TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**



# CÀI ĐẶT THUẬT TOÁN XÉN TỈA ĐOẠN THẲNG

# BẰNG THUẬT TOÁN SUTHERLAND HOGMAN

# GVHD : Ths. ĐOÀN VŨ THỊNH

# SVTH : Đỗ Thúy Quỳnh

# MSSV : 59132066

# Lớp : 59.CNTT-3

Khánh Hòa, tháng 01 năm 2020

# MỤC LỤC

[MỤC LỤC 2](#_Toc30078132)

[TÓM TẮT 3](#_Toc30078133)

[1. GIỚI THIỆU 5](#_Toc30078134)

[1.1. Thuật toán xén tỉa đoạn thẳng 6](#_Toc30078135)

[1.1.1. Thuật toán Cohen –Sutherland 7](#_Toc30078136)

[1.1.2. Thuật toán Liang-Barsky 9](#_Toc30078137)

[1.1.3. Thuật toán cắt tỉa đa giác 12](#_Toc30078138)

[1.1.4.Thuật toán Sutherland-Hogman 13](#_Toc30078139)

[2. PHƯƠNG PHÁP NGHIÊN CỨU 15](#_Toc30078140)

[2.1. Cài đặt Dev –C, Graphics.h ,Winbmin 15](#_Toc30078141)

[2.3 Nhập dữ liệu 17](#_Toc30078142)

[2.4. Các bước xén tỉa đa giác 18](#_Toc30078143)

[2.4.1. Trường hợp cắt bên phải 18](#_Toc30078144)

[2.4.2. Trường hợp cắt bên trái 19](#_Toc30078145)

[2.4.3. Trường hợp cắt bên trên 20](#_Toc30078146)

[2.4.4. Trường hợp cắt bên dưới 21](#_Toc30078147)

[3.KẾT QUẢ 22](#_Toc30078148)

[3.1 Dữ liệu nhập từ bàn phím 22](#_Toc30078149)

[3.2.Kết quả của xén tỉa đa giác 22](#_Toc30078150)

[4.KẾT LUẬN 23](#_Toc30078151)

[PHỤ LỤC 24](#_Toc30078152)

[TÀI LIỆU THAM KHẢO 35](#_Toc30078153)

# TÓM TẮT

Trong phần Giới thiệu đã đưa ra được những khái niệm, sơ lược về các thuật toán xén tỉa đa giác , đoan thẳng. Kỹ thuật xén tỉa đa giác phải thông qua nhiều thuật toán tăng dần theo từng giai đoạn : Thuật toán Cohen –Sutherland, thuật toán Liang-Barsky, thuật toán Cyrus – Beck và thuật toán tối ưu nhất là thuật toán Sutherland-Hodgman.

Trong phần phương pháp cho biết cách cài đặt Dev-C/C++, thư viện graphics.h và winbmin.Sau đó , nhập dữ liệu vào máy tinh bằng cách nào.Tiếp theo đó là các bức xén tỉa đa giác với các trường hợp xảy ra khi cắt tỉa.

Tiếp theo , phần kết quả đưa ra các kết quả của đa giác sau khi được cắt xén theo thứ tự trái , phải, trên, dưới .

Cuối cùng , phần Thảo luận đưa ra những ưu điểm , nhược điểm của thuật toán Sutherland-Hodgman.

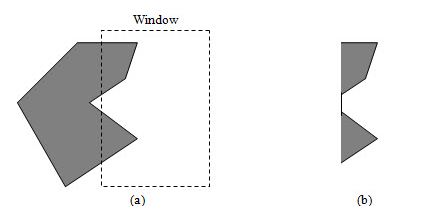
# 1. GIỚI THIỆU

Đồ họa máy tính là một trong những lĩnh vực hấp dẫn và phát triển nhanh của Công nghệ Thông tin. Nó được ra đời bởi sự kết hợp của hai lĩnh vực thông tin và truyền thông , và được sử dụng rộng rãi trong hầu hết các ứng dụng như khoa học và công nghệ, y học, giáo dục, kiến trúc và kể cả giải trí.

Thuật toán Sutherland của Hodgman là một thuật toán được sử dụng để cắt các đa giác. Nó hoạt động bằng cách lần lượt mở rộng từng dòng của đa giác clip lồi và chỉ chọn các đỉnh từ đa giác chủ đề nằm ở phía có thể nhìn thấy.

Thuật toán bắt đầu với một danh sách đầu vào của tất cả các đỉnh trong đa giác chủ đề. Tiếp theo, một bên của đa giác clip được mở rộng vô hạn theo cả hai hướng và đường dẫn của đa giác chủ đề được đi qua. Các đỉnh từ danh sách đầu vào được chèn vào danh sách đầu ra nếu chúng nằm ở phía có thể nhìn thấy của đường đa giác clip mở rộng và các đỉnh mới được thêm vào danh sách đầu ra nơi đường dẫn đa giác của đối tượng đi qua đường đa giác clip mở rộng.

Quá trình này được lặp lại lặp đi lặp lại cho mỗi bên đa giác clip, sử dụng danh sách đầu ra từ một giai đoạn làm danh sách đầu vào cho lần tiếp theo. Khi tất cả các cạnh của đa giác clip đã được xử lý, danh sách các đỉnh được tạo cuối cùng sẽ xác định một đa giác đơn mới hoàn toàn có thể nhìn thấy. Thuật toán xén tỉa trong đồ họa máy tính có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm phổ biến hiện này. Từ những phần mềm đơn giản như Paint, Powerpoint trong bộ Office của Window đến những ứng dụng thiết kế đồ họa chuyên nghiệp như Photoshop, AutoCad.



Hình 1.1. Clipping đa giác lõm trong hình (a)bởi một cửa sổ tạo ra hai vùng nối nhau trong hình (b).

## 1.1. Thuật toán xén tỉa đoạn thẳng

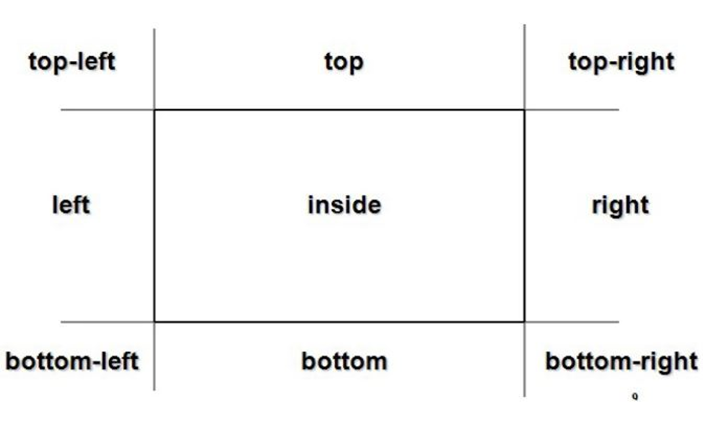
Thuật toán xén tỉa đoạn thẳng đầu tiên ra đời vào năm 1967 được phát minh bởi Danny Cohen and Ivan Sutherland.

