Visual Basic.NET[1] による応答的学習プログラム
――英語構文の学習に向けて――

門田　幸太郎*

本稿では英語構文の学習を想定して，応答的プログラムの作成とその解説を行なった。プログラムユーザーである学習者が問題として示された日本語文に対応する英文を１文字ずつ入力するたびに正誤が判定される。誤ることなく最初から正答を入力した場合は黒字，誤った後に自力で正答を入力した場合は緑字，誤った後にヒントボタンをクリックした場合は赤字，自分で入力することなく最初からヒントボタンをクリックした場合は青字で，それぞれ表示する。これにより，学習者は自分の考えた答案の正誤を即時に知ることができると同時にその履歴も知ることができる。また，表示された結果の全体を見ることによって，自分の理解度を知ることもできるようになる。

キーワード：応答的プログラミング，Visual Basic.NET，フィードバック，英語構文の学習

１．応答的プログラムの基本構想

学習活動における内発的な動機づけ（intrinsic motivation）を高めるためには応答的環境が重要であることが指摘されてから久しい（波多野誼余夫・稲垣佳世子；1973，1977, 1981）。ここでいう応答的環境とは，学習者の求めに応じて，学習を促す情報を提供する環境や，学習者の反応に対して，できるだけ時間間隔を置かずに，正誤の情報を肌理細かく与える環境を意味する。学習を促進させる役割を人間が担う場合，学習者と１対１で付きっ切りならば，学習者の質問に答えたり，誤りを指摘したりして，ある程度応答的環境を作ることができる。しかし，学習者の意向に合わせて，長時間，常時，この環境を用意するのは非常に難しい。これに対して，学習を促進させる役割をコンピュータが担う場合，学習者の意向に合わせて，何時間であろうと，恒常的に，応答的環境を提供することができる。また，付随する事柄として，学習者の相手を人間がする場合，学習者が同じような質問や誤りを繰り返すと，そのことが教授者にとってストレスとなり，教授者が感情的になることも充分考えられる。そしてまた，教授者が感情的になることが学習者にとってストレスとなり，さらに学習を妨げる要因になるという悪循環が起こりやすくなる。しかし，コンピュータ

＊立命館大学産業社会学部教授

の場合，同じ質問を繰り返しても，何度間違っても，このようなストレスからはフリーである。もとより，コンピュータを用いて教授できることには限界があり，利用方法に慎重を期すことが必要であるが，応答的環境という意味では，コンピュータの備えている機能は大きな可能性を持っているといえる。

一般に，学習過程において，学習課題に対する反応である答案とその正誤を確認するまでの時間が短いほど，学習は成立しやすくなる。学習心理学でいうところの随伴性（contingency）の問題である。また，自分の起こした行動の結果にかんする情報は KR（Knowledge of Results；結果の知識）ともいわれ，KR を本人に返すことはフィードバック（feedback）といわれる。答案が作成されると，時間間隔を置くことなく，KR がフィードバックされるなら，自分の起こした行動（反応）とその結果との結びつきが容易になり，学習を促進させることができる。特に，学習過程において，知識の定着が不充分で，試行錯誤的に自分の持っている知識の正誤を確かめようとする段階では，このような応答的な環境が重要である。学習を促す情報をヒントとして与えたり，答案の正誤を即時に返すと同時に，誤反応に対しては，どのように誤ったのかを履歴情報として与えたりする機能を持たせることによって，応答的環境としてのコンピュータを活用する可能性を探ってみた。そのような応答的な機能を備えているプログラムを，VisualBasic.NET を用いて作成することが本稿の目的である。

学習課題としては，英語の構文の習得というような，定型的知識の獲得を取り上げる。学習者は，問題として表示された日本語文に対する答案として英文を１文字ずつ入力する。問題が表示された時点での学習者の反応としては，２通りの反応が考えられる。正誤はともかく，自力で答案を思いつく場合と，まったく思いつかない場合である。前者では，自分が正しいと思う１文字を答案として入力することになり，後者では，自分で答案を入力することなく，ヒントボタンをクリックすることによって正解を表示させることになる。また，前者の場合，答案が正答の場合と誤答の場合とが考えられる。入力された答案が，想定された正解と一致している場合には正答とし，その１文字を黒字で表示する。答案が正解と一致していない場合には誤答とし，ただちに，誤りであるというメッセージが表示され，新たな反応が求められる。この場合，新しい反応（答案）の内容によって，3種類のケースが考えられる。①新しい答案が再入力され，それが再び誤答である場合，②新しい答案が再入力され，それが正答である場合，③新しい答案が再入力されず，ヒントボタンがクリックされることによって正解が表示される場合である。図１にフィードバックの方法を含めたジョブの流れの概念図を示す。

前述の各ケースに対する応答的なフィードバックとして，次のようなフィードバックが与えられる。①では，誤りであるというメッセージが再び現れて，改めて答案を入力するように求められる。この手続きは，その後，正答が入力されて②となるか，ヒントボタンがクリックされて③となるかのどちらかになるまで繰り返される。②では，学習者が自力で正解にたどりついたことになる。この場合，RichTextBox1 に正解となった１文字を緑色で表示する。③では，学習者が自力で正解にたどりつけずに，ヒントボタンをクリックしたことになる。この場合，RichTextBox1 に正

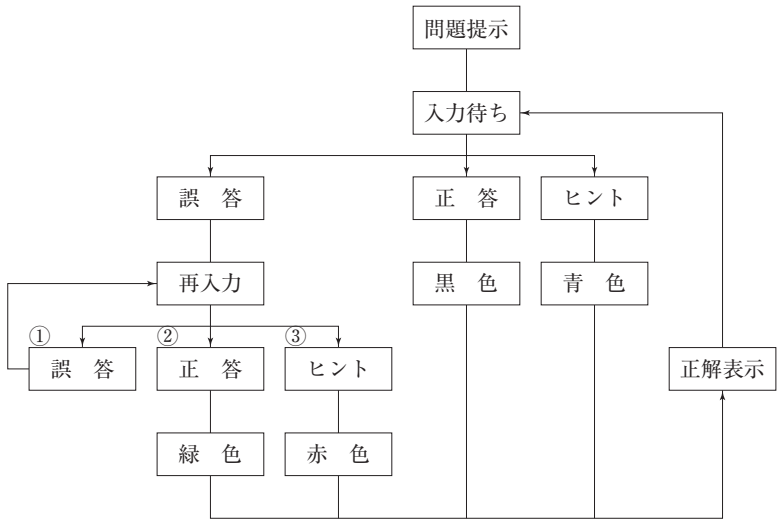

図 1　ジョブの概念図

解の 1 文字を赤色で表示する。ある問題が終了した時のフィードバックの例を図 9 に示す。このように，自分の反応が正反応なのか誤反応なのか，また，誤反応をした場合，どのような経過をたどったものなのかの情報を視覚的に提示することができる。それによって，学習者は，どの部分が理解不充分なのか，どこでミスをしたのかを容易に確認することができる。また，RichTextBox1 に表示されたすべての文字の色によって理解の程度を判断することができる。表示文字に，赤色や青色の文字が多い場合は理解がまだ不充分だということを示しており，緑色が多くなってきた場合は理解が進んできていることを示している。黒色の表示文字が多い場合は理解が定着してきたことを意味する。全文が黒字になった場合は，学習が一応のレベルに到達したといえる。

