



UNIDADE II

Estudos Disciplinares
Lógica de Programação

Prof. Me. Marcelo Santos

Introdução

- Fluxograma
- Técnicas básicas de programação

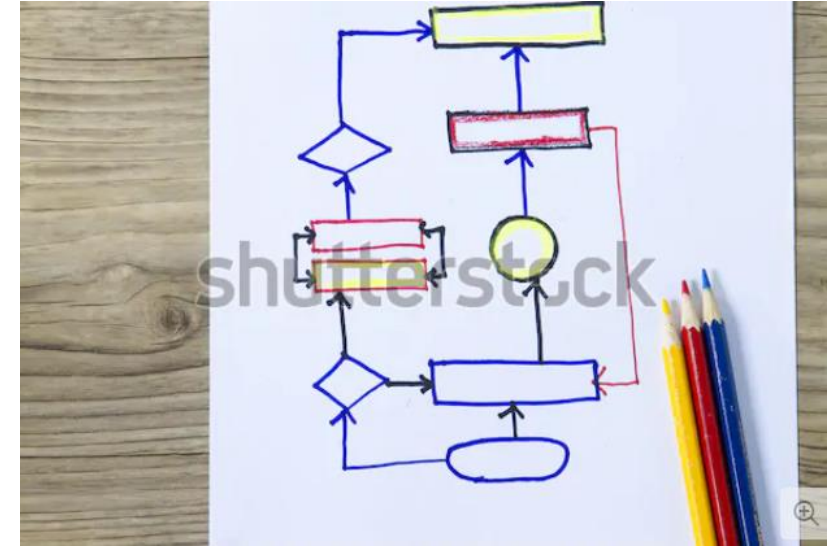
Tipos de dados

- Tipos primitivos (*primitive types*)

Operadores

- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos

Expressões e variáveis

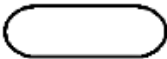
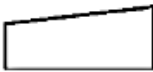




Fonte:

https://www.shutterstock.com/image-photo/process-flowchart-complete-polygonal-diagram-arrow-667076494?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fimages%2Fsearch%2Ffluxograma%2F

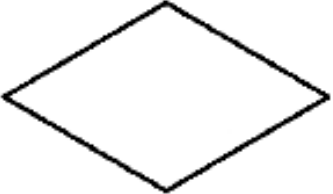


Fluxograma

- Manzano (2019) apresenta os símbolos normalizados que devem ser utilizados na construção do fluxograma e as devidas descrições.

Símbolo	Significado	Descrição
	Terminal <i>Terminator</i>	O símbolo representa a definição de início e fim do fluxo lógico de um programa. Também é utilizado na definição de sub-rotinas de procedimento ou de função.
	Entrada manual <i>Manual input</i>	Representa a entrada manual de dados, normalmente efetuada em um teclado conectado diretamente ao console do computador.
	Processamento <i>Process</i>	Representa a execução de uma operação ou grupo de operações que estabelecem o resultado de uma operação lógica ou matemática.
	Exibição <i>Display</i>	Representa a execução da operação de saída visual de dados em um monitor de vídeo conectado ao console do computador.

Símbolos normalizados – ISO 5807
Fonte: Manzano (2019, p. 36).

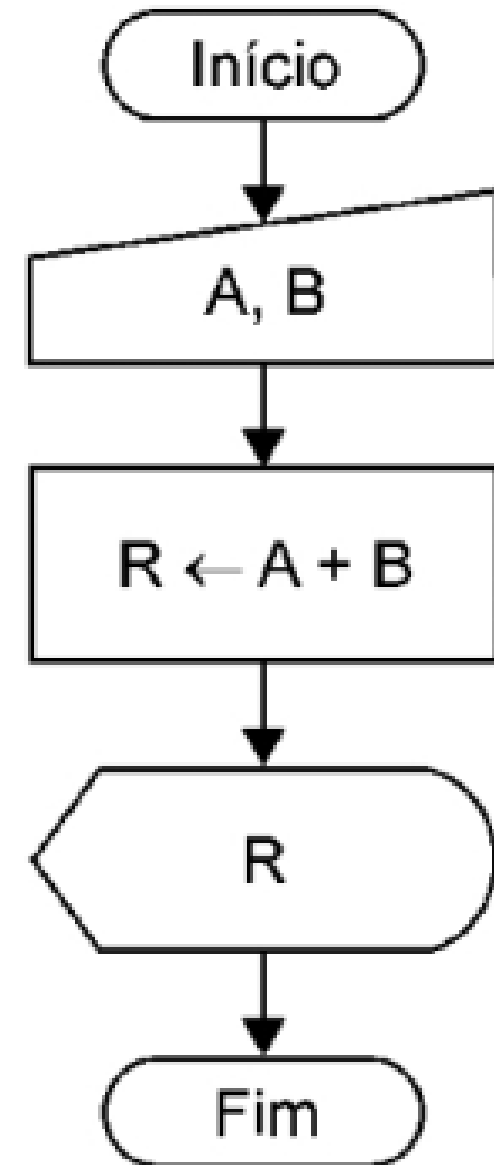
Fluxograma

	Decisão <i>Decision</i>	O símbolo representa o uso de desvios condicionais para outros pontos do programa de acordo com situações variáveis.
	Preparação <i>Preparation</i>	Representa a modificação de instruções ou grupo de instruções existentes em relação à ação de sua atividade subsequencial.
	Processo predefinido <i>Predefined process</i>	Definição de um grupo de operações estabelecidas como uma sub-rotina de processamento anexa ao diagrama de blocos.

Símbolos normatizados – ISO 5807
Fonte: Manzano (2019, p. 36).

Programando um computador

- A próxima figura apresenta a estrutura lógica de operação computacional mais trivial e comum: a estrutura de operação computacional de sequência. Sendo definidas 3 variáveis (A, B e R), pede-se para realizar a soma das variáveis A e B e exibir o resultado (R).



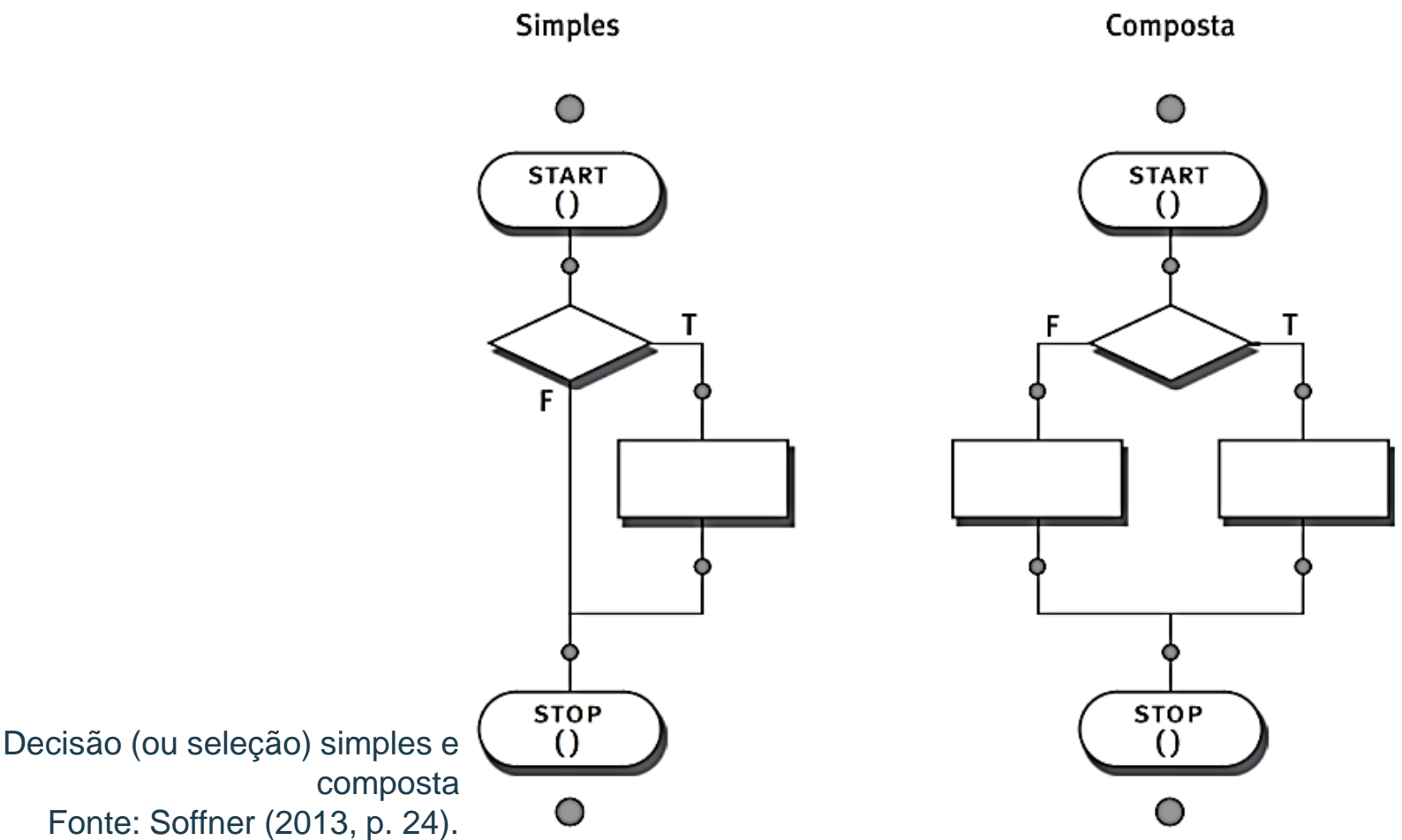
Estrutura de operação
computacional de sequência
Fonte: Manzano (2019, p. 39).

