四海學報，第八期（八十二年九月）225～243
私立四海工商專校

Computer Graphics of point, line, and N_points polygon on PC Super VGA Monitor

(PC Super VGA 螢幕上點，線，N_點多邊形　電腦繪圖)

Wen-Kuei Hsieh
謝文魁*

摘　要

本文討論利用 BRESENHAM 劃線演算法劃線在 PC Super VGA 螢幕上，同時比較一般直線演算法，DDA (Digital Differential Analyzer) 演算法，和 BRESENHAM 劃線演算法。

並擴展 BRESENHAM 劃線演算法爲 N_ 點多邊形演算法；用 N-S 圖和虛擬碼對 N_ 點多邊圖形作程式分析；最後使用 TURBO PASCAL 完成 N_ 點多邊形繪圖。

目　次

*謝文魁：銀行保險科專任講師

四海學報，第八期（八十二年九月）225～243
私立四海工商專校

Computer Graphics of point, line, and N_points polygon on PC Super VGA Monitor

Wen-Kuei Hsieh

ABSTRACT

A comparison among General Line Algorithm, Digital Differential Analyzer (DDA) Algorithm, and Bresenham's line Algorithm. Implementing Bresenham's line Algorithm on PC Super VGA monitor.

Extend Bresenham's line Algorithm to N_points Polygon Algorithm; Using N-S chart and Pseudo-code to N_points Polygon Algorithm in program analysis. Finally, implementing N_points Polygon Algorithm by using TURBO PASCAL computer language.

壹、作業環境

1-1 硬體

CPU　　：LEO PC/AT 486

速度　　：33 MHz

記憶體　：4 MB

1.44MB 軟碟　*1

1.2 MB 軟碟　*1

120 MB 硬碟　*1

顯示器　：Super VGA

1-2 軟體

1. 作業系統：MS-DOS 5.0 (MICROSOFT)

2. 程式語言：TURBO PASCAL (MICROSOFT)

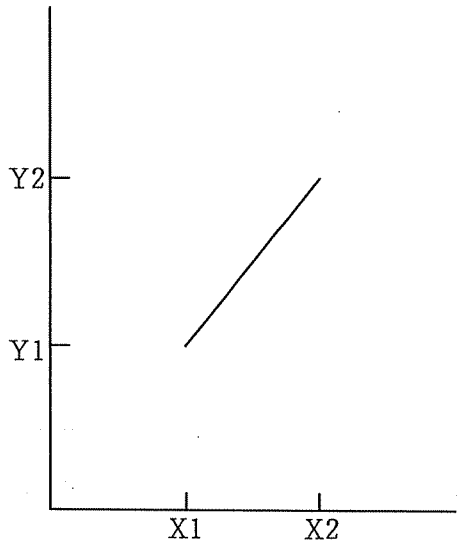


圖 2-1: 由 (X1, Y1) (X2,Y2) 組成線段

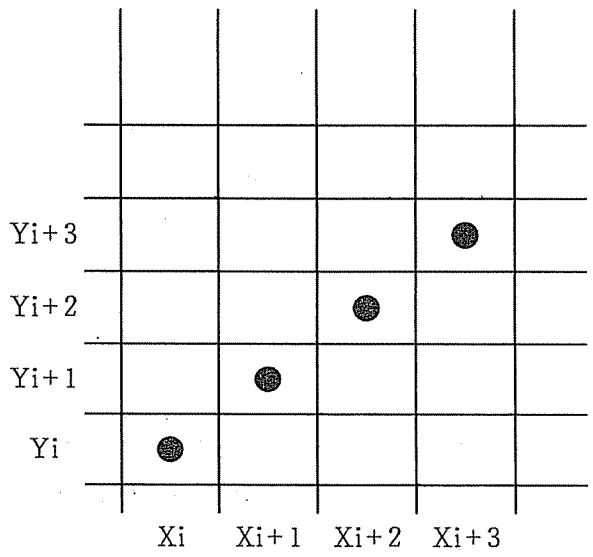


圖 2-2 ：可能的直線路徑

貳、線的求法

2-1　一般直線演算法

直線之基礎數學式爲：

$$Y=m*X+b \qquad (1)$$

其中：m：線的斜率，　b：Y的截距

$$m=(Y2-Y1)/(X2-X1) \qquad (2)$$

$$b=Y1-m*X1 \qquad (3)$$

$$\triangle Y=m*\triangle X$$

2-2　DDA 演算法

Digital Differential Analyzer (DDA) 演算法，採單位步驟，由某一軸座標位置（如：X軸），計算另一軸（Y軸）座標位置，分析如下：

CASE 1, 正斜率，如圖 2-1，當斜率小於或等於 1.

