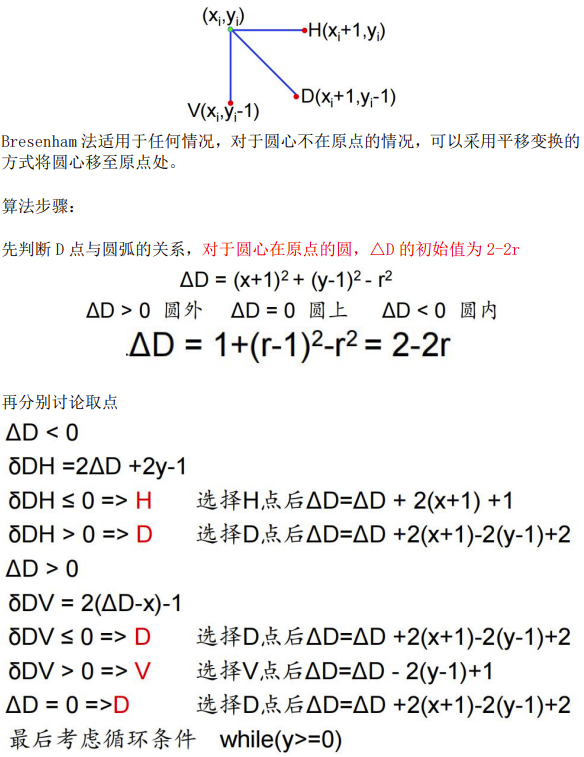
**模块1-2 扫描转换算法**

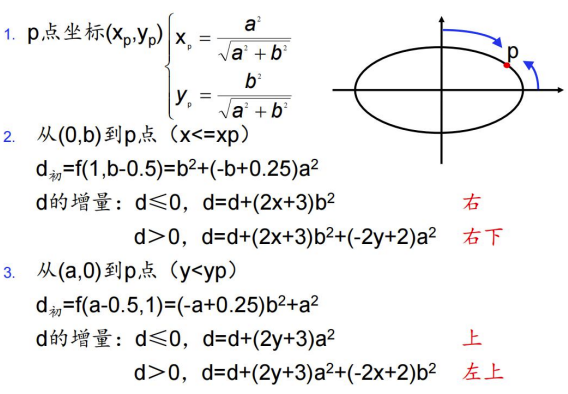
开篇注释：本实验报告的**实验结果贴图**部分在【三：实验内容】中，**实验代码**部分在【四：程序说明】中。若只需要查看上述两部分内容，可在本文档中分别跳转到**第（4）页和第（10）页**，即为各部分的起始页码，感谢查阅。

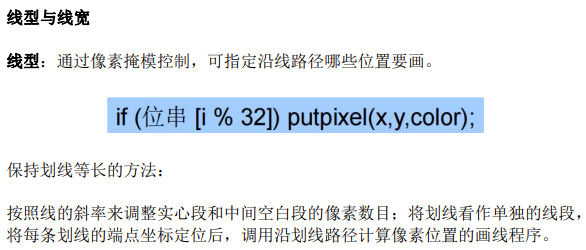
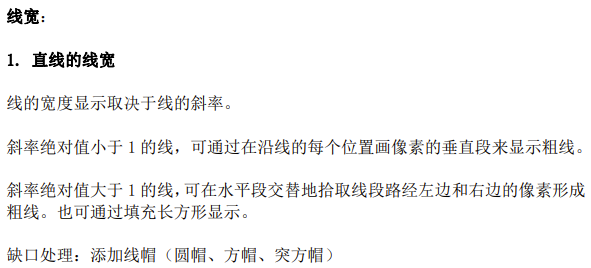
**一 实验目的**

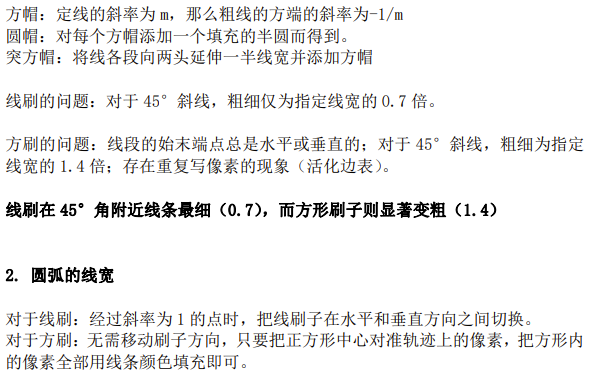
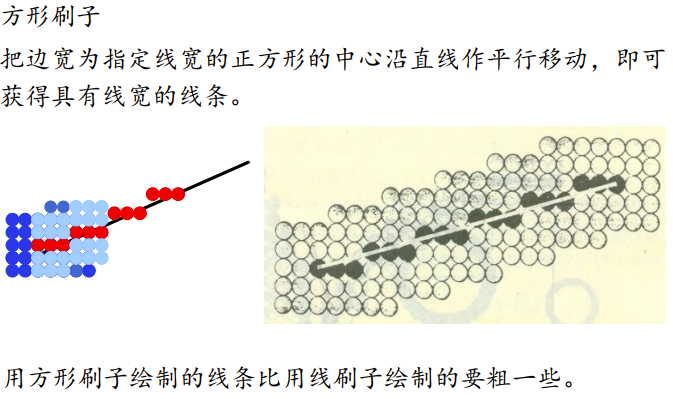
1. 编写弧线的光栅扫描转换算法，并对线宽与线形的算法加以探讨
2. 熟悉圆和椭圆画线的算法

