1　初めての GUI Application

プログラミング I の授業では Pascal 言語の基本を学ぶために Console Application を作成しました。序論 B では GUI Application [1]を作成します。

1.1　Console Application と GUI Application

Console Application と GUI Application の主だった違い。

	Console アプリケーション	GUI アプリケーション
入出力	コンソール画面（横 80 文字×縦 25 行の黒い画面）を介して行われる 黒地に白い文字だけを書ける	いろいろなコンポーネント（部品）を介して行われる 地にも文字にも色をつけられる 図形もかける
実行	**逐次実行方式**（プログラム内の命令が定められた順番に実行される）	**イベント駆動方式**（マウスを動かす，クリックする，キーインする，一定の時間が経過するなどのイベントによって，それぞれに応じたプロシージャが実行される）
プログラムの必須ファイル	プロジェクト　～.dpr	プロジェクト　～.dpr ユニット　～.pas フォーム　～.dfm

注 プログラムを作成して実行すると次のファイルが作成されます。

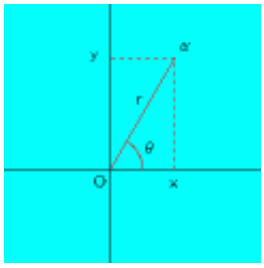
- ○.dpr　必須
- ●.pas　必須
- ●.dfm　必須
- ○.res　削除しないほうがよい
- ○.dof　削除してよい（他の計算機室で実行できないとき，これを削除して開きなおすと実行できることがある）
- ○.cfg　削除してよい
- ○.exe　削除してよい（実行ファイル，完成したらこれさえあれば Delphi がなくても実行できる）
- ●.dcu　削除してよい
- ●.˜pa　pas または dfm を修正したとき，修正前の pas のバックアップ
- ●.˜df　pas または dfm を修正したとき，修正前の dfm のバックアップ
- ○.˜dp　dpr を修正したとき，修正前の dpr のバックアップ

プロジェクト名○とユニット名●は同じ名前をつけることはできません。たとえばプログラム名が Fukusosuu の場合，プロジェクト名を FukusosuuP，ユニット名を FukusosuuU という風にプログラム名の後に P または U をつけて区別することを推奨します。

[1]GUI は Graphic User Interface の略です。

1.2　プログラム Fukusosuu

複素数 α を座標 x, y を用いて $x+yi$ の形で表したものを**直交形式**，原点からの距離 r と偏角 θ を用いて $r\,\mathrm{cis}\,\theta$（すなわち $r\,(\cos\theta + i\sin\theta)$）の形で表したものを**極形式**といいます。両者の間には次の関係があります。



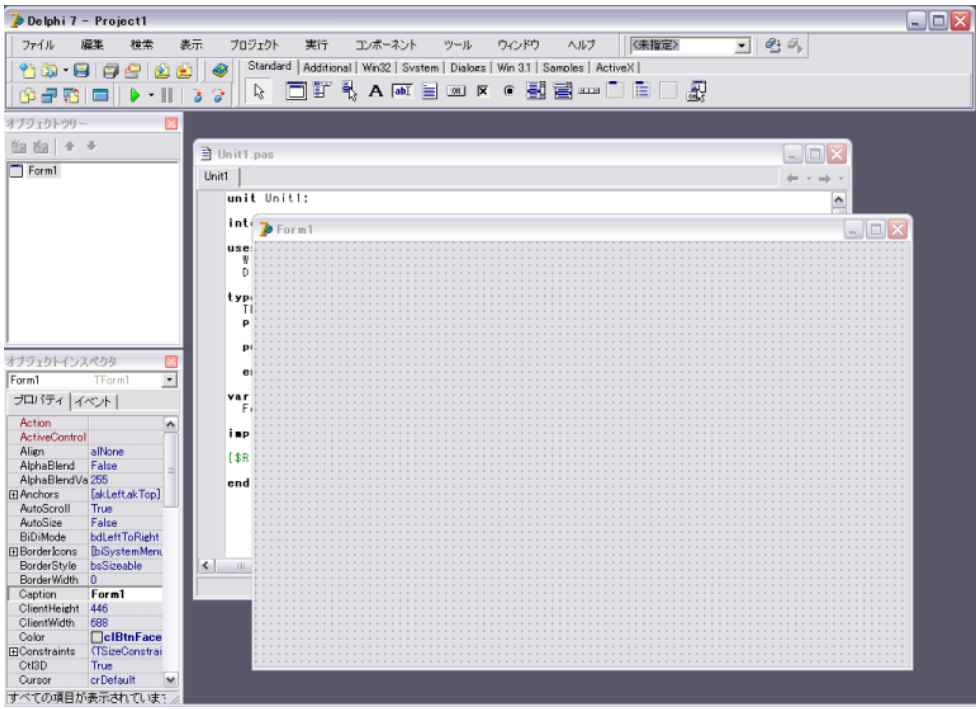
$$r = \sqrt{x^2 + y^2}$$
$$x = r\cos\theta$$
$$y = r\sin\theta$$

