelo usuário. Ou seja, para a realização de cálculos matemáticos por exemplo, será necessário converter o valor **string** para inteiro (**int**) ou real (**float**), conforme os exemplos a seguir.

3.2.2 Conversão para inteiro: função int

Converte um valor para o tipo numérico **inteiro**:

```
1 x = 10.7
2 y = '10'
3 print(f'Valor de x original (real): {x}')
4 print(f'Valor de x como um inteiro: {int(x)}')
5 print(f'Valor de y original (string): {y}')
6 print(f'Valor de y como um inteiro: {int(y)}')
```

3.2.3 Convertão para real: função float

Converte um valor para o tipo numérico **real**:

```
1 x = 10
2 y = '10.2'
3 print(f'Valor de x original (inteiro): {x}')
4 print(f'Valor de x como um real: {float(x)}')
5 print(f'Valor de y original (string): {y}')
6 print(f'Valor de y como um real: {float(y)}')
```

3.2.4 Convertão para string: função str

Converte um valor para o tipo **string**:

```
1 var_inteira = 10
2 var_string = str(var_inteira)
3 soma = var_inteira + 5
4 concatenacao = var_string + '5'
5 print(soma)
6 print(concatenacao)
```

Como resultado da execução do programa anterior teríamos como valor para a variável **soma** o número inteiro 15, que será impresso no primeiro **print**, como valor para a variável **var_string** o valor string `"10"` e como valor para a variável **concatenação** o valor string `"105"`, que será impresso pelo segundo **print**.

Observe também que aqui o operador "+" funciona de maneira distinta para tipos numéricos e tipos string. Para valores numéricos ele realiza a **soma matemática** e para valores string ele realiza a **concatenação textual**. O **Python** não permite usar este operador para um operando numérico e outro operando string, se você tentar fazer isso ocorrerá um erro no programa.

3.2.5 Cuidados especiais com conversão de tipos

Existe um problema com a conversão de tipos envolvendo **strings**. Se o texto não puder ser representado pelo tipo convertido, ocorrerá um erro de execução. Isto vale para qualquer tipo de conversão. Veja os exemplos a seguir.

```
1 y = '10.2'
2 print(f'Valor de y como um inteiro: {int(y)}')
```

```
1 y = 'ops'
2 print(f'Valor de y como um inteiro: {int(y)}')
```

```
1 y = 'ops'
2 print(f'Valor de y como um real: {float(y)}')
```

3.2.6 Conversão de dados fornecidos pelo usuário

Dados fornecidos pelo usuário são normalmente armazenados em uma variável, assim, basta converter o valor da variável fornecida:

```
1 peso = input('Forneça o peso total (número real): ')
2 peso = float(peso)
3 unid = input('Forneça a quantidade de unidades (número inteiro): ')
4 unid = int(unid)
5 print(f'O peso de cada unidade é {peso/unid:.2f}')
```

Observe que os valores esperados pelo programa anterior são: **número real** para o peso e **número inteiro** para a quantidade de unidades. Caso o usuário forneça valores que não possam ser convertidos, ocorrerá erro de execução, pois as conversões feitas levam em consideração os tipos esperados.

Outra observação interessante é que foram usadas as próprias variáveis atribuídas pelos **inputs** para receberem os resultados das conversões. Isto é possível porque o **Python** é uma linguagem **dinamicamente tipada**, o que significa que as variáveis podem mudar de tipo durante a execução do programa.

Porém, sempre que precisarmos fazer a conversão desta forma, podemos fazê-la em uma única linha de instrução, colocando o **input** dentro da função de conversão utilizada:

```
1 peso = float(input('Forneça o peso total (real): '))
2 unid = int(input('Forneça a quantidade de unidades (inteiro): ' ))
3 print(f'O peso de cada unidade é {peso/unid:.2f}')
```

O resultado final é exatamente o mesmo, mas obtemos um programa mais compacto e ele realiza menos operações (**economizamos** duas atribuições).

3.3 Entrada e Saída: Exemplos Práticos

Exemplo 8:

Implemente um programa que leia dois valores inteiros e calcule sua soma, armazenando o resultado em uma variável. A seguir imprima o resultado da soma.

