11　プロシージャ

付録の xxii–xxiii ページに，前回の課題 BunsuuNoKeisan のプログラムが載っています。四則を行うので，約分する操作—分子と分母の最大公約数を求めて，それぞれを割る—が何箇所もあります。この操作を関数のような副プログラムとして定義して使うと便利だと思うのですが，戻す値が約分した後の分子と分母の 2 つあるため，関数として定義することはできません。関数は戻す値が 1 つだけある操作を定義するための副プログラムなのです。

戻す値が 1 つでない（1 つもない，あるいは 2 つ以上ある）操作を定義する副プログラムは，**プロシージャ（procedure，手続き）**と言って次のようなものです。

11.1　約分をするプロシージャ

```
procedure Yakubun(Si1,Bo1 : Integer; var Si2,Bo2 : Integer);
          { 約分 (reduce)              }
          { Si1/Bo1 =約分⇒ Si2/Bo2 }
  var
    D : Integer;
  begin
    D := Gcd(Si1,Bo1);
    Si2 := Si1 div D;
    Bo2 := Bo1 div D;
  end; {Yakubun}
```

ヘッダー（procedure で始まっている行）の括弧の中に書いてある変数 Si1,Bo1,Si2,Bo2 を**引数**（ひきすう）と言います。前に，var が書いてある引数 Si2,Bo2 を**変数引数**（へんすう），var が書いてない引数 Si1,Bo1 を**値**（あたい）**引数**と言います。

プロシージャ Yakubun を使うとき，値引数の Si1,Bo1 に対して，定数，すでに値が代入されている変数，さらには式も書くことができます。一方，変数引数の Si2,Bo2 に対しては，変数しか書くことができません。

プロシージャの中で引数に値を代入したとき，値引数と変数引数とで影響が異なります。

- 値引数の場合，使用元の変数は変更されません。
- 変数引数の場合，使用元の変数にその値が代入されます。したがって，値を戻したいときに変数引数を使います。

例 11.1

```
Yakubun(105,140,Bunsi,Bunbo);            // Bunsi=3, Bunbo=4 になる

Bunsi := 105;
Bunbo := 140;
Yakubun(Bunsi,Bunbo,Bunsi,Bunbo);        // Bunsi,Bunbo が約分されて，3,4 に変わる

Bo3 := Bo1*(Bo2 div Gcd(Bo1,Bo2));
Yakubun(Si1*(Bo3 div Bo1)+Si2*(Bo3 div Bo2),Bo3,Si3,Bo3);  // 足し算
```

プロシージャや関数を使うと，同じ処理を何回も書かなくて済むという利点の他に，プログラムがわかりやすくなるという非常に大きな利点があります。

例 11.2 (掛け算) 分数の掛け算は，小学校で習った方法を，そのまま書けばいいのです。

```
Yakubun(Bunsi1,Bunbo2);        // 分子 1 と分母 2 を約分して
Yakubun(Bunsi2,Bunbo1);        // 分子 2 と分母 1 を約分して
Bunsi3 := Bunsi1*Bunsi2;       // 分子同士
Bunbo3 := Bunbo1*Bunbo2;       // 分母同士を掛ける
```

足し算，引き算で，2 つの分母の最小公倍数を共通の分母としますが，これをわかりやすくするために，最小公倍数（least common multiple）も関数として定義するとよいでしょう。

分数の四則はもちろん，分数を読む，分数を書く操作もプロシージャにするとよいでしょう。

そのように，改良したプログラムを付録の xxiv–xxvi に載せました。プログラム全体は大きくなりますが，メインプログラムは小さくなり，わかりやすくなったでしょう。

11.2　モジュール化

このように，関数やプロシージャ（合わせて，副プログラム，モジュールなどという）に分割してプログラムをわかりやすくすることを**モジュール化**と言います。

メインプログラム，各モジュールの実行部がせいぜい 2,30 行，1 画面に納まる程度になるように細分すると，わかりやすい良いプログラムができます。

11.3　ルート整理　Root2.dpr

第 8 回の課題プログラム Root.dpr を次のようにモジュール化しなさい。（プログラムを変更する前に，［プロジェクトに名前を付けて保存］をしてプログラム名を変えること。）

```
procedure Root(N : Integer; var P,Q : Integer);
          { √n ＝整形⇒ p√q                                    }
          { ただし，n＞0，qは1より大きい平方数で割り切れない }
```

を定義して用いるようにする。

この中では，出力（Write 文）は行いません。出力はメインで行います。