TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**



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

**BẰNG THUẬT TOÁN SUTHERLAND HOGMAN**

**GVHD : Ths. ĐOÀN VŨ THỊNH**

**SVTH : Huỳnh Phạm Thúy Hồng**

**MSSV : 59130833**

**Lớp : 59.CNTT-1**

Khánh Hòa, tháng 01 năm 2020

**MỤC LỤC**

[**TÓM TẮT 1**](#_Toc30073870)

[**1. GIỚI THIỆU 2**](#_Toc30073871)

[**1.1. Các thuật toán về xén tỉa đoạn thẳng 2**](#_Toc30073872)

[**1.2. Thuật toán về xén tỉa đa giác 5**](#_Toc30073873)

[**1.3. DevC/C++ và thư viện grpahics.h 6**](#_Toc30073874)

[**2. PHƯƠNG PHÁP NGHIÊN CỨU 7**](#_Toc30073875)

[**2.1. Cài đặt Dev C++ và Graphics.h 7**](#_Toc30073876)

[**2.2. Nhập dữ liệu 8**](#_Toc30073877)

[**2.3. Các bước xén tỉa đa giác 8**](#_Toc30073878)

[2.3.1. Trường hợp cắt cạnh bên trái 9](#_Toc30073879)

[2.3.2. Trường hợp cắt cạnh bên phải 9](#_Toc30073880)

[2.3.3. Trường hợp cắt cạnh bên trên 10](#_Toc30073881)

[2.3.4. Trường hợp cắt cạnh bên dưới 11](#_Toc30073882)

[**3. KẾT QUẢ 11**](#_Toc30073883)

[**4. KẾT LUẬN 12**](#_Toc30073884)

# TÓM TẮT

Đề tài được chia làm bốn phần: Giới thiệu, phương pháp, kết quả và kết luận của thuật toán xén tỉa đa giác bằng thuật toán Sutherland-Hodgman.

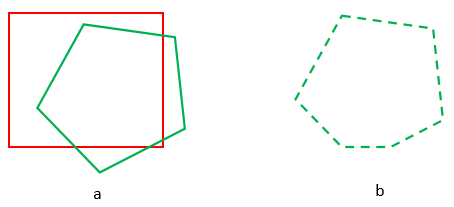
Trong phần giới thiệu nêu ra các ứng dụng phổ biến của thuật toán được sử dụng trên các phần mềm và tìm hiểu sơ lược về các thuật toán xén tỉa đoạn thẳng và đa giác. Từ đó tìm hiểu về phần mềm Dev C++ và thư viện Graphics.h.

Trong phần phương pháp, chúng ta biết được cách cài đặt phần mềm Dev C++ và thư viện Graphics.h. Sau đó, nhập dữ liệu từ phím để thực hiện xén tỉa đa giác bằng các thuật toán.

Trong phần kết quả sẽ hiển thị ra đa giác đã cắt xén.

# 1. GIỚI THIỆU

Thuật ngữ xén tỉa được sử dụng để mô tả quá trình tìm phần giao nhau của đoạn thẳng (hay đa giác) với đa giác khác. Nó có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm phổ biến hiện nay. Ứng dụng phổ biến của giải thuật xén tỉa đa giác trong xây dựng như Autocad *(*[*https://www.autodesk.com/products/autocad/overview*](https://www.autodesk.com/products/autocad/overview)*),* 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 PowerPoint2013 (Hình 1.1).

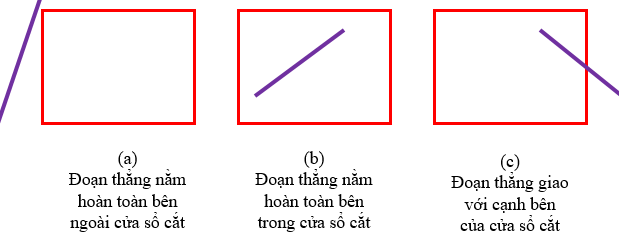


Hình 1.1. Ví dụ minh họa của ứng dụng xén tỉa đa giác bằng phần mềm Powerpoint 2013

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. Các thuật toán về xén tỉa đoạn thẳng

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. Xén tỉa đoạn thẳng có 2 trường hợp: 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.a và Hình 1.2.b), cửa sổ xén tỉa phân chia đoạn thẳng thành hai phần (điểm cắt nằm trong và ngoài cửa sổ xén tỉa (Hình 1.2.c).



