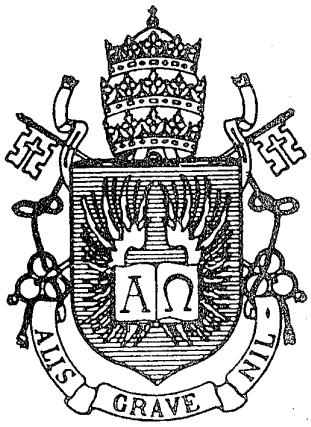

PUC

Series: Monografias em Ciência da Computação
Nº 6/82

INSTRUCTIONAL GRAPHICS PACKAGES TO BE USED WITH A LINE PRINTER

A.L. Furtado
A.A.B. Furtado
F.A. Messeder

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Series: Monografias em Ciência da Computação, Nº 6/82
Series Editor: Marco A. Casanova July 1982

INSTRUCTIONAL GRAPHICS PACKAGES TO BE USED WITH A LINE PRINTER*

A.L. Furtado
A.A.B. Furtado
F.A. Messeder

* This work has been sponsored in part by FINEP.

Abstract

Two similar packages - one in WATFIV-S and the other in Pascal - for teaching elementary graphics are completely described. The packages are to be used with conventional line printers, offering a reasonable alternative to installations where either specific graphics hardware is not available or the size of the student classes makes its usage inconvenient.

Keywords: Graphics, problem solving, introduction to programming.

Resumo

Dois pacotes semelhantes - um em WATFIV-S e o outro em Pascal - para o ensino elementar de processamento gráfico são completamente descritos. Os pacotes são para serem usados com impressoras convencionais, o que oferece uma alternativa razoável para instalações onde ou o equipamento específico não é disponível ou o tamanho das turmas de alunos torna seu uso inconveniente.

Palavras chaves: Processamento gráfico, resolução de problemas, introdução a programação.

1. Introduction.

It has been claimed that introductory courses on programming, with an emphasis on problem solving, should not be restricted to purely numerical applications. In [1] a number of elementary graphics problems are proposed, to be attacked by students using the UCSD Pascal implementation, assuming that the installation has the appropriate graphics equipment.

Besides agreeing with Seymour Papert's conjecture that graphics are a good tool to develop problem solving skills in students, we also acknowledge the importance of graphics as the basis for important practical applications, especially in the area of computer aided design.

In our university we offer a course on Intruduction to Computer Science, which is compulsory for all undergraduate engineering and natural sciences students. Recently we decided to incoporate some elementary graphics applications in the curriculum. Although the purchase of graphics equipment had already been contemplated, it is not likely that relatively large classes of students would have access to it.

The present report describes the solution adopted, which we hope will be useful to installations in the same situation as ours.

Our undergraduate students write their first programs in WATFIV-S, a structured dialect of FORTRAN (rather close to FORTRAN-77) designed by the University of Waterloo. The WATFIV-S compiler is ideal for handling small student jobs, being core-resident and issuing good compile-time and run-time error messages.

What we did was to create a sub-program library, accesible to the WATFIV-S compiler, containing eight sub-routines (plus one auxiliary function) to provide a rudimentary but hopefully adequate graphics capability using a line printer.

Another slightly more versatile package was produced in Pascal, to be used by our M. Sc. computer science students as a first step towards graphics applications. An interesting M. Sc. project in any installation is to design some higher-level language (say, like [4]) to be compiled or pre-processed, into calls to sub-routines performing the basic graphics operations.

Both packages are to a large extent compatible with the UCSD Pascal extensions. This makes it possible for us to use or adapt the set of problems in [1].

Section 2 describes the WATFIV-S package and section 3 the Pascal package. Section 4 offers some remarks directed to prospetive implementors. Section 5 contains the conclusions. Complete listings of the packages appear in appendices A and B, and example pictures are shown in appendix C.

2. The WATTLY-S package

The sub-program headings and their usage are explained below. A number of "system" variables are shared through a COMMON area, which however does not have to be declared in the users' programs.

