TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**CÀI ĐẶT THUẬT TOÁN XÉN TỈA ĐA GIÁC**

**BẰNG THUẬT TOÁN SUTHERLAND HODGMAN**

**GVHD: Ths. ĐOÀN VŨ THỊNH**

**SVTH: Đỗ Thị Thanh Thúy – MSSV: 59132551**

**Lớp: 59CNTT-1**

Khánh Hòa, tháng 01 năm 2020

# MỤC LỤC

[MỤC LỤC 1](#_Toc30079366)

[TÓM TẮT 2](#_Toc30079367)

[1. GIỚI THIỆU 3](#_Toc30079368)

[1.1. Thuật toán xén tỉa 4](#_Toc30079369)

[1.2. Thuật toán xén tỉa đoạn thẳng 5](#_Toc30079370)

[1.3.1 Khái niệm đa giác 8](#_Toc30079371)

[2. PHƯƠNG PHÁP NGHIÊN CỨU 15](#_Toc30079372)

[2.1 Cài đặt Dev – C++ và thư viện Graphics.h 15](#_Toc30079373)

[2.2 Nhập dữ liệu 16](#_Toc30079374)

[2.3. Các bước xén tỉa đa giác 17](#_Toc30079375)

[2.3.1. Trường hợp cắt bên trái của đa giác 17](#_Toc30079376)

[2.3.2. Trường hợp cắt bên phải 18](#_Toc30079377)

[2.3.3Trường hợp cắt bên trên 19](#_Toc30079378)

[2.3.4 Trường hợp cắt bên dưới 20](#_Toc30079379)

[2.3.5 Chương trình chính 22](#_Toc30079380)

[3. KẾT QUẢ 22](#_Toc30079381)

[3.1 Dev-C++ và thư viện Graphics.h 22](#_Toc30079382)

[3.2 Thuật toán xén tỉa đa giác 23](#_Toc30079383)

[4. KẾT LUẬN 24](#_Toc30079384)

# TÓM TẮT

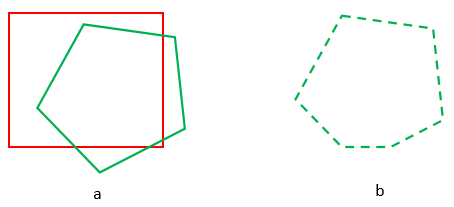
Đồ họa máy tính là một lĩnh vực của [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh) nghiên cứu về cơ sở [toán học](https://vi.wikipedia.org/wiki/To%C3%A1n_h%E1%BB%8Dc), các [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) cũng như các kĩ thuật để cho phép tạo, hiển thị và điều khiển hình ảnh trên màn hình [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh). Đồ họa máy tính có liên quan ít nhiều đến một số lĩnh vực như [đại số](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%A1i_s%E1%BB%91), [hình học giải tích](https://vi.wikipedia.org/wiki/H%C3%ACnh_h%E1%BB%8Dc_gi%E1%BA%A3i_t%C3%ADch), [hình học họa hình](https://vi.wikipedia.org/wiki/H%C3%ACnh_h%E1%BB%8Dc_h%E1%BB%8Da_h%C3%ACnh), [quang học](https://vi.wikipedia.org/wiki/Quang_h%E1%BB%8Dc),... và [kĩ thuật máy tính](https://vi.wikipedia.org/w/index.php?title=K%C4%A9_thu%E1%BA%ADt_m%C3%A1y_t%C3%ADnh&action=edit&redlink=1), đặc biệt là chế tạo [phần cứng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_c%E1%BB%A9ng) (các loại [màn hình](https://vi.wikipedia.org/wiki/M%C3%A0n_h%C3%ACnh), các thiết bị xuất, nhập, các [vỉ mạch đồ họa](https://vi.wikipedia.org/w/index.php?title=V%E1%BB%89_m%E1%BA%A1ch_%C4%91%E1%BB%93_h%E1%BB%8Da&action=edit&redlink=1)...). Kỹ thuật xén tỉa đa giác là một trong số rất nhiều thuật toán quan trọng của ngành đồ họa máy tính và các phần mềm thương mại hay các thiết bị xử lý đồ họa đều vận dụng các thuật toán này. Thuật toán xén tỉa đa giác với giải thuật Sutherland –Hodgman được cài đặt trong lần thực tập lần này giúp hiểu rõ hơn về một trong những thuật toán cơ bản của ngành đồ họa máy tính. Quy trình thực hiện được trải qua các bước từ thiết kế giao diện, cài đặt thuật toán, hiển thị kết quả đầu ra trên màn hình hiển thị đều được thực hiện trên môi trường C++ thông qua ứng dụng DevC/C++ có kết hợp với thư viện graphics.h.

Kết quả của việc cài đặt thuật toán đáp ứng được các yếu cầu đặt ra của đợt thực tập cơ sở lần này.

# 1. GIỚI THIỆU

Cắt đa giác là một hoạt động quan trọng mà máy tính, thực hiện mọi lúc. Một thuật toán cắt một đa giác khá phức tạp. Mỗi cạnh của đa giác phải được kiểm tra đối với từng cạnh của cửa sổ cắt, thường là một hình chữ nhật. Do đó, các cạnh mới có thể được thêm vào và các cạnh hiện có có thể bị loại bỏ, giữ lại hoặc chia. Nhiều đa giác có thể là kết quả của việc cắt một đa giác. Sau khi cắt, chúng ta có thể có một tập hợp các phân đoạn, phải được xử lý để tạo đa giác bị cắt. Công việc này đề xuất hai thuật toán mới: cắt đa giác trên cửa sổ hình chữ nhật và tái tạo đa giác từ một tập hợp các phân đoạn.

Ứng dụng phổ biến của giải thuật xén tỉa đa giác không còn xa lạ với mọi người , từ phần mềm thiết kế đồ họa trong xây dụng như Autocad (<https://www.autodesk.com/products/autocad/overview>), phần mềm thiết kế đồ họa trong phim ảnh như Adobe Illustrator ([https://www.adobe.com/sea/products/illustrator.html](https://www.adobe.com/sea/products/illustrator.html?sdid=YB1TGRSV&mv=search&s_kwcid=AL!3085!10!79096184450641!79096306250771&ef_id=WvwhngAAALJZuxMJ:20200111082307:s)) và một phần mềm khác rất quen thuộc với người dùng windows là Powerpoint (Hình 1.1).



Hình 1.1. Ví dụ minh họa của ứng dụng xén tỉa đa giác bằng phần mềm Powerpoint 2013.

Trong đó, hình 1.1.a thể hiện đa giác cần xén tỉa (màu xanh) và cửa sổ cắt (màu đỏ); hình 1.1.b biểu diển kết quả sau khi xén tỉa ( vùng nét đứt). Tùy thuộc vào từng ứng dụng cụ thể mà cửa sổ xén có thể có dạng là đa giác hay là đường cong khép kín. Để đơn giản trường hợp xén tỉa đa giác là trường hợp cụ thể của xén tỉa đoạn thẳng.

