Conceitos Básicos de Programação

Eldrey Galindo

Lembre-me:

✓ **Chamada** está disponível no Classroom

✓ Aula está sendo gravada e estará disponível para os alunos que solicitarem na secretaria.

Problemas e Soluções

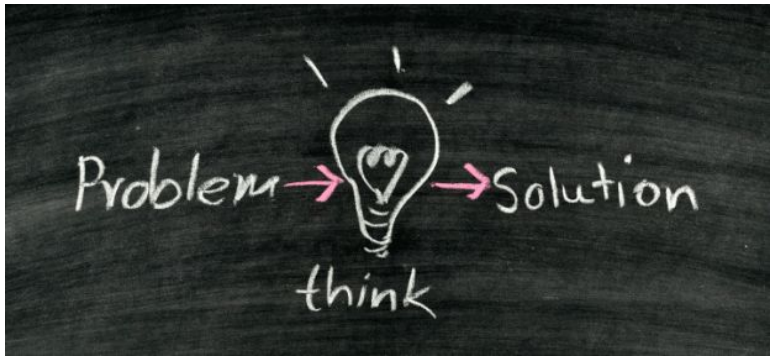
No mundo real temos:

- Problemas: representam situações do mundo físico que podem ser transformadas em modelos que os representem

- Soluções: são as respostas possíveis para problemas do mundo físico; utilizam métodos e cálculos para resolução

Problemas e Soluções

- Pode existir mais de uma solução para os problemas
- Algumas soluções são melhores que outras sob algum aspecto
- Alguns problemas são casos particulares de outros similares
- Sempre que possível, é melhor resolver o problema mais genérico



Problemas e Soluções

- Exemplo
  - Contar o número de pessoas em um evento

Contando pessoas em um evento

Solução 1: contar pessoa por pessoa, um por vez

- Não contar a mesma pessoa mais de uma vez
- Não se esquecer de contar alguém

Vantagens: simples, fácil de executar, não exige conhecimento prévio, sem recursos extras

Desvantagens: tempo de contagem (se sala for grande e estiver cheia), altas chances de errar

Contando pessoas em um evento

Solução 2: contar cadeiras vazias

Vantagens: simples, fácil de executar, sem recursos extras

Desvantagens: requer saber quantas cadeiras há, tempo de contagem (se sala for grande e estiver quase vazia)

Contando pessoas em um evento

Solução 2: contar cadeiras vazias

Vantagens: simples, fácil de executar, sem recursos extras

Desvantagens: requer saber quantas cadeiras há, tempo de contagem (se sala for grande e estiver quase vazia)

E se fosse um comício?

Contando pessoas em um evento

Solução 3: contagem por estatística

Vantagens: mais rápido que outras soluções para públicos maiores

Desvantagens: deve-se saber a metragem da praça de antemão e estimar pessoas por metro quadrado, número impreciso

Contando pessoas em um evento

Solução 4: roletas/catracas

Vantagens: numero exato

Desvantagens: instalação dos equipamentos, ambiente controlado (evitar de burlar catracas)

C.E.S.A.R school

Contando pessoas em um evento

Outras soluções

Solução 5: filas organizadas e de mesmo tamanho

Solução 6: trabalho distribuído em paralelo (cada primeiro conta sua fila e ao final soma-se os valores de cada fileira)

Algoritmos

O que são?

Algoritmo

- Algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais devendo ser executadas mecânica ou eletronicamente em um intervalo de tempo finito e com uma quantidade de esforço finita.
    - Quando executado, produz uma solução
    - Registra uma solução para um problema
    - Outra pessoa pode executar
- Resumindo: É uma sequência finita de passos visando resolver um determinado problema.

Algoritmo

Exemplos?