2.1. SUBROUTINE CLEAR(C) -

Initializes the screen, filling it with the character supplied in parameter C. In the screen both the x- and the y-coordinates go from -35 to 35. CLEAR places the "pen" at point (0,0), in the up position, turned along the zero-degree angle. Often, but not necessarily, the character in C will be a blank; any character can be used.

2.2. SUBROUTINE COLOR(C) -

Defines as C the character to be employed in the next lines to be drawn as the pen moves. If C is zero the pen goes to the up position and nothing is drawn as it moves, whereas with any other character the pen is down. Note that, if the screen has been initialized with a non-blank character and COLOR is called with C being a blank, the drawing will be done in "negative".

2.3. SUBROUTINE TURN(ANGLE) -

The positive or negative integer in ANGLE is added to the angle along which the pen is directed. The angle is kept in degrees.

2.4. SUBROUTINE TURNTO(ANGLE) -

The positive, null or negative integer in ANGLE becomes the angle along which the pen is directed.

2.5. SUBROUTINE MOVE(DIST) -

The pen moves, in the direction of the current angle, DIST screen units, where DIST is an integer. If the pen is up it is merely displaced; otherwise a straight line is drawn with the current character. If DIST is zero and the pen is down, the character is drawn at the current position but the pen is not displaced.

2.6. SUBROUTINE MOVETO(XPOS,YPOS) -

The pen moves to the point whose coordinates are given in screen units by the integers (XPOS,YPOS). If the pen is up it is merely displaced; otherwise a straight line is drawn with the current character. If the coordinates of the current point already are (XPOS,YPOS) and the pen is down, the character is drawn, but the pen is not displaced.

2.7. SUBROUTINE WHERE(XPOS,YPOS,DIR) -

The coordinates of the current point and the current angle are assigned to the integer variables XPOS,YPOS,DIR, respectively.

2.8. SUBROUTINE SHOW -

The contents of the screen are printed on the line printer.

2.9. INTEGER FUNCTION ROUND(X) -

The value of the real X is rounded to the next integer. If X is positive, 0.5 is added to its value before truncation; if X is negative, -0.5 is added. This is an auxiliary function, called from some of the sub-routines.

Note: MOVE and MOVETO issue error messages if their execution would cause the pen to wander off the screen. The contents of the screen prior to the erroneous move are printed and the program execution is terminated.

The sub-programs are listed in appendix A.

The sample program below draws a "spirolateral", an example taken from [1]. The input values are angle = 120, size = 10, m = 3, n = 5, seq = 1112. The program can be changed to produce a "negative" version, by passing '#' (for example) and ' ' as arguments to CLEAR and COLOR, respectively. Both versions are shown as the first two figures in appendix C.

```
$JOB            FUR$$$$$,KP=29,NOEXT,NOWARN
      CHARACTER*1 SEQ(10)
      INTEGER M,N,I,K,ANGLE,SIZE,X,Y,A
      READ, ANGLE,SIZE,M,N
      DO 10 I = 1,N
         READ 1, SEQ(I)
    1    FORMAT(A1)
   10 CONTINUE
      CALL CLEAR(' ')
      CALL MOVETO(0,17)
      CALL COLOR('-')
      DO 2 K = 1,M
        DO 3 I = 1,N
          CALL MOVE(SIZE * I)
          IF (SEQ(I) .EQ. 'R') THEN DO
            CALL TURN(-ANGLE)
          ELSE DO
            CALL TURN(ANGLE)
          END IF
          CALL WHERE(X,Y,A)
          PRINT, X,Y,A
    3   CONTINUE
    2 CONTINUE
      CALL SHOW
      STOP
      END
```

3. The Pascal package

The procedure headings and their usage are explained below. They closely resemble those in [1].

3.1. PROCEDURE CLEARSCREEN(C: INTEGER) -

Initializes the screen, filling it with the grey-scale level indicated by the integer C. If C is zero, the screen will be blank. The grey-scale lies in the sub-range 0..7 and follows the overprinting patterns suggested in [3]. In the screen both the x- and the y- coordinates go from -35 to 35. CLEARSCREEN places the "pen" at point (0,0), in the up position, turned along the zero-degree angle.

