フローチャートからVisual Basic プログラムコードを作成するプログラムの作成

塩澤哲時　　岡部建次
駿河台大学　文化情報学部

簡単に利用できるフローチャートからプログラムコードを作成するプログラムを作成した。プログラム自身を小さくしてあるので作成コスト、変更・メインテナンス作業は小さくて済む。画面にフローチャートを記述すると Visual Basic のプログラムの配列上に記号化されたフローチャート部品が記述され、これをプログラムで解読し Visual Basic プログラムコードに変換する。プログラム作成の自動化・省力化を意図している。更に実用的なプログラミング機能を実装し、実用的なものにする研究をしている。

Development of Visual Basic program code preparation program from the flow chart

Tetutoki Shiozawa　　Kenji　Okabe
Faculty of Cultural Information Resources　Surugadai University

We had developed the program which is able to convert flow chart to Visual Basic codes. The program is a small one so that the maintenance and improvement are easily done. The diagram in the flow chart is described as a symbolic code into the matrix within Visual Basic program. After the completion of the flow chart drawing then the chart is converted to Visual Basic program codes automatically. We intend to grade-up the program for practical use and further automation.

１　はじめに

本研究の目的は Visual Basic(以下 VB)によるプログラム開発を相当部分プログラムにより自動化することにより、複雑なプログラムが容易に作成できる方法の確立をめざす。研究の最初の段階としてフローチャートを自動的に VB のコードに変換するプログラムを作成した。

今日あらゆる分野において情報技術がますます活用され、情報技術を支えるソフトウエアに対するニーズは大きい。CASE 技術で代表されるようにプログラム作成工程の支援は多くの研究がなされている。しかし安価で、簡単に使えるものが VB のプログラム作成に活用されていない。本研究は安価で簡単に使える小さな VB プログラム作成システムを目指している。システムを小さく作ることは変更・メインテナンスを容易にし、寄り効果的なシステムを作る可能性を持つ。

２．プログラムの使用例

基本制御構造の順次、選択、繰返しを用いたフローチャートを作成し、それをもとに Visual1Basic のコードを自動生成する。生成されたコードはクリップボードへ複写されるので、Visual1Basic ソースの目的の場所に貼付して使用する。プログラムを起動し、**図１**に示す最小値を求めるフローチャートを作成する。上段の「ファイル(F)」メ

ニューの“コード化"をクリックすると、フローチャートを解釈して VisualBasic のコードになる。
図２に作成されたコードを示します。
　このコードをフォーム上の最小値を求めるコマンドボタンのコードとして貼り付ける。

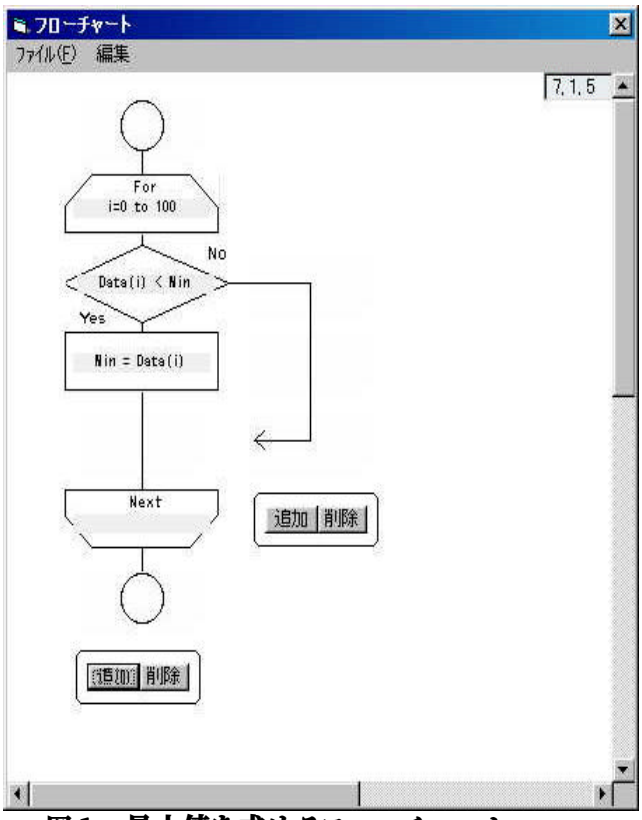

図１　最小値を求めるフローチャート

```
For i=0 to 100
        If Data(i) < Min Then
                Min = Data(i)
        else
        end if
Next
```