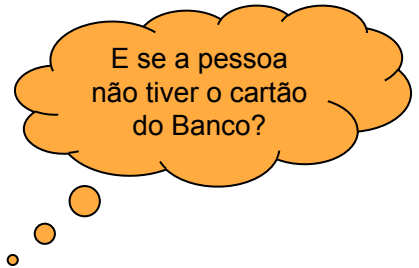
Exemplos de algoritmos

Roteiro de um filme

Manual de instruções de um eletrodoméstico

Receita de bolo

Exemplo de algoritmo



- Sacar dinheiro de um caixa eletrônico

1. Inserir o cartão do banco no leitor;
2. Digitar senha da conta/cartão;
3. No menu que vai aparecer, escolher a opção “Saque”;
4. Digitar o valor desejado e apertar a tecla “Entra”;
5. Confirmar o saque e apertar a tecla “Entra”;
6. Ficar esperando em frente ao caixa até a saída do dinheiro;
7. Retirar o dinheiro.

Algoritmo

- Então toda “solução” pode ser representada por um algoritmo?
- Podemos construir um algoritmo para qualquer coisa?

Algoritmo

- Para um problema ser solucionado por um algoritmo, ele precisa ter lógica.
- Como encontrar a lógica?

O que é um programa?

Um programa é um conjunto de algoritmos (instruções) que são executados por um computador

Programação

- Mas como dizer ao computador para executar os passos do algoritmo?
  - Ele entende português?
  - Inglês?
  - Espanhol?
  - ...

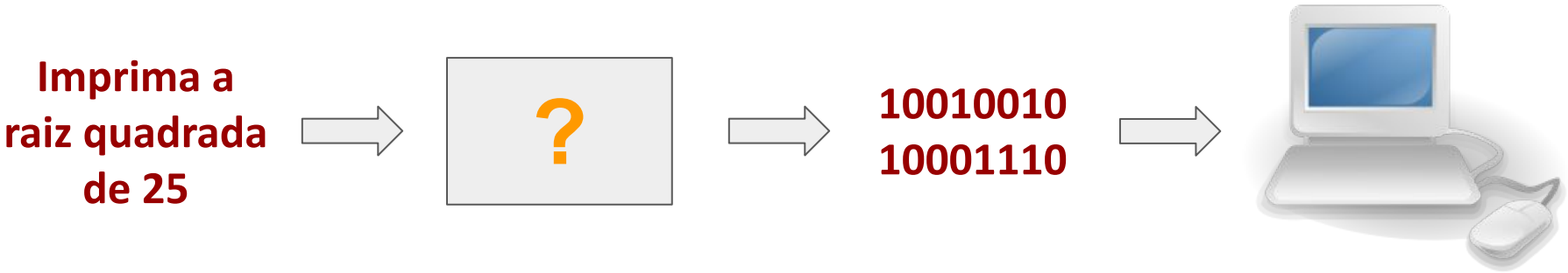
Linguagens de programação

- Os programas têm que ser escritos em uma linguagem de programação:
  - uma linguagem que pode ser **entendida** pelo computador

10010010
10001110

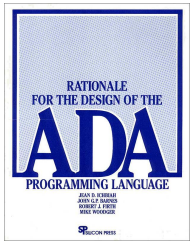
Linguagens de programação

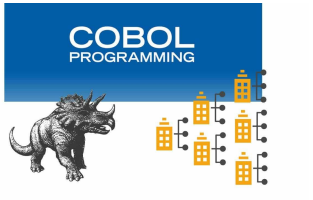
- Uma linguagem que **entendemos** e que possa ser **traduzida** para a linguagem entendida pelo computador

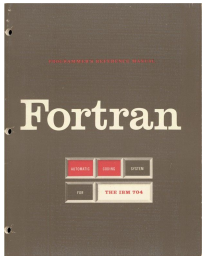


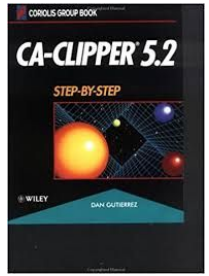
Linguagens de programação

- Existem muitas linguagens de programação









Linguagens de programação

- Usamos a programação para criar um programa através de algoritmos
- Um algoritmo deve passar por algumas fases antes de se tornar programa
- Escrevemos o algoritmo em uma linguagem
    - O algoritmo deve ser processado, adaptado para o computador
    - Em linguagem do computador, o programa deve ser carregado em memória

