**COMPUTER GRAPHICS USING C**

**LAB INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | OBJECTIVE | DATE | SIGNATURE |
| 1 | DDA Algorithm for the Line Generation  Sol.  #include<graphics.h>  #include<stdio.h>  #include<conio.h>  #include<math.h>  int main( )  {  float x,y,x1,y1,x2,y2,dx,dy,pixel;  int i,gd,gm;  printf("Enter the value of x1 : ");  scanf("%f",&x1);  printf("Enter the value of y1 : ");  scanf("%f",&y1);  printf("Enter the value of x2 : ");  scanf("%f",&x2);  printf("Enter the value of y2 : ");  scanf("%f",&y2);  detectgraph(&gd,&gm);  initgraph(&gd,&gm,"c:\\tc\\bgi");  dx=abs(x2-x1);  dy=abs(y2-y1);  if(dx>=dy)  pixel=dx;  else  pixel=dy;  dx=dx/pixel;  dy=dy/pixel;  x=x1;  y=y1;  i=1;  while(i<=pixel)  {  putpixel(x,y,1);  x=x+dx;  y=y+dy;  i=i+1;  delay(100);  }  getch();  closegraph();  } |  |  |
| 2 | Breshanham Algorithm for the line generation.  Sol. |  |  |
| 3 | Breshanham algorithm for the circle generation.  Solution. #include<conio.h>  #include<graphics.h>  void main()  {  int xc,yc,r;  int gd=DETECT,gm,d,x,y;  initgraph(&gd,&gm,"c:\\tc\\bgi");  printf("Enter center coordinates of circle: ");  scanf("%d %d",&xc,&yc);  printf("Enter radius of circle: ");  scanf("%d",&r);  x=0;  y=r;  p=3-2\*r;  while (x<y)  {  putpixel(xc+x, yc+y, RED);  putpixel(xc-x, yc+y, RED);  putpixel(xc+x, yc-y, GREEN);  putpixel(xc-x, yc-y, GREEN);  putpixel(xc+y, yc+x, WHITE);  putpixel(xc-y, yc+x, WHITE);  putpixel(xc+y, yc-x, RED);  putpixel(xc-y, yc-x, RED);;  x++;  if(p<0)  p=p+4\*x+6;  else  {  y--;  p=p+4\*(x-y)+10;  }  delay(50);  getch();  }  closegraph();  } |  |  |
| 4 | Program for the Mid-point circle algorithm.  Solution.  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  #include<graphics.h>  void main()  {  int p,x,y,xc,yc,r,gd=DETECT,gm;  printf("Enter the radius and center coordinates");  scanf("%d %d %d",&r,&xc,&yc);  initgraph(&gd,&gm,"c:\\tc\\bgi");  x=0;  y=r;  p=1-r;  while(x<y)  {  if(p<0)  {  x=x+1;  p=p+2\*x+1;  }  else  {  x=x+1;  y=y-1;  p=p-2\*(y-x)+10;  }  putpixel(xc+x,yc+y,RED);  putpixel(xc-y,yc-x,BLUE);  putpixel(xc+y,yc-x,WHITE);  putpixel(xc-y,yc+x,GREEN);  putpixel(xc+y,yc+x,WHITE);  putpixel(xc-x,yc-y,BLUE);  putpixel(xc+x,yc-y,GREEN);  putpixel(xc-x,yc+y,BLUE);  }  getch();  } |  |  |
| 5 | 2D-Translation of a polygon using graphics in c.  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  #include<graphics.h>  #include<math.h>  void main()  {  int gd,gm;  int  x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;  int sx,sy,xt,yt,r;  float t;  detectgraph(&gd,&gm);  initgraph(&gd,&gm,"c://tc//bgi");  printf("\t Program for basic transactions");  printf("\n\t Enter the points of triangle");  scanf("%d%d%d%d%d  %d",&x1,&y1,&x2,&y2,&x3,&y3);  line(x1,y1,x2,y2);  line(x2,y2,x3,y3);  line(x3,y3,x1,y1);  printf("\n Enter the translation factor");  scanf("%d  %d",&xt,&yt);  nx1=x1+xt;  ny1=y1+yt;  nx2=x2+xt;  ny2=y2+yt;  nx3=x3+xt;  ny3=y3+yt;  line  (nx1,ny1,nx2,ny2);  line  (nx2,ny2,nx3,ny3);  line  (nx3,ny3,nx1,ny1);  }  Closegraph();  }  getch();  } |  |  |
| 6 | 2D Rotation of the Polygon.  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  #include<graphics.h>  #include<math.h>  void main()  {  int gd,gm;  int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;  int sx,sy,xt,yt,r;  float t;  detectgraph(&gd,&gm);  initgraph(&gd,&gm,"c://tc//bgi");  printf("\t Program for basic transactions");  printf("\n\t Enter the points of triangle");  scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);  line(x1,y1,x2,y2);  line(x2,y2,x3,y3);  line(x3,y3,x1,y1);  printf("\n Enter the angle of rotation");  scanf("%d",&r);  t=3.14\*r/180;  nx1=abs(x1\*cos(t)-y1\*sin(t));  ny1=abs(x1\*sin(t)+y1\*cos(t));  nx2=abs(x2\*cos(t)-y2\*sin(t));  ny2=abs(x2\*sin(t)+y2\*cos(t));  nx3=abs(x3\*cos(t)-y3\*sin(t));  ny3=abs(x3\*sin(t)+y3\*cos(t));  line(nx1,ny1,nx2,ny2);  line(nx2,ny2,nx3,ny3);  line(nx3,ny3,nx1,ny1);  }  closegraph();  getch(); } |  |  |