Hình 1.2. Các trường hợp đoạn thẳng so với cửa sổ cắt

|  |  |
| --- | --- |
| **TH1: Trường hợp cắt cửa sổ bên trái** |  |
| **TH2: Trường hợp cắt cửa sổ bên phải** |  |
| **TH3: Trường hợp cắt cửa sổ bên trên** |  |
| **TH4: 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;

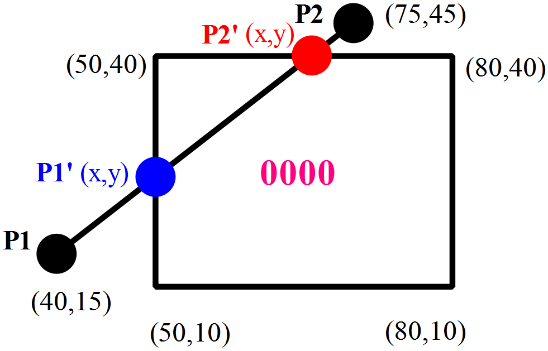
Bit code (ABRL) P1: 0001 #0

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):



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

Thuật toán Cohen-Sutherland là một thuật toán đồ họa máy tính được sử dụng để cắt dòng. Thuật toán chia không gian hai chiều thành 9 vùng và sau đó xác định hiệu quả các đường và các phần của đường có thể nhìn thấy ở khung nhìn.Thuật toán được phát triển vào năm 1967 trong quá trình mô phỏng chuyến bay của Daniel Cohen và Ivan Sutherland. Hạn chế của thuật toán cắt dòng Cohen Sutherland là thuật toán chỉ áp dụng cho các cửa sổ hình chữ nhật và không áp dụng cho các cửa sổ hình lồi khác.

Thuật toán Cyrus-beck được phát triển vào năm 1978 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#!). Đây là thuật toán cắt đoạn thẳng tổng quát sử dụng với cửa sổ cắt đa giác lồi, không giống như [thuật toán Cohen- Sutherland](https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm), chỉ có thể được sử dụng trên một khu vực cắt hình chữ nhật. (<https://en.wikipedia.org/wiki/Cyrus%E2%80%93Beck_algorithm>)

Thuật toán Liang-Barsky (1983) được phát triển bởi Liang, You-Dong và Barsky, Brian A. Thuật toán này sử dụng phương trình tham số của một dòng và bất đẳng thức mô tả phạm vi của cửa sổ cắt để xác định các giao điểm giữa đoạn thẳng và cửa sổ cắt. Thuật toán này hiệu quả hơn so với Cohen-sutherland, nhưng Cohen-sutherland xác định vị trí tương đối của đoàn thẳng so với vùng nhanh hơn nhiều, còn thuật toán này xem xét các đoạn thẳng cần xén tỉa sẽ hoàn toàn vào hoặc ra khỏi của sổ cắt. Thuật toán Liang-Barsky hiệu quả hơn phiên bản Cyrus-Beck vì thử nghiệm loại bỏ tầm thường bổ sung có thể tránh tính toán cả bốn giá trị tham số cho các đường không giao nhau với hình chữ nhật clip. *(*[*https://stackoverflow.com/questions/48352036/how-can-i-measure-the-overlap-between-a-line-and-a-rectangle*](https://stackoverflow.com/questions/48352036/how-can-i-measure-the-overlap-between-a-line-and-a-rectangle)*)*

Thuật toán Nicholl-Lee-Nicholl được phát triển vào năm 1987 bởi Tina M.Nicholl, DT Lee, Robin A. Nicholl. Đây là thuật toán cắt dòng nhanh làm giảm khả năng cắt một đoạn thẳng đơn nhiều lần, như có thể xảy ra trong [thuật toán Cohen-Sutherland](https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm).([*http://what-when-how.com/computer-graphics-and-geometric-modeling/clipping-basic-computer-graphics-part-3/*](http://what-when-how.com/computer-graphics-and-geometric-modeling/clipping-basic-computer-graphics-part-3/)*)*

Ngoài ra còn có một số thuật toán khác như Fast Clipping ,O(lg N) Algorithm, Skala, See also, References. Có rất nhiều thuật toán để cắt đoạn thẳng, trong đó phổ biến nhất là “Cohen – Sutherland” và “Liang-Barsky”.