2．応答的プログラムの作成

VisualBasic.NET では，プログラムを構成する単位としては，下位から上位に向かって，ステートメント，プロシージャ（または，サブプログラム），モジュール，プロジェクト，ソリューションと分かれる。下位の単位は上位の単位の構成要素となっている。たとえば，複数のプロシージャが集まってモジュールを構成したり，複数のモジュールの集合がプロジェクトを構成したりしている。大規模なシステム設計のように複雑な課題の場合，ソリューションが複数のプロジェクトから構成されることもあるが，ここでは単一のプロジェクトからなるものを扱う。プロジェクトは，遂行するアプリケーションを管理する基本単位である。プロジェクトはさらに，フォーム（.frm），標準（.bas），クラス（.cls）などのモジュールに分けられる。ここでは，標準モジュールとクラスモジュールは VisualBasic.NET のシステムが作成するディフォルト（既定値）を用いることとし，プログラムの中心となるフォーム・モジュールについて取りあげる。モジュールはさらに，プロシージ

ャ単位に分けることができる。各プロシージャはPrivate Subで始まり，End Subで終わるサブルーチン・プロシージャが中心となり，さらに関数の値を返す関数プロシージャも含まれる。

本プログラムの構成はPublic変数の宣言と７つのプロシージャからなる。以下に，それぞれのステートメントを掲げ，プログラムの意図と働きについて述べる。中核をなす**「２－４　正誤判定のためのプロシージャ」**については，フローチャートも含めて，詳しく述べる。

２－１　Public変数の宣言にかんするステートメント

```
'問題として表示された日本語文に対応する英文を RichTextBox1 に入力する。
'答案が１文字ずつ入力されるたびに正誤を判定し，間違いが出た時には，チェックできるようにせよ。
'ヒントボタンをクリックするたびに，正解の１文字ずつ追加表示していく。
'１つの問題が終了した時には，「Good!」の文字を点滅表示する。
Dim i As Integer '日本文と英文をセットにしたファイル上の位置
Dim SrchStr(100, 2) As String 'データ入力用の配列。SrchStr(i, 1)= 日本語文。SrchStr(i, 2)= 英文
Dim ltr = 1 'RichTextBox 1 に入れられた文字数の初期値
Dim ErrorCase As Integer '入力された文字が間違っている時には ErrorCase=1 とする。
Dim hintbtn As Integer 'ヒントボタンのクリックされた回数。
```

各Private Subは互いに独立したものであり，各サブプログラムでの変数は，たとえ同一の変数名であろうとも，お互いに無関連に機能するようになっている。本プロジェクトでは，全部で大小７つのサブプログラムで構成されている。しかし，時として，本プロジェクトのように，異なるサブプログラム間で，同一の変数名に関連性を持たせたい，つまり同一の値を共有させたいという場合がある。そのためには，Public変数の宣言が必要となる。Public変数の宣言は各サブプログラムに先立って行なわれなければならない。これにより，以下のサブプログラムで現れる変数は互いに値を共有するという関連性を持つことになる[2)]。

上に示したPublic変数の宣言にかんするステートメントは２つの部分から構成されている。前半の４行のステートメントはコメント文であり，後半の５行からなるステートメントはPublic変数宣言文である。コメント文は「'」（シングル・クオテーション）をつけることにより，それ以後，つまり「'」マークの右側に書かれたものを注釈文として扱うことになる。

Dimで始まる後半の５行は変数の型を宣言するものである。他のプロシージャの前の領域にこの宣言文を置くことによって，その有効範囲（スコープ）はすべてのプロシージャで有効となり，その値が共有される。「Dim i As Integer」は変数名iを整数として宣言することを意味するステートメントである。iはいわば，問題番号にあたる数値である。「Dim SrchStr(100, 2) As String」は問題となる日本語文と正解となる英文を入れるために100行２列の文字変数型の２次元配列を作成す

る。第 1 次元の値は問題と正解の対の順番を意味する。第 2 次元の値が 1 の場合，問題となる日本語文を，2 の場合，正解となる英文を意味する。「Dim Itr = 1」は変数名 Itr の変数型をディフォルトである整数型とし，その値を 1 とする宣言である。Itr は，RichTextBox1 に表示する文字数を意味する。「Dim ErrorCase As Integer」は変数 ErrorCase を整数型として宣言している。変数宣言された場合，その値は初期化されて 0 となる。ErrorCase は，答案として RichTextBox1 に書き込まれた文字が間違っている時に 1 となる。「Dim hintbtn As Integer」も変数名 hintbtn を整数型として宣言されている。hintbtn は，ヒントボタンがクリックされているかどうかを示す変数であり，クリックされた時に 1 が加算される。

2－2　データ読み込みのためのプロシージャ

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
  ’問題データと正解データの読み込み
  FileOpen(1, "060129 英作文 .txt", OpenMode.Input)
  Do While Not EOF(1)
    Input(1, SrchStr(i + 1, 1)  ’日本文の読み込み
    Input(1, SrchStr(i + 1, 2)  ’英文の読み込み
    i = i + 1
  Loop
    FileClose(1)
  MsgBox（" データの読み込みが完了しました。" & vbCrLf & "「問題」をクリックしてください。", _
  MsgBoxStyle.OKOnly + MsgBoxStyle.Information, " さあ始めましょう！ "）
  Label3.  Text = " 問題の数 " : TextBox2Text = i & " 問 "
  i = 0
End Sub
```

上に示した，「Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)Handles MyBase.Load」は，問題データとなる日本語文と正解データの英文のペアを読み込むためのサブプログラムである。(ByVal sender As System.Object, ByVal e As System.EventArgs) は，プロシージャ間の引数を渡す方法を指定するものであるが，ここでは，あらかじめ VisualBasic.NET のシステムによって設定された System.Object と System. EventArgs が用いられることになる。これによって，Sub ステートメントや Function ステートメント，Property ステートメントなど各プロシージャで用いられる基本的なステートメントが有効となってくる。Private Sub Form1_Load は，プログラムを実行させるデバグ・コマンドのクリックと同時に実行されるこ

とになる。

FileOpen 関数の構文は，

