BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---🙢🕮🙠---**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**SỬ DỤNG THƯ VIỆN GRAPHICS.H MÔ PHỎNG GIẢI THUẬT TÌM XÉN TỈA ĐOẠN THẰNG BẰNG GIẢI THUẬT HODGMAN**

**Giảng viên hướng dẫn : Đoàn Vũ Thịnh**

**Sinh viên thực hiện : Huỳnh Trần Quỳnh Nga**

**Mã số sinh viên : 631322308**

**Lớp : 64.CNTT-3**

Khánh Hòa -2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---🙢🕮🙠---**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**SỬ DỤNG THƯ VIỆN GRAPHICS.H MÔ PHỎNG GIẢI THUẬT TÌM XÉN TỈA ĐOẠN THẰNG BẰNG GIẢI THUẬT HODGMAN**

**Giảng viên hướng dẫn : Đoàn Vũ Thịnh**

**Sinh viên thực hiện : Huỳnh Trần Quỳnh Nga**

**Mã số sinh viên : 631322308**

**Lớp : 64.CNTT-3**

Khánh Hòa -2024

MỤC LỤC

[Lời cảm ơn 4](#_Toc186888103)

[Chương 1. Giới Thiệu 5](#_Toc186888104)

[Chương 2. Giải thuật xén tỉa đa giác (Sutherland Hodgman) 6](#_Toc186888105)

[2.1. Khái niệm 6](#_Toc186888106)

[2.2. Giải thuật Hodgman 6](#_Toc186888107)

[2.3. Ví dụ minh hoạ 7](#_Toc186888108)

[Chương 3: Phương Pháp 9](#_Toc186888109)

[3.1 Giới thiệu công cụ và ngôn ngữ sử dụng 9](#_Toc186888110)

[3.1.1 Phần mềm DevC/C++ và thư viện graphics.h, winbmin 9](#_Toc186888111)

[3.1.2 Cài đặt DevC/C++ và thư viện graphics.h, winbmin 9](#_Toc186888112)

[Chương 4: Viết Chương trình cài đặt thuật toán Sutherland Hodgman 11](#_Toc186888113)

[TÀI LIỆU THAM KHẢO: 14](#_Toc186888114)

**MỤC LỤC HÌNH ẢNH**

[Hình 1. 1. Ví dụ minh họa cho ứng dụng thuật toán xén tỉa đa giác trong phần mềm MS PowerPoint 2013 5](#_Toc186248758)

[Hình 2. 1. Qui tắt tìm hướng dương của một đa giác hướng 6](#_Toc186248759)

[Hình 2. 2Các trường hợp trong giải thuật Hodgman 6](#_Toc186248760)

[Hình 2. 3 Xén tỉa 2 đa giác 7](#_Toc186248761)

[Hình 2. 4 Sơ đồ khối thuật toán Hodgman 7](#_Toc186248762)

[Hình 2. 5 Ví dụ dùng giải thuật Hodgman xén tỉa đa giác 8](#_Toc186248763)

[Hình 2. 6 Ví dụ dùng thuật toán xén tỉa P1P2 8](#_Toc186248764)

[Hình 3. 1 Thiết lập phiên bản TDM-GCC phù hợp với DevC/C++ phiên bản 32 bit 10](#_Toc186248765)

[Hình 3. 2 Thiết lập project khởi động đồ họa sử dụng thư viện graphics.h 11](#_Toc186248766)

[Hình 3.3. Code minh họa trong đồ họa 11](#_Toc186248767)

[Hình 4.1. Trường hợp 1: Cả hai điểm trong khu vực cắt 14](#_Toc186248768)

[Hình 4.2. Trường hợp 2: Một điểm trong và một điểm ngoài khu vực cắt 14](#_Toc186248769)

[Hình 4.3. Trường hợp 3: Cả hai điểm ngoài khu vực cắt 14](#_Toc186248770)

[Hình 4.4. Trường hợp 4: Một điểm trên biên khu vực cắt 15](#_Toc186248771)

# Lời cảm ơn

Để có thể hoàn thành đợt thực tập lần này, em xin chân thành cảm ơn đến thầy khoa Công nghệ Thông tin đã tạo điều kiện hỗ trợ và giúp đỡ em trong quá trình học tập và nghiên cứu đề tài này.