Solução:

Para a resolver este problema, devemos começar pensando em um **encadeamento de passos**, ou seja, um **algoritmo**, a ser executado:

1. Entrada dos dados: duas variáveis com valor inteiro;
2. Cálculo da soma dos valores das duas entradas;
3. Saída do resultado: valor da soma.

O fluxograma da Figura 2 apresenta estas etapas de uma forma gráfica e um pouco mais detalhada. Com isso, fica mais fácil converter as etapas em código para a construção da solução que é apresentada de forma **iterativa** a seguir.

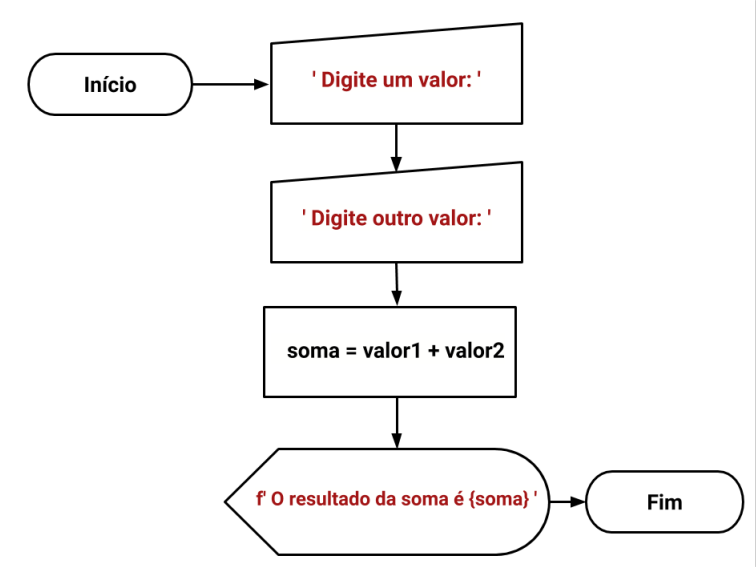

Figura 2: Fluxograma de solução para o exemplo 8.

Como primeiro passo definimos as entradas dos dados. No exemplo são solicitadas as leituras de dois valores inteiros, assim declaramos duas variáveis **valor1** e **valor2**. E através da função **input** associada à função **int**, definimos as entradas:

```
1 # Entrada
2 valor1 = int(input('Digite um valor: '))
3 valor2 = int(input('Digite outro valor: '))
```

Na etapa seguinte, declaramos a variável **soma** que será responsável por receber e armazenar o valor correspondente ao somatório dos valores contidos nas variáveis **valor1** e **valor2**:

```
1 # Entrada
2 valor1 = int(input('Digite um valor: '))
3 valor2 = int(input('Digite outro valor: '))
4 # Processamento
5 soma = valor1 + valor2
```

No final, como saída para o usuário, utilizaremos a função **print** e como argumento da função: a **f-string** *f'O resultado da soma é {soma}'* onde o valor da variável **soma** será inserido no texto na respectiva posição.

```
# Entrada
valor1 = int(input('Digite um valor: '))
valor2 = int(input('Digite outro valor: '))
# Processamento
soma = valor1 + valor2
# Saída
print(f'O resultado da soma é {soma}')
```

Exemplo 9:

Modifique o programa anterior, onde os dois valores passam a ser reais e o resultado da soma o numerador de uma divisão. O denominador será um novo valor real lido. O programa imprime apenas o resultado final da divisão, que também deverá estar armazenado em uma variável. A saída do resultado deve ser arredondada para duas casas decimais de precisão.

Solução:

Novamente, vamos começar pensando em um **algoritmo**:

1. Entrada dos dados: três variáveis com valor real;
2. Cálculo da soma dos dois primeiros valores;
3. Cálculo do resultado final: soma dividida pelo terceiro valor;
4. Saída do resultado final: valor da divisão.

Agora vamos observar o fluxograma da Figura 3 e convertê-lo em código de forma **iterativa** a seguir.