將X軸座標位置設爲1，然後陸續計算出Y軸座標位置，如

$$Y_{i+1} = Y_i+m \qquad (5)$$

CASE 2, 當斜率大於 1，Y軸往前推一單位，計算出相對應的X軸座標位置，如

$$X_{i+1} = X_i+(1/m) \qquad (6)$$

CASE 1 與 CASE 2 均採由左端點向右端點推進，如圖 2-1，若採由右端點向左端點推進，則需考慮下列數學式：

CASE 3, 當 $\triangle X=-1$，採

$$Y_{i+1} = Y_i - m \qquad (7)$$

CASE 4, 當 $\triangle Y=-1$，則採

$$X_{i+1} = X_i - (1/m) \qquad (8)$$

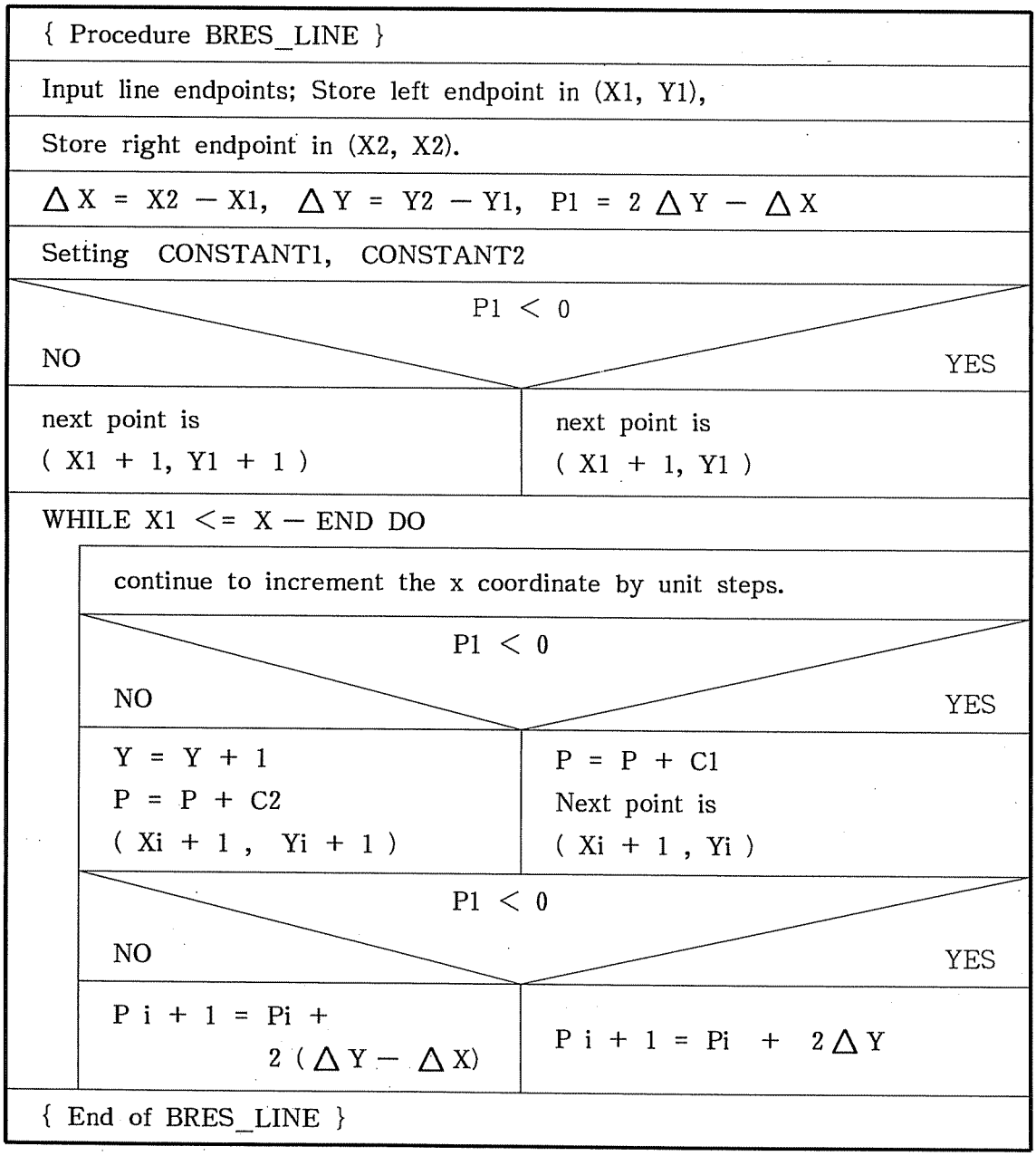
DDA 演算法的計算速度比一般直線演算法要快，因爲它免除了公式(1)中的乘法運算；然而 DDA 演算法使用了除法運算和浮點（實數）運算，因此，它的計算速度仍然受到影響。

