教養の数学・計算機 正誤表 (1997.11.1 版)

p.8, 上から 9 行目: 例えば ⟶ 例えば，全ファイルを表すワイルド・カード *.* を用いて

p.12 上から 4 行目: して使用する ⟶ しそのドライヴから立ち上げる

p.12 上から 5 行目: A:ファイル名 ⟶ ファイル名

p.13 上から 1 行目: A:SPE ⟶ SPE

p.29 下から 16 行目: irregal ⟶ illegal

p.31 下から 2 行目: 21 4748 6647 ⟶ 21 4748 3647

p.37 上から 16 行目: でプリンターの… 確保し
⟶ という宣言文によりプリンターを割り当てるべきファイルを定義し

p.37 下から 10 行目: `fclose("prn")` ⟶ `fclose(file)`

p.62 上から 5 行目: 文字列は ⟶ 文字列には二種類あり，一つは

p.62 上から 7 行目: のように… するときは
⟶ のように宣言するもので，次のように値を一回だけ代入することができる:

p.62 上から 7 行目: “ とする．”を削除

p.62 上から 11〜12 行目: また，文字列… も変化する．
⟶ 末尾には終了を表すコード 0 が追加され，そこまでか moji の実際の内容となる．

p.62 下から 9 行目: 次を追加:

上記の文字列については再代入による内容の変更操作等が保証されないので，むしろ文字定数のように取り扱った方が安全である．文字列操作のためには，文字配列と呼ばれるもう一つの文字型変数を用いるのが良い．これは

```
char moji[100]
```

のように宣言して用いるもので，ほぼ FORTRAN の固定長文字列 `CHARACTER MOJI*100` と同じと思って良いが，この場合も文字列の最後に終了コード 0 が追加されたものが格納されるので，実際に必要な長さよりも一つ長く宣言しておかねばならない．この変数に対する種々の文字操作は標準ライブラリー関数により行われる．

p.62 下から 8〜4 行目: 上に述べた… これらの言語においては
⟶ 上に述べた種々の言語系での文字列処理のなかでは，ひとり BASIC だけが(長さ 255 という制限は有るものの）真の意味での不定長文字列の取り扱いをしており，文字変数の実際の内容は文字列操作の度に新たな主記憶の空き領域を求めて移動してゆく.(アドレス函数で調べた文字変数自身の位置は不動であるが，そこにはこの実際の文字列内容の存在場所の先頭アドレス(ポインター）と文字列の長さとが保持されている.）この種の言語においては

p.72 下から 1 行目: 第2 項の符号を –，第4 項の符号を +に訂正

p.73 上から 1 行目: 第2 項の符号を –，第4 項の符号を +にし係数の分子 2 を 1 にする

p.89 上から 9 行目: (-1) の冪を $n-1$ に修正

p.91 上から 14 行目: 一桁 ⟶ オーダー一つ分

p.100 下から 13 行目: 総和記号の前に 2 を挿入する

p.100 下から 12 行目: 分子に $(b-a)$ を追加

p.100 下から 11 行目: 分母の $k!$ を $(2k)!$ にする； 分子に $(b-a)^{2k}$ を追加

p.100 下から 5 行目: $j=k,...,4, k=0,1,2,3,4$ ⟶ $k=j,...,4, j=1,2,3,4$

p.110 最後に追加:

注．二次元配列，例えば A(2,3) の要素は主記憶内では

```
A(1,1), A(2,1), A(1,2), A(2,2), A(1,3), A(2,3)
```

の如く一次元化され，詰めて格納されている．故に配列を引数に持つ副プログラムを呼ぶとき，主プログラムと引数の寸法が異なると値が正しく渡されない．

p.185 下から 9〜11 行目： 実際に (21.4) を … 面積の総和

⟶ 一般に上例のようなパラメータ空間の規則正しい分割に対して (21.6) の総和

p.185 下から 2 行目〜p.186 上から 2 行目： 三角形毎には … のみである．

⟶ 一般の三角形分割でも三角形毎には近似のオーダーが $O(h^4)$ となることが容易にわかる．従って三角形の個数が $O(1/h^2)$ であるような分割では全体の誤差は上と同様 $O(h^2)$ となる．しかし，このことは一般には期待できない．

p.211 下から 6 行目： 約三万個 ⟶ 約三十万個

p.211 下から 4 行目： 27786 個 ⟶ 363452 個

p.211 下から 1 行目： この数字の半分程度に ⟶ 後者の数字くらいに

p.212 上から 1 行目に追加:（ 最近，高速の PC が何台か使えるようになったので，半年くらいかけて計算したところ，ぴったり後者の数になった.）

p.241 下から 2 行目： 次を追加

CTRL+\O 欧文／和文相の切り替え（ 漢字を色付き 1 バイト文字 2 個／漢字 1 個として表現）

p.249 末尾に次を追加:

【 リンク時のエラー・メッセージ】

Unsatisfied external(s): （ この後にサブルーチン名が並ぶ）

主プログラムから呼ばれているサブルーチンがリンクを指定されたファイルたちの中に見つからない．このエラーは主に以下の場合に生じる：

① 実際に自分で作ったサブルーチンプログラムをリンクし忘れている．

② 主プログラムの CALL 文で綴りを間違えたために有りもしないサブルーチンを呼んだことになった．

③ コンパイルとリンクでメモリーモデルの統一がとれていない．この場合は@で始まる奇妙な名前が並ぶのでそれと判定できる．

p.251 上から 14 行目： 0 〜 3 ⟶ 0 〜 15

p.251 上から 15 行目： `0 < 1 < 2 < 3` ⟶ `0 < 1 < ... < 15`

p.253 上から 11 行目： ≦ 24 ⟶ ≦ 80

p.258 上から 18 行目： 主プログラムの開始 ⟶ 主プログラムの実行の開始

p.262 上から 32 行目： “EXTERNAL EPTEND” を削除

p.266 上から 11 行目： [1] 石田晴久『 C プログラミング』を次と入れ替える (あいうえお順は狂うが，引用番号が狂わないよう同じ場所に置く)：

[1] カーニハン=リッチー『 プログラミング言語C 』第二版，共立出版

p.277 上から 14,16 行目： `IX,IY,1,IC)` ⟶ `IX,IY,IC,1)`