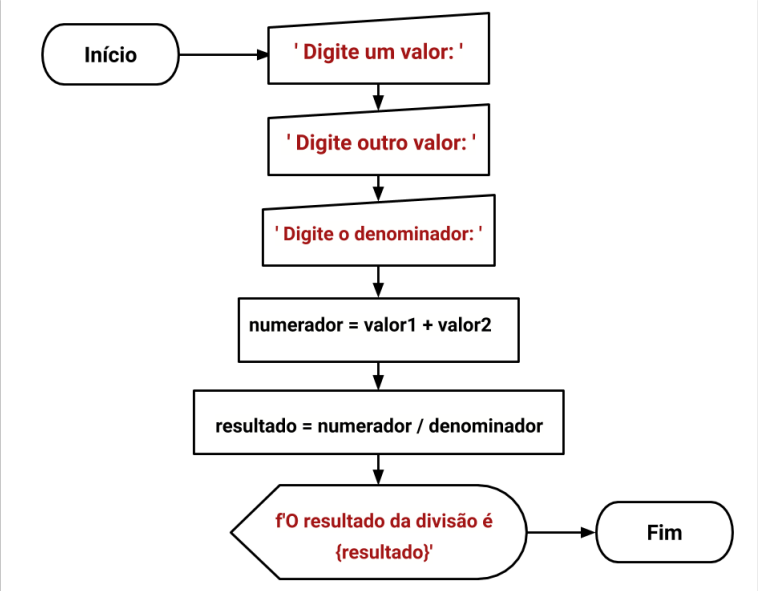

Figura 3: Fluxograma de solução para o exemplo 9.

Na resolução deste exemplo, declaramos três variáveis **valor1**, **valor2** e **denominador**, que serão informadas pelo usuário. As variáveis **valor1** e **valor2** armazenarão os valores que serão somados. E a variável **denominador** armazenará o valor de denominador utilizado posteriormente na operação de divisão.

```
# Entrada
valor1 = float(input('Digite um valor: '))
valor2 = float(input('Digite outro valor: '))
denominador = float(input('Digite o denominador: '))
```

Na etapa seguinte definimos a variável **numerador**, sendo esta inicializada com a soma dos valores contidos nas variáveis **valor1** e **valor2**. E a variável **resultado**, que é inicializada com a divisão do valor armazenado na variável **numerador** pela variável **denominador**.

```
# Entrada
valor1 = float(input('Digite um valor: '))
valor2 = float(input('Digite outro valor: '))
denominador = float(input('Digite o denominador: '))
# Processamento
numerador = valor1 + valor2
resultado = numerador / denominador
```

No final, como solução para o exemplo, definimos uma saída através da função **print**, que recebe a **f-string** *f'O resultado da divisão é {resultado:.2f}'* como argumento. Observe que, conforme solicitado no enunciado, formatamos a saída com duas casas decimais de precisão.

```
# Entrada
valor1 = float(input('Digite um valor: '))
valor2 = float(input('Digite outro valor: '))
denominador = float(input('Digite o denominador: '))
# Processamento
numerador = valor1 + valor2
resultado = numerador / denominador
# Saída
print(f'O resultado da divisão é {resultado:.2f}')
```

Exemplo 10:

Implemente um programa que imprima a hipotenusa (h) de um triângulo retângulo de acordo com a leitura de seus catetos (a e b). A saída deve ser arredondada para três casas decimais.

Observação: $h = \sqrt{a^2 + b^2}$.

Solução:

Novamente, o algoritmo segue uma estrutura semelhante: 1. Entradas; 2. Processamento; 3. Saída; que será diretamente representado pelo fluxograma da Figura 4.

Inicialmente declaramos as variáveis **a** e **b**, responsáveis por armazenar os valores correspondentes aos catetos, e serão inicializadas pelo usuário através da função **input** e seus valores convertidos pela função **float**.

```
# Entrada
a = float(input('Digite o cateto a: '))
b = float(input('Digite o cateto b: '))
```

Na sequência, declaramos a variável **h** que armazenará o valor do cálculo da hipotenusa. Note que para calcular o valor da raiz utilizamos a função **sqrt**, disponível no módulo **math** através da declaração da diretiva **import** no início do programa.