3.2. PROCEDURE PENCOLOR(COLOR: INTEGER) -

Defines as the integer C the grey-scale level to be employed in the next lines to be drawn as the pen moves. If C is the pre-defined constant NONE the pen goes to the up position, whereas with a value from 0 to 7 the pen is down.

3.3. PROCEDURE TURN(ANGLE: INTEGER) -

The positive or negative integer in ANGLE is added to the angle along which the pen is directed. The angle is kept in degrees.

3.4. PROCEDURE TURNTO(ANGLE: INTEGER) -

The positive, null or negative integer in ANGLE becomes the angle along which the pen is directed.

3.5. PROCEDURE MOVE(DISTANCE: INTEGER) -

The pen moves, in the direction of the current angle, DISTANCE screen units, where DISTANCE is an integer. If the pen is up it is merely displaced; otherwise a straight line is drawn in the current grey-scale level. If DISTANCE is zero and the pen is down, a point is drawn at the current position but the pen is not displaced.

3.6. PROCEDURE MOVETO(XPOS,YPOS: INTEGER) -

The pen moves to the point whose coordinates are given in screen units by the integers (XPOS,YPOS). If the pen is up it is merely displaced; otherwise a straight line is drawn in the current grey-scale level. If the coordinates of the current point already are (XPOS,YPOS) and the pen is down, a point is drawn at the current position but the pen is not displaced.

3.7. PROCEDURE WHEREAMI(VAR XPOS,YPOS,DIRECTION: INTEGER) -

The coordinates of the current point and the current angle are assigned to the integer variables XPOS, YPOS, DIRECTION, respectively.

3.8. PROCEDURE DISPLAY -

The contents of the screen are printed on the line printer.

3.9. PROCEDURE FILL(X,Y,C: INTEGER) -

A closed area is entirely "painted" in the current grey-scale level. The integer C, in the sub-range 0..7, must be the present grey-scale level of the points inside the closed area. By "closed" area we mean that the area, which may have any shape, must be totally surrounded by points in a grey-scale level distinct from that of the points inside. The integers (X,Y) must be the coordinates of any point lying inside the closed area. Note that the combinned use of the "backgound color" (indicated by CLEARSCREEN), the colors of the lines (drawn with MOVE or MOVETO) and the painting of areas provide the ability to produce contrast.

3.10. PROCEDURE PIXEL(X,Y: INTEGER) -

Paints the point whose coordinates are given by the integers (X,Y) in the current grey-scale level. This is an auxiliary procedure called from some of the others.

Note: MOVE and MOVETO issue error messages if their execution would cause the pen to wander off the screen. The contents of the screen prior to the erroneous move are printed and the program execution is terminated.

The procedures are listed in appendix B.

The sample program below draws the 4-colored spirolateral shown as the third figure in appendix C. Notice the declaration of the constant NONE and of the five global variables (starting with SYS), used by the procedures to store the screen and the current values of the coordinates, the angle and the grey-scale level.

To help finding out points within the areas that we wish to paint, we can use the information about the points reached through the successive MOVES by calling WHEREAMI in a previous run (see the CALL WHERE in the WATFIV-S sample program at the end of section 2).

```
PROGRAM SPIROLATERAL(INPUT,OUTPUT);
  CONST NONE = -1;
  VAR SYSTAB: PACKED ARRAY(.1..71,1..119,1..4.) OF CHAR;
      SYSC,SYSA: INTEGER;
      SYSX,SYSY: REAL;
      M,N,I,K,ANGLE,SIZE:INTEGER;
      SEQ: ARRAY(. 1..10 .) OF CHAR;


       . . . . package procedures . . . .


      BEGIN
        READLN(ANGLE,SIZE,M,N);
        FOR I := 1 TO N DO
          READ(SEQ(.I.));
        CLEARSCREEN(4);
        MOVETO(0,17);
        PENCOLOR(0);
        FOR K := 1 TO M DO
          FOR I := 1 TO N DO
            BEGIN
              MOVE(SIZE*I);
              IF SEQ(.I.) = 'R' THEN TURN(-ANGLE)
              ELSE TURN(ANGLE)
            END;
        PENCOLOR(0); FILL(-15,20,4);
        PENCOLOR(0); FILL(10,30,4);
        PENCOLOR(0); FILL(0,-20,4);
        PENCOLOR(7); FILL(5,15,4);
        PENCOLOR(7); FILL(-15,-5,4);
        PENCOLOR(7); FILL(10,-12,4);
        PENCOLOR(2); FILL(-3,2,4);
        PENCOLOR(2); FILL(7,3,4);
        PENCOLOR(2); FILL(0,-5,4);
        PENCOLOR(0); FILL(0,8,4);
        PENCOLOR(0); FILL(-5,-2,4);
        PENCOLOR(0); FILL(5,-2,4);
        DISPLAY
      END.
```

