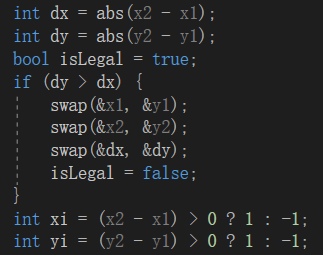
作业3

# Basic

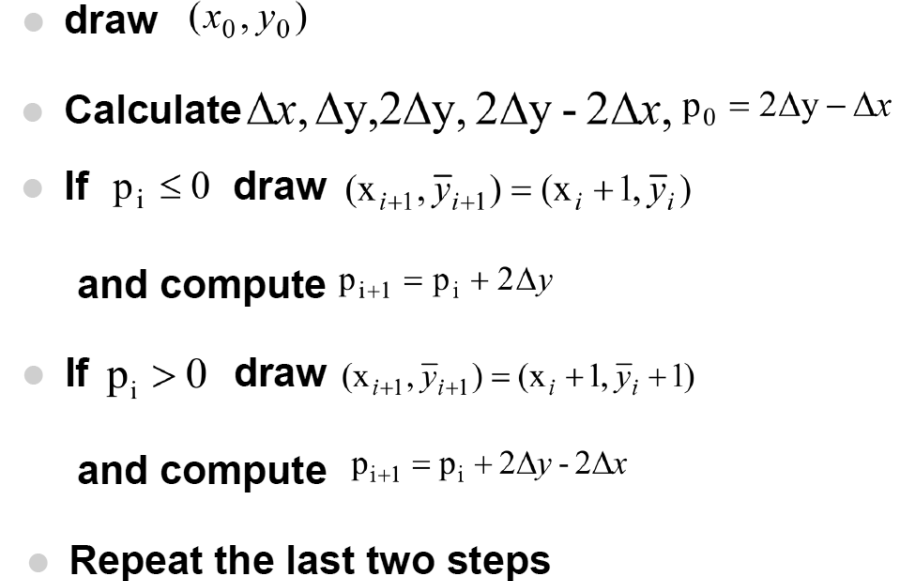
## 使用Bresenham算法(只使用integer arithmetic)画一个三角形边框：input为三个2D点；output三条直线（要求图元只能用 GL\_POINTS ，不能使用其他，比如 GL\_LINES 等）。

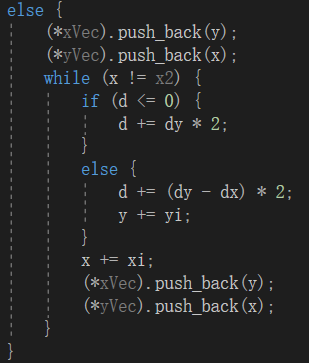
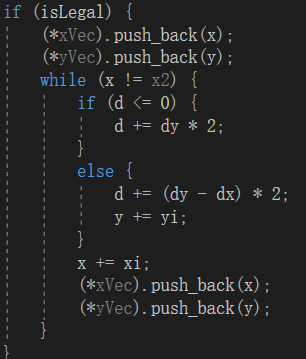
### Bresenham算法

由于Bresenham算法只支持斜率为0-1的直线，所以当abs(y2-y1)>abs(x2-x1)时，将x与y互换后再进行计算。而当x2小于x1或y2小于y1时，对直线的方向进行判定选择。