**二 实验算法理论分析**

**Bresenham法（1/4圆）**：

**椭圆扫描转换——两头逼近法：**

**处理线宽问题：**

方形刷子宽度存在的问题：

1：线段的始末端点总是水平或垂直的。

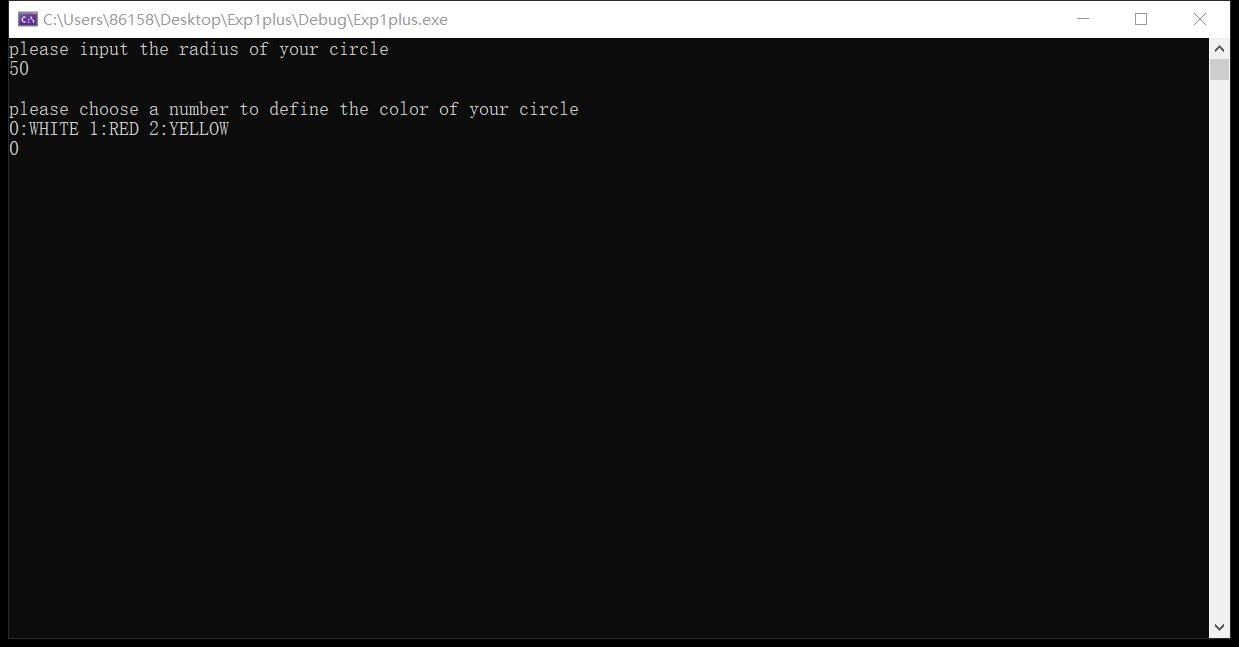
2：水平或垂直方形刷子与线条垂直，其粗细与线宽相等，而对于45°斜线，粗细为指定线宽的1.4倍。

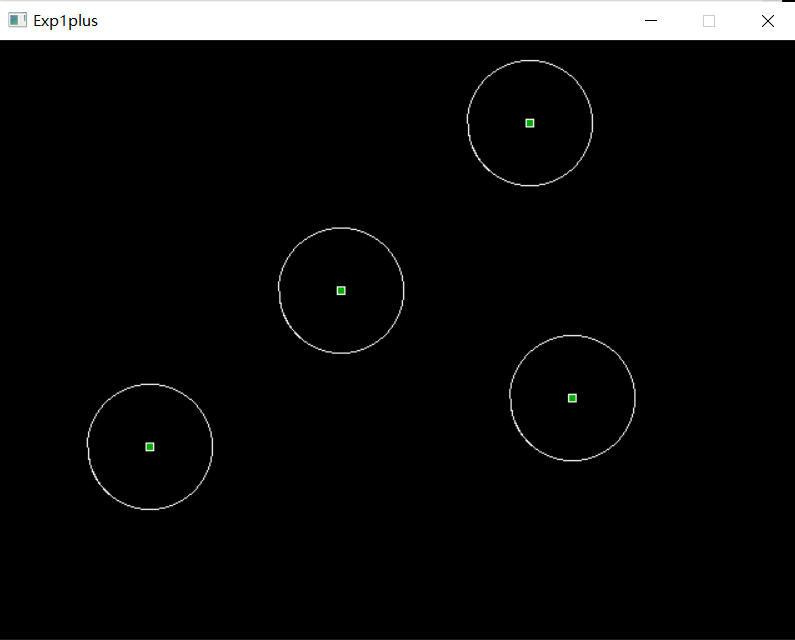
3：重复写象素。为每条扫描线建一个活化边表，存放该扫描线与线条的相交区间左右端点位置。

