Desenvolvimento de Sistemas

Manipulação de dados

No conteúdo “Algoritmos” desta Unidade Curricular, entre vários assuntos, você aprendeu que um algoritmo recebe dados de entrada e, ao final da execução, retorna uma ou mais saídas. No entanto, para que isso seja possível, é necessário que esses dados e instruções sejam armazenados na memória do computador para seu acesso posterior. Esse aprendizado prévio é uma base necessária para o entendimento da manipulação de dados por meio das variáveis e constantes.

Ao escrever algoritmos, inevitavelmente você lida com informações – dados constantes (um texto fixo ou um número, por exemplo) ou variáveis (dados informados pelo usuário, por exemplo). No primeiro caso, a informação está dentro do código, escrita pelo próprio programador, e não muda. No segundo caso, a informação com que o algoritmo tem que trabalhar vai se alterando e pode ser imprevisível, como no caso de entradas de usuário. Algumas características, como o tipo da informação (se é um texto ou um número, por exemplo), ajudam a ter alguma previsibilidade sobre elas. Conforme essas características, o algoritmo vai operar sobre os dados, seja realizando operações matemáticas, seja comparando valores entre várias outras situações

A seguir, você iniciará sua jornada de estudos e verá como os dados são representados internamente para que o computador possa entendê-los e utilizá-los corretamente.

Representação de dados

Representação interna

De modo geral, os humanos utilizam o sistema de representação numérica chamado de **sistema decimal** (um sistema de base 10). Esse sistema tem origem no fato de os dez dedos das mãos serem utilizados para a realização de contas e terem dez diferentes algarismos para representar uma quantidade infinita de valores (0,1, 2, 3, 4, 5, 6, 7, 8 e 9).

Nos computadores digitais, a notação utilizada possui apenas dois algarismos: o zero e o um (0 e 1) para representar os dados desejados. Esse sistema é conhecido como **sistema binário** (sistema de base 2) e pode representar os valores 0 ou 1, os estados ligado ou desligado ou simplesmente verdadeiro ou falso. O elemento mínimo capaz de armazenar esses valores é conhecido como *bit*, pois deriva de ***Binary Digit*** ou simplesmente Dígito Binário.

Por serem necessários muitos dígitos para representar números grandes no sistema binário, outros sistemas numéricos são utilizados: o **sistema hexadecimal** (sistema de base 16) que utiliza dezesseis dígitos (0,1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F); e o **sistema octal** (sistema de base 8) que utiliza oito algarismos (0, 1, 2, 3, 4, 5, 6 e 7).

A seguir, veja a tabela 1 que representa a equivalência dos valores 1 ao 15 nos 4 sistemas numéricos apresentados.

Decimal	Binário	Octal	Hexadecimal
0d	0000b	0o	0h
1d	0001b	1o	1h
2d	0010b	2o	2h
3d	0011b	3o	3h
4d	0100b	4o	4h
5d	0101b	5o	5h
6d	0110b	6o	6h
7d	0111b	7o	7h
8d	1000b	10o	8h
9d	1001b	11o	9h
10d	1010b	12o	Ah
11d	1011b	13o	Bh
12d	1100b	14o	Ch
13d	1101b	15o	Dh
14d	1110b	16o	Eh
15d	1111b	17o	Fh

Tabela 1 – Equivalência entre os sistemas numéricos. A letra em subscrito identifica a base em que o número está escrito
Fonte: adaptado de Ferrari e Cechinel (2008)

Para representar um determinado número decimal, a quantidade de algarismos varia conforme o sistema de numeração que será utilizado. No sistema decimal, o maior número é representado por 10N, em que “N” é a quantidade de algarismos utilizados. Imagine agora que os números serão restringidos a dois algarismos no máximo. Logo, 102 = 100, ou seja, quaisquer números de 0 a 99.

No sistema binário, da mesma forma, o maior número é representado por 2N, em que “N” é a quantidade de algarismos utilizados. Portanto, para representar números binários com 4 dígitos, tem-se 24 = 16, ou seja, quaisquer números de 0 a 15.

Após estudar como o computador lida com os dados internamente, é hora conhecer os tipos de dados. Para que esse aprendizado seja ainda mais significativo, a partir deste momento, será utilizada a ferramenta Portugol Webstudio para a prática dos novos conhecimentos.

Tipos Primitivos

Em um computador, os dados devem ser armazenados de acordo com o tipo de informação que ele representa e com a operação a ser realizada com eles. O uso da representação correta otimiza recursos computacionais, propiciando um processamento mais rápido. Os tipos de dados mais comuns encontrados na maioria das linguagens de programação são a base de como a informação deve ser armazenada. Veja esses tipos mais comuns elencados a seguir.

Inteiro

São os números pertencentes ao conjunto dos Números Inteiros ($\mathbb{Z}$ = {..., - 3, - 2, - 1, 0, 1, 2, 3, ...}), ou seja, não têm parte fracionária e podem ser negativos, nulos ou positivos. Exemplos do cotidiano: 10 pratos, tênis nº 41, 21 anos, 0 pessoa na fila, punição de -3 pontos no jogo.

Real

São os números pertencentes ao conjunto dos Números Reais (R = {..., -2, -½, -1, 0, 1, √2, ½, 2, 3, π, ...}), ou seja, que têm parte fracionária. Na computação, também são chamados de números de ponto flutuante. Os números reais são formados pela

união dos conjuntos dos números Naturais, Inteiros, Racionais e Irracionais (R = N U Z U Q U I). Exemplos do cotidiano: saldo de conta-corrente de R$ -735.59, valor de π = 3,141592653589..., ½ xícara de açúcar, 1,5 kg de carne, temperatura de - 2.8 C.

Caractere

Representa qualquer caractere da união dos valores pertencentes ao sistema numérico decimal (0...9), dos caracteres alfabéticos (a...z, A...Z) e dos caracteres especiais (“, ‘, !, @, #, $, %, ¨, &, ~, ^, ´, `, ...). Esse conjunto de caracteres é comumente conhecido por caracteres alfanuméricos que são armazenados internamente no computador na forma de números binários utilizando o padrão ASCII (*American Standard Code for Information Interchange*). Esse padrão nada mais é do que um conjunto de códigos utilizados para representar os diversos caracteres existentes (alfanuméricos) por meio de combinações de 8 *bits*. Exemplos com um caractere apenas: ‘C’, ‘1’, ‘~’, ‘$’, ‘§’. Observe que nesses exemplos foram utilizadas aspas simples (‘) para indicar o início e o fim da representação do caractere.

Cadeia

Representa uma formação literal com 2 ou mais caracteres alfanuméricos e é necessária em situações que se precisa armazenar um texto ou uma quantidade grande de caracteres. Exemplos do cotidiano: “Rua Venâncio Aires, n. 2189”, “João Francisco”, “Programa Mais Você”, “XYZ-3131”. Note que foram utilizadas aspas duplas (“) para indicar o início e o fim de uma cadeia de caracteres e que os espaços em branco que separam cada palavra também são caracteres.

Lógico

O tipo lógico é representado por apenas dois valores: verdadeiro ou falso (*true* ou *false*) ou simplesmente V ou F, nunca os dois ao mesmo tempo. Também pode representar os estados aberto ou fechado, ligado ou desligado, aberto ou fechado etc. Exemplos do cotidiano: rádio ligado, luz desligada, portão aberto, fogo aceso.

A seguir, na tabela 2, veja o quadro comparativo da definição dos tipos de dados conceituados anteriormente com os tipos de dados primitivos possíveis na ferramenta Portugol Webstudio.