図２　作成されたコード

３　プログラムの使用法説明

3.1　起動

　プログラムを起動すると、**図３**のように左上に“ファイル",“編集"メニューその下にフローチャートの端子と“追加"“削除"のポタンが表示される。この下にチャートの部品を追加していくことでフローチャートを作成する。

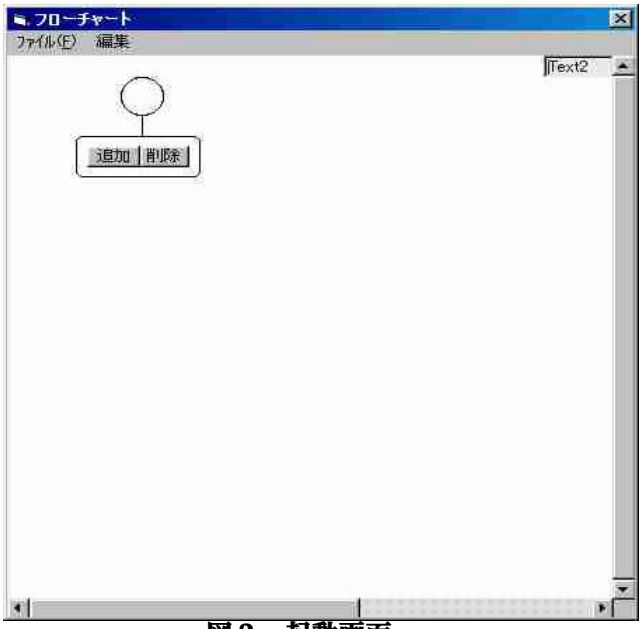

図３　起動画面

3.2　部品の追加

　追加ポタンをクリックすると、部品選択のウィンドウが開く(**図４**)。部品は 7 種類ある。**表１**

部品名	説明
線	上下を結ぶ線
順次	シーケンス処理
選択	条件選択　(If 文)
復帰	条件選択の終わり
繰返　始	ループ構造の始め
繰返　終	ループ構造の終わり
終端	チャートの終わり

表１　フローチャートの部品

順次、選択、繰返しでは条件または処理と、

その説明を記述できる。記述した内容は作成されるチャートの条件または処理に反映される(**図5**)。

3.3　コードへの変換

選択は If 文、繰返しは Do～Loop または For～Next として解釈される。また、選択や繰返しの中は構造がわかりやすいようにインデントされる。生成されたコードはクリップポードへ複写されるので、VisualBasic ソースの目的の場所に貼付して使用する。

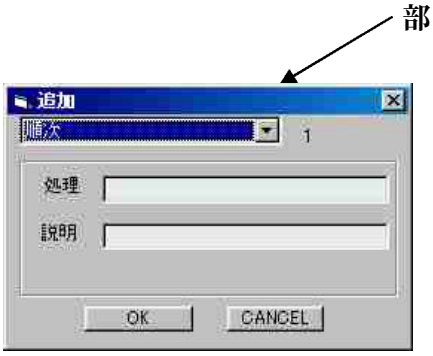

図4　部品選択

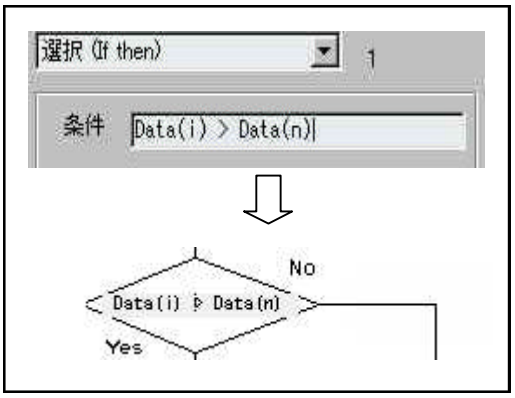

図5　部品追加の例

コード化においてフローチャートが正しく記述されていない場合、例えば選択に対応する復帰がない場合などは正しくコード化されない。このような場合も特にエラーメッセージなどは表示されない。生成されたコードに不具合のある場合はフローチャートを見直します。

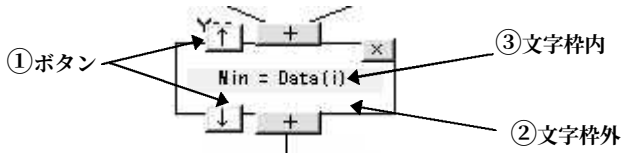

図6　追加，削除，編集

3.4　その他の編集機能

追加された部品の上にマウスカーソルをあわせるといくつかのポタンが表示される。また、文字枠外をクリックするとメニュー、枠内をクリックすると部品の編集ウィンドウが表示される。これらを使って部品の追加、削除、編集ができる。(**図6**)

