MINISTÉRIO DA AERONÁUTICA
DEPARTAMENTO DE PESQUISAS E DESENVOLVIMENTO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA AEROESPACIAL

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Praça Mal. Eduardo Gomes, 50 - V. Acácias
12228-900 - São José dos Campos - SP
Brasil
Tel. : (012) 340 - 3304 / Fax : (012) 341-7068

MINISTÉRIO DA AERONÁUTICA
DEPARTAMENTO DE PESQUISAS E DESENVOLVIMENTO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA AEROESPACIAL

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Guia para Leitura Correta do Livro Introdução à Ciência da Computação

Nei Yoshihiro Soma, Prof. Ph.D.
Fábio Carneiro Mokarzel, Prof. Dr.

ITA-IEC/MT- 003/2013

Praça Mal. Eduardo Gomes, 50 - V. Acácias
12228-900 - São José dos Campos - SP
Brasil
Tel. : (012) 340 - 3304 / Fax : (012) 341-70

Guia para Leitura Correta do Livro Introdução à Ciência da Computação

Sumário

Resumo

Este documento apresenta instruções para a leitura correta do livro “Introdução à Ciência da Computação” da autoria dos professores Nei Yoshihiro Soma e Fábio Carneiro Mokarzel. Apesar de muito zelo e técnica empregados na edição desse livro, ocorreram erros de digitação e de impressão.

Com o intuito de orientar os leitores do livro a utilizar corretamente os conhecimentos ali transmitidos, os autores decidiram publicar este guia indicando os pontos a serem corrigidos e fornecendo os textos substitutivos.

O documento está dividido em tantas seções principais quantos são os capítulos do livro, incluindo a parte inicial contendo o Prefácio e o Sumário e a Bibliografia.

Abstract

This document presents instructions for the correct reading of the book “Introdução à Ciência da Computação” written by Professors Nei Yoshihiro Soma e Fábio Carneiro Mokarzel. Although the employment of much zeal and technique in the edition of that book, typing and printing errors occurred.

The authors decided to publish this document as a guide for the readers to correctly utilize the knowledge transmitted by that book, showing the points to be corrected and giving the respective correct texts.

The document is divided in as much main section as are the chapters of the book, including the Preface, the Summary and the Bibliography.

Palavras-Chave: Ciência da Computação, Programação de Computadores, Linguagem C, Leitura correta.

1 Introdução

O livro “Introdução à Ciência da Computação”, escrito pelos mesmos autores deste documento, tem o objetivo de ensinar os primeiros passos de programação de computadores, de apresentar características da estrutura básica desses equipamentos e de fazer uma sucinta introdução às estruturas de dados, de grande utilidade para o desenvolvimento de algoritmos e programas manipuladores de quantidades significativas de informação, como são os programas desenvolvidos para as diversas modalidades de Engenharia.

Mesmo tendo sido escrito e editado com muito zelo e técnica, ocorreram não poucos erros de digitação e impressão, e os autores se sentem responsáveis por produzir um guia para uma leitura correta do mesmo, afim de que os conhecimentos ali transmitidos possam ser devidamente retidos pelos leitores. Este é um dos objetivos deste documento. Além disso, ele deverá ser um instrumento para os editores na elaboração de uma edição corrigida do livro.

O documento está dividido em tantas seções principais quantos são os capítulos do livro, incluindo a parte inicial contendo o Prefácio e o Sumário. A cada erro reportado, é indicada sua localização e seu texto substitutivo.

Há, porém, erros que aparecem ao longo de todos os capítulos. Esses são reportados logo a seguir.

Erros ao longo de todos os capítulos:

1) Deve-se eliminar o trema (‘¨’) das palavras que o apresentarem (seqüência, freqüente, etc.).
2) Onde está escrito “indentação”, deve-se substituir por “endentação”.
3) Em códigos escritos em C, os caracteres ‘‶’ e ‘″’ devem ser substituídos pelo caractere ‘"’.
4)
5)

6) 2 Correções no Prefácio e Sumário

Prefácio 2ª página, 2º parágrafo: Onde está escrito:
Observa-se, também, que muitas atividades de Engenharia envolvem a manipulação de quantidades imensas de informação das mais variadas espécies,

Deve-se substituir por:
Observa-se também que muitas atividades de Engenharia envolvem a manipulação de quantidades imensas de informações das mais variadas espécies,

Prefácio 2ª página, 4º parágrafo: Onde está escrito:
Os autores acreditam que, para que o engenheiro esteja preparado para o pleno uso da ferramenta disponível na computação,

Deve-se substituir por:
Os autores acreditam que, para que o engenheiro esteja preparado para o pleno uso das ferramentas disponíveis na computação,

Prefácio 4ª página, 6º parágrafo: Onde está escrito:
É visto o conceito de arquivo binário, comparando-o com os arquivos textos abordados no Capítulo VI e são estudadas .....

Deve-se substituir por:
É visto o conceito de arquivo binário, comparando-o com os arquivos textos abordados no Capítulo 6 e são estudadas .....

Sumário: Onde está escrito:
9 Variáveis do Tipo Ponteiro

Deve-se substituir por
9 Ponteiros

Sumário: Deve-se substituir todo o sumário por:

3 Correções no Capítulo 1 – Introdução

Página 2, 3º parágrafo: Onde está escrito:

Apesar da sofisticada estrutura interna dos computadores modernos, os princípios básicos de funcionamento dessas máquinas pode ser explicado de forma .....

Deve-se substituir por:

Apesar da sofisticada estrutura interna dos computadores modernos, os princípios básicos de funcionamento dessas máquinas podem ser explicados de forma .....

Página 2, último parágrafo: Onde está escrito:

Em seguida, faz o mesmo com a instrução contida em Esc 2, de Esc 3, e assim ele vai executando .....

Deve-se substituir por:

Em seguida, faz o mesmo com a instrução contida em Esc 2, Esc 3, Esc 4, e assim ele vai executando .....

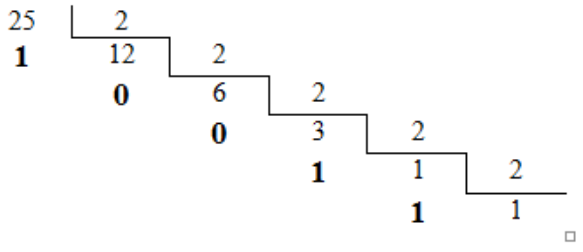
Página 13, título da Figura 1.7: Onde está escrito:

Figura 1.7 Memória de 1 mega, palavras de 2 bytes

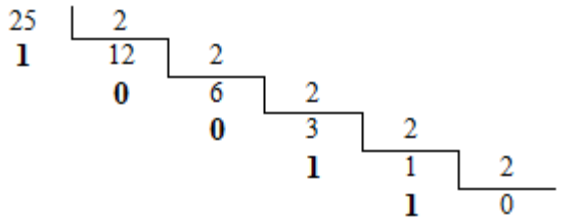
Deve-se substituir por:

Figura 1.7 Memória de 1 mega palavras de 2 bytes

Página 16, Exemplo 1.5: Onde está escrito:



Deve-se substituir por:



Página 17, Exemplo 1.6: Onde está escrito:

$(0.2B)_{16} = 2 * 16^{-1} + 11 * 16^{2} = ...$

Deve-se substituir por:

$(0.2B)_{16} = 2 * 16^{-1} + 11 * 16^{-2} = ...$

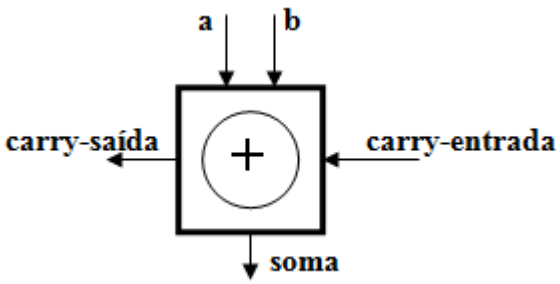
Página 17, Exemplo 1.7: Onde está escrito:

2 * 0.3 = 0 + 0.6
2 * 0.6 = 1 + 0.2
2 * 0.2 = 0 + 0.6
2 * 0.4 = 0 + 0.2
2 * 0.8 = 1 + 0.6
2 * 0.6 = 1 + 0.2

Deve-se substituir por:

2 * 0.3 = 0 + 0.6
2 * 0.6 = 1 + 0.2
2 * 0.2 = 0 + 0.4
2 * 0.4 = 0 + 0.8
2 * 0.8 = 1 + 0.6
2 * 0.6 = 1 + 0.2

Página 22: Deve-se substituir a Figura 1.11 por:



Página 25, 2º parágrafo: Onde está escrito:

Mas, se ao contrário esse bit for um, antes da conversão o conjunto de bits .....

Deve-se substituir por:

Mas, se ao contrário esse bit for um, antes da conversão, o conjunto de bits .....

Página 27, 2º parágrafo: Onde está escrito

Carrys:
Os dois mais à esquerda são diferentes

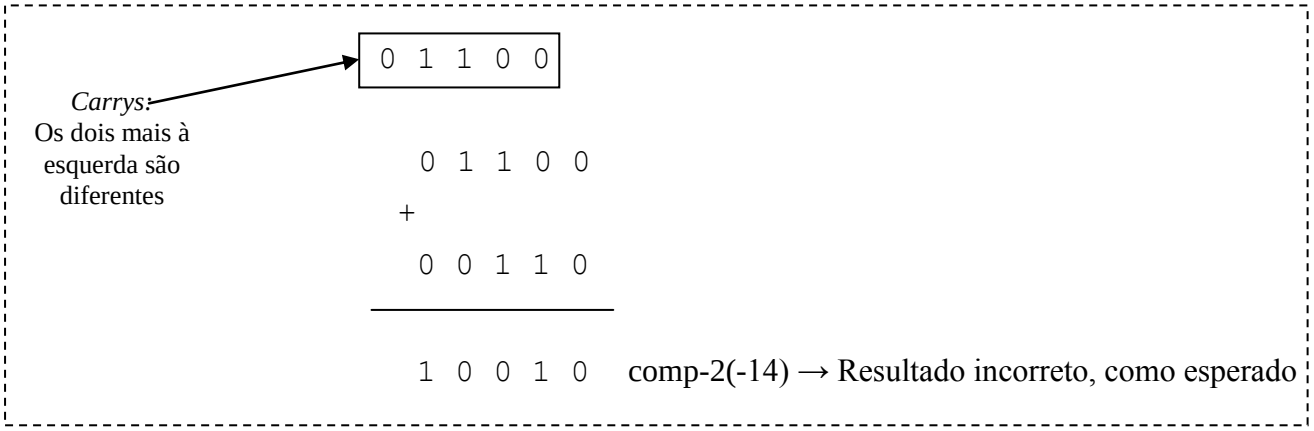
```
0 1 1 0 0

  0 1 1 0 0
+
  0 0 1 1 0
-----------
0 1 0 0 1 0
```

Como o número de bits do resultado deve ser 5, elimina-se o bit mais à esquerda, resultando

1 0 0 1 0 = comp-2 (-14) → Resultado incorreto, como esperado

Deve-se substituir por:



Página 28, 1º parágrafo: Onde está escrito

onde ***mantissa***, **base** e **expoente**

Deve-se substituir por:

onde ***mantissa***, ***base*** e ***expoente***

Página 29, Exemplo 1.17, 2º parágrafo: Onde está escrito:

....., seu intervalo de representação será [-128, +127].

Deve-se substituir por:

....., seu intervalo de representação seria [-128, +127].

Página 31, 1º parágrafo após o Exemplo 1.19: Onde está escrito:

Assim, do exemplo anterior, ASCII (‘K’) = 75, ASCII (‘&’) = 39 e .....

Deve-se substituir por:

Assim, do exemplo anterior, ASCII (‘K’) = 75, ASCII (‘&’) = 38 e .....

Página 35, 4º parágrafo: Onde está escrito:

A Tabela 1.13 apresenta o significado de alguns termos que aparecem na coluna Ação Realizada da Tabela 1.13.

Deve-se substituir por:

A Tabela 1.14 apresenta o significado de alguns termos que aparecem na coluna **"Ação Realizada"** da Tabela 1.13.

Página 37, último parágrado: Onde está escrito

No início dos anos 50, apareceram as chamadas ***linguagens de montagem***, ou **linguagens Assembly**, ou simplesmente **Assembly's**, com a finalidade .....

Deve-se substituir por:

No início dos anos 50, apareceram as chamadas ***linguagens de montagem***, ou ***linguagens Assembly***, ou simplesmente os ***Assembly*'s**, com a finalidade .....

Página 38, 1º parágrafo: Onde está escrito:

A Tabela 1.12 mostra os mnemônicos das instruções .....

Deve-se substituir por:

A Tabela 1.13 mostra os mnemônicos das instruções .....

Página 38, Tabela 1.16: Em toda coluna Ender:

Substituir 13 por 10 e 14 por 11

Página 39, 2º parágrafo: Onde está escrito:

No programa da Tabela 1.16, além das instruções executáveis .....

Deve-se substituir por:

No programa da Tabela 1.17, além das instruções executáveis .....

Página 39, último parágrafo: Onde está escrito:

A Figura 1.15 apresenta o mesmo programa da Tabela 1.16, só que acrescido .....

Deve-se substituir por:

A Figura 1.15 apresenta o mesmo programa da Tabela 1.17, só que acrescido .....

Página 41, 1º parágrafo: Onde está escrito:

$$\frac{(A - B) * C}{(D - E) * (F + G)}$$

Deve-se substituir por

$$\frac{(A + B) * C}{(D - E) * (F + G)}$$

Página 42, Figura 1.18: Onde está escrito:

```
G:          CONST   0
Inicio      READ A
```

Deve-se substituir por:

```
G:          CONST   0
X:          CONST   0
Inicio      READ A
```

Página 43, 1º parágrafo: Onde está escrito:

linguagem de programação de alto nível (só itálico)

Deve-se substituir por

linguagem de programação de alto nível (itálico e negrito)

Página 45, 2º parágrafo: Onde está escrito:

..... pois sendo o programa principal nenhum módulo do programa o chamará durante sua execução.

Deve-se substituir por:

..... pois sendo o programa principal, nenhum módulo do programa o chamará durante sua execução.

Página 45, 2º parágrafo: Onde está escrito:

Nesse caso, o tipo de valor que ela produz é declarado com a palavra **void**. Os eventuais parâmetros ....

Deve-se substituir por:

Nesse caso, o tipo de valor que ela produz é declarado com a palavra **void**. No entanto, há ambientes de programação que só aceitam a palavra **int** antes de **main**. Os eventuais parâmetros ....