Tipo de Dado	Conceitual	Portugol Webstudio
Inteiro	Inteiro	`inteiro`
Real	Real	`real`
Caractere	Caractere	`caractere`
Cadeia	Cadeia	`cadeia`
Lógico	Lógico	`logico`

Tabela 2 – Comparação entre os tipos de dados conceituados com os dados primitivos possíveis na ferramenta Portugol Webstudio

Perceba que no Portugol Webstudio os acentos devem ser suprimidos.

É hora de realizar um desafio!

Classifique os dados especificados a seguir de acordo com seu tipo, assinalando com **I** os dados do tipo inteiro, com **R** os reais, com **C** os caracteres, com **Ca** cada cadeia de caracteres, com **L** os lógicos e com **N** aqueles em que não for possível definir um tipo de dado. Para resolver o exercício, lembre-se de que as aspas simples definem um caractere e as aspas duplas definem uma cadeia.

(___) 0.21
(___) 1
(___) V

(___) “0.”
(___) 1%
(___) “José”
(___) 0.35
(___) .F.
(___) -0.001
(___) F
(___) 3257
(___) ‘a’
(___) “+3257”
(___) 3257.
(___) “-0.0”
(___) “.F.”
(___) -3
(___) .V.
(___) .V
(___) “abc”
(___) ‘F’
(___) C
(___) Maria
(___) -36

Neste momento, é importante saber que os tipos citados estão presentes na maioria das linguagens de programação, como Java, C# e C++. No entanto, outros tipos podem aparecer, tais como: *byte*, *short*, *long*, *double*, *Array*, *List* etc.

Agora que você aprendeu que todos os dados têm um tipo, precisa aprender como esses dados são armazenados na memória RAM do computador.

Constantes e Variáveis

O algoritmo inevitavelmente lidará com a manipulação de dados, seja do tipo que for. Em alguns momentos, esses dados serão constantes (não se modificam do início ao fim do algoritmo ou programa) e em outros variarão (se modificam seja por cálculos internos no algoritmo, seja por interferência do usuário).

Para melhor esclarecer, lembre-se do ensino médio. Imagine um algoritmo que precisa calcular a área de um círculo. No cálculo, usa-se a fórmula A=πr^2, em que o valor de “π"é **constante**, pois é sempre igual a 3,1416..., e o raio “r” é **variável**, pois

pode assumir diferentes valores a cada cálculo da área. Seguindo a mesma lógica, o resultado, que é a área (“A”) calculada para diferentes “r”, também é **variável**.

Em resumo, pode-se ter em algoritmos:

Constantes: Um dado deve ser definido como constante quando o seu valor não muda com o passar do tempo da execução de um algoritmo, ou seja, sempre será o mesmo valor desde o início até o fim da execução.

Variável: Um dado deve ser definido como variável quando o seu valor pode ser alterado durante a execução de um algoritmo.

O ato de criar uma variável é conhecido como **declaração de variável**. É um passo fundamental em algoritmo (também em Portugol e em várias linguagens de programação), pois não é possível usar uma variável sem antes declará-la, e ao declará-la, ou seja, ao criá-la, é necessário dar-lhe um nome para que possa ser referenciada posteriormente no código. Deve-se informar um tipo para representar que tipo de informação ela poderá carregar.

Apesar da possibilidade de serem usadas constantes no código sem declaração (um texto fixo ou um número, por exemplo), também há a possibilidade de declará-las, o que significa que se pode identificar um valor constante com um nome mais adequado.

Tecnicamente declarar uma variável significa reservar um espaço na memória do computador para guardar um determinado valor, como se fossem abertos um contêiner ou uma gaveta própria onde se guarda e se recupera um dado (um texto, um número necessário a um cálculo).

Se você pensar em declaração de variáveis e de constantes, como criar contêineres para armazenamento de dados, entre elas não há distinção, pois quando você declara no algoritmo uma variável ou uma constante, de qualquer maneira está

apenas reservando um espaço na memória RAM. Veja na figura 1 uma projeção gráfica dessa reserva na memória do computador para o armazenamento de um número inteiro.

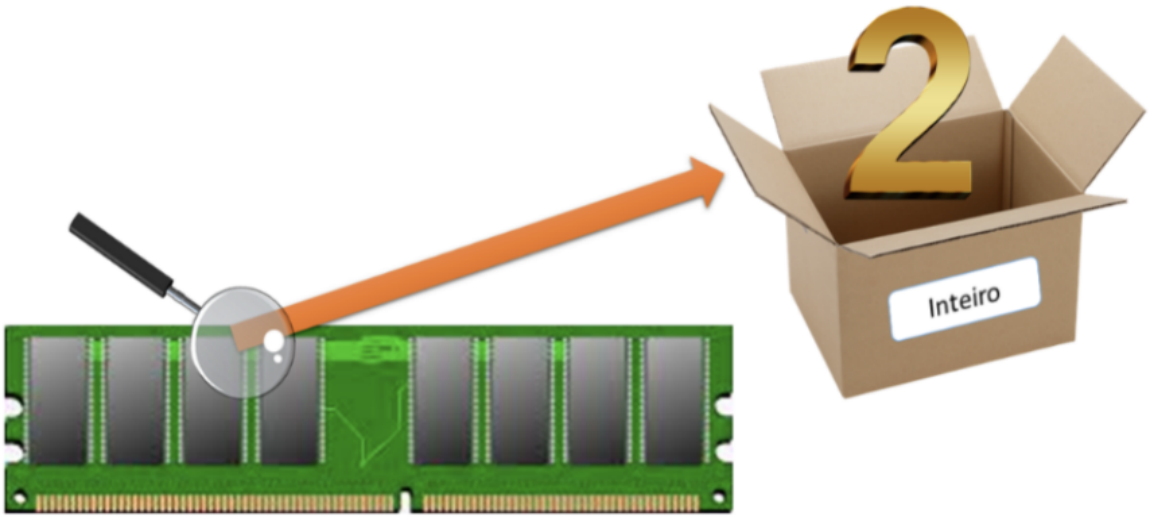


Figura 1 – Projeção gráfica da reserva de um pequeno espaço em memória RAM para o armazenamento de um número inteiro

À esquerda e abaixo, há um retângulo verde com oito pequenos retângulos pretos que juntos representam um módulo de memória RAM. Sobre um dos pequenos retângulos pretos que representam um *chip* de memória há uma lente de aumento (lupa). Dessa lupa, sai uma seta laranja para a diagonal superior direita que aponta para uma caixa aberta. Nessa caixa há uma etiqueta branca colada nela com a palavra “Inteiro”. O algarismo número “dois” está sendo colocado no interior dessa caixa. A ideia central da projeção é entender que uma variável ou constante é um pequeno espaço na memória do computador capaz de armazenar dados de acordo com o tipo que o algoritmo vai utilizar.

Esse espaço em memória será referenciado sempre pelo nome dado à variável ou à constante no momento da declaração.

Você verá mais detalhadamente a declaração de variáveis e constantes em Portugol. Para exemplificar essa discussão, veja o trecho de código a seguir:

```
inteiro meuNumero //declaração de variável
const real MEIO = 0.5 //declaração de constante
```

Na primeira linha, é declarada a variável “meuNumero” de tipo “inteiro”. Isso significa que, a partir do momento dessa declaração, há um espacinho na memória que pode receber números inteiros e que poderá ser referenciado pelo programador com o nome “meuNumero” até o fim do código. Usando esse nome, é possível ler o valor, além de trocá-lo, se necessário.