①ポタン

矢印(↑↓)：部品を上下に移動します下に移動すると“線"が挿入される。

プラス(十)：クリックしたポタン場所への部品を追加挿入する。

パツ(×)：部品を削除する。

②メニュー(文字枠外をクリヅク)

上下への部品の追加と、部品の削除ができる。

③編集ウィンドウ(文字枠内をクリック)

処理、条件と説明の変更ができます。ただし部品の種類は変更できない。

4　コードへの変換の仕組み

フローチャートを VB プログラムコードに変換するプログラムについて報告する。

4.1　全体の流れ

プログラム全体の流れを次頁**図 7** に示す。

4.2　チャート作成

チャート作成面ではチャート上の部品を配列で管理している。**図 8** で示すように上左端から（1，0）で始まり、以下（列、行）であらわされている。配列は位置座標の役割を果たしている。配列の中身は部品の種類である。　追加ボタン

または“＋”をクリックすると部品追加のための部品選択ダイアログボックスが表示される（図８）。

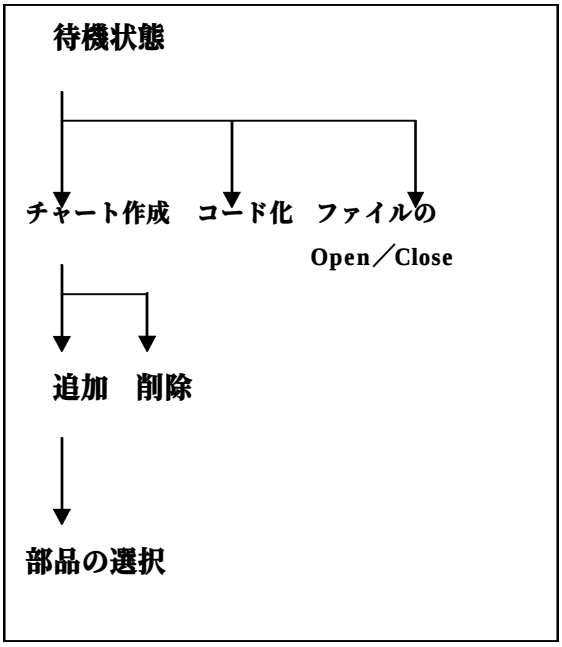

図７　処理全体の流れ

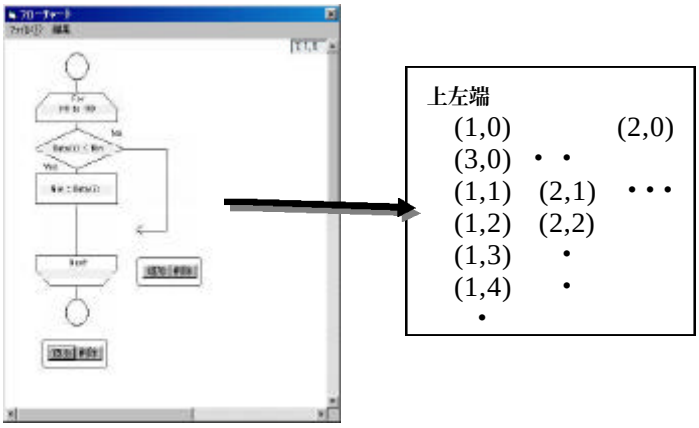

図８　部品配列

部品は“線”、“順次”、“選択”、“復帰”、“繰返_始”、“繰返_終”、“終端”の７種類あり、順番に１から７の番号がふられている。部品が追加されると、チャート上の部品を示す配列にその番号が代入される。ただし、部品“選択”を追加した場合に右隣へ追加される左から下への折れ線 ┐ (図 10)が４番のパーツとなるので以下の番号がずれて、配列中の番号と部品の関係は表２のようになる。