## 1.1. Thuật toán xén tỉa

Trước hết việc xác định các phần của hình ảnh bên trong và loại bỏ phần bên ngoài của một vùng không không gian xác định được gọi là thuật toán xén tỉa, hoặc đơn giản là xén tỉa. Trong đó:

Vùng mà đối tượng được xén tỉa được gọi là cửa sổ xén tỉa. Cửa sổ xén có thể là hình chữ nhật, hình tròn, cửa sổ lòi, cửa sổ lõm, tùy vào các chiến lượt xén tỉa khác nhau dẫn đến các thuật toán xén tỉa giống nhau. Cửa sổ xén tỉa là hình chữ nhật, và đối tượng xén tỉa là 1 đa giác. Sử dụng thuật toán xén tỉa để loại bỏ phần hình ảnh bên ngoài cửa sổ xén ta được phần hình ảnh của đối tượng nằm bên trong cửa sổ xén tỉa.

Đối tượng được xén tỉa có thể là đường thẳng, đa giác, hình tròn, ký tự hoặc đường cong không đều. Để xén tỉa được một đa giác trên ta phải xén tỉa đoạn thẳng.

## 1.2. Thuật toán xén tỉa đoạn thẳng

Trong [đồ họa máy tính](https://en.wikipedia.org/wiki/Computer_graphics), cắt đoạn thẳng là quá trình loại bỏ các dòng hoặc một phần các dòng bên ngoài một khu vực quan tâm. Trong đồ họa máy tính xén tỉa đoạn thẳng là quá trình bỏ đoạn thẳng hoặc một phần đoạn thẳng bên ngoài cửa sổ cắt. Thông thường đoạn thẳng hoặc một phần nằm ngoài cửa sổ cắt sẽ bị xóa. Về xén tỉa đoạn thẳng có 2 trường hợp:

Giả sử (x,y) là toạ độ của một điểm, vậy điểm đó được hiển thị khi thoả mãn:

Xwmin <= x <= Xwmax

Ywmin <= y <= Ywmax

(1) Các đoạn thẳng không cắt cửa sổ xén tỉa: hoặc nằm trong hoàn toàn, hoặc nằm ngoài hoàn toàn cửa sổ cắt (Hình 1.2 trái ).

(2) Cửa sổ xén tỉa phân chia đoạn thẳng thành 2 phần: điểm cắt nằm trong và ngoài cửa sổ xén tỉa (Hình 1.2 phải).

|  |  |
| --- | --- |
| (1) | (2) |

Hình 1.2 Ví dụ minh họa các trường hợp đoạn thẳng được xén tỉa

Trường hợp cắt cửa sổ bên trái:

Trường hợp cắt cửa sổ bên phải:

Trường hợp cắt cửa sổ bên trên:

Trường hợp cắt cửa sổ bên dưới

Cụ thể, cho cửa sổ cắt tỉa hình chữ nhật có tạo độ A(50,10); B(80,10); C(80,40); D(50,40), mã vùng dành cho các điểm cuối của đoạn thẳng có P1(40,15), P2(75,15) được xác định như sau:

Ta có: và

xwmin=50; xwmax=80; ywmin=10; ywmax=40;

Bit code (ABRL) P1: 0001 #0

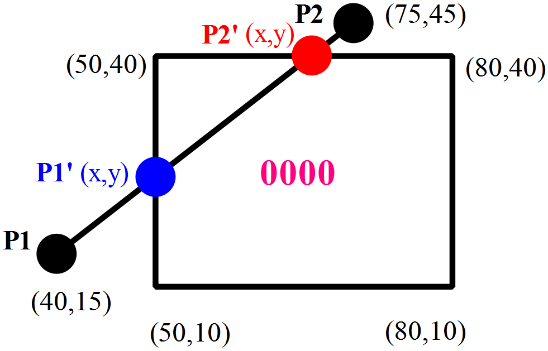
Bit code (ABRL) P2: 1000 #0

AND(P1,P2)=0000 nên P1P2 thỏa điều kiện cần xén

Xác định điểm cắt thứ nhất P1’(50,y):

Xác định điểm cắt thứ hai P2’(x,40):

Xác định điểm cắt thứ hai P2’(x,40):

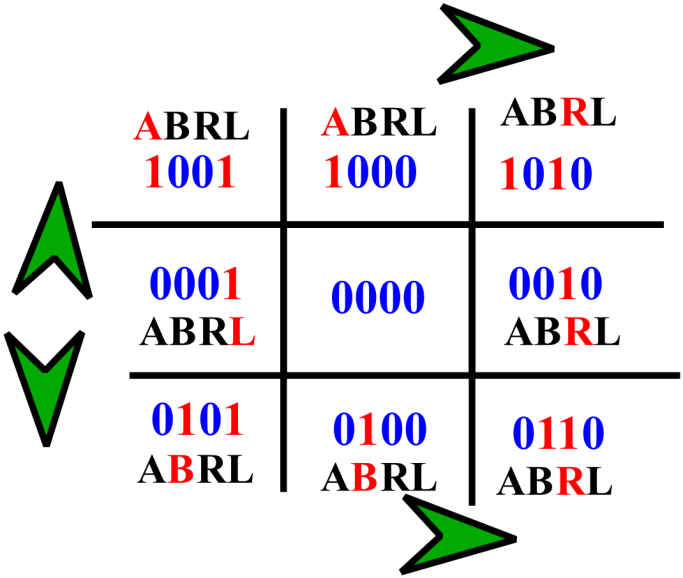
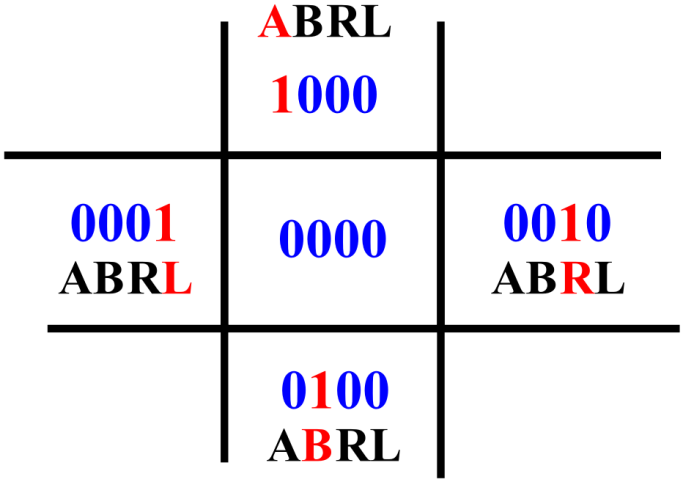


Hình 1.3. Xác định tọa độ các điểm cắt của đoạn thẳng P1 P2 và cửa sổ cắt

Có rất nhiều thuật toán để cắt đoạn thẳng, trong đó phổ biến nhất là “Cohen – Sutherland” và “Liang-Barsky”. Ra đời sớm nhất và thông dụng nhất là Thuật toán Cohen – Sutherland 1967

(<https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm>). Thuật toán được phát triển trong quá trình mô phỏng chuyến bay của Daniel Cohen và Ivan Sutherland. Thuật toán Cohen- Sutherland (được đặt theo tên của Daniel Cohen và Ivan Sutherland) là một thuật toán cắt dòng. Thuật toán chia một không gian 2D thành 9 vùng, trong đó chỉ có phần giữa (khung nhìn) được hiển thị.

