計算機数学
–Pascal プログラミング–

上越教育大学 数学教室

はじめに

この講義の目的は、Pascal 言語によるプログラム作成を通して、パーソナルコンピュータの操作法と、最も基本的なソフトウェアとしてのエディターの使用法を習得し、構造化プログラミングの方法を学ぶことです。例題は素朴な数学的問題ばかりで、表計算ソフトウェアを用いれば簡単に解けてしまうかもしれませんが、小さなプログラムを自分の頭で考えて手作りすることを通して、ソフトウェアがどうやってできているかを理解してください。電卓を使えば四則演算が簡単にできるからといって、小学校の算数で四則演算を学ぶ必要がなくならないように、高機能で使い易いアプリケーションソフトウェアがあるからといって、プログラミングを学ぶ必要がなくなることはありません。

Pascal コンパイラーはフリーソフトウェアである Free Pascal を使用します。また、パスカルプログラム作成の一連の作業を効率良くするための機能を追加したエディタ CPad for Pascal を使用します。Microsoft Windows は現在最も普及しているパーソナルコンピュータ用の OS です。将来において他の様々なシステムを使用する場合でも、これらのシステムで学んだことはプラスになるでしょう。情報化社会へと向かう現代において、計算機の利用環境は今後ますます激しく変化していくと思われます。学生諸君にとって、この講義で習得したことが将来にわたって計算機を理解して利用していくための確かな基盤となることを期待します。

2010 年 10 月

中川仁

目 次

1　Pascal 入門

1.1　最初のプログラム

画面に'Hello.' と表示させるだけのプログラムを Pascal で書くと次のようになります。

例 1.1. 'Hello.' を表示。

```
program Lesson1;
begin
   writeln('Hello.');
end.
```

プログラムの解説。

1 行目　Pascal のプログラムは、**program** プログラム名; で始まる。文の終わりには区切り記号;(セミコロン) をつける。

2 行目　プログラム本体 (実行文部) の始めには、**begin** と書く。

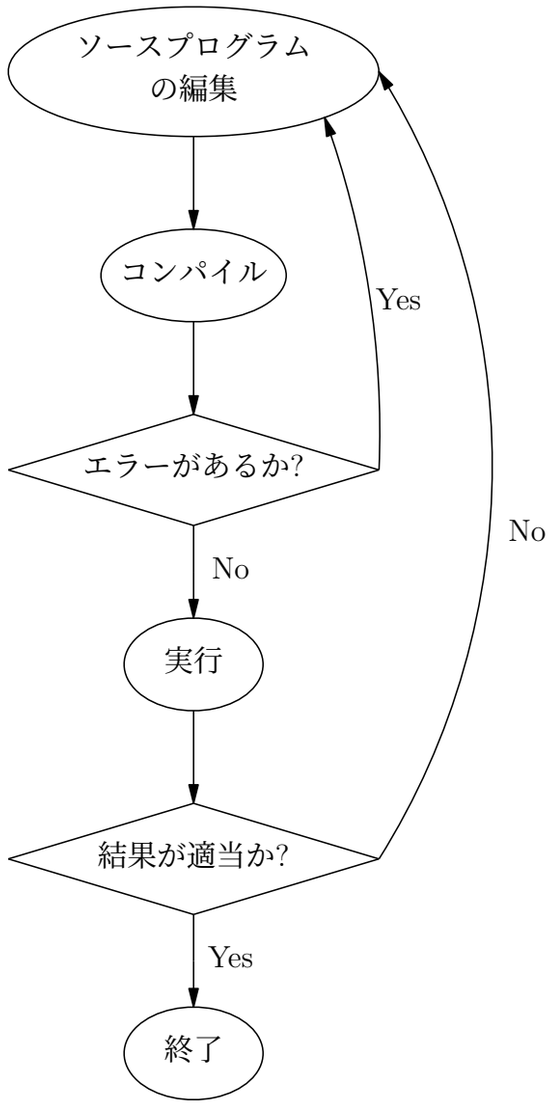
3 行目　'Hello.' というメッセージを画面に表示させる。

4 行目　プログラム本体 (実行文部) の終わりには、**end.** と書く。

1.2　プログラムの編集とコンパイル

上のような内容のテキストファイルをエディターと呼ばれるテキストファイルを作成・編集するためのソフトウェアを用いて作成し、それを Pascal コンパイラーと呼ばれるソフトウェアで処理すると実際に画面に'Hello.' と表示させる実行形式のファイルができます。