Na segunda linha do exemplo, analogamente, há a declaração da constante “MEIO”, nome que também referencia a memória em um espaço que contém o valor numérico “0.5”. Por ser constante, “MEIO” não pode ter seu valor trocado durante o código. Não é obrigatório criar constantes todo momento. No exemplo, seria possível usar o valor “0.5” durante o código a qualquer momento, sem precisar de “MEIO”, mas, em alguns casos, a declaração de constantes pode tornar o código mais legível e mais fácil de alterá-lo.

Uma vez que você entender o que são variáveis e constantes, passará a aprender como identificá-las, ou seja, como declará-las corretamente nos algoritmos.

Manipulação dos dados

Identificação

Para que você possa manipular dados no computador, é necessário que eles sejam identificados, ou seja, tenham um nome atribuído a eles. Portanto, conclui-se que, toda vez que você precisar utilizar um dado, deverá recuperá-lo pelo seu nome.

Para criar os nomes de identificadores, algumas regras, que estão descritas a seguir, devem ser seguidas. Essas regras também podem ser utilizadas no Portugol Webstudio.

Regras para criar os nomes de identificadores

1. Deve iniciar com um caractere do alfabeto (letra).
2. Pode ser seguido por um ou mais caracteres alfanuméricos (letras e números).
3. Não pode conter caracteres especiais nem espaços, com exceção do sublinhado (‘_’).
4. Não pode conter palavras reservadas (palavras próprias da linguagem de programação, como instruções ou comandos, tipos de variáveis etc.).
5. Não pode conter caracteres do alfabeto latino (ç, ~, ´, `, ^, ¨).
6. Evitar iniciar com letras maiúsculas. No caso de nomes compostos, a segunda palavra deve iniciar por letra maiúscula (*camel case*).

Identificadores válidos: nome, nomeCompleto, cpf, rg, qtde_filhos, BJ33F, num_infectados.

Identificadores inválidos: 1dia, (alpha), feliz!, nota final, a*b, nomeCapacitação, título.

Chegou a hora de mais um desafio!

Assinale com **C** os identificadores corretos e com **X** os incorretos. Justifique indicando o que está errado nos identificadores assinalados com **X**.

(___) valor01
(___) _c10
(___) media*do*aluno
(___) a1b2c3
(___) 5 x 7
(___) nome_pai
(___) m/s
(___) pi
(___) cnpjda empresa
(___) nomeCompletoMae
(___) numSala
(___) “nota”
(___) Bah!
(___) cálculoRescisão

É muito importante que se crie o hábito de dar nomes que expressem o conteúdo armazenado em um identificador, pois isso facilita a leitura do algoritmo e torna seu processo de documentação muito menos trabalhoso.

Imagine que você precisará armazenar a quantidade, a descrição e o valor unitário de um produto. Utilizando a boa prática, deverá nomear os identificadores da seguinte forma: qtdeProduto, descrProduto e vlrUnitProduto. Lembre-se: se você utilizar nomenclaturas aleatórias (q, d e v, por exemplo), será muito mais difícil sua compreensão e sua utilização correta.

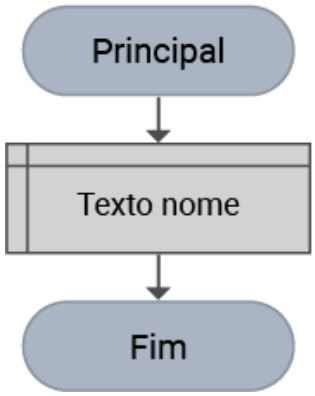
Definição ou declaração

Todo dado será manipulado ou acessado por meio do seu identificador, desse modo, deve-se inicialmente nomear corretamente o identificador e em seguida definir o tipo de dado que o identificador armazenará. As variáveis são definidas em função do tipo de dado que receberão. Já as constantes são definidas em função do valor.

Toda vez que se declara um identificador e define-se um tipo de dado (inteiro, real, caractere, cadeia ou lógico) para ele, esse identificador estará limitado a armazenar somente valores daquele tipo.

A declaração de variáveis para a escrita de algoritmos é realizada conforme a tabela 3 apresentada a seguir.

Definição ou Declaração de Variáveis

Uma variável	 Figura 2 – Fluxograma com definição de uma variável No passo 1, há uma elipse contendo o texto “Principal”, que indica o início de execução algorítmica. No passo 2, há um retângulo com a definição de uma variável chamada “nome” do tipo texto. No passo 3, há uma elipse contendo o texto “fim” que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	``` programa { funcao inicio() { cadeia nome } } ```

Mais de uma variável de mesmo tipo

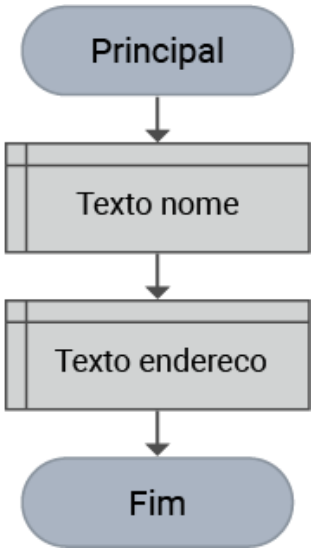
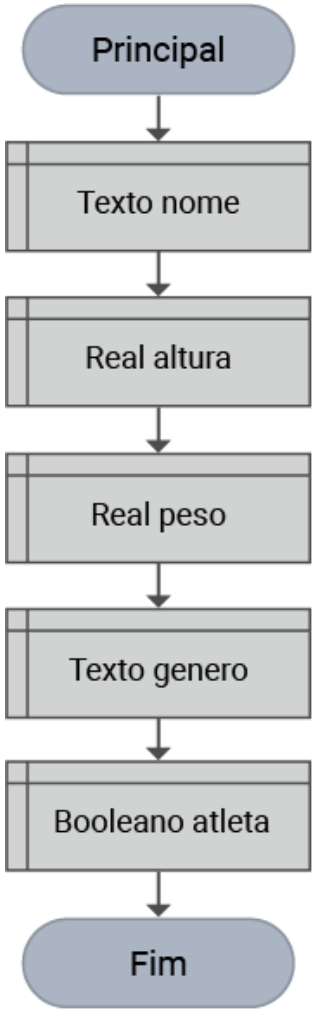


Figura 3 – Fluxograma com definição de variáveis do mesmo tipo

No passo 1, ha uma elipse contendo o texto “Principal” que indica o início de execução algorítmica. No passo 2, há um retângulo com a definição de uma variável chamada “nome” do tipo texto. No passo 3, há um retângulo com a definição de uma variável chamada “endereco” do tipo texto. No passo 4, há uma elipse contendo o texto “fim” que

```
programa {
  funcao inicio() {
    cadeia nome, endereco
  }
}
```

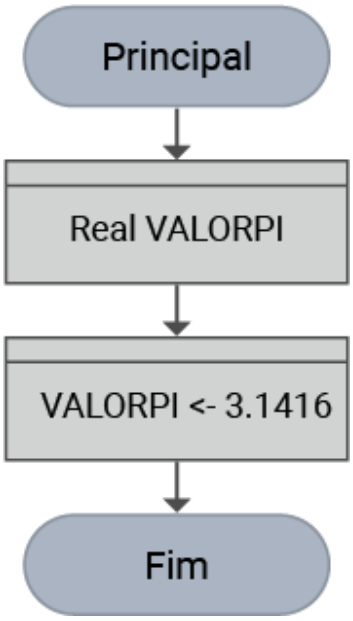
	indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	
Variáveis de tipos diferentes	 Figura 4 – Fluxograma com definição de variáveis de tipos diferentes	``` programa { funcao inicio() { cadeia nome real altura, peso caracter genero logico atleta } } ```