4. Implementation considerations

If displacements were achieved only by MOVETO there would be no inconvenience in storing the coordinates as integers. However we soon learned that, after a number of calls to MOVE, we might not be precisely where we should expect. Thus the coordinates are kept internally in floating point representation. The angle is kept in degrees modulo 360 and is converted into radians when needed inside MOVE.

Both MOVETO and MOVE are based on the simple-DDA line drawing algorithm [2]. Since ours is a software implementation, there is no harm using division as well as the computation of sines and cosines.

The FILL procedure in the Pascal version is based on the recursive algorithm in [2]. After painting a point it calls itself to process the points immediately above, below, on the right and on the left. Unlike [2], it does not have to look at the other four neighbouring points; notice that, if this were permitted, FILL might erroneously cross the limits established by a closed curve at some place where the curve had two consecutive points lying, say, on a 45 degrees line.

The reader may want to try grey-scales different from the one that we took from [3], perhaps because of the different fonts available on his printer. If so, he is advised to change both the procedure PIXEL and the function ISCOLOR internal to procedure FILL. In the latter we compare grey-scale levels by only looking at the first character of the overprinting patterns, which were arranged so as to have the first character distinct for each level.

Several aspects of the implementation are machine-dependent. Overprinting, for example, is achieved on our IBM machine by placing a '+' in the first position of the output buffer. The WATFIV-S version assumes that the printer works with 10 characters per inch horizontally and 8 characters per inch vertically. In the Pascal version we assumed 6 characters per inch vertically, having in mind another printer device. These considerations determined the the scale factor for balancing the horizontal and vertical displacements. The actual dimensions of the screen, on the other hand, must be adjusted to the size of the printed page.

It should not be difficult for installations not having WATFIV-S to convert the WATFIV-S package into ordinary FORTRAN, particularly the 1977 standard. As to the Pascal package, the only non-standard feature that we used is the HALT statement in procedures MOVE and MOVETO for terminating the execution in the event of attempts to wander off the screen; HALT exists in the Pascal 8000 implementation of Pascal for the IBM 370 (documentation available from the Australian Atomic Energy Commission).

Comparing the two packages, we would say that the WATFIV-S package is simpler, less powerful, and uses less space; we chose to allow different characters for drawing, instead of providing overprinting. The Pascal version has overprinting, the ability to produce contrast, and recursion, which is very convenient for certain applications (see the GROWTREE and HILBERT programs in [1]). The major drawback of the Pascal package is the lack of external procedures; they are allowed in certain Pascal extensions but the lack of something like the COMMON feature of FORTRAN seems to be a hindrance here.

One may wonder if the operations provided indeed constitute a complete basic set. Even if they do, it may be useful to add other operations. For example, some authors argue that special algorithms for drawing circles are needed. With the present set of operations, circles can be formed using straight lines to draw regular many-sided (say, 40 sides) regular polygons. Another possibility is to perform a loop changing the current angle from zero, by suitable steps, up to 360 degrees, at each iteration moving along the radius (with the pen in the up position) and executing a MOVE(0) with the pen down.