作業過程を図示すれば、次の図のようになります。



上の過程を、メニューを選択するだけで簡単に行えるようにするプログラム、PCPad.exe を用意しました。

1.3　プログラムの基本型

2つの整数aとbを入力し、これらの和を求めるプログラムをPascalで書くと次のようになります。

例 1.2. 2つの整数の和。

```
program Lesson2;
var a,b,wa:integer;
begin
   write('a= ');
   readln(a);
   write('b= ');
   readln(b);
   wa:=a+b;
   writeln('a+b= ', wa);
end.
```

プログラムの解説。

1行目 Pascalのプログラムは'**program** プログラム名;'で始まる。文の終わりには区切り記号**;**(セミコロン)をつけなければならない。英字の大文字と小文字は区別しない。

2行目 使用する変数名はすべて前もって宣言しなければならない。ここでは、変数a, b, waを整数型の変数として使用することを宣言している。

3行目 プログラム本体(実行文部)の始めには、**begin**と書く。

4行目 'a= 'というメッセージを画面に表示する。**write**では、カーソルはメッセージの右端に移り、**writeln**では、カーソルはメッセージの次行の左端に移る。

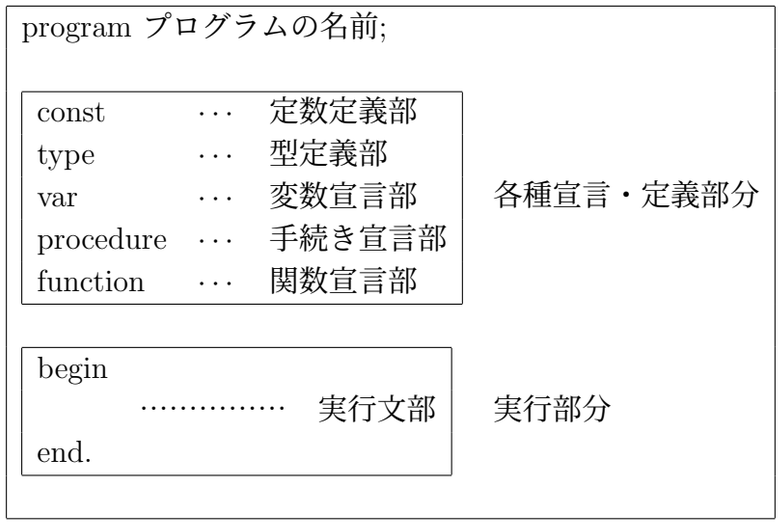
5行目 キーボードからの入力を変数aに読み込む。

8行目 a+bの結果を変数waに代入する。代入(左辺に右辺を代入)は=ではなく、:=を用いる。

9行目 'a+b= 'という文字列と変数waの値を表示する。

10行目 プログラム本体(実行文部)の終わりには、**end.**と書く。

Pascalの一般的なプログラムの構造は次のようになります。



定数定義部 定数定義部は次のような構造をしています。

```
const <名前>=<定数>;
      ·····
      <名前>=<定数>;
```

定数は、数、または文字列であり、プログラムを見やすくします。例えば、プログラム中で何回も繰り返し用いられる定数は、定数として定義しておけば、変更する場合にこの部分の1行だけを変更するだけで済みます。

例 1.3. n=10、v=2.71×10^{-12}、name='太郎'と定数を定義する。

```
const n=10;
      v=2.71E-12;
      name='太郎';
```

型定義部 Pascalでは、すべての変数は型を持っています。さらに、新しい型を定義することができます。それによって、複雑な構造のデータの処理を簡明に記述できます。

型定義部は次のような構造をしています。

```
type   <型名>=<型>;
       ..................
       <型名>=<型>;
```