Mã vùng được xác định theo 9 vùng của mặt phẳng mà các điểm cuối nằm vào đó. Một bít được cài đặt true (1) hoặc false (0) (Hình 1.4).



Hình 1.4. Mã vùng 4 bit cho mỗi điểm của đoạn thẳng ABRL – Above Below – Right - Left

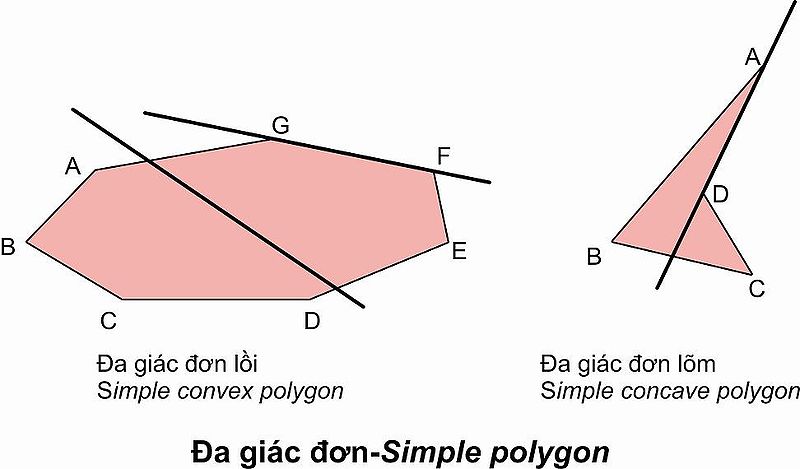
Sau thuật toán Cohen- Sutherland là  Thuật toán Cyrus THER Beck 1978 (<https://en.wikipedia.org/wiki/Cyrus%E2%80%93Beck_algorithm>) bởi [Mike Cyrus](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213" \l "!) và [Jay Beck](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213#!) được đơn giản hóa, được tối ưu hóa cho một cửa sổ clip hình chữ nhật. Thuật toán Cyrus THER Beck chủ yếu dành cho việc cắt một dòng ở dạng tham số đối với đa giác lồi ở 2 chiều hoặc đối với đa diện lồi trong 3 chiều. Tiếp đến là Thuật toán Liangky Barsky 1984(<https://en.wikipedia.org/wiki/Liang%E2%80%93Barsky_algorithm>) bởi Liang, You-Dong và Barsky, Brian A. Sử dụng phương trình tham số của một dòng và bất đẳng thức mô tả phạm vi của hộp cắt để xác định các giao điểm giữa dòng và hộp 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 trinh Sutherland, nhưng CohenTHER Sutherland chấp nhận và từ chối nhanh hơn nhiều, vì vậy nó nên được xem xét thay vào đó nếu hầu hết các dòng bạn cần quay sẽ hoàn toàn vào hoặc ra khỏi [cửa sổ clip](https://en.wikipedia.org/wiki/Clip_window).

Ngoài ra còn có một số thuật toán khác như Fast –clipping, Skala. Các thuật toán xén tỉa này cũng được cải thiện về độ chính xác và thời gian xử lí được cải thiện đáng kể ***(***[*https://en.wikipedia.org/wiki/Category:Polygon\_clipping\_algorithms*](https://en.wikipedia.org/wiki/Category:Polygon_clipping_algorithms)).

**1.3.Thuật toán xén tỉa đa giác**

### 1.3.1 Khái niệm đa giác

Đa giác lồi: toàn bộ đa giác nằm về một phía của đường thẳng chứa cạnh bất kì nào của đa giác. Khi đó, đoạn thẳng nối tiếp hai điểm bất kì nào của đa giác đều nằm hoàn toàn trong đa giác.

**

*Hình 1.5. Đa giác lồi và đa giác lõm*

**Qui ước xác định chiều các điểm của đa giác theo chiều kim đồng hồ như sau**: Theo qui ước: một đa giác với các đỉnh P1, .....,PN (các cạnh là Pi-1 và PNP1) được gọi là theo hướng dương nếu các hình theo thứ tự đã cho tạo thành mạch ngược chiều kim đồng hồ.

Nếu bàn tay dọc theo bất kỳ cạnh Pi-1 Pi hoặc PN P1 cũng chỉ về bên trong đa giác.

Quy tắc khác:

Tính tổng các cạnh của đa giác: (X2-X1)(Y2+Y1)

Nếu kết quả cho số dương thì chiều của đa giác thuận theo kim đồng hồ và ngược lại

point[0] = (5,0) edge[0]: (6-5)(4+0) = 4

point[1] = (6,4) edge[1]: (4-6)(5+4) = -18

point[2] = (4,5) edge[2]: (1-4)(5+5) = -30

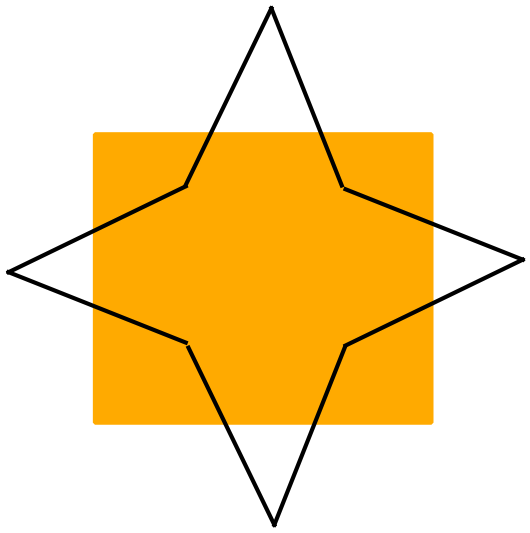
point[3] = (1,5) edge[3]: (1-1)(0+5) = 0

point[4] = (1,0) edge[4]: (5-1)(0+0) = 0

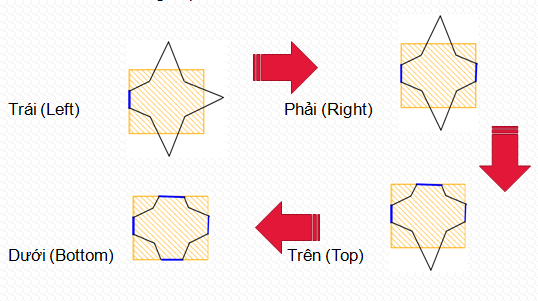
Total: -44 (đa giác có chiều âm)