Thuật toán xén tỉa đoạn thẳng là thuật toán xác định các điểm của đoạn thẳng nằm trong hay nằm ngoài của cửa sổ xén.

Thuật toán được chia thành các bước sau:

**Bước 1:** Gán mã vùng 4-bit cho mỗi điểm cuối của đoạn thẳng.

**Bước 2:** Mã vùng được xác định theo 9 vùng (hình 1.4) của mặt phẳng mà các điểm cuối nằm vào đó.



Hình 1.2. Mã vùng của 9 vùng mặt phẳng

*(*[*http://2.bp.blogspot.com/-i5W4-i8naHQ/VePDJkxzXlI/AAAAAAAAAFQ/1h7LzA-CbDQ/s1600/kiemtailieu6.jpg*](http://2.bp.blogspot.com/-i5W4-i8naHQ/VePDJkxzXlI/AAAAAAAAAFQ/1h7LzA-CbDQ/s1600/kiemtailieu6.jpg)*)*

**Bước 3:** Sử dụng các mã vùng để xác định các trường hợp của đoạn thẳng. Xét mã vùng của 2 điểm đầu cuối P1, P2 của đoạn thẳng cần xén. Ta có các trường hợp sau:

Nếu mã của P1 hoặc P2 đều = 0000 thì toàn bộ đoạn thẳng thuộc phần hiển thị.

Nếu mã của P1 và P2 có cùng một vị trí mà P1 AND P2 != 0000 => cùng phía.

Nếu không nằm trong 2 trường hợp sau đường thẳng cần được xén tỉa.

Tìm giao điểm của đường thẳng với cửa sổ, (với phần mở rộng của đường biên).

Nếu: Bit 1 là 1: cắt y = ymax.

Bit 2 là 1: cắt y = ymin.

Bit 3 là 1: cắt x = xmax.

Bit 4 là 1: cắt x = xmin.

Đối với thuật toán Cohen Shutherland, khi tìm giao điểm của đoạn thẳng cần xén tỉa với các cạnh của cửa sổ cắt bằng cách dùng các tham số của phương trình đường thẳng (hệ số góc được tính bằng công thức xuất hiện phép chia). Dẫn tới không tối ưu về mặt thời gian.

Để khắc phục điều đó thuật toán Liang-Barsky ra đời. thuật toán Liang-Barsky (được đặt theo tên của You-Dong Liang và Brian A. Barsky) sử dụng phương trình tham số của đường thẳng và bất đẳng thức mô tả phạm vi của cửa sổ cắt để xác định các giao điểm giữa đường thẳng và ửa sổ cắt. Với các giao điểm này, nó biết phần nào của đường nên được vẽ. Thuật toán này hiệu quả hơn đáng kể so với Cohen-Sutherland.

Cả 2 thuật toán trên đều có điểm hạn chế là cửa sổ xén là hình chữ nhật và không cho phép cửa sổ hình đa giác khác.

Thuật toán Cyrus – Beck (1978) đã có thể xén tỉa trên cửa sổ xén là đa giác.

### 1.1.1. Thuật toán Cohen –Sutherland

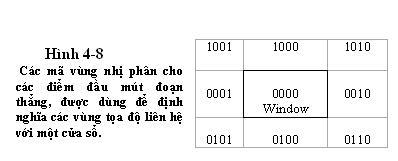
Ivan Edward Sutherland (sinh ngày 16 tháng 5 năm 1938) là một nhà khoa học máy tính và nhà tiên phong Internet của Mỹ, được nhiều người coi là "cha đẻ của đồ họa máy tính". Ông giảng dạy về đồ họa máy tính cùng với David C. Evans tại Đại học Utah vào những năm 1970. Sutherland, Evans và các sinh viên của họ thời đó đã phát minh ra một số nền tảng của đồ họa máy tính hiện đại. Ông đã nhận được giải thưởng Turing từ Hiệp hội Máy tính vào năm 1988 cho phát minh ra Sketchpad, tiền thân của loại giao diện người dùng đồ họa.

Kéo dài các biên cuẩ cửa sổ , chia mặt phẳng thành 9 vùng gồm cửa sổ và tám vùng xung quanh.

**Khái niệm mã vùng**

Một con số 4 bit nhị phân gọi là mã vùng sẽ được gán cho mỗi vùng để mô tả vị trí tương đối của vùng đó so với cửa sổ.

Bằng cách đánh số 1 đến 4 theo thứ tự từ phải sang trái , các bit của mã vùng dùng theo quy ước sau để chỉ một trong bốn vị trí tương đối của vùng so với cửa sổ bao gồm: trái, phải, trên, dưới.



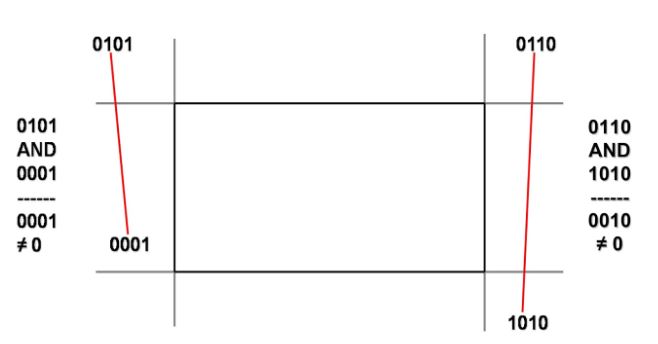
Hình 1.3 Các mã vùng nhị phân cho các điểm đầu mút đoạn thẳng , được dùng để định nghĩa các vùng tọa độ liên hệ với một cửa sổ

Giá trị 1 tương ứng với vị trí nào trong ã vùng sẽ chỉ ra rằng điểm đó ở vị trí tương ứng, ngược lại bit đó sẽ được đặt bằng 0.

Các giá trị bit trong mã vùng được tính bằng xác định tọa độ của điểm (x,y) thuộc vùng đó của cửa sổ .Bit 1 được đặt là 1 nếu x<xmin , các bit khác được tính tương tự.