番号	部品	
1	線	
2	順次	
3	選択	
4	折れ線	
5	復帰	
6	繰返　始	
7	繰返　終	
8	終端	

表２　配列中の番号と部品の関係

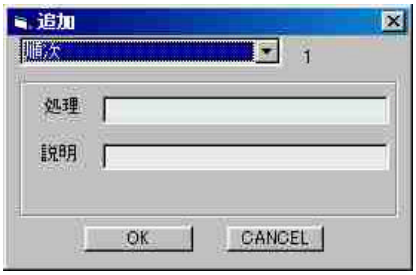

図９　部品選択

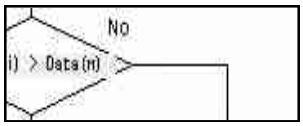

図 10“選択”の折れ線

4.3　コード化のアルゴリズム

上部タスクバーのファイル(F)をクリック(図 1 参照)表示されるメニューの“コード化”をクリックすると、作成されたチャートを Visual Basic のコードに変換する。コード化は基本的に部品を頭から命令に直しているだけである。“順次”は記述された命令がそのままコードになり、

“選択”では If 文、“繰返し”では Do～Loop または For～Next になる。**図 11** にフローチャートを VB プログラムコードに変換するプログラムのソースコード（ファイルメニュー“コード化”のクリックイベント）を示す。

“Select Case Node(j, i)” 以下（②）で部品の種類ごとの処理を行っている。配列 Node(j, i)がチャート上の部品を管理する配列であり、Select Case によって部品番号ごとに処理をしている。

基本的にチャートの部品配列の１列目を若い順にチェックしていく。そして、“選択”によって分岐するところでは分岐した右隣の列を“復帰”まで調べてスタックし、“else”に続く文として If Then の後ろに加えている（①）。

作成されたコードはクリップボードに落ちる。これを作成しているアプリケーションのフォームのプログラムコードを記述すべきオブジェクトに貼り付ける。

```
Private Sub Coding_Click()

    Dim K(5) As Integer
    For N = 1 To 5: K(N) = -1: Next
    Dim L(5) As Integer
    Dim Tem(5) As String
    temp = ""
    i = 0: j = 1

    Do Until Node(1, i) = 0 Or Node(1, i) = 8
        If K(j + 1) <> -1 And i - 1 = L(j + 1) Then
            temp = Left(temp, Len(temp) - 1)
            Tem(j + 1) = Left(Tem(j + 1),
Len(Tem(j + 1)) - 1)
            temp = temp & "else" & Tem(j + 1) &
"end if" & Chr(13) & Chr(10)
            K(j + 1) = -1
            tabLN = tabLN - 1
            t = 0
            Do Until tabLN <= t
                temp = temp & Chr(9)
                t = t + 1
            Loop
        End If
```

①

```
        Select Case Node(j, i)
        Case 2                                  ‘順次
            temp = temp & Text1(j * 100 + i).Text
        Case 3                                  ‘選択
            tabLN = tabLN + 1
            temp = temp & "If " & Text1(j * 100
+ i).Text & " Then"
            Tem(j) = temp
            temp = ""
            j = j + 1
            K(j) = i
            L(j) = 0
        Case 5                                  ‘復帰
            Tem(j) = temp
            temp = Tem(j - 1)
            L(j) = i - 1
            i = K(j)
            j = j - 1
        Case 6                                  ‘繰返し始め
            tabLN = tabLN + 1
            Select Case NodeTag(j, i)
            Case 0
               temp = temp & "For " & Text1(j *
100 + i).Text
            Case 1
                temp = temp & "Do "
            Case 2
                temp = temp & "Do While " &
Text1(j * 100 + i).Text
            Case 3
                temp = temp & "Do Until " &
Text1(j * 100 + i).Text
            End Select

        Case 7                                  ‘繰返し終わり
            tabLN = tabLN - 1
            temp = Left(temp, Len(temp) - 1)
            Select Case NodeTag(j, i)
            Case 0
                temp = temp & "Next "
            Case 1
                temp = temp & "Loop "
            Case 2
                temp = temp & "Loop While " &
Text1(j * 100 + i).Text
            Case 3
                temp = temp & "Loop Until " &
Text1(j * 100 + i).Text
            End Select
        End Select

        If Node(j, i) <> 1 Then
            temp = temp & Chr(13) & Chr(10)
            t = 0
            Do Until tabLN <= t
                temp = temp & Chr(9)
```

②

```
                t = t + 1
            Loop
        End If

        i = i + 1
    Loop

    CodeText = temp
    frmCode.Show

End Sub
```

図11　フローチャートをプログラムに変換するプログラムソースリスト

5　利用例

本プログラムを利用して実際に作成したプログラム例を示す。

5.1 フィボナッチ数列を求めるプログラムの作成

次頁**図 12** に示すフローチャートを作成する。フローチャートをプログラムコードに自動変換すると**図 13** に示すコードとなる。コードはクリップボードにコピーされているので、これを**図14**のコマンドボタンに張り付けてプログラムを実行すると**図14**のテキストボックスにフィボナッチ数列が表示される。

```
Max = Text1.Text
a = 1
b = 0
c = 0
For n = 1 To Max
    t = t & Str(a) & ","
    c = a
    a = a + b
    b = c
Next
Text2.Text = t
```