**三 实验内容**

4：用四分法画半径为r的一个完整的圆，用鼠标选择圆心位置。

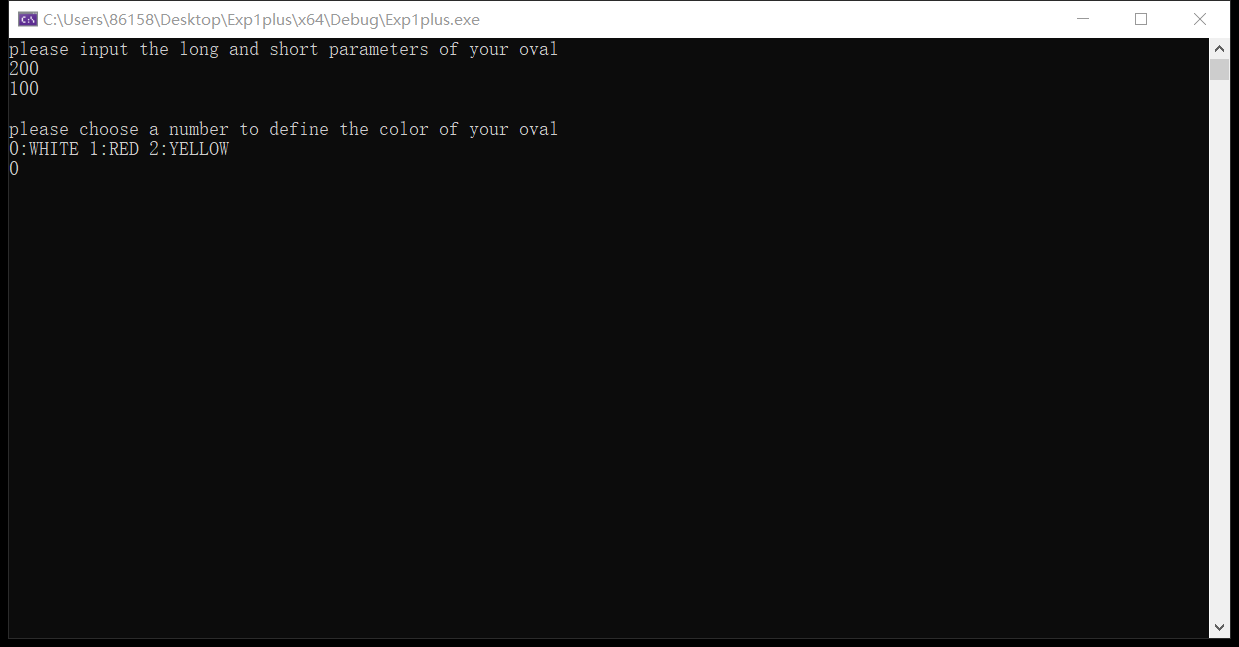
实验结果如下图所示：

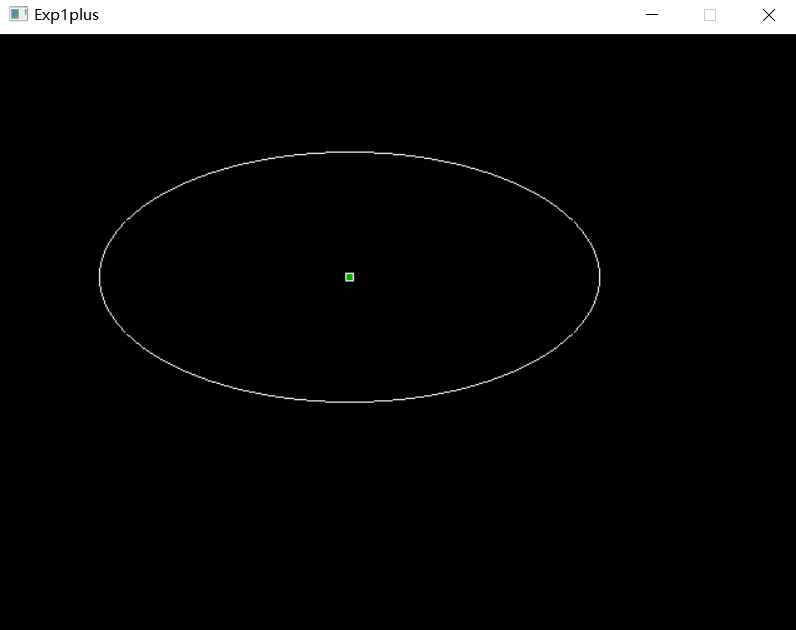
第一步：输入所画圆的半径和勾勒的颜色（此处以50像素为圆的半径，白色为勾勒的颜色为例）。