Linguagens de programação

- Linguagens existentes:
  - Linguagem natural
  - Linguagem de alto nível
  - Linguagem de baixo nível
  - Linguagem de máquina

Linguagens de programação

- Linguagens existentes:
  - **Linguagem natural**: falada e escrita pelas pessoas. Exemplo: português, inglês, alemão
  - **Linguagem de alto nível**: subconjunto da linguagem natural. Códigos legíveis aos humanos usados para programar. Exemplo: C, C++, Java, Pascal, Fortran, Cobol, Python

Linguagens de programação

- Linguagens existentes:
  - **Linguagem de baixo nível**: subconjunto da linguagem natural. Menos legível, mais próximo da linguagem de máquina. Exemplo: Assembly
  - **Linguagem de máquina**: linguagem entendível pelo computador, “incompreensível” para humanos. Exemplo: Binário

Linguagens de programação

- Componentes de software:
  - **Compilador (Compiler):** transforma o código-fonte de linguagem de alto nível para código objeto.
  - **Montador (Assembler):** transforma o código-fonte de linguagem de baixo nível para código objeto
  - **Ligador (Linker):** transforma o código objeto para formato executável
  - **Carregador (Loader):** carrega arquivos em formato executável para execução

Linguagens de programação

Normalmente...

- Não é necessário fazer todos os passos
- Os compiladores compilam e ligam os arquivos automaticamente

Com **Python**, temos uma linguagem **interpretada**

- Não possuem compilação
- Não geram código objeto
- Existe um programa que lê o código-fonte, interpreta a instrução e ele mesmo a executa

Linguagens de programação

- Passos “grosseiros” para escrever um programa:

1. Entender o problema
2. Planejar a lógica (elaborar o algoritmo)
3. Escrever o programa (programar)
4. Traduzir o programa para linguagem de máquina (compilação)
5. Testar o programa
6. Instalar o programa para uso

Construção de Algoritmos

- Identificação do problema;
- Identificação das “entradas de dados”;
- Identificação das “saídas de dados”;
- Identificação de regras do problema e limitações do agente;
- Determinar o que fazer para transformar as “entradas” em “saídas”;
- Construção do algoritmo;
- Teste de solução.

Construção de Algoritmos

- Como um algoritmo é construído?

Representação de Algoritmos

- As formas de representação de algoritmos mais conhecidas são:
  - Descrição narrativa;
  - Fluxograma convencional;
  - Pseudo-código:
    - Linguagem estruturada;
    - Portugol.

Descrição Narrativa

- Os algoritmos são expressos em linguagem natural.

Trocar um pneu furado

1. Afrouxar ligeiramente as porcas
2. Suspender o carro
3. Retirar as porcas e o pneu
4. Colocar o pneu reserva
5. Apertar as porcas
6. Abaixar o carro
7. Dar o aperto final nas porcas

Calcular média de um aluno

1. Obter as suas 2 notas de provas
2. Calcular a média aritmética
3. Se a média for maior que 7, o aluno foi aprovado, senão ele foi reprovado

Descrição Narrativa

- Os algoritmos são expressos em linguagem natural.

Trocar um pneu furado

1. Afrouxar ligeiramente as porcas
2. Suspender o carro
3. Retirar as porcas e o pneu
4. Colocar o pneu reserva
5. Apertar as porcas
6. Abaixar o carro
7. Dar o aperto final nas porcas

Ambiguidade

Calcular média de um aluno

1. Obter as suas 2 notas de provas
2. Calcular a média aritmética
3. Se a média for maior que 7, o aluno foi aprovado, senão ele foi reprovado

Fluxograma Convencional

- Representação do algoritmo em formato gráfico;
- Formas geométricas implicam ações distintas;
- Facilita o entendimento do algoritmo.

