

V i s u a l

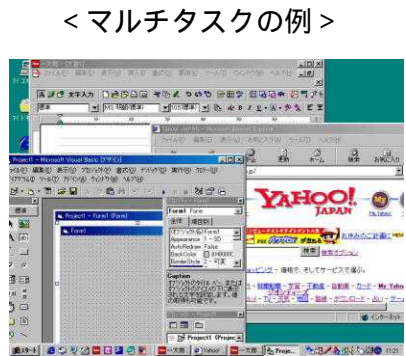
B A S I C

言語実習

# 1 Windowsアプリケーションについて

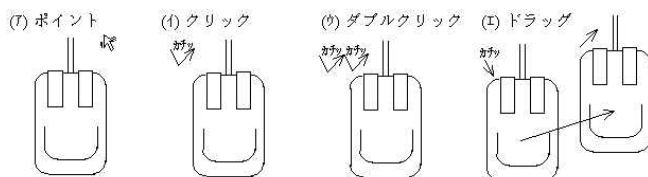
## 1 Windows の概要

Windows はマイクロソフト社が開発したOS（オペレーティングシステム）である。このソフトウェアは、キーボードやハードディスク・プリンタなどのハードウェア（機械部）の働きを制御したり、ワープロや表計算などのアプリケーションソフトの使用環境を整備したり、ネットワークに接続・管理する基本のソフトウェアである。Windows の特徴は、画面を多くの窓（ウインドウ）に分けて、その中ごとに違ったソフトウェアを起動・実行することができる。このように複数のアプリケーションを同時に起動することをマルチタスクと呼び、複数の画面のことをマルチウインドウと呼ぶ。また、現在操作の対象となっているアプリケーションのウインドウをアクティブウインドウと呼ぶ。アクティブウインドウの境界線にポインタを移動すると、矢印に変化しウインドウの大きさを変えることができ、これをサイズ変形ハンドルと呼ぶ。ウインドウの最上部に表示されている領域をタイトルバーと呼び、アプリケーション名が表示される。メニューバーはその下の「ファイル(F)」などの項目がある部分である。また、その下のツールバーは絵（アイコンまたはボタン）で「印刷」などの機能を表している。左下に「スタート」とあるスタートメニューには各種のアプリケーションソフトが登録されている。



## 2 マウスの基本操作

Windows はマウスを使った操作が基本であり、次に示す 4 つのマウス操作だけでメニューの選択やウインドウの移動など、ほとんどの操作をすることができる。



### (1) ポイント（矢印）

マウスを動かして、選択する項目の上にマウスポインタの先端を置くことをポイントという。項目をポイントした後で、次の 3 つのマウス操作を行う。

### (2) クリック

マウスを右手で持ち、ボタンを 1 回押してすぐに離す操作をクリックという。マウスのボタンは左側ボタン（人差し指）と右側ボタン（中指）があるが、通常クリックというときは左ボタンのことである。右ボタンを使うときは通常のクリックと区別し、右という言葉をつけ右クリックという。

### (3) ダブルクリック

ドアをノックをするときのように、マウスのボタンを 2 回続けてすばやくクリックする操作をダブルクリックという。クリックと同様に通常は左ボタンを使う。

### (4) ドラッグ

マウスのボタンを押し、そのまま指を離さずにマウスを移動させる操作をドラッグという。ドラッグを終えるにはマウスのボタンから指を離す。クリックやダブルクリックと同様に通常は左ボタンを使う。なお、ドラッグして目的の場所でマウスのボタンを離すことをドラッグ&ドロップという。

### <練習>

Windows の起動と終了をしてみよう。

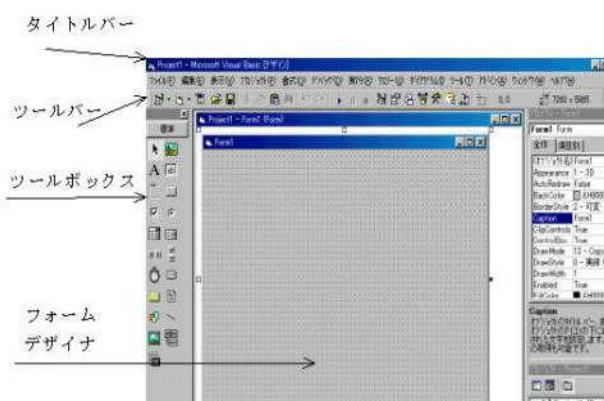
## 2 <sup>ビジュアルベーシック</sup> Visual BASICの概要

従来の BASIC 言語を改良して、さまざまな部品（ツール）を画面（フォーム）上に視覚的（ビジュアル）に配置することができるようにした言語である。そのため、作業者は短時間で簡単に美しい画面を制作することができる。グラフィカルな画面を、マウスなどで簡単に制御できることをグラフィカル・ユーザ・インターフェース(GUI)と呼ぶ。