第二步：通过鼠标左键点击画布，编译后的文件自动以当前的鼠标坐标为圆心，画出目标半径和颜色的圆形图案。

5：画一个半长轴为a和半短轴为b的完整椭圆，用鼠标选择圆心位置。

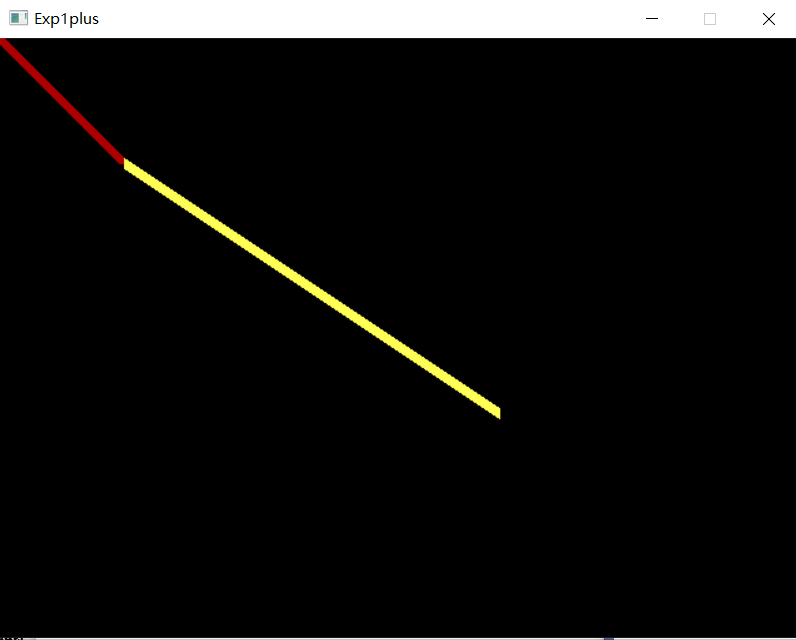
实验结果如下图所示：

第一步：输入所画椭圆的半长轴、半短轴和勾勒的颜色（此处以200像素和100像素为椭圆的半长轴和半短轴，白色为勾勒的颜色为例）。

第二步：通过鼠标左键点击画布，编译后的文件自动以当前的鼠标坐标为圆心，画出目标半长轴、半短轴和颜色的椭圆形图案。

6：设计具有宽度的画线算法，并处理线条连接处出现缺口的问题。

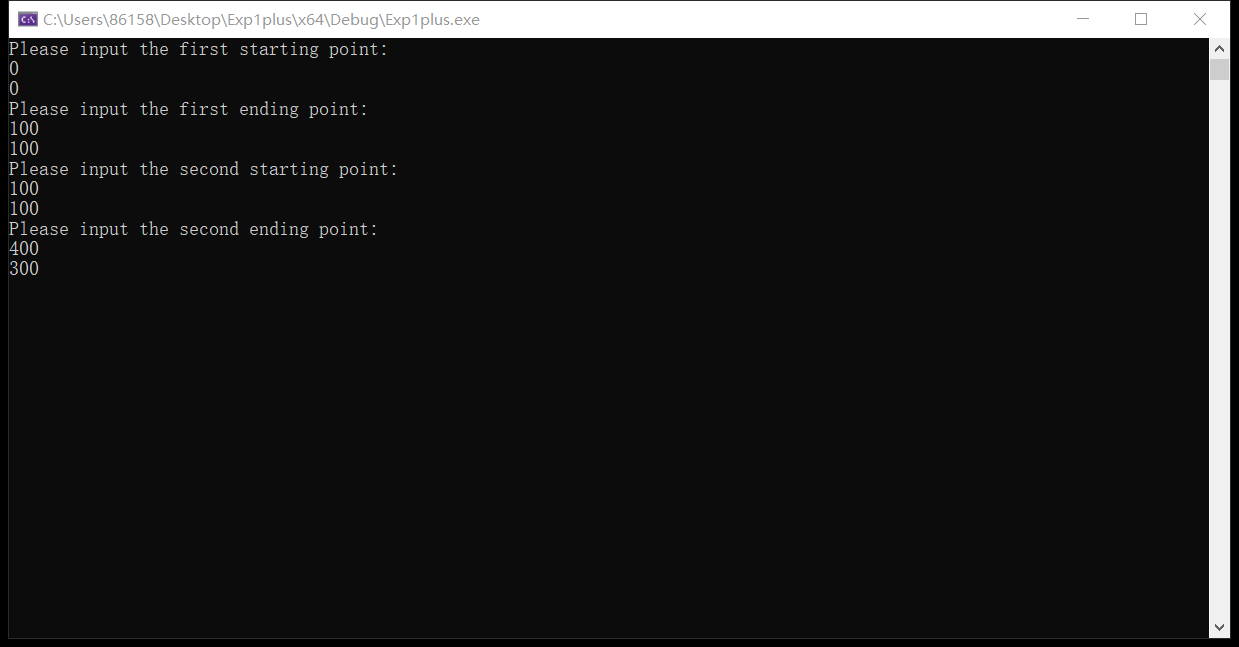
实验结果如下图所示：

第一步：编写代码查看未处理缺口的图像，以便后续进行对比。【此处以第一条直线起点（0，0）、终点（100，100），第二条直线起点（100，100）、终点（400，300）为例】