例 1.4. 3 個の実数の配列を vector という名前の型として定義する。

```
type  vector=array[1..3] of real;
```

変数宣言部 プログラム中 (実行文部中で) 使用される変数は、すべて変数宣言しなければなりません。変数宣言部は次のような構造をしています。

```
var   <変数名>, …, <変数名>:<型>;
      ..................................
      <変数名>, …, <変数名>:<型>;
```

例 1.5. a,b を整数型の変数、x,y,r を実数型の変数、ans を文字型の変数、name を長さ 30 以内の文字列型の変数として宣言する。

```
var  a,b:integer;
     x,y,r:real;
     ans:char;
     name:string[30];
```

手続き宣言部 Pascal で扱われる手続きは BASIC のサブルーチンに相当し、ある一定の手順が決まったまとまりのある仕事や、プログラム中で繰り返し行われる処理を記述することに用いられます。

```
procedure <手続き名>(仮引数部>, …, <仮引数部>);
```

例 1.6. a と b を交換する手続き。

```
procedure exchg(var a,b:integer);
  var c:integer;
  begin
        c:=a; a:=b; b:=c;
  end;
```

関数宣言部 Pascal では、sin などの標準で組み込まれている関数以外にも、新しい関数を定義して利用することができます。

```
function <関数名>(仮引数部>, …, <仮引数部>):<結果型>;
<関数ブロック>
```

例 1.7. a と b の最大値を与える関数。

```
function max(a,b:real):real;
begin
      if a>b then max:=a
      else max:=b;
end;
```

実行文部 BASIC や FORTRAN のメインプログラムに相当します。**begin** で始まり **end.** で終わります。

2　変数の型、演算、配列

2.1　整数型

Pascal では、次の 5 つの型が**整数型**として用意されています。

```
byte                0 から 255          までの整数
shortint         -128 から 127          までの整数
integer        -32768 から 32767        までの整数
word                0 から 65535        までの整数
longint  -2147483648 から 2147483647 までの整数
```

また、整数型の定数として、次の 2 つが用意されています。

```
MaxInt=32767
MaxLongInt=2147483647
```

例 2.1. 2 つの整数 a, b の和、差、積、商、余りを求める。

```
program Lesson3;
var a,b,wa,sa,seki,shou,amari:integer;
begin
   write('a= '); readln(a);
   write('b= '); readln(b);
   wa:=a+b;
   sa:=a-b;
   seki:=a*b;
   shou:=a div b;
   amari:=a mod b;
   writeln('a+b= ', wa);
   writeln('a-b= ', sa);
   writeln('a*b= ', seki);
   writeln('a div b= ', shou);
   writeln('a mod b= ', amari);
end.
```

プログラムの解説。

2 行目　変数 a, b, wa, seki, shou, amari を整数型の変数として使用することを宣言する。

6 行目　a+b の結果を変数 wa に代入する。

7 行目　a-b の結果を変数 sa に代入する。

8 行目　a*b の結果を変数 seki に代入する。

9 行目　a div b の結果を変数 shou に代入する。

10 行目　a mod b の結果を変数 amari に代入する。

11 行目　'a+b = ' という文字列と変数 wa の値を表示する。

12 行目　'a-b = ' という文字列と変数 sa の値を表示する。

13 行目　'a*b = ' という文字列と変数 seki の値を表示する。

14 行目　'a div b = ' という文字列と変数 shou の値を表示する。

15 行目　'a mod b = ' という文字列と変数 amari の値を表示する。

2.2　実数型

Pascal では、次の型が**実数型**として用意されています。

real　2.9×10^{-39} から 1.7×10^{38} までの実数 (絶対値)、有効数字 11 桁。

実数型データの和・差・積の計算には、整数型データのときと同様に、+、−、* を用います。また、商を計算するには、/ を用います。整数型から実数型への変換はそのまま型変換できますが、実数型から整数型への変換は、次のような関数を利用しなければなりません:

`trunc(x)`　実数 x の小数部分を切り捨てて整数型に変換する;
`round(x)`　実数 x の小数部分を四捨五入して整数型に変換する。

例 2.2. 時速 v と距離 x が与えられたときの所用時間の計算。

```
program Lesson4;
var v, x:real;
    h, m:integer;
begin
  write('時速？ ');
  readln(v);
  write('距離 (km)？ ');
  readln(x);
  h:=trunc(x/v);
  m:=round((x/v - h)*60);
  writeln(h,' 時間 ',m,' 分');
end.
```

プログラムの解説。

2 行目　変数 v, x を実数型の変数として使用する。

3 行目　変数 h, m を整数型の変数として使用する。

5 行目 ’時速？ ’というメッセージを画面に表示する。

6 行目 キーボードからの入力を変数 v に読み込む。

7 行目 ’距離 (km)？ ’というメッセージを画面に表示する。

8 行目 キーボードからの入力を変数 x に読み込む。

9 行目 x/v の整数部分を変数 h に代入する。

10 行目 x/v-h の小数部分に 60 をかけた数の小数点以下を四捨五入して変数 m に代入する。

11 行目 h の値、’ 時間 ’ という文字列、m の値、’ 分’ という文字列を表示する。

2.3 文字型・文字列型

文字型は、a, . . ., z、A, . . ., Z、0, . . ., 9 などの英数字 1 文字を表現します。文字型変数は次のように宣言します:

```
var c1,c2: char;
```

文字列型は文字列を表現します。漢字 1 字は長さ 2 の文字列として表現されます。文字列型変数は次のように宣言します:

```
var name: string[20];
```

例 2.3. 名前 (ファーストネーム) をローマ字で入力して、その文字数、頭文字を表示する。

```
program Lesson5;
 var name: string[20];
     init: char;
        n: integer;
 begin write('Your first name? ');
       readln(name);
       init:=name[1];
       n:=Length(name);
       writeln('Length is ', n);
       writeln('Initial is ', init);
end.
```

プログラムの解説

2 行目 変数 name を長さ 20 文字以内の文字列型の変数として宣言する。

3 行目 変数 init を文字型の変数として宣言する。

4 行目 変数 n を整数型の変数として宣言する。

5 行目 'Your first name? ' というメッセージを画面に表示する。

6 行目 キーボードからの入力を変数 name に読み込む。

7 行目 変数 init に文字列 name の第 1 番目の文字を代入する。

8 行目 変数 n に文字列 name の長さを代入する。

9 行目 'Length is ' というメッセージと変数 n の値を表示する。

10 行目 'Initial is ' というメッセージと変数 init の値を表示する。

2.4 論理型

論理型は true(真)、false(偽) の 2 つを表現します。論理型変数は次のように宣言します:

```
var c: boolean;
```

論理型の値をとる式を**論理式**といいます。論理式は論理型変数や論理値をとる関数を次の演算子でつないだものです:

```
not, and, or, =, <>, <=, >=, <, >
```

例 2.4. 「a=b かつ c≥d」が真のとき、true をとり、そうでないとき false をとる論理式。

```
(a=b) and (c>=d)
```

2.5　配列

100 個の数値データを入力して、その合計、平均、標準偏差を表示するプログラムを作ることを考えてみます。数学では、100 個の数値データを入れるために $a_1, a_2, \ldots, a_{100}$ という変数を用いればよいでしょう。Pascal でも、これに対応するものとして**配列**が用意されています。a という変数を配列として宣言して、a[1], a[2], …, a[100] と利用します。

例 2.5. a を添字 1 から 100 を持つ整数の配列、b を添字 1 から 100 を持つ実数の配列として変数宣言する。

```
var a: array[1..100] of integer;
    b: array[1..100] of real;
```

3　条件判定

3.1　if 文

ある条件を満たしたときに、特定の処理を行うのが **if 文**の役目です。if 文の書式には次のように 2 通りあります:

```
if　論理式　then　文
if　論理式　then　文 1　else　文 2
```