Gán mã vùng tương ứng cho các điểm đầu cuối của đoạn thẳng P1, P2 cần xén là c1,c2 .Ta có nhận xét:

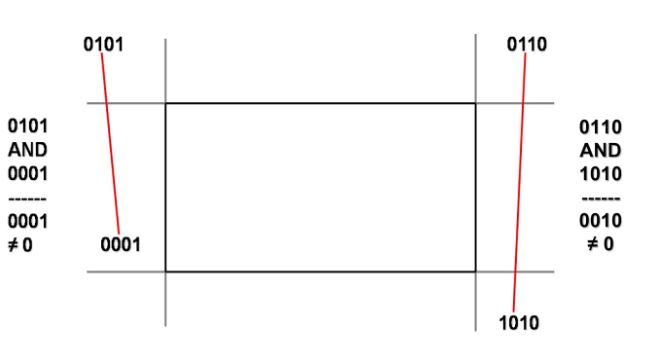
Các đoạn thẳng nằm hoàn toàn bên trong cửa sổ sẽ có c1=c2=0000, ứng với các đoạn này ,kết quả sau khi xén là chính nó.



Hình…

*(Nguồn:* [*http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien\_30.html*](http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien_30.html)*)*

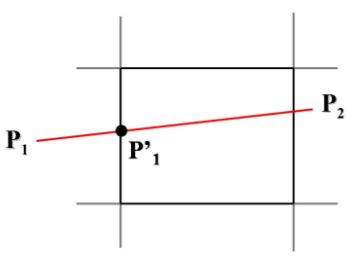
Nếu tồn tại k€ 1, ..., 4, sao cho với bit thứ k của c1, c2 đều có giá trị 1, lúc này đoạn thẳng sẽ nằm về cùng phía ứng với bit k so với cửa sổ, do đó nằm hoàn toàn ngoài cửa sổ. Đoạn này sẽ bị loại bỏ sau khi xén.



Hình…

*(Nguồn:* [*http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien\_30.html*](http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien_30.html)*)*

Nếu c1,c2 không thuộc về hai trường hợp trên , đoạn thẳng có thể hoặc không cắt ngang cửa sổ , chắc chắn sẽ tồn tại một điểm nằm ngoài cửa sổ , không mất tính tổng quát giả sử điểm đó là P1.



*(Nguồn* [*http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien\_30.html*](http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien_30.html)*)*

Các điểm giao với các biên cửa sổ của đoạn thẳng có thể tính từ phương trình tham số.

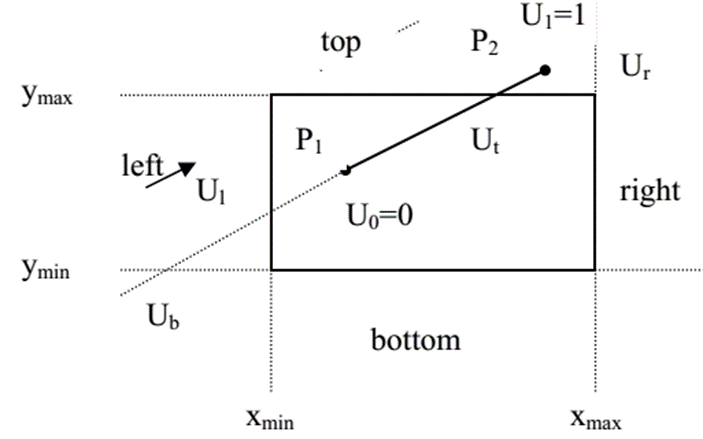
### 1.1.2. Thuật toán Liang-Barsky

Trong đồ họa máy tính, thuật toán Liang-Barsky (được đặt theo tên của You-Dong Liang và Brian A. Barsky). Liangky-Barsky sử dụng phương trình tham số của đườngh thẳng và bất đẳng thức mô tả phạm vi của cửa sổ cắt để xác định các giao điểm giữa đường thẳng và cửa sổ cắt. Với các giao điểm này, nó biết phần nào của đường nên được vẽ. Thuật toán này hiệu quả hơn đáng kể so với Cohen-Sutherland. Brian A. Barsky là giáo sư tại Đại học California, Berkeley, làm việc trong ngành đồ họa máy tính và mô hình hình học cũng như về khoa học thị giác. Ông cũng là thành viên của Nhóm nghiên cứu sau đại học về Kỹ thuật sinh học, một chương trình liên trường, giữa UC Berkeley và UC San Francisco.

Thuật toán Liang-Barsky được phát triển dựa vào việc phân tích dạng tham số của phương trình đoạn thẳng.

X=x1+t(x2-x1)=x1+tDx, Dx=x2-x1

Y=y1+t(y2-y1)=y1+tDy, Dy=y2=y1, 0<=t<=1



Ứng với mỗi giá trị t, ta có một điểm P tương ứng thuộc đường thẳng.

Các điểm ứng với t >=1 sẽ thuộc về tia P2x.

Các điểm ứng với t <=0 sẽ thuộc về tia P2x’.

Các điểm ứng với 0<=t <=**1 sẽ thuộc về đoạn thẳng P1P2.**

Tập hợp các điểm thuộc về phần giao giữa đoạn thẳng và cửa sổ ứng với các giá trị t thỏa hệ bất phương trình:

xmin<=x1+tDx<=xmax ymin<=y1+tDy<=ymax

0<=t<=1

Đặt: P1=-Dx, q1=x1-xmin.

P2=Dx, q2=xmax-x1.

P3=-Dy, q3=y1-ymin.

P4=Dy, q4=ymax-y1.

Lúc này ta viết hệ phương trình trên dưới dạng:

Pkt<=qk, k=1,2,3,4

0<=t<=1

Như vậy việc tìm đoạn giao thức chất là tìm nghiệm của hệ bất phương trình này. Có hai khả năng xảy ra đó là:

Hệ bất phương trình vô nghiệm , nghĩa là đường thẳng không có phần giao với cửa sổ nên sẽ bị loại.

Hệ bất phương trình có nghiệm , lúc này tập nghiệm sẽ là các giá trị t thỏa t € [t1,t2][0,1].

Ta xét các trường hợp :

Nếu k€ {1,2,3,4}: (Pk=0) (qk<0) thì rõ ràng bất phưng trình ứng với k trên là vô nghiệm do đó hệ phương trình vô nghiệm.

Nếu thì các bất phương trình mà ứng với Pk=0 là các bất phương trình hiển nhiên , lúc này hệ bất phương trình cần giải tương đương với hệ bất phương trình có Pk.

Với các bất phương trình Pkt, ta có t

Với các bất phương trình Pkt mà Pk>0, ta có t .

Vậy nghiệm của hệ bất phương trình là [t1,t2] với:

t2

t1

Nếu hệ trên có nghiệm thì đoạn giao Q1,Q2 sẽ là Q1(x1+t1Dx, y1+t1Dy), Q2(x1+t2Dx, y1+t2Dy).

Nếu nhận xét thuật toán này ở khía cạnh hình học ta có:

Trường hợp tương ứng với trường hợp đoạn thẳng cần xét song song với một trong các biên của cửa sổ (Pk=0) và nằm ngoài cửa sổ (qk<0) nên sẽ bị loại bỏ sau khi nén.