Fluxograma

- A decisão simples testa uma condição e realiza uma ação caso esta seja verdadeira, sem se preocupar em realizar uma ação no caso de verificação da condição oposta.
- Por exemplo, se um número digitado for menor que zero, solicito uma nova digitação; se não for, o programa simplesmente continua na próxima linha abaixo da decisão.
- A decisão composta, ao contrário, tem uma ação prevista em caso de verificação da condição oposta. Por exemplo, se a média de um aluno for maior ou igual a seis, vou imprimir na tela “Aprovado”. Se não for (ou seja, se a média for menor que seis), imprimirei “Reprovado” (SOFFNER, 2013, p. 24).

Fluxograma

Abaixo segue uma imagem que representa a estrutura de operação computacional de decisão simples e a estrutura de operação computacional de decisão composta.

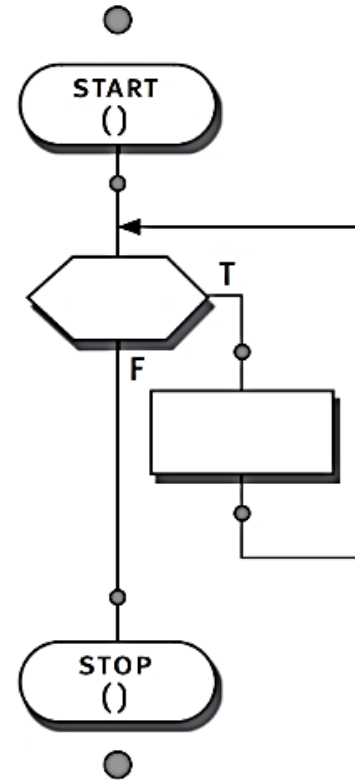


Fluxograma

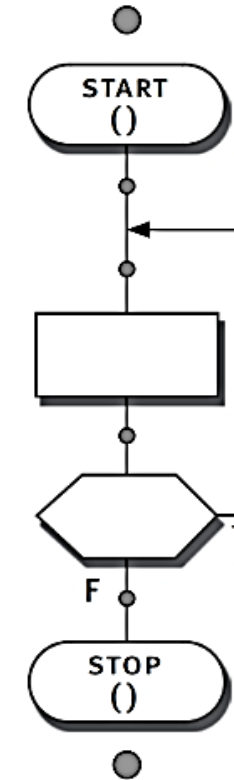
- A repetição com teste no início avalia uma condição antes de executar as ações previstas e repetitivas; se válida, o processamento entra em iteração (*loop*), até que tal condição não seja mais verdadeira, quando o programa seguirá normalmente para o restante das rotinas programadas.
- Essa repetição é perfeita para testes de senhas antes do acesso a funções repetitivas do programa. Já a repetição com teste no fim executará uma ação pelo menos uma vez antes de decidir se ela continuará.
 - É muito utilizada em validações de entradas de dados, antes que se dê a sequência ao programa (SOFFNER, 2013, p. 24).

Fluxograma

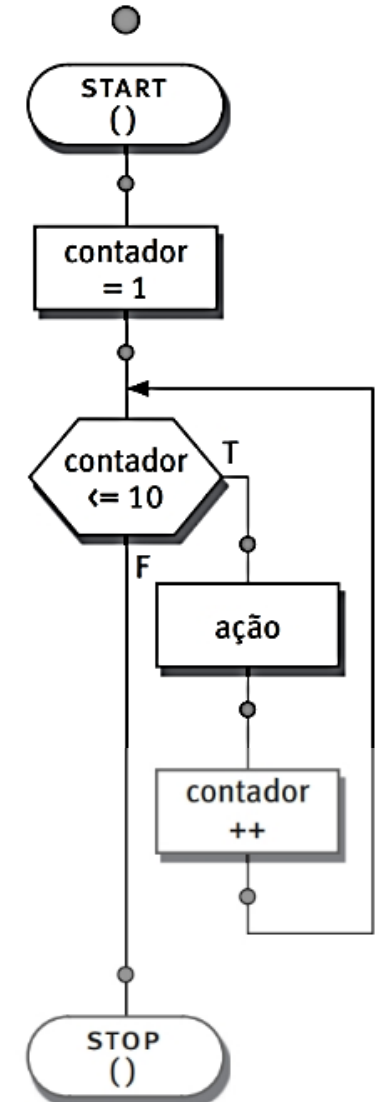
a) Com teste no início



b) Com teste no fim



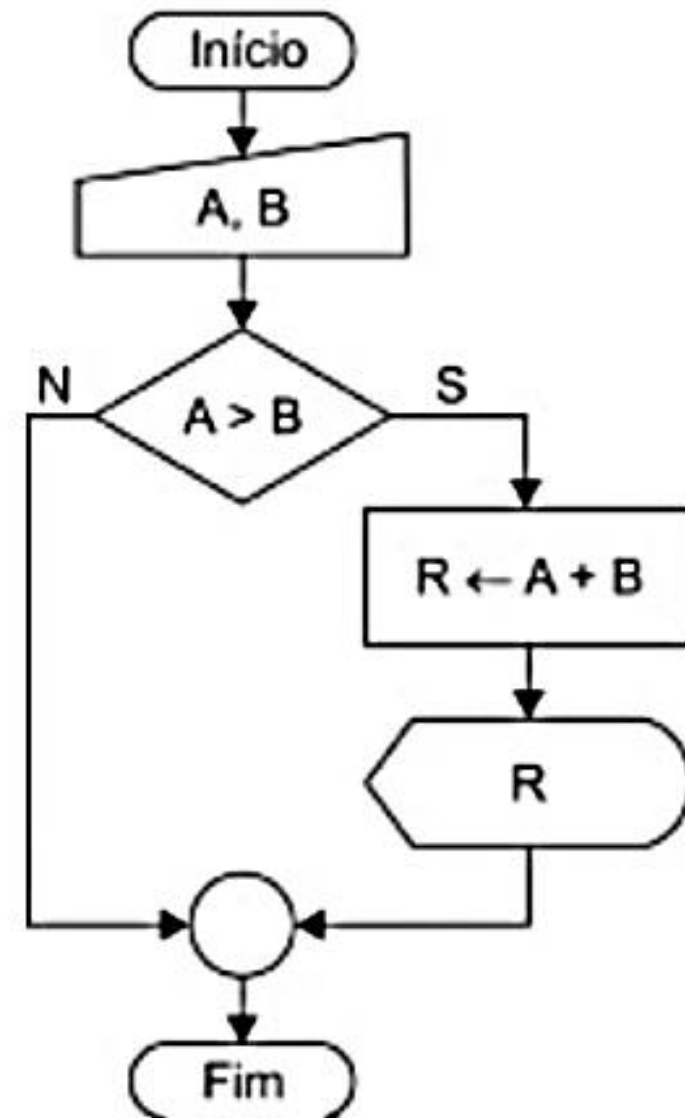
c) Com variável de controle



Estrutura de repetição
Fonte: Soffner (2013, p. 26).