放大后可观察到连接处有一定程度的缺口。

第二步：通过改进bresenham画线算法，弥补缺口。

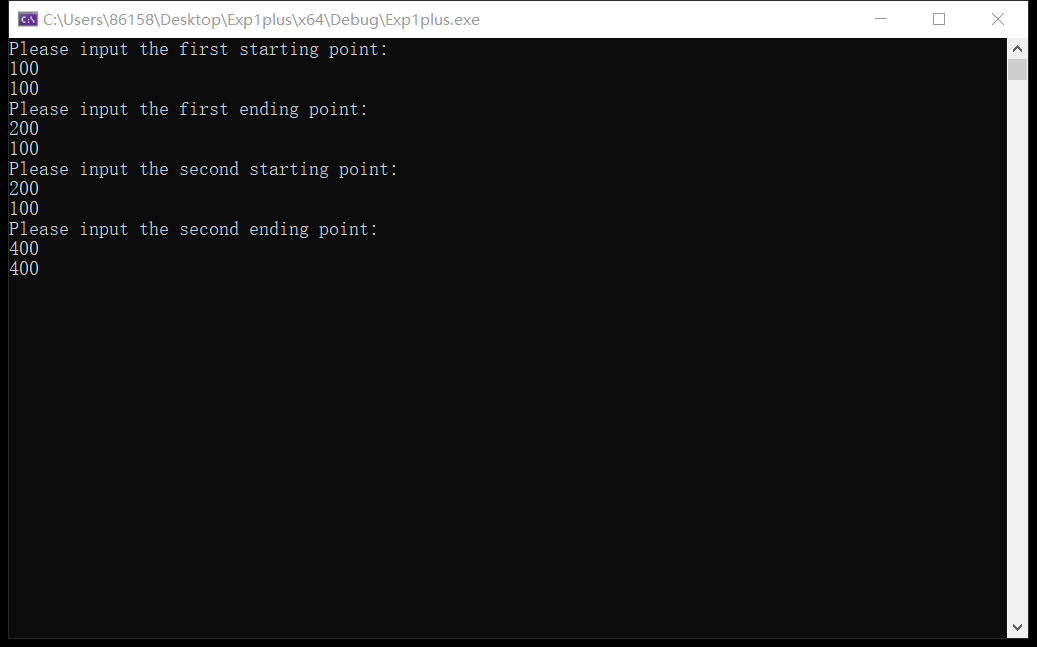
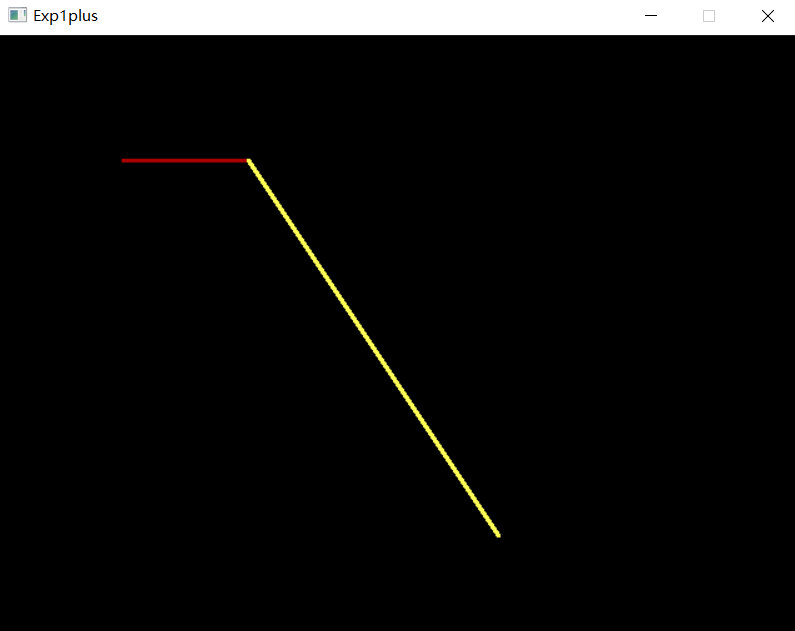
方法一：通过硬修改直线端点处的宽度绘画法连接两条直线。【此处以第一条直线起点（0，0）、终点（100，100），第二条直线起点（100，100）、终点（400，300）为例】