Với Ph, giá trị sẽ tương ứng với giao điểm của đoạn thẳng với biên k kéo dài của cửa sổ. Trường hợp Pk < 0, kéo dài các biên cửa sổ và đoạn thẳng về vô cực , ta có đường thẳng đang xét sẽ có hướng đi từ bên ngoài vào bên trong cửa sổ.

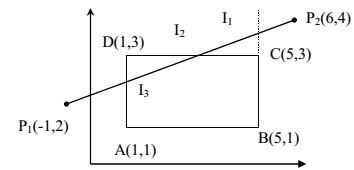
Nếu Ph>0 đường thẳng sẽ có hướng đi từ bên trong cửa sổ đi ra . Do đó hai đầu mút của đoạn giao sẽ ứng với các giá trị t1,t2 được tính như sau: Giá trị t1 chính là giá trị lớn nhất của các mà Ph<0 (đường thẳng đi từ ngoài vào trong cửa sổ) và 0; giá trị t2 chính là giá trị nhỏ nhất của các đừng thẳng đi từ trong cửa sổ đi ra ) và 1.

Với , giá trị sẽ tương ứng với giao điểm của đoạn thẳng với biên k kéo dài cửa sổ. Trường hợp Ph , kéo dài các biên cửa sổ và đoạn thẳng về vô cực,ta có đường thẳng đang xét sẽ có hướng đi từ bên trong cửa sổ. Nếu Pk , đường thẳng sẽ có hướng đi từ bên trong cửa sổ đi ra. Do đó hai đầu mút của đoạn giao sẽ ứng với các giá trị t1,t2 được tính như sau: Giá trị t1 chings là giá trị lứn nhất của các rh=qh/Ph, mà ph<0 (đường thẳng đi từ ngoài vào trong cửa sổ) và 0; giá trị t2 chính là giá trị nhỏ nhất của các rh=qh/Ph mà Ph0 ( đường thẳng đi từ trong cửa sổ đi ra )và 1.

### 1.1.3. Thuật toán cắt tỉa đa giác

Để cắt đa giác, chúng ta cần sửa đổi các bước xén tỉa đoạn thẳng. Một cạnh đa giác được xử lý với một cạnh của của sổ xén tùy thuộc vào hướng của cạnh đa giác đến cửa sổ xén. Những gì chúng ta cần là phần giới hạn sau khi cắt. Để xén tỉa đa giác, chúng ta cần một thuật toán sẽ tạo ra một hoặc nhiều vùng xén (các cạnh của cửa sổ xén). Đầu ra của một đa giác đã được xén tỉa phải là một chuỗi các đỉnh xác định ranh giới của đa giác được xén tỉa.

*Bài mẫu:* hãy sử dụng thuật toán Hodgman để cắt xén đoạn thẳng nối P1(-1,2) đến P2(6,4) trên cửa sổ A(1,1), B(5,1), C(5,3) và D(1,3)



Theo thuật toán Hodgman ta xén P1P2 dựa trên từng cạnh.  
 AB: ta xét C=(x2-x1)(y-y1) – (y2-y1)(x-x1)

Điểm P1: C=(5-1)(2-1)-(1-1)(-1-1) = 4 >0 nằm bên trái

Điểm P2: C= (5-1)(4-1)-(1-1)(-1-1) = 12 >0 nằm bên trái

Vậy P1P2 đều được lưu.

**Xét đoạn BC:**

Điểm P1: C = (5-5)(2-1)-(3-1)(-1-5)= 12 >0 nằm bên trái.

Điểm P2: C = - (3-1)(6-5) = -2 <0 nằm bên phải.

**Giao điểm I1:**

I1(5,26/7) vậy P1I1 lưu lại

**Xét đoạn CD:**

Điểm P1: C = (1-5)(2-3) = 4 >0 nằm bên trái

Điểm I1: C=(1-5)(26/7 -3) = -20/7 <0 nằm bên phải

**Giao điểm I2:**

VậyI2(5/2,3) có P1I2 được lưu

**Xét đoạn CD**

Điểm P1: C = (1-5)(2-3) = 4 >0 nằm bên trái

Điểm I1: C=(1-5)(26/7 -3) = -20/7 <0 nằm bên phải

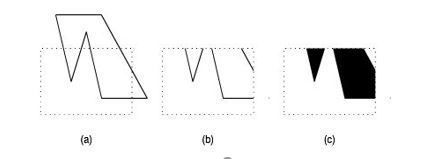
Giao điểm I2:

Vậy I2(5/2,3) có P1I2 được lưu

### 1.1.4.Thuật toán Sutherland-Hogman

Chúng ta có thể hiệu chỉnh các thuật toán xén đoạn thẳng để xén đa giác bằng cách xem đa guacs như là một tập các đoạn thẳng liên tiếp nối với nhau. Tuy nhiên , kết quả sau khi xén nhiều khi lại là tập các đoạn thẳng rời nhau.

Điều chúng ta mong muốn ở đây đó là kết quả sau khi nén pahir là một các đa giác để sau này có thể chuyển thành các vùng tô.



Hình 1.4. Thuật toán cắt tỉa Sutherland-Hogman

Thuật toán này sẽ tiến hành xén đa giác lần lượt với các biên cửa sổ .Đầu tiên, đa giác sẽ được xén dọc theo biên trái của cửa sổ, kết quả sau bước này sẽ được dùng để xén tiếp biên phải , rồi cư tương tự như vậy cho các biên trên , dưới. Sau khi xén hết vưới bốn biên của cửa sổ , ta được kết quả cuối cùng.

Với mỗi lần xén đa giác dọc theo một biên nào đó của cửa sổ , nếu gọi Vi,Vi+1 là hai đỉnh kề cạnh Vi, Vi+1 ta có 4 trường hợp có thể xảy ra khi xét từng cặp đỉnh của đa giác ban đầu với biên của cửa sổ như sau:

|  |  |
| --- | --- |
| **TH1:** Nếu Vi nằm ngoài , Vi+1 nằm trong ta lưu giao điểm I của ViVi+1 với biên của cửa sổ và Vi+1(hình ...) | hình |
| **TH2:** Nếu cả Vi,Vi+1 đều nằm trong thì lưu cả Vi,Vi+1. |  |
| **TH3:** Nếu Vi nằm trong , Vi+1 nằm ngoài thì lưu Vi và I. |  |
| **TH4:** Nếu Vi, Vi+1 nằm ngoài thì không lưu gì cả. |  |