| 7 | 2D scaling for the polygon.  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  #include<graphics.h>  #include<math.h>  void main()  {  int gd,gm;  int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;  int sx,sy,xt,yt,r;  float t;  detectgraph(&gd,&gm);  initgraph(&gd,&gm,"c://tc//bgi");  printf("\t Program for basic transactions");  printf("\n\t Enter the points of triangle");  scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);  line(x1,y1,x2,y2);  line(x2,y2,x3,y3);  line(x3,y3,x1,y1);  printf("\n Enter the scalling factor");  scanf("%d%d",&sx,&sy);  nx1=x1\*sx;  ny1=y2\*sy;  nx2=x2\*sx;  ny2=y2\*sy;  nx3=x3\*sx;  ny3=y3\*sy;  line(nx1,ny1,nx2,ny2);  line(nx2,ny2,nx3,ny3);  line(nx3,ny3,nx1,ny1);  }  closegraph();  getch();  } |  |  |
| 8 | Program for the Cohen Sutherland Window Line Clipping using c.  Sol.  #include <stdio.h>  #include <stdlib.h>  #include <graphics.h>  #define MAX 20    enum { TOP = 1, BOTTOM = 2, RIGHT = 4, LEFT = 8 };    enum { FALSE, TRUE };  typedef unsigned int outcode;    outcode compute\_outcode(int x, int y,  int xmin, int ymin, int xmax, int ymax)  {  outcode oc = 0;    if (y > ymax)  oc |= TOP;  else if (y < ymin)  oc |= BOTTOM;      if (x > xmax)  oc |= RIGHT;  else if (x < xmin)  oc |= LEFT;  return oc;  }    void cohen\_sutherland (double x1, double y1, double x2, double y2, double xmin, double ymin, double xmax, double ymax)  {  int accept;  int done;  outcode outcode1, outcode2;    accept = FALSE;  done = FALSE;    outcode1 = compute\_outcode (x1, y1, xmin, ymin, xmax, ymax);  outcode2 = compute\_outcode (x2, y2, xmin, ymin, xmax, ymax);  do  {  if (outcode1 == 0 && outcode2 == 0)  {  accept = TRUE;  done = TRUE;  }  else if (outcode1 & outcode2)  {  done = TRUE;  }  else  {  double x, y;  int outcode\_ex = outcode1 ? outcode1 : outcode2;  if (outcode\_ex & TOP)  {  x = x1 + (x2 - x1) \* (ymax - y1) / (y2 - y1);  y = ymax;  }    else if (outcode\_ex & BOTTOM)  {  x = x1 + (x2 - x1) \* (ymin - y1) / (y2 - y1);  y = ymin;  }  else if (outcode\_ex & RIGHT)  {  y = y1 + (y2 - y1) \* (xmax - x1) / (x2 - x1);  x = xmax;  }  else  {  y = y1 + (y2 - y1) \* (xmin - x1) / (x2 - x1);  x = xmin;  }  if (outcode\_ex == outcode1)  {  x1 = x;  y1 = y;  outcode1 = compute\_outcode (x1, y1, xmin, ymin, xmax, ymax);  }  else  {  x2 = x;  y2 = y;  outcode2 = compute\_outcode (x2, y2, xmin, ymin, xmax, ymax);  }  }  } while (done == FALSE);    if (accept == TRUE)  line (x1, y1, x2, y2);  }        void main()  {  int n;  int i, j;  int ln[MAX][4];  int clip[4];  int gd = DETECT, gm;    printf ("Enter the number of lines to be clipped");  scanf ("%d", &n);    printf ("Enter the x- and y-coordinates of the line-endpoints:\n");  for (i=0; i<n; i++)  for (j=0; j<4; j++)  scanf ("%d", &ln[i][j]);    printf ("Enter the x- and y-coordinates of the left-top and right-bottom corners of the clip window:\n");  for (i=0; i<4; i++)  scanf ("%d", &clip[i]);    initgraph (&gd, &gm, "..//bgi");  rectangle (clip[0], clip[1], clip[2], clip[3]);  for (i=0; i<n; i++)  line (ln[i][0], ln[i][1], ln[i][2], ln[i][3]);  getch();  cleardevice();  rectangle (clip[0], clip[1], clip[2], clip[3]);  for (i=0; i<n; i++)  {  cohen\_sutherland (ln[i][0], ln[i][1], ln[i][2], ln[i][3],  clip[0], clip[1], clip[2], clip[3]);  getch();  }  closegraph();  } |  |  |
| 9 | Program for the Sutherland-Hodgeman for the window polygon clipping using c. |  |  |
| 10 | Making Different Shapes like Hut Kites Using Graphics using C.  >Programs logic is explained and done kindly choose the shape and write program yourself as it is a simple one |  |  |
| 11. | Program in graphics for the Polygon Shearing Using c. |  |  |