***(***[*https://en.wikipedia.org/wiki/Category:Polygon\_clipping\_algorithms*](https://en.wikipedia.org/wiki/Category:Polygon_clipping_algorithms)*)*

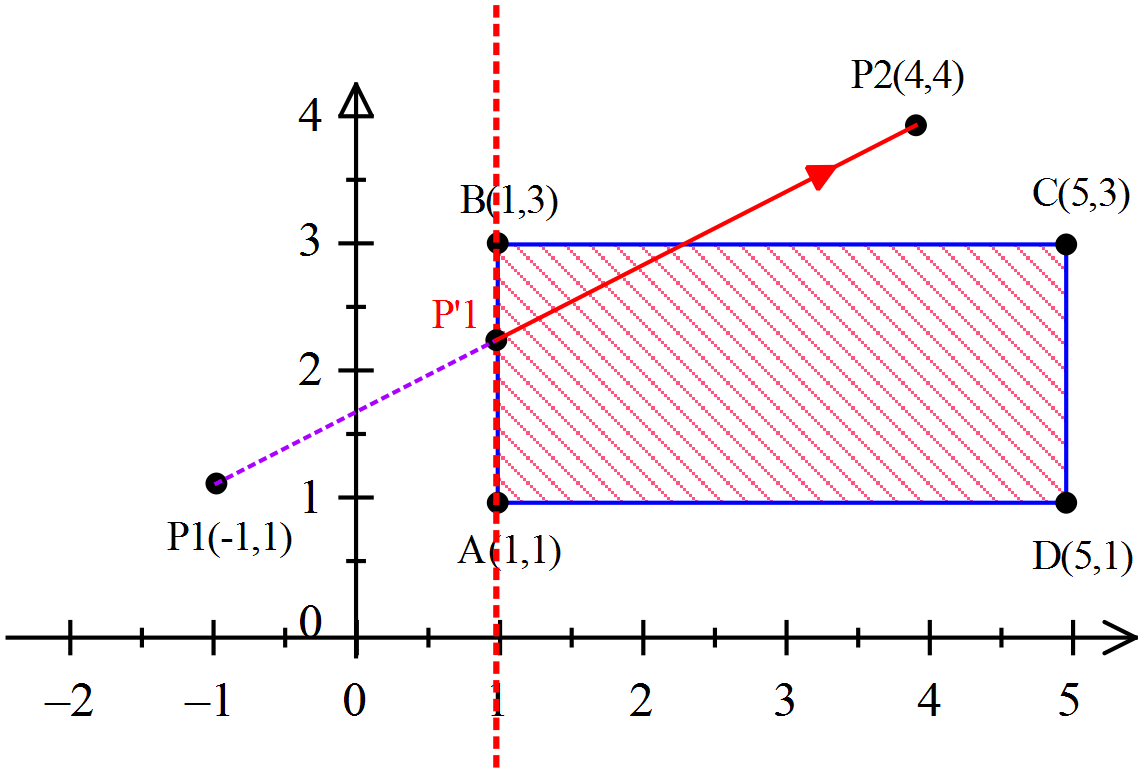
## 1.2. Thuật toán về xén tỉa đa giác

Có một số thuật toán xén tỉa đa giác nổi tiếng từng có những điểm mạnh và điểm yếu của nó. Một trong những thuật toán lâu đời nhất (từ năm 1974) được gọi là thuật toán Sutherland-Hodgman. Về cơ bản, thuật toán xén tỉa đa giác là trường hợp mở rộng của xén tỉa đoạn thẳng. Do đó, với mỗi đoạn thẳng (cạnh của đa giác) được xét trong 4 trường hợp (các cạnh của cửa sổ cắt: trái, phải, trên, dưới). 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 1.4. Vị trí của đoạn thẳng so với cửa sổ cắt

Xét đoạn thẳng P1, P2 và cửa sổ cắt có tọa độ như hình 1.5. Xác định tọa độ điểm giao nhau giữa đoạn thẳng và cửa sổ cắt.