Fluxograma

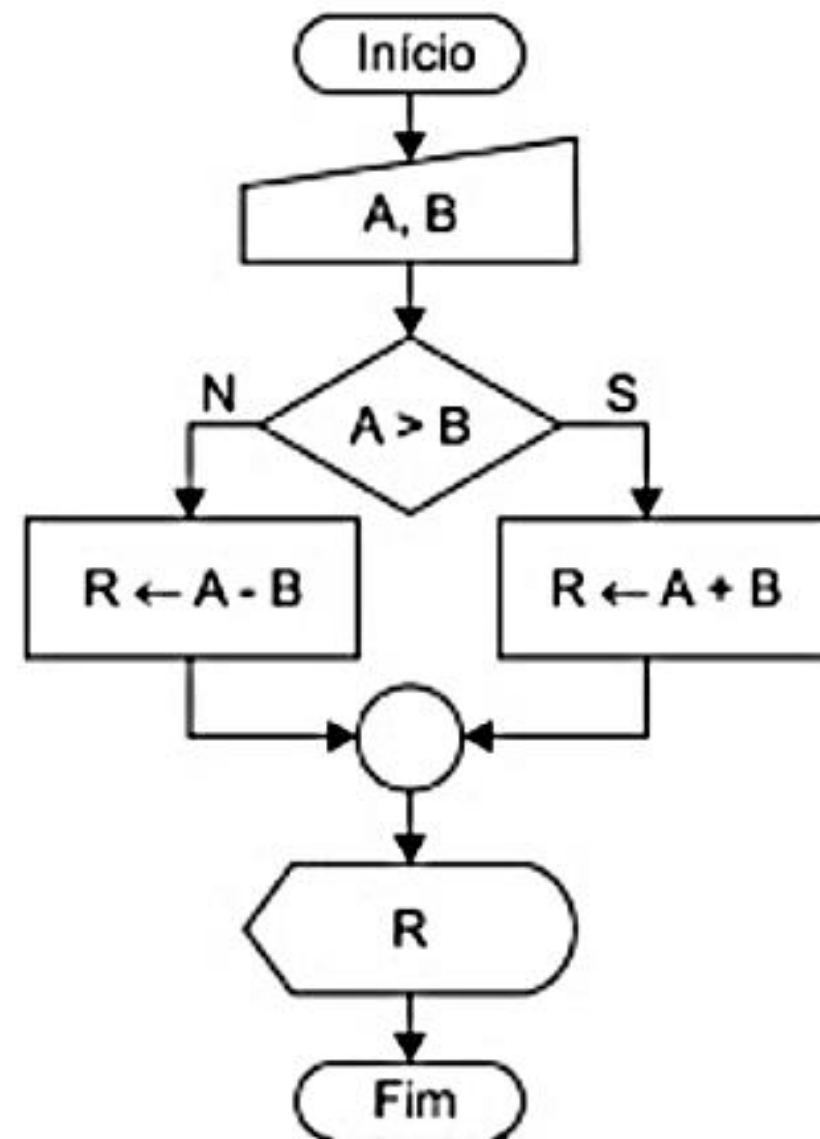
- A próxima figura apresenta a estrutura de operação computacional de decisão simples. Sendo definidas 3 variáveis (A, B e R), pede-se para realizar a soma das variáveis A e B e exibir o resultado (R) caso o valor atribuído para a variável A seja maior do que a variável B.



Estrutura de operação
computacional de decisão simples
Fonte: Manzano (2019, p. 39).

Fluxograma

- A próxima figura apresenta a estrutura de operação computacional de decisão composta. Sendo definidas 3 variáveis (A, B e R), se o valor de A for maior do que o valor de B ($A > B$), pede-se para realizar a soma das variáveis A e B e exibir o resultado (R). Se o valor de A for menor ou igual ao valor de B, pede-se para realizar a subtração das variáveis A e B e exibir o resultado (R).

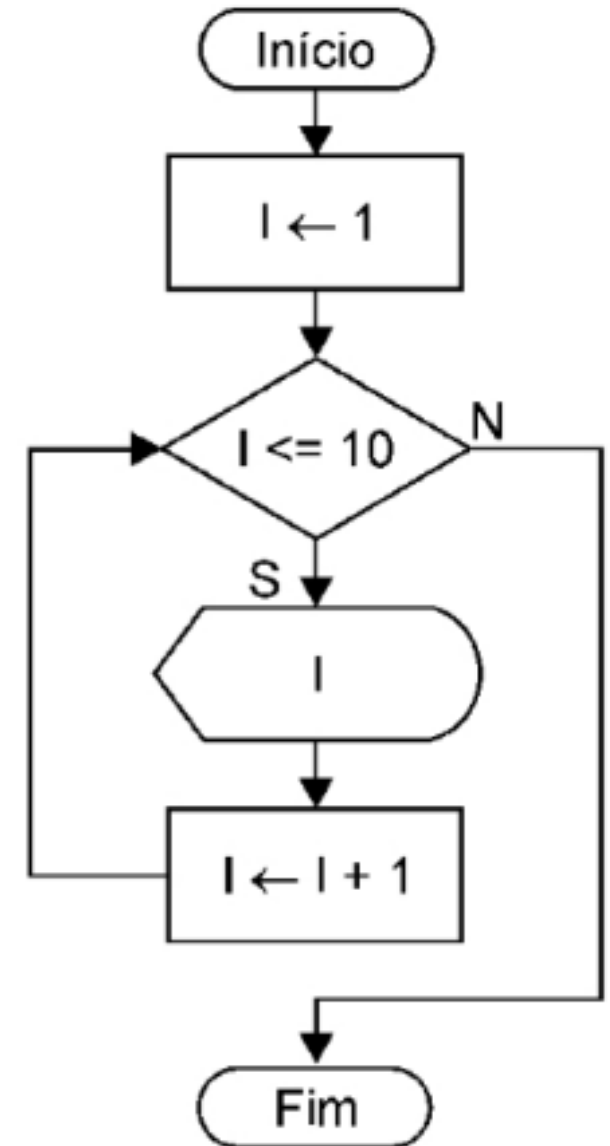


Estrutura de operação
computacional de decisão simples
Fonte: Manzano (2019, p. 39).

Fluxograma

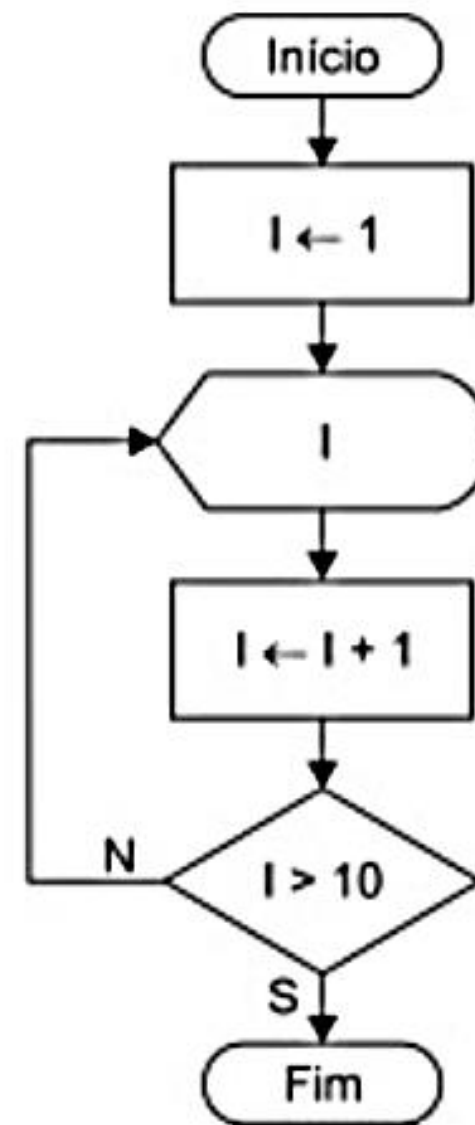
- A próxima figura apresenta a estrutura de operação computacional de laço de repetição condicional pré-teste. É definida a variável I e atribuído um valor inicial de 1 (um). A instrução será repetida até que a variável I seja menor ou igual a 10 (observe que o teste acontece no **início** do processamento). Em caso afirmativo, será apresentado o valor da variável I e incrementado 1 (um) no valor dessa mesma variável.

Estrutura de operação
computacional de decisão
simples
Fonte: Manzano (2019, p. 40).