```
FileOpen（ファイル番号，ファイル名，オープンモード）
  実行文
FileClose（ファイル番号）
```

となっている。ここでは VisualBasic.NET のシステムが自動的に作成する Bin フォルダーに格納されたファイルを使用している。Bin フォルダーのファイル番号は１と設定されている。ファイル名は，読み込むべきデータが格納されているファイルの名前である。ここでは，"060129 英作文.txt" としている。オープンモードはファイルからの読み込みやファイルへの書き込みを設定するものである。ここでは，OpenMode.Input を指定している。これは，読み込み専用のモードで，テキストファイルのデータを先頭から順番に読み込むためのものである。これにより，問題文となる日本語文と正解となる英文とが読み込まれることになる。データの構造としては，文末に必ず，半角のスペースを入れておくようにする。これは，データの文字数をカウントするために行われるものである。FileClose（ファイル番号）はオープンしたファイルを閉じるためのものであり，FileOpen 関数と常に対置されるものである。このファイル番号は FileOpen 関数のファイル番号と同一の番号を指定する。

Do While の構文は

```
Do While 条件式
  実行文
Loop
```

となっている。これは，条件式が成立している限り，実行文を繰り返し実行するというものである。EOF は End Of File の略で，ファイルの最後に来た時に真値１を取る関数である。ここでは，EOF の前に否定の Not がつけられているので，指定されたファイル番号のファイルの末尾にならない限り，実行文を繰り返すことを意味する。実行文には２つの Input 関数と i のインクリメントが実行される。「Input(1, SrchStr(i + 1, 1)」関数は，シーケンシャル入力モードで開いたファイルからデータを読み込んで，それを格納する変数を指定するものである。その構文は

```
Input（ファイル番号，変数名）
```

である。ここでは，先の FileOpen 文で指定したファイル番号１のファイルから，データを読み込み，それを SrchStr(i + 1, 1) に格納させている。SrchStr は Public 領域で宣言された２次元の配列で，(100,2) の構造を持っている，100行２列の２次元で，行番号は日本語文と英文のセットの番号で，いわば，問題番号に相当するものであり，列番号は１列目が問題文となる日本語文，２列目が正解となる英文に対応している。

実行文の１行目の「Input(1, SrchStr(i + 1, 1)」 は，ファイル番号１から読み込んだデータを SrchStr(i + 1, 1) に格納することを意味する。i の初期値は０であるので，ファイルの最初の日本語

文を SrchStr(1,1) に納める。次に，実行文の２行目の「Input(1, SrchStr(i + 1, 2)」は，ファイル番号１から読み込んだデータを SrchStr(i + 1, 2) に納めることになる。実行文の３行目の「i = i + 1」は i を１ずつ増加させるためのものである。右辺の結果を左辺に代入させるというのが等式の意味である。右辺の i は０であるので，０＋１の結果である１が左辺の i に代入され，i の値は０から１へと変化する。この３行の実行文を EOF，つまり，ファイルの末尾になるまで繰り返すことになる。配列に格納されたデータを例示すると以下のようになる。

SrchStr(1, 1) ← " お正月がこんなに盛大に祝われるのは日本だけだ。アメリカでは，人々は１月１日には仕事を休むだけで，これといった特別の料理を食べるわけではない。"

SrchStr(1, 2) ← "It is only in Japan that New Year’s Day is celebrated on such a grand scale. In the United States, people rest from work on January 1,　but they do not eat any special dishes to speak of on that day. "

SrchStr(2, 1) ← " 泊まり客のうちから赤痢患者が出たとき，伝染を恐れてホテルの建物全体は徹底的に消毒された。"

SrchStr(2, 2) ← "For fear of contagion, the entire hotel building was thoroughly disinfected after a case of dysentery was discovered among the guests. "

SrchStr(3, 1) ← " 資金不足のために体育館の建設工事は土台だけで，それから先は続けられなかった。"

SrchStr(3, 2) ← "For lack of funds, the construction of the gymnasium was discontinued after only the foundations had been laid. "

…（ 以下省略 ）

データの読み込みが終了すると，ファイルが閉じられる。その後，データの読み込みと開始方法を指示するメッセージが表示される。

MsgBox はメッセージ・ボックスを表示するためのコマンドである。その構文は

```
MsgBox（" メッセージ "，ボタンとアイコンの指定，" タイトル "）
```

である。したがって，ここでの MsgBox 関数の意味は次のようになる。「データの読み込みが完了しました。」と「「問題」をクリックしてください。」が，メッセージ・ボックスのコメント文として，２行に分けて表示される。& vbCrLf & は改行のオプションである。MsgBoxStyle.OKOnly はメッセージ・ボックスに表示されるボタンのキャプションをＯＫとすることを意味し，MsgBoxStyle.Information は，メッセージ・ボックスのタイトル・バーに「さあ始めましょう！」という表示をすることを意味する。図２に，データの読み込みが完了した時のメッセージ・ボックスを示す。

図３に示された，読み込み完了時の実行画面において，「 Label3.Text = " 問題の数 " : TextBox2.Text = i & " 問 "」の「Label3.　Text = " 問題の数 "」は，Label3 に「問題の数」と表示し，「TextBox2.Text = i & " 問 "」は，問題が全部で何問であるかを示すためのものである。データ

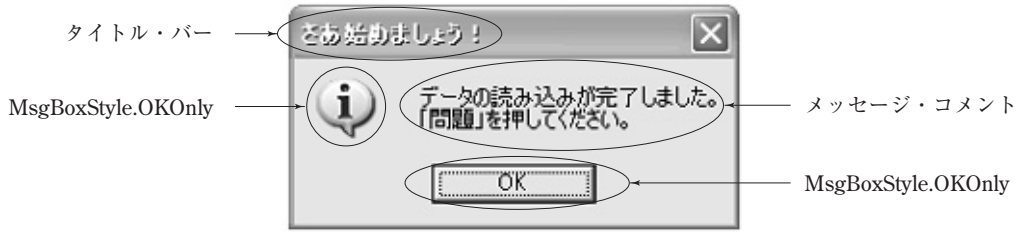

図２　読み込み完了のメッセージ・ボックス

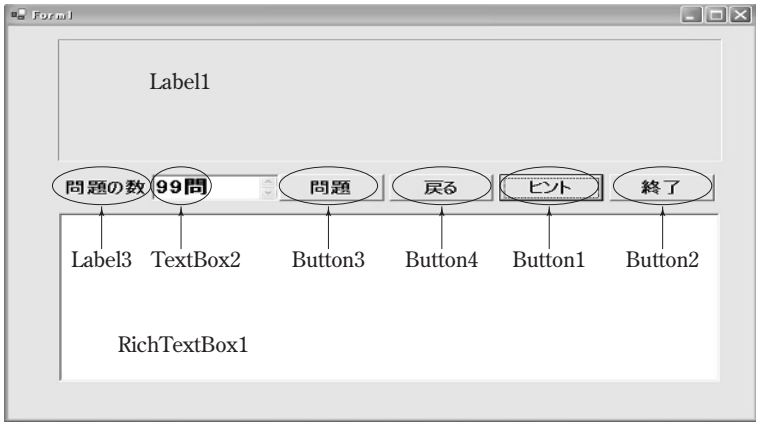

図３　読み込み完了時の実行画面と各オブジェクト名

を最後まで読み込んだ時の i の値と「問」という文字を連続表示する。「:」は，1行に２行分の実行文を書く時に用いるものである。本来，VisualBasic.NET では１行は１ステートメントで構成される。あえて，１行に２つのステートメント書き込む場合は，間にコロン記号を入れなければならない。最後に，「i = 0」で i を初期値に戻しておく。これは，次に想定される「問題」というキャプションが付いた Button3 がクリックされた時に，１番目の問題から提示させるための準備である。このサブプログラムの最後に，「End Sub」として，この Private Sub Form1_Load のプロシージャを終了させる。

２－３　問題表示のためのプロシージャ

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.  Click
  ' 問題表示
  Timer1.Enabled = False
  PictureBox1.Visible = False
  Label3.Text = " 文の長さ "
  ErrorCase = 0
```

```
    ltr = 1
    hintbtn = 0
    i = i + 1
    If Len(SrchStr(i, 2)= 0 Then
      MessageBox.Show（" これ以上はありません。" & vbCrLf & "「戻る」か「終了」をクリック
      してください。"）
      i = i - 1
      Exit Sub
    End If
    Label1.Text = "(" & i & ")" & SrchStr(i, 1)
    RichTextBox1.Text = ""
    TextBox2.Text = Len(SrchStr(i, 2)& " 文字 "
    RichTextBox1.Focus()
End Sub
```

「Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)Handles Button3.Click」は，問題となる日本語文を表示するためのサブプログラムである。これは，「問題」というキャプションが付けられた Button3 をクリックした場合に実行されるものである。

「Timer1.Enabled = False」は，タイマーの実行を一時停止させるためのものである。これは，後述の Private Sub Timer1_Tick のサブプログラムでタイマー利用が想定されているので，その実行を停めるものである。「PictureBox1.Visible = False」は Timer1 による PictureBox1 の表示を一時停止するためのものである。「Label3.Text = " 文の長さ "」は，Form1_Load の時には「問題の数」というキャプションが付いていた場所に，問題となる英文の長さを文字数で示そうとするものである。「ErrorCase = 0」，「ltr = 1」，「hintbtn = 0」とも初期化にかんするものである。ErrorCase は RichTextBox1 に入力された答案が正しい場合は 0 のままだが，間違っている場合は 1 となる。これによって，答案の正誤の情報が保持される。ltr は答案の何文字目を入力し，添削しているのかを示すものであり，ltr = 1 は，その初期値を設定するものである。hintbtn は，ヒントボタンがクリックされているかどうかという情報を保持するためのものであり，クリックされた場合，1 に設定されることになる。これらの 3 変数は，共通変数なので，その値は他のサブプログラムでも共通のものである。「i = i + 1」の i は前述のように，Form1_Load で読み込まれた日本語文と英文とを表示するセットの番号を示すものである。Form1_Load のサブプログラムの終了直前に i = 0 として初期化されているので，Button3 がクリックされた場合，最初の問題が表示されることになる。

次の If 文は End If までの 3 行の実行文を含むものである。ここでの If 文の構文は，

```
If 条件式 Then
   実行文
End If
```

である。条件式の中の Len 関数は（）内の変数の長さを返すものである。Len(SrchStr(i,2) は英文の配列 SrchStr(i,2) に含まれている文字数を返すものである。この（）内の i は i 番目の日本語文と英文のセットの順番を示し，２はそのうちの英文を意味している。Len(SrchStr(i,2)=0 は i 番目の長さが 0 の場合，Then 以下の実行文が起動する。「MessageBox.Show」は，図 4 のようなメッセージを示す。

これによって，データ配列の終了が知らされる。「i = i - 1」は i の値を１つ減少させ，データ配列の終了の前に戻させる。「Exit Sub」は，このサブプログラムからの離脱を意味する。したがって，「End If」より下のプログラムは，データ配列が終了していない場合にのみ実行されることになる。図 4 に課題が最後まで進んだ時の実行画面を示す。

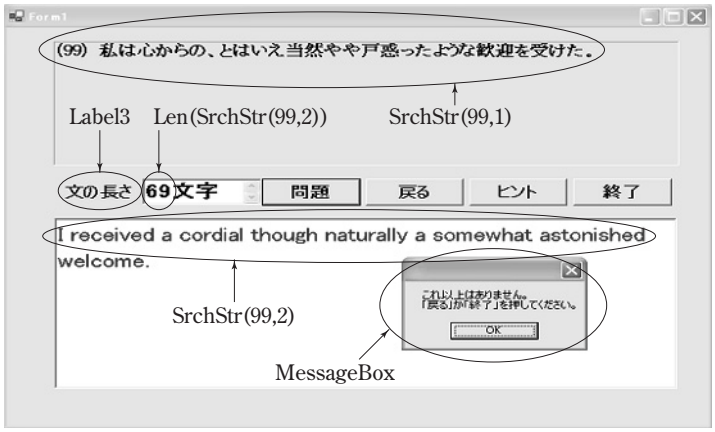

図 4　課題完了時の実行画面

「Label1.Text = "(" & i & ")" & SrchStr(i, 1)」は Label1 に問題となる日本語文を表示するためのものである。問題番号 i に括弧をつけて，その後に i 番目のデータ配列の日本語文を表示する。& マークはそれらを連続表示することを意味している。「RichTextBox1.Text = ""」は答案を入れる RichTextBox1 を初期化する。「TextBox2.Text = Len(SrchStr(i, 2) & " 文字 "」は i 番目の英文の長さを文字数で示し，その後に「文字」を付け加える。「RichTextBox1.Focus()」は自動的に RichTextBox1 を入力待ちの状態にするメソッドである。これによって，カーソルを移動させてクリックするというイベントがなくても，同等の働きがあることになる。図 5 に，課題開始時の実行画面を示す。

