**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**TRƯỜNG ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

**KHOA TOÁN-TIN HỌC**

**- - -**    **- - -**



**BÁO CÁO ĐỒ ÁN MÔN**

**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**CHỦ ĐỀ 6:**

**KIỂM TRA SỰ CẮT NHAU GIỮA ĐƯỜNG THẲNG VÀ CÁC ĐỐI TƯỢNG HÌNH HỌC**

***Giáo viên hướng dẫn: TS.Nguyễn Ngọc Long***

***Sinh viên thực hiện : Phạm Thị Nhung-19110407***



***Nguyễn Thanh Tâm -19110435***

***Năm học 2021 - 2022***

# I.             MỤC TIÊU

-                    Nắm được cách xây dựng một ứng dụng Windows Desktop (Windows Desktop Application).

-                    Xây dựng được các đối tượng hình học:

+    Điểm (Point)

+    Đường thẳng (Line)

+    Hình tròn (Circle)

+    Hình ellipse (Ellipse)

+    Hình bán nguyệt (Semicircle)

+    Hình đa giác (Polygon)

+    Hình chữ nhật (Rectangle)

+    Hình vuông (Square)

+    Hình tam giác (Triangle)

-                    Xây dựng các class có tính đa hình, tính kế thừa, tính đóng gói.

-                    Sử dụng tính đa hình của class để định nghĩa các hàm kiểm tra sự cắt nhau giữa đường thẳng và đối tượng hình học, hàm vẽ các đối tượng.

-                    Điều chỉnh màu vẽ cho các đối tượng (các hình vẽ có đường thẳng đi qua sẽ có màu khác với các hình vẽ không có đường thẳng đi qua).

# II.             QUÁ TRÌNH THỰC HIỆN

**1.  Xây dựng môi trường ứng dụng Windows Desktop bằng win32 API**

Windows API là các hàm được sử dụng để tạo các ứng dụng Windows. Windows SDK (Software Development Kit) bao gồm header file, library (Windows API), sample, documentation và các tool (Windows SDK đã được tích hợp trong Visual Studio).

Mọi chương trình ứng dụng trong Windows bắt buộc phải có 2 hàm:

**WinMain()**: tương tự như hàm main() (trong Dos hoặc UNIX) khởi tạo chương trình ứng dụng. Có 2 nhiệm vụ chính:

–  Hiển thị cửa sổ ứng dụng lên màn hình

–

|  |
| --- |
|  |
|  |  |

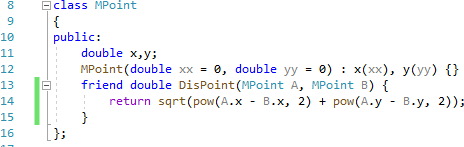
Tiến hành vòng lặp Message Khai báo hàm WinMain():

**WndProc():** sẽ nhận và xử lý các message gửi đến. Các message nào không được xử lý trong hàm này sẽ được xử lý trong hàm DefWindowProc() của HĐH Windows.

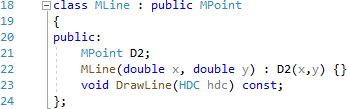
|  |
| --- |
|  |
|  |  |

Khai báo hàm WndProc():

# Xây dựng đối tượng Đường thẳng:

Ta cần xây dựng được đối tượng về Điểm trước khi xây dựng đối tượng Đường thẳng:

Từ đó ta có thể xây dựng đường thẳng dạng ax+by=0 (đi qua (0,0) và một điểm bất kỳ) theo cách:

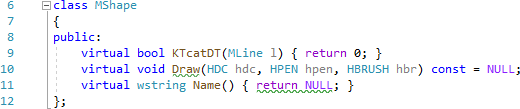


+ Các thuộc tính của đường thẳng: MPoint D2 là điểm thứ hai mà đường thẳng sẽ được vẽ tới, điểm bắt đầu là (0,0) vì phương trình đường thẳng là ax+by=0.

+ Phương thức vẽ đường thẳng:

Ta sử dụng 2 hàm: MoveToEx() và LineTo() để vẽ đường thẳng.

# Xây dựng các đối tượng hình học:

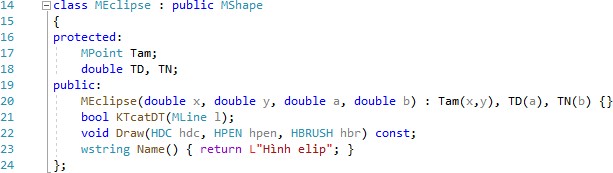
Tạo một class chung cho các đối tượng hình học (Shape), ta sẽ coi như class này là class Cha:

+ Hàm ảo virtual bool KTcatDT(): Kiểm tra xem đường thẳng có cắt đối tượng hình học hay không.

