整数データと浮動小数データ

整数データと浮動小数データの違い

本日の内容

例題1．単純な金種計算

例題2．硬貨の金種計算

例題3．うるう年の判定

整数の変数

浮動小数と整数の違い

例題4．複利計算

整数の変数と，浮動小数の変数を混在させるときに気を付けねばならないこと

今日の到達目標

- プログラムでの「整数データ」と「浮動小数データ」の違いについて理解する
- 目的に応じて，整数の変数，浮動小数の変数を正しく使い分けることができるようになる

例題1．単純な金種計算

- 金額を読み込んで，適切な紙幣と小銭の枚数を求め，表示するプログラムを作る．

  例）金額が1050円のとき，

  千円札：1枚

  1円玉：50枚

- 例題では，簡単のため，紙幣は千円札のみ，硬貨は1円玉のみ（種別は考えない）ことにする
- 金額，千円札の枚数，1円玉の枚数を扱うために，整数の変数を使う

- 1000円札の枚数
  金額を1000で割った商（小数点以下切り捨て）
- 1円玉の枚数
  金額を1000で割った余り

```
program sum;
{$APPTYPE CONSOLE}
uses SysUtils;
var kingaku, sen, en: integer;
begin
  write('kingaku ?: ');
  readln(kingaku);
  sen := kingaku div 1000;
  en := kingaku mod 1000;
  writeln('1000 en =', sen:8, ' mai');
  writeln('1 en =   ', en:8, ' mai');
  readln
end.
```

キーボードからのデータの読み込みを行っている部分

計算を行っている部分

画面へのデータの書き出しを行っている部分

単純な金種計算

実行結果の例

```
kingaku ?: 3020
1000 en =        3 mai
1 en =          20 mai
```

```
kingaku ?: 12345678
1000 en =    12345 mai
1 en =         678 mai
```

実行結果画面（例）

プログラム実行順

```
write('kingaku ?: ');
```
メッセージ「金額は？:」を表示

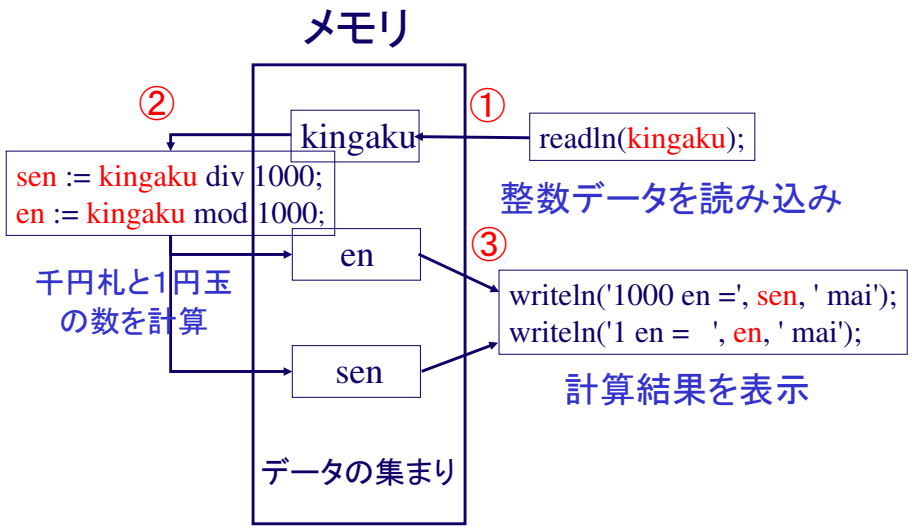
```
readln(kingaku);
```
整数データを読み込み

```
sen := kingaku div 1000;
en := kingaku mod 1000;
```
千円札と1円玉の数を計算

```
writeln('1000 en', sen, ' mai');
writeln('1 en   ', en, ' mai');
```
計算結果を表示

```
readln
```
終わりの readln

プログラムとデータ



整数データと浮動小数データ

- 整数データ
  - 整数（正か負か0）でおよそ9桁くらいまで
  - 割り算では小数点以下切り捨て

例）
0
3
28
4778
−1
−10
−1250

- 浮動小数データ
  - 小数付きの数も可
  - 「○○×10^N」と書いたとき「○○」の部分がおよそ12桁くらいまで

例）
0
3
1278748623
−4563759398
2. 190872
0. 000178

整数データと浮動小数データの精度

- 整数データ (integer)
  - 整数で，およそ9桁くらいまでの数（正か負か0）
  - 数学での「整数」では無い（桁数に制限がある）
- 浮動小数データ (real)
  - 「○○×10^N」と書いたとき，「○○」の部分がおよそ12桁くらいまでの数．Nがおおよそ2桁くらいまで
  - 数学での「実数」とは違う
  - 小数つきの数も許す（整数データとの大きな違い）