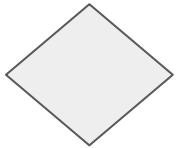
Fluxograma Convencional

Início ou final do fluxograma

Entrada de dados

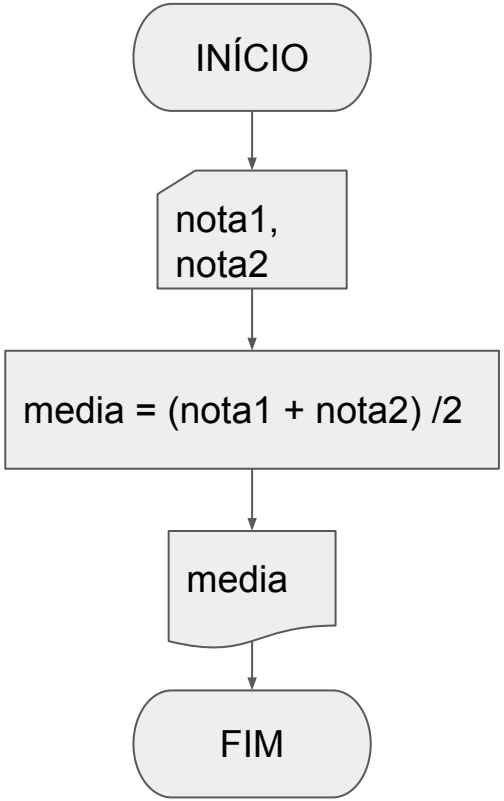
Saída de dados

Operação de atribuição

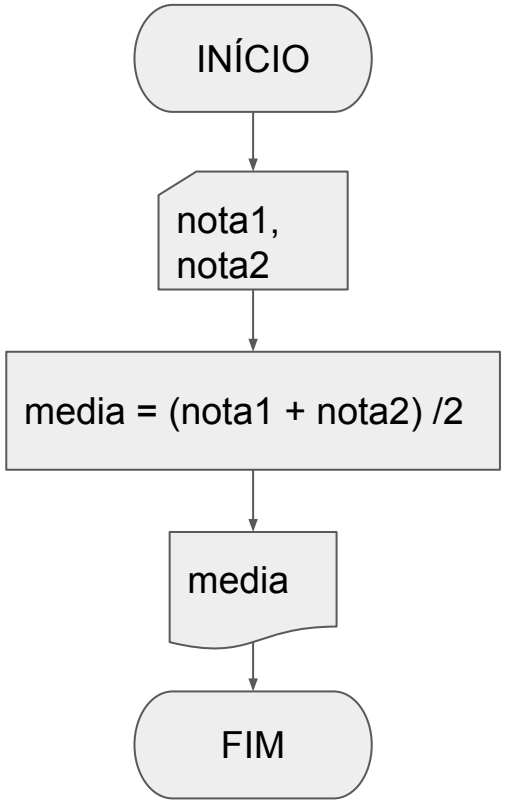


Decisão

Fluxograma Convencional



Fluxograma Convencional



Como ficaria esse fluxograma se adicionássemos uma “decisão”?

Fluxograma Convencional

- Figuras conhecidas mundialmente;
- As figuras podem facilitar o entendimento do que deve ser feito no algoritmo;
- Não oferece recursos para descrever dados;
- A medida que o fluxograma cresce, o algoritmo fica complicado de entender.

Pseudo-código

- Informações ricas em detalhes;
- Assemelha-se com a forma em que os programas serão escritos;
- <nome_do_algoritmo>
  <declaração_de_variáveis>
  inicio
  <corpo_do_algoritmo>
  fim

Sintaxe e Semântica

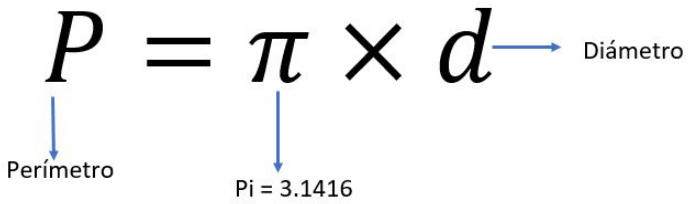
- Antes de entrar em maiores detalhes sobre a construção de algoritmos é necessário entender a diferença entre sintaxe e semântica.
- Sintaxe:
  - Nome dado ao conjunto de regras a serem seguidas para a escrita dos algoritmos.
  - O computador só é capaz de entender algoritmos sintaticamente corretos.
  - Sintaxe está associada a forma de um comando.
- Semântica:
  - Refere-se ao que é efetuado pelo computador quando ele encontra um comando.
  - Semântica está relacionada ao conteúdo de um comando.