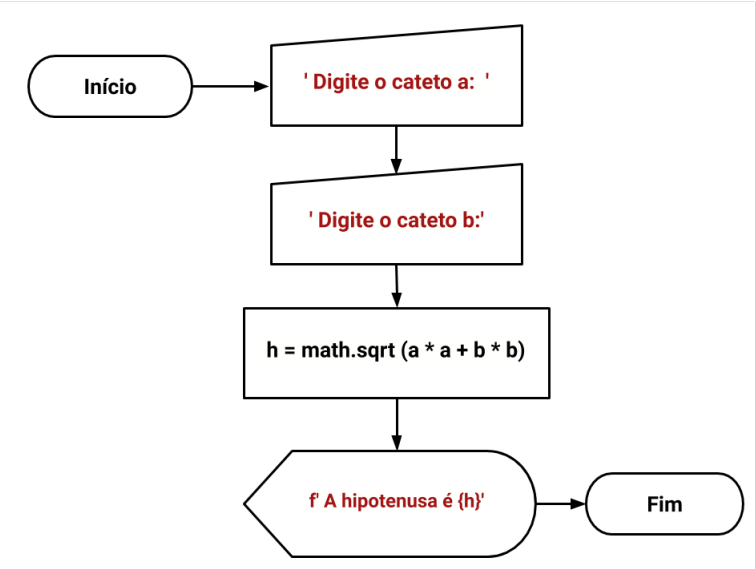

Figura 4: Fluxograma de solução para o exemplo 10.

```
1 import math
2 # Entrada
3 a = float(input('Digite o cateto a: '))
4 b = float(input('Digite o cateto b: '))
5 # Processamento
6 h = math.sqrt(a*a + b*b)
```

Ao final, como saída do programa definiremos como parâmetros para a função **print** a **f-string** *f'A hipotenusa é {h:.2f}'*, onde o valor da hipotenusa armazenado na variável **h** é inserido com três casas decimais.

```
1 import math
2 # Entrada
3 a = float(input('Digite o cateto a: '))
4 b = float(input('Digite o cateto b: '))
5 # Processamento
6 h = math.sqrt(a*a + b*b)
7 # Saída
8 print(f'A hipotenusa é {h:.3f}')
```

Exemplo 11:

Implemente um programa que leia do teclado um valor de temperatura em Celsius (C), calcule e imprima essa temperatura em Fahrenheit (F) e em Kelvin (K), com precisão de uma casa decimal.

Equações de conversão:

$$F = C * 1.8 + 32$$

$$K = C + 273.15$$

Solução:

O **algoritmo** tem as mesmas características dos exemplos anteriores, e seu fluxograma é apresentado na Figura 5.

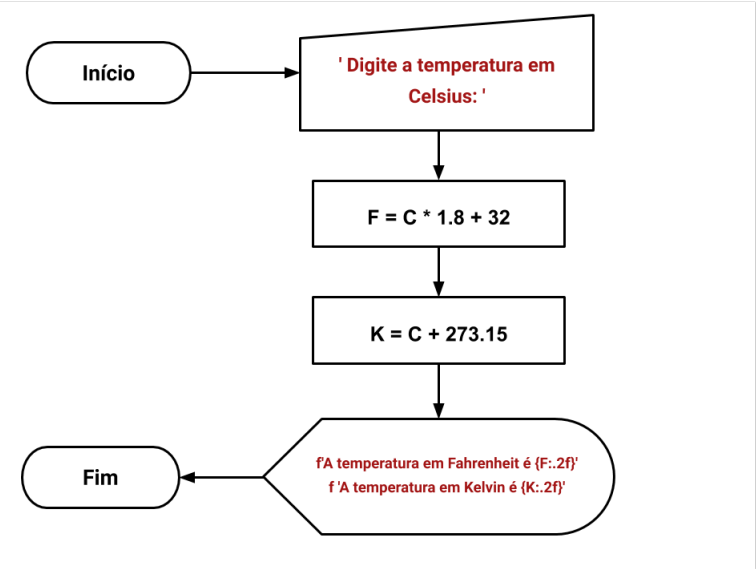

Figura 5: Fluxograma de solução para o exemplo 11.

O programa construído é apresentado a seguir:

```
1 # Entrada
2 C = float(input(' Digite a temperatura em Celsius: '))
3 # Processamento
4 F = C * 1.8 + 32
5 K = C + 273.15
6 # Saída
7 print(f'A temperatura em Fahrenheit é {F:.1f}')
8 print(f'A temperatura em Kelvin é {K:.1f}')
```

3.4 Exercícios Propostos

Exercício 1

Elabore um programa que leia o valor do raio de um círculo qualquer (número inteiro) e calcule sua respectiva área. Ao final imprima como saída o valor da área do círculo (com duas casas decimais).