Fluxograma

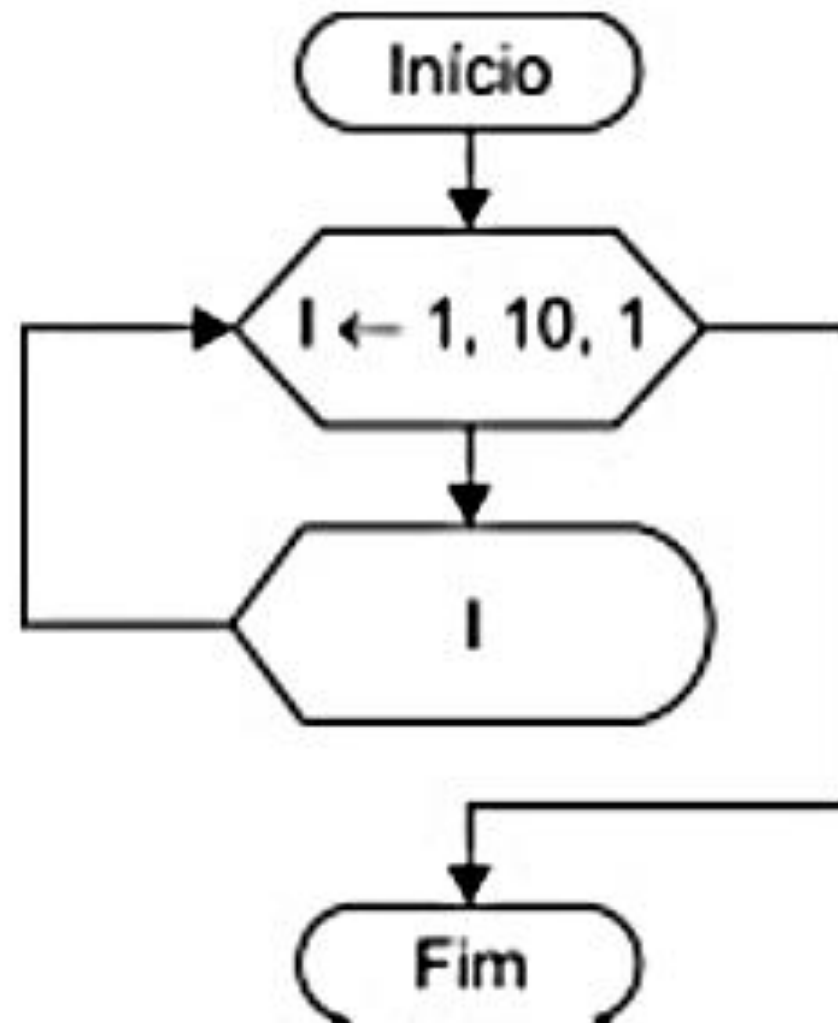
- A próxima figura apresenta a estrutura de operação computacional de laço de repetição condicional pós-teste. É definida a variável I e atribuído um valor inicial de 1 (um). A instrução será repetida até que a variável I seja menor ou igual a 10 (observe que o teste acontece no **final** do processamento). Em caso afirmativo, será apresentado o valor da variável I e incrementado 1 (um) no valor dessa mesma variável.



Estrutura de operação
computacional de decisão simples
Fonte: Manzano (2019, p. 40).

Fluxograma

- A próxima figura apresenta a estrutura de operação computacional de laço de repetição incondicional. O laço de repetição é composto pela declaração dos 3 argumentos necessários (inicialização, validação/verificação e incremento/decremento). No caso do exemplo será exibido o valor da variável *I* até que o contator seja menor ou igual a 10.



Estrutura de operação
computacional de decisão
simples

Fonte: Manzano (2019, p. 40).

Vantagens do fluxograma

- O fluxograma é uma excelente maneira de comunicar a lógica de um programa.
- O fluxograma é uma maneira fácil e eficiente para analisar os problemas.
- Durante o ciclo de desenvolvimento do programa, o fluxograma desempenha o papel de um *blueprint*, que facilita o processo de desenvolvimento do programa.
- Após o desenvolvimento bem-sucedido de um programa, ele precisa de manutenção contínua e oportuna durante todo o seu funcionamento (e vida útil). O fluxograma facilita o processo de alterações e manutenção do sistema.
 - É fácil converter o fluxograma em qualquer linguagem de programação.

Vantagens do fluxograma

- Ao usar ferramentas gráficas para representar a linha lógica de raciocínio a ser implementada em um computador eletrônico, como é o caso dos diagramas de blocos ou dos diagramas de quadros, torna-se necessário diferenciar quatro subcategorias entre esses diagramas (MANZANO, 2019, p. 41).
- O algoritmo e o fluxograma incluem os seguintes tipos de estruturas de controle:
- Sequence (sequência): na estrutura de sequência, as instruções são colocadas uma após a outra e a execução ocorre de cima para baixo.



Fonte:
<https://pixabay.com/photos/whiteboard-man-presentation-write-849811/>

Vantagens do fluxograma

- Branching (ramificação/seleção): no controle de ramificação, existe uma condição e de acordo com uma condição, uma decisão de verdadeiro ou falso é alcançada. No caso de true, um dos dois ramos é explorado; mas no caso de condição false, a outra alternativa é tomada. Geralmente, o “IF-THEN” é usado para representar o controle de ramificação.
- Loop (repetição): a estrutura de repetição permite que uma instrução seja executada repetidamente com base em certas condições de *loop*.

Fluxograma

Fonte:

<https://pixabay.com/photos/whiteboard-man-presentation-write-849813/>



Interatividade

Os símbolos de identificação gráfica representam sempre uma operação ou conjunto de operações similares, podendo ser identificados por um rótulo relacionado à própria ação do símbolo em uso, somente quando necessário.

Selecione a alternativa **incorreta**.

- a) O fluxograma é uma excelente maneira de comunicar a lógica de um programa.
- b) O fluxograma é uma maneira fácil e eficiente para analisar os problemas.
- c) É fácil converter o fluxograma em qualquer linguagem de programação.
- d) Durante o ciclo de desenvolvimento do programa, o fluxograma desempenha o papel de um *blueprint*, que facilita o processo de desenvolvimento do programa.
- e) O fluxograma não necessita de manutenção contínua, pois não deverá sofrer alterações ao longo do funcionamento do programa.

Resposta

Os símbolos de identificação gráfica representam sempre uma operação ou conjunto de operações similares, podendo ser identificados por um rótulo relacionado à própria ação do símbolo em uso, somente quando necessário.

Selecione a alternativa **incorreta**.

- a) O fluxograma é uma excelente maneira de comunicar a lógica de um programa.
- b) O fluxograma é uma maneira fácil e eficiente para analisar os problemas.
- c) É fácil converter o fluxograma em qualquer linguagem de programação.
- d) Durante o ciclo de desenvolvimento do programa, o fluxograma desempenha o papel de um *blueprint*, que facilita o processo de desenvolvimento do programa.
- e) O fluxograma não necessita de manutenção contínua, pois não deverá sofrer alterações ao longo do funcionamento do programa.

Técnicas básicas de programação: tipos de dados

- O computador digital moderno foi inventado e planejado como um dispositivo que deve facilitar e acelerar os cálculos complicados e demorados.
- Na maioria dos aplicativos, sua capacidade de armazenar e acessar grandes quantidades de informação desempenha o papel dominante e é considerado sua principal característica, e sua capacidade de calcular, ou seja, calcular, executar aritmética, em muitos casos se tornou quase irrelevante.
- Em todos esses casos, a grande quantidade de informações a serem processadas em algum sentido representa uma abstração de uma parte da realidade.

Fonte:
<https://pixabay.com/photos/laptop-code-programming-computer-2557586/>



Técnicas básicas de programação: tipos de dados

- As informações disponíveis para o computador consistem em um conjunto selecionado de dados sobre o problema real, ou seja, os dados representam uma abstração da realidade no sentido de que certas propriedades e características dos objetos reais são ignoradas porque são periféricas e irrelevantes para o problema em particular.
- Uma abstração é, portanto, também uma simplificação dos fatos. Podemos considerar um arquivo pessoal de um empregador como um exemplo. Todo funcionário é representado (de forma resumida) nesse arquivo por um conjunto de dados relevantes para o empregador ou para os seus procedimentos contábeis.
 - Esse conjunto pode incluir alguma identificação do funcionário, por exemplo, seu nome e salário. Mas isso provavelmente não incluirá dados irrelevantes, como cor, peso e altura do cabelo. Ao resolver um problema com ou sem um computador, é necessário escolher uma abstração da realidade, ou seja, definir um conjunto de dados que representa a situação real.

Técnicas básicas de programação: tipos de dados

- Essa escolha deve ser guiada pelo problema a ser resolvido. Em seguida, segue uma escolha de representação dessas informações. Essa escolha é guiada pela ferramenta que será utilizada para resolver o problema, isto é, pelas facilidades oferecidas pelo computador. Na maioria dos casos, essas duas etapas não são totalmente separadas.
- A escolha da representação dos dados geralmente é bastante difícil e não é determinada exclusivamente pelas instalações disponíveis.
- Sempre deve ser tomada à luz das operações que devem ser executadas a partir dos dados.
 - Um bom exemplo é a representação de números, que são abstrações de propriedades de objetos a serem caracterizados.

Técnicas básicas de programação: tipos de dados

- Os computadores usam uma representação interna baseada em dígitos binários (bits). Essa representação é inadequada para os seres humanos devido ao número geralmente grande de dígitos envolvidos, mas é mais adequado para circuitos eletrônicos, porque os dois valores 0 e 1 podem ser representados convenientemente e de forma confiável pela presença ou ausência de correntes elétricas, carga elétrica ou campos magnéticos.
- Nesse contexto, o significado das linguagens de programação se torna aparente. Uma linguagem de programação representa um computador abstrato capaz de interpretar os termos usados nesse idioma, que podem incorporar um certo nível de abstração dos objetos usados pela máquina real.

Técnicas básicas de programação: tipos de dados

- A importância de usar uma linguagem que ofereça um conjunto conveniente de abstrações básicas comuns à maioria dos problemas do processamento de dados está principalmente na área de confiabilidade dos programas resultantes.