+ Hàm ảo virtual void Draw(): Vẽ đối tượng hình học bằng GDI (Graphic Device Interface).

+ Hàm ảo virtual wstring Name(): Trả về tên của đối tượng hình học.

Các đối tượng hình học còn lại sẽ kế thừa đối tượng Shape, ta có thể áp dụng tính đa hình của Class ở đây với các phương thức như: void Draw(), bool KTcatDT(), wstring Name()

*- Hình ellipse:*

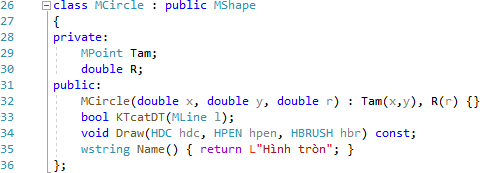
+ Các thuộc tính bao gồm: MPoint Tam là tâm của hình Ellipse mang kiểu dữ liệu là MPoint đã định nghĩa bên trên, double TD, TN là các thông số chỉ Trục dài và Trục ngắn của hình Ellipse.

+ Thuật toán kiểm tra đường thẳng cắt hình Ellipse:

Dùng phương trình hoành độ giao điểm của đường thẳng và hình Ellipse. Ta tạo ra được một phương trình bậc hai có nghiệm là hoành độ x. Sau đó ta kiểm tra delta của phương trình này để xác định xem đường thẳng có cắt hình Ellipse hay không.

+ Phương thức vẽ hình Ellipse:

Dùng hàm BOOL Ellipse (HDC hdc, int left, int top, int right, int bottom) thuộc thư viện wingdi.h. Với các tham số là: hdc (HDC) là một handle tới device context (thiết bị ngữ cảnh) của ứng dụng, các thông số tọa độ của các điểm trái tên và phải dưới của một region hình chữ nhật để vẽ hình Ellipse nội tiếp region hình chữ nhật này.

*- Hình tròn:*

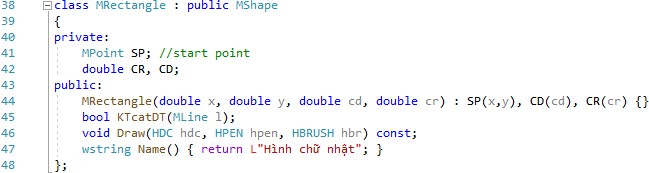
+ Các thuộc tính bao gồm: MPoint Tam là tâm của hình tròn, double R là bán kính của hình tròn.

+ Thuật toán kiểm tra đường thẳng cắt hình tròn:

Ta sẽ tính khoảng cách từ tâm hình tròn đến đường thẳng và kiểm tra xem nó có bé hơn bán kính hay không để suy ra được đường thẳng cắt hình tròn.

+ Phương thức vẽ hình tròn:

Vẫn dùng hàm BOOL Ellipse() như cách vẽ hình Ellipse nhưng khác ở các tham số.

*- Hình chữ nhật:*

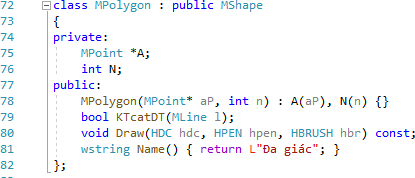
+ Các thuộc tính của hình chữ nhật: MPoint SP là điểm bắt đầu của hình chữ nhật để xác định vị trí hình hoặc vẽ hình (ta sẽ ngầm định đây là điểm trái trên của hình chữ nhật), double CR, CD là các độ dài của chiều rộng và chiều dài của hình chữ nhật.

+ Thuật toán kiểm tra đường thẳng cắt hình chữ nhật:

Ta dùng lý thuyết về dấu của các điểm trên tọa độ khi ở 2 mặt của một đường thẳng để xác định đường thẳng có cắt hình chữ nhật không.

+ Phương thức vẽ hình chữ nhật:

Ta sẽ dùng hàm BOOL Rectangle (HDC hdc, int left, int top, int right, int bottom) để vẽ hình chữ nhật với các tham số tương tự như vẽ hình Ellipse.

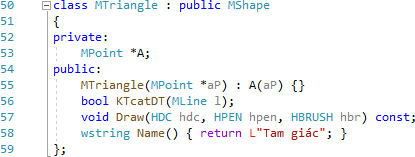
*- Hình đa giác:*

+ Các thuộc tính của hình đa giác: một mảng với kiểu dữ liệu là MPoint (tọa độ của các điểm để vẽ hình đa giác) và N là số điểm muốn vẽ.