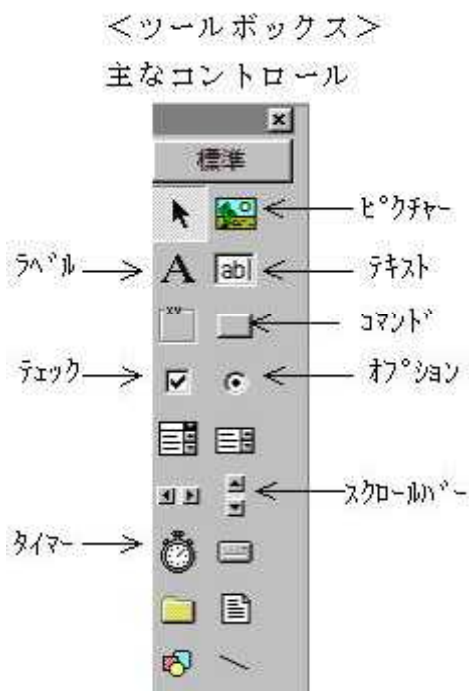
### 1 Visual BASIC の起動

- (1) 「スタート」「プログラム」「Microsoft Visual Basic 6.0」をマウスで選択し「Microsoft Visual Basic 6.0」をクリックして起動する。あるいは、デスクトップ上の「Microsoft Visual Basic 6.0」アイコンをダブルクリックする。
- (2) 新規作成の「標準 EXE」を選び、「開く(O)」をクリックする。

### 2 画面構成



### 3 主なコントロールの機能



コントロール	機能
ピクチャー	絵を表示
ラベル	文字の表示
テキスト	文字の入力と表示
コマンドボタン	押しボタン
チェック	選択
オプションボタン	ひとつを選択
スクロールバー	移動量の選択
タイマー	タイマー時間の設定

### 3 基本プログラミング

#### 1 プログラムの考え方

Visual BASIC (以下VBと呼ぶ)では実行画面の製作から行う。画面(フォームデザイナー)に行いたいこと(コントロール:操縦)を配置してから、何か起きたとき(イベント:出来事)にどうするか(プロシージャ:手続き)を記述し、プログラムとして管理(プロジェクト)して完成させる。このことをイベント・ドリブン型プログラムと言う。

これを専門的に表現すると、フォームに貼り付けたコントロールに発生するイベントにより、プロシージャを実行するプロジェクト製作である。

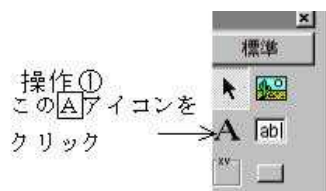
#### 2 簡単なプログラムの作成

(コマンドコントロールボタン(Command1)をクリックすると、ラベルコントロール(Label1)に自分の名前を表示するプログラムを作る)

(1) フォームデザイナーに各コントロールを配置する。

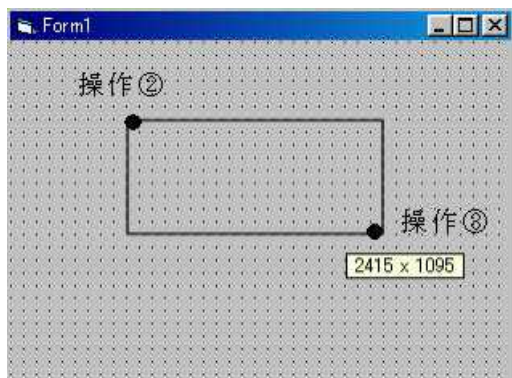
ラベルコントロールを配置する。

<ツールボックス>



操作 ツールボックスのラベルコントロール(A)をクリック(選択)する。

<フォームデザイナー>



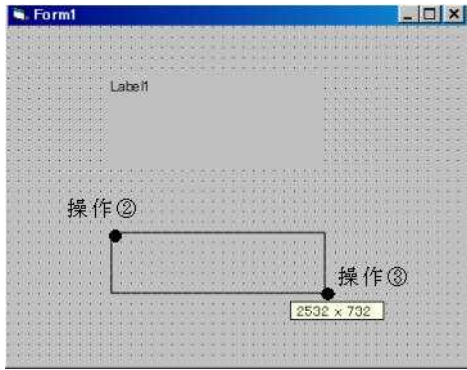
操作 フォーム上の、ラベルコントロールを貼り付けたい位置に、マウスポインタを移動する。(マウスポインタが矢印から十字に変化する。)

操作 選択したポイントから対角に向けてマウスでドラッグする。(左ボタンを離すと、ラベルコントロールが貼りついて、「Label1」と表示される。)

コマンドコントロールボタンを配置する。



操作 コマンドコントロールの四角をクリックする。



操作 コマンドコントロールを貼りつける位置を決める。

操作 選択したポイントから対角に向けてマウスでドラッグする。  
(「Command1」が表示される)

\* コントロールを消去するときには、消去したいコントロールをアクティブな状態にして「デリート ( Delete )」キーを押す。

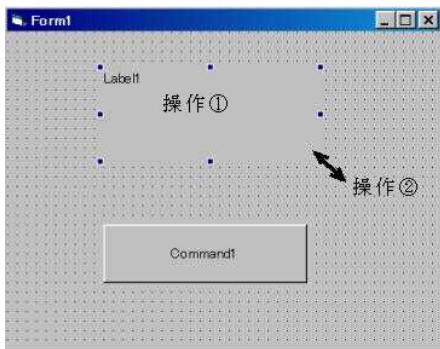
(2) 配置した各コントロールを変更する。  
ラベルコントロールを移動する。

### <フォームデザイナー>



操作 フォームデザイナーに貼り付けたラベルコントロールの上にマウスポインタをおき、ドラッグして好きなところに移動する。

ラベルコントロールの大きさを変える。

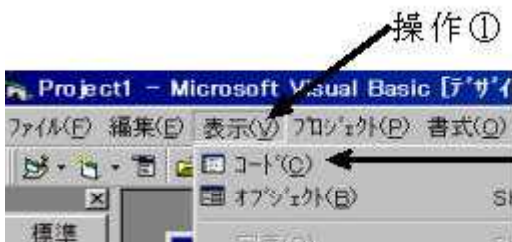


操作 ラベルコントロールをクリックしてアクティブにする。  
(各所に青い点が見れる。これをハンドルと言う)