複素数を表す 2 つの形式の相互変換（直交形式から極形式，極形式から直交形式）を行うプログラム Fukusosuu を，順を追って作成していきます。ホームページに完成例があります。

1.2.1　新規プログラムファイル作成

1. Delphi を起動します。

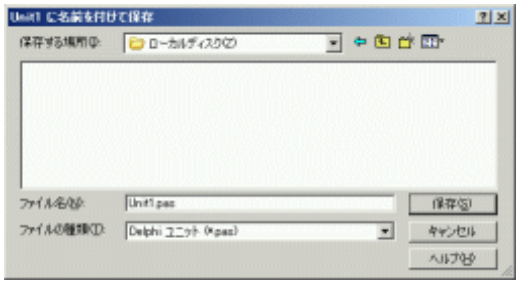
   すでに Delphi で他のプログラムを開いているときは，［ファイル｜新規作成］を行い，アプリケーションを選択します。



   上図はフォームを少し右下に移動してあります（タイトルバーを左クリックしてマウスボタンを押したまま右下にずらすと移動できます）。

   Delphi のタイトルバーにプロジェクト名（Project1.dpr），ユニットのタイトルバーにユニット名（Unit1.pas）が表示されています。

2. ユニットを保存するために,［ファイル｜名前をつけて保存する］を行います。



「保存する場所」に保存したい場所を指定するのですが，まず Z ドライブを指定して，その中に新しいフォルダー JoronB を作り[a]，さらにその中に新しいフォルダー Fukusosuu を作って，それを指定します。
「ファイル名」を FukusosuuU にして，「保存」をクリックします。

[a]「保存する場所」の右に 4 つ並んでいるアイコン（小さな図）の 3 つ目をクリックすると新しいフォルダーが作られます。

ユニットのタイトルバーのユニット名が FukusosuuU.pas に変わったことを確認しましょう。

3. プロジェクトを保存するために,［ファイル｜プロジェクトに名前をつけて保存する］を行います。

   またダイアログが出ますが,「保存する場所」は Fukusosuu になっているはずです。

   「ファイル名」を FukusosuuP にして,「保存」します。

   Delphi のタイトルバーのプロジェクト名が FukusosuuP.dpr に変わったことを確認しましょう。

1.2.2　フォーム設計

フォームの中に必要なコンポーネント（部品）を配置してプロパティ（性質）を設定します。
Delphi 画面の上方，メニュバーの下にコンポーネントのアイコンが並んでいます。

コンポーネントはいくつかのグループに分けられていますが，Delphi 起動時は Standard グループのものが表示されています。グループ名が書いてあるタブをクリックすると，そのグループのコンポーネントのアイコンが表示されます。
アイコンの上にマウスを合わせると，コンポーネントの名前が現れるので，目的のものを探してください。目的のコンポーネントが見つかったら，クリックして選択します。選択した後で，フォームの中の配置したい場所をクリックすると，そこにコンポーネントが置かれます。
大きさや正確な位置など，プロパティを左にあるオブジェクトインスペクタで指定することができます。
では下記の手順に従って設計してください。

1. メインフォーム

   Form1 のプロパティを変更します。

`Name`	`FormMain`	
`Caption`	複素数（直交形式 ⇔ 極形式）	
`Position`	`poScreenCenter`	実行すると中央に現れる

2. メインパネル

   Panel をメインフォームの中央に置きます。

Name	PanelMain	
Caption		PanelMain を消す
Color	clAqua	あなたの好きな色
Height	240	縦のサイズ
Width	320	横のサイズ
Top	100	上の位置
Left	180	左の位置