而Bresenham算法的主要内容则是如下图一所示。下图二和三的代码则是参照下图一进行实现。区别在于之前是否有进行x和y的交换。

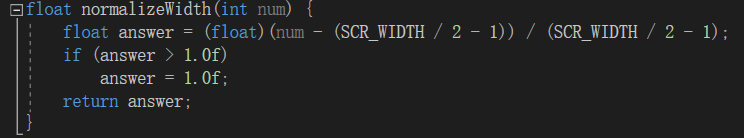


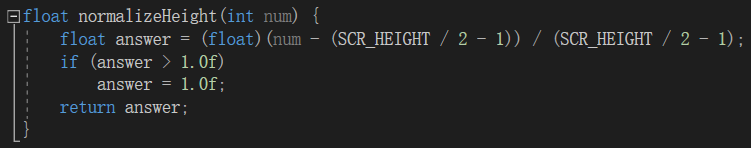


### 其余主要代码

#### normalizeWidth && normalizeHeight

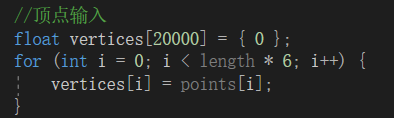
将输入的数值进行标准化，转换为-1.0f~1.0f。

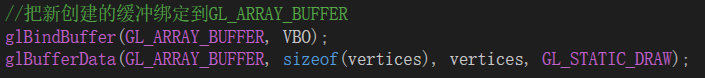


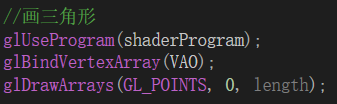


#### 顶点数组

由于c++传递的数组会转换成指针类型，所以在show()函数内声明了一个数组储存main函数传递的数据并进行使用。最后使用GL\_POINTS画点

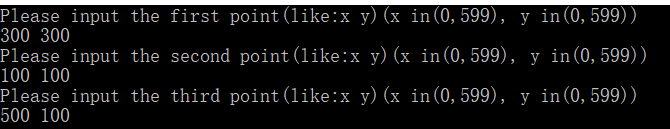


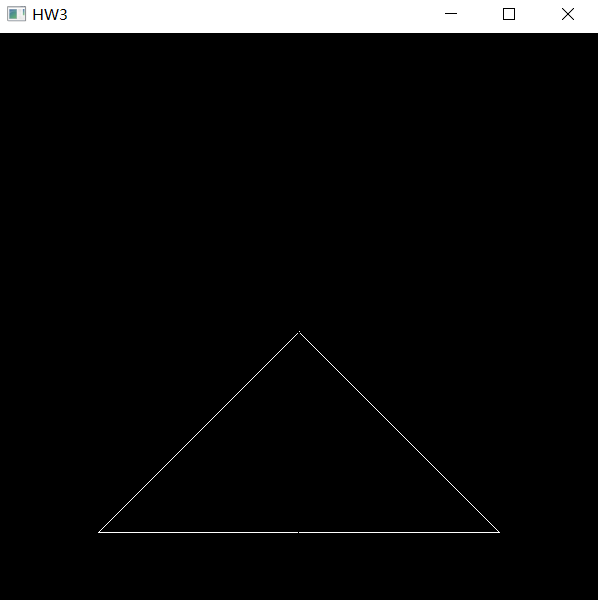




### 结果

程序运行一开始需要输入三个点的坐标。输入坐标后即可得到结果。



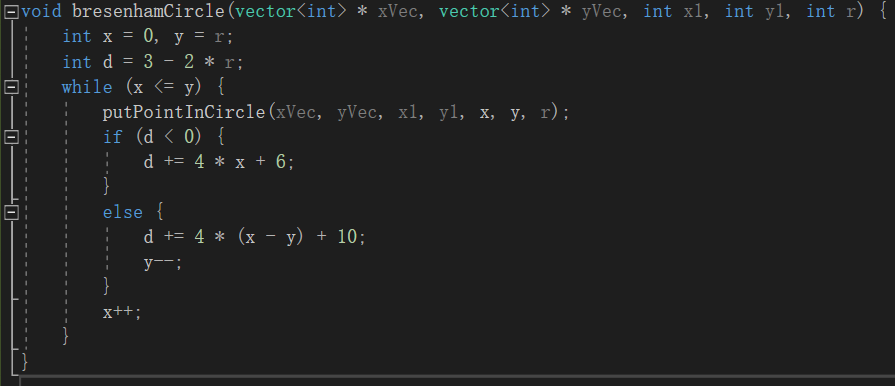


## 2. 使用Bresenham算法(只使用integer arithmetic)画一个圆：input为一个2D点(圆心)、一个integer半径；output为一个圆。

本题代码与上题类似，主要变换为修改了Bresenham算法进行画圆。

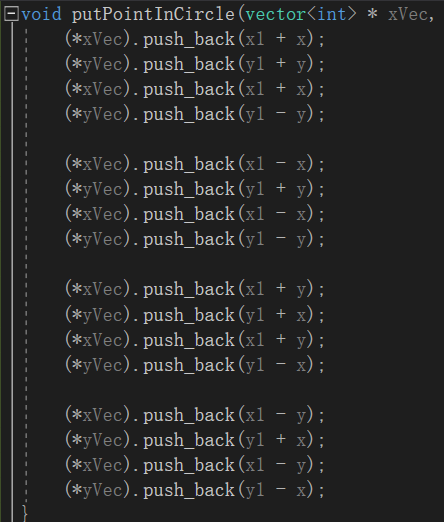
### Bresenham算法

在画圆的算法中，可以利用八对称性，只用计算八分之一圆弧的坐标即可。



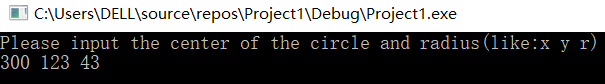
### putPointInCircle

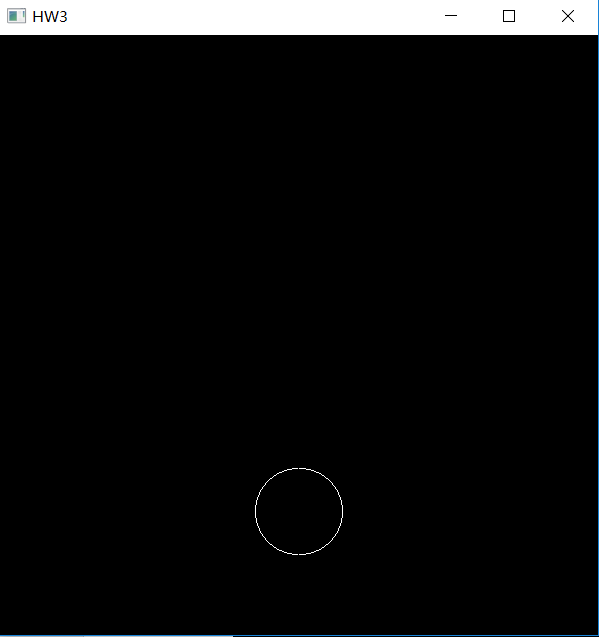
因为之前的Bresenham算法假设的圆心是(0,0)所以在这里需要按照给定圆心进行偏移，并将所有对称点按顺序按对放入向量保存。