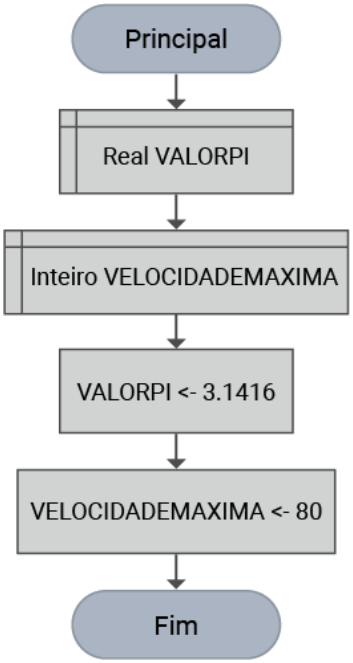
No passo 1, há uma elipse contendo o texto “Principal” que indica o início de execução algorítmica. No passo 2, há um retângulo com a definição de uma variável chamada “nome” do tipo texto. No passo 3, há um retângulo com a definição de uma variável chamada “altura” do tipo real. No passo 4, há um retângulo com a definição de uma variável chamada “peso” do tipo real. No passo 5, há um retângulo com a definição de uma variável chamada “genero” do tipo texto. No passo 6, há um retângulo com a definição de uma variável chamada “atleta” do tipo booliano. No passo 7, há uma elipse contendo o texto “fim”

	que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	

Tabela 3 – Definição ou declaração de variáveis em algoritmos escritos em pseudocódigo

Para a definição de constantes, existe uma pequena diferença em relação às variáveis. Essa diferença se dá pelo fato de serem definidas com base na natureza do seu valor. Observe essa sutil diferença na tabela 4 a seguir.

Definição ou Declaração de Constantes		
Uma constante	 Figura 5 – Fluxograma com definição de uma constante No passo 1, há uma elipse contendo o texto “Principal” que indica o início de uma execução algorítmica. No passo 2, há um retângulo com a definição de uma constante chamada “VALORPI” do tipo real. No passo 3, há um retângulo com a atribuição do valor de “pi” (3.1416) à constante criada no passo 2. No	``` programa { funcao inicio() { const real PI = 3.1416 } } ```

	passo 4, há uma elipse contendo o texto “fim” que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	
Mais de uma constante	 Figura 6 – Fluxograma com definição de mais de uma constante No passo 1, há uma elipse contendo o texto “Principal” que indica o início de uma execução algorítmica. No passo 2, há um	``` programa { funcao inicio() { const real PI = 3.1416 const inteiro VEL_MAXIMA = 80 } } ```

	retângulo com a definição de uma constante chamada “VALORPI” do tipo real. No passo 3, há um retângulo com a definição de uma variável chamada “VELOCIDADEMAXIMA” do tipo inteiro. No passo 4, há um retângulo com a atribuição do valor de “pi” (3.1416) à constante criada no passo 2. No passo 5, há um retângulo com a atribuição do valor 80 à constante criada no passo 3. No passo 6, há uma elipse contendo o texto “fim” que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	

Tabela 4 – Definição ou declaração de constantes em algoritmos

Veja a seguir, na tabela 5, um quadro comparativo de declaração de variáveis e constantes válidas tanto em fluxogramas quanto no Portugol Webstudio.

Variáveis e Constantes Válidas

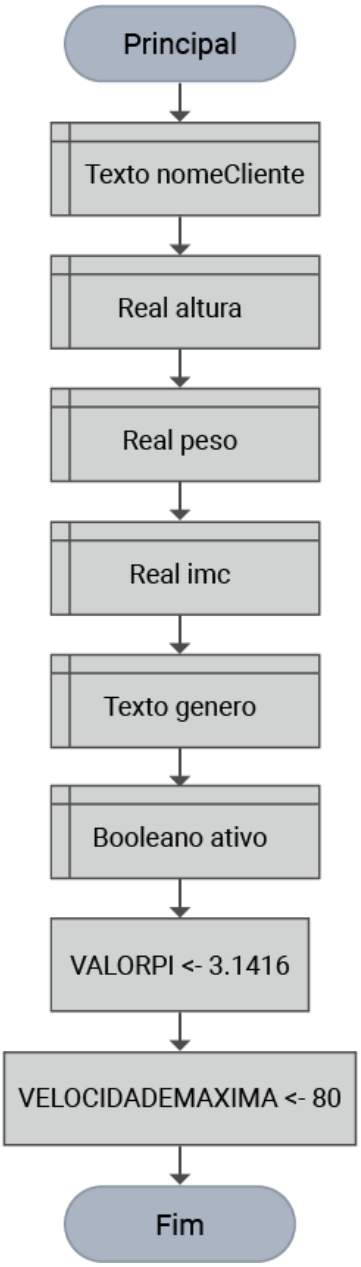
Fluxograma (Flowgorithm)	Portugol Webstudio
	`programa { funcao inicio() { cadeia nomeCliente real altura, peso, imc caracter genero lógico ativo const real VALOR_PI = 3.1416 const int VELOCIDADE_MAXIMA = 80 } }`

Figura 7 – Fluxograma com variáveis e constantes

No passo 1, há uma elipse contendo o texto “Principal” que indica o início de uma execução algorítmica. No passo 2, há um retângulo com a definição de uma variável chamada “nomeCliente” do tipo texto. No passo 3, há um retângulo com a definição de uma variável chamada “altura” do tipo real. No passo 4, há um retângulo com a definição de uma variável chamada “peso” do tipo real. No passo 5, há um retângulo com a definição de uma variável chamada “imc” do tipo real. No passo 6, há um retângulo com a definição de uma variável chamada “genero” do tipo texto. No passo 7, há um retângulo com a definição de uma variável chamada “ativo” do tipo booliano. No passo 8, há um retângulo com a atribuição do valor 3.1416 à constante “VALORPI”. No passo 9,

há um retângulo com a atribuição do valor 80 à constante “VELOCIDADEMAXIMA”. No passo 10, há uma elipse contendo o texto “fim” que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	

Tabela 5 – Comparativo de declarações de variáveis e constantes

Note que as constantes são identificadas com todas as letras em maiúsculas e caracterizam um padrão de projetos de acordo com a linguagem de programação. No Portugol Webstudio, esse padrão também é utilizado. Veja o trecho da documentação que cita essa recomendação na tabela 6.

Para declarar uma constante, basta adicionar a palavra reservada **const** seguida do tipo de dado pelo nome da constante e atribuir um valor a ela.

Exemplo de Sintaxe

```
1.  const inteiro NOME_DA_CONSTANTE = 3
2.  const real NOME_DA_CONSTANTE2 = 45
```

Por uma questão de convenção, é aconselhável deixar o nome da sua constante em caixa-alta (todas as letras maiúsculas).

Tabela 6 – Comparativo de declarações de variáveis e constantes

Fonte: Portugol Webstudio, tópico da “Declaração de Constantes” de sua documentação oficial.

Disponível em: https://portugol-webstudio.cubos.io/ide/ajuda#. Acesso em: 27 dez. 2021

Atribuição de valores

Depois de realizada a declaração de uma variável ou constante, é possível iniciar a manipulação de dados utilizando esse identificador. A forma mais básica de manipulação é a atribuição de valores à variável ou à constante, representados pelo sinal de igual (=), e seu significado no algoritmo é “recebe”. Veja exemplos de declaração e como realizar sua correta leitura na tabela 7.