Técnicas básicas de programação: tipos de dados

- É mais fácil projetar um programa baseado no raciocínio com noções familiares de números, conjuntos, sequências e repetições do que em bits e unidades de armazenamento. Obviamente, um computador real representa todos os dados, sejam números, conjuntos ou sequências, como uma grande massa de bits.
- Mas isso é irrelevante para o programador, desde que não seja necessário se preocupar com os detalhes de representação das abstrações escolhidas e desde que possamos utilizar a representação correspondente escolhida pelo computador (ou compilador) para o desenvolvimento dos fins declarados.

Fonte:
<https://pixabay.com/photos/laptop-code-programming-computer-2557576/>



Técnicas básicas de programação: tipos de dados

- Um tipo de dados é um conjunto de valores (os dados) e um conjunto de operações definidas nos dados.
- Uma implementação de um tipo de dados é uma expressão dos dados e operações em termos de uma programação específica como as linguagens Java, C, C++, C# etc. Um tipo de dados abstratos (ADT – *abstract data type*) é uma especificação de um tipo de dados, sem levar em consideração nenhuma linguagem de implementação ou programação específica.
- Finalmente, a realização de um ADT envolve duas partes: a interface, especificação ou documentação do ADT: qual é o objetivo de cada operação e qual é a sintaxe para usá-lo e a implementação do ADT: como cada operação é expressa usando as estruturas de dados e as declarações de uma linguagem de programação (University of Crete, 2020).

Técnicas básicas de programação: tipos de dados

De acordo com Yang (2017), um tipo de dados define uma coleção de valores de dados e um conjunto de operações predefinidas nesses valores. A partir desse princípio, é possível afirmar que:

- Programas de computador produzem resultados manipulando dados.
- Um descritor é a coleção dos atributos de uma variável.
- Em uma implementação, um descritor é uma coleção de células de memória que armazenam as variáveis e os atributos.
- Se os atributos forem estáticos, o descritor será necessário apenas no momento da compilação.
 - Esses descritores são criados pelo compilador, geralmente como parte da tabela de símbolos, e são usados durante a compilação.
 - Para atributos dinâmicos, parte ou todo o descritor deve ser mantido durante a execução.
 - Os descritores são usados para verificação de tipo e por operações de alocação e desalocação.

Técnicas básicas de programação: tipos de dados

- Wirth (1985) afirma que na matemática é costume classificar variáveis de acordo com certas características importantes.
- Claro que são feitas distinções entre variáveis reais, complexas e lógicas ou entre variáveis que representam os valores individuais, ou conjuntos de valores, ou conjuntos de conjuntos, ou entre funções, ou conjuntos de funções.
- Essa noção de classificação é igual se não mais importante no processamento de dados. A partir do princípio de que toda constante, variável, expressão ou função é de um certo tipo, esse tipo caracteriza o conjunto de valores aos quais uma constante pertence ou que pode ser assumida por uma variável ou expressão ou que pode ser gerada por uma função.

Técnicas básicas de programação: tipos de dados

- Em textos matemáticos, o tipo de uma variável é geralmente dedutível do tipo de letra sem consideração de contexto; isso não é viável em programas de computador.
- Geralmente, há um tipo de letra disponível no computador, portanto, a regra é amplamente aceita de que o tipo associado deve ser feito de maneira explícita em uma declaração da constante, variável ou função, e que essa declaração precede textualmente da aplicação dessa constante, variável ou função.
- É costume classificar variáveis de acordo com certas características importantes. Claro que devem ser feitas distinções entre variáveis reais, complexas e lógicas ou entre variáveis que representam os valores individuais ou conjuntos de valores ou conjuntos de conjuntos ou entre funções, conjuntos de funções e assim por diante. Essa noção de classificação é igualmente aplicada no processamento de dados.

Técnicas básicas de programação: tipos de dados

- Essa regra é particularmente sensata se considerarmos o fato de que um compilador deve fazer uma escolha de representação do objeto.
- Evidentemente, a quantidade de armazenamento alocado para uma variável terá que ser escolhida de acordo com o tamanho do intervalo de valores que a variável pode assumir. Se essa informação é conhecida por um compilador, a chamada alocação dinâmica de armazenamento pode ser evitada.
- Muitas vezes, essa é a chave para uma realização eficiente de um algoritmo.
 - Como consequência, um compilador pode usar essas informações em tipos para verificar a legalidade de várias construções.

Técnicas básicas de programação: tipos de dados

- Por exemplo, a atribuição incorreta de um valor booleano (lógico) a uma variável aritmética pode ser detectada sem executar o programa.
- Esse tipo de redundância no texto do programa é extremamente útil como uma ajuda no desenvolvimento de programas e deve ser considerada como uma vantagem.
- Evidentemente, os dados serão finalmente representados por um grande número de dígitos binários, independentemente de o programa ter sido inicialmente ou não concebido em uma linguagem de alto nível usando o conceito de tipo.

Fonte:
<https://pixabay.com/photos/business-technology-notebook-laptop-2717063/>



Técnicas básicas de programação: tipos de dados

- Os dados são elementos do mundo exterior que representam, dentro de um computador digital, as informações manipuladas pelos seres humanos.
- Os dados a serem utilizados devem primeiramente ser abstraídos para serem, então, processados.
- Eles podem ser classificados em três tipos primitivos ou tipos básicos: numéricos (representados por valores numéricos inteiros ou reais), caracteres (representados por valores alfabéticos ou alfanuméricos) e lógicos (valores dos tipos falso e verdadeiro).

(MANZANO, 2019, p. 51)

Interatividade

Um tipo de dados define uma coleção de valores de dados e um conjunto de operações predefinidas nesses valores.

Em relação a tipo de dados, selecione a alternativa **incorreta**:

- a) Programas de computador produzem resultados manipulando dados.
- b) Um descritor é a coleção dos atributos de uma variável.
- c) Em uma implementação, um descritor é uma coleção de células de memória que armazenam as variáveis e os atributos.
- d) Se os atributos forem estáticos, o descritor não será necessário no momento da compilação.
- e) Para atributos dinâmicos, parte ou todo o descritor deve ser mantido durante a execução.

Resposta

Um tipo de dados define uma coleção de valores de dados e um conjunto de operações predefinidas nesses valores.

Em relação a tipo de dados, selecione a alternativa **incorreta**:

- a) Programas de computador produzem resultados manipulando dados.
- b) Um descritor é a coleção dos atributos de uma variável.
- c) Em uma implementação, um descritor é uma coleção de células de memória que armazenam as variáveis e os atributos.
- d) Se os atributos forem estáticos, o descritor não será necessário no momento da compilação.
- e) Para atributos dinâmicos, parte ou todo o descritor deve ser mantido durante a execução.

Tipos primitivos (*primitive types*)

- Wirth (1985) apresenta que os tipos primitivos padrão são aqueles que estão disponíveis na maioria dos computadores como recursos internos.
- Eles incluem os números inteiros, os valores lógicos (verdadeiro ou falso) e um conjunto de caracteres (texto). Em muitos computadores, números fracionários também são incorporados, juntamente com as operações aritméticas padrões dos números reais.
- Nós classificamos esses identificadores a partir da seguinte nomenclatura: INTEGER (números inteiros), REAL (números fracionários), BOOLEAN (valores lógicos: verdadeiro e falso) e CHAR (caractere).

Fonte:
<https://pixabay.com/photos/code-programming-hacking-html-web-820275/>



Tipos primitivos (*primitive types*)

- Inteiro: os dados numéricos positivos e negativos pertencem ao conjunto de números inteiros, excluindo dessa categoria qualquer valor numérico fracionário (que pertence ao conjunto de números reais), por exemplo, os valores 35, 0, 234, -56, -9, entre outros. Nessa obra, a representação do dado inteiro é feita em português estruturado com o comando inteiro. O tipo de dado inteiro é utilizado em operações de processamento matemático.
- Real: são reais os dados numéricos positivos e negativos que pertencem ao conjunto de números reais, incluindo nessa categoria todos os valores fracionários e inteiros, por exemplo, os valores 35, 0, -56, -9, -45.999, 4.5, entre outros. Nessa obra, o tipo de dado real será representado em português estruturado pelo comando real. O tipo de dado real é utilizado em operações de processamento matemático.

Tipos primitivos (*primitive types*)

- Caractere/cadeia: são caracteres delimitados pelos símbolos aspas (“ ”). Eles são representados por letras (de A até Z), números (de 0 até 9), símbolos (por exemplo, todos os símbolos imprimíveis existentes em um teclado) ou palavras contendo esses símbolos. O tipo de dado caractere é conhecido também como alfanumérico, *string* (em inglês, cordão, colar), literal ou cadeia, por exemplo, os valores “PROGRAMAÇÃO”, “Rua Alfa, 52 – Apto. 1”, “Fone: (0xx99) 5544-3322”, “CEP: 11222-333”, “ ” (espaço em branco), “7”, “-90”, “45.989”, entre outros.
- Lógico: são lógicos os dados com valores binários do tipo sim e não, verdadeiro e falso, 1 e 0, entre outros, em que apenas um dos valores pode ser escolhido.
 - Para que um dado do tipo lógico seja devidamente usado, é necessário estabelecer a forma de sua representação, que, nesse caso, será feita com os valores .F. (para representar falso, pode-se também fazer referência como .FALSO.) e .V. (para representar verdadeiro, pode-se também fazer referência como .VERDADEIRO.) (MANZANO, 2019, p. 51-52).