# 2. PHƯƠNG PHÁP NGHIÊN CỨU

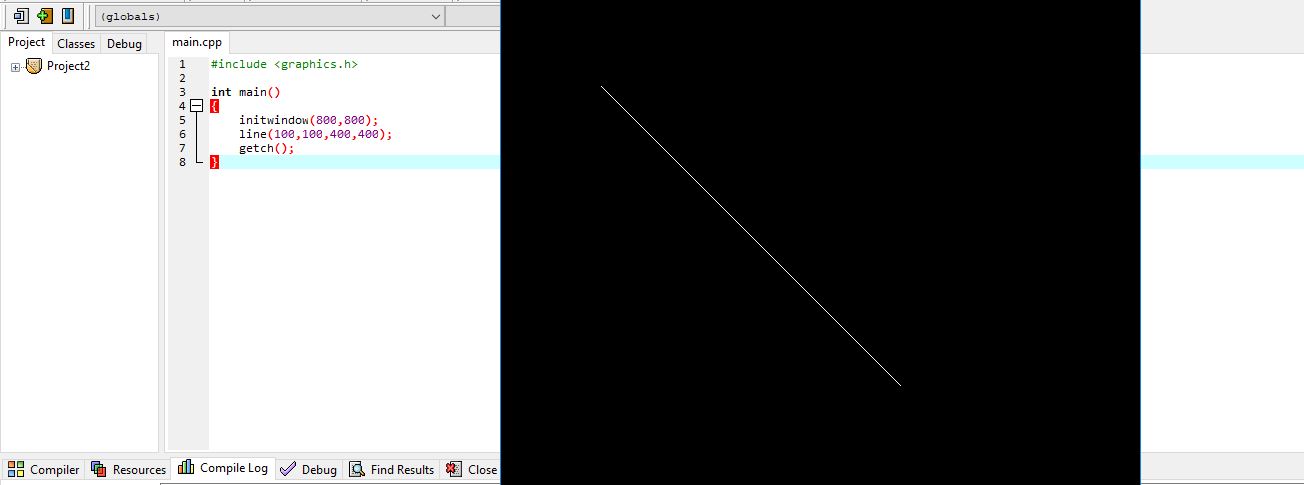
## 2.1. Cài đặt Dev –C, Graphics.h ,Winbmin

Dev-C++ là một [môi trường phát triển tích hợp](https://vi.wikipedia.org/wiki/M%C3%B4i_tr%C6%B0%E1%BB%9Dng_ph%C3%A1t_tri%E1%BB%83n_t%C3%ADch_h%E1%BB%A3p) [tự do](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do) (IDE) được phân phối dưới hình thức [giấy phép Công cộng GNU](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_C%C3%B4ng_c%E1%BB%99ng_GNU) hỗ trợ việc lập trình bằng [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh))/[C++](https://vi.wikipedia.org/wiki/C%2B%2B). Nó cũng nằm trong bộ trình dịch [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) [MinGW](https://vi.wikipedia.org/w/index.php?title=MinGW&action=edit&redlink=1). Chương trình IDE này được viết bằng ngôn ngữ [Delphi](https://vi.wikipedia.org/wiki/Delphi_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)).

Dev-C++ nói chung là một chương trình chỉ chạy trên [Windows](https://vi.wikipedia.org/wiki/Microsoft_Windows). Tuy nhiên cũng có một [phiên bản cho Linux](http://freshmeat.net/projects/dev-cpp), nhưng vẫn trong giai đoạn [alpha](https://vi.wikipedia.org/w/index.php?title=V%C3%B2ng_%C4%91%E1%BB%9Di_c%E1%BB%A7a_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) và chưa được cập nhật trong vòng hơn 6 năm qua.

Trước hết phải đảm bảo đã cài đặt phần mềm Devc++ trên máy tính ***(nguồn:*** [*https://sourceforge.net/projects/orwelldevcpp/*](https://sourceforge.net/projects/orwelldevcpp/)*).*

File cài đặt (kèm Devc++ và thư viện đồ họa):



hình

**Thêm thư viện đồ họa:**

**Đường dẫn (Path):**

Với Windows 32 bit:

Path=”C:\Program Files\Dev-Cpp”

Với Windows 64 bit:

Path=”C:\Program Files (x86)\Dev-Cpp”

**Hướng dẫn cài đặt:**

**Đối với Dev C++ 32 bit:**

**Bước 1:** Copy 2 file “*graphics.h*” và “winbgim.h” vào thư mục (Path)\MinGV32\include”.

**Bước 2:** Copy file “*libbgi.a*” vào thư mục “(Path)\MinGV32\lib”.

**Bước 3:** Copy 2 file “6-ConsoleAppGraphics.template” và “ConsoleApp\_cpp\_graph.txt” vào thư mục “(Path)\Templates”.

**Đối với Dev C++ 64 bit:**

**Bước 1:** Copy 2 file “*graphics.h*” và “*winbgim.h*” vào thư mục “(Path)\MinGW64\x86\_64-w64-mingw32\include”.

**Bước 2:** Copy file “*libbgi.a*” vào thư mục “(Path)\MinGW64\x86\_64-w64-mingw32\lib”.

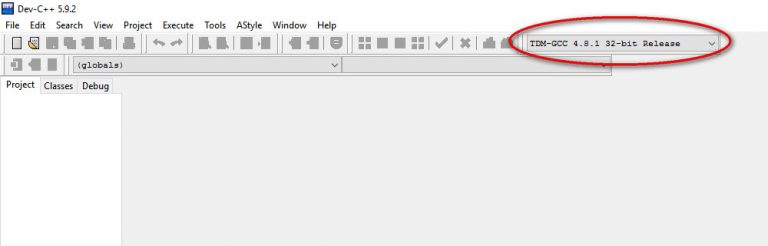
**Bước 3:** Copy 2 file “6-ConsoleAppGraphics.template” và “ConsoleApp\_cpp\_graph.txt” vào thư mục “(Path)\Templates”.

*(*[*https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library*](https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library)*).*

**Hướng dẫn sử dụng:**

Sau khi cài đặt theo các bước như hướng dẫn ở trên, các bạn mở Dev C++ lên.

**Bước 1*:*** Nếu bạn cài Dev C++ 32 bit thì bỏ qua bước này, còn nếu là 64 bit thì các bạn chọn trình biên dịch là***TDM-GCC 32 bit*** ứng với phiên bản Dev C++ bạn cài đặt **(Đây là bước bắt buộc và phải làm từ đầu, nếu không thì sẽ không chạy được đồ họa).**



Hình 2.1.Chọn trình biên dịch

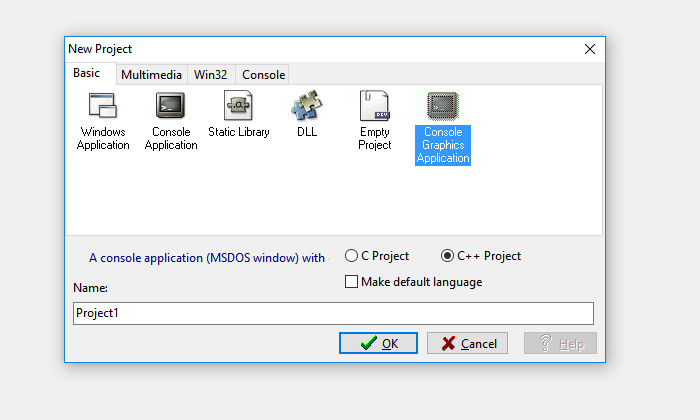
***Bước 2*:** Tạo một Project mới bằng cách chọn File -> New -> Project…

Chọn Console Graphics Application.