操作 右下の点 (ハンドル) にマウスを合わせるとマウスの表示が変化する。

操作 この時にマウスをドラッグして大きさを変化させる。

(3) プログラム (コード) の記述 (コーディング) する。  
コードエディタウィンドウ (コード編集画面) にする。



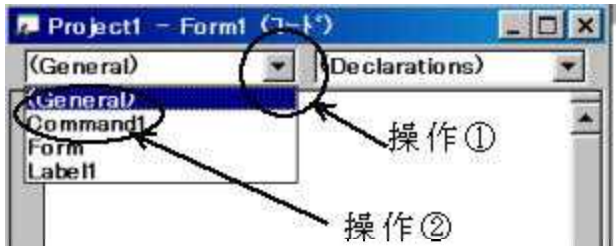
操作 ツールバーの「表示 (V)」をクリックする。

操作 「コード (C)」をクリックするとコードエディタウィンドウになる。

\* フォームデザイナーに戻すには「表示 (V)」「オブジェクト (B)」とクリックする。

コマンド(Command1)プロシージャを呼び出す。

<コードエディタウィンドウ>



操作 下三角形 ( ) をクリックする。

操作 「Command1」をクリックする。

コマンドコントロールボタン1のプログラム(コード)を入力する。

<コマンドプロシージャ>

(オブジェクト名Command1の Clickイベント)

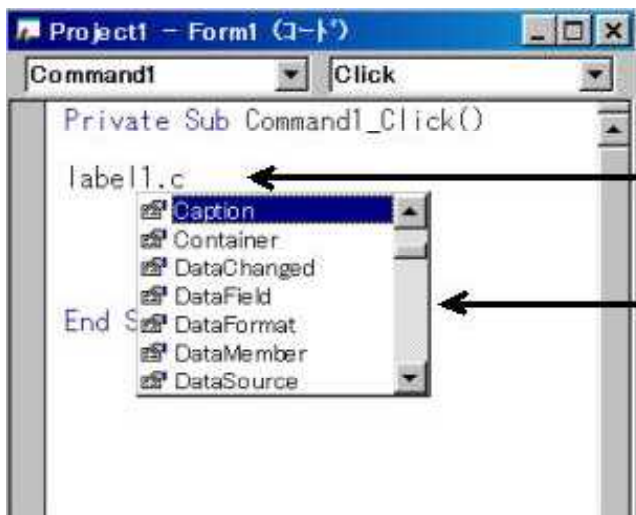


「コマンド1をクリックする」というプロシージャの開始。

この間に、何をするかをコードで記述する。

この行でプロシージャが終了 (End) する。

操作 キーボードで label1. と入力すると、ドロップダウンリストが現れる。  
(入力は半角で行い、小文字でよい。正しく入力された場合は、改行(「Enter」キー)すると頭文字が大文字になる。)



操作①

頭文字を打つと、それに近いものを表示してくれる。

「タブキー (Tab)」を押すと、暗転しているものが選ばれる。

操作 " と " の間に自分の名前を入力する。  
(「ダブルコーテーション ( " )」は「シフト (Shift)」キーを押しながら数字の「2ふ」を押す。)



(4) プロジェクトを実行する。  
 入力したプログラムを実行する。



操作 「実行(R)」を選択して「開始(S)」をクリックする。  
 ( 三角のアイコン (▶) をクリックしても同じである。)

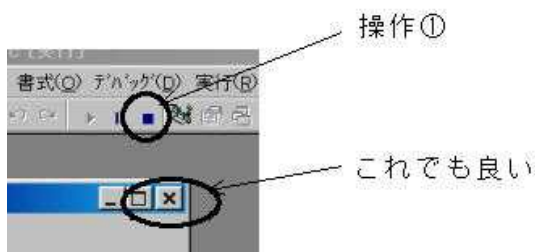
< 実行画面 1 >  
 ( フォームデザイナーにデザインされたウィンドウが表示される。)



< 実行画面 2 >  
 ( Command1 をクリックすると Label1 の表示が変わる。)



プログラムを終了する。



操作 プログラムを終了してコードエディタウィンドウに戻るときは、四角 ( ) あるいは [x] のアイコンをクリックする。

### 3 プログラムの変更・追加

#### (1) 文字と表示枠のサイズ変更をする。

(ラベルコントロールに表示する文字のサイズ変更とラベルコントロール枠のサイズ調整をする)

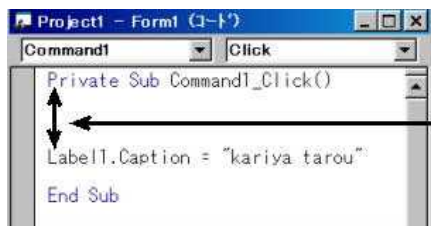
コマンドプロシージャ (Command1 の Click イベント) に文字サイズ(FontSize)を追加する。