Página 50, último parágrafo: Onde está escrito:

campilador

Deve-se substituir por:

compilador

Página 53, Exercício 6: Onde está escrito:

Tabela 1.12

Deve-se substituir por

Tabela 1.13

Página 54, Exercícios 7 e 8: Onde está escrito:

Tabela 1.12

Deve-se substituir por:

Tabela 1.13

4 Correções no Capítulo 2 – Algoritmos e Programas

Página 58, Figura 2.3: Onde está escrito:

$$X1 \leftarrow x = \frac{-B + \sqrt{Delta}}{2 * A};$$

Deve-se substituir por:

$$X1 \leftarrow \frac{\mathbf{-B + \sqrt{Delta}}}{\mathbf{2 * A}};$$

Página 60, 1º parágrafo: Onde está escrito:

$$\boldsymbol{S = \frac{n * (a_1 - a_n)}{2}}$$

Deve-se substituir por:

$$\boldsymbol{S = \frac{n * (a_1 + a_n)}{2}}$$

Página 61, Figura 2.7: Onde está escrito:

$x = (b – a)/n$

Deve-se substituir por:

$\Delta x = (b – a)/n$

Página 63, Figura 2.9: Substituir toda a Figura 2.9 por:

```
IntegralTrapézioPrecisão {
        Variáveis Reais: a, b, S1, S2, STrap, Dx, p;
        Variáveis Inteiras: i, n;
        Função Real: f (Parâmetro Real);

        Ler (a, b, p);
        S2 ← 0; n ← 5;
        repetir {
                S1 ← S2; n ← 2*n; Dx ← (b-a)/n; S2 ← 0; i ← 1;
                enquanto (i ≤ n) {
                        STrap ← (Dx * (f(a+(i -1)*Dx) + f(a+i *Dx))) / 2;
                        S2 ← S2 + STrap; i ← i + 1;
                }
        } enquanto (|S2 – S1| > p);
        Escrever ("A integral de f(x) no intervalo [", a, " , ", b,
                "], com precisão ", p, " é ", S2);
}
```

Página 64, Seção 2.1.5: Acrescentar o seguinte parágrafo antes do último parágrafo da Seção 2.1.5:

Observa-se também nos referidos algoritmos o uso de pares de chaves '{' e '}' para delimitar certas sequências de comandos, principalmente quando se deseja definir o escopo de comandos condicionais e comandos repetitivos. Ao conjunto formado por um par de chaves e por uma sequência de comandos por elas delimitada dá-se o nome de ***comando composto***.

Página 66, Figura 2.13: Onde está escrito:

```
"Fatorial (", n,
  ") =  ", fat
```

Deve-se substituir por:

```
"Progressão
aritmética\n\n",
"Primeiro termo:", a1,
"Razão:", r, "Número
de termos:", n,
\n\nSoma:", soma
```

Página 67, 1º parágrafo: Onde está escrito:

O comando **switch-case** da Linguagem C, usado para comandos condicionais de várias alternativas; - comando que será abordado em capítulos posteriores, tem um esquema de execução .....

Deve-se substituir por:

O comando **switch-case** da Linguagem C, usado para comandos condicionais de várias alternativas (comando esse que será abordado em capítulos posteriores), tem um esquema de execução .....

Página 68, 1º parágrafo: Onde está escrito:

....., **sqrt (Delta)** significa raiz quadrada de delta e .....

Deve-se substituir por:

....., **sqrt (Delta)** significa raiz quadrada de Delta e .....

Página 69, Figura 2.18: Substituir toda a Figura 2.18 por:

```
IntegralTrapézioPrecisão {
        float a, b; double S1, S2, STrap, Dx, p;
        int i, n;
        double f (float);

        read (a, b, p);
          S2 = 0; n = 5;
          do {
                        S1 = S2; n = 2*n; Dx = (b-a)/n; S2 = 0; i = 1;
                  while (i <= n) {
                              STrap = Dx * (f(a + (i -1) * Dx) + f(a + i * Dx))  / 2;
                              S2 ← S2 + STrap; i ← i + 1;
                        }
          } while (fabs (S2 – S1) > p);
          Escrever ("A integral de f(x) no intervalo [", a, " , ", b,
                  "], com precisão ", p, " eh ", S2);
}
```

Página 71, primeiro parágrafo: Onde está escrito:

```
i = 7;
while (i >= 4)
        Lista[i+1] = Lista[i];
Lista[4] = “Marcos”;
```

Deve-se substituir por:

```
i = 7;
while (i >= 4) {
        Lista[i+1] = Lista[i];
        i = i-1;
}
Lista[4] = “Marcos”;
```

Página 74, Figura 2.24: Onde está escrito:

```
printf (“X1 = %f , e X2 = %f”, X1, X2);
```

Deve-se substituir por:

```
printf (“X1 = %f e X2 = %f”, X1, X2);
```

Página 75, penúltimo parágrafo: Onde está escrito:

Figura 2.24

Deve-se substituir por:

Figura 2.26

Página 77, 2º parágrafo: Onde está escrito:

Normalmente são programas cujas chamadas especificam um arquivo de dados nos quais eles deverão atuar.

Deve-se substituir por:

Normalmente são programas cujas chamadas especificam um arquivo de dados nos quais eles deverão atuar. Deve-se relembrar que, conforme mencionado na Seção 1.3.4, alguns ambientes de programação só aceitam a forma

int main ( )

Página 77, último parágrafo: Onde está escrito:

Aqueles cujos nomes vêm delimitados pelos caracteres ‘<’ e ‘>’ estão em linguagem de máquina, e aqueles entre aspas estão em C.

Deve-se substituir por:

Aqueles cujos nomes vêm delimitados pelos caracteres ‘<’ e ‘>’ contêm protótipos de funções da referida biblioteca, e aqueles entre aspas estão em C. Protótipos de funções são declarações das mesmas feitas fora de suas definições e são estudados no Capítulo 10 sobre subprogramação.

Página 79, 1º parágrafo: Onde está escrito:

Inicialmente ocorre o pré-processamento da diretiva **#include <stdio.h>**, reservando, no arquivo **preproc.c**, local para o arquivo em linguagem de máquina **stdio.h** pertencente à biblioteca da Linguagem C.

Deve-se substituir por:

Inicialmente ocorre o pré-processamento da diretiva **#include <stdio.h>**. Os protótipos contidos no arquivo **stdio.h** passam a ocupar o lugar desta diretiva no arquivo **preproc.c**. As funções correspondentes a esses protótipos, pertencentes à biblioteca da Linguagem C, já estão em linguagem de máquina e devem ser ligadas ao código objeto de **preproc.c** pelo editor de ligações.

Página 79, Tabela 2.2: Onde está escrito:

Figura 2.25

Deve-se substituir por:

Figura 2.27

Página 79, 2º parágrafo após a Tabela 2.2: Onde está escrito:

Figura 2.26

Deve-se substituir por:

Figura 2.28

Página 79, Figura 2.28: Substituir toda a Figura 2.28 por:

Arquivo **preproc.c**:

```
Protótipos das funções do stdio.h
void main () {
        int i;
        printf (“LIMITE_1: %d\n”, 100);
        i = 100;
        if (i == 200)
                printf (“i: %d”, i);
        else
                printf (“LIMITE_2: %d”, 200);
}
```

Página 80, último parágrafo: Onde está escrito:

Figura 2.27

Deve-se substituir por:

Figura 2.29

Página 81, Figura 2.29: Trocar as cinco primeiras linhas por:

```
/*********************************************************************
 *     Programa para resolver uma equacao do segundo grau            *
 *     Calcula e imprime raizes reais e complexas                    *
 *     Os coeficientes da equacao devem ser digitados pelo operador  *
 *********************************************************************/
```

Página 81, Figura 2.29:

Alinhar o penúltimo caractere **‘}’** com a palavra **else**.

Página 82, Exercício 7:

Eliminar o negrito dos itens a) e b)**.**

Página 82, Exercício 7:

Alinhar a palavra ***Divisor*** com os itens a) e b)**.**

5 Correções no Capítulo 3 – Declarações, Variáveis e Comandos de Atribuição

Página 84, 3º parágrafo: Onde está escrito:

..... por um dígito, usado para dar nome às variáveis, ...

Deve-se substituir por:

..... por um dígito, usada para dar nome às variáveis, ...

Página 88, penúltimo parágrafo: Onde está escrito:

..... com palavras de 2 bytes.

Deve-se substituir por:

..... com palavras de 4 bytes.

Página 89, Tabela 3.3: Substituir toda a Tabela 3.3 por:

Tipo	Nº de bytes	Intervalo dos valores
int	4	[-2147483648, +2147483647]
short	2	[-32768, +32767]
long	4	[-2147483648, +2147483647]
unsigned	4	[0, +4294967295]
unsigned short	2	[0, +65535]
unsigned long	4	[0, +4294967295]

Página 93, 3º parágrafo: Onde está escrito:

A Figura 3.11 mostra a ordem de execução dos operadores de uma expressão relacional, obedecendo sua precedência e associatividade.

Deve-se substituir por:

A Figura 3.11 mostra a ordem de execução das operações de uma expressão contendo operadores aritméticos, relacionais, lógicos e uma função de potenciação, obedecendo a sua precedência e associatividade.

Página 93, Tabela 3.6: Onde está escrito:

`—(prefixa)`

Deve-se substituir por:

`--(prefixa)`

Página 94, Figura 3.11: Substituir o título da Figura 3.11 por:

Figura 3.11 Esquema de execução de uma expressão

Página 94, Figura 3.13: Substituir toda a Figura 3.13 por:

```
Tamanho de tipos primitivos

              i+j:  4 bytes
             char:  1 bytes
              int:  4 bytes
             long:  4 bytes
            float:  4 bytes
           double:  8 bytes
```

Página 95, 1º parágrafo da Seção 3.6.4: Onde está escrito:

A Linguagem C tem ferramenta para resolver esse problema, ...

Deve-se substituir por:

A Linguagem C tem uma ferramenta conhecida como ***casting*** para resolver esse problema, ...

Página 97, Figura 3.16:

Alinhar o penúltimo caractere **'}'** com a palavra **while**.

Página 99, Figura 3.19: Onde está escrito:

```
a = 0; b = -5; c = 1; d = 7; e = 2;
y = 5; z = 4;
```

Deve-se substituir por: (tudo em negrito)

```
a = 0; b = -5; c = 1; d = 7; e = 2;
y = 5; z = 4;
```

6 Correções no Capítulo 4 – Comandos de Decisão

Página 104, 3º parágrafo: Onde está escrito:

Dependendo da avaliação daquele conjunto de condições, sendo elas *verdadeira* (TRUE) ou *falsa* (FALSE), ...

Deve-se substituir por:

Dependendo da avaliação daquele conjunto de condições, sendo ela *verdadeira* (TRUE) ou *falsa* (FALSE), ...

Página 104, penúltimo parágrafo: Onde está escrito:

No Capítulo 2 foi apresentado o algoritmo para resolver equações do segundo grau, e o programa, agora completo na linguagem C, é apresentado a seguir;

Deve-se substituir por:

No Capítulo 2 foi apresentado um algoritmo para resolver equações do segundo grau, e um programa, agora completo na linguagem C, é apresentado a seguir;

Página 105, Figura 4.1:

Substituir todas as ocorrências da letra ‘π’ pela palavra “Condição”.

Página 106, 2º parágrafo: Onde está escrito:

Aqui também o retorno do fluxo de processamento é feito em um único ponto.

Deve-se substituir por:

Seja a Figura 4.3, onde se expande o caso de $\mathbf{A \neq 0}$. Aqui também o retorno do fluxo de processamento é feito em um único ponto.

Página 108, Figura 4.7: Substituir toda a Figura 4.7 por:

```
#include <stdio.h>
#include <math.h>
main(){
	float A, B, C, Delta;
	printf("\n\n\t\tA = ");
	scanf("%f",&A);
	printf("\n\n\t\tB = ");
	scanf("%f",&B);
	printf("\n\n\t\tC = ");
	scanf("%f",&C);
	if(A != 0){
		Delta = B*B - 4*A*C;
		if(Delta>=0){
			printf("\n\n\t\t\tHa duas raizes reais\n\n");
		}else{
			printf("\n\n\t\t\tHa duas raizes Complexas\n\n");
		}
	}else{
		if(B != 0){
			printf("\n\n\t\t\tHah uma raiz Real\n\n");
		}else{
			if(C != 0){
				printf("\n\n\t\t\tNao hah solucao\n\n");
			}else{
				printf("\n\n\t\t\tHah infinitas solucoes\n\n");
			}
		}
	}
}
```

Página 108, último parágrafo: Onde está escrito:

....., o mesmo ocorrendo para o caso *else* caso a *condição* seja FALSE.

Deve-se substituir por:

....., o mesmo ocorrendo para o caso *else* caso a *condição* seja FALSE (Ver Figura 4.8).

Página 110, Exemplo 4.3: Onde está escrito:

Exemplo 4.13: Quantidade de raízes de uma equação do segundo grau. Se fosse pedido somente para dizer se $A*x^2 + B*x + C = 0$ tem duas raízes, já que se considera a possibilidade de se ter raízes complexas, bastaria que $A \neq 0$. Isso pode ser expresso como visto na Figura 4.13.

Deve-se substituir por:

Exemplo 4.13: Quantidade de raízes de uma equação do segundo grau.

Se fosse pedido somente para dizer se $A*x^2 + B*x + C = 0$ tem duas raízes, já que se considera a possibilidade de se ter raízes complexas, bastaria que $A \neq 0$. Isso pode ser expresso como visto na Figura 4.13.

Página 110, Figura 4.14: Substituir toda a Figura 4.14 por:

```
#include <stdio.h>
main(){
	float A, B, C;
	printf("\n\n\t\tA = ");
	scanf("%f",&A);
	printf("\n\n\t\tB = ");
	scanf("%f",&B);
	printf("\n\n\t\tC = ");
	scanf("%f",&C);

	printf("\n\n\t A quantidade exata de raizes complexas de:");
	printf("\n\n\t\t %4.2f*X^2 + %4.2f*X + %4.2f = 0 ",A,B,C);
	if(!A)
		printf("nao");
	printf(" eh de duas.");
	printf("\n\n");
}
```

Página 111, Exemplo 4.4: Onde está escrito:

Exemplo 4.4: Faça um programa em C para testar se um dado número inteiro é não negativo, divisível por 5 e se o mesmo termina em 5. No primeiro momento, mostra-se um programa incorreto que tem um tipo de erro muito comum de ser feito quando da indentação incorreta aliado ao uso de comandos If com e sem Else.

Deve-se substituir por:

Exemplo 4.4: Faça um programa em C para testar se um dado número inteiro é não negativo, divisível por 5 e se o mesmo termina em 5.

No primeiro momento, mostra-se na Figura 4.15 um programa incorreto que tem um tipo de erro muito comum de ser feito quando da endentação incorreta aliado ao uso de comandos If com e sem Else.

Página 111, Figura 4.15: Onde está escrito:

```
printf("\n\n\t\tn = ", &n);
```

Deve-se substituir por

```
printf("\n\n\t\tn = ");
```

Página 111, último parágrafo: Onde está escrito:

..... um *else* se refere sempre ao primeiro *if* que não o tenha.

Deve-se substituir por:

..... um *else* se refere sempre ao *if* sem *else* anterior mais próximo dele.