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

Cho P1,......,PN là danh sách các đỉnh của đa giác. Cho cửa sổ cắt tỉa ABCD. Thứ tự các trường hợp sẽ được xén như sau:



*Hình 1.6. Hình cửa sổ cắt và đa giác cần cắt*



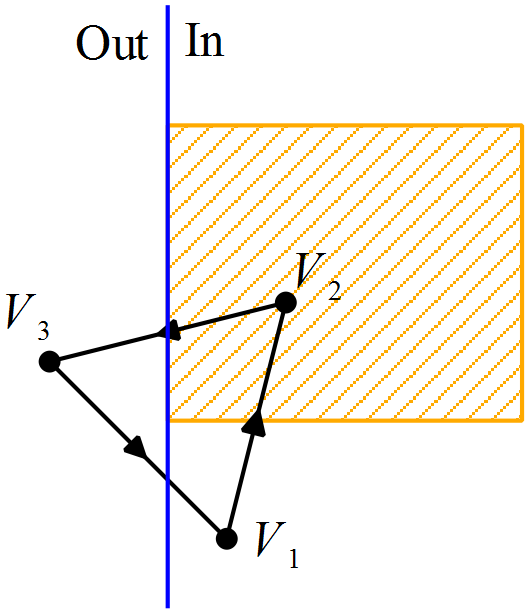
*Hình 1.7. Minh họa thứ tự xén*

Mỗi lần xét như vậy là phụ thuộc 4 trường hợp như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Output (Out-In): v’1v2 | Output (In-Out): v’1 | Output (In-In): v2 | Output: NULL |

Hình 1.8 Vị trí của đoạn thẳng so với cửa sổ

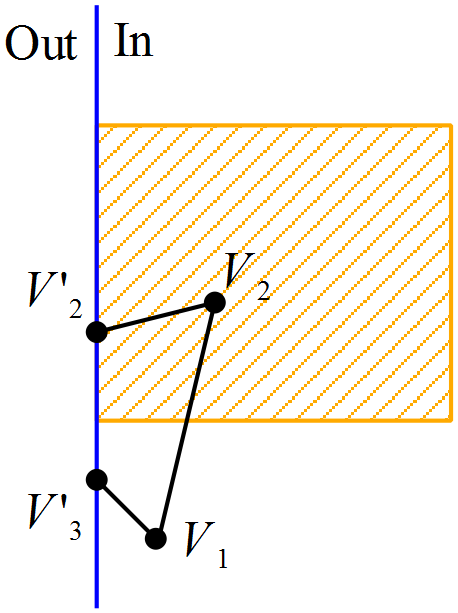
Cho đa giác dương sau và cửa sổ cắt, hãy xác định các đỉnh cắt:



Hình 1.9 Vị trí cạnh cần cắt của đa giác so với cạnh của cửa sổ cắt

**Bước 0:** Áp dụng quy tắc bàn tay trái cho đa giác dương để xác định chiều của các cạnh

**Bước 1:** Cắt bên trái

**

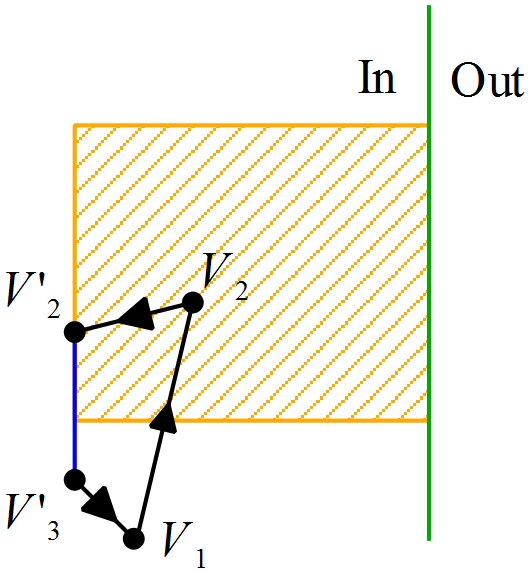
Hình 1.10. Vị trí của cạnh cần cắt của đa giác so với cạnh trái của cửa sổ cắt

Xét cạnh V1-V2 (In-In): V2

Xét cạnh V2-V3 (In-Out): V’2

Xét cạnh V3-V1(Out-In): V3-V3’

**Bước 2**: Cắt bên phải



Hình 1.11. Vị trí của cạnh cần cắt của đa giác so với cạnh phải của cửa sổ cắt

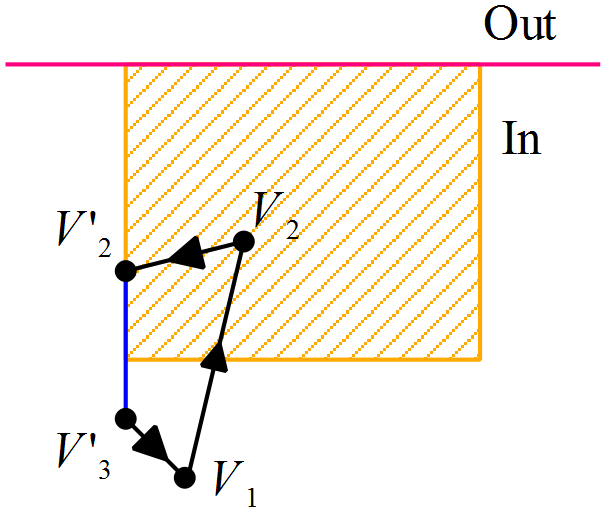
Xét cạnh V1-V2 (Out-In): V2

Xét cạnh V2-V’2 (In-Out): V’2

Xét cạnh V’2-V’3 (In-In): V’3

Xét cạnh V’3-V1(In-In): V1

**Bước 3:** Cắt bên trên

**

Hình 1.12. Vị trí của cạnh cần cắt của đa giác so với cạnh bên trên của cửa sổ cắt