操作 「表示(V)」をクリックする。

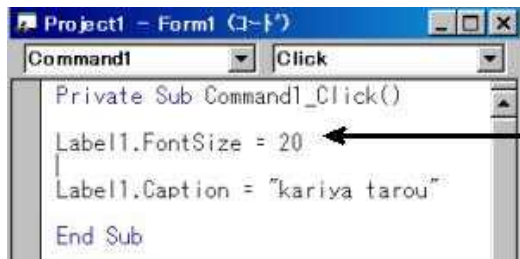
操作 「コード(C)」をクリックする。

#### < Command1 の Click イベント >



操作 Label1 の L にカーソルを合わせて「Enter」を押し、3行ほど改行する。  
(End Sub を改行してもよい。)

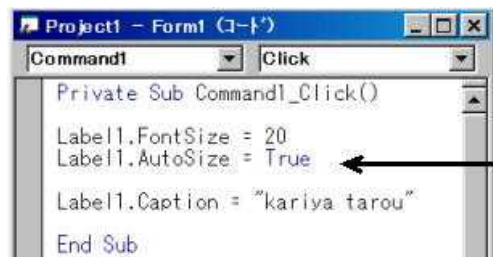
操作③  
Enterキーを  
押して改行する。



操作 フォントサイズ(例:20ポイント)を追加する。

操作 プログラムの実行と終了をして確認する。

ラベルの枠の大きさを自動調整(AutoSize)する。



上記の操作 , を行う

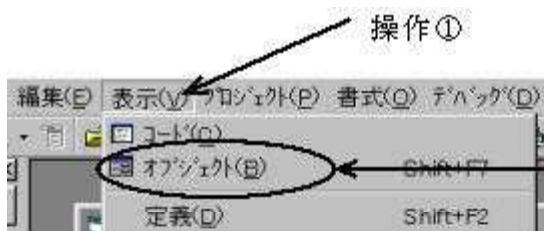
操作 オートサイズの設定(True)を追加する。  
( True : 設定する ( 真の、正の ) )

操作 プログラムを実行して確認する。

#### (2) コマンドコントロールボタン 2 を追加する。

(コマンドボタン(Command2)を追加し、これをクリックするとプログラムが終了するようにする)

フォームデザイナーにコマンドボタン 2 を追加する。

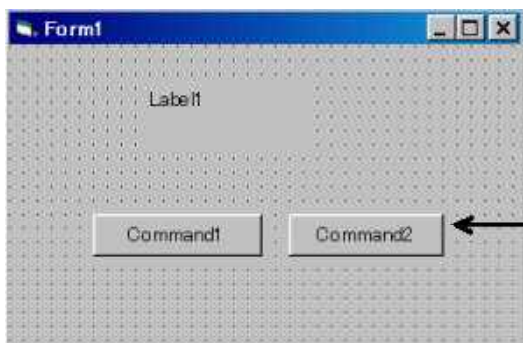


操作 ツールバーの「表示(V)」をクリックする。

操作 「オブジェクト(O)」をクリックする。  
(フォームデザイナーになる。)



< フォームデザイナ >



操作 コマンドコントロールボタン 2 を配置する。

コードを入力する。



操作 コードエディタウインドウに切り換え、 をクリックする。

操作 Command2 をクリックする。

< Command2 の Click イベント >



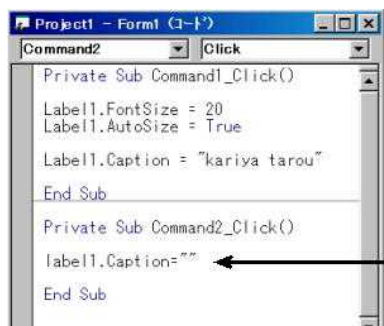
操作 「 End (プログラム終了)」 と入力する。

操作 プログラムを実行して確認する。

(3) コマンドボタン (Command2) の内容を変更する。

(コマンドボタン 2 をクリックすると、ラベルコントロール 1 の表示を消すプログラムに変更する)

< Command2 の Click イベント >



操作 「 End 」を削除し、「 Label1.Caption = "" 」と書きかえる。  
(表示を消すには "" の間に何も書かない。)

操作 プログラムを実行して、確認する。

(4) 実行画面の各コントロールの表示を変更する。

実行画面のラベルコントロールの表示(Label1)を消去しておく。

(プログラム開始時に読み込み( Load )した画面( Form )上に配置した、ラベルコントロール1の表示を消しておく。)

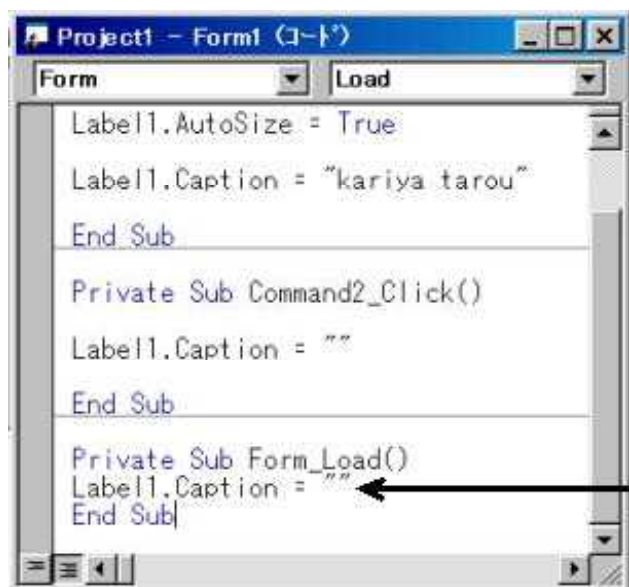


操作 をクリックする。

操作 「Form」をクリックする。

< Form の Load イベント >

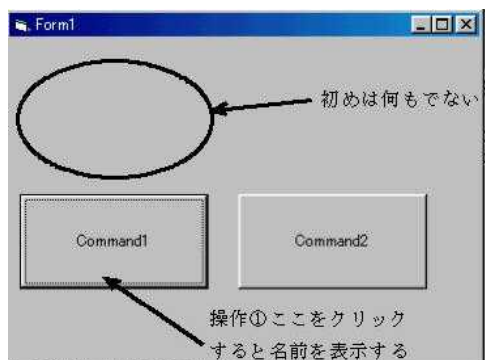
(実行画面( Form )を読み込み( Load )したときの表示を設定する。)