An aspect of graphics terminals that we badly miss is the interactive mode. It is much better to see the drawing developing before our eyes, and change it as needed, than to wait until the printed output is delivered. We tried to alleviate this difficulty in two ways: by causing the screen to be printed in case of erroneous moves, to allow the user to see how far he managed to go and to ponder how to correct his possible mistakes, and by allowing the inspection of the drawings through the screen of our conventional IBM 3270 terminals (which however do not show the overprinting and have a different horizontal/vertical ratio).

5. Conclusion

With instructional packages motivation is a crucial issue. The pictures obtained with a line printer cannot, of course, be compared with those made possible with more specialized equipment. Figure 3.1 in reference [1] is a case in point: although we managed to produce it on our line printer the result was barely recognizable.

Accordingly, one must choose kinds of applications where the drawings produced are still sufficiently pleasant to the eye, so as not to discourage the student.

Finally, the applications must be related to the subject of each student class, with a more artistic or technological bent, according to the case.

Acknowledgement

The authors are grateful to M. Stanton for useful suggestions.

References

1. K. L. Bowles - "Problem solving using Pascal" - Springer-Verlag (1977).

2. W. M. Newman and R. F. Sproull - "Principles of interactive computer graphics" - Mc Graw-Hill (1979).

3. L. Press - "Computers and serial imagery" - in "Artist and computer" - R. Leavitt (ed.) - Creative Computing Press (1976).