Tipos primitivos (*primitive types*)

- Os **tipos primitivos** são aqueles que estão disponíveis na maioria dos computadores como recursos internos.
- Eles incluem os **números inteiros**, os **valores lógicos** da verdade e um **conjunto de caracteres** imprimíveis.
- Em muitos computadores, os **números fracionários** também são incorporados, juntamente com as operações aritméticas padrões. Podemos classificar os tipos primitivos como sendo os elementos: **INTEGER, REAL, BOOLEAN, CHAR.**

Fonte:
<https://pixabay.com/photos/code-programming-hacking-html-web-820275/>



Tipos primitivos (*primitive types*)

- Yang (2017) apresenta que a linguagem de programação C e os tipos de dados se referem a um extenso sistema usado para declarar as variáveis ou funções de diferentes tipos. O tipo de uma variável determina quanto espaço ela ocupa no armazenamento e como o padrão de bits armazenado é interpretado.
- Os tipos em C podem ser classificados da seguinte forma:

TIPOS E DESCRIÇÃO	
Tipos básicos (<i>basic types</i>)	Eles são tipos aritméticos e consistem em dois tipos: tipos inteiros (<i>integer</i>) e tipos de ponto flutuante/números reais (<i>floatingpoint</i>).
Tipos enumerados (<i>enumerated types</i>)	São novamente tipos aritméticos e são usados para definir as variáveis que só podem ser atribuídos certos valores inteiros discretos (<i>discrete integer</i>) ao longo do programa.
Tipo nulo (<i>the type void</i>)	O especificador de tipo void indica que nenhum valor está disponível.
Tipos derivados (<i>derived types</i>)	Eles incluem os seguintes tipos: ponteiro (<i>Pointer</i>), matriz (<i>Array</i>), estrutura (<i>Structure types</i>), Union e função (<i>Function</i>).

Tipos primitivos (*primitive types*)

- Os **tipos primitivos** são aqueles que estão disponíveis na maioria dos computadores como recursos internos.
- Eles incluem os **números inteiros**, os **valores lógicos** da verdade e um **conjunto de caracteres** imprimíveis.
- Em muitos computadores, os **números fracionários** também são incorporados, juntamente com as operações aritméticas padrões. Podemos classificar os tipos primitivos como sendo os elementos: **INTEGER, REAL, BOOLEAN, CHAR.**