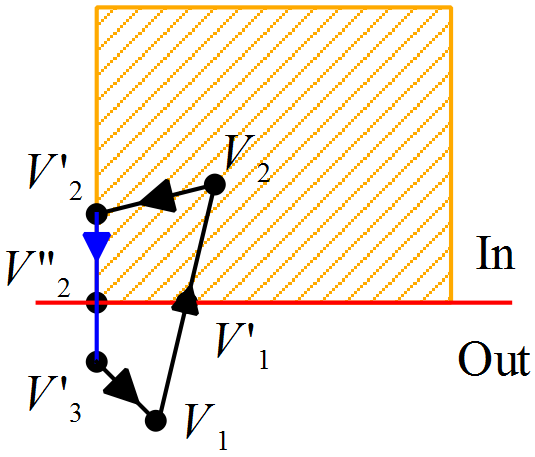
Xét cạnh V1-V2 (In-In): V2

Xét cạnh V2-V’2 (In-In): V’2

Xét cạnh V’2-V’3 (In-In): V’’3

Xét cạnh V’3-V1(Out-Out): NULL

**Bước 4:** Cắt bên dưới

**

*Hình 1.13 Vị trí của cạnh cần cắt của đa giác so với cạnh bên dưới của cửa sổ cắt.*

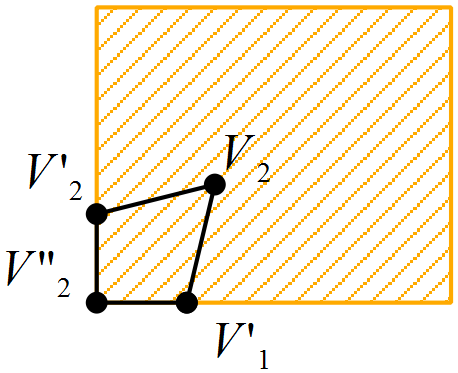
*Xét cạnh V1-V2 (Out-In): V’1V2*

*Xét cạnh V2-V’2 (In-In): V’2*

*Xét cạnh V’2-V’3 (In-Out): V’’2*

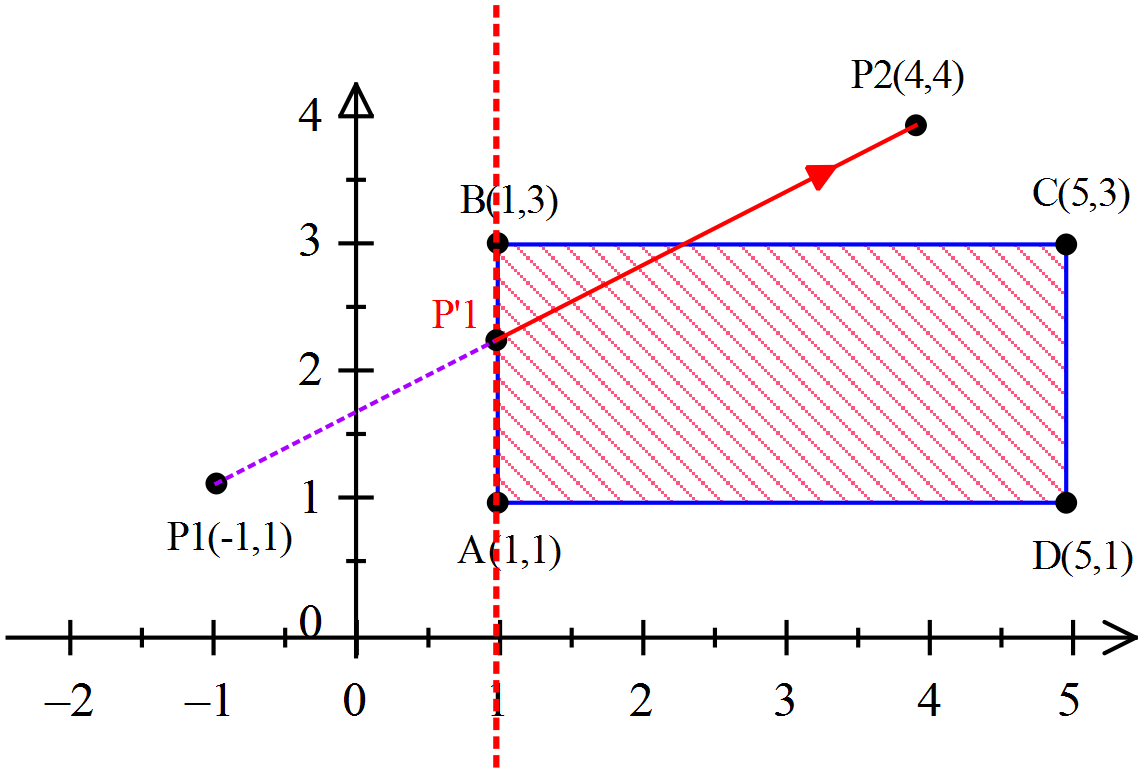
*Xét cạnh V’3-V1(Out-Out): NULL*

**Bước 5:** Kết quả sau khi thực hiện thao tác cắt

******

Hình 1.14 Kết quả của thuật toán Sutherland –Hodgman sau khi cắt.

Ví dụ xét đoạn thẳng P1, P2 và cửa sổ cắt có tọa độ như hình 1.11. Xác định tọa độ điểm giao nhau giữa đoạn thẳng và cửa sổ cắt.



Hình 1.15 Ví dụ minh họa xác định điểm giao nhau giữa cửa sổ cắt với đoạn thẳng

**Bước 1:**

Cạnh AB: Chiều Out-IN

Điểm P1 nằm bên trái AB

Điểm P2 nằm bên phải AB

Tìm giao điểm P’1 thuộc đường thằng P1P2

P’1(1;11/5)

**Bước 2:**

Cạnh DC: Chiều In-In

Điểm P’1: nằm bên trái DC

Điểm P2: nằm bên trái DC

**Bước 3:**

Cạnh BC: Chiều In – Out

Điểm P’1: nằm bên trong BC

Điểm P2: nằm bên ngoài BC

Tìm giao điểm P’2  thuộc đường thằng P’1P2

P’2(7/3,3)

**Bước 4:**

Cạnh AD: Chiều In-In.

Điểm P’1: nằm bên trong AD.

Điểm P’2: nằm bên trong AD.

Thuật toán Sutherland Hodgman 1947 do Ivan Sutherland và Gary W. Hodgman đề ra dùng để xén tỉa đa giac, thuật toán này hoạt động bằng cách lần lượt mở rộng từng cạnh của cửa sổ xén và chỉ chọn các đỉnh từ đa giác 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 cần xén. Tiếp theo, một bên của cửa sổ xén được mở rộng vô hạn theo cả hai hướng, các đỉnh từ danh sách đầu vào được chèn vào danh sách đầu ra nếu chúng nằm ở phía có thể nhìn thấy cạnh của cửa sổ xén mở rộng và các đỉnh mới được thêm vào đầu danh sách đầu ra khi chúng giao với cạnh của cửa sổ xén.