Algoritmos

- Estruturas para construção de algoritmos
  - Variáveis;
  - Constantes;
  - Operadores aritméticos, relacionais e lógicos;
  - Atribuição;
  - Comandos de condição ou seleção;
  - Comandos de repetição.

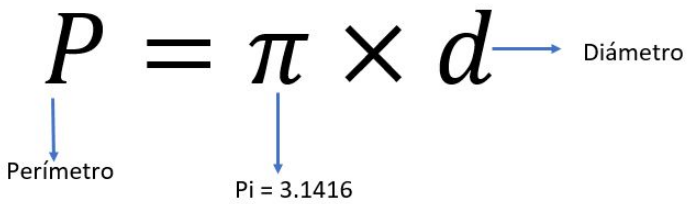
Variáveis

- Entidade que armazena valores;
- Possuem um tipo de dado;
- Valores modificáveis;
- Armazenam apenas um valor a cada instante;
- Deve receber um nome para referenciação e modificação;
- Devem ser declaradas antes de serem utilizadas:
  - Geralmente no início do programa.
- Ao término da execução são apagadas da memória



Constantes

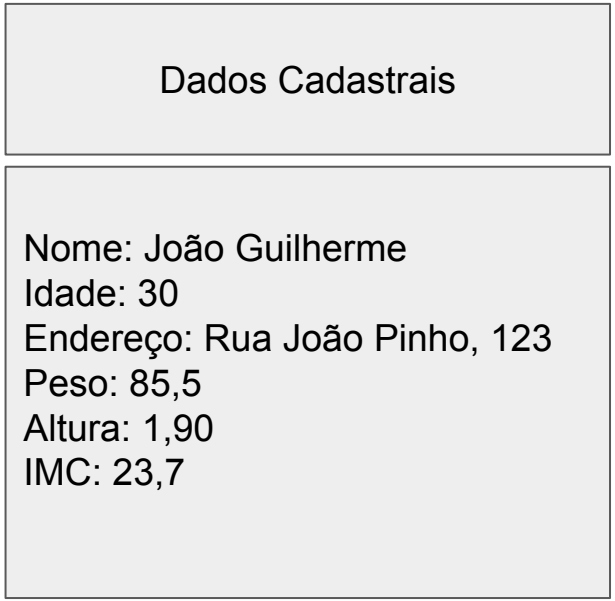
- É um valor fixo, que não muda durante a execução de um programa.



Tipos de dados

- Dado pode ser definido como informação em estado primitivo cujo processamento pode gerar informação útil.
- Devido a natureza do que vai ser armazenado em memória, existem tipos de dados diferentes.
- Tipo de dado representa o conjunto de valores possíveis para um dado.

Variáveis


Dados Cadastrais
Nome: João Guilherme
Idade: 30
Endereço: Rua João Pinho, 123
Peso: 85,5
Altura: 1,90
IMC: 23,7

Variáveis

- Elas podem ser basicamente de três tipos:
  - Numéricas;
  - Textual;
  - Lógicas.

Variáveis

- Variáveis Numéricas
  - Podem ser divididos em dois conjuntos:
    - Inteiro - Podem ser positivos, negativos ou nulos, mas não possuem componente decimal:
      - Ex.: 5; -9; 0; 189;-800.
    - Real - Podem ser positivos, negativos ou nulos, e possuem componente decimal:
      - Ex.: 5,89; -6,8978; 0; 7,986; -1458,252.
    - Os inteiros são compatíveis com os reais, mas os reais não são compatíveis com os inteiros;
    - Os inteiros consomem menos espaço de armazenamento na memória.