整数データと浮動小数データの違い

	整数データ	浮動小数データ
変数宣言	kingaku :integer;	teihen :real;
四則演算	+, -, *, div •div は割り算（余りは切り捨て）	+, -, *, /
剰余	mod	

変数

- 変数には，名前と型（型とはデータの種類のこと）がある
- 変数宣言では，名前と型を書く

「型」の例

– 整数データ　integer

– 浮動小数データ　real

整数データの算術演算子

＋　和

－　差

*　積

div　割り算の商（余りは切り捨て）

mod　割り算の剰余

練習1．例題1を実行せよ

① 例題1のプログラム（資料の次ページ）

② Borland Delphi 6 の起動

「スタート」→「プログラム」→「Borland Delphi 6」→「Delphi 6」

③ コンソールアプリケーションの新規作成

「ファイル」→「新規作成」→「その他」⇒ ウインドウが現れる

「コンソールアプリケーション」→「OK」 ⇒ ウインドウが現れる

④ ①のプログラムコードを「コピー」して「貼り付け」

⑤ コンパイル

「プロジェクト」→「〇〇をコンパイル」

⑥ 実行

「実行」→「実行」

3020, 1234567 など，さまざまな金額を試してみよ

```
program sum;
  {$APPTYPE CONSOLE}
  uses SysUtils;
  var kingaku, sen, en: integer;
  begin
       write('kingaku ?: ');
       readln(kingaku);
       sen := kingaku div 1000;
       en := kingaku mod 1000;
       writeln('1000 en =', sen:8, ' mai');
       writeln('1 en =   ', en:8, ' mai');
       readln
  end.
```

例題2. 硬貨の金種計算

- 金額を読み込んで，適切な小銭の枚数を求め，表示するプログラムを作る．

  例）金額が768円のとき，

  500円玉：1枚
  100円玉：2枚
  50円玉：1枚
  10円玉：1枚
  5円玉：1枚
  1円玉：3枚

- 例題では，簡単のため，紙幣は考えない（硬貨のみ）ということにする
- 各硬貨の枚数を扱うために，整数の変数を使う

```
program sum;
{$APPTYPE CONSOLE}
uses SysUtils;
var kingaku, n500, n100, n50, n10, n5, n1: integer;
begin
  write('kingaku ?: ');
  readln(kingaku);
  n500 := kingaku div 500;
  n100 := ( kingaku mod 500 ) div 100;
  n50 := ( kingaku mod 100 ) div 50;
  n10 := ( kingaku mod 50 ) div 10;
  n5 := ( kingaku mod 10 ) div 5;
  n1 := kingaku mod 5;
  writeln('500 en', n500:4, ' mai');
  writeln('100 en', n100:4, ' mai');
  writeln('50 en ', n50:4, ' mai');
  writeln('10 en ', n10:4, ' mai');
  writeln('5 en  ', n5:4, ' mai');
  writeln('1 en  ', n1:4, ' mai');
  readln
end.
```

キーボードからのデータの読み込みを行っている部分

計算を行っている部分

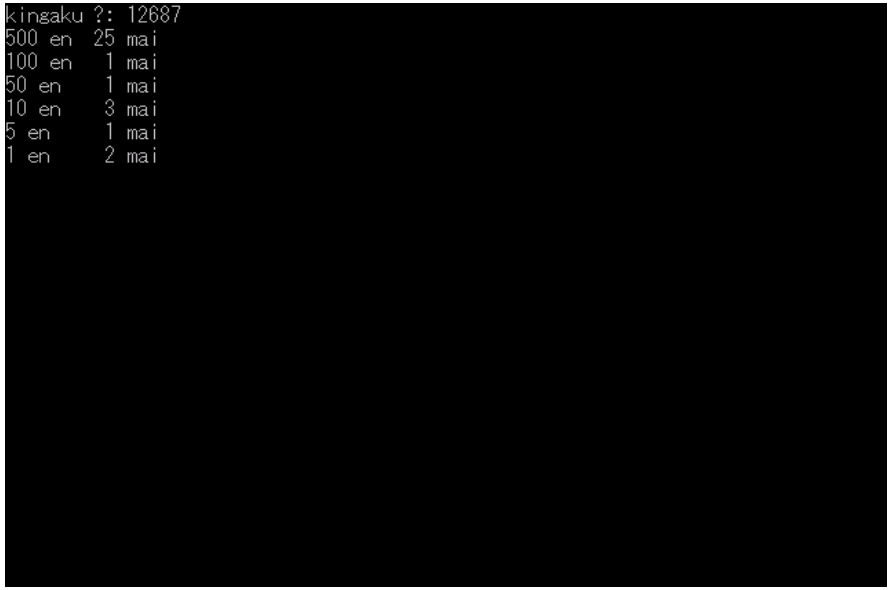
画面へのデータの書き出しを行っている部分

硬貨の金種計算

実行結果の例

```
kingaku ?: 12687
500 en =    25 mai
100 en =    1 mai
50 en =     1 mai
10 en =     3 mai
5 en =      1 mai
1 en =      2 mai
```

