第7回
条件判定

(テキスト第3章)

今日の内容

第2回レポート課題の発表

1. 文の分類
2. プログラム中で文を実行する順序
3. プログラムへのコメント
4. if 論理式 then 文
5. if 論理式 then 文1 else 文2
6. begin と end の付け方の注意
7. 論理式の書き方
8. 閏年の判定
9. case文
10. 今日の練習問題

第2回レポート課題

課題　条件判定、繰り返しまたは配列を利用したプログラムを作る。

形式　テキスト pp.47～48

演習問題　テキスト pp.32～33、pp.44～46、pp.58～60

提出日

1. 文の分類

- 文
  - 単純文
    - 代入文　x := a + b
    - 手続き文　入力文　readln
    - 出力文　writeln
  - 構造化文
    - (逐次構造)　複文　begin ... end
    - (選択構造)　条件文　if ... then ...　if ... then ... else ...　case
    - (反復構造)　繰り返し文　for、while、repeat

2. プログラム中で文を実行する順序

- プログラムは必ず begin と end で囲まれた複文になっている。

```
program sample;
begin                    複文
    writeln('Hello!')
end.
```

- 複文の中では、下の原則に従って文を実行する。
  - 原則: 単純文または構造化文を単位として、上から下へ、左から右へと順番に実行する。
- 構造化文のうち条件文と繰り返し文では、その構造に従った実行順序で実行される。
  - 例: 条件文では、条件が成り立たないときには、thenの後の文は実行されない。

複文での begin と end の意味

- 複文の begin と end は、左括弧「(」と右括弧「)」に対応する。
  - Pascal の複文は、複数の文をあたかも1つの文であるように見せるためのものである。そのため、複数の文をbeginとendで括る。(括弧で括るイメージ)
- もし、対応する end がない場合にはエラーとなる。
  - コンピュータは、begin に対応する end が必ずあるはずとしてプログラムを解釈する。
  - 運悪く見つからない場合もある。
  - プログラムする場合には、begin と end の対応に注意する。

3. プログラムへのコメント

- 複文などを使用すると、プログラムが長くなる。
  - 長いプログラムは読むのが難しい（理解しにくい）。
  - 時間が経つと、自分が作ったプログラムでもなかなか理解しにくい。
- プログラムのまとまりごとに、そこは何をしている部分であるかをコメント（注釈）として書くようにしよう。
  - コメントがある方がプログラムは読みやすくなる。
  - 集合括弧「{ }」で囲まれた文字列はコメントとなる。
  - ある行に「//」があれば、それより後ろにある文字列はコメントとなる。
  - コメントは、何をかいてもプログラムの動作には影響しない。

Literacy

4. if 論理式 then 文

一般形　if 論理式 then 文

意味　論理式が成り立つ時だけ 文 を実行して、
論理式が成り立たない場合は何も実行しない。
実行する 文 は1つの文だけ。

例
```
if (age >= 20) and (money > 0) then begin
        writeln('You can drink beer.');
        writeln('You can drink whisky.');
        writeln('You can drink wine.')
end
```
文（複文）

Literacy

5. if 論理式 then 文1 else 文2

一般形　if 論理式 then 文1 else 文2

意味　論理式が成り立つ時だけ 文1 を実行して、
成り立たない場合は 文2 を実行する。
文1、文2 ともに1つの文。

例
```
if (age >= 20) and (money > 0) then begin
        writeln('You can drink beer.');
        writeln('You can drink whisky.');   文1
        writeln('You can drink wine.')
end
else begin
        writeln('You can drink water.');    文2
        writeln('You can drink juice.')
end
```

Literacy

6. begin と end の付け方の注意

if 論理式 then 文において

誤

```
if not(x = 0) then y:=1/x; z:=2/x;
writeln(y,z)
```

文法的には正しいが、x=0の時にz:=2/x
の計算をしようとしてエラーとなる。

正

```
if not(x = 0) then begin y:=1/x; z:=2/x end
writeln(y,z)
```
括ることで1つの文（複文）

Literacy

7. 論理式の書き方

7.1 関係演算子　(=, >, <, <>, >=, <=)

Pascal	数学表記	備考
x = 2	x ＝ 2	代入文 x := 2 とは違う
x > 2	x ＞ 2	
x < 2	x ＜ 2	
x <> 2	x ≠ 2	
x >= 2	x ≧ 2	greater than or equal to
x <= 2	x ≦ 2	less than or equal to

x や 2 の所には、整数型または実数型の定数、変数、計算式が書ける。
例：　sqr(b)-4*a*c > 0
「=>」「=<」という表現は Pascal にはないので注意。

Literacy

7.2 論理演算子(not, and, or)による複雑な論理式

Pascal	数学表記
not(x = 2)	x ≠ 2
(0 < x) and (x <= 10)	0 ＜ x ≦ 10
(x < 0) or (x > 10)	x ＜ 0, x ＞ 10

括弧（ ）は絶対に必要

論理演算子の優先順位　(高not、中and、低or)
(x<0) or not(x=10) and (x>5)　　と
(x<0) or ((not(x=10)) and (x>5))　は同じ意味で、
ともに「x<0, 5<x<10, 10<x」を意味する。

Literacy

8. 閏年の判定 (uruu1.pas)

「西暦 i年が閏年かどうか」を判定する

(ふつうの言葉) 「iが4で割り切れて100で割り切れないか、400で割り切れる」

↓

(論理的な言葉) 「「iが4で割り切れる」かつ「iが100で割り切れない」」または「iが400で割り切れる」

↓

(Pascalの言葉)
```
if ((i mod 4=0) and (i mod 100 <>0))
    or (i mod 400=0)
  then writeln('leap year')
  else writeln('NOT leap year')
```

Literacy

9. case文

例(month1) 各月の日数を計算する。

```
writeln('日数を知りたい月は？');
readln(month);
case month of
  1,3,5,7,8,10,12: days:=31;
  4,6,9,11:        days:=30;
  2:               days:=28
  else writeln('input error')
end;
writeln(month, '月は', days, '日です');
```

- 変数 month が case ラベル 4 or 6 or 9 or 11 に一致する場合は、文「days:=30」を実行する。
- 変数は整数型または文字型であること。
- case のラベル「3..6」は「3, 4, 5, 6」 と同じ。

Literacy

case文（その2） 成績判定

```
write('あなたの点数は？');
readln(score);
case score of
   80..100: judge := 'A';
   70..79 : judge := 'B';
   60..69 : judge := 'C';
   0 ..59 : judge := 'D';
   else     judge := '不明'
end;
writeln('判定は、', judge, 'です。');
```

Literacy

10. 今日の練習問題

Level	問題
C	閏年を判定するプログラムuruu1を完成させよ。
C+	各月の日数を計算するプログラムmonth1を完成させよ。
B	month1を閏年にも対応させよ。
A	テキストp.33の問題8 ([Zellarの公式]を使った問題)のプログラムを完成させよ。

Literacy