Variáveis

- Variáveis Textuais
  - Dados compostos por caracteres alfanuméricos:
    - Números
    - Letras
    - Caracteres especiais

Variáveis

- Variáveis Lógicas
  - Dados que assumem apenas dois valores: Verdadeiro ou Falso.

Dados Cadastrais
Nome: João Guilherme Idade: 30 Endereço: Rua João Pinho, 123 Peso: 85,5 Altura: 1,90 IMC: 23,7 Peso Ideal: (x) Sim ( ) Não

IMC	Classificação
abaixo de 18,5	abaixo do peso
entre 18,6 e 24,9	Peso ideal (parabéns)
entre 25,0 e 29,9	Levemente acima do peso
entre 30,0 e 34,9	Obesidade grau I
entre 35,0 e 39,9	Obesidade grau II (severa)
acima de 40	Obesidade III (mórbida)

Variáveis

- Para utilizar uma variável em um programa é preciso definir que valores ela pode assumir

```
real x     = 1,8
texto h    = “123”
inteiro y  = 10
inteiro j  = 20,34
logico z   = y > 1
texto w    = 10==2
```

Variáveis

- Para utilizar uma variável em um programa é preciso definir que valores ela pode assumir

```
real x     = 1,8     ✔
texto h    = “123”   ✔
inteiro y  = 10      ✔
inteiro j  = 20,34   ✖
logico z   = y > 1   ✔
texto w    = 10==2   ✖
```

Nomeando Variáveis

- Há algumas regras para nomear variáveis:
  - O nome de uma variável deve ser único dentro de um programa;
  - O primeiro caractere deve ser uma letra ou _;
  - Não é permitido o uso de caracteres especiais (exceto o _);
  - Espaços não são permitidos;
  - Não deve ser utilizado os nomes reservados da linguagem de programação utilizada.

Nomeando Variáveis

Nomes válidos	Nome inválidos
peso_ideal	peso Ideal
nomeCompleto	nome%completo
endereco	endereço
num1	num=1

Nomeando Variáveis

- Regras de estilo - dicas:
  - Não utilizar acentuação;
    - Ex.: ~~diâmetro~~ -> diametro; ~~média~~ -> media
  - Letras maiúsculas apenas para constantes;
    - Ex.: PI
  - Letras minúsculas apenas para variáveis;
    - Ex.: raio, diametro
  - Palavras compostas: separadas por *underline* (_)
    - Ex.: nome_completo, dia_semana, horario_intervalo

Declarando Variáveis

- Declarar uma variável significa definir o seu tipo e seu nome;
- Devem ser declaradas antes de seu uso no programa, geralmente no topo;
- Duas variáveis não devem ter o mesmo nome;
- Utilizaremos a seguinte sintaxe:
  - tipo nome_da_variavel
  - Ex.: inteiro idade, real peso, cadeia nome, caractere sexo, logico peso_ideal

Comando de entrada de dados

- Normalmente precisamos de dados de entrada para serem processados pelos algoritmos.
- Desta forma, precisamos de um comando para solicitar e obter dados fornecidos pelo usuário.

```
leia(<variável1>, <variável2>, ...)
```

- Quando um computador encontra um comando de entrada de dados, ele suspende a execução do programa até que os dados sejam fornecidos.

Comando de saída de dados

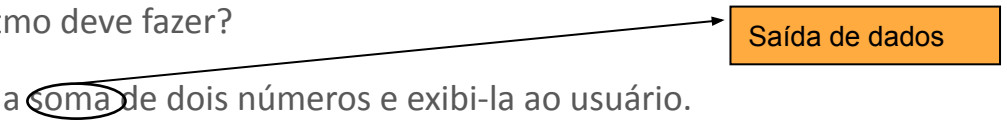
- Da mesma forma que precisamos receber dados externos, precisamos fornecer dados e instruções para os usuários.
- Desta forma, precisamos de um comando para retornar dados e/ou mensagens em um dispositivo de saída.

escreva(<var ou expressão ou mensagem>,...)