Với lòng biết ơn , em xin gửi đến thầy cô ở Khoa Công Nghệ Thông Tin, đặc biệt là thầy Đoàn Vũ Thịnh người đã trực tiếp hướng dẫn em. Cảm ơn thầy đã truyền đạt vốn kiến thức cho em trong suốt thời gian học tập tại trường vừa qua. Nhờ có những lời hướng dẫn, dạy bảo của các thầy nên đề tài thực tập cơ sở của em mới có thể hoàn thiện tốt đẹp.

Bài báo cáo thực tập thực hiện trong khoảng thời gian 6 tuần. Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

# Chương 1. Giới Thiệu

Thuật toán Hodgman (hay Thuật toán xén tỉa đa giác Hodgman) là một thuật toán được sử dụng trong xử lý đồ họa máy tính để cắt một đa giác bằng cách sử dụng một đa giác khác. Mục tiêu của thuật toán này là cắt một đa giác theo các cạnh của một đa giác khác, từ đó tạo ra một vùng giao nhau giữa hai đa giác.

Giải thuật Hodgman là một phương pháp để cắt xén các đoạn thẳng của một đa giác lồi. Thuật toán hoạt động dựa trên nguyên lý kiểm tra các điểm của đa giác, từ đó xác định xem những điểm này nằm bên trong hay bên ngoài vùng cắt. Sau đó, thuật toán thực hiện cắt xén dựa trên các điểm giao cắt.

Xén tỉa là quá trình xác định các điểm của một đối tượng đồ họa nằm trong hay ngoài cửa sổ hiển thị. Những điểm nằm trong cửa sổ hiển thị sẽ được giữ lại, còn những điểm nằm ngoài sẽ bị loại bỏ.

Việc loại bỏ từng điểm ảnh của đối tượng thường mất thời gian hơn, đặc biệt là khi phần lớn đối tượng nằm ngoài cửa sổ hiển thị.

Thao tác loại bỏ các phần hình ảnh nằm ngoài một vùng cho trước được gọi là xén hình. Vùng dùng để xén hình gọi là cửa sổ cắt (clip window). Tùy thuộc vào từng ứng dụng cụ thể, cửa sổ xén có thể có dạng là đa giác hoặc là đường cong khép kín. Kỹ thuật xén tỉa hình ảnh được ứng dụng rộng rãi trong các phần mềm đồ họa như AutoCAD, Photoshop, Adobe Illustrator, và cả phần mềm PowerPoint 2013.(Hình 1. 1)

A star and a star with green lines

Description automatically generated with medium confidence

Hình 1. . Ví dụ minh họa cho ứng dụng thuật toán xén tỉa đa giác trong phần mềm MS PowerPoint 2013

Giả sử, có một **hình ngôi sao** cần xén và một **hình chữ nhật** là cửa sổ cắt. Sau khi áp dụng thuật toán Hodgman, phần ngôi sao nằm bên ngoài hình chữ nhật sẽ bị loại bỏ, chỉ còn lại phần ngôi sao nằm trong vùng của hình chữ nhật, tạo ra một kết quả như hình minh họa.

# Chương 2. Giải thuật xén tỉa đa giác (Sutherland Hodgman)

## 2.1. Khái niệm

- Theo qui ước: một đa giác với các đỉnh P1, …., (các cạnh là Pi và P1) được gọi là theo hướng dương nếu các hình theo thứ tự đã cho tạo thành mạch ngược chiều kim đồng hồ.

A hexagon with arrows and letters

Description automatically generated

Hình 2. . Qui tắt tìm hướng dương của một đa giác hướng

Nếu bàn tay phải của 1 người dọc theo bất kỳ cạnh Pi hoặc P1 cũng chỉ về bên trong đa giác

## 2.2. Giải thuật Hodgman

Quy tắc này yêu cầu khoảng cách giữa một điểm và một mặt phẳng, nhưng chúng ta chỉ cần dấu của khoảng cách. Nếu khoảng cách giữa một điểm và một mặt phẳng là dương, thì điểm đó được coi là ở phía trước của mặt phẳng, nếu không thì ở phía sau của mặt phẳng.