Quá trình này được lặp đi lặp lại cho mỗi cạnh của cửa sổ, sử dụng danh sách đầu ra từ một giai đoạn trước ( sau khi xén lần trước ) làm danh sách đầu vào cho lần tiếp theo. Khi tất cả các cạnh của cửa sổ xén đã được xử lý, danh sách các đỉnh được tạo cuối cùng sẽ được xác định một đa giác mới.

**DevC/C++ và thư viện Graphics.h**

Dev-C++ (Dev-Cpp,  phiên bản  hiện tại  là  4.9.9.2) là một bộ công cụ phát triển tích hợp (IDE Integrated Development Environment) các ứng dụng C/C++ thuộc dạng mã nguồn mở. DevCpp dựa  trên  trình biên dịch mã nguồn mở MinGW  (Minimalist GNU\* for Windows, http://www.mingw.org/). MinGW sử dụng GCC (the GNU g++ compiler collection) sử dụng cho cả hệ thống Windows và Linux. Hiện nay DevCpp  là công cụ phát triển các ứng dụng C/C++ được sử dụng rộng  rãi để dạy về lập trình cũng như để phát  triển các ứng dụng mã nguồn mở.  Mặc  dù  không  có  nhiều  tính  năng  cao  cấp  như  các  công  cụ  IDE  thương mại  khác  (Visual Studio của Microsoft) nhưng DevCpp vẫn là một công cụ rất thích hợp trong môi trường giáo dục ở bậc đại học khi dạy và học các ngôn ngữ C/C++.

Do thư viện đồ họa graphics và winbgim không được tích hợp sẵn trong thư viện của phần mềm Devc++ do đó cần phải thêm thư viện này vào công cụ DevC/C++**.**

Thư viện **Graphics.h** hay tên chính xác và đầy đủ của nó là Borland Graphics Interface – còn được biết đến với tên gọi BGI – là một thư viện đồ họa rất phổ biến trên DOS và các máy tính chạy hệ điều hành Windows thời kì đầu như Windows 95, Windows 98. Thư viện này cung cấp cho người dùng 2 file: graphics.h và graphics.lib để có thể sử dụng được với ngôn ngữ C/C++ cũng như module graph nếu người dùng sử dụng ngôn ngữ Pascal. Bộ thư viện này đi kèm với IDE Borland C++ 3.1 (1992). Nó chứa một số hàm vẽ ảnh và cách tạo ảnh chuyển động, in chữ. Một trong những điểm mạnh của thư viện này là việc khởi tạo cũng như sử dụng rất đơn giản, vì vậy mặc dù ra đời rất lâu nhưng hiện tại vẫn có rất nhiều trường đại học sử dụng cho mục đích giảng dạy.

# 2. PHƯƠNG PHÁP NGHIÊN CỨU

## 2.1 Cài đặt Dev – C++ và thư viện Graphics.h

DevC/C++ phiên bản 5.9.2 được tải về và cài đặt theo đường dẫn sau: [*https://sourceforge.net/projects/orwelldevcpp/*](https://sourceforge.net/projects/orwelldevcpp/)*,* sau đó thư viện graphics.h được tải về và cài đặt theo các bước sau:

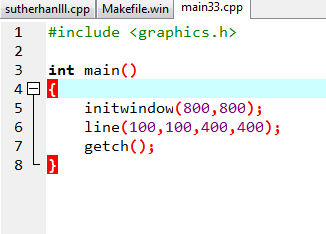
**Bước 1:** Copy tập tin 6-ConsoleAppGraphics.template và ConsoleApp\_cpp\_graph.txt vào thư mục C:\Program Files\Dev-Cpp\Templates

**Bước 2:** Copy graphics.h và winbgim.h vào thư mục C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\include.

**Bước 3:** Copy libbgi.a vào thư mục C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\lib.

**Bước 4:** Chọn trình biên dịch là**TDM-GCC 32 bit hay 64 bit** ứng với phiên bản Dev C++ đã cài đặt.

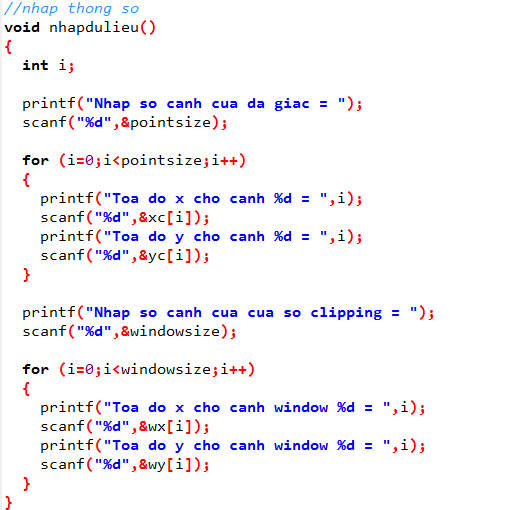
**Bước 5:** Tạo một Project mới bằng cách chọn File → New → Project → Console Graphics Application.



Hinh 2.1 Thử nghiệm thư viện với thao tác vẽ đường thẳng

## 2.2 Nhập dữ liệu

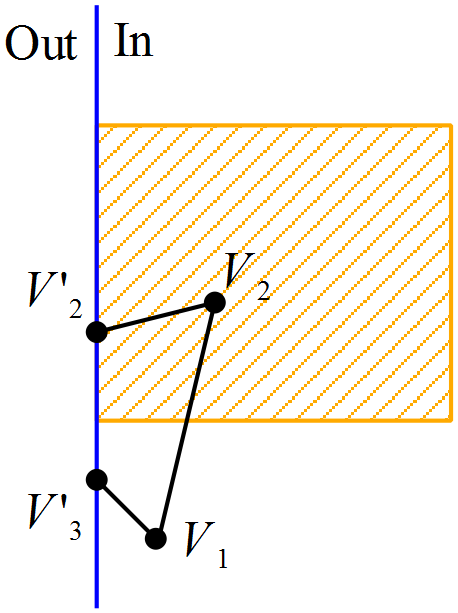
Dữ liệu đầu vào bao gồm 2 tham số: (1) cửa sổ cắt và (2) đa giác cần cắt. Dữ liệu này được nhập từ bàn phím thông qua giao diện hỏi và đáp như hình 1.7. Danh sách các đỉnh của đa giác được lưu trữ trong mảng một chiều có tên xc và yc, các đỉnh của cửa sổ cắt được lưu trữ trong danh sách có tên wx và wy.