3. 終了ボタン

   Button をメインパネルの中央に置きます。

Name	ButtonClose	
Caption	終了（&X）	表示は 終了（X）となる

1.2.3　ユニット記入

終了ボタンを配置したので，それをクリックしたときのイベントに対応するプログラム（**イベントハンドラー**）を書きます。

[F12] キーを押すと（または，少し見えているユニットの一部をクリックすると）ユニットが前面に現れます。現在は次のようになっているでしょう。

```
unit FukusosuuU;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;

type
  TFormMain = class(TForm)
    PanelMain: TPanel;
    ButtonClose: TButton;
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  FormMain: TFormMain;

implementation

{$R *.dfm}

end.
```

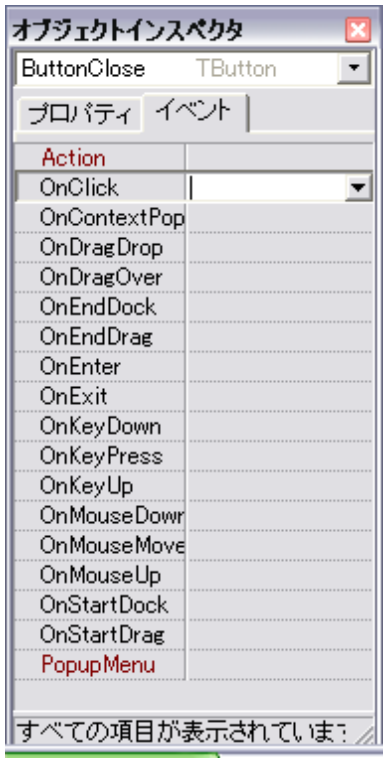
イベントハンドラーは最も下，end. のすぐ上に追加されます。イベントハンドラーと自前のプロシージャがごちゃ混ぜにならないように，記入する場所を分けておきます。

```
（前半省略）
implementation

{$R *.dfm}
(******************** 一般のプロシージャ ********************)
(******************** フォームのメソッド ********************)
(******************** イベントハンドラー ********************)

end.
```

1. オブジェクトインスペクタのウィンドウが ButtonClose になっていることを確認します。もし他のコンポーネントになっていたら， ButtonClose に変更します。
2. 「イベント」と書かれたタブをクリックしてイベント一覧を前面に出します。
3. OnClick の右の欄をダブルクリックすると，ユニットにイベントハンドラーのスケルトン（骨格）が書き込まれます。上の方の TFormMain の定義の中にイベントハンドラーのヘッダー（先頭行）も追加されています。
4. イベントハンドラーの実行部を記入します。今回は，(*_*) を付した Close; の一行だけです。




```
unit FukusosuuU;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TFormMain = class(TForm)
    PanelMain: TPanel;
    ButtonClose: TButton;
    procedure ButtonCloseClick(Sender: TObject);
  private
```

```
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  FormMain: TFormMain;

implementation

{$R *.dfm}
(******************** 一般のプロシージャ ********************)
(******************** フォームのメソッド ********************)
(******************** イベントハンドラー ********************)
procedure TFormMain.ButtonCloseClick(Sender: TObject);
begin
  Close;                (*_*)
end;

end.
```

1.2.4　実行

ここで，とりあえず実行してみましょう。

実行すると，フォームが画面中央に現れます（poScreenCenter にしたので）。

終了ボタンをクリックするか，Alt-X を（Alt キーを押しながら X を）キーインすると終了します（終了ボタンに &X を書いたので）。

1.2.5　設計続行

1. 直交形式用パネル

   Panel をメインパネルの中に置きます。

Align	alTop	メインパネルの上辺に接する
Name	PanelChokkou	
Height	72	
ParentColor	True	メインパネルと同じ色
Font.Name	MS 明朝	MS P 明朝 ではない
Font.Size	14	
Caption	α＝ □□□ x＋ □□□ y i	□□□の前後に半角空白

2. 直交形式の x, y を入力するウィンドウ

   Edit を 2 つ，直交形式用パネルの □□□ が隠れるように置きます。

	左の方	右の方
Name	EditX	EditY
Font.Size	10	10
Text	-1.000	-1.732

3. 極形式用のパネルとウィンドウ

   直交形式用をコピーして修正することにします。

   直交形式用のパネルを右クリックして［編集 | コピー］，メインパネルを右クリックして［編集 | 貼付け］をすると，元の直交用パネルより上にコピーができますから，このコピーの方のプロパティを変更します。