注意 3.1. else の前の文 1 の最後にセミコロン; をつけてはいけません。

例 3.2. 2 次方程式を解く。

```
program Lesson6;
  var a,b,d,x1,x2: real;
  begin
    writeln('x^2 + ax + b = 0');
    write('a= '); readln(a);
    write('b= '); readln(b);
    d:=a*a - 4*b;
    if d>=0 then begin
      x1:=(-a - sqrt(d))/2;
      x2:=(-a + sqrt(d))/2;
      writeln(x1:5:3);
      writeln(x2:5:3);
    end
    else writeln('実根はありません。');
end.
```

3.2　case 文

1 つの変数や式がさまざまな条件で異なる処理を行う場合は、**case 文**が役に立ちます。case 文の書式は次の通りです:

```
case　式　of
  case ラベル，……，case ラベル: 文;
  case ラベル，……，case ラベル: 文;
end;
```

例 3.3. 入力された月に対して、春夏秋冬を答える。

```
program Lesson7;
  var mon:integer;
  begin
    write('何月ですか？ ');
    readln(mon);
    case mon of
      3..5     : write('春です。');
      6..8     : write('夏です。');
      9..11    : write('秋です。');
      12, 1..2 : write('冬です。');
    end;
  end.
```

プログラムの解説

2 行目 変数 mon を整数型の変数として宣言する。

4 行目 ’何月ですか？ ’というメッセージを画面に表示する。

5 行目 キーボードからの入力を変数 mon に読み込む。

6〜11 行目 変数 mon の値が 3 から 5 のときは、’春です。’というメッセージを表示、変数 mon の値が 6 から 8 のときは、’夏です。’というメッセージを表示、変数 mon の値が 9 から 11 のときは、’秋です。’というメッセージを表示、変数 mon の値が 12 あるいは 1 から 2 のときは、’冬です。’というメッセージを表示する。

4 繰り返し処理

4.1 for 文

ある処理を決まった回数だけ繰り返し行うために用いるのが **for 文**です。for 文には次の 2 種類があります:

```
for 制御変数:=初期値 to 終値 do 文      (制御変数が 1 ずつ増加)
for 制御変数:=初期値 downto 終値 do 文 (制御変数が 1 ずつ減少)
```

例 4.1. 1 から n までの自然数の和を求める。

```
program Lesson8;
  var a,n,s: integer;
  begin
    write('n = '); readln(n);
    s:=0;
    for a:=1 to n do s:=s + a;
    writeln('1 + ... + ',n,' = ',s);
  end.
```

プログラムの解説

2 行目 変数 a, n, s を整数型の変数として宣言する。

4 行目 ’n= ’というメッセージを画面に表示し、キーボードからの入力を変数 n に読み込む。

5 行目 変数 s に値 0 を代入する。

6 行目 変数 a の値が 1 から n まで動くとき、変数 s に s+a の値を代入することを繰り返す。

7 行目 ’1 + ... + ’というメッセージ、n の値、s の値を表示する。

例 4.2. 3 辺の長さ a≤b<c が整数の直角三角形で c≤n のものをすべて求める。

```
program Lesson9;
var a,b,c,n: integer;
begin
 write('n = '); readln(n);
 for a:=1 to n do
  for b:=a to n do
   for c:=b to n do
    if a*a+b*b=c*c then
    writeln(a:5,b:5,c:5);
end.
```

プログラムの解説

2 行目 変数 a, b, c, n を整数型の変数として宣言する。

4 行目 ’n= ’というメッセージを画面に表示し、キーボードからの入力を変数 n に読み込む。

5〜8 行目 変数 a の値が 1 から n まで、b の値が a から n まで、c の値が b から n まで動くとき、もし、a*a+b*b=c*c が成り立っていれば、a、b、c の値を表示する。

例 4.3. 10 個の数値データを入力して、その平均と標準偏差を求める。

```
program Lesson10;
const n=10;
var a: array[1..n] of integer;
```

```
    h,s,t,v:real;
    i,x:integer;
begin
    for i:=1 to n do readln(a[i]);
    s:=0;
    for i:=1 to n do s:=s+a[i];
    h:=s/n; (* 平均 *)
    t:=0;
    for i:=1 to n do t:=t+sqr(a[i]-h);
    v:=sqrt(t/n); (* 標準偏差 *)
    writeln(h:5:2, ' ', v:5:2);
end.
```