Atribuição de Valores às Variáveis		
	Portugol Webstudio	**Leitura**
Variáveis	`programa {` `  funcao inicio() {` `  cadeia nomeCliente` `  nomeCliente = "João da Silva"` `  }` `}`	Linha 3: lê-se variável nome tipo cadeia. Linha 4: lê-se variável nome recebe João da Silva.
Constantes	`programa {` `  funcao inicio() {` `  const real PI = 3.1416` `  }` `}`	Linha 3: lê-se constante tipo real PI recebe 3.1416.

Tabela 7 – Leitura da atribuição de valores às variáveis e constantes

Lembre-se que depois de declarada uma variável ou constante, esta só poderá receber o mesmo tipo de dado definido para ela. Veja a seguir como devem ser atribuídos os valores às variáveis ou às constantes na tabela 8.

Atribuição de Valores		
Tipo de Dado	**Portugol Webstudio**	**Leitura**
Cadeia	`programa { funcao inicio() { cadeia nomeCliente nomeCliente = "João da Silva" } }`	Para atribuir uma cadeia de caracteres a uma variável do tipo cadeia, é preciso que o valor seja delimitado por aspas duplas (" ").
Caractere	`programa { funcao inicio() { caractere genero genero = "M" } }`	Para atribuir um caractere a uma variável do tipo caractere, é preciso que o valor seja delimitado por aspas simples (' ').
Real	`programa { funcao inicio() { real altura altura = 1.85 } }`	Para atribuir um número decimal (números com vírgula) a uma variável do tipo real, é preciso apenas informar o valor substituindo a vírgula por ponto.

Inteiro	`programa { funcao inicio() { inteiro idade idade = 21 } }`	Para atribuir um número inteiro (números sem vírgula) a uma variável do tipo inteiro, é preciso apenas informar o valor.
Lógico	`programa { funcao inicio() { logico ligado ligado = verdadeiro } }`	Para atribuir verdadeiro ou falso a uma variável do tipo lógico, é preciso apenas informar o valor.

Tabela 8 – Atribuição de valores

Caso ocorra a tentativa de atribuir um valor diferente do tipo estabelecido ao identificador, na maioria dos casos, poderão ocorrer erros de compilação, execução, perda de dados ou resultado errado de algum cálculo. Observe os exemplos com alguns erros frequentes no aprendizado de algoritmos.

Algoritmo 1

```
programa {
  funcao inicio() {
    cadeia meuNome //declaração da variável nome
    meuNome = João da Silva //atribuição de valor a variável meuNome. Observe que o n

    escreva(meuNome) //instrução para exibir na tela do computador o valor armazenado
  }
}
```

Saída - Algoritmo 1

```
Traduzindo erro sintático: org.antlr.runtime.NoViableAltException
line 4:20 - no viable alternative at input 'ã'
Contexto atual: referencia

ERRO: O caracter 'ã' não é reconhecido em uma referência. Linha: 4, Coluna: 20

Programa finalizado.
```

No algoritmo 1, o Portugol gerou um erro de compilação. Ele não reconheceu João da Silva como uma cadeia de caracteres válida, ou seja, uma *string*. **Para corrigir, basta colocar o João da Silva entre aspas duplas**.

Algoritmo 2

```
programa {
  funcao inicio() {
    inteiro valor //declaração da variável valor
    valor = 5.15 //atribuição de um valor decimal a variável valor que é do tipo inte

    escreva(valor) //instrução para exibir na tela do computador o valor armazenado n
  }
}
```

Saída - Algoritmo 2

```
AVISO: O valor da expressão à direita da atribuição será truncado. Linha: 4, Coluna: 1


5
Programa finalizado.
```

No algoritmo 2, não houve um erro explícito como no exemplo anterior. No Portugol, foi emitido um aviso (*warning*) informando que o valor foi truncado, ou seja, como a variável é do tipo inteiro e o valor atribuído foi um número decimal, apenas a porção inteira foi recebida pela variável. Veja na saída que foi impresso na tela o valor 5.

Para corrigir, há duas soluções possíveis: a primeira é mudar o valor da atribuição de um número decimal para um inteiro; a segunda é mudar o tipo de variável para real.

Lembre-se de que os valores podem ser informados pelo usuário, mas este não sabe o tipo de dado esperado. Veja a seguir o mesmo exemplo com dados informados pelo usuário.

Algoritmo 3

```
programa {
  funcao inicio() {
    inteiro valor //declaração da variável valor

    escreva("Digite um valor: ") //Mensagem impressa na tela para o usuário digitar u
    leia(valor) //Leitura do valor digitado pelo usuário que será armazenado na variá
    escreva("Valor digitado foi: ", valor) //impressão em tela do texto "Valor digita
  }
}
```

Saída - Algoritmo 3

```
Digite um valor: 7.5

Erro de execução: java.lang.NumberFormatException: For input string: "7.5"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
    at java.base/java.lang.Integer.parseInt(Integer.java:770)
    at br.univali.portugol.Console.solicitaEntrada(Console.java:458)
    at br.univali.portugol.nucleo.programa.Programa.leia(Programa.java:1035)
    at br.univali.portugol.nucleo.programa.Programa.leiaInteiro(Programa.java:1008)
    at programas.Programa1637012049532.executar(Programa1637012049532.java:28)
    at br.univali.portugol.nucleo.programa.TarefaExecucao.run(TarefaExecucao.java:44)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:5
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecuto
    at java.base/java.lang.Thread.run(Thread.java:834)

Linha: 6, Coluna: 8

Programa finalizado.
```

No algoritmo 3, houve um erro no tempo de execução informado imediatamente após a entrada de dados. Nesse caso, foi declarada a variável de tipo inteiro “valor”, e o valor digitado pelo usuário foi um valor real 7.5 (no Portugol, o ponto é o separador de casas decimais). Após pressionada a tecla “enter”, a exceção “NumberFormatException” foi levantada, pois, ao tentar converter o valor digitado para inteiro e armazená-lo na variável de tipo inteiro, o Portugol verificou que eram tipos diferentes de dados. **Para evitar que essas situações aconteçam, informe ao usuário o tipo de dado que ele deve digitar ou o formato esperado**.

Utilizando o Portugol Webstudio, tente corrigir os algoritmos 1, 2 e 3 apresentados acima.

Uma variável pode armazenar apenas um valor por vez, ou seja, toda vez que um novo valor é atribuído à variável, o valor anterior será perdido. Observe a seguir na tabela 9.