Tick vào ô C++ Project.

Gõ tên Project vào ô Name.

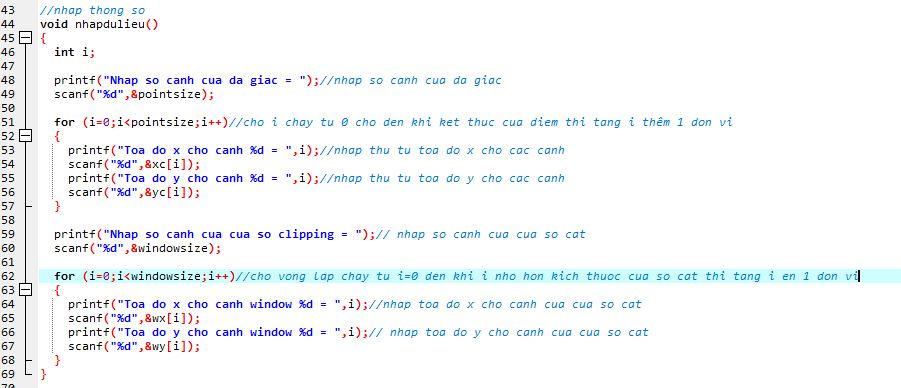
Click Ok và chọn vị trí lưu.



Hình 2.2. Đặt tên cho project

## 2.3 Nhập dữ liệu

Dữ liệu đầu vào bao gồm 2 tham số: cửa sổ cắt và đa giác cần cắt. Dữ liệu này được nhập từ bàn phím . Danh sách các đỉnh của đa giác được lưu trữ trong mảng một chiều có tên xc và yc, các đỉnh của cửa sổ cắt được lưu trữ trong danh sách có tên wx và wy.



Hình 2.3. Nhập dữ liệu từ bàn phím

## 2.4. Các bước xén tỉa đa giác

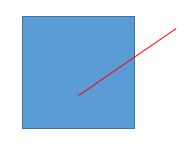
Cho 2 đỉnh P1(x1,y1),P2(x2,y2) và cửa sổ ABCD(AB-cạnh trái,BC cạnh trên,CD-cạnh phải,DA cạnh dưới.

Đỉnh P1 nằm bên ngoài cửa sổ thì đỉnh P2 nằm bên trong cửa sổ nên đỉnh P2 được thêm vào. Các điểm giao nhau của ranh giới cửa sổ và mặt đa giác cũng được thêm vào dòng đầu ra.

Cả hai các đỉnh P1P2 đều nằm bên trong ranh giới cửa sổ thì chỉ có đỉnh thứ hai được thêm vào.

Đỉnh P1 nằm bên trong cửa sổ và đỉnh P2 là nằm ở cửa sổ bên ngoài. Cạnh cắt cửa sổ sẽ được thêm vào danh sách đầu ra. Cả hai đỉnh bên ngoài cửa sổ thì không có gì được thêm vào.

### 2.4.1. Trường hợp cắt bên phải

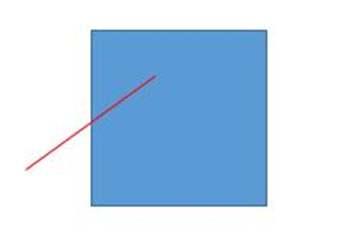


Hình 2.4. Vị trí của cạnh cần cắt của đa giác so với cạnh phải của cửa sổ cắt

Có 4 trường hợp xảy ra:

|  |  |
| --- | --- |
| **TH1:** Từ ngoài vào trong. |  |
| **TH2:** Từ trong ra ngoài. |  |
| **TH3:** Nằm hoàn toàn bên trong. |  |
| **TH4:** Nằm hoàn toàn bên ngoài. |  |

### 2.4.2. Trường hợp cắt bên trái

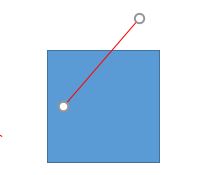


Hình 2.5. Vị trí của cạnh cần cắt của đa giác so với cạnh trái của cửa sổ cắt

Có 4 trường hợp xảy ra :

|  |  |
| --- | --- |
| **TH1:**Từ ngoài vào trong |  |
| **TH2:**Từ trong ra ngoài |  |
| **TH3:**Nằm hoàn toàn bên trong |  |
| **TH4:**Nằm hoàn toàn bên ngoài |  |

### 2.4.3. Trường hợp cắt bên trên

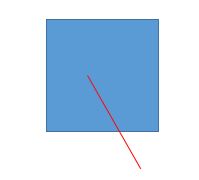


Hình 2.6. Vị trí của cạnh cần cắt của đa giác so với cạnh trên của cửa sổ cắt

Có 4 trường hợp xảy ra:

|  |  |
| --- | --- |
| **TH1:**Từ ngoài vào trong |  |
| **TH2:**Từ trong ra ngoài |  |
| **TH3:** Nằm hoàn toàn bên trong |  |
| **TH4:**Nằm hoàn toàn bên ngoài |  |

# 2.4.4. Trường hợp cắt bên dưới



Hình 2.7. Vị trí của cạnh cần cắt của đa giác so với cạnh dưới của cửa sổ cắt

Có 4 trường hợp xảy ra:

|  |  |
| --- | --- |
| **TH1:** Từ ngoài vào trong |  |
| **TH2:** Từ trong ra ngoài |  |
| **TH3:** Nằm hoàn toàn bên trong |  |
| **TH4:** Nằm hoàn toàn bên ngoài |  |