### 结果

程序运行一开始需要输入圆心的坐标和半径。输入坐标后即可得到结果。



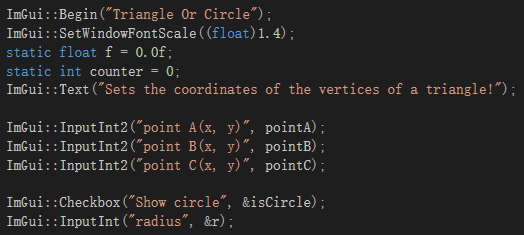


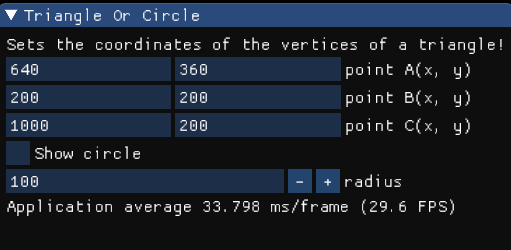
## 3. 在GUI在添加菜单栏，可以选择是三角形边框还是圆，以及能调整圆的大小(圆心固定即可)

本题只需要将前面两题的代码与GUI结合即可。这里采用的是IMGUI。

### GUI界面

这里在GUI中可以输入三角形三个点的坐标，当选则Show Circle时则可以展示圆，可以通过改变radius值，改变圆的半径。具体如下图二所示。在这里输入点的坐标时使用的是ImGui::InputInt2，这可以接受一个大小为2的一维数组并修改其中的数值。





### 结果

本题具体结果录屏见文件中[HW3 basic-3](HW3%20basic-3.mp4)

# Bonus

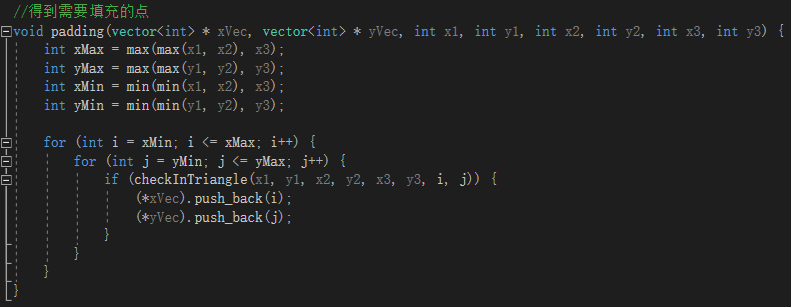
## 1.使用三角形光栅转换算法，用和背景不同的颜色，填充你的三角形

填充三角形的关键是判断点是否在三角形中。而在这里判断点是否在三角形才用的方法是用同侧法，即通过目标点与三角形的其中一点是否处于另外两点构成直线的同侧来判断点是否处于三角形中。

具体计算方法为当f(x0,y0)/f(x3,y3)>0时，目标点与三角形其中一点处于另外两点构成直线的同侧，其中f(x,y)= (y1-y2)\*x+(x2-x1)\*y+x1\*y2-x2\*y1。而当目标点与三个点都处于另外两点构成直线的同侧时则可以认为目标点在三角形中。具体代码如下。

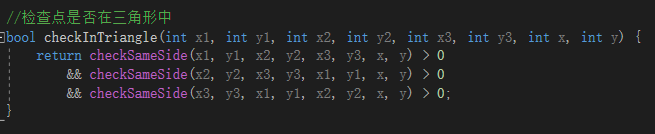
### Padding

通过三角形的三个顶点得到包围三角形的最小矩形，在这个矩形中进行扫描，将在三角形中的点放入向量。



### checkInTriangle&&checkSameSide

在checkInTriangle中分布检测目标点与三个顶点是否在同侧，当都处于同侧时，这个点在三角形中。在checkSameSide中通过函数计算f(x0,y0)/f(x3,y3)。





### 结果

在这里三角形被填充了白色。