Restrição das Variáveis	
Fluxograma (Flowgorithm)	**Portugol Webstudio**

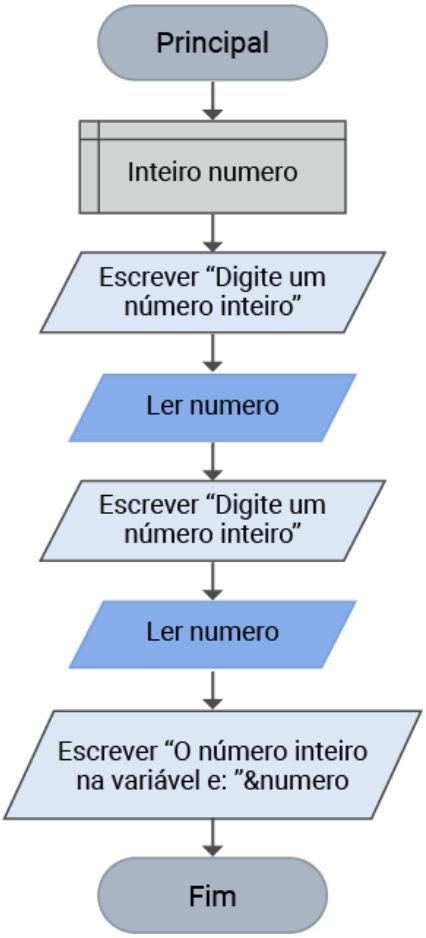
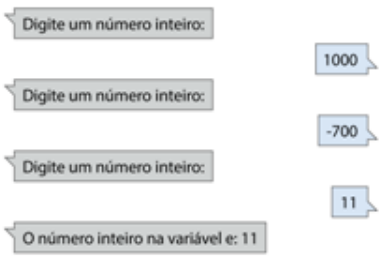


Figura 8 – Fluxograma

```
programa {
   funcao inicio() {
      inteiro numero

      escreva("Digite um número inteiro: ")
      leia(numero)
      escreva("Digite um número inteiro: ")
      leia(numero)
      escreva("Digite um número inteiro: ")
      leia(numero)
      escreva("O número inteiro na variável é: ", n
   }
}
```

No passo 1, há uma elipse contendo o texto "Principal" que indica o início de uma execução algorítmica. No passo, 2 há um retângulo com a definição de uma variável chamada "numero" do tipo inteiro. No passo 3, há um retângulo inclinado para a direita com a instrução "escrever" com a mensagem

“Digite um número inteiro:”, que solicita ao usuário para ele digitar um número inteiro. No passo 4, há a instrução “ler número”, que armazenará o valor digitado pelo usuário na variável “número”, criada no passo 2. No passo 5, há um retângulo inclinado para a direita com a instrução “escrever”, com a mensagem “Digite um número inteiro:”, que solicita ao usuário para ele digitar um número inteiro. No passo 6, há a instrução “ler número”, que vai armazenar o valor digitado pelo usuário na variável “número”, criada no passo 2. No passo 7, há um retângulo inclinado para a direita com a instrução “escrever”, com a mensagem “Digite um número inteiro:”, que solicita ao usuário para ele digitar um número inteiro. No passo 8, há a instrução “ler número”, que vai armazenar o valor digitado pelo usuário na variável “número”, criada no passo 2. No passo 9, há um retângulo inclinado para a direita com a instrução

“escrever”, com a mensagem “O número inteiro na variável é: ”, seguido de & (utilizado para concatenar textos e valores armazenados em variáveis) e a variável “numero”. No passo 10, há uma elipse contendo o texto “fim”, que indica o final da execução algorítmica. Todas as figuras são ligadas por setas apontando para o próximo passo.	
 (objetos/figura_9_grande.png) Figura 9 – Resultado da execução algorítmica No passo 1, há uma caixa de diálogo com a mensagem “Digite um número inteiro: ”. No passo 2, há uma caixa de diálogo com o valor 1000 digitado pelo usuário. No passo 3, há uma caixa de diálogo com mensagem “Digite um	``` Digite um número inteiro: 1000 Digite um número inteiro: -700 Digite um número inteiro: 11 O número inteiro na variável é: 11 Programa finalizado. ```

número inteiro: ". No passo 4, há uma caixa de diálogo com o valor -700 digitado pelo usuário. No passo 5, há uma caixa de diálogo com mensagem "Digite um número inteiro: ". No passo 6, há uma caixa de diálogo com o valor 11 digitado pelo usuário. No passo 7, há uma caixa de diálogo com a mensagem "O número inteiro na variável é 11".	

Tabela 9 – Restrições das variáveis

No algoritmo apresentado na tabela, há os seguintes passos:

1Início do programa.

2Declaração da variável numero do tipo inteiro.

3Escrita na tela de uma mensagem de texto para o usuário digitar um número inteiro.

4O algoritmo lê o valor digitado (1000) e armazena-o na variável numero.

5Escrita na tela de uma mensagem de texto para o usuário digitar um número inteiro.

6O algoritmo lê o valor digitado (-700) e armazena-o na variável numero.

7Escrita na tela de uma mensagem de texto para o usuário digitar um número inteiro.

8O algoritmo lê o valor digitado (11) e armazena-o na variável numero.

9Escrita na tela de uma mensagem de texto O número inteiro na variável é: seguido do valor da variável.

10Fim do programa.

Na primeira interação do usuário, a variável recebeu o valor 1000. Na segunda interação do usuário, a variável recebeu o valor -700 e o valor 1000 foi eliminado. Na terceira e última interação, o usuário digitou 11 e obviamente o valor -700 foi substituído por 11, que é o valor impresso na tela.

Perceba que, para escrever o valor contido na variável, esta não precisa ficar entre aspas. Se optar por escrever uma mensagem para o usuário, esta deve sempre estar entre aspas duplas (" ") e com a vírgula após a variável.

Um erro muito comum é a não distinção entre nome de variável e *string*. Veja um exemplo: quando é colocado o código meuNumero, é muito diferente de colocar “meuNumero”. No primeiro, recupera-se o valor armazenado na variável meuNumero, já o segundo é apenas uma constante textual.

Agora você dará mais um passo na aprendizagem dos algoritmos, pois já sabe como atribuir corretamente os valores a variáveis e constantes. O mais novo desafio é compreender e utilizar corretamente as expressões aritméticas.

Expressões

Expressões aritméticas

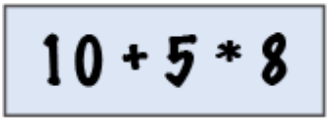
São aquelas em que **os operadores são aritméticos e os operandos são valores do tipo inteiro ou real**. Esses valores numéricos podem ser utilizados por meio dos seus valores absolutos ou manipulados por meio de identificadores constantes ou variáveis.

As operações aritméticas fundamentais são: adição, subtração, multiplicação, divisão e resto da divisão. A tabela 10 apresenta os operadores para cada uma dessas operações aritméticas que o Portugol utiliza.

Tipo de Dado	Símbolo	Prioridade
Adição	+	1
Subtração	-	1
Multiplicação	*	2
Divisão	/	2
Resto da Divisão	%	2

Tabela 10 – Tabela dos operadores aritméticos e sua prioridade

A prioridade indica qual operação deve ser realizada primeiro quando houver várias juntas. Quanto maior a prioridade, antes a operação ocorre. Veja o exemplo a seguir.



Nessa operação, a multiplicação tem prioridade 2 e deverá ocorrer antes da soma, logo o resultado esperado é 50. Veja como fica em Portugol com interação do usuário:

```
programa {
   funcao inicio() {
      inteiro valorA, valorB, valorC, resultado //criação de variáveis do tipo inte
iro

      escreva("Digite o primeiro valor inteiro: \n") //solicitação de digitação do
 1º valor
      leia(valorA)                                   //leitura do 1º valor digitado
pelo usuário
      escreva("Digite o segundo valor inteiro: \n")  //solicitação de digitação do
 2º valor
      leia(valorB)                                   //leitura do 2º valor digitado
pelo usuário
      escreva("Digite o terceiro valor inteiro: \n") //solicitação de digitação do
 3º valor
      leia(valorC)                                   //leitura do 3º valor digitado
pelo usuário

      resultado = valorA+valorB*valorC //variável resultado recebe o resultado da o
peração aritmética

      escreva(valorA , " + " , valorB , " * " , valorC , " = " , resultado) //escri
ta da expressão e seu resultado
   }
}
```

Nesse algoritmo foram incorporados alguns caracteres ASCII ainda desconhecidos.