4. C. J. van Wyk - "A higher-level language for specifying pictures" - ACM Trans. on Graphics - April (1982).

APPENDIX A

```
      SUBROUTINE CLEAR(C)
        INTEGER SYSA,I,J
        REAL SYSX,SYSY
        CHARACTER C*1,SYSTAB*1(71,89),SYSC*1
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        DO 1 I=1,71,1
           DO 2 J=1,89,1
              SYSTAB(I,J)=C
2          CONTINUE
1       CONTINUE
        SYSX=45.0
        SYSY=36.0
        SYSA=0
        SYSC='0'
        RETURN
        END

      SUBROUTINE COLOR(C)
        REAL SYSX,SYSY
        INTEGER SYSA
        CHARACTER C*1,SYSC*1,SYSTAB*1(71,89)
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        SYSC = C
        RETURN
        END

      SUBROUTINE TURN(ANGLE)
        INTEGER ANGLE,SYSA
        REAL SYSX,SYSY
        CHARACTER SYSTAB*1(71,89),SYSC*1
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        SYSA = MOD(SYSA + ANGLE,360)
        RETURN
        END

      SUBROUTINE TURNTO(ANGLE)
        INTEGER ANGLE,SYSA
        REAL SYSX,SYSY
        CHARACTER SYSC*1,SYSTAB*1(71,89)
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        SYSA = MOD(ANGLE,360)
        RETURN
        END
```

```
      SUBROUTINE MOVE(DIST)
        INTEGER DIST,SYSA,IX,IY,NX,NY,K,L,ROUND
        REAL RADS,SYSX,SYSY,NEWX,NEWY,X,Y,DX,DY
        CHARACTER SYSTAB*1(71,89),SYSC*1
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        IF(DIST .NE. 0) THEN DO
          RADS = SYSA * 0.0174532925
          NEWX = DIST*1.25*COS(RADS) + SYSX
          NEWY = DIST*SIN(RADS) + SYSY
          NX = ROUND(NEWX)
          NY = ROUND(NEWY)
          IF(NX .GT. 89 .OR. NY .GT. 71 .OR.
     *       NX .LT. 1 .OR. NY .LT. 1) THEN DO
            SYSTAB(ROUND(SYSY),ROUND(SYSX)) = '@'
            CALL WHERE(IX,IY,K)
            SYSX = NEWX
            SYSY = NEWY
            CALL WHERE(NX,NY,K)
            PRINT, ' ERRO NA MOVE'
            PRINT, ' PENA AGORA EM  (',IX,IY,') ANGULO',K
            PRINT, ' PENA IRIA PARA (',NX,NY,')'
            CALL SHOW
            STOP
          END IF
          IF(SYSC .NE. '0') THEN DO
            IX = ROUND(SYSX)
            IY = ROUND(SYSY)
            L = IABS(NX - IX)
            IF(IABS(NY - IY) .GT. L) THEN DO
              L = IABS(NY - IY)
            END IF
            DX = FLOAT(NX - IX)/FLOAT(L)
            DY = FLOAT(NY - IY)/FLOAT(L)
            X = IX + 0.5
            Y = IY + 0.5
            DO 1 K = 1,L,1
              SYSTAB(IFIX(Y),IFIX(X)) = SYSC
              X = X + DX
              Y = Y + DY
1           CONTINUE
            SYSTAB(NY,NX) = SYSC
          END IF
          SYSX = NEWX
          SYSY = NEWY
        ELSE DO
          IF (SYSC .NE. '0') THEN DO
            SYSTAB(ROUND(SYSY),ROUND(SYSX)) = SYSC
          END IF
        END IF
        RETURN
        END
```

```
SUBROUTINE MOVETO(XPOS,YPOS)
  INTEGER XPOS,YPOS,SYSA,IX,IY,NX,NY,K,L,ROUND
  REAL SYSX,SYSY,NEWX,NEWY,X,Y,DX,DY
  CHARACTER SYSTAB*1(71,89),SYSC*1
  COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
  IF(IABS(XPOS) .GT. 35 .OR. IABS(YPOS) .GT. 35) THEN DO
    SYSTAB(ROUND(SYSY),ROUND(SYSX)) = '@'
    CALL WHERE(IX,IY,K)
    PRINT, ' ERRO NA MOVETO'
    PRINT, ' PENA AGORA EM  (',IX,',IY,')'
    PRINT, ' PENA IRIA PARA (',XPOS,',YPOS,')'
    CALL SHOW
    STOP
  END IF
  NEWX = XPOS * 1.25 + 45.0
  NEWY = YPOS + 36.0
  NX = ROUND(NEWX)
  NY = ROUND(NEWY)
  IX = ROUND(SYSX)
  IY = ROUND(SYSY)
  IF((NX .NE. IX) .OR. (NY .NE. IY)) THEN DO
    IF(SYSC .NE. '0') THEN DO
      L = IABS(NX - IX)
      IF(IABS(NY - IY) .GT. L) THEN DO
        L = IABS(NY - IY)
      END IF
      DX = FLOAT(NX - IX)/FLOAT(L)
      DY = FLOAT(NY - IY)/FLOAT(L)
      X = IX + 0.5
      Y = IY + 0.5
      DO 1 K = 1,L,1
        SYSTAB(IFIX(Y),IFIX(X)) = SYSC
        X = X + DX
        Y = Y + DY
1     CONTINUE
      SYSTAB(NY,NX) = SYSC
    END IF
    SYSX = NEWX
    SYSY = NEWY
  ELSE DO
    IF(SYSC .NE. '0') THEN DO
      SYSTAB(IY,IX) = SYSC
    END IF
  END IF
  RETURN
  END
```

```
      SUBROUTINE WHERE(XPOS,YPOS,DIR)
        INTEGER XPOS,YPOS,DIR,SYSA,ROUND
        REAL SYSX,SYSY
        CHARACTER SYSC*1,SYSTAB*1(71,89)
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        XPOS = ROUND((SYSX-45.0)*0.8)
        YPOS = ROUND(SYSY-36.0)
        DIR = SYSA
        RETURN
        END

      SUBROUTINE SHOW
        INTEGER I,J,SYSA,M
        REAL SYSX,SYSY
        CHARACTER SYSTAB*1(71,89),SYSC*1
        COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
        PRINT 7
7       FORMAT('1')
        DO 6 I=1,71,1
          M = 72 - I
          PRINT 8,(SYSTAB(M,J),J=1,89,1)
8         FORMAT(' ',7X,89A1)
6       CONTINUE
        PRINT 7
        RETURN
        END

      INTEGER FUNCTION ROUND(X)
        REAL X
        IF(X .GT. 0.0) THEN DO
          ROUND = X + 0.5
        ELSE DO
          ROUND = X - 0.5
        END IF
        RETURN
        END
```