- Ex:
  - escreva(“O saldo atual é”, saldo)
  - escreva(soma)

Exemplo de construção de algoritmo

- Exemplo 1
  - Exibir a soma de dois números inteiros fornecidos pelo usuário.
- Solução
  - Objetivo é construir um algoritmo que ensine o computador a executar uma solução para o problema proposto.
  - O que o algoritmo deve fazer?
    - Calcular a soma de dois números e exibi-la ao usuário.



  - O valor da soma deverá ser armazenado em memória antes de ser exibido
    - É necessário então a criação de uma variável soma
    - É necessário que ao término do algoritmo a variável soma seja exibida para o usuário

Exemplo de construção de algoritmo

- Solução (cont.)
  - O enunciado afirma que os valores a serem somados devem ser informados pelo usuário
    - O algoritmo possui dados de entrada, e estes dados de entrada deverão ser armazenados em memória. É necessária a criação de duas variáveis num1 e num2 para armazenar os dados fornecidos pelo usuário.
    - É necessário a obtenção destes dados do usuário através do comando de entrada de dados.
  - O processamento do algoritmo deve possuir apenas a operação de soma.

```
leia(num1)
leia(num2)
```

    - Utilizaremos o operador de adição (+) para ensinar ao computador que operação, e com quais valores, deve ser realizada.
    - O resultado da soma deverá ser armazenado em memória pela variável soma, logo precisaremos utilizar o comando de atribuição.

```
soma = num1 + num2
```

Exemplo de construção de algoritmo

- Exemplo 1
  - Exibir a soma de dois números inteiros fornecidos pelo usuário.
- Solução

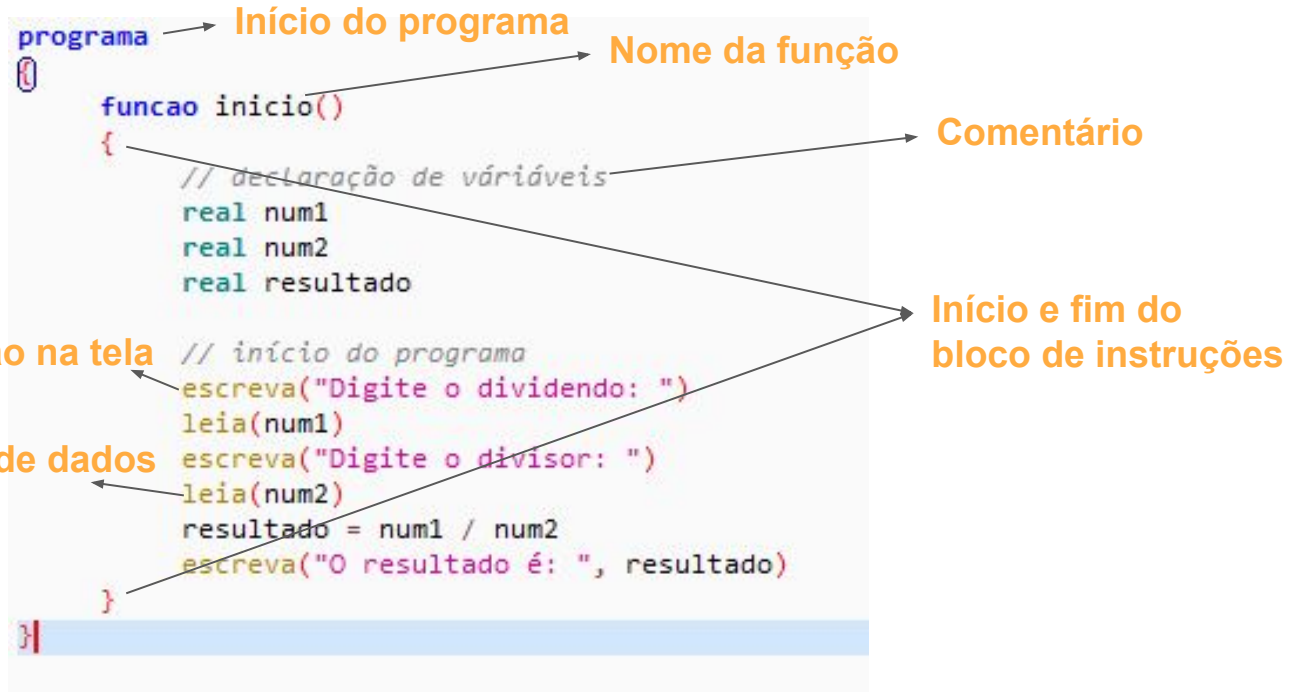
```
Variáveis
    inteiro num1, num2, soma
Início
    leia(num1)
    leia(num2)
    soma = num1 + num2
    escreva(soma)
Fim
```