输入坐标数据内容：

程序生成图像：

放大后可观察到连接处较平稳：但是缺点明显，比如斜率绝对值过度小于1的直线宽度从视觉上来说不够。

方法二：通过改进bresenham算法，将putpixel()函数改进为minepixel()函数，通过方形刷子的方法进行缺口填缺。【此处以第一条直线起点（100，100）、终点（200，100），第二条直线起点（200，100）、终点（400，400）为例】

输入坐标数据内容：

程序生成图像：

放大后可观察到连接处较平稳：

**四 程序说明**

Project中程序的调用：

将当前cpp文件的属性——常规——从生成中排除中选择否，其他文件选择是，即可运行当前的cpp文件

|  |
| --- |
| 4题 |
| //////////////////////////////////////////////////////  // 程序名称：画圆算法  // 功 能：用四分法画半径为r的一个完整的圆，用鼠标选择圆心位置  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-3-10  #include <graphics.h>  #include <conio.h>  #include <iostream>  using namespace std;  void Circle(int myx, int myy, int r, int color) {  int x, y, delta, delta1, delta2, direction;  //画第一象限的圆弧  x = 0;  y = r;  delta = 2 \* (1 - r);  while(y >= 0) {  putpixel(myx + x, myy + y, color);  if (delta < 0) {  delta1 = 2 \* (delta + y) - 1;  if (delta1 <= 0) {  direction = 1;  }  else {  direction = 2;  }  }  else if (delta > 0) {  delta2 = 2 \* (delta - x) - 1;  if (delta2 <= 0) {  direction = 2;  }  else {  direction = 3;  }  }  else {  direction = 2;  }  switch (direction) {  case 1:  x++;  delta += 2 \* x + 1;  break;  case 2:  x++;  y--;  delta += 2 \* (x - y + 1);  break;  case 3:  y--;  delta += (-2 \* y + 1);  break;  }  }  //画第二象限的圆弧  x = 0;  y = r;  delta = 2 \* (1 - r);  while (y >= 0) {  putpixel(myx + x, myy + y, color);  if (delta < 0) {  delta1 = 2 \* (delta + y) - 1;  if (delta1 <= 0) {  direction = 1;  }  else {  direction = 2;  }  }  else if (delta > 0) {  delta2 = 2 \* (delta - x) - 1;  if (delta2 <= 0) {  direction = 2;  }  else {  direction = 3;  }  }  else {  direction = 2;  }  switch (direction) {  case 1:  x--;  delta += 2 \* (-x) + 1;  break;  case 2:  x--;  y--;  delta += 2 \* (-x - y + 1);  break;  case 3:  y--;  delta += (-2 \* y + 1);  break;  }  }  //画第三象限的圆弧  x = 0;  y = -r;  delta = 2 \* (1 - r);  while (y <= 0) {  putpixel(myx + x, myy + y, color);  if (delta < 0) {  delta1 = 2 \* (delta - y) - 1;  if (delta1 <= 0) {  direction = 1;  }  else {  direction = 2;  }  }  else if (delta > 0) {  delta2 = 2 \* (delta + x) - 1;  if (delta2 <= 0) {  direction = 2;  }  else {  direction = 3;  }  }  else {  direction = 2;  }  switch (direction) {  case 1:  x--;  delta += 2 \* (-x) + 1;  break;  case 2:  x--;  y++;  delta += 2 \* (-x + y + 1);  break;  case 3:  y++;  delta += (2 \* y + 1);  break;  }  }  //画第四象限的圆弧  x = 0;  y = -r;  delta = 2 \* (1 - r);  while (y <= 0) {  putpixel(myx + x, myy + y, color);  if (delta < 0) {  delta1 = 2 \* (delta - y) - 1;  if (delta1 <= 0) {  direction = 1;  }  else {  direction = 2;  }  }  else if (delta > 0) {  delta2 = 2 \* (delta - x) - 1;  if (delta2 <= 0) {  direction = 2;  }  else {  direction = 3;  }  }  else {  direction = 2;  }  switch (direction) {  case 1:  x++;  delta += 2 \* x + 1;  break;  case 2:  x++;  y++;  delta += 2 \* (x + y + 1);  break;  case 3:  y++;  delta += (2 \* y + 1);  break;  }  }  }  int main() {  //用户定义圆的相关参数  int r,colornum;  cout << "please input the radius of your circle" << endl;  cin >> r;  cout << endl;  cout << "please choose a number to define the color of your circle" << endl;  cout << "0:WHITE 1:RED 2:YELLOW" << endl;  cin >> colornum;  int colorarray[3] = { WHITE,RED,YELLOW };  //图形界面  initgraph(640, 480);  ExMessage m;  int x0, y0, x1, y1;  while (true) {  m = getmessage(EX\_MOUSE | EX\_KEY);  switch (m.message) {  case WM\_LBUTTONDOWN:  x0 = m.x;  y0 = m.y;  setfillcolor(GREEN);  fillrectangle(m.x - 3, m.y - 3, m.x + 3, m.y + 3);  Circle(m.x, m.y, r, colorarray[colornum]);  case WM\_KEYDOWN:  if (m.vkcode == VK\_ESCAPE)  return 0; // 按 ESC 键退出程序  }  }  closegraph();  return 0;  } |