Página 112, Figura 4.16: Onde está escrito:

```
printf("\n\n\t\tn = ", &n);
```

Deve-se substituir por

```
printf("\n\n\t\tn = ");
```

Página 112, Exemplo 4.5: Onde está escrito:

Exemplo 4.5: **Determinação do IMC. O índice de massa corporal (IMC) definido como a relação do peso (será usado este termo que embora não seja o correto, invés de massa, posto que a maioria assim o usa) em kg pela altura em metros elevada ao quadrado, é comumente usado para avaliar o perfil nutricional de uma pessoa. Segundo a Organização Mundial de Saúde (OMS) tem-se o quadro visto na Figura 4.17.**

Deve-se substituir por:

Exemplo 4.5: **Determinação do IMC.**

O índice de massa corporal (IMC) definido como a relação do peso (será usado este termo que embora não seja o correto, invés de massa, posto que a maioria assim o usa) em kg pela altura em metros elevada ao quadrado, é comumente usado para avaliar o perfil nutricional de uma pessoa. Segundo a Organização Mundial de Saúde (OMS) tem-se o quadro visto na Figura 4.17.

Página 113, Figura 4.18: Substituir toda a Figura 4.18 por:

```
#include <stdio.h>
main(){
	float peso, altura;
	float imc;
	printf("\n\n\t\tPeso = ");
	scanf("%f",&peso);
	printf("\n\t\tAltura = ");
	scanf("%f",&altura);
	imc = peso/(altura*altura);
	printf("\n\n\t\tSeu IMC eh igual a %6.2f (Kg/m^2)",imc);
	printf("\n\t\tSegundo a OMS voce esta ");
	if(imc<18.50)
		printf("abaixo do peso normal");
	else{
		printf("com ");
		if(18.50 <= imc && imc< 25)
			printf("o peso normal");
		if(25 <=imc && imc < 30)
			printf("sobrepeso");
		else
			printf("a classificacao como obeso");
	}
	printf("\n\n");
}
```

Página 113, Exemplo 4.6: Onde está escrito:

Exemplo 4.6: **Leia dois números inteiros e imprima o maior deles. Usando a expressão condicional ternária tem-se o que é visto na Figura 4.19.**

Deve-se substituir por:

Exemplo 4.6: **Leia dois números inteiros e imprima o maior deles.**

Usando a expressão condicional ternária tem-se o que é visto na Figura 4.19.

Página 114, Figura 4.19: Onde está escrito:

```
printf("\n\n\t\tm = ",&m);
scanf("%d",&m);
printf("\n\n\t\tn = ",&n);
```

Deve-se substituir por:

```
printf("\n\n\t\tm = ");
scanf("%d",&m);
printf("\n\n\t\tn = ");
```

Página 115, Exemplo 4.7: Onde está escrito:

Exemplo 4.7: Determine se é par um número *num*, inteiro e positivo, e indique seu dígito menos significativo, escrevendo-o também por extenso. Todavia, não sendo este o caso, imprima *num*, seu último dígito, e diga que ele é ímpar. Assim, se *num* = 531212, a mensagem a ser impressa é: "O numero 531212 termina com o dígito 2 (dois)". Mas se *num* = 121253 então a mensagem a ser impressa deverá ser: "O numero 121253 termina com o digito 3 e eh impar".

Deve-se substituir por:

Exemplo 4.7: Paridade e último dígito de um número.

Determine se é par um número *num*, inteiro e positivo, e indique seu dígito menos significativo, escrevendo-o também por extenso. Todavia, não sendo este o caso, imprima *num*, seu último dígito, e diga que ele é ímpar. Assim, se *num* = 531212, a mensagem a ser impressa é: "O numero 531212 termina com o dígito 2 (dois)". Mas se *num* = 121253 então a mensagem a ser impressa deverá ser: "O numero 121253 termina com o digito 3 e eh impar".

Página 115, Figura 4.21: Onde está escrito:

```
printf("\n\n\n\t\tnumero = ",&num);
```

Deve-se substituir por:

```
printf("\n\n\n\t\tnumero = ");
```

Página 16, Figura 4.22: Substituir toda a Figura 4.22 por:

```
#include <stdio.h>
main(){
        int num;
        printf("\n\n\n\t\tnumero = ");
        scanf("%d",&num);
        printf("\n\n\t\tO numero %d termina com o digito ",num);
        if(num%2==0)
            switch(num%10){
                case 0:
                        printf("%d (zero)",num%10);
                        break;
                case 2:
                        printf("%d (dois)",num%10);
                        break;
                case 4:
                        printf("%d (quatro)",num%10);
                        break;
                case 6:
                        printf("%d (seis)",num%10);
                        break;
                case 8:
                        printf("%d (oito)",num%10);
            }
        else printf("%d e eh impar",num%10);
        printf("\n\n");
}
```

Página 117, Figura 4.25: Onde está escrito:

printf("\n\n\n\t\tnumero = ",&num);

Deve-se substituir por:

printf("\n\n\n\t\tnumero = ");

Página 118, Exemplo 4.8: Onde está escrito:

Exemplo 4.8: Suponha que se tenha que ler a altura de uma pessoa no formato de metros e centímetros havendo um arredondamento caso os milímetros sejam também considerados. Após a inserção do valor da altura, que para o caso deve ser expresso na base 10, deve-se imprimir o valor por extenso. Admita, ainda, que o intervalo de valores varie entre um metro e dois metros e meio. Caso os valores estejam fora desses intervalos deve haver uma mensagem alertando para o fato. Resumindo, se, por exemplo, alguém digitar o valor 1.794, seu programa deve imprimir "um metro e setenta e nove centímetros", mas se for 1.798, a impressão deve ser "um metro e oitenta centímetros".

Deve-se substituir por:

Exemplo 4.8: Escrita por extenso da altura de uma pessoa.

Suponha que se tenha que ler a altura de uma pessoa no formato de metros e centímetros havendo um arredondamento caso os milímetros sejam também considerados. Após a inserção do valor da altura, que para o caso deve ser expresso na base 10, deve-se imprimir o valor por extenso. Admita, ainda, que o intervalo de valores varie entre um metro e dois metros e meio. Caso os valores estejam fora desses intervalos deve haver uma mensagem alertando para o fato. Resumindo, se, por exemplo, alguém digitar o valor 1.794, seu programa deve imprimir "um metro e setenta e nove centímetros", mas se for 1.798, a impressão deve ser "um metro e oitenta centímetros".

Página 119, 4º parágrafo: Onde está escrito:

Por enquanto, note apenas que cada expansão feita nos fluxogramas poderiam ser funções a serem chamadas para compor o programa fonte completo.

Deve-se substituir por:

Por enquanto, note apenas que cada expansão feita nos fluxogramas poderia ser uma função a ser chamada para compor o programa fonte completo.

Página 126, Exercício 4: Onde está escrito:

printf("\n\n\tEntre com um numero inteiro: ",&n);

Deve-se substituir por:

printf("\n\n\tEntre com um numero inteiro: ");

Página 127, Exercício 10: Onde está escrito:

..... Além diso, deve haver .....

Deve-se substituir por:

..... Além disso, deve haver .....

7 Correções no Capítulo 5 – Comandos de Repetição

Página 129, 2º parágrafo: Onde está escrito:

A transcrição para a linguagem C da repetição de comandos, tem, então, duas possibilidades, que são mostradas a seguir.

Deve-se substituir por:

A transcrição para a linguagem C da repetição de comandos, tem, então, duas possibilidades, que são mostradas nas Figuras 5.2 e 5.3.

Página 130, Figura 5.4: Onde está escrito:

```
printf("\t 12\t|  x   |   y   |\n");
printf("\t 13\t|  w   |   z   |\n");
```

Deve-se substituir por:

```
printf("\t 12\t|  w   |   x   |\n");
printf("\t 13\t|  y   |   z   |\n");
```

Página 134, 1º parágrafo: Onde está escrito:

..... na seqüência há o a troca (ou não) das condições, ...

Deve-se substituir por:

..... na sequência há a troca (ou não) das condições, ...

Página 134, 2º e 3º parágrafos: Onde está escrito:

O próximo exemplo ilustra como é possível ter no mesmo programa da soma dos primeiros inteiros positivos com expressões separadas apenas usando o operador vírgula.

É possível ter o mesmo programa da soma dos primeiros inteiros positivos com expressões separadas apenas usando o operador vírgula. As explicações seguem após o programa-fonte da Figura 5.13.

Deve-se substituir por:

O próximo exemplo ilustra como é possível ter no mesmo programa da soma dos primeiros inteiros positivos com expressões separadas apenas usando o operador vírgula. As explicações seguem após o programa-fonte da Figura 5.13.

Página 134, último parágrafo: Onde está escrito:

Antes de apresentar a fórmula para determinar *qtdbits*, sejam as seguintes definições: a *função menor inteiro maior ou igual a* x, em que $x \in$ R, é representada por *x*. Como exemplos, $\lfloor 4.15 \rfloor = 5$; $\lfloor -4.15 \rfloor = -3$; $\lfloor 4 \rfloor = 4$. O logaritmo de *x*, tal que $x \varepsilon R^+$, na base 2, em computação é representado por *lg x*, ou seja, $log_2 x = lg\ x$. O valor de *qtdbits* é dado como visto na Figura 5.14. Da mesma forma, a função maior inteiro menor ou igual a *x* é representada por $\lceil x \rceil$. Por exemplo, $\lceil 4.15 \rceil = 5$ e $\lceil -4.14 \rceil = -5$.

Deve-se substituir por:

Antes de apresentar a fórmula para determinar *qtdbits*, sejam as seguintes definições: a função *menor inteiro maior ou igual a x*, em que $x \in$ R, é representada por $\lceil x \rceil$. Como exemplos, $\lceil 4.15 \rceil = 5$; $\lceil -4.15 \rceil = -4$; $\lceil 4 \rceil = 4$. Da mesma forma, a função *maior inteiro menor ou igual a x* é representada por $\lfloor x \rfloor$. Por exemplo, $\lfloor 4.75 \rfloor = 4$ e $\lfloor -4.14 \rfloor = -5$. O logaritmo de *x*, tal que $x \varepsilon R^+$, na base 2, em computação é representado por *lg x*, ou seja, $log_2 x = lg\ x$. O valor de *qtdbits* é dado como visto na Figura 5.14.

Página 135, 1º parágrafo após a Figura 5.15: Onde está escrito:

No programa da Figura 5.14, ...

Deve-se substituir por:

No programa da Figura 5.15, ...

Página 136, Exemplo 5.5: Onde está escrito:

Será apresentada aqui mais uma variação, sendo que .....

Deve-se substituir por:

Será apresentada aqui mais uma variação (Figura 5.17), sendo que .....

Página 137, 1º parágrafo: Onde está escrito:

Considere o programa na Figura 5.17, que apresenta .....

Deve-se substituir por:

Considere o programa na Figura 5.18, que apresenta .....

Página 137, Figura 5.18: Substituir toda a Figura 5.18 pela seguinte, observando as endentações corretas:

```
#include <stdio.h>
#define qtdtracos 19
#define Linha {int i; for(i=0;i<qtdtracos;i++)printf("_");}
#define NovaLinha printf("\n")
#define Tabulacao printf("\t")
void main (){
   long i, j, fatorial, Continua, MInt;
        int k;
        for(MInt = 0,k=1;k<32;k++)
                MInt=(MInt<<1)+1;
        NovaLinha;
        NovaLinha;
        Tabulacao;
        Linha;
        printf("\n\t   n         n!\n\t");
        Linha;
        NovaLinha;
        Continua = 1;
        for(i=0;i<=MInt;i++){
                fatorial = 1;
                for(j=2;i>=2&&j<=i;j++)
                        if(MInt/fatorial > j)
                                fatorial = fatorial * j;
                        else{
                                Continua = 0;
                                break;
                        }
                if(!Continua)break;
                printf("\t %3d %12d\n",i,fatorial);
        }
        Linha;
        NovaLinha;
        NovaLinha;
        Tabulacao;
        printf("Fim");
        NovaLinha;
        NovaLinha;
}
```

Página 138, 1º parágrafo da Seção 5.4: Onde está escrito:

Considere agora o programa da Figura 5. 19, que .....

Deve-se substituir por:

Considere agora o programa da Figura 5.19, que .....

Página 138, Figura 5.19: Substituir o título da Figura 5.19 por:

Pirâmide invertida.

Página 139, Figura 5.20: Substituir o título da Figura 5.20 por:

Impressão de uma linha da pirâmide.

Página 139, 1º parágrafo após a Figura 5.21: Onde está escrito:

O código da Figura 5.19 mostra que, caso o resto da divisão do número *i* seja diferente de 0, o comando .....

Deve-se substituir por:

O código da Figura 5.21 mostra que, caso o resto da divisão do número *i* por 1000 seja diferente de 0, o comando .....

Página 140, Figura 5.23: Onde está escrito:

printf("\n\n\t\tErro, n deve ser maior que 2");

Deve-se substituir por:

printf("\n\n\t\tErro, n deve ser maior que 1");

Página 142, Exemplo 5.9: Onde está escrito:

Exemplo 5.9: O programa na Figura 5.26 só contém atribuições e comandos for. Que valores estarão armazenados em *z* ao final da execução do programa? Admita que os valores de *n* são números inteiros positivos. Supondo que sua máquina tenha precisão arbitrariamente grande, deve ser mostrado ainda que para $n \geq 10$ tem-se que:

Deve-se substituir por:

Exemplo 5.9: Programa com muitos comandos for aninhados.

O programa na Figura 5.26 só contém atribuições e comandos for. Que valores estarão armazenados em *z* ao final da execução do programa? Admita que os valores de *n* são números inteiros positivos. Supondo que sua máquina tenha precisão arbitrariamente grande, deve ser mostrado ainda que para $n \geq 10$ tem-se que:

Página 143, Figura 5.26: Substituir toda a Figura 5.26 pelo seguinte programa, observando-se as endentações corretas:

```
#include <stdio.h>
main(){
        unsigned long z, n, i1, i2, i3, i4, i5, i6, i7;
        int continua;
                printf("\n\n\n\t\tn = ");
                scanf("%d",&n);
                if(n<0||n>=13){
                        printf("\n\n\t\t\tErro!!\n\n");
                        continua = 0;
                }else{ continua = 1;}
                do{
                        for(z=1, i1 =1; i1<=n; i1++){
                                for(i2=0, i3=1; i3<=i1; i3++, i2++);
                                for(i4=0, i5=1; i5<=i2; i5++){
                                        for(i6=0, i7=1; i7<=z; i7++, i6++);
                                        i4+=i6;
                                }
                                z = i4;
                        }
                        continua = 0;
                }while(continua);
        printf("\n\n\n\t\tz = %d\n\n\n",z);
}
```

Página 143, 1º parágrafo após a Figura 5.26: Onde está escrito:

Figura 5.25

Deve-se substituir por:

Figura 5.26

Página 143, Exercício 1: Onde está escrito:

char n = 65

Deve-se substituir por:

char c = 65

Página 144, Exercício 5: Onde está escrito:

Figura 5.12

Deve-se substituir por:

Figura 5.13

Página 144, Exercício 6: Onde está escrito:

Figura 5.15 e Figura 5.14

Deve-se substituir respectivamente por:

Figura 5.16 e Figura 5.15

Página 144, Exercício 7: Onde está escrito:

Figura 5.16

Deve-se substituir por:

Figura 5.17

Página 144, Exercício 8: Onde está escrito:

Figura 5.19

Deve-se substituir por:

Figura 5.21

Página 144, Exercício 9: Onde está escrito:

Figura 5.20

Deve-se substituir por:

Figura 5.22

Página 144, Exercícios 10, 11 e 12: Onde está escrito:

Figura 5.22

Deve-se substituir por:

Figura 5.23

Página 144, Exercício 13: Onde está escrito:

Figura 5.24

Deve-se substituir por:

Figura 5.25

Página 144, Exercício 14: Onde está escrito:

Considere-se o programa da Figura 5.26.