図13　自動コード化により作成したフィボナッチ数列を求めるプログラム

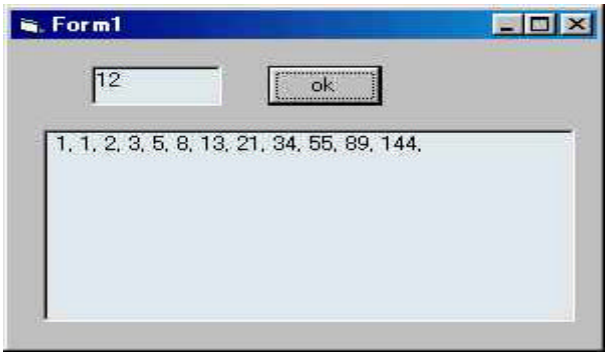

図14　フィボナッチ数列表示

入力された数字の長さのフィボナッチ数列を計算する。

5.2　二分探索のプログラムを作成

図 15 に示すフローチャートを作成する。フローチャートをプログラムコードに自動変換すると**図16**に示すコードとなる。コードはクリップボードにコピーされているので、これを**図17**のコマンドボタンに張り付けてプログラムを実行すると**図 17** のメッセージ・ボックスに探索結果が表示される。

```
Target = Text1.Text
n = 0
l = 0
h = max
f = False
Do
    If data(n) = Target Then
        MsgBox "find"
        f = True
    Else
        If data(n) < Target Then
            l = n
            n = Int((h - n) / 2) + n
        Else
            h = n
            n = Int((n - l) / 2) + l
        End If
        If n = l Then
            MsgBox "nothing"
            f = True
        Else
        End If
    End If
Loop Until f = True
```