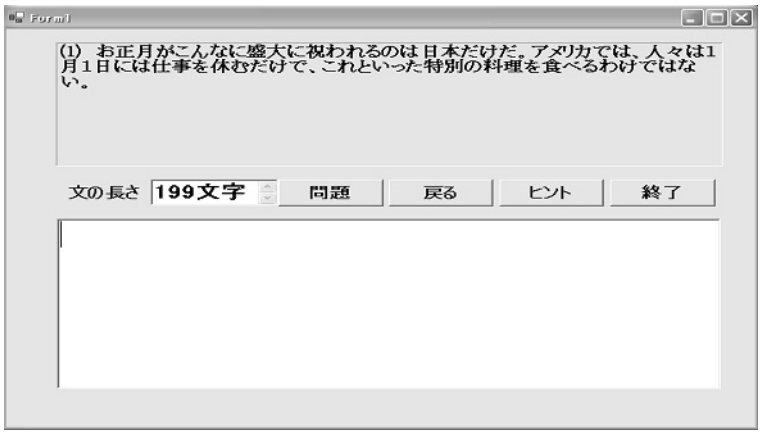

図 5　課題開始時の実行画面

2－4　正誤判定のためのプロシージャ

```
Private Sub RichTextBox1_TextChanged(ByVal sender As Object,ByVal e As System.
EventArgs)Handles RichTextBox1.
  TextChanged
  ’正誤判定
  PictureBox1.Visible = False
  If RichTextBox1.Text = "" Then Exit Sub ’新しい問題になったときに必要
  ’Mid(Str,st,len)= 文字列 Str の開始位置 st から文字数 len を取り出す。
  If RichTextBox1.Text <> Microsoft.VisualBasic.Mid(SrchStr(i, 2), 1, ltr) ① Then
    If hintbtn >= 1 ② Then Exit Sub ③
    MsgBox(Microsoft.VisualBasic.Rig(RichTextBox1.Text, 1) & " は間違っています。", _
    MsgBoxStyle.OKOnly + MsgBoxStyle.Exclamation, " 気をつけて !")
    RichTextBox1.Select(ltr - 1, 1)’カラー表示すべき対象
    ErrorCase = 1
  Else
    RichTextBox1.Select(ltr - 1, 1) ⑤
    If ErrorCase = 0 ⑥ Then
      If hintbtn = 0 ⑦ Then
        RichTextBox1.SelectionColor = Color.Black ⑧
      Else
        RichTextBox1.SelectionColor = Color.Blue ⑨
      End If
    Else
```

④

```
        If hintbtn = 0  ⑩ Then
          RichTextBox1.SelectionColor = Color.Green ⑪
        Else
          RichTextBox1.SelectionColor = Color.Red ⑫
        End If
      End If
      RichTextBox1.SelectedText = Microsoft.VisualBasic.Mid(SrchStr(i, 2), ltr, 1)
      ErrorCase = 0                                                              } ⑬
      ltr = ltr + 1
    End If
    RichTextBox1.Focus()
    If ltr >= Len(SrchStr(i, 2)Then
      Timer1.Enabled = True
      PictureBox1.Visible = True
      PictureBox1.Image = System.Drawing.Image.FromFile("GoodD.BMP")
      MsgBox("「問題」を押してください。", MsgBoxStyle.OKOnly + MsgBoxStyle.Information,
      " やりました !")
      Button3.Focus()
      ltr = 1
    End If
    hintbtn = 0
End Sub
```

「Private Sub RichTextBox1_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs)Handles RichTextBox1.TextChanged」は，答案の正誤を判定するためのサブプログラムである。これは，RichTextBox1 のテキストが変わるたびに起動するものである。ここでは，答案として RichTextBox1 に入力された文字が正しいか誤っているかを，１文字ごとに判定するために用いられる。このサブプログラムのフローチャートを図６に示す。

一般に，テキスト表示には，テキスト・ボックスが用いられる。テキスト・ボックスでも，ForeColor プロパティを変更することにより表示文字全体の色を変えることはできる。しかし，部分的に表示文字の色を変えようという場合はリッチテキスト・ボックスを利用しなければならない。図１，図６，図９のように，表示する文字の色を場合によって使い分けることが必要となる。そのために，どのようにプログラミングするかというのが，この課題での主要な問題となる。

「PictureBox1.Visible = False」は前述の Button3_Click のサブプログラムと同様，PictureBox1 の表示を一時停止するためのものである。「If RichTextBox1.Text = "" Then Exit Sub」は，

PictureBox1 がクリアされた時には正誤の判定の必要がないので，ただちに Exit Sub を実行することによって，このサブプログラムから抜け出るようにする。次の If 文はこのサブプログラムの中心をなすものである。その構文は

```
If 条件式 Then
  実行文
Else
  実行文
End If
```

となっている。それぞれの実行文はさらに If 文を含む入れ子構造をしている。Then 以下には 1 つの If 文が，Else 以下には 3 つの入れ子構造の If 文が含まれている。その構文は図 7 のようになっている。**条件式①**「RichTextBox1.Text <> Microsoft.VisualBasic.Mid(SrchStr(i, 2), 1, ltr)」の <> は左辺と右辺が等しくないときに真となる論理式である。右辺は i 番目の問題に対する正解となる英文データが入っている配列 SrchStr(i, 2) の開始位置 1 番目から文字数 ltr を取り出すことを意味している。RichTextBox1 のテキストの内容が正解の SrchStr(i, 2) の初めから ltr 文字分と等しくない場合，**If 条件式② Then 実行文③**から**実行文④**までを実行し，等しい場合は Else 以下の**実行文⑤**から，**実行文⑬**の後の End If までを実行する。

条件式②「hintbtn >= 1」の hintbtn はヒントボタンが何回クリックされたかをカウントするものである。これが 1 以上ということは，すでにヒントボタンが 1 回以上クリックされていることを意味する。後述のように，ヒントボタンは，それをクリックすることによって，RichTextBox1 に次の 1 文字を表示することになる。この場合，当然正解が付け加えられることになるので，正誤を判定する必要はない。しかし，Private Sub RichTextBox1_TextChanged は，RichTextBox1 のテキストが変化すれば必ず起動するので，不必要な判定を避け，**実行文③**「Exit Sub」を実行して，このサブプログラムから抜け出ることにする。

実行文④は，hintbtn >= 1 でない場合，すなわち hintbtn が初期値 0 のままの場合に実行される。「MsgBox(Microsoft.VisualBasic.Right(RichTextBox1.Text, 1) & "は間違っています。",」と「MsgBoxStyle.OKOnly + MsgBoxStyle.Exclamation, "気をつけて！")」は誤反応であることをメッセージ・ボックスで示す。「 _」（スペースとアンダーバー）は次の行に続くことを意味する。Microsoft.VisualBasic. Right（RichTextBox1. Text, 1) は RichTextBox1 のテキストの右端の 1 文字を返す関数である。これは，直近に入力された文字を指す。これが間違っているというメッセージを OK ボタンとともに示す。同時に，エクスクラメーションマークとタイトル・バーに「気をつけて！」という表示をする。図 8 参照。「RichTextBox1.Select(ltr - 1, 1)」は，RichTextBox1 のテキストの (ltr - 1) 文字目から 1 文字，つまり，右端の 1 文字を選び，それをその後，カラー表示するための対象とする。「ErrorCase = 1」は誤反応があったことを示すものである。

Else 以下は，**条件式①**が成り立たない場合，つまり，RichTextBox1 に正しい文字が入力された場合に実行される。**実行文⑤**「RichTextBox1.Select(ltr - 1, 1)」は前述同様，PictureBox1 のテキス