Align	alBottom	メインパネルの下辺に接する
Name	PanelKyoku	
Caption	α＝ □□□ cis □□□ °	

   2 つの Edit の位置を□□□が隠れるように調整して，プロパティも変更します。

	左の方	右の方
Name	EditR	EditT
Text	0.000	0.000

4. 直交から極への変換ボタン

   Button を終了ボタンの左側に置きます。

Height	48	
Width	48	
Name	ButtonCkaraK	Chokkou kara Kyoku の略
Caption	↓	したや を変換する
Font.Size	16	
Font.Style.fsBold	True	

5. ButtonCkaraK の OnClick イベントハンドラー

   オブジェクトインスペクターの OnClick 欄をダブルクリックしてスケルトンを作り，実行部分を書き足します。

```
procedure TFormMain.ButtonCkaraKClick(Sender: TObject);
begin
  ReadChokkou;
  ChokkouKaraKyoku(X,Y,R,T);
  WriteKyoku;
end;
```

6. 自前のプロシージャ

   この中で 3 つのプロシージャが使われていますが，その定義も記入しないといけません。

ReadChokkou	EditX と EditY から x, y の値を読み込む
WriteKyoku	r, θ の値を EditR と EditT に書き出す
ChokkouKaraKyoku	x, y から r, θ を計算する

   最初の 2 つは，FormMain にあるコンポーネントを使っているので，FormMain のメソッドとして定義します。最後の 1 つは，数学の計算をするもので FormMain に関係ないので一般のプロシージャとして定義します。

```
(******************** 一般のプロシージャ ********************)
procedure ChokkouKaraKyoku(X,Y : Real; var R,T : Real);
          { 直交形式 (X,Y) ⇒ 極形式 (R,T) }
begin
  R := Sqrt(Sqr(X)+Sqr(Y));
  T := RadianToDegree(ArcTan2(Y,X));
end; {ChokkouKaraKyoku}

(******************** フォームのメソッド ********************)
procedure TFormMain.ReadChokkou;
          { 直交形式を読む }
begin
  X := StrToFloat(EditX.Text);
  Y := StrToFloat(EditY.Text);
end; {ReadChokkou}

procedure TFormMain.WriteKyoku;
          { 極形式を書く }
begin
  EditR.Text := Format('%6.3f',[R]);
  EditT.Text := Format('%6.1f',[T]);
end; {WriteKyoku}
```

ChokkouKaraKyoku の中でまた数学の関数 RadianToDegree を使っているので，それも定義します。これは，ChokkouKaraKyoku より上に書かないといけません。

```
(******************** 一般のプロシージャ ********************)
function  RadianToDegree(Radian : Real) : Real;
          { 弧度法 Radian ⇒ 60 分法 }
begin
  RadianToDegree := Radian/Pi*180;
end; {RadianToDegree}
```

ChokkouKaraKyoku の中でもう 1 つ数学の関数 ArcTan2 を使っていますが，これは Delphi のライブラリ Math の中で定義されています。ライブラリ Math を使う（uses）ことを宣言しないといけません。uses の最後に追加します。

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Math;
```

変数 X,Y,T,R とメソッドのヘッダーを，FormMain の Private 部に宣言しないといけません。

```
private
  { Private 宣言 }
  X,Y : Real;          // 直交形式
  R,T : Real;          // 極形式
  procedure ReadChokkou;
  procedure WriteKyoku;
```

7. 実行

   実行して，次の複素数の極形式を答えなさい。

$$
\begin{aligned}
-1-\sqrt{3}\,i &= \\
1+i &= \\
-3+\sqrt{3}\,i &= \\
3-4i &=
\end{aligned}
$$

課題　極形式から直交形式への変換もできるようにして完成させなさい。

次の複素数の直交形式を答えなさい。

$$
\begin{aligned}
2\,\mathrm{cis}\,120^\circ &= \\
4\,\mathrm{cis}\,315^\circ &= \\
\mathrm{cis}\,90^\circ &= \\
13\,\mathrm{cis}\,67.4^\circ &=
\end{aligned}
$$