実行結果画面(例)



硬貨の金種計算

- 500円玉の枚数は
  「(金額)を1000円で割った余り」割る 500

例えば, 12687 円のとき

500円玉: 12687 div 500 → 25

100円玉: (12687 mod 500) div 100 → 1

12687 を 500 で割った余り(値は 187)

50円玉: (12687 mod 100) div 50 → 1

12687 を 100 で割った余り(値は 87)

以下同様に考える

練習2. 例題2を実行せよ

① 例題2のプログラム(資料の次ページ)

② Borland Delphi 6 の起動

「スタート」→「プログラム」→「Borland Delphi 6」→「Delphi 6」

③ コンソールアプリケーションの新規作成

「ファイル」→「新規作成」→「その他」⇒ ウインドウが現れる

「コンソールアプリケーション」→「OK」 ⇒ ウインドウが現れる

④ ①のプログラムコードを「コピー」して「貼り付け」

⑤ コンパイル

「プロジェクト」→「○○をコンパイル」

⑥ 実行

「実行」→「実行」

768, 12687, 1234567 など, さまざまな金額を試してみよ

```
program sum;
  {$APPTYPE CONSOLE}
  uses SysUtils;
  var kingaku, n500, n100, n50, n10, n5, n1:
  integer;
  begin
      write('kingaku ?: ');
      readln(kingaku);
      n500 := kingaku div 500;
      n100 := ( kingaku mod 500 ) div 100;
      n50 := ( kingaku mod 100 ) div 50;
      n10 := ( kingaku mod 50 ) div 10;
      n5 := ( kingaku mod 10 ) div 5;
      n1 := kingaku mod 5;
      writeln('500 en', n500:4, ' mai');
      writeln('100 en', n100:4, ' mai');
      writeln('50 en ', n50:4, ' mai');
      writeln('10 en ', n10:4, ' mai');
      writeln('5 en  ', n5:4, ' mai');
      writeln('1 en  ', n1:4, ' mai');
      readln
  end.
```

例題3. うるう年の判定

- 「西暦年」を読み込んで，うるう年かどうか表示するプログラムを作る.
  - うるう年の判定のために，比較演算と論理演算を組み合わせる
  - 西暦年が 4, 100, 400 の倍数であるかを調べるために mod を使う

例） 2001 → うるう年でない

2004 → うるう年である

グレゴリオ暦でのうるう年

- うるう年とは： 2月が29日まである年
- うるう年は400年に97回で，1年の平均日数は365.2422日
- うるう年の判定法
  - 年数が4の倍数の年 → うるう年
  - 但し，100の倍数の年で400の倍数でない年
    → うるう年ではない
    （4の倍数なのだが例外とする）

（例） 2008年：うるう年（4の倍数）
2004年：うるう年（4の倍数）
2000年：うるう年（4の倍数）
1900年：うるう年ではない
（100の倍数だが400の倍数でない）
1800年：うるう年ではない
（100の倍数だが400の倍数でない）

```
program sum;
{$APPTYPE CONSOLE}
uses SysUtils;
var y: integer;
begin
   write('year ?: ');
   readln(y);
   if ((y mod 400) = 0) or (((y mod 100) <> 0) and ((y mod 4) = 0)) then begin
      writeln(y, ' is leap year');
   end
   else begin
      writeln(y, ' is NOT leap year');
   end;
   readln
end.
```

条件式

条件が成り立つ場合に実行される部分

条件が成り立たない場合に実行される部分

うるう年の判定

実行結果の例

```
year ?: 2001
2001 is NOT leap year
```

```
year ?: 2000
2000 is leap year
```

```
year ?: 1900
1900 is NOT leap year
```

実行結果画面（例）

うるう年の判定式

((y mod 400) = 0) or (((y mod 100) <> 0) and ((y mod 4) = 0))

400の倍数である　　100の倍数でない　　4の倍数である

かつ

または

練習3. 例題3を実行せよ

① 例題3のプログラム（資料の次ページ）

② Borland Delphi 6 の起動

「スタート」→「プログラム」→「Borland Delphi 6」→「Delphi 6」

③ コンソールアプリケーションの新規作成

「ファイル」→「新規作成」→「その他」⇒ ウインドウが現れる

「コンソールアプリケーション」→「OK」 ⇒ ウインドウが現れる

④ ①のプログラムコードを「コピー」して「貼り付け」

⑤ コンパイル

「プロジェクト」→「○○をコンパイル」

⑥ 実行

「実行」→「実行」

2002, 2001, 2000, 1999, 1998, 1997, 1996, 1900, 1600 などを試してみよ

```
program sum;
  {$APPTYPE CONSOLE}
  uses SysUtils;
  var y: integer;
  begin
       write('year ?: ');
       readln(y);
       if ((y mod 400) = 0) or (((y mod 100) <>
  0) and ((y mod 4) = 0)) then begin
              writeln(y, ' is leap year');
       end
       else begin
              writeln(y, ' is NOT leap year');
       end;
       readln
  end.
```