* Nếu cả  và đều nằm bên trái của cạnh này thì được lưu lại (đưa vào output) của đa giác cắt tỉa
* Nếu cả và đều nằm bên phải của cạnh thì không có đỉnh nào được lưu lại
* Nếu nằm bên trái và nằm bên phải của cạnh thì giao điểm I của với cạnh sẽ được lưu lại.
* Nếu nằm bên phải và nằm bên trái thì cả giao điểm I và đều được lưu

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

Hình 2. Các trường hợp trong giải thuật Hodgman

Thuật toán kết thúc bằng một đa giác mới có các đỉnh là điểm tiếp xúc của hai đa giác va chạm

A group of squares in different colors

Description automatically generated

Hình 2. Xén tỉa 2 đa giác

Đa giác mới sau khi được xén tỉa có màu xanh lá.

## 2.3. Ví dụ minh hoạ

A black and white logo

Description automatically generated

Hình 2. Ví dụ dùng giải thuật Hodgman xén tỉa đa giác

Kết quả sau khi xén là đa giác chuyển thành vùng tô

**Ví dụ**: Hãy sử dụng thuật toán Hodgman để xén tỉa 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)

A diagram of a rectangle with a line

Description automatically generated

Hình 2. Ví dụ dùng thuật toán xén tỉa P1P2

Theo thuật toán Hogman ta xén P1P2 dựa trên từng cạnh.

* AB: Ta xét C=(x2-x1) (y-y1) - (y2-y1) (x-x1)

Điểm P1: C = (5-2) (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 phải

Vậy P1P2 được lưu

* 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

(5,26/7) do vậy lưu lại

* CD:

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

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

(5/2,3) do vậy lưu lại

* DA:

Điểm P1: C = -(1-3) (-1-1) = -4 nằm bên phải

Điểm : C = - (1-3) (5/2-1) = 7/2 nằm bên trái

(1,18/7)

lại: Cắt xén đoạn được đoạn có (5/2,3) và (1,18/7)

# Chương 3: Phương Pháp

## 3.1 Giới thiệu công cụ và ngôn ngữ sử dụng

### 3.1.1 Phần mềm DevC/C++ và thư viện graphics.h, winbmin

Để thực hiện các thuật toán xén tỉa đoạn thẳng và xén tỉa đa giác trên máy tính, chúng ta có thể sử dụng nhiều phần mềm phát triển, chẳng hạn như DevC++ hoặc Visual Studio Code. Trong bài báo cáo này, chúng tôi sử dụng DevC++ phiên bản 5.11, một công cụ phát triển mã nguồn mở dành cho các ứng dụng C/C++. DevC++ dựa vào trình biên dịch MinGW (Minimalist GNU for Windows), một công cụ biên dịch mã nguồn mở với nền tảng GCC (GNU Compiler Collection) để biên dịch mã nguồn cho cả hệ điều hành Windows và Linux.

Mặc dù DevC++ không có nhiều tính năng mạnh mẽ như các công cụ IDE thương mại khác, chẳng hạn như Visual Studio của Microsoft, nhưng nó vẫn là một công cụ lý tưởng trong môi trường giáo dục đại học để dạy và học lập trình C/C++. DevC++ hỗ trợ chuẩn C/C++, và mã nguồn được viết trên DevC++ có thể biên dịch trên các IDE khác, điều này làm cho công cụ này trở thành lựa chọn phù hợp cho việc học và phát triển phần mềm mã nguồn mở.

Để thực hiện các thuật toán xén tỉa trong DevC++, chúng ta cần sử dụng một số thư viện đồ họa hỗ trợ, giúp việc lập trình trở nên đơn giản hơn. Một trong những thư viện nổi bật là Graphics.h, hay còn được biết đến với tên gọi Borland Graphics Interface (BGI). Đây là một thư viện đồ họa phổ biến trên hệ điều hành DOS và các phiên bản Windows cũ như Windows 95 và Windows 98. Graphics.h có ưu điểm là dễ sử dụng và dễ cài đặt, nên cho đến nay vẫn được nhiều trường đại học lựa chọn để giảng dạy.

Thư viện winbgim (Windows BGI Interface Module) là một thư viện hỗ trợ việc sử dụng các hàm đồ họa BGI trên hệ điều hành Windows. Nó giúp bạn thực hiện các thao tác đồ họa cơ bản như vẽ hình dạng, hiển thị văn bản với nhiều kiểu chữ khác nhau, thay đổi màu sắc, và nhiều thao tác khác. Winbgim còn hỗ trợ tính năng chuột đơn giản cho các ứng dụng đồ họa, phù hợp với trình biên dịch mingw32 g++ của C++ hoặc Borland C++ (phiên bản 5.0). Thư viện này cũng có thể làm việc với các trình biên dịch Windows khác.