Hình 1.5. 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

## 1.3. DevC/C++ và thư viện grpahics.h

Để thực hiện các thuật toán xén tỉa đoạn thẳng, xén tỉa đa giác cần sử dụng ngôn ngữ lập trình Dev-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** hay tên chính xác và đầy đủ của nó là Borland Graphics Interface - còn được biết đến với tên gọi BGI - 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) bao gồm 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 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:

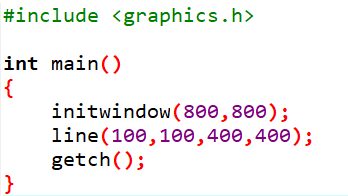
**Bước 1:** Copy tập tin 6-ConsoleAppGraphics.template và ConsoleApp\_cpp\_graph.txt vào thư mục C:\Program Files\Dev-Cpp\Templates

**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.

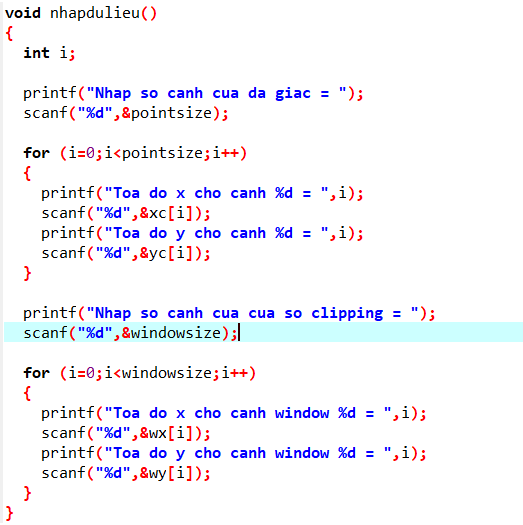
**Bước 5:** Tạo một Project mới bằng cách chọn File → New → Project → Console Graphics Application.