+ Thuật toán kiểm tra đường thẳng cắt hình đa giác: Cũng như thuật toán kiểm tra đường thẳng cắt hình chữ nhật.

+ Phương thức vẽ hình đa giác:

Ta dùng hàm BOOL Polygon (HDC hdc, POINT \*apt, int cpt) thuộc thư viện wingdi.h. Các tham số của hàm: hdc (HDC) là một handle tới device context(thiết bị ngữ cảnh) của ứng dụng, \*apt (POINT) là một mảng có kiểu dữ liệu là POINT (kiểu dữ liệu struct cho tọa độ điểm trong Oxy), cpt (int) số điểm muốn vẽ. Tuy nhiên ta cần chuyển mảng MPoint sang mảng POINT để dùng được hàm này.

*- Hình tam giác:*

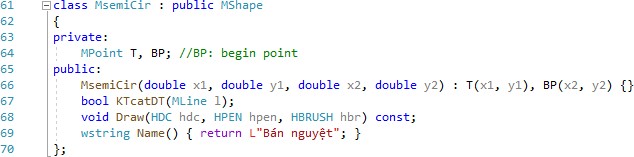
+ Các thuộc tính của hình tam giác: một mảng có kiểu dữ liệu là MPoint, là tọa độ 3 điểm của một hình tam giác.

+ Thuật toán kiểm tra đường thẳng cắt hình tam giác: Cũng như thuật toán kiểm tra đường thẳng cắt hình chữ nhật.

+ Phương thức vẽ hình tam giác:

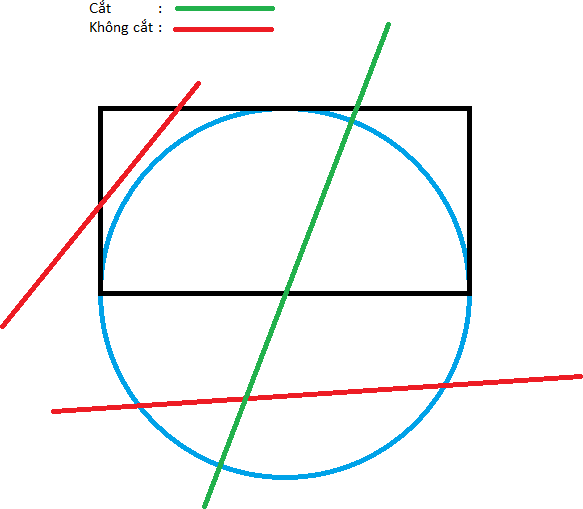
Giống với cách vẽ hình đa giác, nhưng thay mảng tham số và thay cpt thành 3.

*- Hình bán nguyệt:*



+ Các thuộc tính của hình bán nguyệt: MPoint T là tâm của hình bán nguyệt, MPoint BP là điểm bắt đầu vẽ cung tròn của hình bán nguyệt. Điểm cuối cùng của cung ta sẽ tự tính trong phần hàm vẽ.

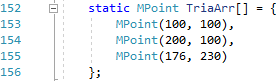
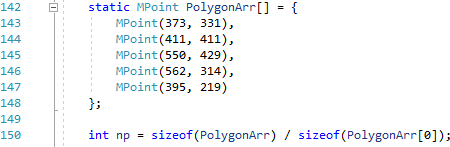
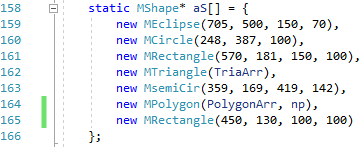
+ Thuật toán kiểm tra đường thẳng cắt hình bán nguyệt:

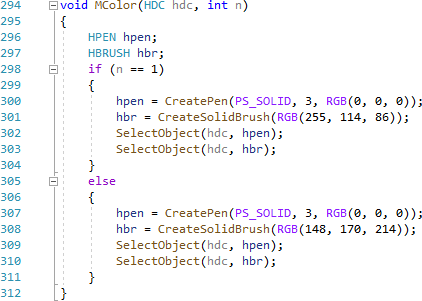
Ta dùng cách giao hình học. Xem như hình bán nguyệt là một hình tròn C toàn vẹn với tâm như tên và bán kính tính bằng T và BP. Ta sẽ xác định một khu vực hình chữ nhật H, có một cạnh trùng với đường kính của hình bán nguyệt. Sau đó ta sẽ kiểm tra xem liệu đường thẳng có cùng cắt cả hình tròn C và hình chữ nhật H hay không, nếu đáp ứng được điều kiện trên thì đường thẳng sẽ cắt hình bán nguyệt của ta, nếu không đáp ứng đủ điều kiện trên thì đường thẳng sẽ không cắt hình bán nguyệt. (Hình minh họa)