2-3　BRESENHAM 劃線演算法

BRESENHAM 劃線演算法採整數運算，找出最接近的整數X軸，Y軸座標值，進而

描繪出直線路徑，組成直線的點位置，在螢幕上，呈獻於四方點陣方格內，如圖 2-2。惟，此一演算法斜率侷限於 0 到 1 之間。

用 N-S 圖 (NASSI-SHNEIDERMAN CHARTS AN ALTERNATIVE TO FLOWCHARTS FOR DESIGN) 和虛擬碼分析 BRESENHAM 劃線演算法如下：



N-S 圖-1 Procedure BRES-LINE

參、BRESENHAM直線劃法

Implementing Bresenham's line algorithm to generate lines with slope between 0 and 1.

```
program PolyLinn;
uses
    graph;
type
    points=array[1..5] of integer;
var
    x,y                 : points;
    GraphDriver         : integer;
    GraphMode           : integer;
    ErrorCode           : integer;
    n, I, inX, inY      : integer;
procedure Bresline (x1,y1,x2,y2:integer);
var
    dx,dy,a,b,p,x_end,y_end,const1,const2   : integer;
begin
    dx := abs(x1-x2);
    dy := abs(y1-y2);
    p  := 2 * dyy - dxx;
    const1 := 2 * ddy;
    const2 := 2 * (dyy - dxx);
    {determine which point to use as start, which as end}
    IF x1  >  x2 then
                begin
                a  := x2;   x_end  := x1;
```

```
                b  := y2;
                end
        else
                begin
                a  := x1;   x_end  := x2;
                b  := y1;
    end;
    putPixel(a,b, green);
    while a < x_end do
    begin
                a := a + 1;
                If P < 0 then
                     P := p + const1
                else begin
                     b := b + 1
                     p := p + const2
                end;
                putPixel(a,b, green)
    end
end; { end of BresLine }
procedure PolyLine ( n: integer; x,y: points);
var
    cnt : integer;
begin
    GraphDriver := Detect;
    InitGraph(GraphDriver, GraphMode, 'E:\tp5_5\graphics');
    ErrorCode :=GraphResult;
```

```
    if ErrorCode <> grOK then
    begin
                Writeln('(Please check your graphics card!)');
                Writeln('Program aborted...');
                Readln;
                Halt(1);
    end;
    If n <. 2 then      putPixel ( x[1], y[1], green)
    else begin
     cnt := 1;
                BresLine(x[cnt], y[cnt], x[cnt+1], y[cnt+1]);
    end;
end;   { End of PolyLine }
{ Main Program }
begin
    write('Input Number of Points 2 which you need :');
    n       := 2;
    writeln('Input ',n,' pairs of two numbers ');
    writeln('( Example: X[1] Y[1] ?: 100 200 )');
    for I := 1 to n do
    begin
    write('X[',I,']  Y[',I,'] ?: ');
    readln(inX,inY);
    x[I] := inX;
    y[I] := inY;
    end;
    PolyLine (n,x,y);
end.  { End of Main Program }
```