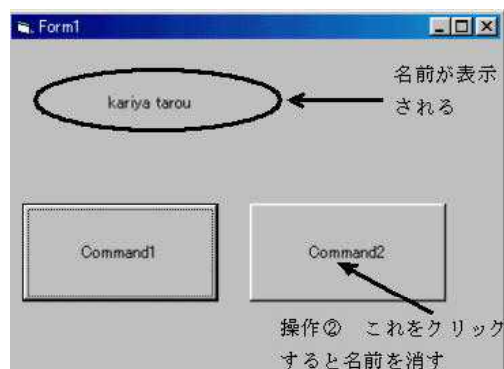
操作 文字を消すプログラムを追加する。

操作 プログラムを実行して確認する。

< 実行画面 1 >



< 実行画面 2 >



< 課題 > 実行画面のコマンドコントロールボタンの表示( Command1, 2 )を変更する。

#### 4 関数を使ったプログラム

##### 1 数当てプログラム 1

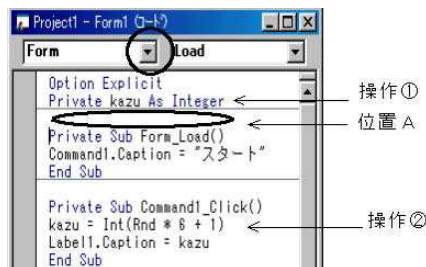
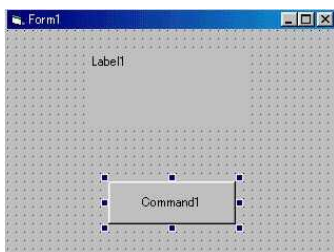
(コマンドボタンをクリックすると1から6までの無作為な数(乱数)を表示するプログラムを作る)

(コマンドボタン1をクリックすると、ラベルコントロール1に1から6までの数字が表示される。)

(1) フォームデザイナーに各コントロールを配置する。

<フォームデザイナーのレイアウト>

<コードエディタウィンドウ>



(2) プログラム(コード)の記述(コーディング)をする。

数字の整数化を行う。

プログラムで扱う数字の整数化を行うため、変数の整数型(Integer)を宣言(定義)する。

(General) (Declarations)

宣言セクション

```
Dim kazu As Integer
```

( Dim の代わりに Private を使ってもよい。)

(位置 A に変数 kazu の定義をすると自動的に上に配置される。)

Rnd関数を使って1から6までの乱数を作る。

[ Rnd関数 - 0以上1未満の数値(0.0001~0.9999)をランダム(Random)に返す。]

1から6までの乱数を作るには、Rnd関数に最大値(6)を掛けて最小値(1)を足し、整数化(Integer)する。

Command1 Click イベント

```
Private Sub Command1_Click()
```

```
kazu = Int(Rnd * 6 + 1)
```

```
Label1.Caption = kazu
```

```
End Sub
```

乱数の乱数系列を再設定し、初期化する。

Form Load イベント

```
Randomize
```

各種の設定を変更する。

Form Load イベント

```
Label1.Caption = ""  
Label1.FontSize = 〇〇  
Command1.Caption = "      "  
Command1.FontSize = 〇〇
```

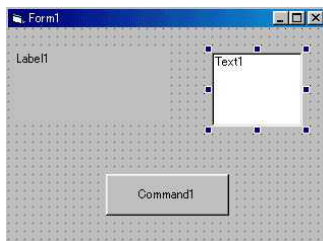
## 2 数当てプログラム 2

(1から100までの乱数を発生させ、その数を当てるプログラムを作る)

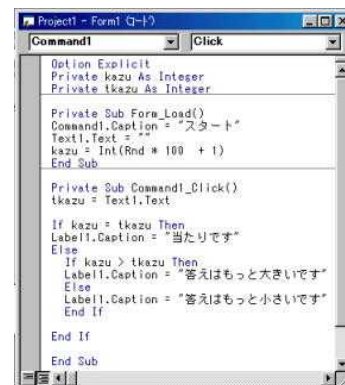
(テキストコントロール1に予想した数字を半角で入力し、コマンドボタン1をクリックすると、その結果がラベルコントロール1に表示される。)

(1) フォームデザイナーに各コントロールを配置する。

<フォームデザイナーのレイアウト>



<コードエディタウィンドウ>



(2) プログラム(コード)の記述(コーディング)をする。

変数 kazu と tkazu を定義する。

(General) (Declarations) セクション

```
Dim kazu As Integer  
Dim tkazu As Integer
```

kazu : 発生させる乱数

tkazu : テキストコントロールに入力する数字

Rnd関数を使って1から100までの乱数を作る。

Form Load イベント

```
kazu = Int(Rnd * 100 + 1)
```

コマンドボタンを押したときに、その数字が当たっていれば“あたりです”、小さければ“答えはもっと大きいです”、大きければ“答えはもっと小さいです”と表示するようにする。