プログラムの解説

2 行目 定数 n を 10 として定義する。

3 行目 配列 a を添字 1 から 10 を持つ整数の配列として宣言する。

4 行目 変数 h, s, t, v を実数型の変数として宣言する。

5 行目 変数 i, x を整数型の変数として宣言する。

7 行目 変数 i が 1 から n まで動くとき、キーボードからの入力を配列 a の i 番目の成分 a[i] に読み込む。

8 行目 変数 s に 0 を代入する。

9 行目 変数 i が 1 から n まで動くとき、変数 s に s+a[i] を代入する。この結果、s の値は a[1] から a[n] までの合計になる。

10 行目 変数 h に s/n の値を代入する。h の値は a[1] から a[n] までの平均値になる。(* *) で囲まれた部分は、何をしているか後で分かるために入れたコメント。

11 行目 変数 t に 0 を代入する。

12 行目 変数 i が 1 から n まで動くとき、変数 t に $t+(a[i]-h)^2$ を代入する。この結果、t の値は $(a[1]-h)^2$ から $(a[n]-h)^2$ までの合計になる。

13 行目 変数 v に t/n の平方根を代入する。v の値は a[1] から a[n] の標準偏差になる。

14 行目 h の値、v の値を表示する。

4.2 while 文

ある条件が満たされている間、繰り返しを行うのが **while 文**です。書式は次の通りです:

```
while  論理式  do  文
```

例 4.4. 素数判定。

```
program Lesson11;
var a,n,amax: integer;
begin
  write('2 以上の整数を入力してください。'); readln(n);
    amax:=trunc(sqrt(n));
    if n mod 2=0 then
begin if n=2 then  writeln(n,' は素数です。')
else writeln(n,' は素数ではありません。');
end
    else begin
           a:=3;
           while (n mod a<>0) and (a<=amax) do a:=a+2;
           if (n mod a=0) and (n>a) then
           writeln(n,' は素数ではありません。')
           else writeln(n,' は素数です。');
         end;
  end.
```

プログラムの解説

2 行目 変数 a, n, amax を整数型の変数として宣言する。

4 行目 '2 以上の整数を入力してください。' というメッセージを表示して、キーボードからの入力を変数 n に代入する。

5 行目 変数 amax に、n の平方根の整数部分を代入する。

6 行目 変数 n を 2 で割った余りが 0 ならば (n が偶数ならば)、

7〜9 行目 n=2 ならば、n の値、’ は素数です。’ というメッセージを表示し、そうでなければ、n の値、’ は素数ではありません。’ というメッセージを表示する。

10 行目 n が奇数ならば、

11 行目 変数 a に 3 を代入する。

12 行目 n を a で割った余りが 0 でなく、かつ a≤amax であるという条件が満たされている間、a に a+2 を代入することを繰り返す。

13 行目 n を a で割った余りが 0 であり、かつ n>a ならば、n の値、’ は素数ではありません。’ というメッセージを表示し、

14 行目 そうでなければ、n の値、’ は素数です。’ というメッセージを表示する。

4.3 repeat 文

repeat 文は、ある条件が満たされない場合に繰り返しを行い、条件が成立した場合に繰り返しを行わないで次の処理に進みます。書式は次の通りです:

```
repeat　文　until　論理式
```

例 4.5. ユークリッドの互除法によって最大公約数を求める。

```
program Lesson12;
var a,b,r: integer;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  if b>a then begin
           r:=a; a:=b; b:=r;
         end;
  repeat
    r:=a mod b; a:=b; b:=r;
  until b=0;
  writeln(' 最大公約数は ',a);
end.
```