Fonte:
<https://pixabay.com/photos/programming-html-css-javascript-1873854/>

```
37 }
38
39 $sort_order = array();
40
41 foreach ($quotes as $key => $value) {
42     $sort_order[$key] = $value['sort_order'];
43 }
44
45 array_multisort($sort_order, SORT_ASC, $quotes);
46
47 $this->session->data['lpa']['shipping_methods'] = $quotes;
48 $this->session->data['lpa']['address'] = $address;
49
50 if (empty($quotes)) {
51     $json['error'] = $this->language->get('
52     error_no_shipping_methods');
53 }
54 else {
55     $json['quotes'] = $quotes;
56 }
57
58 if (isset($this->session->data['lpa']['shipping_method'])) {
59     empty($this->session->data['lpa']['shipping_method']) ||
60     $json['selected'] = $this->session->data['lpa']['
61     shipping_method']['code'];
62 }
63 else {
64     $json['selected'] = '';
65 }
66 }
67
68 $json['error'] = $this->language->get('error_shipping_methods');
69 }
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```


Tipos primitivos (*primitive types*)

- A maioria das linguagens declara os tipos inteiro, real, caractere, cadeia de caracteres e lógico. A Linguagem C não definiu como tipos originais o lógico e a cadeia de caracteres (que chamamos de *string* – exemplo: “UNIP”); assim, devemos utilizar funções predefinidas para manipular esse tipo de dado, em C. Em Linguagem C, temos os seguintes tipos de dados primitivos:
 - Inteiro: palavra reservada *int* (exemplos: 3, 6, 9, 27)
 - Real: palavra reservada *float* (exemplos: 3.1416, 8.8)
 - Caractere: palavra reservada *char* (exemplos: ‘a’, ‘8’)
 - Real de precisão dupla: palavra reservada *double*
 - Tipo sem um valor inicial: palavra reservada *void*

(SOFFNER, 2013, p. 32)

Operadores

- Um operador é um símbolo que ajuda o usuário a comandar o computador para realizar um determinado cálculo matemático ou desenvolver alguma manipulação lógica. Os operadores são usados na linguagem de programação para operar os dados e as variáveis.

Os operadores podem ser classificados como:

- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos
- Operadores de incremento e decréscimo

Operadores aritméticos

- Uma operação muito comum em programação de computadores é usar expressões aritméticas para o estabelecimento de processamentos matemáticos. As expressões aritméticas são realizadas a partir do relacionamento existente entre variáveis e constantes numéricas com a utilização dos operadores aritméticos (MANZANO, 2019, p. 54).
- Você pode usar um operador aritmético com um ou dois argumentos para adicionar, subtrair, multiplicar e dividir os valores numéricos.

Os operadores aritméticos podem ser definidos nos seguintes itens:

- + (Adição)
- (Subtração)
- * (Multiplicação)
- / (Divisão)
- % (MOD/Divisão do módulo)

Operadores aritméticos

Exemplos:

$a - b$

$a + b$

$a * b$

a / b

$a \% b$

Fonte:
<https://pixabay.com/photos/accountant-accounting-adviser-1238598/>



- A divisão do módulo produz o restante de uma divisão inteira.
- O operador de divisão de módulo não pode ser usado em dados de ponto flutuante.

Aritmética inteira (números inteiros)

- Quando ambos os operandos em uma única expressão aritmética são inteiros, a expressão é chamada de expressão inteira e a operação é chamada aritmética inteira. Durante a divisão do módulo, o sinal do resultado é sempre o sinal do primeiro operando.

Seguem alguns exemplos da operação utilizando o operador % (MOD):

$$-14 \% 3 = -2$$

$$-14 \% -3 = -2$$

$$14 \% -3 = 2$$

Aritmética real (números reais)

Uma operação aritmética envolvendo apenas operandos reais é chamada aritmética real. Se x e y são números reais, teremos:

$$x = 6,0 / 7,0 = 0,857143$$

$$y = 1,0 / 3,0 = 0,333333$$

- De acordo com Manzano (2019), a próxima imagem apresenta os operadores segundo a ordem de prioridade matemática em que as operações são realizadas.
 - Caso seja necessário alterar o nível de prioridade de um referido cálculo, ele deve ser colocado entre parênteses.

Aritmética real (números reais)

Operador	Operação	Descrição	Tipo	Prioridade	Resultado
+	"+" ou "n"	Manutenção de sinal	Unário	-	Positivo
-	-n	Inversão de sinal	Unário	-	Negativo
←	$x \leftarrow n$	Atribuição do valor "n" a "x"	Binário	-	Positivo ou Negativo
↑	$x \uparrow n$	Exponenciação de x^n	Binário	1	Inteiro ou Real'
$\uparrow (y / n)$	$x \uparrow (y / n)$	Radiciação de $\sqrt[n]{x^y}$	Binário	1	Real
$\uparrow (1 / n)$	$x \uparrow (1 / n)$	Radiciação de $\sqrt[n]{x}$	Binário	1	Real
/	x / n	Divisão de "x" por "n"	Binário	2	Real (quociente relativo)
*	$x * n$	Multiplicação de "x" por "n"	Binário	2	Inteiro ou Real
+	$x + n$	Adição de "x" com "n"	Binário	3	Inteiro ou Real
-	$x - n$	Subtração de "n" de "x"	Binário	3	Inteiro ou Real
div	$x \text{ div } n$	Divisão de "x" por "n"	Binário	4	Inteiro (quociente absoluto)

Estrutura de operação computacional de decisão simples

Fonte: Manzano (2019, p. 54).

Operadores relacionais

Uma operação aritmética envolvendo apenas operandos reais é chamada aritmética real. Se x e y são números reais, teremos:

$$x = 6,0 / 7,0 = 0,857143$$

$$y = 1,0 / 3,0 = 0,333333$$

- De acordo com Manzano (2019), a próxima imagem apresenta os operadores segundo a ordem de prioridade matemática em que as operações são realizadas.
 - Caso seja necessário alterar o nível de prioridade de um referido cálculo, ele deve ser colocado entre parênteses.

Operadores relacionais

- As comparações podem ser feitas com a ajuda de operadores relacionais. A expressão contendo um operador relacional é denominada como uma expressão relacional. Os valores de um relacionamento dessa expressão são os valores um ou zero.

Segue a lista dos operadores relacionais:

- a. $<$ (menor que)
- b. $<=$ (menor ou igual a)
- c. $>$ (maior que)
- d. $>=$ (maior ou igual a)
- e. $==$ (igual a)
- f. $!=$ (não é igual a)

Operadores relacionais

- Os operadores lógicos são fundamentais para auxiliar no processo de validação e verificação de nossos programas.

A princípio, trabalharemos com os três operadores lógicos principais, que são: `&&` (e / AND), `||` (ou / OR) e o `!` (negação / NOT). Por exemplo:

- a. se (idade > 55 && sal < 1000)
- b. se (número < 0 || número > 100)
- c. se ! (número < 0)

Interatividade

Um operador é um símbolo que ajuda o usuário a comandar o computador para realizar um determinado cálculo matemático ou desenvolver alguma manipulação lógica.

Selecione a alternativa que apresenta o motivo para utilizar os operadores na linguagem de programação:

- a) Os operadores são usados na linguagem de programação para operar os dados e as variáveis.
- b) Os operadores são usados na linguagem de programação para manipular os dados e as variáveis.
- c) Os operadores são usados na linguagem de programação para processar os dados e as variáveis.
- d) Os operadores são usados na linguagem de programação para otimizar os dados e as variáveis.
- e) Os operadores são usados na linguagem de programação para unificar os dados e as variáveis.

Resposta

Um operador é um símbolo que ajuda o usuário a comandar o computador para realizar um determinado cálculo matemático ou desenvolver alguma manipulação lógica.

Selecione a alternativa que apresenta o motivo para utilizar os operadores na linguagem de programação:

- a) Os operadores são usados na linguagem de programação para operar os dados e as variáveis.
- b) Os operadores são usados na linguagem de programação para manipular os dados e as variáveis.
- c) Os operadores são usados na linguagem de programação para processar os dados e as variáveis.
- d) Os operadores são usados na linguagem de programação para otimizar os dados e as variáveis.
- e) Os operadores são usados na linguagem de programação para unificar os dados e as variáveis.

Expressões

- Informalmente, um algoritmo é uma sequência finita de instruções inequívocas para executar uma tarefa específica.
- Um algoritmo tem um nome e a princípio é necessário iniciar com uma entrada especificada com precisão e termina com uma saída especificada com precisão.
- De acordo com Soicher & Vivaldi (2004, p. 8), a entrada e a saída são sequências finitas de objetos matemáticos. Diz-se que um algoritmo está correto se, ao receber uma entrada, conforme descrito nas especificações de entrada:
 - (i) o algoritmo termina em um tempo finito;
 - (ii) na terminação, o algoritmo retorna a saída conforme descrito nas especificações de saída. Essas expressões aritméticas e algébricas são formadas pela montagem de modos familiares de números e operadores aritméticos.

Expressões

Segue abaixo um modelo dessas expressões aritméticas e matemáticas:

$$1 + \frac{1}{2 + \frac{1}{3 + \frac{1}{4 + \frac{1}{5}}}}$$

$$\frac{(x - y)(x + y)(x^2 + y^2)(x^4 + y^4)}{x^8 - y^8}$$

$$(x - 1, x + 1, x^2 + x + 1, x^2 + 1)$$

$$\left\{ 0, 1, \frac{1}{2}, \frac{1}{3}, \frac{2}{3}, \frac{1}{4}, \frac{3}{4} \right\}$$

Representação das expressões
algébricas e aritméticas
Fonte: Soicher & Vivaldi (2004, p. 8).

Expressões

- Expressões em programação são como fórmulas em matemática: ambos usam valores para calcular um resultado.
- Mas diferente da matemática, as expressões que podemos criar pelas linguagens de programação podem calcular resultados a partir de uma ampla variedade de tipos (por exemplo, booleano e string), não apenas resultados matemáticos. As expressões são como as moléculas: elas são construídas a partir de átomos (literais e nomes que possuem valores) e ligações químicas que mantêm os átomos no lugar (operadores e chamadas de função).
 - A programação matemática prática em larga escala envolve mais do que apenas a minimização ou maximização de uma função.

Expressões

- Antes que qualquer algoritmo de otimização possa ser aplicado, é necessário algum esforço para formular o modelo subjacente e gerar as estruturas de dados computacionais necessárias.
- Se os algoritmos podem lidar com problemas de otimização como as pessoas, então, a formulação e as fases de geração e modelagem podem ser relativamente fáceis.
- Na realidade, porém, existem algumas diferenças entre a forma pela qual os matemáticos e os programadores entendem um problema e a maneira como são implementados os algoritmos e a sua codificação.