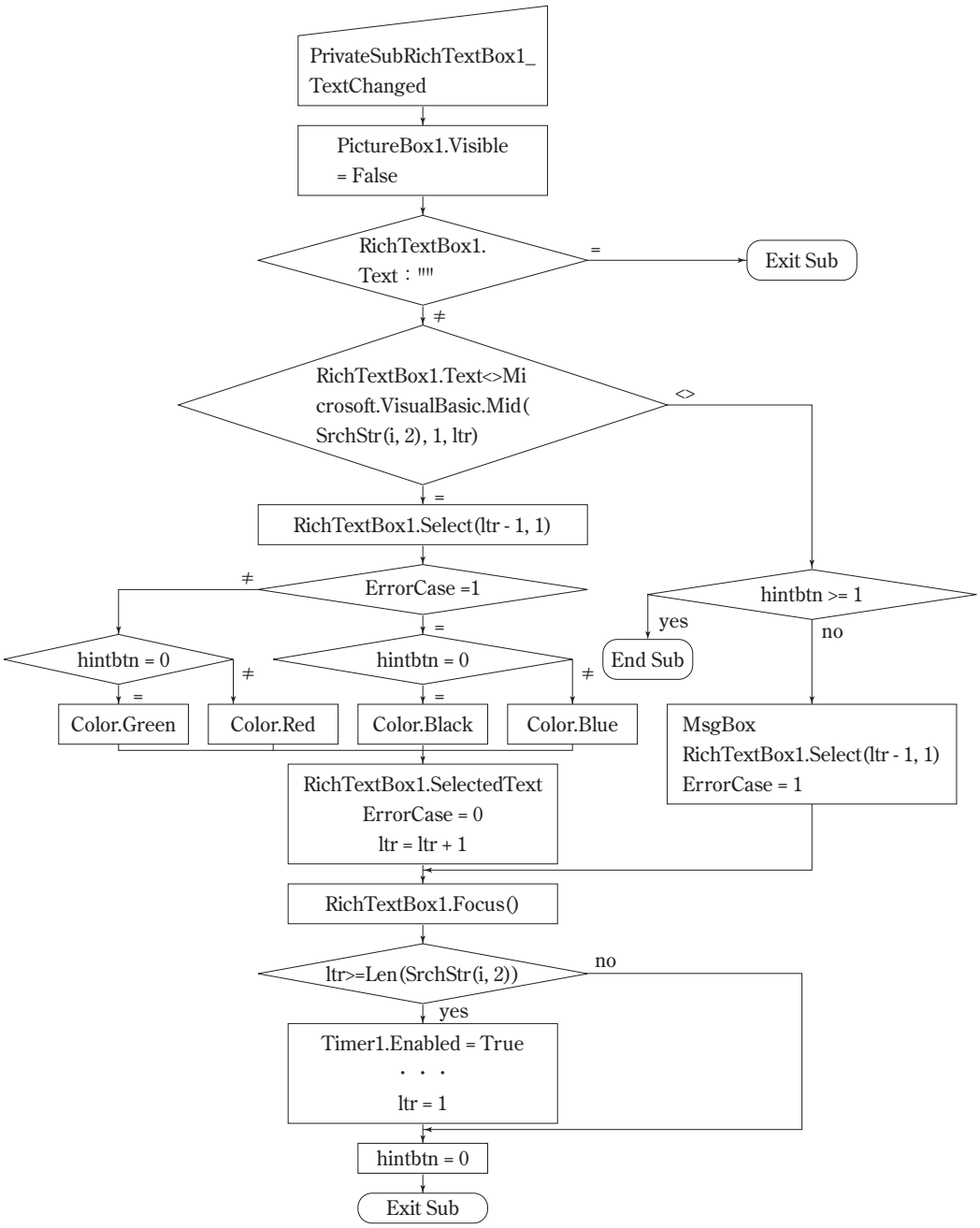

図６　PrivateSubRichTextBox1_TextChanged のフローチャート

トの右端の１文字を選ぶ。

次の**条件式⑥**を含む **If** 文から**実行文⑬**の後の **End If** までは，先に **RichTextBox1.Select** で選ばれた文字を，図１，図６で示したように，入力された状況に応じてカラー表示させるためのものである。ここで，表示文字の色を決定するのに重要な役割を果たしているのが，変数 **ErrorCase** と

```
If条件式①Then
  If条件式②Then実行文③
  実行文④
Else
  実行文⑤
  If条件式⑥Then
    If条件式⑦Then
      実行文⑧
    Else
      実行文⑨
    End If
  Else
    If条件式⑩Then
      実行文⑪
    Else
      実行文⑫
    End If
  End If
  実行文⑬
End If
```

図 7　Private Sub RichTextBox1_TextChanged の If 文の構文

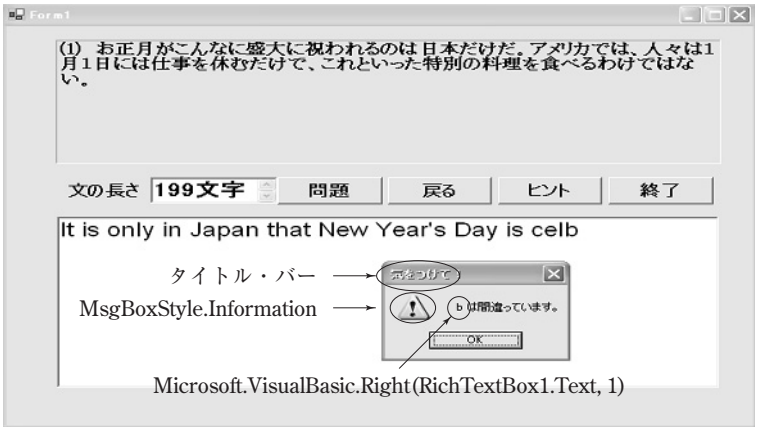

図 8　誤答に対する実行画面の例

hintbtn の値の組合せである。

まず，If 文で，**条件式⑥**「ErrorCase = 0」が成り立つ場合と成り立たない場合に分ける。これは，誤答を入力していない場合と，誤答を入力している場合に分けることを意味している。次の If 文で，**条件式⑦**「hintbtn = 0」が成り立つ場合と成り立たない場合に分ける。hintbtn = 0 はヒントボタンをクリックしていないことを意味する。これは ErrorCase = 0 と hintbtn = 0 との論理積を求めていることになる。つまり，誤答をしていなくて，かつヒントボタンをクリックしていないことを条件にしている。この時，**実行文⑧**「RichTextBox1. SelectionColor = Color.Black」で，表

示される文字を黒色とする。次の**実行文⑨**「RichTextBox1.SelectionColor = Color.Blue」は，**条件式⑦**が成り立たない場合，すなわち hintbtn ≠ 0 で，すでにヒントボタンがクリックされている場合は表示される文字を青色とすることを指示している。これは，ErrorCase = 0 と hintbtn ≠ 0 の両方が同時に成り立つことを条件とすることを意味する。**条件式⑩**「hintbtn = 0」を含む If 文から**実行文⑫**の後の End If までは，**条件式⑥**「ErrorCase = 0」が成り立たない場合，つまりすでに誤答をしている場合を扱うことになる。