Deve-se substituir por:

Considere-se o programa da Figura 5.27.

Página 145, Exercício 18: Onde está escrito:

..... programa da Figura 5.27 a considere correta.

Deve-se substituir por:

..... programa da Figura 5.28 a considere correta.

Página 146, Exercício 23: Substituir as fórmulas por:

$$\text{sen}\,\alpha = \alpha - \frac{\alpha^3}{3!} + \frac{\alpha^5}{5!} - \frac{\alpha^7}{7!} + \frac{\alpha^9}{9!} - \cdots \qquad \alpha_{\text{radianos}} = \frac{\pi * \alpha_{\text{graus}}}{180}$$

$$\cos\alpha = 1 - \frac{\alpha^2}{2!} + \frac{\alpha^4}{4!} - \frac{\alpha^6}{6!} + \frac{\alpha^8}{8!} - \cdots$$

Página 147, Exercício 27: Onde está escrito:

Considerando o programa em C mostrado na Figura 5.28, dizer .....

Deve-se substituir por

Considerando o programa em C mostrado na Figura 5.29, dizer .....

8 Correções no Capítulo 6 – Comandos Básicos de Entrada e Saída

Página 151, 2º parágrafo: Onde está escrito:

Se a expressão fosse **`5*12`**, seu valor seria guardado em 2 bytes no formato inteiro.

Deve-se substituir por:

Se a expressão fosse **`5*12`**, seu valor seria guardado em 2 ou 4 bytes no formato inteiro, num computador com palavras de 2 ou 4 bytes, respectivamente.

Página 151, Tabela 6.1: Substituir o título da Tabela 6.1 por:

Tabela 6.1 Tabela de formatos de escrita da Linguagem C, para palavras de 2 bytes

Página 151, Exemplo 6.2: Onde está escrito:

O resultado no vídeo aparece na Figura 6.3.

Deve-se substituir por:

O resultado no vídeo para computadores com palavras de 2 bytes aparece na Figura 6.3.

Página 153, penúltimo parágrafo: Onde está escrito:

Por exemplo, caso a cadeia de controle das duas chamadas de **printf ( )** do programa da Figura 6.8 fosse **`"%-3d%-9d%-9d%20f%11f\n",`**a tabela escrita seria a da Figura 6.10.

Deve-se substituir por:

Por exemplo, caso as cadeias de controle da primeira e terceira chamadas de **printf ( )** do programa da Figura 6.8 fossem respectivamente **`"%-3s%-9s%-9s%20s%11s\n"`** e **`"%3d%9d%9d%20f%11f\n",`** a tabela escrita seria a da Figura 6.10.

Página 157, Figura 6.19: Alterar a endentação da Figura 6.19 conforme abaixo:

```
#include <stdio.h>
void main () {
      int x, j, func;

/*    Tracado da regua     */

      printf ("----|----|----|----|----|----|----|----|\n");

/*    Calculo da funcao no intervalo [-5, +7]    */

      for (x=-5;x<=7;x++) {
            func = 5 + x*x - 2*x - 3;

/*   Se x!=0, marcar o ponto da curva e um ponto do eixo das abscissas    */

            if (x != 0) {
                  if (func < 5) printf ("%*c%*c", func, '*', 5-func, '|');
                  else if (func == 5) printf ("%*c", func, '*');
                  else printf ("%*c%*c", 5, '|', func-5, '*');
            }

/*   Se x=0, tracar tambem o eixo das ordenadas    */

            else
                  for (j=1; j<=40; j++)
                        if (j == func) printf ("%c", '*');
                        else if (j == 5) printf ("%c", '|');
                        else printf ("%c", '-');
            printf ("\n");
      }
}
```

Página 159, Figura 6.23: Alterar a endentação da Figura 6.23 conforme abaixo:

```
#include <stdio.h>
#include <ctype.h>
void main () {
      char c;
      printf ("Digite uma frase encerrada por enter:\n\n\t");
      c = getchar ();
      printf ("\nLetras maiusculas digitadas:\n\n\t");
      while (!iscntrl (c)) {
            if (isupper (c))
                  printf ("%c", c);
            c = getchar ();
      }
}
```

Página 160, 2º parágrafo: Onde está escrito:

Eles devem ser convertidos no número **-4137** em complemento de 2 binário de 2 bytes, ...

Deve-se substituir por:

Eles devem ser convertidos no número **-4137** em complemento de 2 binário de 4 bytes, ...

Página 160, 4º parágrafo: Eliminar a frase:

Por exemplo, a expressão **&(n+3)** representa o endereço do local de memória dado pela soma da constante 3 com o conteúdo do local reservado para a variável **n**.

Página 160, último parágrafo: Onde está escrito:

....., a chamada dessa função irá convertê-los em complemento de 2 de 2 bytes, ...

Deve-se substituir por:

....., a chamada dessa função irá convertê-los em complemento de 2 de 2 bytes, ...

Página 161, Tabela 6.2: Substituir o título da Tabela 6.2 por:

Tabela 6.2 Tabela de formatos de leitura da Linguagem C, para palavras de 2 bytes

9 Correções no Capítulo 7 – Variáveis Indexadas

Página 179, Figura 7.1: Alterar as endentações da Figura 7.7 conforme abaixo:

```
#include <stdio.h>
void main () {
      int n, cand0, cand1, cand2, cand3, cand4, cand5, cand6, cand7,
      nulos, voto, i;

      printf ("Apuracao de eleicao:\n\n\tNumero de votos: ");
      scanf ("%d",&n); printf ("\n");
      cand0 = cand1 = cand2 = cand3 = cand4 = cand5 = cand6 = cand7 =
            nulos = 0;
      for (i = 1; i <= n; i++) {
            printf ("\t\tVoto: "); scanf ("%d", &voto);
            switch (voto) {
                  case 0: cand0++; break; case 1: cand1++; break;
                  case 2: cand2++; break; case 3: cand3++; break;
                  case 4: cand4++; break; case 5: cand5++; break;
                  case 6: cand6++; break; case 7: cand7++; break;
                  default: nulos++;
            }
      }
      printf ("\n\nResultado: \n\n\tcand0: %d; cand1: %d",
            cand0, cand1);
      printf ("\n\tcand2: %d; cand3: %d", cand2, cand3);
      printf ("\n\tcand4: %d; cand5: %d", cand4, cand5);
      printf ("\n\tcand6: %d; cand7: %d", cand6, cand7);
      printf ("\n\tnulos: %d", nulos);
}
```

Página 181, 2º parágrafo após a Figura 7.3: Onde está escrito:

..... e **B** é uma matriz tridimensional .....

Deve-se substituir por:

..... e **X** é uma matriz tridimensional .....

Página 183, Figura 7.7: Alterar as endentações da Figura 7.7 conforme abaixo:

```
      printf ("   ");
      for (j=0; j<=9; printf ("__%d", j), j++);
      printf ("_");
      for (i=0; i<=9; i++) {
            printf ("\n %d|",i);
            for (j=0; j<=9; j++) {
                  if (Cruza[i][j]) printf ("%3d", Cruza[i][j]);
                  else printf ("   ");
            }
            printf (" |\n  |%32c", '|');
      }
      printf ("\n  -");
      for (j=0; j<=9; printf ("---"), j++);
      printf ("--");
```

Página 184: Onde está escrito:

C[0] = 0.13; C[1] = -5.23; C[2] = 4.1; C[3] = -2.8;

Deve-se substituir por:

D[0] = 0.13; D[1] = -5.23; D[2] = 4.1; D[3] = -2.8;

Página 186: Onde está escrito:

$$\mathbf{C_{i,j} = \sum_{k=0}^{n-1} A_{i,k} + B_{k,j}}$$

Deve-se substituir por:

$$\mathbf{C_{i,j} = \sum_{k=0}^{n-1} A_{i,k} * B_{k,j}}$$

Página 190, Figura 7.13: Alterar as endentações da Figura 7.13 conforme abaixo:

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
#define logic char
typedef int vetor[50];
void main () {
      int n, i, p, aux; logic trocou;
      vetor Vet;

      printf ("Ordenacao de numeros pelo Bubble-Sort\n\n");
      printf ("\tNumero de elementos: "); scanf ("%d",&n);
      printf ("\n\tElementos: ");
      for (i=0; i<n; i++)scanf("%d", &Vet[i]);
      printf ("\n\nVetor desordenado:\n\n");
      for (i=0; i<n; i++) printf("%4d", Vet[i]);
      for (trocou = TRUE, p = n-2; p>=0 && trocou == TRUE; p--) {
            for (trocou = FALSE, i = 0; i<=p; i++)
                  if (Vet[i] > Vet[i+1]) {
                        aux = Vet[i]; Vet[i] = Vet[i+1];
                        Vet[i+1] = aux; trocou = TRUE;
                  }
      }
      printf ("\n\nVetor ordenado:\n\n");
      for (i=0; i<n; i++) printf("%4d", Vet[i]);
}
```

Página 192 e 193: Onde está escrito:

sup = med -- 1;

Deve-se substituir por:

sup = med -1;

Página 194, Figura 7.17: Onde está escrito:

```
do c = getche();
while (c != 's' && c != 'n' && c != 's' && c != 'n');
```

Deve-se substituir por

```
do c = getche();
while (c != 's' && c != 'n' && c != 'S' && c != 'N');
```

Página 197, parágrafo após "Soma dos polinômios": Onde está escrito:

....., indo do índice **zero** até aquele correspondente grau do polinômio resultado, ...

Deve-se substituir por:

....., indo do índice **zero** até aquele correspondente ao grau do polinômio resultado, ...

Página 199, Figura 7.20: Onde está escrito:

```
i—
```

Deve-se substituir por:

```
i--
```

Página 200, Figura 7.20: Onde está escrito:

```
printf ("\n\nPolinomio Soma PS: Grau %d\n\n", ns);
for (i = 0; i <= ns; i++)
     if (PS[i] != 0.0 || ns == 0) {
          printf (" + (%g)", PS[i]);
          if (i != 0) {
               printf ("*X");
               if (i > 1)
                    printf ("**%d", i);
          }
}
```

Deve-se substituir pelo seguinte trecho, observando as endentações:

```
      printf ("\n\nPolinomio Soma PS: Grau %d\n\n", ns);
      for (i = 0; i <= ns; i++)
            if (PS[i] != 0.0 || ns == 0) {
                  printf (" + (%g)", PS[i]);
                  if (i != 0) {
                        printf ("*X");
                        if (i > 1)
                              printf ("**%d", i);
                  }
            }
```

Página 200, Figura 7.20: Onde está escrito:

```
/* Multiplicacao dos polinomios e escrita do resultado   */

      nm = n1 + n2;
      for (j = 0; j <= n2; j++)
            if (P2[j] != 0.0) {
                  polin Paux = {0.0};
                  for (i = 0; i <= n1; i++)
                        Paux[i+j] = P1[i] * P2[j];
                  naux = n1 + j;
                  for (i = 0; i <= naux; i++)
                        PM[i] = Paux[i] + PM[i];
      }
      printf ("\n\nPolinomio Produto PM: Grau %d\n\n", nm);
      for (i = 0; i <= nm; i++)
            if (PM[i] != 0.0 || nm == 0) {
                  printf (" + (%g)", PM[i]);
                  if (i != 0) {
                        printf ("*X");
                        if (i > 1)
                              printf ("**%d", i);
            }
      }
}
```

Deve-se substituir pelo seguinte trecho, observando as endentações:

```
/* Multiplicacao dos polinomios e escrita do resultado   */

      nm = n1 + n2;
      for (j = 0; j <= n2; j++)
            if (P2[j] != 0.0) {
                  polin Paux = {0.0};
                  for (i = 0; i <= n1; i++)
                        Paux[i+j] = P1[i] * P2[j];
                  naux = n1 + j;
                  for (i = 0; i <= naux; i++)
                        PM[i] = Paux[i] + PM[i];
            }
      for (i = 0; i <= nm; i++)
            if (PM[i] != 0.0 || nm == 0) {
                  printf (" + (%g)", PM[i]);
                  if (i != 0) {
                        printf ("*X");
                        if (i > 1)
                              printf ("**%d", i);
                  }
            }
}
```

Página 202, Figura 7.22: Onde está escrito:

```
      for (i = 0; i < n; i++)
      for (j = 0; j < n; j++) scanf ("%d", &A[i][j]);
```

Deve-se substituir por:

```
      for (i = 0; i < n; i++)
            for (j = 0; j < n; j++) scanf ("%d", &A[i][j]);
```

Página 202, Figura 7.22: Onde está escrito:

```
                  for (j = 1; j < n; j++) {
                        aux = A[i][j]; A[i][j] = A[j][i]; A[j][i] = aux;
                  }
```

Deve-se substituir por:

```
                  for (j = i+1; j < n; j++) {
                        aux = A[i][j]; A[i][j] = A[j][i]; A[j][i] = aux;
                  }
```

Página 205, Figura 7.24: Na Figura 7.24, alterar as endentações conforme abaixo:

```
/*    Sintetizacao da Matriz B na matriz A          */

            for (i = 0; i < 5; i++)
                  for (j = 0; j < 5; j++) {
                        aux = 0;
                        for (x = 4*i; x <= 4*i + 3; x++)
                              for (y = 4*j; y <= 4*j + 3; y++)
                                    aux += B[x][y];
                        A[i][j] = aux/16;
                  }
```

Página 206, Figura 7.26: Alterar as endentações da Figura 7.26 conforme abaixo:

```
LifeGame {
      Apresentar título do Life Game na tela;
      Indagar o operador sobre do desejo de brincar;
      Em caso positivo {
            Ler de arquivo em disco o estado inicial do tabuleiro;
            Sinalizar que o Life está em estado inicial;
            Repetir os comandos {
                  Escrever na tela o estado atual do tabuleiro;
                  Se o estado for o inicial {
                        Avisar isso ao operador;
                        Pedir para digitar algo para continuar;
                  }
                  Determinar o número de vizinhos vivos de cada célula do
                        tabuleiro;
                  Determinar o estado seguinte do tabuleiro;
                  Se o sinal de estado inicial estiver aceso
                        Apagá-lo;
                  Senão
                        Indagar o operador sobre do desejo de continuar;
            } Enquanto o operador quiser continuar a brincadeira;
      }
}
```

Página 208 e 209, Figura 7.28: Alterar as endentações da Figura 7.28 conforme abaixo:

```
/*    Declaracoes      */

#include <stdio.h>
#include <conio.h>

typedef char matriz [15][25];

void main () {

      matriz Vizinhos, Tabuleiro; int i, j, inic, x, y, aux; char c;
      FILE *file;

/*    Apresentacao do titulo do brinquedo na tela   */

      gotoxy (30, 12); printf ("L I F E    G A M E");
      gotoxy (1, 25); printf ("Deseja brincar? (s/n): ");
      do c = getche (); while (c != 's' && c != 'n' && c != 'S' && c != 'N');
      if (c == 's' || c == 'S') {

/*    Leitura do estado inicial do tabuleiro       */

            file = fopen ("life.dat", "r");
            for (i = 0; i < 15; i++)
                  for (j = 0; j < 25; j++) fscanf (file, "%d", &Tabuleiro[i][j]);
            inic = 1;

/*    Inicio do processo repetitivo     */

            do {

/*    Escrita na tela do estado atual do tabuleiro       */

                  clrscr ();
                  printf ("\n                ------------------------");
                  for (i = 3; i <= 17; i++) {
                        gotoxy (15, i); printf ("%c%26c", '|', '|');
                  }
                  printf ("\n                ------------------------");
                  for (i = 0; i < 15; i++) {
                        gotoxy (16, i+3);
                        for (j = 0; j < 25; j++) {
                              if (Tabuleiro[i][j] == 0) printf (" ");
                              else printf ("*");
                        }
                  }

/*    Aviso na tela sobre o estado inicial    */
```

```
                if (inic == 1) {
                      gotoxy (1, 25);
                      printf ("Estado inicial! Digite algo para continuar!");
                      getch ();
                }

/*    Calculo da matriz dos vizinhos das celulas    */

                for (i = 0; i < 15; i++) {
                      for (j = 0; j < 25; j++) {
                            aux = 0;
                            for (x = i-1; x <= i+1; x++)
                                  for (y = j-1; y <= j+1; y++)
                                        if (x >= 0 && x <= 15 && y >= 0 && y <= 24)
                                              if (x != i || y != j)
                                                    if (Tabuleiro[x][y] == 1)
                                                    aux++;
                            Vizinhos[i][j] = aux;
                      }
                }

/*    Calculo do estado seguinte do tabuleiro      */

                for (i = 0; i < 15; i++)
                      for (j = 0; j < 25; j++)
                            if (Tabuleiro[i][j] == 0) {
                                  if (Vizinhos[i][j] == 3)
                                  Tabuleiro[i][j] = 1;
                            }
                            else {
                                  if (Vizinhos[i][j] != 2 && Vizinhos[i][j] != 3)
                                        Tabuleiro[i][j] = 0;
                            }

/*    Indagar sobre o desejo de continuar a brincadeira       */

                if (inic == 1) inic = 0;
                else {
                      gotoxy (1, 25);
                      printf ("                                                    ");
                      gotoxy (1, 25); printf ("Continua? (s/n): ");
                      do c = getche ();
                      while (c != 's' && c != 'n' && c != 'S' && c != 'N');
                }
/*    Final do processo repetitivo      */

          } while (c == 's' || c == 'S');
     }
}
```

Página 212, último parágrafo antes da Figura 7.31: Onde está escrito:

O resultado de sua execução, fazendo a leitura de 4 frases aparece na Figura 7.32.

Deve-se substituir por:

O resultado de sua execução, fazendo a leitura de 4 frases, aparece na Figura 7.32.

Página 214 216, Figuras 7.33 e 7.34: Onde está escrito:

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
```

Deve-se substituir por:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
```

Página 216, Figura 7.38: Substituir o título da Figura 7.38 por:

Figura 7.38 Resultado do programa da Figura 7.37

Página 217, Figura 7.39: Onde está escrito:

```
p—
```

Deve-se substituir por:

```
p--
```

Página 217, Figura 7.39: Alterar as endentações da Figura 7.39 conforme abaixo:

```
for (trocou = TRUE, p = n-2; p>=0 && trocou == TRUE; p--) {
      for (trocou = FALSE, i = 0; i<=p; i++)
            if (strcmp (Vet[i], Vet[i+1]) > 0) {
                  strcpy (aux, Vet[i]); strcpy (Vet[i], Vet[i+1]);
                  strcpy (Vet[i+1], aux); trocou = TRUE;
            }
}
```

Página 217, 1º parágrafo da Seção 7.6.2: Onde está escrito:

O programa da Figura 7.17, que aplica o método da procura binária de números em vetores numéricos ordenados também pode ser adaptado para vetores de nomes.

Deve-se substituir por:

O programa da Figura 7.17, que aplica o método da procura binária de números em vetores numéricos ordenados, também pode ser adaptado para vetores de nomes.

Página 218, Figura 7.40: Onde está escrito:

```
do c = getche();
while (c != 's' && c != 'n' && c != 's' && c != 'n');
```

Deve-se substituir por

```
do c = getche();
while (c != 's' && c != 'n' && c != 'S' && c != 'N');
```

Página 220, Figura 7.42: Substituir toda a Figura 7.42 por:

```
#include <stdio.h>
#include <conio.h>
void main () {
     int n, i, j, d; char V[501];
     printf ("Digite um numero inteiro > 0 e <= 500: ");
     scanf ("%d", &n); printf ("\n\n");
     if (n < 1 || n > 500) printf ("\tDados incompativeis");
     else {
          for (V[1] = 'N', i = 2; i <= n; i++) V[i] = 'S';
          for (d = 2; d*d <= n; d++) {
               for ( ; d <= n && V[d] == 'N'; d++);
               for (j = d*d; j <= n; j += d) V[j] = 'N';
          }
          for (i = 1, j = 0; i <= n; i++)
               if (V[i] == 'S') {
                    printf ("%10d", i); j++;
                    if (j % 5 == 0) printf ("\n\n");
               }
     }
}
```

Página 223, Exercício 14: Onde está escrito:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Deve-se substituir por:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

10 Correções no Capítulo 8 – Tipos Enumerativos e Estruturas

Página 233, Figura 8.4: Onde está escrito:

```
            for(i = 0; i < n; i++) {
                        printf ("\nEmpregado n.o %d:\n", i+1);
                  printf ("\tNome: ");
```

Deve-se substituir por:

```
            for(i = 0; i < n; i++) {
                  printf ("\nEmpregado n.o %d:\n", i+1);
                  printf ("\tNome: ");
```

Onde está escrito:

```
                  printf ("\tSalario: "); scanf ("%f", &Cadastro[i].salario);
      }
            printf ("\nListagem dos Empregados:\n");
            for(i = 0; i < n; i++) {
                        printf("\n%d) %-21s",Cadastro[i].nome);
                  printf("%-16s",Cadastro[i].ender.lograd);
```

Deve-se substituir por:

```
                  printf ("\tSalario: "); scanf ("%f", &Cadastro[i].salario);
            }
            printf ("\nListagem dos Empregados:\n");
            for(i = 0; i < n; i++) {
                  printf("\n%d) %-21s",Cadastro[i].nome);
                  printf("%-16s",Cadastro[i].ender.lograd);
```

Página 234, Exemplo 8.1: Onde está escrito:

```
            complexo A[3][3] = {
                        {{1.0, -0.1},{2.0, -0.2},{3.0, 4.3}},
                  {{4.0, -3.4},{5.0, 4.1},{6.0, -2.6}}
            };
```

Deve-se substituir por:

```
            complexo A[3][3] = {
                  {{1.0, -0.1},{2.0, -0.2},{3.0, 4.3}},
                  {{4.0, -3.4},{5.0, 4.1},{6.0, -2.6}}
            };
```

Página 235: Onde está escriito:

o comando **ender2 = ender1;** equivale a

Deve-se substituir por:

o comando **ender2 = ender;** equivale a

Página 237, 1º parágrafo: Onde está escrito:

O espaço por ela ocupado é o de um valor do tipo **float**, pois dentre os tipos de seus três campos, **float** é o que ocupa maior espaço.

Deve-se substituir por:

O espaço por ela ocupado é o de um valor do tipo **float**, pois dentre os tipos de seus três campos, **float** é o que ocupa maior espaço, num computador com palavras de 2 bytes.

Página 237, Imediatamente antes da Figura 8.8 : Onde está escrito:

Seja o programa da Figura 8.8 cujo resultado é apresentado na Figura 8.9.

Deve-se substituir por:

Seja o programa da Figura 8.8, cujo resultado de sua execução num computador de palavras de 2 bytes é apresentado na Figura 8.9.

Página 238, Exemplo 8.2: Onde está escrito:

OPERADOR: quando ele for um dos caracteres ‘+’, ‘’, ‘*’ ou ‘/’

Deve-se substituir por:

OPERADOR: quando ele for um dos caracteres ‘+’, ‘-’, ‘*’ ou ‘/’

Páginas 239, 240 e 241, Figura 8.10: Alterar a endentação da Figura 8.10, conforme abaixo:

```
/* Inclusao de arquivos da biblioteca de C  */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/* Definicao de constantes para os tipos dos atomos  */

#define NUMERO   1
#define OPERADOR 2
#define ABPAR    3
#define FPAR     4
#define INVAL    5

/* Declaracao de tipos de variaveis usadas no programa  */

typedef char expressao[30];
typedef struct atomo atomo;
typedef union atribatomo atribatomo;
union atribatomo {int valor; char oper, carac;};
struct atomo {int tipo; atribatomo atrib;};

/* Cabecalho da funcao main e declaracoes de variaveis  */

void main() {
     char c; expressao expr; atomo VetAtomo[30]; int natom, i, num;

/* Digitacao da expressao  */

     printf ("ATOMOS DE UMA EXPRESSAO: ");
     printf ("\n\nDigite a expressao:\n\n\t "); fflush (stdin); gets (expr);


/* Formacao dos atomos da expressao  */

     natom = 0; i = 0;
     while (1) {

/*  Procura do primeiro caractere nao branco da expressao  */

          while (isspace (expr[i]) ||
                (iscntrl (expr[i]) && expr[i] != '\0')) i++;
          c = expr[i];

/* Se o primeiro caractere for o '\0', encerrar a formacao  */

          if (c == '\0') break;
```

```
/* Se for um digito, o atomo eh do tipo NUMERO; Deve-se coletar os outros
   digitos e formar o numero correspondente que sera o atributo do atomo */

          if (isdigit(c)) {
               VetAtomo[natom].tipo = NUMERO; num = 0;
               while (isdigit (c)) {
                    num = 10*num + c - '0'; i++; c = expr[i];
               }
               VetAtomo[natom].atrib.valor = num;
          }

/* Se for um dos 4 operadores, o atomo eh do tipo OPERADOR; O atributo eh
   o proprio caractere  */

          else if (c == '+' || c == '-' || c == '*' || c == '/') {
               VetAtomo[natom].tipo = OPERADOR;
               VetAtomo[natom].atrib.oper = c; i++; c = expr[i];
          }

/* Se for um abre ou fecha-parentesis o atomo eh do tipo ABPAR ou FPAR;
   Nao ha atributo  */

          else if (c == '(') {
          VetAtomo[natom].tipo = ABPAR; i++; c = expr[i];
     }
          else if (c == ')') {
          VetAtomo[natom].tipo = FPAR; i++; c = expr[i];
     }

/* Se for qualquer outro caractere, o atomo eh do tipo INVAL; O atributo eh
   o proprio caractere  */

          else {
               VetAtomo[natom].tipo = INVAL;
               VetAtomo[natom].atrib.carac = c; i++; c = expr[i];
          }

/* Acrescimo de um ao numero de atomos ja formados*/

          natom++;
     }
```

```
/* Escrita dos atomos  */

      printf ("\nTIPO     |  ATRIBUTO");
      printf ("\n-----------------------");
      for (i = 0; i < natom; i++) {
            printf ("\n");
            switch (VetAtomo[i].tipo) {
                  case NUMERO:
                        printf ("NUMERO   |  %d", VetAtomo[i].atrib.valor);
                  break;
                  case OPERADOR:
                        printf ("OPERADOR |  %c", VetAtomo[i].atrib.oper);
                  break;
                  case INVAL:
                        printf ("INVAL    |  %c", VetAtomo[i].atrib.carac);
                  break;
                  case ABPAR:
                        printf ("ABPAR    |"); break;
                  case FPAR:
                        printf ("FPAR     |"); break;
            }
      }
}
```

Páginas 243, 244 e 245, Figura 8.13: Alterar a endentação da Figura 8.13, conforme abaixo:

```
/* Declaracoes das variaveis  */

void main () {
      morador Moradores[10];
      int i, nmor; char c;

/* Inicio da entrada de dados  */

      printf ("CADASTRO DE MORADORES DE UM PREDIO");
      printf ("\n\nNumero de moradores: "); scanf ("%d", &nmor);
      for (i = 0; i < nmor; i++) {

/* Leitura das informacoes comuns a cada morador  */

            printf ("\nMorador n.o %d:", i+1);
            printf ("\n\tNome: "); fflush (stdin); gets (Moradores[i].nome);
            printf ("\tAndar: "); scanf ("%d", &Moradores[i].andar);
            printf ("\tApto: "); scanf ("%d", &Moradores[i].nap);
            printf ("\tIdade: "); scanf ("%d", &Moradores[i].idade);

/* Leitura das informacoes dos moradores com idade de zero a 3 anos  */
```

```
            if (Moradores[i].idade >= 0 && Moradores[i].idade <= 3) {
                  printf ("\tNumero de Mamadeiras: ");
                  scanf ("%d", &Moradores[i].infoadic.info03.nmamads);
            }

/* Leitura das informacoes dos moradores com idade de 4 a 11 anos  */

            else if (Moradores[i].idade >= 4 && Moradores[i].idade <= 11) {
                  printf ("\tEscolaridade ");
                  printf ("(0 - Nenhuma, 1 - Basica, 2 - Media, 3 - Superior): ");
                  scanf ("%d", &Moradores[i].infoadic.info411.esc);
                  printf ("\tNumero de Brinquedos: ");
                  scanf ("%d", &Moradores[i].infoadic.info411.nbrinq);
            }

/* Leitura das informacoes dos moradores com idade de 12 a 17 anos  */

            else if (Moradores[i].idade >= 12 && Moradores[i].idade <= 17) {
                  printf ("\tEscolaridade ");
                  printf ("(0 - Nenhuma, 1 - Basica, 2 - Media, 3 - Superior): ");
                  scanf ("%d", &Moradores[i].infoadic.info1217.esc);
                  printf ("\tNumero de Livros Lidos: ");
                  scanf ("%d", &Moradores[i].infoadic.info1217.nlivros);
                  printf ("\tPassatempo Predileto: ");
                  fflush (stdin);
                  gets (Moradores[i].infoadic.info1217.passatempo);
            }