Hình 2.2 Các bước nhập dữ liệu từ bàn phím

## 2.3. Các bước xén tỉa đa giác

### 2.3.1. Trường hợp cắt bên trái của đa giác

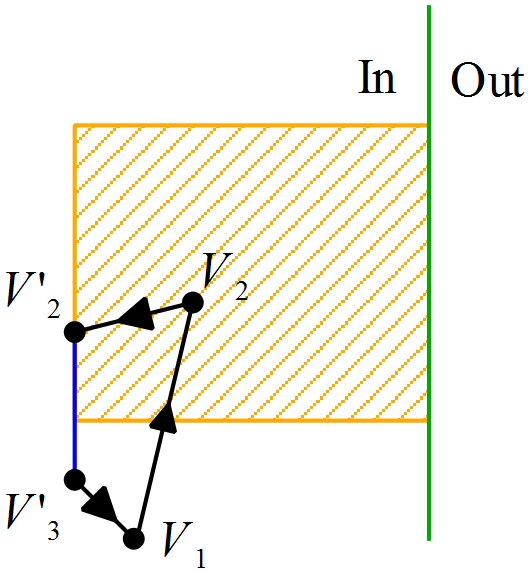
******

Hình 2.3 Vị trí cạnh cần cắt của đa giác so với cạnh trái của cửa sổ cắt

Có 4 trường hợp ứng với vị trí của cạnh đa giác so với cạnh trái của cửa sổ cắt:

|  |  |
| --- | --- |
| **TH1:** Đường thẳng có chiều đi từ ngoài vào trong | **outintrai.PNG** |
| **TH2:** Đường thẳng có chiều đi từ trong ra ngoài | **inout trai.PNG** |
| **TH3:** Đường thẳng nằm hoàn toàn bên trong | **inintrai.PNG** |
| **TH4:** Nằm hoàn toàn bên ngoài | Không có điểm bất kì nào thêm vào danh sách mới |

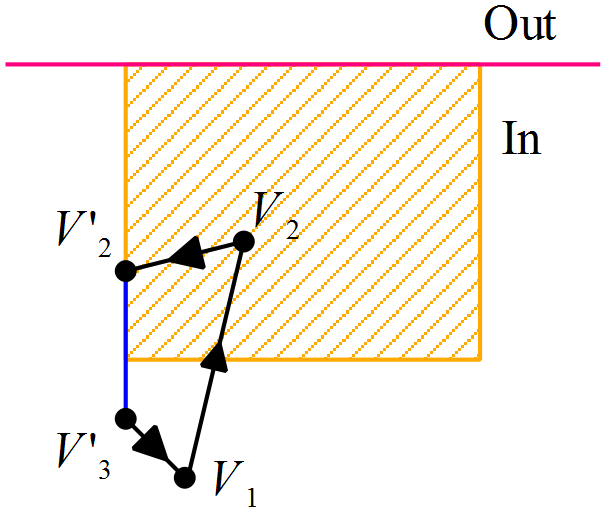
### 2.3.2. Trường hợp cắt bên phải

****

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

|  |  |
| --- | --- |
| **TH1**: Đường thẳng có chiều đi từ ngoài vào trong | outin phai.PNG |
| **TH2**: Đường thẳng có chiều đi trong ra ngoài | *inout phải.PNG* |
| **TH3**: Đường thẳng nằm hoàn toàn bên trong | *ininphai.PNG* |
| **TH4**: Nằm hoàn toàn bên ngoài | Không có điểm bất kì nào thêm vào danh sách mới*.* |

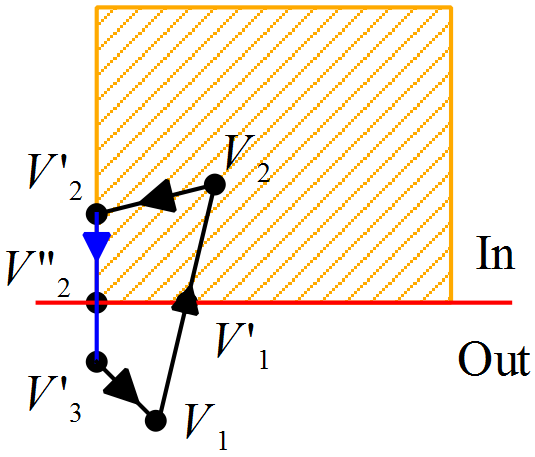
### 2.3.3Trường hợp cắt bên trên

****

Hình 2.5Vị trí các cạnh cần cắt của đa giác so với cạnh phía trên của cửa sổ cắt

|  |  |
| --- | --- |
| **TH1**: Đường thẳng có chiều đi từ ngoài vào trong | outin trên.PNG |
| **TH2**: Đường thẳng có chiều đi trong ra ngoài | inout trên.PNG |
| **TH3**: Đường thẳng có nằm hoàn toàn bên trong | inin trên.PNG |
| **TH4**: Nằm hoàn toàn bên ngoài | Không có điểm bất kì nào thêm vào danh sách mới. |

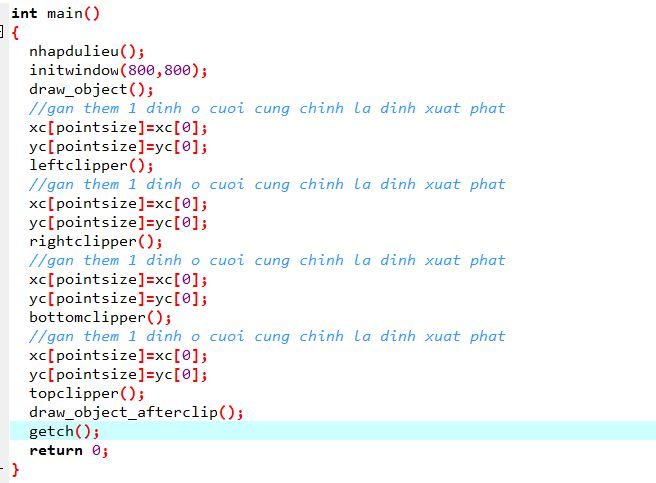
### 2.3.4 Trường hợp cắt bên dưới

****

Hình 2.6 Vị trí các cạnh cần cần của đa giác so với cạnh bên dưới cửa sổ cắt

|  |  |
| --- | --- |
| **TH1**: Đường thẳng có chiều đi từ ngoài vào trong. | outin duoi.PNG |
| **TH2**: Đường thẳng có chiều đi từ trong ra ngoài. | inout duoi.PNG |
| **TH3**: Đường thẳng nằm hoàn toàn bên trong. | ininduoi.PNG |
| **TH4**: Nằm hoàn toàn bên ngoài. | Không có điểm bất kì nào thêm vào danh sách mới. |