条件式⑩「hintbtn = 0」，すなわちヒントボタンをクリックしていない場合は，**実行文⑪**「RichTextBox1.SelectionColor = Color.Green」で先に**実行文⑤**で選ばれた領域の色を指定して，緑字で表示する働きをする。次の**実行文⑫**「RichTextBox1．SelectionColor = Color.Red」は**条件式⑩**「hintbtn = 0」が成り立たない場合，すなわち hintbtn ≠ 0 で，すでにヒントボタンがクリックされている場合は表示される文字を赤色とする。

実行文⑬は３行の実行文からなる。「RichTextBox1．SelectedText = Microsoft.VisualBasic.Mid(SrchStr(i, 2), ltr, 1)」は正解となる英文データ SrchStr(i, 2) の ltr 文字目から１文字を返す Mid 関数を利用して，それを RichTextBox1 に表示することになる。「ErrorCase = 0」は，たとえ直近の試行において，誤反応があったとしても，正解が示されたこの時点で，ErrorCase の値を 0 に設定し直しておくことを意味する。「ltr = ltr + 1」は，次の文字に進むために，文字数のカウンターである ltr の値を１だけ増やしておくことを意味する。以上で，**条件式①**を含む If 文が「End If」文で終了することになる。

次の「RichTextBox1.Focus()」で，RichTextBox1 の直近のカーソルの位置で入力待ちの状態にすることができる。その次の If 文は１つの問題が終了したかどうかを判定するためのものである。「If ltr >= Len(Srch Str(i, 2)Then」は６行分の実行文を含むものである。条件式の右辺の Len 関数は（）内の変数の長さを文字数として返すものである。つまり，１文字ずつアルファベットが入力されて，その文字数の正解の長さまで進んだなら，その問題を終了する。そのために，「Timer1.Enabled = True」で，Timer1 を起動させ，RichTextBox1 の中に隠されていた PictureBox1 を「PictureBox1.Visible = True」で，可視状態，すなわち見えるようにする。

「PictureBox1.Image = System.Drawing.Image.FromFile("GoodD.BMP")」は PictureBox1 に表示する画像ファイルを指定する書式である。ここでは，GoodD.BMP というファイル名の画像ファイルを指定している。「MsgBox("**「問題」をクリックしてください。**", MsgBoxStyle.OKOnly + MsgBoxStyle.Information, "**やりました!**")」は「「問題」をクリックしてください。」というテキストと OK というキャプションの付いたタイトル・バーについてメッセージ・ボックスを表示する。図９に問題終了時の実行画面の例とメッセージ・ボックスを示す。

その後，「Button3.Focus()」で「**問題**」というキャプションが付いた Button3 がすぐ起動できる状態にしておく。さらに，「ltr = 1」で，正解の文字カウンター ltr を１に初期化する。

最後に，「hintbtn = 0」として，ヒントボタンがクリックされていない状態に戻しておく。

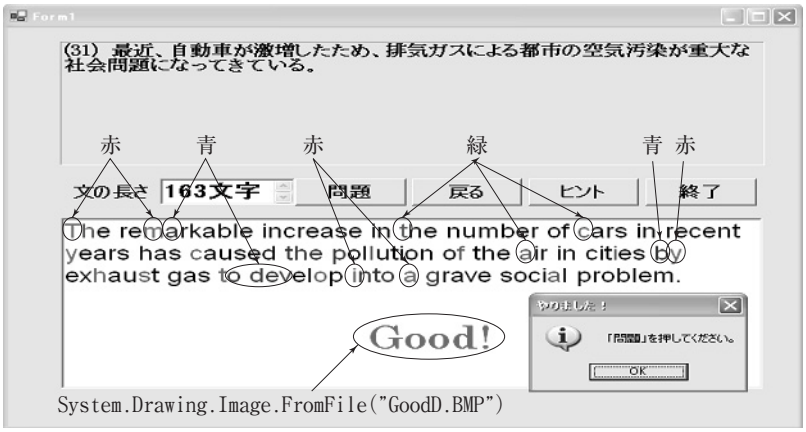

図 9　問題終了時の実行画面の例とメッセージ・ボックス

2－5　ヒント表示のためのプロシージャ

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.Event
Args)Handles Button1. Click
  ’ヒント表示
  hintbtn = hintbtn + 1
  RichTextBox1.SelectedText = Microsoft.VisualBasic.Mid(SrchStr(i, 2), ltr, 1)
End Sub
```

「Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.Event Args)Handles Button1.Click」はヒントを表示させるためのサブプログラムである。これは，「ヒント」というキャプションがついたButton1をクリックすることによって，RichTextBox1に正解となる次の1文字を表示させる。「hintbtn = hintbtn + 1」は，ヒントボタンがクリックされると変数hintbtnの値を1ずつ増加させることになる。「RichTextBox1.SelectedText = Microsoft. VisualBasic.Mid(SrchStr(i, 2), ltr, 1)」)」は先述のPrivate Sub RichTextBox1_TextChangedの実行文⑭に含まれているものと同じで，正解データの英文のltr文字目の1文字をRichTextBox1に表示する。これによって，当然，RichTextBox1のテキストが変化するので，Private Sub RichText Box1_TextChangedが起動することになる。

2－6 終了のためのプロシージャ

```
Private Sub Button 2 _Click（ByVal sender As System.Object, ByVal e As System.EventArgs）
Handles Button2. Click
  ’終了
  End
```

```
End Sub
```

「Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.Event Args)Handles Button2.Click」は，ジョブを終了させるためのサブプログラムである。これは，「終了」というキャプションが付いた Button2 をクリックすることによって，「End」という命令が実行され，課題を強制的に終了させることができる。

２－７　前の問題に戻るためのプロシージャ

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.Event Args)Handles Button4.Click
  ' 前の問題に戻る
  Timer1.Enabled = False
  PictureBox1.Visible = False
  ErrorCase = 0
  ltr = 1
  hintbtn = 0
  i = i - 1
  RichTextBox1.Text = ""
  TextBox2.Text = Len(SrchStr(i, 2)& " 文字 "
  If i <= 0 Then
    i = i + 1
    MsgBox(" データの最初まで戻りました。" & vbCrLf & "「問題」をクリックしてください。", _
    MsgBoxStyle.OKOnly + MsgBoxStyle.Exclamation, " 注意 ")
    Button3.Focus()
    Exit Sub
  End If
  Label1.Text = "(" & i & ")" & SrchStr(i, 1)
  RichTextBox1.Focus()
End Sub
```

「Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.Event Args)Handles Button4.Click」は，前の問題に戻るためのサブプログラムである。「戻る」というキャプションが付いた Button4 をクリックすることにより，問題番号の小さい問題を表示させることができる。いわば，「問題」のキャプションが付いた Button3 が番号順に問題を表示するのに対して，「戻る」の Button4 は逆順に表示することになる。次の５行分は，前述同様，「Timer1.Enabled = False」でタイマーを停止し，「PictureBox1.Visible = False」で PictureBox1 を見えなくし，さ

らに，「ErrorCase = 0」と「hintbtn = 0」で ErrorCase と hintbtn とを 0 に初期化し，「Itr = 1」で Itr を 0 に初期化する。「i = i - 1」は右辺の i の値から 1 を引いた結果を左辺の i に入れる。これによって，i の値を 1 減らすことになる。問題を戻したとき，Label1 に 1 つ前の問題となる日本語文が表示される。この時，RichTextBox1 に残っていた直近の問題に対する答の文字を消去するために，「RichTextBox1.Text = ""」が用いられる。「TextBox2.Text = Len(SrchStr(i, 2) & " 文字 "」は，TextBox2 に正解の文字数を表示する。

次の If 文は i が減少していって，0 以下になった場合の対処方法を設定するものである。「i = i + 1」 で i の値が 0 になった場合，右辺が 1 になる。この値を左辺の i に代入して値を 1 とする。「MsgBox(" データの最初まで戻りました。" & vbCrLf & "「問題」をクリックしてください。",MsgBoxStyle.OKOnly + MsgBoxStyle.Exclamation, " 注意 ")」では，メッセージ・ボックスにテキストとして「データの最初まで戻りました。」と表示し，改行して「「問題」をクリックしてください。」と表示する。この時，エクスクラメーション・マークのアイコンとタイトル・バーに「注意」と表示する。「Button3.Focus()」で，問題を表示するための Button3 を入力待ちの状態にし，最後に Exit Sub で，サブプログラムを抜け出て，「End If」で If 文を終える。

i が 1 以上である場合は，「Label1.Text = "(" & i & ")" & SrchStr(i, 1)」で i が 1 つ減った問題の日本語文を Label1 の領域に表示する。「RichTextBox1.Focus()」で，RichTextBox1 を入力待ちの状態にする。

2－8　画像を点滅させるためのプロシージャ

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)Handles Timer1.Tick
  ' 画像点滅
  If PictureBox1.Visible = True Then
    PictureBox1.Visible = False
  Else
    PictureBox1.Visible = True
  End If
  Timer1.Interval = 300
End Sub
```

「Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)Handles Timer1.Tick」は Timer1 のための関数サブプログラムである。これは，VisualBasic.NET のデザイン画面には現れるが，実行画面には現れない。このプログラムは，各問題が終了するごとに，PictureBox1 に「Good!」という文字を画像として点滅させる役割を担っている。ここでの If 文は，「PictureBox1.Visible = True 」が条件式となっており，PictureBox1 が可視状態ならば，「PictureBox1.Visible = False」で不可視状態に，すなわち見えないようにし，不可視

状態ならば，「PictureBox1.Visible = True」で可視状態，すなわち見えるようにする。この見える状態と見えない状態とを Timer1 が稼動するたびに交替させることによって文字が点滅するようにさせる。「Timer1.Interval = 300 」は Timer1 の稼動の時間間隔を決めるもので，個々では，300msec. すなわち0.3秒間隔としている。これによって，Timer1 が実行状態のときには0.3秒ごとに，このサブプログラムが働くことになる。図９参照。

本稿では，学習結果を視覚的にフィードバックするプログラムの作成とその解説を行なった。しかし，本プログラムでは，新しい問題に替わるたびに学習の履歴情報が消えてしまう。今度の課題としては，間違った部分を履歴情報として蓄積していく機能や間違った部分のみを再入力させる機能を開発することにより，学習者の到達レベルにあった，より応答性の高いプログラムを作成することが求められる。

注

１） ソフトは Microsoft Visual Basic.NET Standard Version 2003 を用いた。

２） Microsoft Visual Basic. NET は Microsoft Visual Basic 6.0 を改定したものである。が，本稿で用いたコマンドの多くは，Microsoft Visual Basic 6.0 と共通している。Microsoft Visual Basic 6.0 については門田（1998, 1999, 1999b, 1999c, 2002）を参照。

参考文献

IT フロンティア 2003「Visual Basic.NET 逆引き大全500の極意」秀和システム

Microsoft Corporation 2002 マイクロソフト（株）訳「Microsoft.Basic.NET」日経 BP ソフトプレス

江川泰一郎　1994「英文法解説」 金子書房

河西朝雄　2003「VB.NET 基礎学習 Bible」技術評論社

金城俊哉　2005「Visual Basic パーフェクトマスター」秀和システム

中尾清秋　1977 「基礎と演習 英作文」 数研出版

波多野誼余夫・稲垣佳世子　1973「知的好奇心」 中央公論社

波多野誼余夫・稲垣佳世子　1977「知力の発達」 岩波書店

波多野誼余夫・稲垣佳世子　1981「無気力の心理学」 中央公論社

門田幸太郎　1998「VISUAL BASIC のプログラミング法１――その特徴とプログラミングの基礎――」立命館大学産業社会論集 第34巻 第３号 pp. 119-136

門田幸太郎　1999「VISUAL BASIC のプログラミング法２――配列データの操作――」立命館大学産業社会論集 第34巻 第４号 pp. 167-187

門田幸太郎　1999b「VISUAL BASIC のプログラミング法３――ファイルの操作；読み込みと表示の基礎――」立命館大学産業社会論集第35巻 第１号 pp. 1-13

門田幸太郎　1999c「VISUAL BASIC のプログラミング法４――ファイルの操作；読み込みと表示の応用――」立命館大学産業社会論集第35巻 第２号 pp125-141

門田幸太郎　2002「VISUAL BASIC による応答的プログラム――ユーザの成績に応じた問題提示法――」立命館大学産業社会論集 第38巻 第２号 pp. 1-17

山本昌弘・重定恕彦　2004「例題でわかる Visual Basic. NET」東京電機大学出版局

若山芳三郎　2004「学生のために Visual Basic. NET」東京電機大学出版局

Responsive Programming by Visual Basic. NET: For the learning of English syntax

MONDEN Kotaro *

Abstract: This study examined the possibility of applying a responsive program to the learning of English syntax. Each time an English letter is entered for the Japanese sentence, it is judged right or wrong. When a correct answer is entered, it is displayed by a black letter. When reentering a correct answer after a mistake, it is displayed in green. When clicking the hint button after a mistake, it is displayed in red.

When clicking the hint button without first attempting a trial, it is displayed by a blue letter. With this program, users can immediately get feedback regarding of the correctness of their choice of letters as well as a record of previous trials. By seeing the whole result displayed, the progress of learning becomes known, too.

Keywords: responsive programming, Visual Basic. NET, feedback, learning of English syntax

＊Professor, Faculty of Social Sciences, Ritsumeikan University