Portugol Studio

- Download em:
  - http://lite.acad.univali.br/portugol/

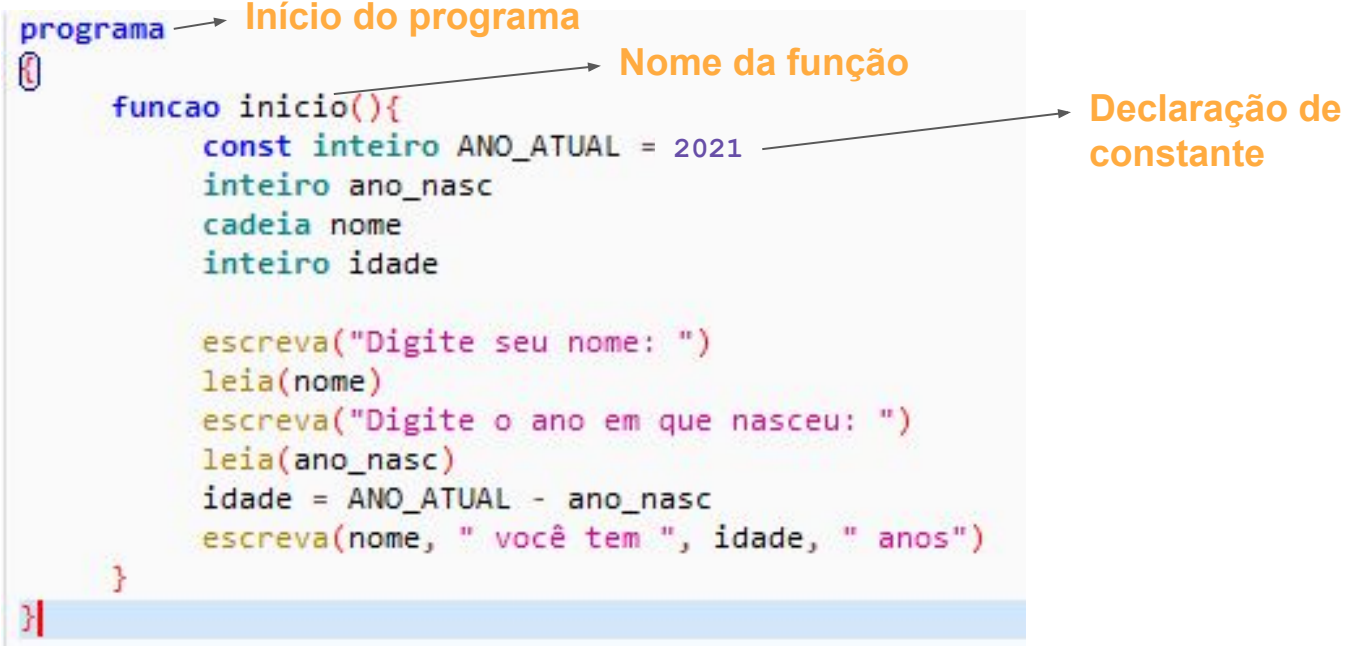
Portugol Studio

- Estrutura do código usando o Portugol Studio:



Portugol Studio

- Estrutura do código usando o Portugol Studio:



Vamos Praticar!

- Lista 01 disponível