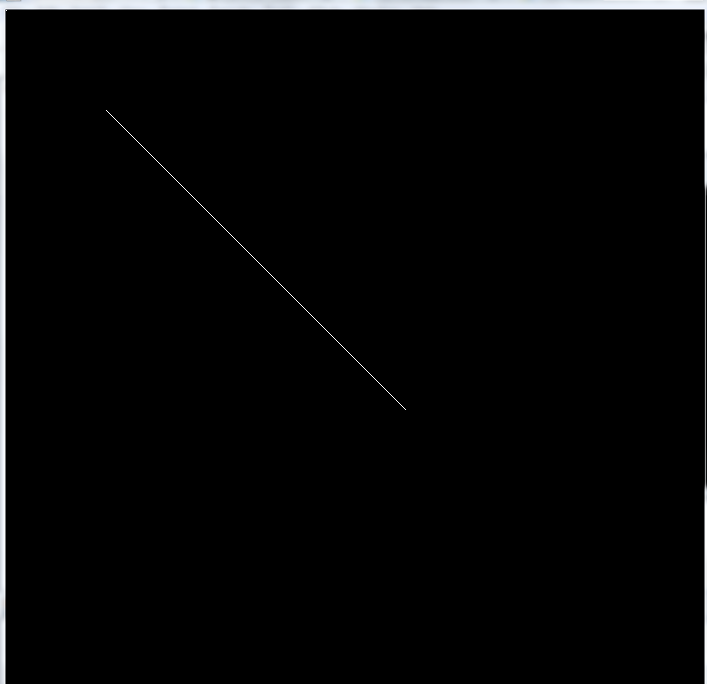
### 2.3.5 Chương trình chính

****

# 3. KẾT QUẢ

## 3.1 Dev-C++ và thư viện Graphics.h

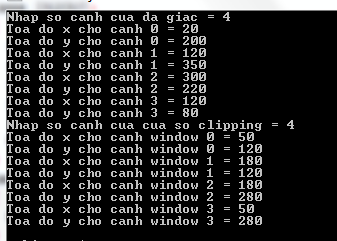
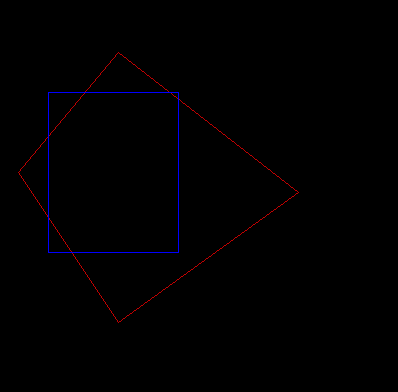
Sau khi cài đặt phần mềm Dev-C++ và thư viện graphics.h đối với đoạn code mẫu ở phần trên, sau khi chạy chương trình thì tạo ra một cửa sổ có kích thước 800x800 và một đoạn thẳng như hình dưới (Hình 3.1).



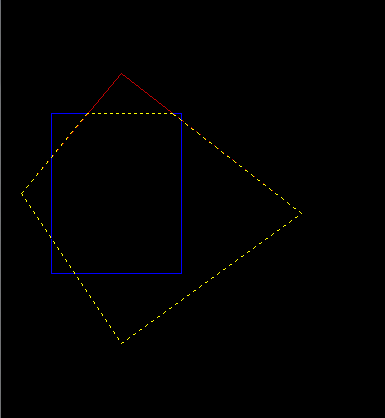
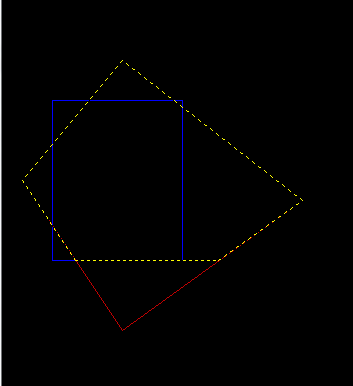
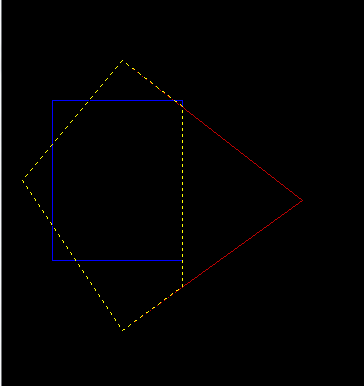
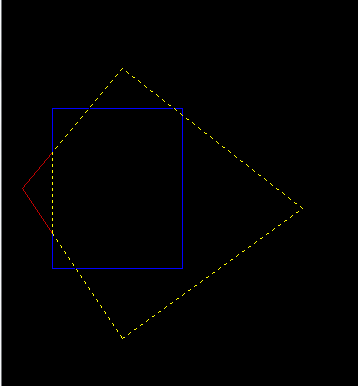
Hình 3.1 Kết quả sau khi chạy đoạn chương trình chính

## 3.2 Thuật toán xén tỉa đa giác

Dữ liệu nhập từ bàn phím cho 2 tham số: cửa sổ cắt và đa giác cần xén tỉa ( Hình 3.2)

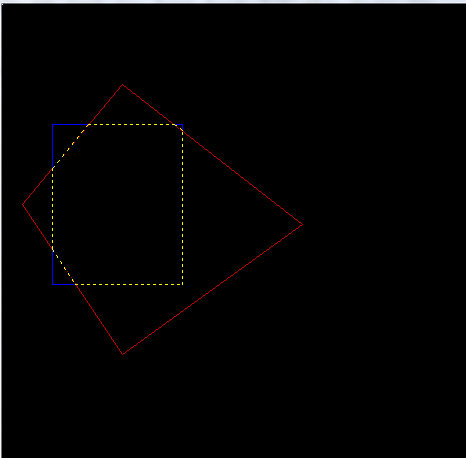
Hình 3.2 Nhập dữ liệu từ bàn phím và đa giác cần được xén tỉa



(a) (b) (c) (d)

Hình 3.3 Các trường hợp xén tỉa của đa giác

Hình 3.3 có cửa sổ cắt màu xanh và đa giác màu vàng. Trong đó, Hình 3.3.a có giao điểm của đa giác và cửa sổ cắt cạnh bên trái là đường màu đỏ. Hình 3.3.b có giao điểm của đa giác và cửa sổ cắt cạnh bên phải là đường màu đỏ. Hình 3.3.c có giao điểm của đa giác và cửa sổ cắt cạnh dưới là đường màu đỏ. Hình 3.3.d giao điểm của đa giác và cửa sổ cắt cạnh bên trên là đường màu đỏ.



Hình 3.4. Đa giác sau khi cắt xén

# 4. KẾT LUẬN

Với đề tài cắt tỉa đa giác bằng thuật toán Sutherland -Hodgman thì thuật toán xén tỉa đa giác sử dụng thuật toán Sutherland – Hodgman được cài đặt theo các yêu cầu của đề tài thực tập cơ sở. Việc nhập dữ liệu được thông qua bàn phím. Các trường hợp xén tỉa đa giác đều được mô hình hóa và cài đặt, trong khuôn khỗ báo cáo này, hạn chế của thuật toán Sutherland –Hodgman đối với đa giác không lồi có đỉnh lõm nằm ngoài cửa sổ xén đã được chỉ ra những chưa thể giải quyết trong đợt thực tập này.