例題4. 複利計算

- 元金をある年利で，ある年数だけ運用したときの利息を表示する
  - 複利計算では，利息が利息を生む．
  - 複利計算を行うために，Power 関数（Power(x,y)でxのy乗）を使う
  - 整数データと浮動小数データが混在する
    - 元金　　gankin: 整数データ（単位は円）
    - 年数　　nensu:　整数データ（単位は年）
    - 年利　　nenri: 浮動小数データ（単位は　%）

```
program sum;
{$APPTYPE CONSOLE}
uses SysUtils, Math;
var gankin, nensu, ganri: integer;
var nenri, r: real;
begin
  write('gankin ? (en): ');
  readln(gankin);
  write('nensu ? (nen): ');
  readln(nensu);
  write('nenri ? (%): ');
  readln(nenri);
  r := 1 + ( nenri * 0.01 );
  ganri := trunc( gankin * Power( r, nensu ) );
  writeln('ganri = ', ganri:8, ' en');
  writeln('risoku =', ( ganri - gankin ):8, ' en');
readln
end.
```

キーボードからのデータの読み込みを行っている部分

計算を行っている部分

画面へのデータの書き出しを行っている部分

硬貨の金種計算

実行結果の例

```
gankin ? (en): 10000
nensu ? (nen): 10
nenri ? (%): 2
ganri =    12189 en
risoku =    2189 en
```

実行結果画面（例）

複利の計算

- 複利の公式:

  元利 = 元金　×　（1＋年利）年数

- べき乗 x^y の計算のために，ライブラリ関数 Power(x,y) を使用する

```
ganri := trunc( gankin * Power( r, nensu ) );
```

整数に関する関数

round(浮動小数の変数や式)

最も近い整数

例）3.4 → 3, 3.6 → 4, -1.6 → 2

trunc(浮動小数の変数や式)

ゼロ方向に向かった最も近い整数

例）3.4 → 3, 3.6 → 3, -1.6 → -1

切り捨て，四捨五入の例

- 1.05倍の後に小数点以下を切り捨てて，

  → trunc を使用

```
n: integer;
x: real;
...
n = trunc( x * 1.05 );
```

- 3.141592倍の後に小数点以下を四捨五入

  →　round を使用

```
m: integer;
x: real;
...
m = round( x * 3.141592 );
```

trunc の意味

```
ganri := trunc( gankin * Power( r, nensu ) );
```

ganri: 整数　gankin: 整数　r: 浮動小数　nensu: 整数

- 右辺は浮動小数の精度で計算される
- 元利は本来，正の整数（利息の1円未満は切り捨て）
- 計算結果が「小数付きの値」であるとき，整数の変数に代入するには，小数点以下切り捨てか，四捨五入が行われねばならない
- trunc の意味:
  - 右辺の計算結果の小数点以下を切り捨てて，左辺の変数に代入する.

課題1．trunc

- 例題4のプログラムを次のように変更し，元金は10000円，年数は10年，年利は2%として実行してみよ．実行結果を報告せよ

変更前

```
var gankin, nensu, ganri: integer;
(途中省略)
ganri := trunc( gankin * Power( r, nensu ) );
writeln('ganri = ', ganri:8, ' en');
writeln('risoku =', ( ganri - gankin ):8, ' en');
```

変更後

```
var gankin, nensu: integer;
var ganri: real;
(途中省略)
ganri := gankin * Power( r, nensu );
writeln('ganri = ', ganri:8:3, ' en');
writeln('risoku =', ( ganri - gankin ):8:3, ' en');
```

課題2．金種計算

- 金額を読み込んで，適切な紙幣と小銭の枚数を求め，表示するプログラムを作りなさい．
  - 但し，すべての種類の紙幣と硬貨を考えること
  - 例題2のプログラムを参考にすること

例）金額が13，486円のとき，

1万円札：1	500円：0
5千円札：0	100円：4
千円札：3	50円：1
	10円：3
	5円：1
	1円：1

課題2の実行結果例

```
金額は？: 13486
10000円札    1
5000円札    0
1000円札    3
500円玉    0
100円玉    4
50円玉    1
10円玉    3
5円玉    1
1円玉    1
```

課題3．時間の換算

- 秒数 x を読み込んで，h 時，m 分，s 秒を計算するプログラムを作りなさい．

例）x=3723 のとき，

1 h, 2 m, 3 s