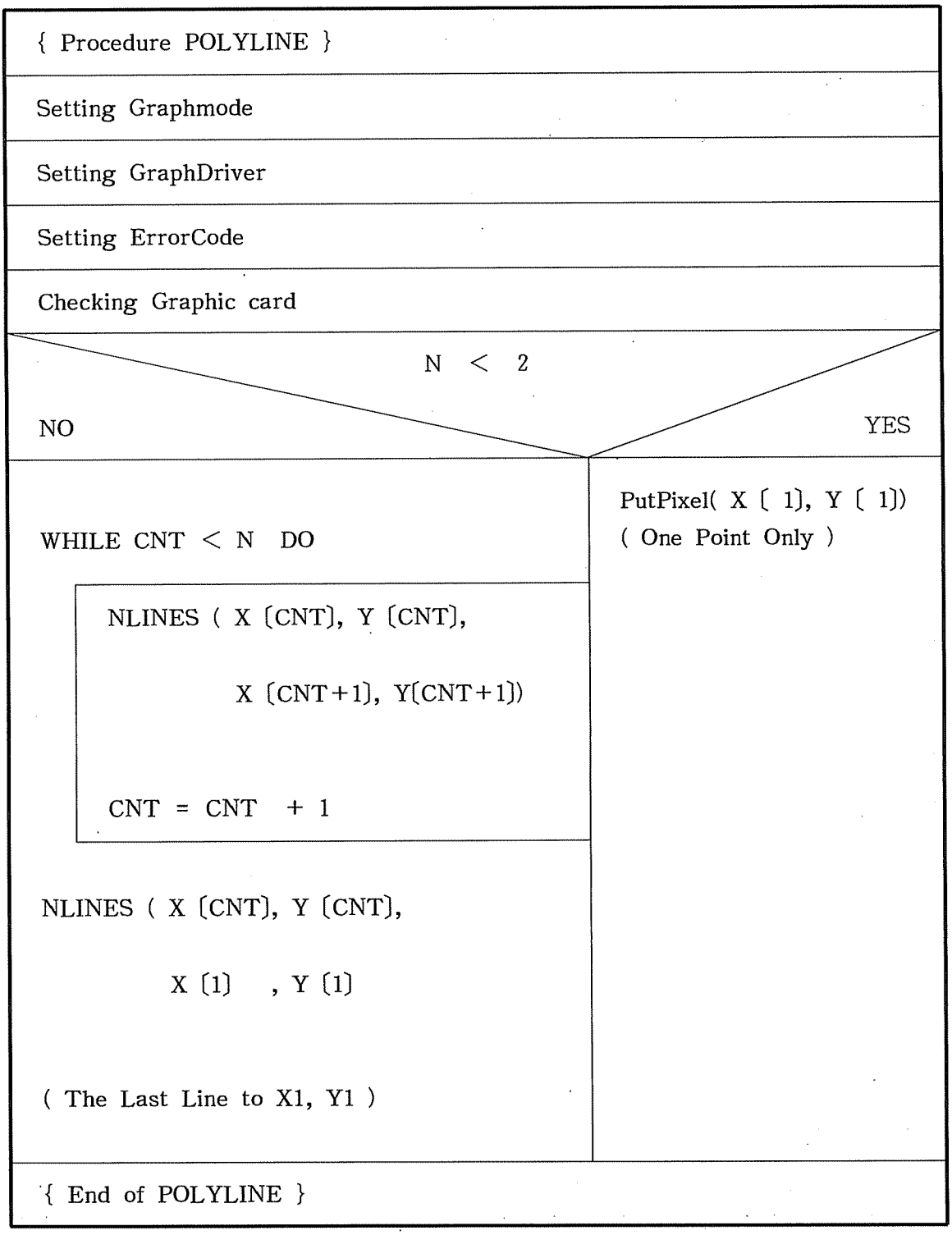
肆、N_點多邊形劃法

4-1 N_點多邊形演算法（當斜率大於1或斜率小於0時：）

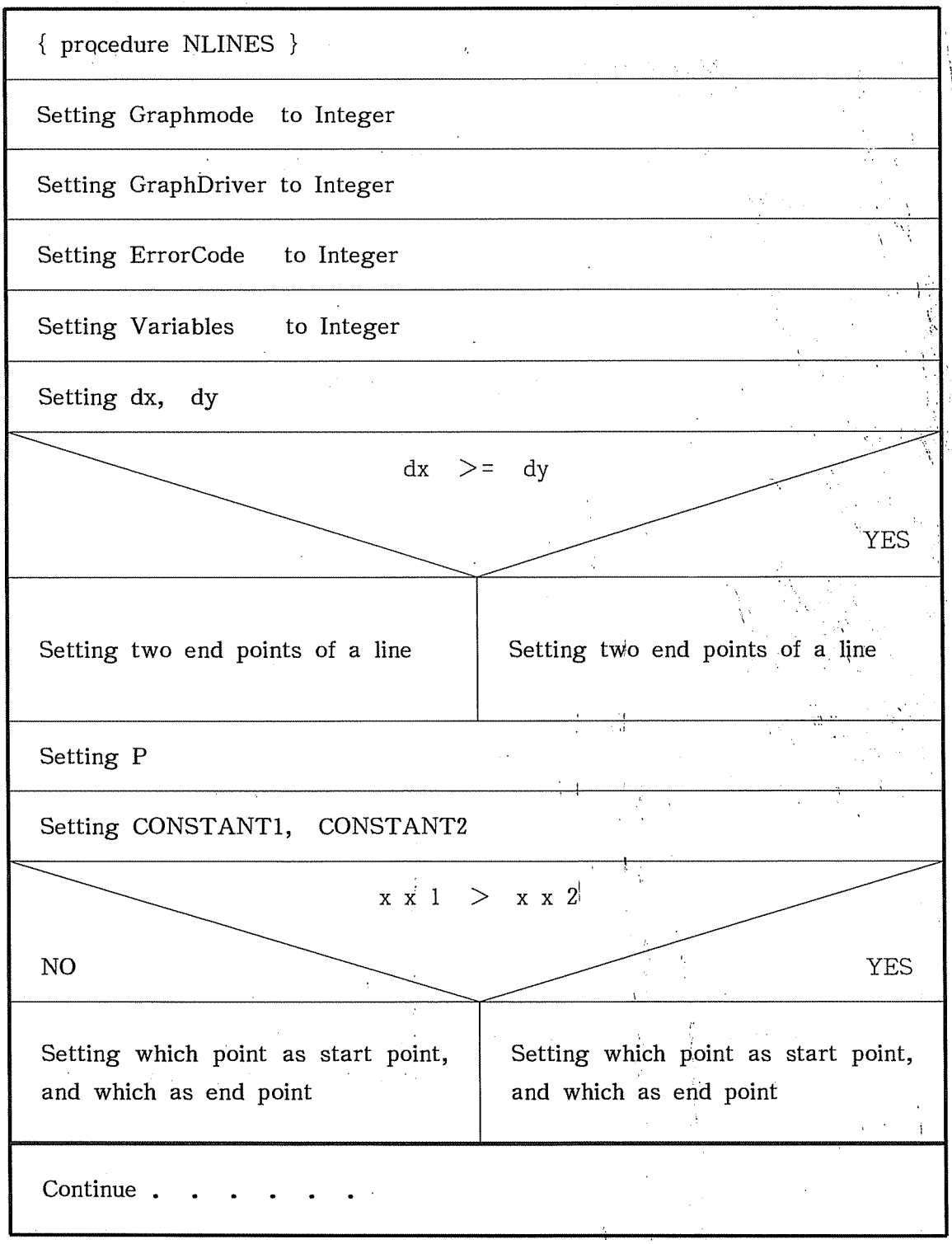
擴展 BRESENHAM 劃線演算法，將斜率大於 1 或斜率小於 0 的情況，用 N-S 圖和虛擬碼分析如下：

{ Main Program }
Input number of points, n, which you want to draw
read n
Input n pairs of two numbers
Example : X〔 1〕 Y〔 1 〕? : 100 200
FOR I : = 1 TO N DO
Read Xi, Yi
X〔 I〕 = Xi
Y〔 I〕 = Yi
POLYLINE (N, X, Y)
{ End of Main Program }