Expressões

- A princípio, o trabalho é dividido entre o humano e o computador.
- Primeiro, uma pessoa que entende a forma que o analista e modelador escreve um programa de computador cuja saída representará as estruturas de dados necessárias.
- Em seguida, o computador deve compilar e executar o programa para criar a forma do algoritmo.
 - Esse arranjo geralmente é caro e propenso a erros; o programa deve ser depurado por um modelador humano, mesmo que a sua saída – a forma do algoritmo – não seja para as pessoas lerem.

Expressões

- Na programação linear, a maior parte do algoritmo e a sua forma de representação da matriz são formadas por linhas e colunas que são numeradas em centenas ou milhares e cujos elementos que são diferentes de zero aparecem em padrões complexos.
- Um programa de computador que produz uma representação compacta dos coeficientes é chamado de gerador de matriz. Várias linguagens de programação foram projetadas especificamente para escrever justamente os geradores de matriz (CREEGAN, 1985) e idiomas padrão como o Fortran também são frequentemente usados (BEALE, 1970).

Expressões

- Uma operação muito comum em programação de computadores é usar expressões aritméticas para o estabelecimento de processamentos matemáticos.
- As expressões aritméticas são realizadas a partir do relacionamento existente entre variáveis e constantes numéricas com a utilização dos operadores aritméticos.
- Como exemplo de uma expressão aritmética, considere a fórmula de cálculo de área de circunferência: $\text{ÁREA} = \pi \cdot \text{RAIO}^2$, em que estão presentes as variáveis ÁREA, RAIO, a constante π ($\pi = 3.14159265$), os operadores aritméticos de multiplicação e exponenciação quando se eleva o valor da variável RAIO ao quadrado.

(MANZANO, 2019, p. 54)

Variáveis

- Variáveis são endereços de memória de trabalho que guardam, temporariamente, um valor utilizado pelo programa.
- Toda variável deve ter atributos do tipo nome, tamanho (que vem do tipo escolhido) e valor. As variáveis podem ter o chamado escopo local, quando são vistas apenas dentro da função em que foram chamadas e perdem seu valor quando o processamento sai de tal função, ou podem ser declaradas *static*, quando guardarão seu valor entre as diversas chamadas dessa função. Podem, ainda, ser de escopo global, quando têm validade em qualquer ponto do programa (SOFFNER, 2013, p. 33).

Variáveis

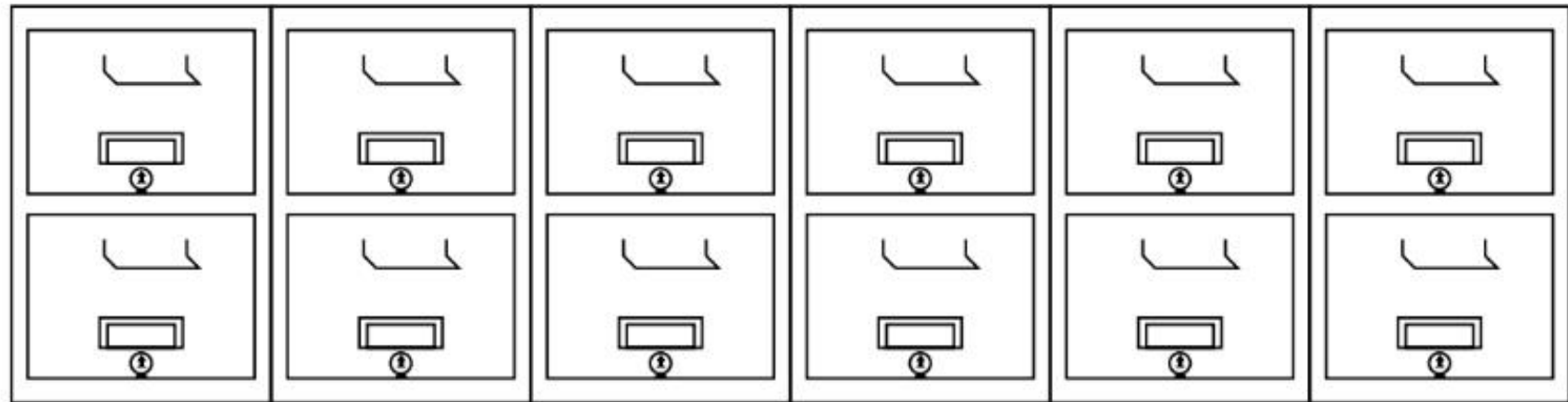
- De acordo com Manzano (2019), podemos conceituar o princípio de variável a partir da ilustração (*slide* 63), pois é possível observar que a composição da memória principal do computador é atrelada a partir da junção de muitas gavetas, e cada uma delas pode armazenar apenas um valor por vez, seja um dado **inteiro**, **real**, **lógico** ou **caractere**.
- Por ser um arquivo com várias gavetas é necessário que cada uma das gavetas seja identificada com um nome.
- Variável é tudo que está sujeito a variações, que é incerto, instável ou inconstante.
 - Quando se fala de computadores, é preciso ter em mente que o volume de dados a ser tratado é grande e diversificado.

Variáveis

- Dessa forma, os dados a serem processados são bastante variáveis. Todo dado a ser armazenado na memória de um computador deve ser previamente identificado segundo seu tipo, ou seja, primeiramente é necessário saber o tipo do dado para depois fazer seu armazenamento adequado. Armazenado o dado desejado, ele pode ser utilizado e processado a qualquer momento (MANZANO, 2019, p. 52).
- Uma variável é um nome simbólico atribuído a um item de dados pelo programador. Em qualquer momento, uma variável representará um dado específico, chamado de valor de uma variável, que pode mudar de tempos em tempos durante um processo de computação.
 - O valor de uma variável pode mudar muitas vezes durante a execução de um programa. Uma variável geralmente deve receber um nome atrelado com o conteúdo que será armazenado e processado pelo processo.

Variáveis

Imagine a memória de um computador como um grande arquivo, com várias gavetas, nas quais é possível guardar apenas um valor por vez, e, como em um arquivo, essas gavetas devem estar identificadas por uma etiqueta com um nome.



Representação gráfica da
memória de um computador
com variáveis
Fonte: Manzano (2019, p. 51).

Variáveis

- O nome de uma variável é utilizado para sua identificação e representação em um programa de computador. É necessário estabelecer e seguir algumas regras para uso de variáveis:
- Os nomes de identificação de uma variável podem utilizar um ou mais caracteres, limitando-se a restrições da própria linguagem formal de programação em uso. No caso do português estruturado, essa restrição não existe.
- O primeiro caractere de identificação do nome de uma variável não pode ser, em nenhuma hipótese, numérico ou um símbolo gráfico (cifrão, tralha, cachimbo, vírgula, ponto e vírgula, traço, parênteses, chaves, colchetes, entre outros). O primeiro caractere deve ser sempre alfabético.
 - Os demais caracteres do nome de uma variável podem ser alfanuméricos (números ou letras).

Variáveis

- Na definição de um nome composto de variável não pode haver espaços em branco. Caso deseje separar nomes compostos, deve-se utilizar o caractere de separação “_” *underline*.
- Jamais uma variável pode ser definida com o mesmo nome de uma palavra que represente um dos comandos ou instruções de uma linguagem de programação de computadores.
- Não pode ser utilizado como nome de variável algum rótulo que já tenha sido usado para identificar o nome de um programa ou mesmo de outra variável. Um nome se torna exclusivo no programa em que foi definido.

(MANZANO, 2019, p. 52.)

Variáveis

A operação de atribuição é usada para atribuir um nome a um valor. Assim, é usado sempre que for necessário acompanhar um valor necessário de forma posterior.

Alguns usos típicos incluem:

1. inicializar uma variável ($\text{contagem} = 0$)
2. aumentar/diminuir um contador ($\text{contagem} = \text{contagem} + 1$)
3. acumular valores ($\text{soma} = \text{soma} + \text{item}$)
4. capturar o resultado de uma expressão matemática
($y = 3 * x + 4$)
5. trocar dois valores ($t = x; x = y; y = t$)

Variáveis

- O operador de atribuição não é comutado, ou seja, $x = y$ não é o mesmo que $y = x$. As variáveis usadas na expressão devem ser definidas (têm valores).
- O tipo da expressão deve ser compatível com o tipo da variável.
- A ordem na qual as tarefas são executadas é importante, por exemplo, se a primeira e a segunda atribuição na sequência de troca foram trocadas, x e y terminariam atribuindo o mesmo valor.

Definição de variáveis

Fonte:

<https://pixabay.com/photos/coding-computer-hacker-hacking-1841550/>



Interatividade

Variáveis são endereços de memória de trabalho que guardam, temporariamente, um valor utilizado pelo programa.

Selecione a alternativa que contém os tipos primitivos:

- a) Inteiro, real, lógico ou caractere.
- b) Registro, vetor, inteiro ou matrizes.
- c) Inteiro, matrizes, *structs* ou registro.
- d) Vetor, lógico, lógico ou caractere.
- e) Matrizes, *structs*, registro ou vetor.

Resposta

Variáveis são endereços de memória de trabalho que guardam, temporariamente, um valor utilizado pelo programa.

Selecione a alternativa que contém os tipos primitivos:

- a) **Inteiro, real, lógico ou caractere.**
- b) Registro, vetor, inteiro ou matrizes.
- c) Inteiro, matrizes, *structs* ou registro.
- d) Vetor, lógico, lógico ou caractere.
- e) Matrizes, *structs*, registro ou vetor.

Referências

- ART, Susan; ELLISON, Robert; FEILER, Peter; EDITED, A.; FRITZSON, Peter. *Overview of Software Development Environments*. 1992. Disponível em: <https://www.ics.uci.edu/~andre/ics228s2006/dartellisonfeilerhabermann.pdf>. Acesso em: 25 abr. 2020.
- BEALE, E. M. L. Matrix Generators and Output Analyzers. *In*. KUHN, H. W. (ed.), *Proceedings of the Princeton Symposium on Mathematical Programming*. Princeton, NJ: Princeton University Press, 1970, p. 25-36.
- BROOKSHEAR, J. G. *Computer Science: An Overview*. Boston, Mass.: Pearson, 2009.
 - CHARNTAWEEKHUN, Kanis & WANGSIRIPITAK, Somkiat. (2006). Visual Programming using Flowchart. Disponível em: 10.1109/ISCIT.2006.339940. Acesso em: 25 abr. 2020.
 - CREEGAN, J. B. *DATAFORM*, a Model Management System. Ketron, Inc., Arlington, VA, 1985.

Referências

- DIFFERENCE BETWEEN SOURCE CODE AND OBJECT CODE. 24/01/2018. Disponível em: <https://www.differencebetween.com/wp-content/uploads/2018/01/Difference-Between-Source-Code-and-Object-Code.pdf>. Acesso em: 25 abr. 2020.
- MANZANO, José Augusto N. G. *Algoritmos: lógica para desenvolvimento de programação de computadores*. 29. ed. São Paulo: Érica, 2019.
- SLONNEGER, Kenneth; KURTZ, Barry L. *Formal syntax and semantics of programming languages: a laboratory*. 1995. Disponível em: <https://www.mobt3ath.com/uplode/book/book-26246.pdf>. Acesso em: 25 abr. 2020.
- SOFFNER, R. *Algoritmos e Programação em Linguagem C*. Saraiva, 2013.
 - SOICHER, Leonard; VIVALDI, Franco. *Algorithmic Mathematics*. University of London, 2004. Disponível em: <http://www.maths.qmul.ac.uk/~lsoicher/ambook.pdf>. Acesso em: 28 abr. 2020.

Referências

- STALLINGS, W. *Computer Organization and Architecture, Designing for Performance*. Boston, Mass.: Pearson, 2010.
- UNIVERSITY OF CRETE. 2020. *Introduction to Data Types and Structures*. Disponível em: <https://www.csd.uoc.gr/~hy252/html/References/Introduction%20to%20Data%20Types%20and%20Structures.pdf>. Acesso em: 28 abr. 2020.
- WALIA, Ravi K. *Algorithm & Flowchart Manual For Students*. University of Horticulture & Forestry – Índia, 2020. Disponível em: <http://www.yspuniversity.ac.in/cic/algorithm-manual.pdf>. Acesso em: 26 abr. 2020.
 - WIRTH, N. *Algorithms and Data Structures*. 1985. Disponível em: <https://inf.ethz.ch/personal/wirth/AD.pdf>. Acesso em: 28 abr. 2020.
 - YANG, Kuo-pao. *Chapter 6 - Data Types*. 2017. Disponível em: <https://www2.southeastern.edu/Academics/Faculty/kyang/2017/Fall/CMPS401/ClassNotes/CMPS401ClassNotesChap06.pdf>. Acesso em: 27 abr. 2020.

ATÉ A PRÓXIMA!