+ Phương thức vẽ hình bán nguyệt:

Ta sử dụng hàm BOOL Pie(HDC hdc, int left, int top, int right, int bottom, int xr1, int yr1, int xr2, int yr2) trong thư viện wingdi.h. Với các tham số là một region hình chữ nhật ngoại tiếp hình bán nguyệt, tọa độ của điểm bắt đầu cung tròn, tọa độ của điểm kết thúc cung tròn. Ta sẽ tính điểm kết thúc bằng công thức trung điểm trong tọa độ.

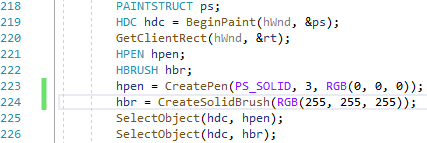
# Thiết lập dữ liệu cho các đối tượng trong chương trình chính (ở đây là trong file ShaNStrai.cpp)

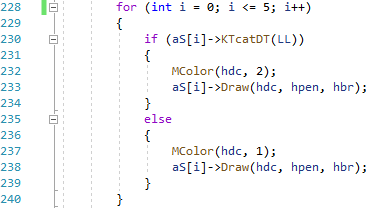
* Trước tiên ta khởi tạo dữ liệu cho đường thẳng:
* Một mảng chứa 3 điểm của hình tam giác:
* Một mảng chứa các điểm của hình đa giác:
* Một mảng kiểu Shape, chứa thông tin của tất cả sáu hình:
* Và một hàm chọn màu đối với các hình bị đường thẳng cắt qua, màu sẽ khác với các hình còn lại:



# Vẽ các hình học với các điều kiện cho trước:

Ta sẽ thực hiện công việc này trong case WM\_PAINT thuộc hàm WndProc().

* Bắt đầu trình vẽ, tạo bút vẽ và bút sơn mặc định:
* Ta sẽ duyệt qua mảng các hình học, kiểm tra và cho các hình bị cắt một màu vẽ khác với các hình không bị cắt:



* Ta vẽ thêm đường thẳng để biểu thị hình ảnh:
* Đóng trình vẽ:



# Tạo đối tượng để cho ra thông báo kết quả bằng chữ:

*- Đầu tiên ta cần một chuỗi thông báo:*

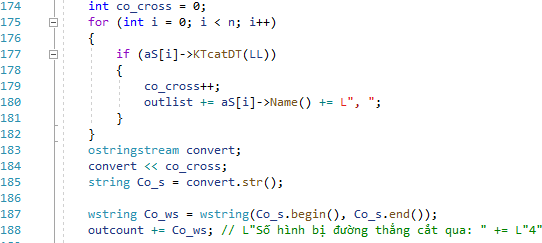
+ Tạo chuỗi kiểu wstring để cho ra chuỗi có ký tự thuộc bảng Unicode:

+ Tạo chuỗi wstring để thông báo số lượng hình học bị cắt:



+ Tạo chuỗi wstring để lưu danh sách các hình học bị cắt:

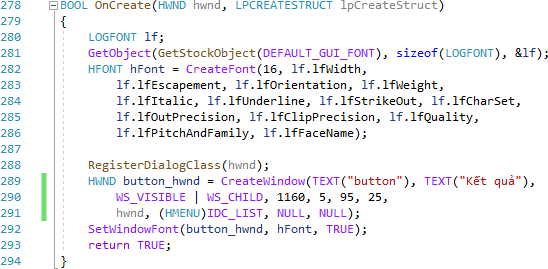
+ Ta đi xử lý chuỗi outcount và chuỗi outlist:



+ Nối 2 chuỗi con vào chuỗi kết quả chính và chuyển nó sang kiểu WCHAR\* để sử dụng trong các MessageBox hoặc DialogBox:

*- Sau đó ta sẽ tạo 1 button Kết quả, khi nhấn vào sẽ hiện ra MessagaBox chứa thông báo của*

*chương trình:*

+ Tạo button Kết quả:

+ Tạo MessageBox hiển thị thông báo của chương trình khi nhấn vào button Kết quả:

MessageBox này chưa chuỗi notiNc mà ta đã xử lý, chuỗi này chứa thông tin về số hình học bị cắt và danh sách các hình.

# KẾT QUẢ CHƯƠNG TRÌNH

**XIN CẢM ƠN.**