| 5题 |
| //////////////////////////////////////////////////////  // 程序名称：画椭圆算法  // 功 能：画一个半长轴为a和半短轴为b的完整椭圆，用鼠标选择圆心位置  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-3-10  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <cmath>  using namespace std;  void Oval(int myx, int myy, int a, int b, int color) {  double xp, yp, d;  int x, y;  //第一象限  xp = (double)a \* a / sqrt(a \* a + b \* b);  yp = (double)b \* b / sqrt(a \* a + b \* b);  //(0,b)=>p  x = 0;  y = b;  d = b \* b + (-b + 0.25) \* a \* a;  while (x <= xp) {  putpixel(x + myx, y + myy, color);  if (d <= 0) {  d = d + (2 \* x + 3) \* b \* b;  x++;  }  else {  d = d + (x \* 2 + 3) \* b \* b + (-2 \* y + 2) \* a \* a;  x++;  y--;  }  }  //p->(a,0)  x = a;  y = 0;  d = (-a + 0.25) \* b \* b + a \* a;  while (y < yp) {  putpixel(myx + x, myy + y, color);  if (d <= 0) {  d += (2 \* y + 3) \* a \* a;  y++;  }  else {  d += (2 \* y + 3) \* a \* a + (-2 \* x + 2) \* b \* b;  x--;  y++;  }  }  //第二象限  xp = (double)-a \* a / sqrt(a \* a + b \* b);  yp = (double)b \* b / sqrt(a \* a + b \* b);  //(0,b)=>p  x = 0;  y = b;  d = b \* b + (-b + 0.25) \* a \* a;  while (x >= xp) {  putpixel(x + myx, y + myy, color);  if (d <= 0) {  d = d + (-2 \* x + 3) \* b \* b;  x--;  }  else {  d = d + (-x \* 2 + 3) \* b \* b + (-2 \* y + 2) \* a \* a;  x--;  y--;  }  }  //p->(a,0)  x = -a;  y = 0;  d = (-a + 0.25) \* b \* b + a \* a;  while (y < yp) {  putpixel(myx + x, myy + y, color);  if (d <= 0) {  d += (2 \* y + 3) \* a \* a;  y++;  }  else {  d += (2 \* y + 3) \* a \* a + (2 \* x + 2) \* b \* b;  x++;  y++;  }  }  //第三象限  xp = (double)-a \* a / sqrt(a \* a + b \* b);  yp = (double)-b \* b / sqrt(a \* a + b \* b);  //(0,-b)=>p  x = 0;  y = -b;  d = b \* b + (-b + 0.25) \* a \* a;  while (x >= xp) {  putpixel(x + myx, y + myy, color);  if (d <= 0) {  d = d + (-2 \* x + 3) \* b \* b;  x--;  }  else {  d = d + (-x \* 2 + 3) \* b \* b + (2 \* y + 2) \* a \* a;  x--;  y++;  }  }  //p->(-a,0)  x = -a;  y = 0;  d = (-a + 0.25) \* b \* b + a \* a;  while (y >= yp) {  putpixel(myx + x, myy + y, color);  if (d <= 0) {  d += (-2 \* y + 3) \* a \* a;  y--;  }  else {  d += (-2 \* y + 3) \* a \* a + (2 \* x + 2) \* b \* b;  x++;  y--;  }  }  //第四象限  xp = (double)a \* a / sqrt(a \* a + b \* b);  yp = (double)-b \* b / sqrt(a \* a + b \* b);  //(0,-b)=>p  x = 0;  y = -b;  d = b \* b + (-b + 0.25) \* a \* a;  while (x <= xp) {  putpixel(x + myx, y + myy, color);  if (d <= 0) {  d = d + (2 \* x + 3) \* b \* b;  x++;  }  else {  d = d + (x \* 2 + 3) \* b \* b + (2 \* y + 2) \* a \* a;  x++;  y++;  }  }  //p->(a,0)  x = a;  y = 0;  d = (-a + 0.25) \* b \* b + a \* a;  while (y >= yp) {  putpixel(myx + x, myy + y, color);  if (d <= 0) {  d += (-2 \* y + 3) \* a \* a;  y--;  }  else {  d += (-2 \* y + 3) \* a \* a + (-2 \* x + 2) \* b \* b;  x--;  y--;  }  }  }  int main() {  //用户定义椭圆的相关参数  int a, b, colornum;  cout << "please input the long and short parameters of your oval" << endl;  cin >> a >> b;  cout << endl;  cout << "please choose a number to define the color of your oval" << endl;  cout << "0:WHITE 1:RED 2:YELLOW" << endl;  cin >> colornum;  int colorarray[3] = { WHITE,RED,YELLOW };  //图形界面  initgraph(640, 480);  ExMessage m;  while (true) {  m = getmessage(EX\_MOUSE | EX\_KEY);  switch (m.message) {  case WM\_LBUTTONDOWN:  setfillcolor(GREEN);  fillrectangle(m.x - 3, m.y - 3, m.x + 3, m.y + 3);  Oval(m.x, m.y, a, b, colorarray[colornum]);  case WM\_KEYDOWN:  if (m.vkcode == VK\_ESCAPE)  return 0; // 按 ESC 键退出程序  }  }  closegraph();  return 0;  } |