Fórmula:

$$rea = \pi.r^2$$

onde:

- π : constante **Pi**;
- r: raio do círculo.

Exemplos de execução:

Exemplo 1:

```
Digite o valor do raio: 5
A área do círculo é: 78.54
```

Exemplo 2:

```
Digite o valor do raio: 9
A área do círculo é: 254.47
```

Exemplo 3:

```
Digite o valor do raio: 15
A área do círculo é: 706.86
```

Exercício 2

Implemente um programa que leia o valor do raio **r** de uma esfera (número real) e calcule o volume da esfera. No final imprima como saída o valor calculado (com duas casas decimais).

Fórmula:

$$Volume = \frac{4\pi r^3}{3}$$

onde:

- π : constante **Pi**;
- r: raio do círculo.

Exemplos de execução:

Exemplo 1:

```
Digite o valor do raio: 5
O volume da esfera é: 523.60
```

Exemplo 2:

```
Digite o valor do raio: 10.5
O volume da esfera é: 4849.05
```

Exemplo 3:

```
Digite o valor do raio: 5.75
O volume da esfera é: 796.33
```

Exercício 3

Implemente um programa que tem como dados de entrada o comprimento, a largura e a altura (números inteiros), calcule e imprima o volume de um paralelepípedo (com duas casas decimais), utilizando a seguinte fórmula:

$$Volume = comprimento * largura * altura$$

Exemplos de execução:

Exemplo 1:

```
Digite o valor do comprimento: 50
Digite o valor da largura: 20
Digite o valor da altura: 15
O volume do paralelepípedo é: 15000.00
```

Exemplo 2:

```
Digite o valor do comprimento: 100
Digite o valor da largura: 50
Digite o valor da altura: 25
O volume do paralelepípedo é: 125000.00
```

Exemplo 3:

```
Digite o valor do comprimento: 75
Digite o valor da largura:  50
Digite o valor da altura:  30
O volume do paralelepípedo é: 112500.00
```

Exercício 4

Elabore um programa que receba como entradas um determinado valor em Real (R$) como saldo de um usuário e um valor para a cotação do Dólar (US$) (números reais). No final apresente como saída o valor convertido em Dólar (com duas casas decimais).

Exemplos de execução:

Exemplo 1:

```
Digite o valor em real: 500
Digite o valor de cotação do dólar: 4.15
O valor convertido é: 120.48
```

Exemplo 2:

```
Digite o valor em real: 1000
Digite o valor de cotação do dólar: 4.18
O valor convertido é: 239.23
```

Exemplo 3:

```
Digite o valor em real: 999
Digite o valor de cotação do dólar: 4.11
O valor convertido é: 243.07
```

Exercício 5

Implemente um programa que receba como entrada o valor de um veículo (número real) e calcule e imprima o valor do IPVA (Imposto sobre a Propriedade de Veículos Automotores) considerando a alíquota de 4.0% (com duas casas decimais).

Exemplos de execução:

Exemplo 1:

```
Digite o valor do veículo: 100000
O valor do IPVA é : 4000.00
```

Exemplo 2:

```
Digite o valor do veículo: 55000
O valor do IPVA é : 2200.00
```

Exemplo 3:

```
Digite o valor do veículo: 35000
O valor do IPVA é : 1400.00
```

3.5 Solução dos Exercícios Propostos

Exercício 1

Elabore um programa que leia o valor do raio de um círculo qualquer (número inteiro) e calcule sua respectiva área. Ao final imprima como saída o valor da área do círculo (com duas casas decimais).

Fórmula:

$$rea = \pi.r^2$$

onde:

- π : constante **Pi**;
- r: raio do círculo.