/* Leitura das informacoes dos moradores com idade de 18 anos ou mais  */

            else {
                  printf ("\tEscolaridade ");
                  printf ("(0 - Nenhuma, 1 - Basica, 2 - Media, 3 - Superior): ");
                  scanf ("%d", &Moradores[i].infoadic.info18.esc);
                  printf ("\tProfissao: ");
                  fflush (stdin);
                  gets (Moradores[i].infoadic.info18.profissao);
                  printf ("\tEh casado? (s/n): ");
                  c = Moradores[i].infoadic.info18.ehcasado = getche ();
                  if (c == 's' || c == 'S') {
                        printf ("\n\tNome do conjuge: ");
                        fflush (stdin);
                        gets (Moradores[i].infoadic.info18.conjuge);
                        printf ("\tNumero de Filhos: ");
                        scanf ("%d", &Moradores[i].infoadic.info18.nfilhos);
                  }
                  else printf ("\n");
                  }
            }

/* Listagem das informacoes digitadas  */
```

```
      printf ("\nListagem dos Moradores:\n");
      for (i = 0; i < nmor; i++) {

/* Informacoes comuns  */

            printf("\n%d) Nome: %-21s; Andar: %4d; Apto: %5d; Idade: %4d",
            i+1, Moradores[i].nome, Moradores[i].andar,
            Moradores[i].nap, Moradores[i].idade);

/* Moradores de zero a 3 anos  */

            if (Moradores[i].idade >= 0 && Moradores[i].idade <= 3) {
                  printf ("\n\tMamadeiras: %d;",
                  Moradores[i].infoadic.info03.nmamads);
            }

/* Moradores de 4 a 11 anos  */

            else if (Moradores[i].idade >= 4 && Moradores[i].idade <= 11) {
                  printf ("\n\tEscolaridade: %d; Brinquedos: %d",
                        Moradores[i].infoadic.info411.esc,
                        Moradores[i].infoadic.info411.nbrinq);
            }

/* Moradores de 12 a 17 anos  */

            else if (Moradores[i].idade >= 12 && Moradores[i].idade <= 17) {
                  printf ("\n\tEscolaridade: %d; Livros: %d; Passatempo: %s",
                        Moradores[i].infoadic.info1217.esc,
                        Moradores[i].infoadic.info1217.nlivros,
                        Moradores[i].infoadic.info1217.passatempo);
            }

/* Moradores de 18 anos ou mais  */

            else {
                  printf ("\n\tEscolaridade %d; Profissao %s; Casado: %c;",
                        Moradores[i].infoadic.info18.esc,
                        Moradores[i].infoadic.info18.profissao,
                        Moradores[i].infoadic.info18.ehcasado);
                        c = Moradores[i].infoadic.info18.ehcasado;
                  if (c == 's' || c == 'S') {
                        printf ("\n\tConjuge: %s, Filhos: %d",
                              Moradores[i].infoadic.info18.conjuge,
                              Moradores[i].infoadic.info18.nfilhos);
                  }
            }
      }
}
```

Página 246, Exercício 2: Onde está escrito:

Seja o programa da Figura 8.4 .....

Deve-se substituir por:

Seja o programa da Figura 8.14 .....

11 Correções no Capítulo 9 – Ponteiros

Página 250: Substituir o título do Capítulo 9 por:

Ponteiros

Página 250, 1º parágrafo: Onde está escrito:

Este capítulo tem o objetivo de apresentar os conceitos e as principais utilidades das variáveis do tipo ponteiro, bem como .....

Deve-se substituir por:

Este capítulo tem o objetivo de apresentar os conceitos e as principais utilidades das constantes e variáveis do tipo ponteiro, bem como .....

Página 250: Substituir os títulos das seções 9.1 e 9.1.1 por:

9.1 Introdução aos ponteiros

9.1.1 Constantes, variáveis, declarações e atribuições de ponteiros

Página 250, 1º parágrafo da Seção 9.1.1: Onde está escrito:

As variáveis dos programas apresentados nos capítulos anteriores são usadas para guardar e manipular valores de determinados tipos. Variáveis do tipo ponteiro são destinadas a guardar e manipular endereços de memória. Considere, por exemplo, o seguinte fragmento de programa:

Deve-se substituir por:

As variáveis dos programas apresentados nos capítulos anteriores são usadas para guardar e manipular valores de determinados tipos. Endereços também podem ser manipulados pela Linguagem C. Por exemplo, sendo **a** uma variável declarada em um programa, **&a** é o seu endereço. Ponteiros ou valores do tipo ponteiro são endereços.

Assim como existem constantes e variáveis de tipo inteiro, real, caractere, cadeia de caractere, vetor, matriz e estrutura, também existem constantes e variáveis do tipo ponteiro, ou seja, do tipo endereço. Uma constante desse tipo não pode ter seu valor alterado durante a execução. Então **&a** é uma constante do tipo ponteiro (constante-ponteiro), já que o endereço de uma variável qualquer

permanece constante durante a execução. Uma variável do tipo ponteiro (variável-ponteiro) é destinada a guardar um valor do tipo ponteiro, ou seja, é destinada a guardar um endereço de memória. Durante a execução, o valor dessa variável pode ser alterado.

Variáveis-ponteiros podem guardar endereços de outras variáveis. Considere por exemplo o seguinte fragmento de programa:

Página 250, penúltimo parágrafo: Onde está escrito:

A Figura 9.1 mostra um mapa da memória depois de sua execução, em certo computador, num dado ambiente .....

Deve-se substituir por:

A Figura 9.1 mostra um mapa da memória depois de sua execução, num computador de palavras de 2 bytes, num dado ambiente .....

Página 255, dois últimos parágrafos: Onde está escrito:

Na Linguagem C, diferentemente do que ocorre nas linguagens de programação usadas quando de sua criação, há uma forte relação entre ponteiros e variáveis indexadas. Tudo começa pelo fato de o nome de uma variável indexada ocupar um local na memória contendo o endereço do primeiro de seus elementos. Então, na realidade esse nome é um ponteiro para tal elemento, que, por apontar para um local fixo, tem valor constante durante a execução do programa. Considerando-se, por exemplo, a seguinte declaração:

```
int A[8];
```

a interpretação gráfica da variável **A** pode ser aquela da Figura 9.9, acrescentando-se o fato de que o ponteiro nela contido não pode ser alterado por nenhum comando do programa.

Deve-se substituir por:

Na Linguagem C, diferentemente do que ocorre nas linguagens de programação usadas quando de sua criação, há uma forte relação entre ponteiros e variáveis indexadas. Tudo começa pelo fato de que o nome de uma variável indexada é uma constante-ponteiro apontando para o primeiro de seus elementos. Esse nome não tem local exclusivo na memória e, por ser uma constante, seu valor não pode ser alterado durante a execução do programa. Considerando-se, por exemplo, a seguinte declaração:

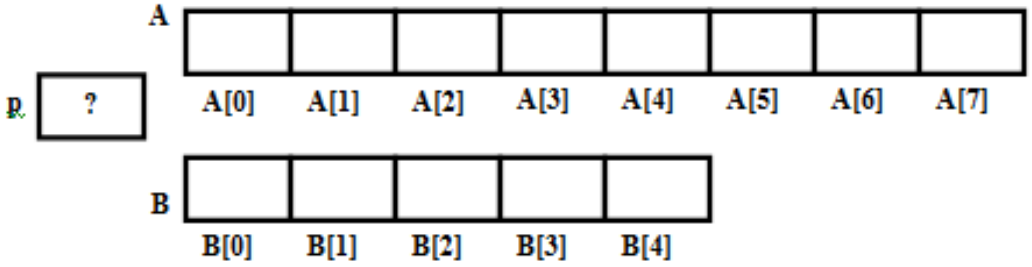
```
int A[8];
```

a representação gráfica da variável **A** pode ser aquela da Figura 9.9.

Página 256, Figura 9.9: Substituir toda a Figura 9.9 por:

Página 256, Figura 9.10: Substituir toda a Figura 9.10 por:

Situação inicial:



Situação logo após a atribuição **p = A**:

Situação logo após a atribuição **p = B**:

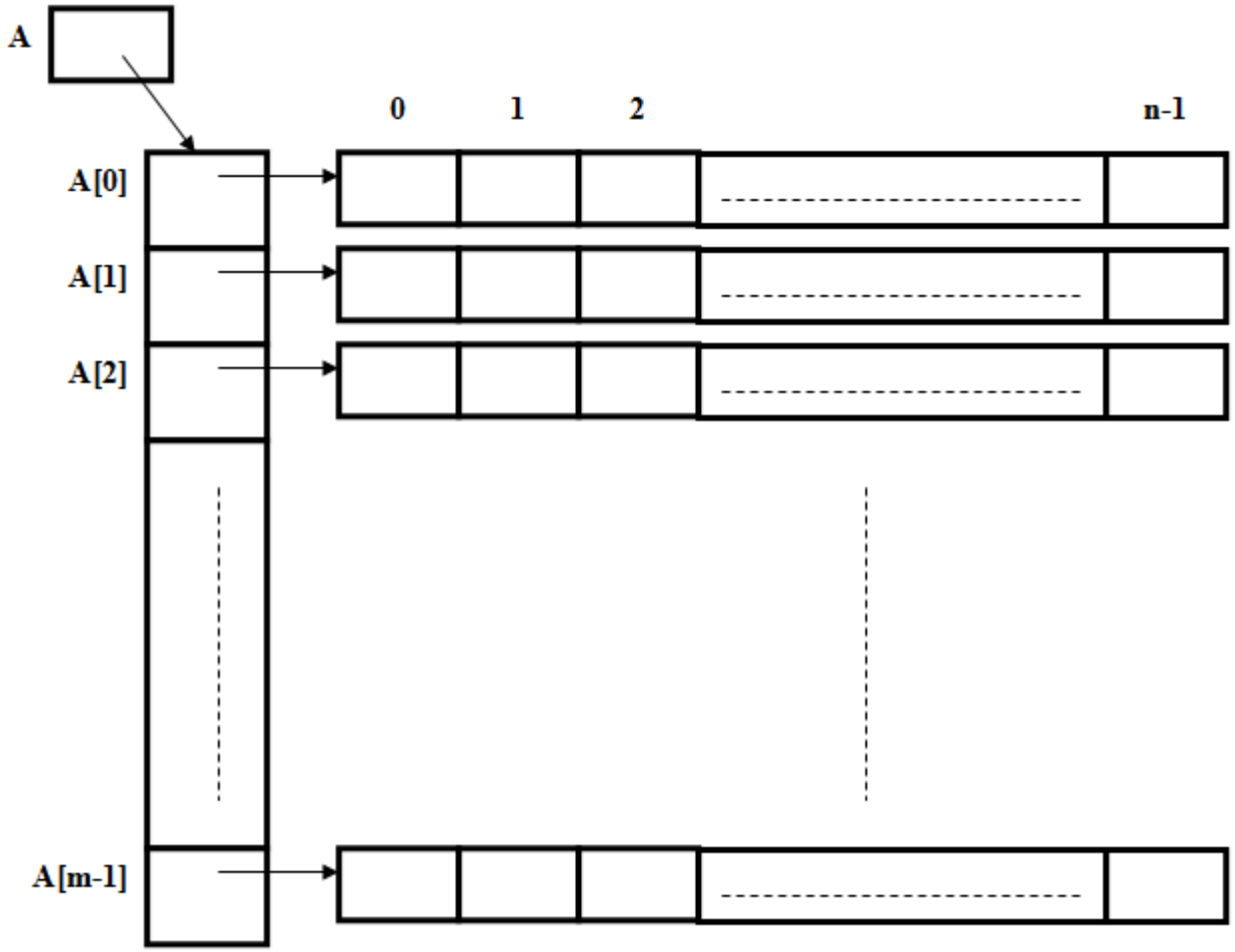
Página 260, 1º parágrafo: Onde está escrito:

Quando uma função é chamada para execução é que sua área de dados é carregada na memória, a partir da fronteira com a área desocupada.

Deve-se substituir por:

Quando uma função é chamada para execução, sua área de dados é carregada na memória, a partir da fronteira com a área desocupada.

Página 261, Figura 9.16: Substituir toda a Figura 9.16 por:



Página 263, Exercício 1: Onde está escrito:

Alterar os programas das figuras 7.10, 7.13, 7.17, 7.20, 7.34 e 7.43 .....

Deve-se substituir por:

Alterar os programas das figuras 7.10, 7.13, 7.17, 7.20, 7.24 e 7.28 .....

12 Correções no Capítulo 10 – Subprogramação

Página 267, 1º parágrafo: Onde está escrito:

..... subprogramas que realizam tarefas relacionadas com os argumentos de chamada e retornem valores.

Deve-se substituir por:

..... subprogramas que realizam tarefas relacionadas com os argumentos de chamada e também retornam valores.

Página 267, 2º parágrafo: Onde está escrito:

A Figura 10.2, que é uma repetição da Figura 2.24, apresenta .....

Deve-se substituir por:

A Figura 10.2, que é uma repetição da Figura 2.26, apresenta .....

Página 269, Figura 10.3: Alterar a endentação da função **main** da Figura 10.3, conforme abaixo:

```
void main ( ) {
      int n, i; char c; long fat;
      printf ("Fatorial de um numero (s/n)? "); c = getche();
      while (c == 's' || c == 'S') {
            printf ("\nDigite o numero: ");
            scanf ("%d", &n);
            fat = fatorial (n);
            if (fat != -1)
                  printf ("Fatorial (%d) =  %ld", n, fat);
            else if (n < 0)
                  printf ("Fatorial (%d): %d eh negativo", n, n);
            else
                  printf ("Fatorial (%d) nao cabe em 4 bytes", n);
            printf ("\n\nFatorial de um numero (s/n)? "); c = getche();
      }
      printf ("\n\nFim do programa!");
}
```

Página 269, último parágrafo: Onde está escrito:

Conforme apresentado nos capítulos iniciais deste livro, a programação de uma tarefa complexa pode ser decomposta num conjunto de tarefas menores.

Deve-se substituir por:

A programação de uma tarefa complexa pode ser decomposta num conjunto de tarefas menores.

Página 275, Tabela 10.1: Substituir o título da Tabela 10.1 por:

Tabela 10.1 Blocos ativos e variáveis alocadas nos pontos da Figura 10.7

Página 275, Tabela 10.1: Substituir toda a Tabela 10.1 por:

Pontos	Blocos ativos	global	main		bb	ff	
		a	b	c	a	c	x
0	main	1	???	???	###	###	###
1	main	1	7	9	###	###	###
2	main - bb	1	7	9	???	###	###
3	main - bb	1	7	9	5	###	###
4	main – bb - ff	1	7	9	5	???	???
5	main – bb - ff	33	7	9	5	11	22
6	main	33	7	9	###	###	###

Legenda:

- variável desalocada;
??? – variável alocada com valor indefinido

Página 278, último parágrafo: Onde está escrito:

O resultado da execução é

```
Antes de ff, a = 5
Durante ff, a = 6
Depois de ff, a = 5
```

Deve-se substituir por:

O resultado da execução é

```
Antes de ff, a = 5
Durante ff, a = 6
Depois de ff, a = 5
```

Página 279, Exemplo10.7: Onde está escrito:

Após a execução de **trocar ( )**, os valores de **i** e **j** são efetivamente trocados, conforme os seguintes resultados:

```
Antes de trocar, i = 3; i = 8;
Depois de trocar, i = 8; i = 3;
```

Deve-se substituir por:

Após a execução de **trocar ( )**, os valores de **i** e **j** são efetivamente trocados, conforme os seguintes resultados:

```
Antes de trocar, i = 3; i = 8;
Depois de trocar, i = 8; i = 3;
```

Página 282, Função void main (): Onde está escrito:

```
/*    Leitura dos dois vetores                                    */

      printf ("Numero de elementos de V1: ");
      scanf ("%d", &m);
      printf ("Digite os %d elementos de V1\n\t", m);
      for (i = 0; i <= m-1; i++)
            scanf ("%d", &V1[i]);

  printf ("\nNumero de elementos de V2: ");
```

Deve-se substituir pelo código abaixo, observando a correta endentação:

```
/*    Leitura dos dois vetores                                    */

      printf ("Numero de elementos de V1: ");
      scanf ("%d", &m);
      printf ("Digite os %d elementos de V1\n\t", m);
      for (i = 0; i <= m-1; i++)
            scanf ("%d", &V1[i]);

      printf ("\nNumero de elementos de V2: ");
```

Página 283, 284 e 285, Figura 10.15: Alterar a endentação da Figura 10.15, conforme abaixo:

```
/*    Declaracoes globais      */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
typedef int matriz[10][10];

/*    Funcao EscreverMatriz: escrever a matriz argumento no video    */

void EscreverMatriz (matriz Mat, int m, int n) {
      int i, j;
      for (i = 1; i <= m; i++) {
            for (j = 1; j <= n; j++)
                  printf ("%5d", Mat[i][j]);
            printf ("\n");
      }
}


      /*    Funcao MultMat: multiplicar duas matrizes                */

void MultMat (matriz M1, int m1, int n1, matriz M2, int m2, int n2,
            matriz M3, int *m3, int *n3) {
      int i, j, k;
      *m3 = m1; *n3 = n2;
      if (m2 == n1) {
            for (i = 1; i <= (*m3); i++)
                  for (j = 1; j <= (*n3); j++) {
                        M3[i][j] = 0;
                        for (k = 1; k <= m2; k++)
                  M3[i][j] += M1[i][k]*M2[k][j];
                  }
      }
      else
            for (i = 1; i <= m1; i++)
                  for (j = 1; j <= n2; j++)
                        M3[i][j] = 0;
}

/*    Programa principal que le um expoente p e uma matriz M e calcula M elevada
      a p                                          */

void main () {
      int p, n, i, j, k; matriz M, Mpot, Maux;
      printf ("POTENCIACAO DE MATRIZ\n\n");

/*    Leitura do expoente e da dimensao da matriz base         */
```

```
     printf ("Digite o expoente: "); scanf ("%d", &p);
     while (p < 0) {
          printf ("\tExpoente < 0; Digite novamente: ");
          scanf ("%d", &p);
     }
     printf ("Digite a dimensao da matriz base: "); scanf ("%d", &n);
     while (n <= 0) {
          printf ("\tDimensao < 1; Digite novamente: ");
          scanf ("%d", &n);
     }



/*   Leitura e escrita da matriz base                          */

     printf ("\nElementos da matriz: \n");
     for (i = 1; i <= n; i++) {
          printf ("\tLinha %d: ", i);
          for (j = 1; j <= n; j++) scanf ("%d", &M[i][j]);
     }
     printf ("\nMatriz lida: \n\n"); EscreverMatriz (M, n, n);
     printf ("\nExpoente de potenciacao: %d\n", p);

/*   Inicializacao da matriz potencia com a matriz identidade */

     for (i = 1; i <= n; i++) {
          for (j = 1; j <= n; j++)
               (i == j) ?
                    (Mpot[i][j] = 1) : (Mpot[i][j] = 0);
     }

/* Calculo da matriz potencia          */

     for (k = 1; k <= p; k++) {
          MultMat (Mpot, n, n, M, n, n, Maux, &n, &n);
          for (i = 1; i <= n; i++)
               for (j = 1; j <= n; j++)
                    Mpot[i][j] = Maux[i][j];
     }

/* Escrita da matriz potencia          */

     printf ("\nMatriz potencia: \n\n"); EscreverMatriz (Mpot, n, n);
}
```

Página 286, último parágrafo: Onde está escrito:
Seção 8.2.2

Deve-se substituir por:
Seção 8.4.2

Página 288, Figura 10.17: Onde está escrito:

```
case '4':
      printf ("Divisao de complexos\n\n");
      printf("\tDigite os dois operandos: ");
      scanf ("%f%f%f%f", &a.real, &a.imag, &b.real, &b.imag);
      r = Divisao (a, b);
      printf ("\n[(%g)+j(%g)] / [(%g)+j(%g)] = [(%g)+j(%g)]",
            a.real, a.imag, b.real, b.imag, r.real, r.imag);
      break;
```

Deve-se substituir por:

```
case '4':
      printf ("Divisao de complexos\n\n");
      printf("\tDigite os dois operandos: ");
      scanf ("%f%f%f%f", &a.real, &a.imag, &b.real, &b.imag);
      if (b.real == 0 && b.imag == 0)
            printf ("\nO divisor eh zero!!!");
      else {
            r = Divisao (a, b);
            printf ("\n[(%g)+j(%g)] / [(%g)+j(%g)] = [(%g)+j(%g)]",
                  a.real, a.imag, b.real, b.imag, r.real, r.imag);
      }
      break;
```

Página 294, Figura 10.20: Onde está escrito:

```
void EscreverMatriz (matriz Mat) {- - - - - - - - - - - - - - - - - - -}
/*    Funcao MultMat: multiplicar as duas matrizes argumentos e retornar
      a matriz resultante         */
```

Deve-se substituir por:

```
void EscreverMatriz (matriz Mat) {- - - - - - - - - - - - - - - - - - -}

/*    Funcao MultMat: multiplicar as duas matrizes argumentos e retornar
      a matriz resultante         */
```

Página 295, 1º parágrafo após a Figura 10.21: Onde está escrito:

```
12

4  2  3
```

Deve-se substituir por:

```
12

 4  2  3
```

Página 298, Exercício 2, item a: Onde está escrito:

Escrever em uma função

Deve-se substituir por:

Escrever uma função

13 Correções no Capítulo 11 – Recursividade

Página 304, parágrafo logo após a Figura 11.2: Onde está escrito:

A função recursiva em C pode ser deduzida, então, com as informações acima e apresentada a seguir.

Deve-se substituir por:

A função recursiva em C pode ser deduzida, então, com as informações acima e apresentada na Figura 11.3.

Página 304, Exemplo 11.2: Onde está escrito:

A função iterativa foi apresentada no Exemplo 1.22.

Deve-se substituir por:

A função iterativa foi apresentada na Figura 1.22.

Página 305, Exemplo 11.3: Onde está escrito:

Exemplo 11.3: **Procriação de coelhos. Suponha que em um ambiente totalmente isolado haja inicialmente um coelho macho e um coelho fêmea, e que se queira saber qual a maior quantidade de pares que poderão existir ao final de um ano. Outras informações necessárias:**

Deve-se substituir por:

Exemplo 11.3: **Procriação de coelhos.**

Suponha que em um ambiente totalmente isolado haja inicialmente um coelho macho e um coelho fêmea, e que se queira saber qual a maior quantidade de pares que poderão existir ao final de um ano. Outras informações necessárias:

Página 312, 2º parágrafo: Onde está escrito:

O número *n* pode ser escrito como:

$$n = 1.2^9 + 0.2^8 + 1.2^7 + 0.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 = 1350$$

Deve-se substituir por:

O número *n* pode ser escrito como:

$$n = 1.2^{10} + 0.2^{9} + 1.2^{8} + 0.2^{7} + 1.2^{6} + 0.2^{5} + 0.2^{4} + 0.2^{3} + 1.2^{2} + 1.2^{1} + 0.2^{0} = 1350$$

Página 312, último parágrafo: Onde está escrito:

Considere a função RepBin dada na Figura 11.16, na qual a chamada do caso *n* é feita para o caso *n*/2 e o critério de parada ocorre com *n* = 0 ou *n* = 1.

Deve-se substituir por:

Considere a função RepBinRec dada na Figura 11.16, na qual a chamada do caso *n* é feita para o caso *n*/2 e o critério de parada ocorre com *n* = 0 ou *n* = 1.

Página 314, logo após a Figura 11.18: Onde está escrito:

Para os casos em que *n* = 3 e 4 têm-se que J(3) = 3 e J(4) = 1.

Deve-se substituir por:

Para os casos em que *n* = 3, 4 e 5, têm-se que J(3) = 3, J(4) = 1 e J(5) = 3, conforme as Figuras 11.19 e 11.20.

Página 315, parágrafo imediatamente antes de *Observação 2*: Onde está escrito:

J(10) = 2.31 = 5

Deve-se substituir por:

J(10) = 2*3 – 1 = 5

Página 317, Equação 11.5: Substituir a Equação 11.5 por:

$$\mathbf{Q(}\boldsymbol{n}\mathbf{)} = \frac{\boldsymbol{n}(\boldsymbol{n}+\mathbf{1})(\mathbf{2}\boldsymbol{n}+\mathbf{1})}{\mathbf{6}} \qquad \textbf{Equação 11.5}$$

Página 318, Equação 11.6: Substituir a Equação 11.6 por:

$$Q(1) = \frac{1.(1+1).(2.1+1)}{6} = \frac{1.2.3}{6} = 1, \qquad \text{Equação 11.6}$$

Página 318: Onde está escrito:

Mas, pela *hipótese indutiva*, Q(n) é dada pela Equação 11.6. Assim:

Deve-se substituir por:

Mas, pela *hipótese indutiva*, Q(n) é dada pela Equação 11.5. Assim:

Página 318, Equação 11.7: Substituir a Equação 11.7 por

$$\mathrm{Q}(n+1) = (n+1)^2 + Q(n) = (n+1)^2 + \frac{n(n+1)(2n+1)}{6} \qquad \text{Equação 11.7}$$

Página 318, parágrafo logo após a Equação 11.7: Onde está escrito:

Todavia, fazendo uma simples substituição de variáveis na Equação 12.5 de *n* para *n* + 1, isto é, onde ocorre *n* muda-se para *n* + 1, nem que:

Deve-se substituir por:

Todavia, fazendo uma simples substituição de variáveis na Equação 11.5 de *n* para *n* + 1, isto é, onde ocorre *n* muda-se para *n* + 1, tem-se que:

Página 318, parágrafo imediatamente antes do Exemplo 11.6: Onde está escrito:

Equação 11.6

Deve-se substituir por:

Equação 11.7

Página 319: Onde está escrito:

$$\boldsymbol{T(n) = \begin{cases} 1, & \text{para } n = 1, \\ 2T(n) + 1, & \text{para } n > 1 \end{cases}}$$

Deve-se substituir por:

$$T(n) = \begin{cases} 1, & \text{para } n = 1, \\ 2T(n) + 1, & \text{para } n > 1 \end{cases}$$ **Equação 11.8**

Página 319, 4º parágrafo: Onde está escrito:

Assume-se que T(n) = 2^n – 1, para todos os valores de n, desde 1. Para o caso em que se tem n+1 discos, pela Equação 11.7, T(n+1) = 2T(n) +1. Mas, pela *hipótese indutiva*, T(n) = 2^n-1; assim, chega-se a T(n+1) = 2(2^n-1)+1 = 2^{n+1} 2 + 1 = 2^{n+1}-1. ⊂

Deve-se substituir por:

Assume-se que T(n) = 2^n – 1, para todos os valores de n, desde 1. Para o caso em que se tem n+1 discos, pela Equação 11.8, T(n+1) = 2T(n) +1. Mas, pela *hipótese indutiva*, T(n) = 2^n-1; assim, chega-se a T(n+1) = 2(2^n-1)+1 = 2^{n+1} - 2 + 1 = 2^{n+1}-1. ⊂

Página 319: Eliminar Equação 11.8 de onde está escrito:

T(1)	=	1	⇒	$(T(1))_2$	=	$(1)_2$	=	1
T(2)	=	2.T(1)	⇒	$(T(2))_2$	=	$(10)_2$		
T(2)	=	T(2)+1	⇒	$(T(2))_2$	=	$(11)_2$	=	3
T(3)	=	2.T(2)	⇒	$(T(3))_2$	=	$(110)_2$		
T(3)	=	T(3)+1	⇒	$(T(3))_2$	=	$(111)_2$	=	7
T(4)	=	2T(3)	⇒	$(T(4))_2$	=	$(1110)_2$		
T(4)	=	T(4)+1	⇒	$(T(4))_2$	=	$(1111)_2$	=	15
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T(n)	=	2T(n-1)	⇒	$(T(n))_2$	=	$(11 \propto 10)_2$		
T(n)	=	2T(n-1)	⇒	$(T(n))_2$	=	$(11 \propto 11)_2$	=	2^n-1

Equação 11.8

Página 320, 1º parágrafo: Onde está escrito:

Demonstração: A função recursiva apresentada na Figura 11.23 .....

Deve-se substituir por:

Demonstração: A função recursiva apresentada na Figura 11.22 .....

Página 321: Onde está escrito:

Agora, pela equação 11.8, $J(59) = J(2^5+27) = 2.27+1 = 55$.

Deve-se substituir por:

Agora, pela equação 11.9, $J(59) = J(2^5+27) = 2.27+1 = 55$.

Página 322: Onde está escrito:

A função retorna ou o índice em que *valor* aparece no vetor ou, então, 1, que indica que a busca não teve sucesso.

Deve-se substituir por:

A função acima retorna ou o índice em que *valor* aparece no vetor ou, então, -1, que indica que a busca não teve sucesso.

Página 324, último parágrafo: Onde está escrito:

O Exercício 15 pede .....

Deve-se substituir por:

O Exercício 14 pede .....

Página 325, último parágrafo: Onde está escrito:

Figura 11.28

Deve-se substituir por:

Figura 11.29

Página 330, 1º parágrafo: Onde está escrito:

Em uma delas, deve ser ordenado um vetor com os índices de *esquerda* até *pivô* 1, e outro .....

Deve-se substituir por:

Em uma delas, deve ser ordenado um vetor com os índices de *esquerda* até *pivô* - 1, e outro .....

Página 332, Exercício 18: Substituir toda a fórmula de **Det** por:

Det (A, n) =

$$\begin{cases} A[0,0], \text{ se } n = 1 \\ A[0,0]*A[1,1] - A[0,1]*A[1,0], \text{se } n = 2 \\ A[0,0]*A[1,1]*A[2,2] + A[0,1]*A[1,2]*A[2,0] + \\ A[0,2]*A[1,0]*A[2,1] - A[0,1]*A[1,0]*A[2,2] - \\ A[0,0]*A[1,2]*A[2,1] - A[0,2]*A[1,1]*A[2,0], \text{se } n = 3 \text{ (Regra de Sarrus)} \\ \sum_{i=0}^{n-1} (-1)^i * A[0,i] * \text{Det (MenorComplementar (A,0,i), n - 1), se } n > 3 \text{ (Regra de Laplace)} \end{cases}$$

Página 333, Exercício 19: Onde está escrito:

n+1, sem = 0

Deve-se substituir por:

n+1, se m = 0

14 Correções no Capítulo 12 – Metodologia *Top-Down* Auxiliada por Subprogramação

Página 348, Figura 12.13: Alterar a endentação da Figura 12.13 conforme abaixo:

```
Pivotar (sistema *Sist, result *Res, int i) {
      int j, indpivo; double pivo;
      pivo = fabs (Sist->Mat[i][i]); indpivo = i;
      for (j = i+1; j <= Sist->n; j++)
            if (fabs (Sist->Mat[j][i]) > pivo){
                  pivo = fabs (Sist->Mat[j][i]);
                  indpivo = j;
            }
      if (pivo == 0) Res->temsolucao = FALSE;
      else if (indpivo != i)
            Trocar (Sist, i, indpivo);
}
```

Página 352, Função EscreverSolucao: Alterar a endentação conforme abaixo:

```
/*    Funcao EscreverSolucao: escreve o valor calculado para as incógnitas ou
      informa que o sistema nao tem solucao   */

      void EscreverSolucao (result *Result) {
            int i;
            if (Result->temsolucao == TRUE) {
                  printf ("\nSolucao: \n\n");
                  for (i = 1; i <= Result->n; i++)
                        printf ("X[%2d] = %10g \n", i, Result->X[i]);
            }
            else printf ("\nSistema sem solucao\n");
}
```

Página 353, Função CalcularIncognitas: Alterar a endentação conforme abaixo:

```
/*    Funcao CalcularIncognitas: uma vez triangularizado o sistema,
      calcula o valor de todas as suas incognitas              */

void CalcularIncognitas (sistema *Sist, result *Res) {
      int i, j;
      for (i = Sist->n; i >= 1; i--) {
            Res->X[i] = Sist->Vet[i];
            for (j = i+1; j <= Sist->n; j++)
                  Res->X[i] -= Sist->Mat[i][j]*Res->X[j];
            Res->X[i] /= Sist->Mat[i][i];
      }
}
```

Página 361, Figura 12.23: Alterar a endentação da Figura 12.23 conforme abaixo:

```
átomo NovoÁtomo () {
      átomo Atom;
      while (Expressão[cursor] == ' ') cursor++;
      if (isdigit (Expressão[cursor])) {
            cursor++;
            while (isdigit (Expressão[cursor])) cursor++;
            if (Expressão[cursor] == '.') cursor++;
            while (isdigit (Expressão[cursor])) cursor++;
            Atom.tipo = num;
      }
      else {
            switch (Expressão[cursor]) {
                  case '+': case '-': Atom.tipo = opad; break;
                  case '*': case '/': Atom.tipo = opmult; break;
                  case '~': Atom.tipo = menun; break;
                  case '(': Atom.tipo = abpar; break;
                  case ')': Atom.tipo = fpar; break;
                  case '$': Atom.tipo = dolar; break;
                  default: Atom.tipo = inval;
            }
            cursor++;
      }
      return Atom;
}
```

Página 364, Figura 12.28, e página 367, Figura 12.29: Alterar a endentação conforme abaixo:

```
        else {
              switch (Expressão[cursor]) {
                    case '+': case '-':
                          Atom.tipo = opad;
                          Atom.atributo.carac = Expressão[cursor];
                          break;
                    case '*': case '/':
                          Atom.tipo = opmult;
                          Atom.atributo.carac = Expressão[cursor];
                          break;
                    case '~': Atom.tipo = menun; break;
                    case '(': Atom.tipo = abpar; break;
                    case ')': Atom.tipo = fpar; break;
                    case '$': Atom.tipo = dolar; break;
                    default:
                          Atom.tipo = inval;
                          Atom.atributo.carac = Expressão[cursor];
              }
              cursor++;
        }
        return Atom;
  }
```

15 Correções no Capítulo 13 – Noções de Estruturas de Dados

Página 377, Figura 13.13: Eliminar a seguinte frase:

Final da Fila

Página 381, Função main: Alterar a endentação da função main conforme abaixo:

```
/* Funcao main: Calcula o valor da expressao lida em polonesa */

void main () {
      int num, val;

/*    Leitura da expressao, inicializacao da pilha e preparo para comecar o
percurso na expressao  */

      printf ("Digite a expressao:\n\n\t");
      gets (expr); InicPilha (&P); erro = 0; InicExpr ();

      /*    Processo repetitivo percorrendo a expressão, empilhando os operandos
      e executando as operacoes aritmeticas                              */

      while (1) {
            c = PegaNaoBranco ();
            if (c == '\0') break;

/*    Formacao e empilhamento de um operando       */

            if (isdigit (c)) {
                  num = FormarNumero ();
                  Empilhar (num, &P);
            }

      /*    Execucao de uma operacao aritmética desempilhando dois operandos e
      empilhando o resultado; tambem sinaliza casos de erros na expressão */

            else if (c == '+' || c == '*') {
                  ExecutarOperacao ();
                  if (erro == 1) break;
                  c = ProxCarac ();
            }
            else {erro = 1; break;}
      }

/* Final do processo repetivo com teste de erro e escrita do resultado */

      if (! erro) {
            val = Topo (P); Desempilhar (&P);
```

```
            if (! Vazia (P)) erro = 1;
      }
      if (erro) printf ("\n\tErro na expressao!");
      else printf ("\n\tValor da expressao = %d", val);
}
```

Página 382, Figura 13.17: Alterar a endentação de um trecho da Figura 13.17, conforme abaixo:

```
/* Funcao para executar operacao aritmética detectando casos de erro na
expressao    */

void ExecutarOperacao () {
      int val;
      if (Vazia (P)) {erro = 1; return;}
      val = Topo (P); Desempilhar (&P);
      if (Vazia (P)) {erro = 1; return;}
      if (c == '+') val += Topo (P);
      else val *= Topo (P);
      Desempilhar (&P); Empilhar (val, &P);
}

/* Funcoes para operacoes em pilhas  */

void Empilhar (int x, pilha *P) {
      noh *temp;
      temp = *P;  *P = (noh *) malloc (sizeof (noh));
      (*P)->elem = x; (*P)->prox = temp;
}

void Desempilhar (pilha *P) {
      noh *temp;
      if (! Vazia(*P)) {temp = *P; *P = (*P)->prox; free (temp); }
}

int Topo (pilha P) {if (! Vazia(P)) return P->elem; else return -1;}

void InicPilha (pilha *P) {*P = NULL;}

char Vazia (pilha P) {if (P == NULL) return 1; else return 0; }

/* Funcoes para percorrer a expressao digitada  */

char PegaNaoBranco () {
      while (isspace (expr[i]) || (iscntrl (expr[i]) && expr[i] != '\0'))
            i++;
      return expr[i];
}
```

Página 384, último parágrafo: Onde está escrito:

struct fila {noh fr, tr;};

Deve-se substituir por:

struct fila {noh *fr, *tr;};

Página 387, penúltimo parágrafo: Onde está escrito:

Na figura anterior, a raiz é o nó **A**.

Deve-se substituir por:

Na referida figura, a raiz é o nó **A**.

Página 392, parágrafo após a Figura 13.24: Onde está escrito

```
struct arvore {celula Espaco; int numnos;}
```

Deve-se substituir por:

```
struct arvore {celula Espaco[max]; int numnos;}
```

Página 395, 1º parágrafo: Onde está escrito:

O subprograma de procedimento para fazer essa listagem tem .....

Deve-se substituir por:

O subprograma procedimento para fazer essa listagem tem .....

Página 395, 2º parágrafo: Onde está escrito:

Cada vez que o cursor vai para a próxima linha do vídeo, o conteúdo de **fp** passa para **fs** e essa última é reinicializada.

Deve-se substituir por:

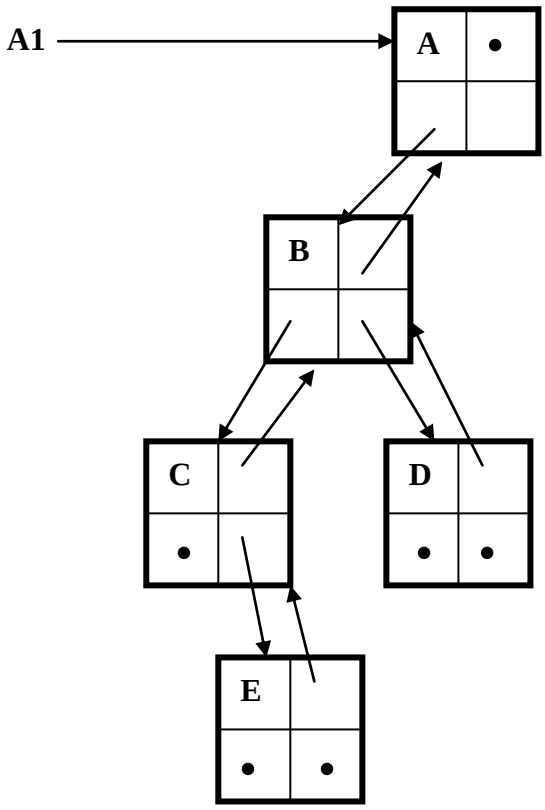
Cada vez que o cursor vai para a próxima linha do vídeo, **fp** recebe o conteúdo de **fs** e essa última é reinicializada.

Página 397: Alterar a endentação da função EscreverArvore, conforme abaixo:

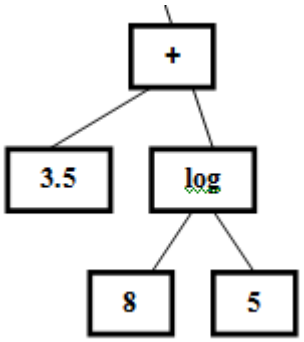
```
/*    Funcao EscreverArvore: escreve os nos da arvore em ordem de nível */

void EscreverArvore (arvore A) {
      fila fp, fs; noh x, y;
      if (A == NULL) printf ("  Arvore vazia");
      else {
            InicFila (&fs); EntrarFila (A, &fs);
            do {
                  fp = fs;
                  InicFila (&fs);
                  while (FilaVazia(fp) == FALSE) {
                        x = RemoverFila(&fp); printf ("  %c(", x->info);
                        if (x->pai == NULL) printf ("#)");
                        else printf ("%c)", x->pai->info);
                        for (y = x->fesq; y != NULL; y = y->idir)
                              EntrarFila (y, &fs);
                  }
                  printf ("\n");
            } while (FilaVazia(fs) == FALSE);
      }
   printf ("\n\n");
}
```

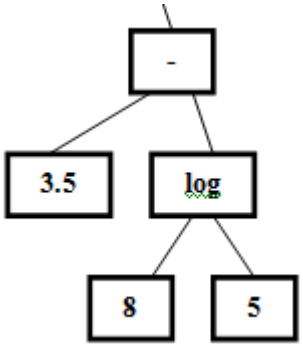
Página 399, Figura 13.31: Alterar toda a Figura 13.31 conforme abaixo:



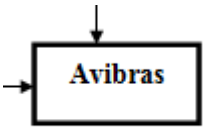
Página 400, Figura 13.32: Onde está desenhado:



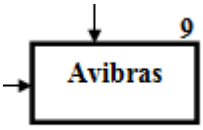
Deve-se substituir por:



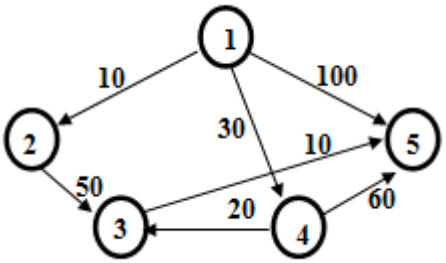
Página 403, Figura 13.35: Onde está desenhado:



Deve-se substituir por:



Página 405, Figura 13.38: Substituir a Figura 13.38 por:



16 Correções no Capítulo 14 – Manipulação de Arquivos

Página 423, 1º parágrafo da Seção 14.5.1: Onde está escrito:

As funções **fread ( )** e **fwrite ( )** deixam os cursores dos arquivos por ela manipulados .....

Deve-se substituir por:

As funções **fread ( )** e **fwrite ( )** deixam os cursores dos arquivos por elas manipulados .....

Página 424, 1º parágrafo da Seção 14.5.2: Onde está escrito:

arqele.c

Deve-se substituir por:

arqelet.c

16 Correções na Bibliografia

Página 429: Acrescentar à Bibliografia o seguinte livro:

Saliba, W. L. C. *Técnicas de Programação: Uma Abordagem Algorítmica*. Makron,1992.

Página 429: Onde está escrito:

36. Yarasse, H.; Arenales, M.; Morabito, R.; Armentano, V. Pesquisa Operacional para cursos de Engenharia. Rio de Janeiro: Campus/Elsevier, 2006.

Deve-se substituir por:

36. Yanasse, H.; H.; Arenales, M.; Morabito, R.; Armentano, V. *Pesquisa Operacional para Cursos de Engenharia*. Rio de Janeiro: Campus/Elsevier, 2006.

17 Conclusões e Observações

De acordo com os objetivos dos autores deste documento, foram apresentados vários pontos a serem corrigidos do livro de sua autoria e, ao lado de cada um deles, foram fornecidos textos substitutivos. Apesar de muito zelo e técnica na sua edição, não poucos erros de digitação e de impressão foram encontrados, de forma a tornar necessária uma edição revisada do livro.

Há ainda probabilidade de erros que escaparam da revisão aqui relatada, por isso os autores pedem aos leitores que lhes escrevam caso encontrem alguns desses erros.

Referências Bibliográficas

1. MOKARZEL, Fábio Carneiro e SOMA, Nei Yoshihiro. *Introdução à Ciência da Computação*. Rio de Janeiro: Campus-Elsevier, 2008. 429p.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO N°	4. N° DE PÁGINAS
MT	18 de fevereiro de 2013	DCTA/ITA/MT-003/2013	101

5. TÍTULO E SUBTÍTULO:

Guia para Leitura Correta do Livro Introdução à Ciência da Computação.

6. AUTOR(ES):

Nei Yoshihiro Soma; Fábio Carneiro Mokarzel

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELOS AUTORES:

Ciência da Computação, Programação de Computadores, Linguagem C, Leitura correta.

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Programas de computadores; Linguagens de programação; C (Linguagem de programação); Computação.

10. APRESENTAÇÃO: **X Nacional** **Internacional**

Divisão de Ciência da Computação; São José dos Campos, SP, Brasil; Janeiro, 2013; Instruções para Leitura Correta do Livro Introdução à Ciência da Computação.

11. RESUMO:

Este documento apresenta instruções para a leitura correta do livro “Introdução à Ciência da Computação” da autoria dos professores Nei Yoshihiro Soma e Fábio Carneiro Mokarzel. Apesar de muito zelo e técnica empregados na edição desse livro, ocorreram erros de digitação e de impressão.

Com o intuito de orientar os leitores do livro a utilizar corretamente os conhecimentos ali transmitidos, os autores decidiram publicar este guia indicando os pontos a serem corrigidos e fornecendo os textos substitutivos.

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) CONFIDENCIAL** **( ) SECRETO**