第10回
手続きと関数

（グラフィカルUI）

Literacy

今日の内容

第3回レポート課題発表

1. 手続き（procedure）の必要性
2. 手続き（procedure）の特徴と一般形
3. 手続きNN_table(N)の定義
4. 関数（function）
5. 処理を選択して呼び出すプログラム
6. Delphiによるボタンの作成
7. 今日の練習問題

Literacy

第3回レポート課題発表

課題　今までの総復習
（特に、配列、手続き、関数）
できればグラフィックの機能を使った
プログラムを作成する。

形式　テキスト pp.118～119

演習問題　テキスト p.116 など

提出日

Literacy

1. 手続き(procedure)の必要性

複数のN×Nの表を描こう。

```
program lots_of_calc;
{$APPTYPE CONSOLE}
uses SysUtils;
var i, j: integer;
begin
    writeln('Ｎ×Ｎの表を描く。まず3×3の表');
    for i := 1 to 3 do begin
        for j := 1 to 3 do write(i*j:3);  writeln
    end;
    writeln('次は、6×6の表');
    for i := 1 to 6 do begin
        for j := 1 to 6 do write(i*j:3);  writeln
    end;
    writeln('最後に、9×9の表');
    for i := 1 to 9 do begin
        for j := 1 to 9 do write(i*j:3);  writeln
    end;
    readln
end.
```

Literacy

3、6、9という数字Nが違うだけ。

```
for i:=1 to N do begin
    for j:=1 to N do write(i*j:3); writeln
end
```

これが、ある名前(NN_table(N))で登録(定義)できるとすっきり。
→ procedure文

```
program NN;
{$APPTYPE CONSOLE}
uses SysUtils;
procedure NN_table(N:integer);
begin .... end;
begin
    writeln('Ｎ×Ｎの表を描く。まず3×3の表');
    NN_table(3);
    writeln('次は6×6の表');
    NN_table(6);
    writeln('最後に9×9の表');
    NN_table(9);
    readln
end.
```

Literacy

2. 手続き（procedure）の特徴と一般形

手続き(procedure)の特徴

- 一連の文を一命令として定義
- 変数名は命令の定義を行う所でのみ有効
- program文の書き方とほとんど同じ

手続き(procedure)の一般形　手続き名（変数名:型; 変数名:型; …）

```
procedure sample(a,b:integer; c:real);
var x,y:integer;
begin
    文の列　（処理の手順）
end; {セミコロン}
```

手続きの中だけで使う変数を宣言

a, b, c → 仮引数、　x, y → 局所的変数
手続きsampleの中でだけで使える。(手続きの外には影響しない)

Literacy

3. 手続き NN_table(N) の定義

```
program NN;
{$APPTYPE CONSOLE}
uses SysUtils;
procedure NN_table(N:integer);
var i,j:integer;
begin
    for i:=1 to N do begin
        for j:=1 to N do write(i*j:3); writeln
    end
end;
```

手続きの定義

```
begin   メインプログラム
  ...
  NN_table(3);
  ...
end.
```

- 手続きの呼び出し(処理の依頼)
  手続きを呼び出さなければ、処理は行われない。
- 手続きに渡す値(この例では3)を実引数と呼ぶ。

Literacy

4. 関数(function)

- 自分独自の関数を定義することが可能
  - 「ユーザ定義の関数」と呼ぶ。
  - システムが用意している関数と同じように使用できる。

例: $f(x)=\sin x+\sin 2x+\sin 3x+\sin 4x$

- 関数の定義 : real型データを変数xにもらって計算

```
function f(x:real) : real;
begin
   f:=sin(x)+sin(2*x)+sin(3*x)+sin(4*x)
   {関数名:=計算結果で答えを返す。}
end; {セミコロン}
```

関数が返す値の型

Literacy

関数の呼び出し

```
{メインプログラム}
  begin
    ...
    y := yscale * f(step * i * pi / 180);
    ...
  end.
```

- 関数 f の呼び出し
  標準関数と同様に関数fを使える。

発展課題

次の関数を定義してグラフを描く。

$$y=\begin{cases}\sin(x)+\sin(2x)+\sin(3x)+\sin(4x) & :(0<x<\pi)\\ \sin(x) & :(\pi\le x\le 2\pi)\end{cases}$$

Literacy

5. 処理を選択して呼び出すプログラム

- 各処理を手続きで作成し、メインプログラムから呼び出す。

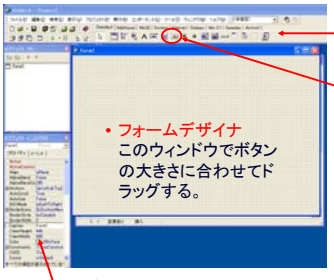
```
begin
    writeln('プログラム集:メニュー');
    writeln('1: 2数の和  2: 関年  3: 1~nの総和');
    write('番号を入れてください:');  readln(num);
    case num of
        1: sum;  2: uruu;  3: sowa
        else writeln('メニューにありません');
    end
end.
```