Verifique nas instruções “escreva”, ao final da mensagem e dentro das aspas duplas, a presença do elemento “**\n**” (nova linha). Por isso na execução do algoritmo, o valor é digitado na linha seguinte.

O resultado da execução é:

```
Digite o primeiro valor inteiro:
10
Digite o segundo valor inteiro:
5
Digite o terceiro valor inteiro:
8
10 + 5 * 8 = 50
Programa finalizado.
```

Agora você deve estar se questionando sobre a última linha do código, não é? Para entendê-la, acompanhe os números em forma de comentário na linha inferior à instrução.

```
  escreva(valorA , " + " , valorB , " * " , valorC , " = " , resultado)
//    1      2       3        4       5        6       7         8
```

Legenda:

1Instrução **escreva**. Função utilizada para imprimir expressões, *strings* e valores de variáveis na tela.

2**valorA**. Variável inteira, que armazena o valor inteiro 10. Será substituída por esse valor.

3**" + "**. Mensagem do programador para ser exibida depois do valor 10, que representa, em forma de texto, o sinal da soma.

4**valorB**. Variável inteira que armazena o valor inteiro 5. Será substituída por esse valor.

5**" * "**. Mensagem do programador para ser exibida depois do valor 5, que representa, em forma de texto, o sinal da multiplicação.

6**valorC**. Variável inteira que armazena o valor inteiro 8. Será substituída por esse valor.

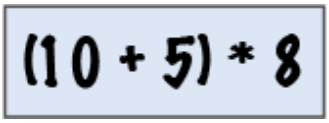
7**" = "**. Mensagem do programador para ser exibida depois do valor 8, que representa, em forma de texto, o sinal de igual.

8**resultado**. Variável inteira que armazena o valor inteiro 50, que é o resultado da operação aritmética calculada. Será substituída por esse valor.

Cada um dos itens acima foi separado por uma vírgula para serem impressos individualmente na mesma ordem que aparecem. No Portugol, pode-se utilizar o sinal de soma (+) dentro do escreva para concatenar (juntar) *strings* com valores. Veja a seguir a mesma expressão utilizando o caractere concatenar:

```
escreva(valorA + " + " + valorB + " * " + valorC + " = " , resultado)
```

Sempre que for necessário mudar a prioridade de uma operação, deve-se utilizar os parênteses. Para o mesmo exemplo, se for realizada a soma antes da multiplicação, a soma deve ser colocada entre parênteses. Observe no exemplo a seguir:



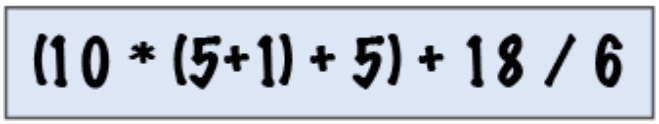
Nessa operação, como a soma está entre parênteses, ela será realizada antes da multiplicação, logo o resultado esperado será 120. Veja no Portugol:

```
programa {
   funcao inicio() {
      inteiro valorA, valorB, valorC, resultado //criação de variáveis do tipo inte
iro

      escreva("Digite o primeiro valor inteiro: \n") //solicitação de digitação do
 1º valor
      leia(valorA)                                   //leitura do 1º valor digitado
pelo usuário
      escreva("Digite o segundo valor inteiro: \n")  //solicitação de digitação do
 2º valor
      leia(valorB)                                   //leitura do 2º valor digitado
pelo usuário
      escreva("Digite o terceiro valor inteiro: \n") //solicitação de digitação do
 3º valor
      leia(valorC)                                   //leitura do 3º valor digitado
pelo usuário

      resultado =(valorA+valorB)*valorC //variável resultado recebe o resultado da
 operação aritmética

      escreva("(", valorA , " + " , valorB , ") * " , valorC , " = " , resultado)
//escrita da expressão e seu resultado
   }
}
```

O resultado da execução é:

```
Digite o primeiro valor inteiro:
10
Digite o segundo valor inteiro:
5
Digite o terceiro valor inteiro:
8
(10 + 5) * 8 = 120
Programa finalizado.
```

Em caso de parênteses dentro de parênteses, os mais internos são realizados antes dos mais externos. Veja abaixo:



Nessa operação, a expressão (5 + 1) será realizada primeiro, pois está dentro da expressão (10 * (5 + 1) + 5). Em seguida, a operação a ser realizada será (10 * 6 + 5), que resultará em (60 + 5) e fornecerá o valor **65**. Terminado o cálculo dos parênteses, a operação segue a ordem de prioridade dos operadores, logo 65 + 18 / 6, cujo resultado final será **68**.

Utilizando o Portugol Webstudio, implemente dois algoritmos para calcular essa expressão citada. O primeiro com atribuição de valores às variáveis pelo próprio programador e outro com interação do usuário.

Exceções em Expressões Aritméticas

Segundo FERRARI e CECHINEL (2008), para a maioria das expressões aritméticas executadas em um algoritmo, é possível associar um valor definido, ou seja, o resultado da expressão propriamente dito. Por exemplo, a expressão 2 + 3, depois de realizada, tem um valor resultante igual a 5, e a expressão 10 / 2 tem um valor definido de 5. No entanto, nem todas as expressões aritméticas têm um valor definido matematicamente, como o caso de divisões pelo número 0 (zero) ou de raízes quadradas de números negativos. A utilização desse tipo de expressão deve ser sempre evitada com a verificação dos valores que farão parte dela, ou seja, se um denominador é zero ou se o número cuja raiz será extraída é negativo, a operação não deve ser realizada. Exemplos: 15 / 0 ou √(-4). Veja esse primeiro exemplo em Portugol:

```
programa {
   funcao inicio() {

      //declaração de variáveis
      inteiro valorA = 15 //criação de variável do tipo inteiro com inicialização d
e valor
      inteiro valorB = 0  //criação de variável do tipo inteiro com inicialização d
e valor
      inteiro resultado   //criação de variável do tipo inteiro

      //processamento
      resultado = valorA/valorB

      //saída de dados
      escreva(valorA, " / ", valorB, " = ", resultado)

   }
}
```

O resultado da operação é:

```
Erro de execução: Foi efetuada uma divisão por zero.
Linha: 10, Coluna: 6

Programa finalizado.
```

Escrita de operações aritméticas

A partir deste momento, serão apresentadas situações-problema para serem escritas em Portugol e alguns desafios para você exercitar o aprendizado, que é base para a programação. Na codificação dos algoritmos, todas as instruções terão a sua explicação nos comentários ao lado ou antes da instrução.