プログラムの解説

2 行目 変数 a, b, r を整数型の変数として宣言する。

4 行目 ’a= ’ というメッセージを表示して、キーボードからの入力を変数 a に代入する。

5 行目 ’b= ’ というメッセージを表示して、キーボードからの入力を変数 b に代入する。

6 行目 b>a ならば、

7 行目 r に a の値を代入し、a に b の値を代入し、b に r の値を代入する。

9〜11 行目 r に a を b で割った余りを代入し、a に b の値を代入し、b に r の値を代入する。これを b の値が 0 になるまで繰り返す。

12 行目 ’ 最大公約数は ’ というメッセージ、a の値を表示する。

5 手続きと関数

5.1 手続き

手続き宣言部の書式は次の通りです:

```
procedure <手続き名>(仮引数部>, …, <仮引数部>);
<手続きブロック>
```

ここで、仮引数部には最初に **var** の付くものと付かないものがあります。**var** の付かないものを**値引数**といい、付くものを**変数引数**といいます。値引数は手続き側で引数の値だけを参照する場合に用います。これに対して、変数引数は手続きの中でその変数の値を変えて、その結果を呼び出した側に戻す場合に用います。

例 5.1. 複素数の積。

```
program Lesson13;
type complex=array[1..2] of real;
var x,y,z: complex;
(* c:=a*b *)
procedure compmul(var c: complex; a,b: complex);
begin
  c[1]:=a[1]*b[1]-a[2]*b[2];
  c[2]:=a[1]*b[2]+a[2]*b[1];
end;
begin
  write('Re x = '); readln(x[1]);
  write('Im x = '); readln(x[2]);
  write('Re y = '); readln(y[1]);
  write('Im y = '); readln(y[2]);
  compmul(z,x,y);
  writeln('x*y = ',z[1]:4:2,' + ',z[2]:4:2,'i');
end.
```

プログラムの解説

2 行目 complex という型を添字 1 から 2 を持つ実数の配列として定義する。1 番目の成分を実部、2 番目の成分を虚部として用いる。

3 行目 変数 x, y, z を complex 型の変数として宣言する。

5〜9 行目 複素数 a、b の積 c を計算する手続き compmu を定義している。lc は変数引数、a、b は値引数。

11 行目 'Re x = ' というメッセージを表示し、キーボードからの入力を変数 x[1] に代入する。

12 行目 'Im x = ' というメッセージを表示し、キーボードからの入力を変数 x[2] に代入する。

13 行目 'Re y = ' というメッセージを表示し、キーボードからの入力を変数 y[1] に代入する。

14 行目 'Im y = ' というメッセージを表示し、キーボードからの入力を変数 y[2] に代入する。

15 行目 手続き compmul を呼び出して、複素数 x、y の積を変数 z に入れて返す。

16 行目 'x*y = ' というメッセージ、z[1] の値、' + ' というメッセージ、z[2] の値、'i' というメッセージを表示する。

5.2 関数

関数宣言部の書式は次の通りです (下線部分なしの場合もあり):

function <関数名>(仮引数部>, …, <仮引数部>):<結果型>;
<関数ブロック>

例 5.2. 分数を既約分数になおす。

```
program Lesson14;
var a,b,d:integer;
function gcd(a,b:integer):integer;
var r: integer;
begin
  if b>a then begin
           r:=a; a:=b; b:=r;
         end;
  repeat
    r:=a mod b;
    a:=b;
    b:=r;
  until b=0;
  gcd:=a;
end;
begin (* main *)
  write('分子 = '); readln(a);
  write('分母 = '); readln(b);
```

```
  d:=gcd(a,b);
  a:=a div d; b:=b div d;
  write(a,'/',b);
end.
```

プログラムの解説

2 行目　変数 a, b, d を整数型の変数として宣言する。

3〜15 行目　最大公約数を返す関数 gcd を定義している。内容は例 4.5 とほぼ同じ。

14 行目　で、関数の値を返している。

17 行目　'分子 = ' というメッセージを表示して、キーボードからの入力を変数 a に代入する。

18 行目　'分母 = ' というメッセージを表示して、キーボードからの入力を変数 b に代入する。

19 行目　a と b の最大公約数を変数 d に代入する。

20 行目　a に a を d で割った商を代入し、b に b を d で割った商を代入する。

21 行目　a の値、'/' というメッセージ、b の値を表示する。

A　Pascal 予約語、標準関数、手続き

A.1　Pascal 予約語

absolute	**end**	**inline**	**procedure**	**type**
and	**external**	**interface**	**program**	**unit**
array	**file**	**interrupt**	**record**	**until**
begin	**for**	**label**	**repeat**	**uses**
case	**forward**	**mod**	**set**	**var**
const	**function**	**nil**	**shl**	**while**
div	**goto**	**not**	**shr**	**with**
do	**if**	**of**	**string**	**xor**
downto	**implementation**	**or**	**then**	
else	**in**	**packed**	**to**	