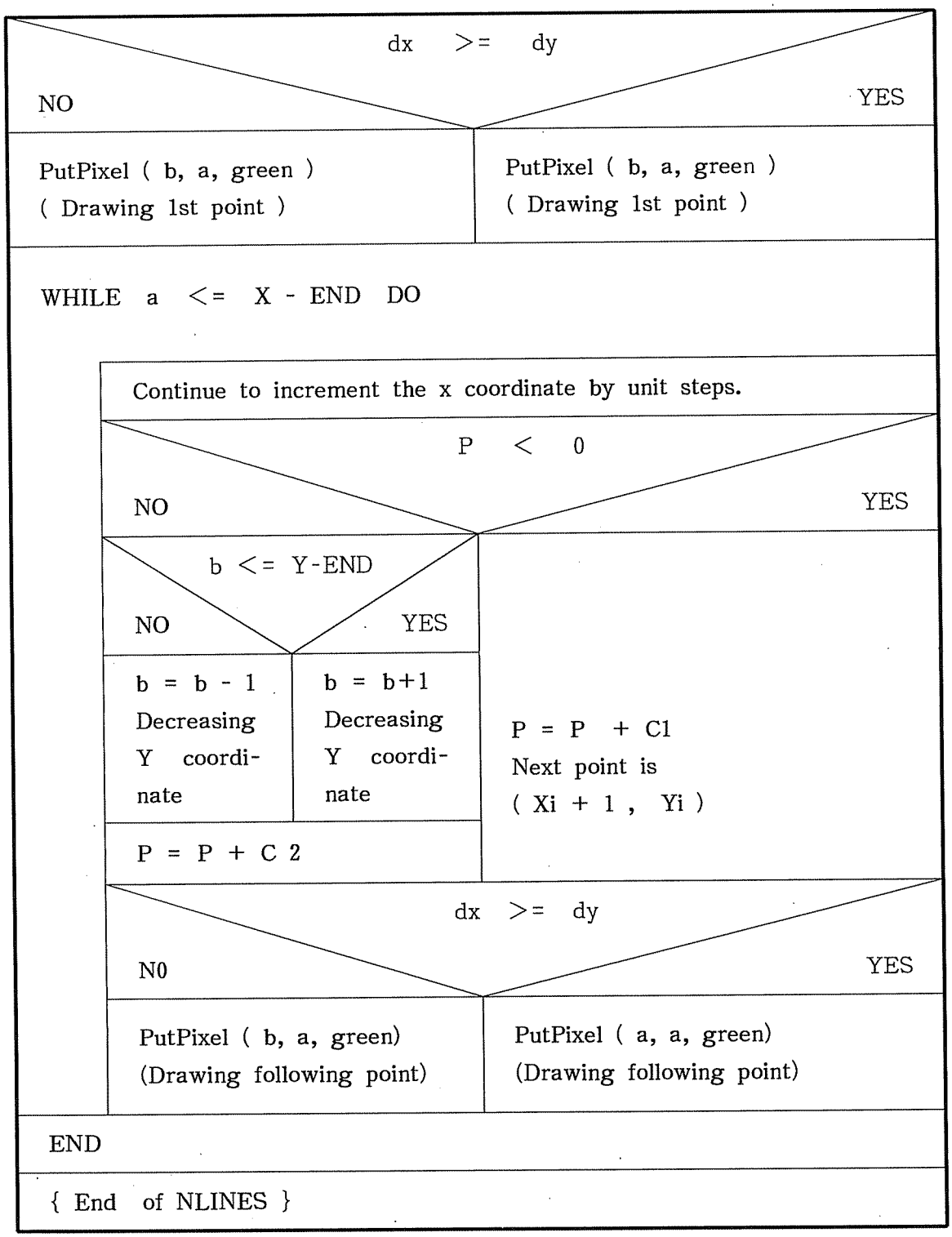
N-S 圖-2　Main Program



N-S 圖-3 Procedure POLYLINE



N-S 圖-4-1 Procedure NLINES



N-S 圖-4-2 Procedure NLINES

4-2 N_點多邊形劃法

Extend Bresenham's line algorithm to N_points polygon algorithm to generate lines with any slope.

Implement the 'polyline' command that displays the set of straight lines between n input points.