- この選択の部分をグラフィカルにできないか？
  - 例えば、ボタンに処理名がかかれていて、ボタンを押すとその処理を開始する。

Literacy

6. Delphi によるボタンの作成

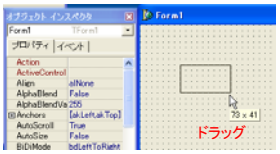
- コンポーネント・ツールボックスからボタンコンポーネントを選び、Formウィンドウの上に置く。



- コンポーネント・ツールボックス
- ボタンコンポーネント
- フォームデザイナ
  このウィンドウでボタンの大きさに合わせてドラッグする。
- ボタンコンポーネントが表示されていない場合は、「Standard」タブをクリックして、コンポーネントツールボックスの内容を変える。
- オブジェクトインスペクタ

Literacy

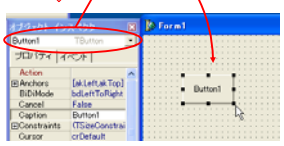
ボタンをフォームに貼り付ける



- ボタンの大きさは後からでも変更ができる。

確定

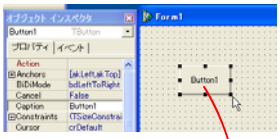
TButton型のButton1という名前のオブジェクト



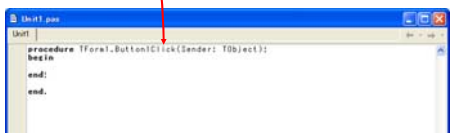
- ボタンを作成するとオブジェクトインスペクタにボタンの情報が表示される。
- 「Caption」の文字列を変更すると、ボタン上の文字列が変わる。

Literacy

ボタンに処理を登録する



- フォームデザイナ上のボタンをダブルクリックすると、コードエディタが現れる。
- コードエディタにはprocedureの先頭部分が書いてある。この名前は変更しないこと。
- このボタンをクリックしたときに実行する処理をこの手続きに記述する。



Literacy

ボタンに貼り付けるプログラムの注意点

- ボタンをダブルクリックして現れる「手続き」の名前は変更しないこと。
- read 文、write 文で文字を入出力する時には、適当な場所に {$apptype console} を書かなければならない。
  - ボタンのあるウィンドウとは別のウィンドウが開くので、そのウィンドウで文字を入出力する。
- Form のウィンドウに文字を出力するときには、別の手続きを使用する。

```
Canvas.TextOut(X, Y:integer; S:string)
```

  - 座標 (X, Y) の位置に文字列Sを表示する。
  - 整数や実数の値をそのまま出力することはできない。
  - 文字の色を変えることができる。(下の代入文)

```
Canvas.Font.Color := 色の名前
```

  - 色の名前はオブジェクトインスペクタのColorの項目を参照する。

Literacy

今までのプログラムをボタンに貼り付ける

```
program sum;
{$apptype console}
uses SysUtils;
var a, b, c : integer;
begin
   write('Two integers :');
   readln(a, b);
   c := a + b;
   writeln('Sum = ', c);
   readln
end.
```

これまでのプログラム

```
uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls;
 ...

implementation
  {$R *.DFM}
  {$apptype console}

procedure TForm1.Button1Click(Sender: Tobject);
var a, b, c : integer;
begin
    write('Two integers :');
    readln(a, b);
    c := a + b;
    writeln('Sum = ', c)
end;
 ...
```

この行が必要

ボタンを使ったプログラム

Literacy

エラーメッセージ・ウィンドウが出た場合

- プログラムに意味的誤りがある場合に起こる。
  - 例えば、プログラム実行中にゼロで割る割り算を実行しようとすると、下のようなメッセージが表示される。

  - まず、このウィンドウの[送信しない]ボタンをクリックする。次に、メニューバーの[実行]→[ステップ実行]を選ぶ。もしくは、[F8]キーを押す。または、下のボタンをクリックする。

  - すると、1行実行して処理が中断する。次に実行する行が反転表示される。ステップ実行のボタンをクリックしながら実行していくと、エラーが発生したところで処理が終了する。

Literacy

7. 今日の練習問題

Level	問題
C	手続き「NN_table」を用いて2×2の表から9×9の表を表示するプログラムを作成せよ。
B	「処理を選択して呼び出すプログラム」を完成させよ。
B	ボタンを押すと、画面の右上に「こんにちは」と表示するプログラムを作成せよ。
A+	$y = f(x)$ の式を与えて $a \leqq x \leqq b$ の間のグラフの概形を描くプログラムを作成せよ。グラフは、$f(x)$の値に応じて「*」を印字する方法で作成できる。

Literacy