A.2　Pascal 標準関数・手続き

i,j,n:整数型、x:実数型、a:整数型または実数型、c,ch:文字型、f:ファイル型、s,st:文字列型、b:論理型、w:write パラメータとする。

write パラメータ	
c, s	文字、文字列そのまま
c:n, s:n	n 文字幅の欄に右詰め
i	i の値を 10 進数表記
i:n	n 文字幅の欄に右詰め
x	x の値を浮動小数点による 10 進数表記
x:n	n 文字幅の欄に右詰め
x:n:j	x の値を小数点以下 j 桁の固定小数点による 10 進数表記で、n 文字幅の欄に右詰め

標準入出力	
read(f, v, . . ., v) readln(f, v, . . ., v) write(f, w, . . ., w) writeln(f, w,. . ., w)	v は変数で、文字、文字列、整数、実数型

算術計算	
a:=abs(a)	絶対値
a:=sqr(a)	2 乗
x:=sin(a)	正弦関数
x:=cos(a)	余弦関数
x:=exp(a)	指数関数
x:=ln(a)	自然対数
x:=sqrt(a)	平方根
x:=arctan(a)	逆正接関数
i:=trunc(x)	小数以下の切り捨て
i:=round(x)	小数以下の四捨五入
b:=odd(i)	奇数なら true、偶数なら false
randomize	乱数発生の初期化
x:=random	0.0 以上 1.0 未満の一様乱数
i:=random(n)	0 以上 n 以下の乱数

文字・文字列関係	
c:=chr(i)	文字コードに対する文字を返す
i:=ord(c)	文字のコードを返す
s:=copy(st,i,n)	文字列 st の i 番目から n 個の文字列を返す
s:=concat($s_1, \ldots, s_n$)	文字列 s_1 から s_n を結合した文字列を返す
i:=length(st)	文字列 st の長さを返す
i:=pos(s,st)	文字列 st 中の文字列 s が最初にある位置を返す (なければ 0 を返す)
ch:=upcase(c)	文字 c の大文字
delete(st,i,n)	文字列 st の i 番目から n 文字削除
insert(s,st,i)	文字列 s を文字列 st の i 番目の位置に挿入
str(w,s)	整数または実数型の w パラメータを文字列 s に変換する
val(s,a,i)	文字列 s を数値に変換し a に代入する (結果の i が 0 でないときは、s の i 番目の文字でエラーが生じたことを意味する)

B　レポートの提出についての注意

提出期限　2/10(木) 正午（それ以後は受け付けない）

提出場所　中川研究室 (自然棟 7F)

内　　容　課題の中から授業中に選択を決めた課題。

形　　式

1. 形式 (A4 用紙 5 枚程度、レポート課題名、学籍番号、コース名、氏名を書く。)
2. レポート課題名
3. プログラムそのものの記述（プリントアウトしたものでも可）
4. プログラムの実行結果の記述（プリントアウトしたものでも可）
5. 考察（ここには次のものが全て含まれていることが望ましい）

(a) プログラムを考案するにいたった過程、特に、基本となるアイディアの選択の過程とそれをプログラムに実現する上での工夫など。

(b) 実行結果との関係で、プログラムの問題点と、それをどのように改善すればよいと考えられるかなど。

(c) 配布の資料以外の参考図書などを参考にした場合には、それを明記する。参考文献としての書き方は、次の表記法を参考にする。

永野三郎・長島　忍・吉村　伸, 『Pascal プログラミング [第 2 版]』, 東京大学出版会, 1992.