Hình 2.1. Thử nghiệm thư viện với thao tác vẽ đường thẳng

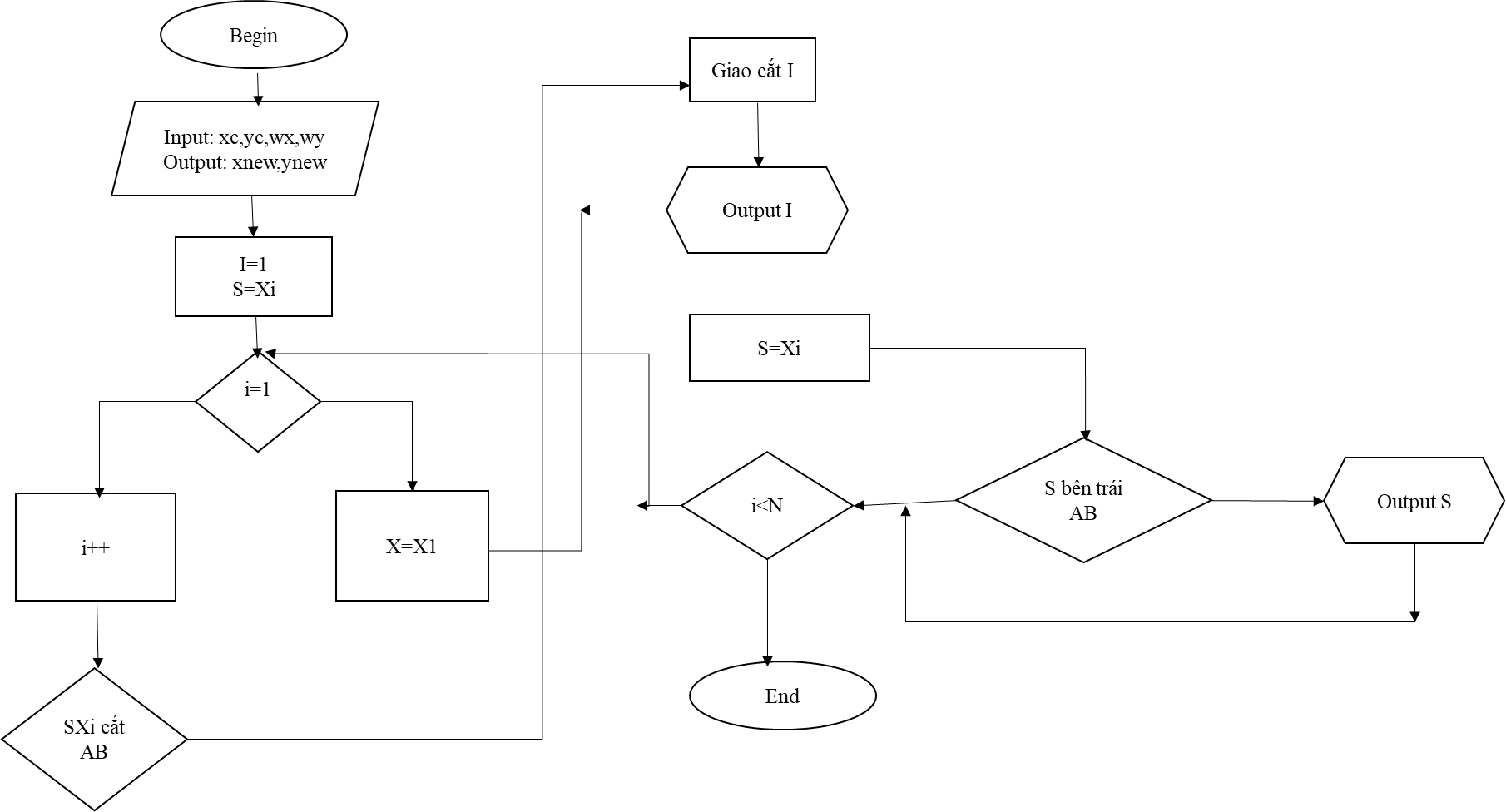
## 2.2. Nhập dữ liệu

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



Hình 2.3: Sơ đồ khối thuật toán Hodgman

Cho X1(x1,y1), X2(x2,y2) và cửa sổ cắt ABCD (AB – cạnh trái, BC – cạnh trên, CD – cạnh phải, AD – cạnh dưới). Vị trí X1X2 so với ABCD thuộc 1 trong 4 trường hợp sau: (1) X1 nằm bên ngoài cửa sổ cắt và X2 nằm trong cửa sổ cắt thì X2 được thêm vào danh sách đầu ra. Các điểm giao nhau của ranh giới cửa sổ và mặt đa giác (cạnh) cũng được bổ sung vào dòng đầu ra. Nếu X1X2 nằm bên trong cửa sổ cắt thì X1X2 được thêm vào danh sách đầu ra. Nếu X1 nằm bên trong cửa sổ và X2 nằm ngoài cửa sổ cắt thì X1 được thêm vào danh sách đầu ra. Nếu X1X2 nằm ngoài cửa sổ thì không có gì được thêm vào danh sách đầu ra.

### 2.3.1. Trường hợp cắt cạnh bên trái

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 |  |
| **TH2**: Đường thẳng có từ trong ra ngoài |  |
| **TH3:** Đường thẳng nằm hoàn toàn bên trong |  |
| **TH4:** Nằm hoàn toàn bên ngoài | Không có bất kỳ điểm nào được thêm vào danh sách mới |

### 2.3.2. Trường hợp cắt cạnh bên phải

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

|  |  |
| --- | --- |
| **TH1:** Đường thẳng có chiều đi từ ngoài vào trong |  |
| **TH2**: Đường thẳng có từ trong ra ngoài |  |
| **TH3:** Đường thẳng nằm hoàn toàn bên trong |  |
| **TH4:** Nằm hoàn toàn bên ngoài | Không có bất kỳ điểm nào được thêm vào danh sách mới |

### 2.3.3. Trường hợp cắt cạnh bên trên

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

|  |  |
| --- | --- |
| **TH1:** Đường thẳng có chiều đi từ ngoài vào trong |  |
| **TH2**: Đường thẳng có từ trong ra ngoài |  |
| **TH3:** Đường thẳng nằm hoàn toàn bên trong |  |
| **TH4:** Nằm hoàn toàn bên ngoài | Không có bất kỳ điểm nào được thêm vào danh sách mới |