Command1 Click イベント

```
tkazu = Text1.Text  
  
If kazu = tkazu Then  
    Label1.Caption = "あたりです"  
Else  
    If kazu > tkazu Then  
        Label1.Caption = "答えはもっと大きいです"  
    Else  
        Label1.Caption = "答えはもっと小さいです"  
    End If  
End If
```

< IF 文について >

もしも～ならば～をして、  
それ以外のときは～をする文法

```
If < 条 件 > Then  
    < すること >  
Else  
    < すること >  
End If
```

乱数の乱数系列を再設定し、初期化する。

Form Load イベント

```
Randomize
```

各種の設定を変更する。

Form Load イベント

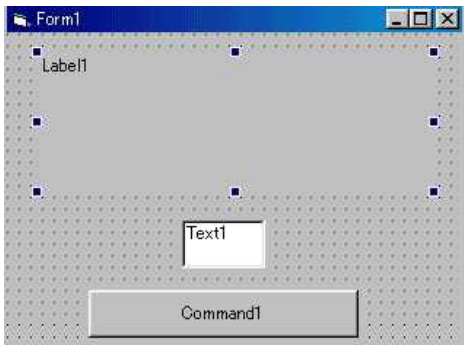
```
Text1.Text = ""  
Label1.Caption = ""  
Label1.FontSize = ○○  
Command1.Caption = "      "  
Command1.FontSize = ○○
```

### 3 血液型占いのプログラム

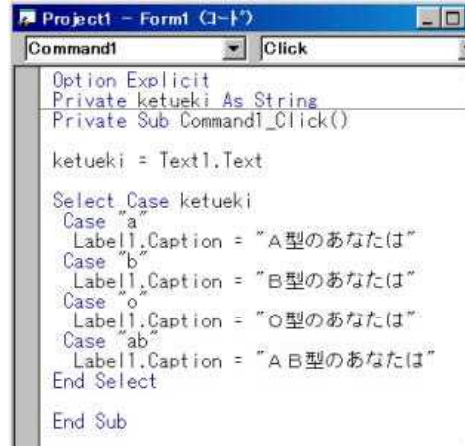
(血液型を入力してコマンドボタンをクリックすると、その性格を表示するプログラムを作る)

(1) フォームデザイナーに各コントロールを配置する。

<フォームデザイナーのレイアウト>



<コードエディタウィンドウ>



(2) プログラム(コード)の記述(コーディング)をする。

変数 ketueki を定義する。

(General) (Declarations) セクション

```
Dim ketueki As String
```

String : 文字列型

テキストコントロールに血液型を入力し、コマンドボタンをクリックするとその血液型の「性格」を表示するようにする。

Command1 Click イベント

```
ketueki = Text1.Text

Select Case ketueki
  Case "a"
    Label1.Caption = "A型のあなたは、……"
  Case "b"
    Label1.Caption = "B型のあなたは、……"
  Case "o"
    Label1.Caption = "O型のあなたは、……"
  Case "ab"
    Label1.Caption = "A B型のあなたは、……"
End Select
```

セレクト ケース  
< Select・Case 文について >

ある特定の変数が、どの範囲なのかによって処理を分岐する。

```
Select Case ketueki
  Case "a"
    <すること>
  Case "b"
    <すること>
    .
    .
End Select
```

各種の設定を変更する。

#### Form Load イベント

```
Text1.Text = ""  
Label1.Caption = ""  
Label1.FontSize = ○○  
Command1.Caption = "      "  
Command1.FontSize = ○○
```

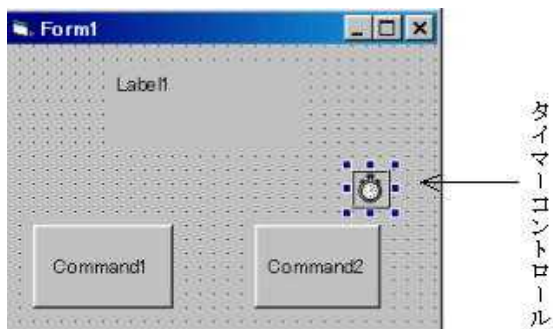
<課題> 血液型の入力をコマンドボタンを配置して入力出来るようにしてみよう

#### 4 スロットゲームのプログラム

(コマンドボタン1をクリックすると1から6までの乱数を 0.2秒間隔で表示し、コマンドボタン2をクリックするとタイマー イベントが停止してその時の数を表示するプログラムを作成する)