Exemplos de execução:

Exemplo 1:

```
Digite o valor do raio: 5
A área do círculo é: 78.54
```

Exemplo 2:

```
Digite o valor do raio: 9
A área do círculo é: 254.47
```

Exemplo 3:

```
Digite o valor do raio: 15
A área do círculo é: 706.86
```

Solução:

```
1 import math
2 # Entrada
3 raio = int(input('Digite o valor do raio: '))
4 # Processamento
5 area = math.pi * raio * raio
6 # Saída
7 print(f'A área do círculo é: {area:.2f}')
```

Exercício 2

Implemente um programa que leia o valor do raio **r** de uma esfera (número real) e calcule o volume da esfera. No final imprima como saída o valor calculado (com duas casas decimais).

Fórmula:

$$Volume = \frac{4\pi r^3}{3}$$

onde:

- π : constante **Pi**;
- r: raio do círculo.

Exemplos de execução:

Exemplo 1:

```
Digite o valor do raio: 5
O volume da esfera é: 523.60
```

Exemplo 2:

```
Digite o valor do raio: 10.5
O volume da esfera é: 4849.05
```

Exemplo 3:

```
Digite o valor do raio: 5.75
O volume da esfera é: 796.33
```

Solução:

```python
import math
# Entrada
raio = float(input('Digite o valor do raio: '))
# Processamento
volume = (4 * math.pi * raio ** 3) / 3
# Saída
print(f'O volume da esfera é: {volume:.2f}')
```

Exercício 3

Implemente um programa que tem como dados de entrada o comprimento, a largura e a altura (números inteiros), calcule e imprima o volume de um paralelepípedo (com duas casas decimais), utilizando a seguinte fórmula:

$$Volume = comprimento * largura * altura$$

Exemplos de execução:

Exemplo 1:

```
Digite o valor do comprimento: 50
Digite o valor da largura: 20
Digite o valor da altura: 15
O volume do paralelepípedo é: 15000.00
```

Exemplo 2:

```
Digite o valor do comprimento: 100
Digite o valor da largura: 50
Digite o valor da altura: 25
O volume do paralelepípedo é: 125000.00
```

Exemplo 3:

```
Digite o valor do comprimento: 75
Digite o valor da largura:  50
Digite o valor da altura:  30
O volume do paralelepípedo é: 112500.00
```

Solução:

```
1 # Entrada
2 comprimento = int(input('Digite o valor do comprimento: '))
3 largura = int(input('Digite o valor da largura: '))
4 altura = int(input('Digite o valor da altura: '))
5 # Processamento
6 volume = comprimento * largura * altura
7 # Saída
8 print(f'O volume do paralelepípedo é: {volume:.2f}')
```

Exercício 4

Elabore um programa que receba como entradas um determinado valor em Real (R$) como saldo de um usuário e um valor para a cotação do Dólar (US$) (números reais). No final apresente como saída o valor convertido em Dólar (com duas casas decimais).

Exemplos de execução:

Exemplo 1:

```
Digite o valor em real: 500
Digite o valor de cotação do dólar: 4.15
O valor convertido é: 120.48
```

Exemplo 2:

```
Digite o valor em real: 1000
Digite o valor de cotação do dólar: 4.18
O valor convertido é: 239.23
```

Exemplo 3:

```
Digite o valor em real: 999
Digite o valor de cotação do dólar: 4.11
O valor convertido é: 243.07
```

Solução:

```
# Entrada
valor = float(input('Digite o valor em real: '))
cotacao = float(input('Digite o valor de cotação do dólar: '))
# Processamento
dolar = valor/cotacao
# Saída
print(f'O valor convertido é: {dolar:.2f}')
```

Exercício 5

Implemente um programa que receba como entrada o valor de um veículo (número real) e calcule e imprima o valor do IPVA (Imposto sobre a Propriedade de Veículos Automotores) considerando a alíquota de 4.0% (com duas casas decimais).

Exemplos de execução:

Exemplo 1:

```
Digite o valor do veículo: 100000
O valor do IPVA é : 4000.00
```

Exemplo 2:

```
Digite o valor do veículo: 55000
O valor do IPVA é : 2200.00
```

Exemplo 3:

```
Digite o valor do veículo: 35000
O valor do IPVA é : 1400.00
```

Solução:

```
# Entrada
valor = float(input('Digite o valor do veículo: '))
taxa = 0.04
# Processamento
ipva = valor * taxa
# Saída
print(f'O valor do IPVA é: {ipva:.2f}')
```