図16　二分探索プログラム

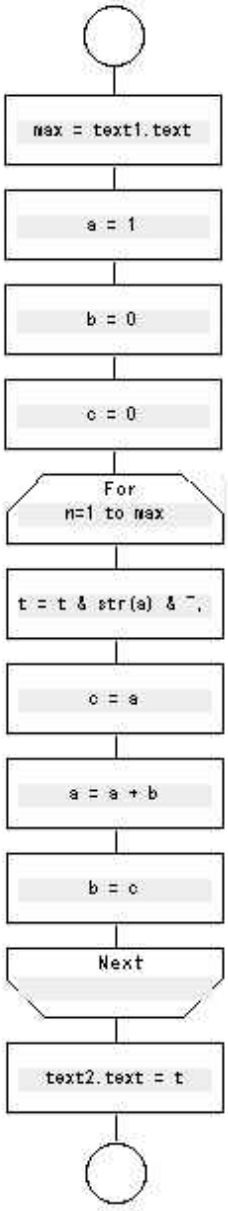

図12　フィボナッチ数列を求める
プログラム・フロー

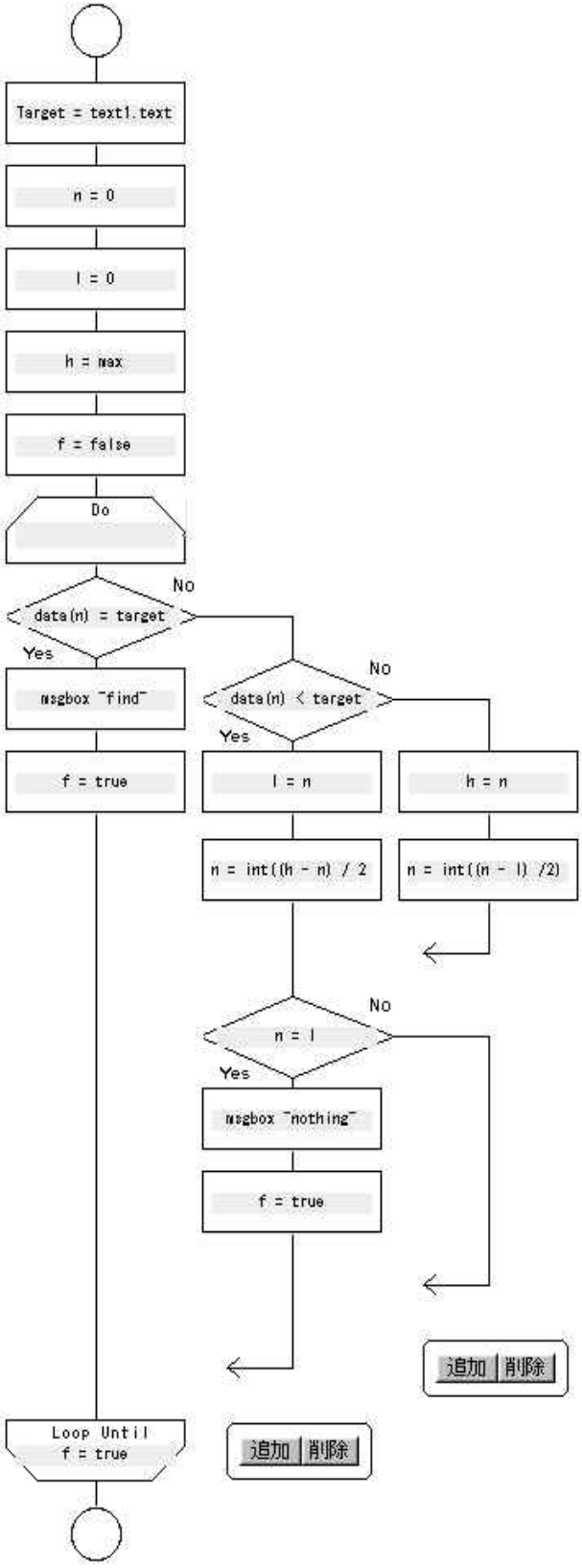

図15　二分探索フロー

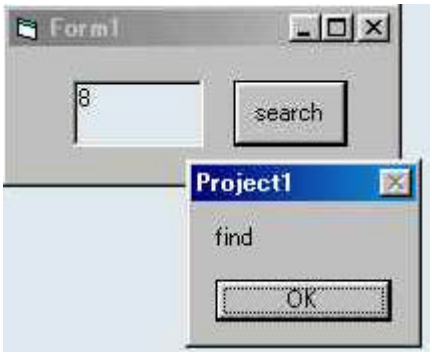

図 17　二分探索結果表示

入力された数字をデータと照合して find または nothing をかえす。

5.3　会話するプログラム作成

複雑なフローチャートとなる会話プログラムのフローチャートを作成して、巧くプログラムに変換されることを確認した。**図 18** に作成したプログラムの実行結果を示す。

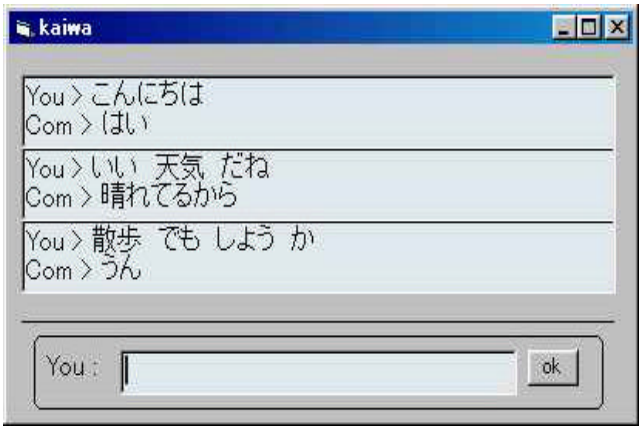

図 18　会話するプログラム実行結果

6　考察

このプログラムでは幾つかの制限がある。

- goto 文は使えない。上から下への構造化プログラミングの原則を重視する。
- if then else の中にもうひとつ if は入れられない。
- フローチャートの横方向は 5 部品までしかおけない。

関数は順次処理のハコのなかに関数名等を記述する。プロシジャーもメインプログラムではプロシジャー名を順次処理のハコの中に書く。プロシージャー自体は本プログラムを用い別途プロシジャーのフローチャートを書きプログラムコード化する。

現在のところテストで作成したプログラムは全て本プログラムを使って作成できたが今後更に様々なプログラムについて出来ないものはないか、出来ないとしたらどのような理由によるかを検討する。

7　まとめ

VB プログラミング初学者の学習に用いるとフローチャートをきちんと作る学習に有効である。

プログラムを機械で作ってしまうということは大量に作る、頻繁に変えることができる、試行錯誤的にいろいろ作る等のメリットがあると考えられる。更に実用的なものとする工夫、システム作成・プログラム作成の半自動化への拡張を進め、プログラム作成の生産性の向上をはかりたい。