### 2.3.4. Trường hợp cắt cạnh bên dưới

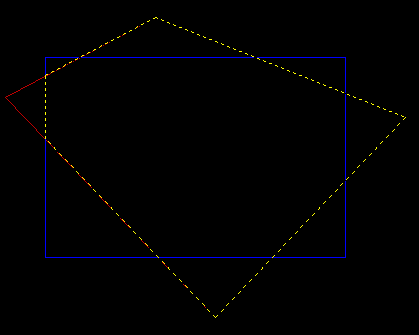
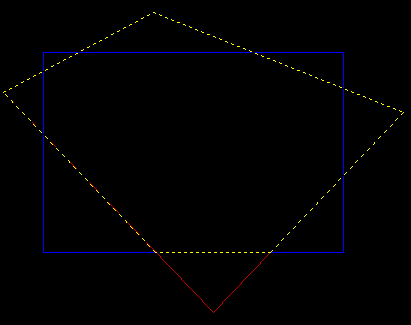
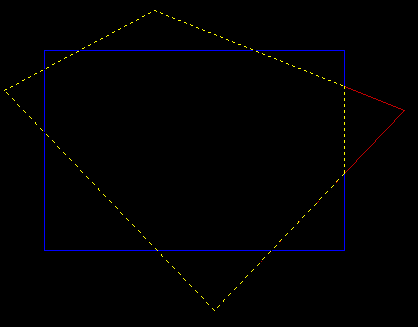
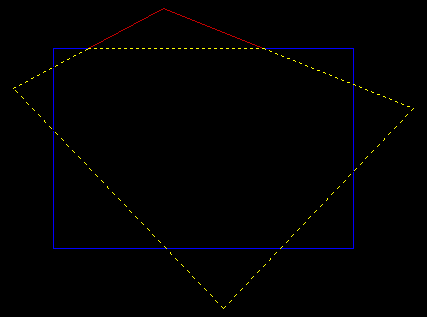
Có 4 trường hợp ứng với vị trí của cạnh đa giác so với cạnh trái của cửa sổ cắt:

|  |  |
| --- | --- |
| **TH1:** Đường thẳng có chiều đi từ ngoài vào trong |  |
| **TH2**: Đường thẳng có từ trong ra ngoài |  |
| **TH3:** Đường thẳng nằm hoàn toàn bên trong |  |
| **TH4:** Nằm hoàn toàn bên ngoài | Không có bất kỳ điểm nào được thêm vào danh sách mới |

# 3. KẾT QUẢ

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

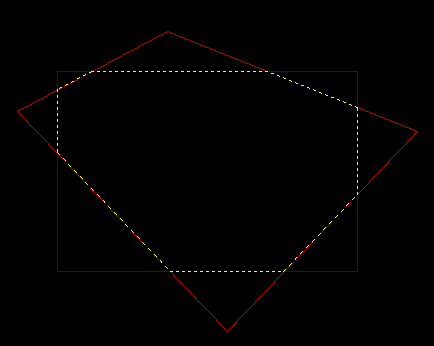
Hình 3.1. Dữ liệu được nhập từ bàn phím cho 2 tham số: cửa sổ cắt và đa giác cần xén tỉa



1. (b) (c) (d)

Hình 3.2. Các trường hợp xén của đa giác

Hình 3.2 có cửa sổ cắt màu xanh và đa giác màu vàng. Trong đó, hình 3.2.a là có đường cắt màu đỏ là giao điểm của đa giác và cửa sổ cắt cạnh bên trái. Hình 3.2.b là có đường cắt màu đỏ là giao điểm của đa giác và cửa sổ cắt cạnh bên phải. Hình 3.2.c là có đường cắt màu đỏ là giao điểm của đa giác và cửa sổ cắt cạnh bên trên. Hình 3.2.d là có đường cắt màu đỏ là giao điểm của đa giác và cửa sổ cắt cạnh bên dưới.



Hình 3.3. Kết quả sau khi xén tỉa

Trong hình 3.3 có cửa sổ cắt màu xanh và và đường màu vàng là đa giác sau khi thực hiện xén tỉa bằng thuật toán Sutherland-Hodgman.

# 4. KẾT LUẬN

Phương pháp này đòi hỏi một lượng bộ nhớ đáng kể. Đầu tiên của tất cả các đa giác được lưu trữ ở dạng ban đầu. Sau đó cắt vào cạnh trái được thực hiện và đầu ra được lưu trữ. Sau đó cắt vào cạnh phải thực hiện, sau đó cạnh trên. Cuối cùng, cạnh dưới được cắt bớt. Kết quả của tất cả các hoạt động này được lưu trữ trong bộ nhớ. Vì vậy, lãng phí bộ nhớ để lưu trữ đa giác trung gian.

Thuật toán Sutherland-Hodgman cho kết quả rất chính xác khi làm việc với các đa giác lồi, tuy nhiên với các đa giác lõm kết quả hiển thị có thể sẽ có đoạn thừa. Điều này xảy ra khi đa giác sau khi xén bị tách thành hai hay nhiều vùng. Do chúng ta chỉ lưu kết quả xuất trong một danh sách các đỉnh nên đỉnh cuối của danh sách ứng với đa giác trước sẽ nối với đỉnh đầu của danh sách ứng với đa giác sau. Một trong nhiều cách để khắc phục điểm này là phân đa giác lõm thành hai hay nhiều đa giác lồi và xử lí mỗi đa giác lồi riêng.