# 3.KẾT QUẢ

# 3.1 Dữ liệu nhập từ bàn phím

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

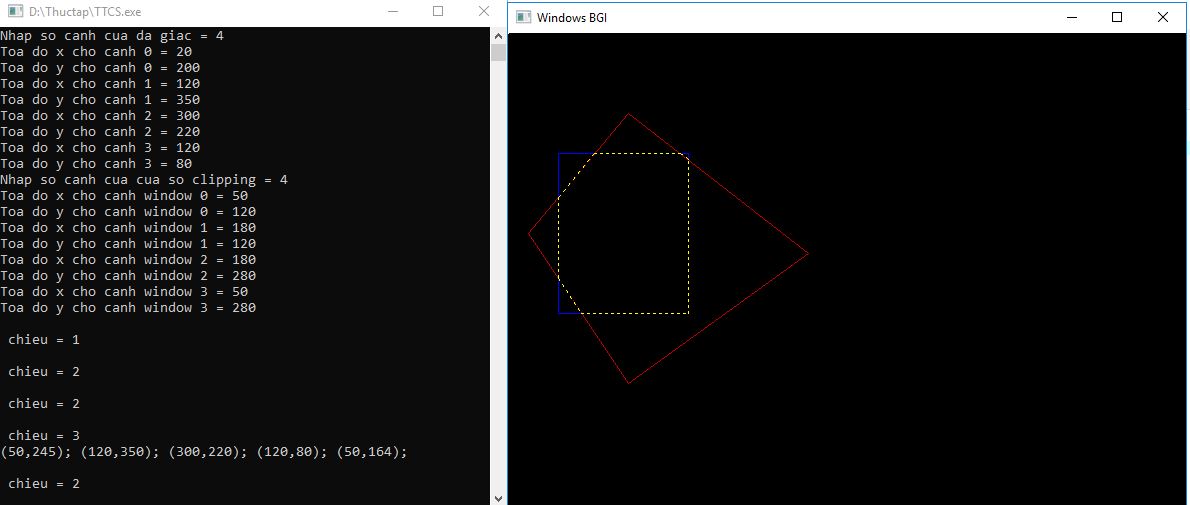
Hình 3.1. Dữ liệu giao diện từ bàn phím

## 3.2.Kết quả của xén tỉa đa giác

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | (b) | (c) | (d) |

Hình 3.2. Các trường hợp xén tỉa của đa giác

Hình 3.2 có cửa sổ màu xanh dương và đa giác màu vàng.Trong đó, hình 3.2a phần màu xanh là vùng giới hạn bởi cửa sổ cắt nằm ở phía bên phải. Hình 3.2b phần màu xanh là vùng giởi hạn bởi cửa sổ cắt nằm ở phía bên trái. Hình 3.2c phần màu xanh là vùng giới giới hạn bởi cửa sổ cắt nằm phía bên dưới. Hình 3.2d phần màu xanh là vùng giới hạn bởi cửa sổ cắt nằm phía bên trên.



Hình 3.3.Kết quả

# 4.KẾT LUẬN

Ưu điểm:

Thuật toán Sutherland –Hodgman cho kết quả chính xác khi làm việc với các đa giác lồi.

Nhược điểm:

Giải thuật xén đa giác Sutherland-Hodgman còn mắc phải nhược điểm đó là:

Trong tình huống mà kết quả xén bắt buộc phải cho ra 2 đa giác riêng biệt thì nó gộp lại làm 1 đa giác bởi một cạnh liên kết nối đến cả 2 đa giác.

# PHỤ LỤC

#include <graphics.h>

#include <math.h>

#define ROUND(a) ((int)(a+0.5))

#define OUT2IN 1

#define INSIDE 2

#define IN2OUT 3

#define OUTSIDE 4

#define INVALID 5

//khai bao bien

int xc[100];

int yc[100];

int wx[100];

int wy[100];

int pointsize;

int windowsize;

int xc\_tam[100];

int yc\_tam[100];

int pointsize\_tam;

//chuong trinh con

void draw\_object()

{

int i;

//ve cac canh cua da giac

setcolor(RED);

for(i=0;i<pointsize;i++)

{

line(xc[i],yc[i],xc[(i+1)%pointsize],yc[(i+1)%pointsize]);

delay(1000);

}

//ve cac canh cua so clipping

setcolor(BLUE);

for(i=0;i<windowsize;i++)

{

line(wx[i],wy[i],wx[(i+1)%windowsize],wy[(i+1)%windowsize]);

delay(1000);

}

}

//nhap thong so

void nhapdulieu()

{

int i;

printf("Nhap so canh cua da giac = ");

scanf("%d",&pointsize);

for (i=0;i<pointsize;i++)

{

printf("Toa do x cho canh %d = ",i);

scanf("%d",&xc[i]);

printf("Toa do y cho canh %d = ",i);

scanf("%d",&yc[i]);

}

printf("Nhap so canh cua cua so clipping = ");

scanf("%d",&windowsize);

for (i=0;i<windowsize;i++)

{

printf("Toa do x cho canh window %d = ",i);

scanf("%d",&wx[i]);

printf("Toa do y cho canh window %d = ",i);

scanf("%d",&wy[i]);

}

}

int clipLeft(int x,int x1,int x2)

{

if(x1<x && x2>=x)

{

return OUT2IN;

}

else

{

if(x1>=x && x2>=x)

{

return INSIDE;

}

else

{

if(x1>=x && x2<x)

{

return IN2OUT;

}

else

{

if(x1<x && x2<x)

{

return OUTSIDE;

}

else

{

return INVALID;

}

}

}

}

}

int clipBottom(int y,int y1,int y2)

{

if(y1<y && y2>=y)

{

return OUT2IN;

}

else

{

if(y1>=y && y2>=y)

{

return INSIDE;

}

else

{

if(y1>=y && y2<y)

{

return IN2OUT;

}

else

{

if(y1<y && y2<y)

{

return OUTSIDE;

}

else

{

return INVALID;

}

}

}

}

}

int clipRight(int x,int x1,int x2)

{

if(x1>x && x2<=x)

{

return OUT2IN;

}

else

{

if(x1<=x && x2<=x)

{

return INSIDE;

}

else

{

if(x1<=x && x2>x)

{

return IN2OUT;

}

else

{

if(x1>x && x2>x)

{

return OUTSIDE;

}

else

{

return INVALID;

}

}

}

}

}

int clipTop(int y,int y1,int y2)

{

if(y1>y && y2<=y)

{

return OUT2IN;

}

else

{

if(y1<=y && y2<=y)

{

return INSIDE;

}

else

{

if(y1<=y && y2>y)

{

return IN2OUT;

}

else

{

if(y1>y && y2>y)

{

return OUTSIDE;

}

else

{

return INVALID;

}

}

}

}

}

void leftclipper()

{

int chieu;

int i;

int xwmin;

int xa;

int ya;

int xb;

int yb;

int j;

int xnew;

int ynew;

xwmin=wx[0];//dinh 1

j=0;

for(i=0;i<pointsize;i++)

{

xa=xc[i];

ya=yc[i];

xb=xc[i+1];

yb=yc[i+1];

chieu=clipLeft(xwmin,xa,xb);

switch (chieu)

{

case OUT2IN:

{

xnew = xwmin;

ynew = ya + (float)(yb-ya)\*(float)(xwmin-xa)/(float)(xb-xa);