That is, use the line drawing routine to draw a polygon of n sides. If n=1, then the routine displays a single point.

```
program PolyLinn;
uses
        graph;
type
        points = array[1..100] of integer;
var
        x,y                     : points;
        GraphDriver             : integer;
        GraphMode               : integer;
        ErrorCode               : integer;
        n,I,inX,inY             : integer;
procedure BresLine (x1,y1,x2,y2:integer);
var
        dx,dy,a,b,x_end,y_end,const1,const2     : integer;
        dxx,dyy,xx1,xx2,yy1,yy2,p,X,Y                   : integer;
begin
        dx := abs(x1 - x2);
        dy := abs(y1 - y2);
        if dx >= dy then
          begin
          dxx  := dx;  dyy  := dy;
```

```
            xx1  := x1;  xx2  := x2;
            yy1  := y1;  yy2  := y2;
            end
          else
            begin
            dxx  := dy;  dyy  := dx;
            xx1  := y1;  xx2  := y2;
            yy1  := x1;  yy2  := x2;
end;
p :=2 * dyy - dxx;
const1 := 2 * dyy;
const2 := 2 * (dyy - dxx);
If xx1 > xx2 then
          begin
          a  := xx2;  x_end  := xx1;
          b  := yy2;  y_end  :=yy1
          end
else
          begin
          a  := xx1;  x_end  := xx2;
          b  := yy1;  y_end  := yy2
end;
          If dx >= dy then
                        putPixel(a,b, green)
          else begin
                        putPixel(b,a,green)
          end;
          while a < x_end do
          begin
                        a := a + 1;
```

```
                    If P < 0 then
                        P := P + const1
                    else begin
                        If b <= y_end  then   b := b + 1
                        else begin
                            b := b - 1
                            end;
                        p := p + const2
                    end;
                    If dx >= dy then
                        putPixel(a,b, green)
                    else begin
                        putPixel(b,a,green)
                            end;
        end
end;  { End of BresLine }
procedure PolyLine ( n: integer; x,y: points );
var
        cnt : integer;
begin
        GraphDriver := Detect;
        InitGraph(GraphDriver,GraphMode,'E:\tp5_5\graphics');
        ErrorCode := GraphResult;
        if ErrorCode <> grOK then
        begin
                    Writeln('(Please check your graphics card!)');
                    Writeln('Program aborted...');
```

```
                    Readln;
                    Halt(1);
        end;
        If n < 2 then  putPixel ( x[1], y[1], green)
        else begin
         cnt := 1;
         while cnt < n do
         begin
                    BresLine(x[cnt], y[cnt], x[cnt+1], y[cnt+1]);
                    cnt := cnt + 1;
         end;
                    BresLine(x[cnt], y[cnt], x[1], y[1]);
        end;
end;  { End of PolyLine }
{ Main Program }
begin
        write('Input Number of Points, n, which you want :');
        readln(n);
        writeln('Input ',n,' pairs of two numbers ');
        writeln('( Example: X[1] Y[1] ?: 100 200 )');
        for I := 1 to n do
        begin
        write('X[',I,']  y[',I,'] ?: ');
        readln(inX,inY);
        x[I] := inX;
        y[I] := inY;
        end;
        PolyLine (n,x,y);
end.  { End of Main Program }
```

伍、結　論

1. 一般直線演算法，由於使用乘法運算，因此計算速度較慢。
2. Digital Differential Analyzer (DDA) 演算法雖然比一般直線演算法來得快，但是 DDA 的計算速度仍然受到影響，因爲它使用了除法運算和浮點（實數）運算。
3. BRESENHAM 劃線演算法使用整數運算，因此可以快速且有效率地劃直線，但是斜率侷限於 ">= 0" 或 "<= 1" 的情況。
4. 本文擴展 BRESENHAM 劃線演算法至任何斜率（N_點多邊形演算法），即斜率大於 1 和斜率小於 0 的情況。

   用 N-S 圖和虛擬碼對 N_點多邊圖形作程式分析；最後使用 TURBO PASCAL 完成 N_點多邊形繪圖。
5. 本文只探討電腦繪圖的一小部份，仍有許主題待後續研究。

參考書目

1. TURBO PASCAL 使用手册，MICROSOFT CO.
2. 薛文證、電腦繪圖學、松崗電腦圖書。
3. Donald Hearn, and M. Pauline Baker. Computer Graphics, Prentice-Hall.
4. Bresenham, J. E.. "Algorithm for Computer Control of Digital Plotter," IBM Systems Journal.
5. Enderle, G., K. Kansy, and G. Pfaff. Computer Graphics Programming: GKS -The Graphics Standard.
6. Marcus, A. "Graphic Design for Computer Graphics," IEEEE Computer Graphics And Applications.
7. Newman, W. M.. Principles of Interactive Computer Graphics. New York: Mcgraw-Hill.
8. Cornelia M. Yoder and Marilyn L. Schrag. NASSI-SHNEIDERMAN CHARTS AN ALTERNATIVE TO FLOWCHARTS FOR DESIGN.