Situação 1

Imagine que você precisa criar um algoritmo que calcule a idade de uma pessoa a partir do ano de nascimento dela. Para resolver esse problema, inicialmente você precisa saber como calcular a idade a partir do ano de nascimento (Idade = Ano Atual – Ano de Nascimento). Em seguida, basta definir as variáveis de acordo com os tipos de dados necessários. Você já sabe quais tipos você precisa? Pergunte-se: ano e idade têm valores com vírgula? Ficou fácil, são inteiros. Agora você pode “codar” (ou programar).

```
programa {
funcao inicio() {

//declaração de variáveis
inteiro anoAtual, anoNasc, idade //declaração das variáveis

//entrada de dados
escreva("Digite o ano atual [aaaa]: ") //solicitação do ano atual ao usuário inform
ando o modelo de 4 dígitos
leia(anoAtual) //leitura do ano atual digitado pelo usuário

escreva("Digite o ano de nascimento [aaaa]: ") //solicitação do ano do nascimento a
o usuário informando o modelo de 4 dígitos
leia(anoNasc) //leitura do ano de nascimento digitado pelo usuário

//processamento
idade = anoAtual-anoNasc //variavel idade recebe o resultado da subtração entre ano
Atual e anoNasc

//saída de dados
escreva("Sua idade é ", idade, " anos.") //imprime na tela o resultado da operação

}
}
```

O resultado da execução é:

```
Digite o ano atual [aaaa]: 2021
Digite o ano de nascimento [aaaa]: 2002
Sua idade é 19 anos.
Programa finalizado.
```

Situação 2

O seu nutricionista lhe pede para que você calcule o seu índice de massa corpórea, o famoso IMC (IMC = Peso/Altura2). Aqui o grau de dificuldade aumenta, pois há a potenciação. No Portugol, sem o uso de bibliotecas auxiliares, deve-se utilizar a lógica. Qualquer número elevado ao quadrado é ele multiplicado por ele mesmo. Agora você deve definir as variáveis. Quais tipos são necessários? A resposta é fácil, são do tipo real, pois altura e peso geralmente têm a presença de vírgula, portanto o IMC que vai receber esse resultado também deve ser real. Resolva esse probleminha no Portugol!

```
programa {
funcao inicio() {

//declaração de variáveis
real imc, altura, peso //declaração das variáveis

//entrada de dados
escreva("Digite seu peso em Kg [ex: 72.3]: ") //solicitação do peso em Kg com exemp
lo
leia(peso) //leitura do peso digitado pelo usuário

escreva("Digite sua altura em Mt [ex: 1.75]: ") //solicitação da altura em Kg com e
xemplo
leia(altura) //leitura da altura digitada pelo usuário

//processamento
//Utilizamos na linha abaixo os parenteses para priorizar a operação de multiplicaç
ão
imc = peso/(altura*altura) //variavel imc recebe o resultado da divisão do peso pel
o quadrado da altura

//saída de dados
escreva("Seu IMC é ", imc) //imprime na tela o resultado da operação

}
}
```

O resultado da execução é:

```
Digite seu peso em Kg [ex: 72.3]: 70
Digite sua altura em Mt [ex: 1.75]: 1.75
Seu IMC é 22.857142857142858
Programa finalizado.
```

Situação 3

Imagine que você recebeu a tarefa de calcular o aumento de salário de um trabalhador com base em uma porcentagem informada pelo usuário. Essa tarefa precisa exibir na saída o nome, o salário antigo, o salário novo e o valor do aumento. Analise as saídas. Para exibir o nome, deve-se solicitar esse nome ao usuário; para calcular o valor do aumento com base no percentual, deve-se solicitar o salário antigo e o percentual (pois terá um novo salário). Cálculo do valor do aumento dado por Valor do Aumento = Salario Antigo x Percentual (valor digitado / 100). O novo salário é Novo Salario = Salario Antigo + Valor do Aumento. Logo após basta exibir os resultados. Quais os tipos das variáveis a serem utilizadas? Nome é uma sequência de caracteres, portanto uma *string*. Já salário, valor do aumento e percentual são decimais, pois dinheiro tem casas decimais, e o resultado de um cálculo percentual geralmente também é um real. Veja como fica!

```
programa {
funcao inicio() {

//declaração de variáveis
real salarioAntigo, salarioNovo, valorAumento, percentual
cadeia nome

//entrada de dados
escreva("Digite o nome do empregado:\n") //solicitação do nome com nova linha ao fi
nal
leia(nome) //leitura do nome digitado pelo usuário

escreva("Digite o salário antigo [ex: 9999.99]:\n") //solicitação do salário com de
finição de formato
leia(salarioAntigo) //leitura do salário antigo digitado pelo usuário

escreva("Digite o percentual do aumento [ex: 99.99]:\n") //solicitação do percentua
l com definição de formato
leia(percentual) //leitura do percentual digitado pelo usuário

//processamento
//Utilizamos na linha abaixo os parenteses para priorizar a operação divisão do per
centual informado por 100
//variavel valorAumento recebe o resultado da multiplicação do salarioAntigo pela d
ivisão do percentual por 100
valorAumento = salarioAntigo*(percentual/100)

//agora que temos o valor do aumento basta somá-lo ao salario antigo e guarda-lo na
variável salarioNovo
salarioNovo = salarioAntigo + valorAumento

//saída de dados
escreva("Empregado nome: ", nome) //imprime na tela o nome
escreva("\nSalario Antigo: R$ ", salarioAntigo) //imprime na tela o salario antigo
escreva("\nSalario Novo: R$ ", salarioNovo) //imprime na tela o salario novo
escreva("\nValor do Aumento: R$ ", valorAumento) //imprime na tela o valor do aumen
to

}
}
```

O resultado da execução é:

```
Digite o nome do empregado:
Giuliano
Digite o salário antigo [ex: 9999.99]:
1000.00
Digite o percentual do aumento [ex: 99.99]:
10.00
Empregado nome: Giuliano
Salario Antigo: R$ 1000.0
Salario Novo: R$ 1100.0
Valor do Aumento: R$ 100.0
Programa finalizado.
```

Chegou a hora de praticar!

Veja alguns desafios para você praticar.

Crie um algoritmo que calcule a área da circunferência a partir do raio informado pelo usuário.

Desenvolva um algoritmo que faça a conversão de valores de dólares para real com base nas seguintes regras:

- O usuário deverá informar o valor do dólar do dia.
- O usuário deverá informar o valor em dólares que deseja converter para real.
- Na saída, deverá ser exibido o valor em dólares $ XXX.XX e o valor correspondente em R$ XXX.XX.

Faça um algoritmo que receba o valor do salário mínimo, o número de horas trabalhadas, o número de dependentes do funcionário e a quantidade horas extras trabalhadas. Calcule e mostre o salário a receber do funcionário seguindo as regras a seguir:

- O valor da hora trabalhada é igual a 1/5 do salário mínimo.
- O salário do mês é igual ao número de horas trabalhadas vezes o valor da hora trabalhada.
- Para cada dependente, acrescentar 32 reais.
- Para cada hora extra trabalhada, calcular o valor da hora trabalhada acrescida de 50%.
- O salário bruto é igual ao salário do mês mais o valor dos dependentes e mais o valor das horas extras.

O custo de um carro novo ao consumidor é a soma do custo de fábrica, da porcentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que o percentual do distribuidor seja de 28% e dos impostos de 45%, escreva um algoritmo para ler o custo de fábrica de um carro, calcular e escrever o custo final ao consumidor.

Encerramento

Neste material, foram trabalhadas a criação de variáveis e constantes e a maneira correta de identificá-las, além de como realizar a manipulação dos dados.

Para que os conteúdos fossem significativos, foram abordados conceitos em uma linguagem simplificada, seguidos de tabelas comparativas e vários exemplos. Para sua apropriação ser duradoura, vários desafios foram propostos e colocados em ordem de dificuldade para sua evolução no aprendizado da programação. Tudo o que foi feito em conjunto até o momento foi pensado e planejado para ser a base para os novos conhecimentos que estão por vir. Caso as dúvidas surjam, revisite este material, pois ele foi pensado e criado especialmente para você.

Assista ao vídeo com exemplos do conteúdo estudado até aqui.


TDS UC02 - Dados