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

xc\_tam[j]=xb;

yc\_tam[j]=yb;

j++;

break;

}

case INSIDE:

{

xnew = xb;

ynew = yb;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case IN2OUT:

{

xnew = xwmin;

ynew = ya + (float)(yb-ya)\*(float)(xwmin-xa)/(float)(xb-xa);

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case OUTSIDE:

{

break;

}

}

printf("\n chieu = %d\n",chieu);

}

//gan lai danh sach canh moi

for(i=0;i<j;i++)

{

xc[i]=xc\_tam[i];

yc[i]=yc\_tam[i];

}

//gan so dinh moi sau khi leftclipper

pointsize=j;

for(i=0;i<j;i++)

{

printf("(%d,%d); ",xc[i],yc[i]);

}

printf("\n");

}

void rightclipper()

{

int chieu;

int i;

int xwmax;

int xa;

int ya;

int xb;

int yb;

int j;

int xnew;

int ynew;

xwmax=wx[2];;//dinh 3

j=0;

for(i=0;i<pointsize;i++)

{

xa=xc[i];

ya=yc[i];

xb=xc[i+1];

yb=yc[i+1];

chieu=clipRight(xwmax,xa,xb);

switch (chieu)

{

case OUT2IN:

{

xnew = xwmax;

ynew = ya + (float)(yb-ya)\*(float)(xwmax-xa)/(float)(xb-xa);

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

xc\_tam[j]=xb;

yc\_tam[j]=yb;

j++;

break;

}

case INSIDE:

{

xnew = xb;

ynew = yb;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case IN2OUT:

{

xnew = xwmax;

ynew = ya + (float)(yb-ya)\*(float)(xwmax-xa)/(float)(xb-xa);

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case OUTSIDE:

{

break;

}

}

printf("\n chieu = %d\n",chieu);

}

//gan lai danh sach canh moi

for(i=0;i<j;i++)

{

xc[i]=xc\_tam[i];

yc[i]=yc\_tam[i];

}

//gan so dinh moi sau khi rightclipper

pointsize=j;

for(i=0;i<j;i++)

{

printf("(%d,%d); ",xc[i],yc[i]);

}

printf("\n");

}

void bottomclipper()

{

int chieu;

int i;

int ywmax;

int xa;

int ya;

int xb;

int yb;

int j;

int xnew;

int ynew;

ywmax=wy[2];//dinh 3

j=0;

for(i=0;i<pointsize;i++)

{

xa=xc[i];

ya=yc[i];

xb=xc[i+1];

yb=yc[i+1];

chieu=clipTop(ywmax,ya,yb);

switch (chieu)

{

case OUT2IN:

{

xnew = xa + (float)(ywmax-ya)\*(xb-xa)/(yb-ya);

ynew = ywmax;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

xc\_tam[j]=xb;

yc\_tam[j]=yb;

j++;

break;

}

case INSIDE:

{

xnew = xb;

ynew = yb;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case IN2OUT:

{

xnew = xa + (float)(ywmax-ya)\*(xb-xa)/(yb-ya);

ynew = ywmax;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case OUTSIDE:

{

break;

}

}

printf("\n chieu = %d\n",chieu);

}

//gan lai danh sach canh moi

for(i=0;i<j;i++)

{

xc[i]=xc\_tam[i];

yc[i]=yc\_tam[i];

}

//gan so dinh moi sau khi rightclipper

pointsize=j;

for(i=0;i<j;i++)

{

printf("(%d,%d); ",xc[i],yc[i]);

}

printf("\n");

}

void topclipper()

{

int chieu;

int i;

int ywmin;

int xa;

int ya;

int xb;

int yb;

int j;

int xnew;

int ynew;

ywmin=wy[0];//dinh 1

j=0;

for(i=0;i<pointsize;i++)

{

xa=xc[i];

ya=yc[i];

xb=xc[i+1];

yb=yc[i+1];

chieu=clipBottom(ywmin,ya,yb);

switch (chieu)

{

case OUT2IN:

{

xnew = xa + (float)(ywmin-ya)\*(xb-xa)/(yb-ya);

ynew = ywmin;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

xc\_tam[j]=xb;

yc\_tam[j]=yb;

j++;

break;

}

case INSIDE:

{

xnew = xb;

ynew = yb;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case IN2OUT:

{

xnew = xa + (float)(ywmin-ya)\*(xb-xa)/(yb-ya);

ynew = ywmin;

xc\_tam[j]=xnew;

yc\_tam[j]=ynew;

j++;

break;

}

case OUTSIDE:

{

break;

}

}

printf("\n chieu = %d\n",chieu);

}

//gan lai danh sach canh moi

for(i=0;i<j;i++)

{

xc[i]=xc\_tam[i];

yc[i]=yc\_tam[i];

}

//gan so dinh moi sau khi rightclipper

pointsize=j;

for(i=0;i<j;i++)

{

printf("(%d,%d); ",xc[i],yc[i]);

}

printf("\n");

}

void draw\_object\_afterclip()

{

int i;

//ve cac canh cua da giac

setcolor(YELLOW);

setlinestyle(1,0,1);

for(i=0;i<pointsize;i++)

{

line(xc[i],yc[i],xc[(i+1)%pointsize],yc[(i+1)%pointsize]);

delay(1000);

}

}

//chuong trinh chinh

int main()

{

nhapdulieu();

initwindow(800,800);

draw\_object();

//gan them 1 dinh o cuoi cung chinh la dinh xuat phat

xc[pointsize]=xc[0];

yc[pointsize]=yc[0];

leftclipper();

//gan them 1 dinh o cuoi cung chinh la dinh xuat phat

xc[pointsize]=xc[0];

yc[pointsize]=yc[0];

rightclipper();

//gan them 1 dinh o cuoi cung chinh la dinh xuat phat

xc[pointsize]=xc[0];

yc[pointsize]=yc[0];

bottomclipper();

//gan them 1 dinh o cuoi cung chinh la dinh xuat phat

xc[pointsize]=xc[0];

yc[pointsize]=yc[0];

topclipper();

draw\_object\_afterclip();

getch();

return 0;

}

# TÀI LIỆU THAM KHẢO

[*http://2.bp.blogspot.com/-i5W4-i8naHQ/VePDJkxzXlI/AAAAAAAAAFQ/1h7LzA-CbDQ/s1600/kiemtailieu6.jpg*](http://2.bp.blogspot.com/-i5W4-i8naHQ/VePDJkxzXlI/AAAAAAAAAFQ/1h7LzA-CbDQ/s1600/kiemtailieu6.jpg)*.*

[*http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien\_30.html*](http://cdtink10-nhom1-dhmt.blogspot.com/2015/08/cac-thuat-toan-xen-hinh-nhom-sinh-vien_30.html)*.*

[*https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library*](https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library)