APPENDIX B

```
  PROCEDURE PIXEL(X,Y:INTEGER);
    VAR I: INTEGER;
    BEGIN
      IF SYSC <> NONE THEN
        BEGIN
          FOR I:= 1 TO 4 DO SYSTAB(.Y,X,I.) := ' ';
          CASE SYSC OF
            0: ;
            1: SYSTAB(.Y,X,1.) := '-';
            2: SYSTAB(.Y,X,1.) := '=';
            3: BEGIN SYSTAB(.Y,X,1.) := '+';
                     SYSTAB(.Y,X,2.) := '+' END;
            4: BEGIN SYSTAB(.Y,X,1.) := '*';
                     SYSTAB(.Y,X,2.) := 'X' END;
            5: BEGIN SYSTAB(.Y,X,1.) := 'X';
                     SYSTAB(.Y,X,2.) := 'X';
                     SYSTAB(.Y,X,3.) := '=' END;
            6: BEGIN SYSTAB(.Y,X,1.) := 'O';
                     SYSTAB(.Y,X,2.) := 'X';
                     SYSTAB(.Y,X,3.) := '*' END;
            7: BEGIN SYSTAB(.Y,X,1.) := '#';
                     SYSTAB(.Y,X,2.) := 'M';
                     SYSTAB(.Y,X,3.) := 'W';
                     SYSTAB(.Y,X,4.) := 'O' END
          END
        END
    END;
  PROCEDURE WHEREAMI(VAR XPOS,YPOS,DIRECTION:INTEGER);
    BEGIN
      XPOS := ROUND((SYSX - 60)*0.4);
      YPOS := ROUND(SYSY - 36);
      DIRECTION := SYSA
    END;
  PROCEDURE DISPLAY;
    VAR X,Y,P:INTEGER;
        C: CHAR;
    BEGIN
      FOR Y := 71 DOWNTO 1 DO
        BEGIN
          C := ' ';
          FOR P := 1 TO 4 DO
            BEGIN
              WRITE(C);
              FOR X := 1 TO 119 DO
                WRITE(SYSTAB(.Y,X,P.));
              WRITELN;
              C := '.'
            END
        END
    END;
  PROCEDURE CLEARSCREEN(C:INTEGER);
```

```
  VAR X,Y: INTEGER;
  BEGIN
    SYSC := C;
    FOR X := 1 TO 119 DO
      FOR Y := 1 TO 71 DO
        PIXEL(X,Y);
    SYSX := 60.0;
    SYSY := 36.0;
    SYSA := 0;
    SYSC := NONE
  END;
PROCEDURE PENCOLOR(COLOR:INTEGER);
  BEGIN
    SYSC := COLOR
  END;
PROCEDURE MOVE(DISTANCE:INTEGER);
  VAR IX,IY,NX,NY,K,L: INTEGER;
      RADS,NEWX,NEWY,X,Y,DX,DY:REAL;
  BEGIN
    IF DISTANCE <> 0 THEN
      BEGIN
        RADS := SYSA * 0.0174532925;
        NEWX := DISTANCE * 1.67 * COS(RADS) + SYSX;
        NEWY := DISTANCE * SIN(RADS) + SYSY;
        NX := ROUND(NEWX);
        NY := ROUND(NEWY);
        IF (NX>119) OR (NY>71) OR (NX<1) OR (NY<1) THEN
          BEGIN
            WHEREAMI(IX,IY,K);
            WRITELN(' ERROR IN MOVE');
            WRITELN(' PEN NOW AT (',IX,IY,') ANGLE',K);
            WRITELN(' PEN WOULD GO TO (',NX,NY,') ');
            DISPLAY;
            HALT
          END;
        IF SYSC <> NONE THEN
          BEGIN
            IX := ROUND(SYSX);
            IY := ROUND(SYSY);
            L := ABS(NX - IX);
            IF ABS(NY - IY) > L THEN L := ABS(NY - IY);
            DX := (NX - IX)/L;
            DY := (NY - IY)/L;
            X := IX + 0.5;
            Y := IY + 0.5;
            FOR K := 1 TO L DO