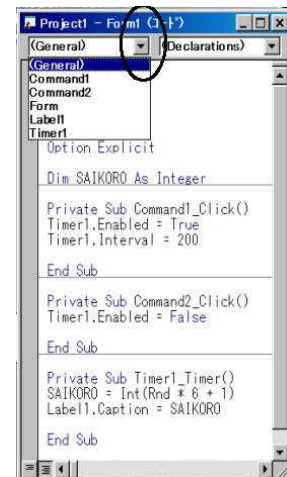
(1) フォームデザイナーに各コントロールを配置する。

<フォームデザイナーのレイアウト>



\* タイマーコントロールは実行画面には表示されないなので、どこに配置しても良い。

<コードエディタウィンドウ>



(2) プログラム (コード) の記述 (コーディング) をする。

変数 saikoro を定義する。

(General) (Declarations) セクション

```
Dim saikoro As Integer
```

コマンドボタン 1 をクリックすると、0.2秒間隔でタイマーイベントを開始するように設定する。

Command1 Click イベント

```
Timer1.Enabled = True  
Timer1.Interval = 200
```

イネーブル  
Enabled : タイマーイベントを開始するか、しないかを設定する。

インターバル  
Interval : タイマーイベントが発生する間隔。  
0.001 秒単位で設定する。

コマンドボタン 2 をクリックすると、タイマーイベントを停止するように設定する。

#### Command2 Click イベント

```
Timer1.Enabled = False
```

フォールス

False : 設定しない (間違った、にせの)

タイマーイベントを開始すると、1 から 6 までの乱数をラベルコントロール 1 に表示するように設定する。

#### Timer1 Timer イベント

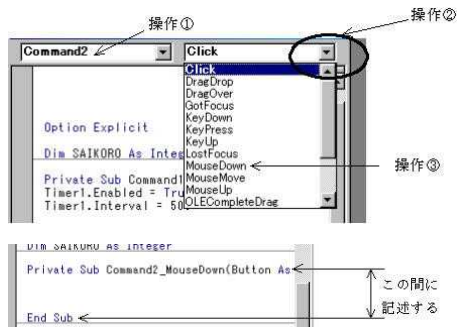
```
saikoro = Int(Rnd * 6 + 1)  
Label1.Caption = saikoro
```

コマンドボタン 2 をクリックして数字を止めたとき、数が「1」なら“当たり” それ以外の時は“はずれ” と表示するように追加する。

#### Command2 Click イベント

```
If saikoro = 1 Then  
    Label1.Caption = "当たり"  
Else  
    Label1.Caption = "はずれ"  
End If
```

コマンドボタン 2 の操作で、「クリック」から「マウスダウン(Mouse Down)」に変更する。  
<コードエディタウィンドウ>



操作 Command2 を選択する。

操作 イベント選択 ( ) をクリックする。

操作 イベントの Mouse Down を選択する。

この間に記述する。

#### Command2.Mouse Down イベント

```
Timer1.Enabled = False  
If saikoro = 1 Then  
    Label1.Caption = "当たり"  
Else  
    Label1.Caption = "はずれ"  
End If
```

操作 「Command2 Click イベント」のプロシージャ  
を  
ドラッグ (暗転させる) して「切り取り」し、  
「Command2.Mouse Down イベント」に「貼  
り付  
け」する。 (カット&ペースト)

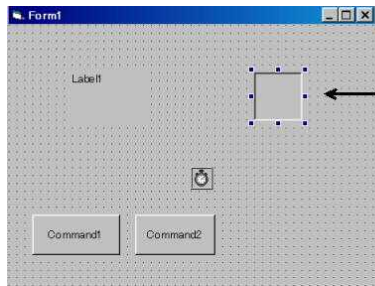


## 5 さいころの画像表示プログラム

(乱数1から6の数に合わせて、さいころの画像を表示するプログラムを作成する)

(1) フォームデザイナーにピクチャーコントロールを追加する。

<フォームデザイナーのレイアウト>



ピクチャー 1 を  
貼りつける

<ツールボックス>



ピクチャー  
コントロール


(2) プロパティウインドウから画像を読み込む。

ピクチャーコントロール 1 にさいころの画像を読み込む。

<プロパティウインドウ>



操作①  
クリックして画  
像を選択する

操作 プロパティウインドウのオブジェクト名が  
Picture1 になっているのを確認する。  
Picture を選び右欄の  をクリックする。

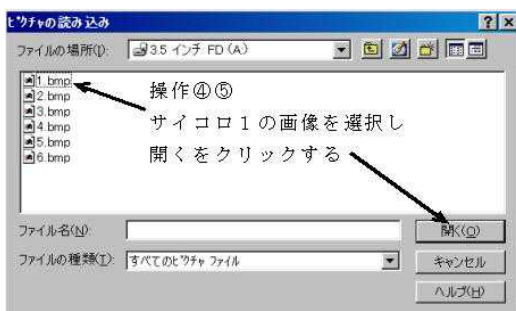


操作②  
▼をクリック

操作③  
FD を選択

操作 ▼ をクリックし 3.5 インチ F D  
(A) を選択する。

\* フロッピーディスクを入れておく。

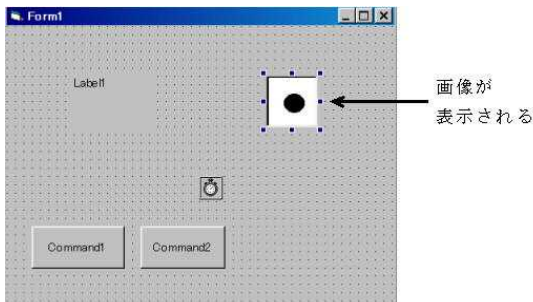


操作④⑤  
サイコロ 1 の画像を選択し  
開くをクリックする

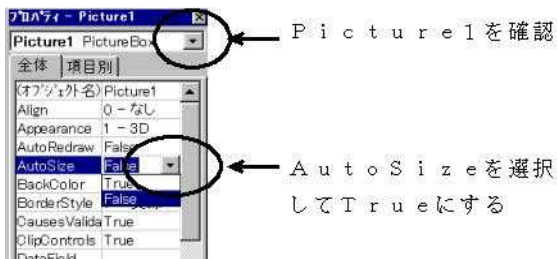
操作 「1.bmp」(さいころの目 1 の画像) をクリック  
する。

(「1.bmp」が暗転する。)