| 6题 方法1 |
| //////////////////////////////////////////////////////  // 程序名称：有宽度线段的连接处处理  // 功 能：设计具有宽度的画线算法，并处理线条连接处出现缺口的问题  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-3-10  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <cmath>  using namespace std;  void bresenham(int x0, int y0, int x1, int y1, int color) {  int dx = abs(x1 - x0), sx = x0 < x1 ? 1 : -1;  int dy = abs(y1 - y0), sy = y0 < y1 ? 1 : -1;  int erro = (dx > dy ? dx : -dy) / 2;  while (putpixel(x0, y0, color), x0 != x1 || y0 != y1) {  int e2 = erro;  if (e2 > -dx) {  erro -= dy;  x0 += sx;  }  if (e2 < dy) {  erro += dx;  y0 += sy;  }  }  }  int main() {  //用户定义相关参数  int x0, y0, x1, y1, width, x2, y2, x3, y3;  cout << "Please input the first starting point:" << endl;  cin >> x0 >> y0;  cout << "Please input the first ending point:" << endl;  cin >> x1 >> y1;  cout << "Please input the second starting point:" << endl;  cin >> x2 >> y2;  cout << "Please input the second ending point:" << endl;  cin >> x3 >> y3;  //cout << "Please input the half width of your lines:" << endl;  //cin >> width;//输入半宽度  int mycolor[2] = { RED, YELLOW };  /\*  double flag1 = (double)(y1 - y0) / (x1 - x0);  double flag2 = (double)(y3 - y2) / (x3 - x2);  //flag1>1 or flag1<-1 斜率大于1 采用横向补点  //flag1<1 and flag1>-1 斜率小于1 采用纵向补点  //flag2同理可得  \*/  //图形界面  initgraph(640, 480);  //线性加粗  bresenham(x0, y0, x1, y1, mycolor[0]);  //宽度确定为10像素  for (int i = 0; i < 5; i++) {  bresenham(x0 + i, y0, x1 + i, y1, mycolor[0]);  bresenham(x0 - i, y0, x1 - i, y1, mycolor[0]);  }  for (int i = 0; i < 5; i++) {  bresenham(x2 + i, y2, x3 + i, y3, mycolor[1]);  bresenham(x2 - i, y2, x3 - i, y3, mycolor[1]);  }  //补充缺口  \_getch();  closegraph();  return 0;  } |
| 6题 方法2 |
| //////////////////////////////////////////////////////  // 程序名称：有宽度线段的连接处处理  // 功 能：设计具有宽度的画线算法，并处理线条连接处出现缺口的问题  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-3-10  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <cmath>  using namespace std;  void mineputpixel(int x, int y, int color) {  putpixel(x, y, color);  putpixel(x-1, y, color);  putpixel(x+1, y, color);  putpixel(x-1, y+1, color);  putpixel(x, y+1, color);  putpixel(x+1, y+1, color);  putpixel(x, y-1, color);  putpixel(x+1, y-1, color);  putpixel(x-1, y-1, color);  }  void bresenham(int x0, int y0, int x1, int y1, int color) {  int dx = abs(x1 - x0), sx = x0 < x1 ? 1 : -1;  int dy = abs(y1 - y0), sy = y0 < y1 ? 1 : -1;  int erro = (dx > dy ? dx : -dy) / 2;  while (mineputpixel(x0, y0, color), x0 != x1 || y0 != y1) {  int e2 = erro;  if (e2 > -dx) {  erro -= dy;  x0 += sx;  }  if (e2 < dy) {  erro += dx;  y0 += sy;  }  }  }  int main() {  //用户定义相关参数  int x0, y0, x1, y1, x2, y2, x3, y3;  cout << "Please input the first starting point:" << endl;  cin >> x0 >> y0;  cout << "Please input the first ending point:" << endl;  cin >> x1 >> y1;  cout << "Please input the second starting point:" << endl;  cin >> x2 >> y2;  cout << "Please input the second ending point:" << endl;  cin >> x3 >> y3;  int mycolor[2] = { RED, YELLOW };  //图形界面  initgraph(640, 480);  bresenham(x0, y0, x1, y1, mycolor[0]);  bresenham(x2, y2, x3, y3, mycolor[1]);  \_getch();  closegraph();  return 0;  } |