              BEGIN
                PIXEL(TRUNC(X),TRUNC(Y));
                X := X + DX;
                Y := Y + DY
              END;
            PIXEL(NX,NY)
          END;
        SYSX := NEWX;
        SYSY := NEWY
      END
    ELSE
      IF (SYSC <> NONE) THEN PIXEL(ROUND(SYSX),ROUND(SYSY))
```

```
    END;
  PROCEDURE TURN(ANGLE:INTEGER);
    BEGIN
      SYSA := (SYSA + ANGLE) MOD 360
    END;
  PROCEDURE MOVETO(XPOS,YPOS:INTEGER);
    VAR IX,IY,NX,NY,K,L:INTEGER;
        NEWX,NEWY,X,Y,DX,DY:REAL;
    BEGIN
      IF (ABS(XPOS) > 35) OR (ABS(YPOS) > 35) THEN
        BEGIN
          WHEREAMI(IX,IY,K);
          WRITELN(' ERROR IN MOVETO');
          WRITELN(' PEN NOW AT (',IX,IY,')');
          WRITELN(' PEN WOULD GO TO (',XPOS,YPOS,')');
          DISPLAY;
          HALT
        END;
      NEWX := XPOS * 1.67 + 60.0;
      NEWY := YPOS + 36.0;
      NX := ROUND(NEWX);
      NY := ROUND(NEWY);
      IX := ROUND(SYSX);
      IY := ROUND(SYSY);
      IF (NX <> IX) OR (NY <> IY) THEN
        BEGIN
          IF SYSC <> NONE THEN
            BEGIN
              L := ABS(NX - IX);
              IF ABS(NY - IY) > L THEN L := ABS(NY - IY);
              DX := (NX - IX)/L;
              DY := (NY - IY)/L;
              X := IX + 0.5;
              Y := IY + 0.5;
              FOR K := 1 TO L DO
                BEGIN
                  PIXEL(TRUNC(X),TRUNC(Y));
                  X := X + DX;
                  Y := Y + DY
                END;
              PIXEL(NX,NY)
            END;
          SYSX := NEWX;
          SYSY := NEWY
        END
      ELSE IF SYSC <> NONE THEN PIXEL(IX,IY)
    END;
  PROCEDURE TURNTO(ANGLE:INTEGER);
    BEGIN
      SYSA :=  ANGLE MOD 360
    END;
  PROCEDURE FILL(X,Y,C: INTEGER);
    PROCEDURE FILL1(X,Y:INTEGER);
      VAR I,J: INTEGER;
      FUNCTION ISCOLOR(X,Y,C:INTEGER): BOOLEAN;
        BEGIN
          ISCOLOR := FALSE;
          IF C <> NONE THEN
```

```
            CASE C OF
              0: ISCOLOR := SYSTAB(.Y,X,1.) = ' ';
              1: ISCOLOR := SYSTAB(.Y,X,1.) = '-';
              2: ISCOLOR := SYSTAB(.Y,X,1.) = '=';
              3: ISCOLOR := SYSTAB(.Y,X,1.) = '+';
              4: ISCOLOR := SYSTAB(.Y,X,1.) = '*';
              5: ISCOLOR := SYSTAB(.Y,X,1.) = 'X';
              6: ISCOLOR := SYSTAB(.Y,X,1.) = 'O';
              7: ISCOLOR := SYSTAB(.Y,X,1.) = '#'
            END
        END;
    BEGIN
      IF ISCOLOR(X,Y,C) THEN
        BEGIN
          PIXEL(X,Y);
          FILL1(X,Y+1);
          FILL1(X,Y-1);
          FILL1(X+1,Y);
          FILL1(X-1,Y)
        END
    END;
BEGIN
  FILL1(ROUND(1.67*X+60.0),Y+36)
END;
```

APPENDIX C

30	17	-120
0	0	-240
-15	26	-120
-35	-9	0
15	-9	-120
10	-18	-240
0	0	0
30	0	120
10	38	240
-15	-9	120
-20	0	0
0	0	-120
-15	-26	0
25	-26	120
0	17	0