操作 「開く(O)」をクリックする。

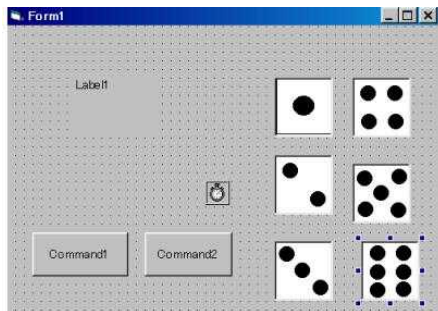


さいころの目の画像がピクチャーコントロール1に表示される。



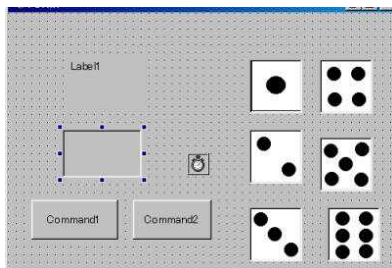
操作 プロパティウィンドウで  
AutoSizeを設定(True)する。

ピクチャーコントロール2～6に、さいころの画像を読み込む。



操作 ~ の作業を、ピクチャーコントロール2から6についても繰り返す。  
表示したい画像を、あらかじめフォーム上に別に貼り付けて(メモリに配置される)おき、それを読み込むようにすると画像の表示を早くすることができる。

さいころ表示用のピクチャーコントロール7を貼り付ける。



操作 さいころ表示用のピクチャーコントロール(Picture7)を貼りつける。

(ピクチャー7に、さいころの画像をスロットーンのように表示する。)

マシ

(3) プログラム(コード)の記述(コーディング)をする。

乱数 1 から 6 に対応して、ピクチャーコントロール 7 にさいころの画像表示をするようにする。

Timer1 Timer イベント

```
saikoro = Int(Rnd * 6 + 1)

Select Case saikoro
  Case 1
    Picture7.Picture = Picture1.Picture
  Case 2
    Picture7.Picture = Picture2.Picture
  Case 3
    Picture7.Picture = Picture3.Picture
  Case 4
    Picture7.Picture = Picture4.Picture
  Case 5
    Picture7.Picture = Picture5.Picture
  Case 6
    Picture7.Picture = Picture6.Picture
End Select
```

（セレクト ケース  
Select・Case 文を使い、  
乱数 1 から 6 に対応して  
ピクチャー 7 に画像表示  
するように変更する。）

(「Label1.Caption = saikoro」を消去し、「Select Case 文」を追加する。)

「Case 1  
Picture7.Picture = Picture1.Picture」  
と入力したら「コピー」し、必要  
な回数「貼り付け」で数字の部分  
だけ訂正する。

(4) 画像をカスタマイズする。

(「ペイントブラシ」を使って、さいころ 1 の画像を書き換える)

「ペイントブラシ」を起動する。

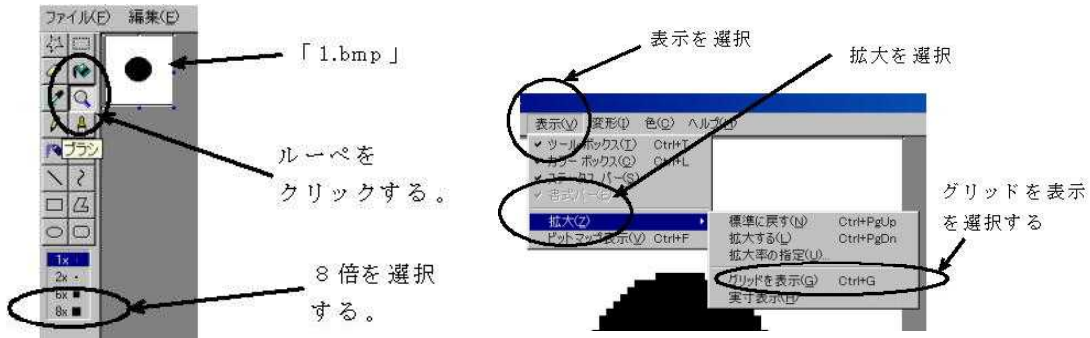
スタートメニューの「スタート」「プログラム」「アクセサリ」とマウスで選択し「ペイント」をクリックする。

「ペイントブラシ」にさいころ 1 の画像を読み込む。

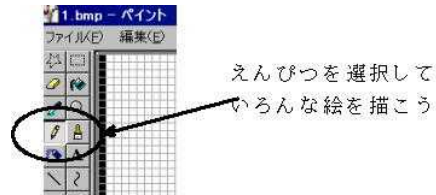
「ファイル(F)」「開く(O)」「」とクリックして「3.5 インチ F D (A)」を選択し、続けて「1.bmp」「開く(O)」とクリックして、さいころ 1 の画像を読み込む。



画像を拡大（8倍）し、グリッド表示にする。



えんぴつ・ブラシなどを使って、  
画像を自由に書き換える。



(5) 作成したファイルの保存

「ファイル(F)」「名前を付けて保存(A)」を選び、「ファイル名(N)」に  
自分の生徒番号（半角、例：E2101-1）で保存する。

< 課題 >

さいころの他の目も変更してみよう。