DevC++ là một công cụ phát triển mạnh mẽ và dễ sử dụng, kết hợp với các thư viện đồ họa như Graphics.h và winbgim, giúp quá trình lập trình đồ họa, đặc biệt là thực hiện các thuật toán xén tỉa, trở nên đơn giản và hiệu quả hơn.

### 3.1.2 Cài đặt DevC/C++ và thư viện graphics.h, winbmin

DevC/C++ phiên bản 4.9.9.2 được tải về và cài đặt theo hướng dẫn tại địa chỉ: <https://sourceforge.net/projects/orwelldevcpp/files/latest/download>

Để lập trình đồ họa trong DevC cần sử dụng thư viện graphics.h theo địa chỉ sau: <https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library>

Và tiến hành cài đặt thư viện graphics.h theo các bước sau:

**Bước 1:** Copy tập tin *6-ConsoleAppGraphics, ConsoleApp\_cpp\_graph* đến đường dẫn *C:\Program Files\Dev-Cpp\Templates*

**Bước 2:** Copy tập tin *graphics, winbgim* đến thư mục *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\include*

**Bước 3:** Copy tập tin *libbgi.a* đến đường dẫn *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\lib*

Bước 4: Hiệu chỉnh phiên bản TDM-GCC cho phù hợp với hệ điều hành trong phần mềm DevC/C++ theo đường dẫn sau: Tools - Compiler Optionta chọn TDM-GCC 4.9.2 32bit Release**.** Thiết lập phiên bản TDM-GCC phù hợp với DevC/C++ phiên bản 32 bit

*A screenshot of a computer

Description automatically generated*

Hình 3. Thiết lập phiên bản TDM-GCC phù hợp với DevC/C++ phiên bản 32 bit

**Bước 5:** Tạo mới dự án (Project) đồ họa sử dụng thư viện graphics.h theo các bước sau: File → New → Project → Console Graphics Application với tên project được đánh dấu bởi vùng màu đỏ (Hình 3. 2)

*A screenshot of a computer program

Description automatically generated*

Hình 3. Thiết lập project khởi động đồ họa sử dụng thư viện graphics.h

Dưới đây là một đoạn code minh họa trong đồ họa

*A white background with blue text

Description automatically generated*

Hình 3.3. Code minh họa trong đồ họa

# Chương 4: Viết Chương trình cài đặt thuật toán Sutherland Hodgman

* **Khai báo thư viện:**

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include <conio.h>

* **Code:**

// Ham tinh giao diem x cua hai doan thang

int tinh\_giao\_diem\_x(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {

int den = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

if (den == 0) {

return -1; // Tra ve -1 neu doan thang song song, khong co giao diem

}

int num = (x1 \* y2 - y1 \* x2) \* (x3 - x4) - (x1 - x2) \* (x3 \* y4 - y3 \* x4);

return num / den;

}

// Ham tinh giao diem y cua hai doan thang

int tinh\_giao\_diem\_y(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {

int den = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

if (den == 0) {

return -1; // Tra ve -1 neu doan thang song song, khong co giao diem

}

int num = (x1 \* y2 - y1 \* x2) \* (y3 - y4) - (y1 - y2) \* (x3 \* y4 - y3 \* x4);

return num / den;

}

// Ham cat da giac voi mot canh cua vung xen

void cat\_voi\_canh(int da\_giac[][2], int \*so\_dinh\_da\_giac, int x1, int y1, int x2, int y2) {

int diem\_moi[MAX\_DIEM][2], so\_dinh\_moi = 0;

// Duyet qua cac dinh cua da giac

for (int i = 0; i < \*so\_dinh\_da\_giac; i++) {

int k = (i + 1) % \*so\_dinh\_da\_giac; // Tim dinh ke cua i

int ix = da\_giac[i][0], iy = da\_giac[i][1];

int kx = da\_giac[k][0], ky = da\_giac[k][1];

// Tinh vi tri cua hai diem so voi canh clipper

int i\_pos = (x2 - x1) \* (iy - y1) - (y2 - y1) \* (ix - x1);

int k\_pos = (x2 - x1) \* (ky - y1) - (y2 - y1) \* (kx - x1);

// Truong hop 1: Ca hai diem deu ben trong

if (i\_pos < 0 && k\_pos < 0) {

diem\_moi[so\_dinh\_moi][0] = kx;

diem\_moi[so\_dinh\_moi][1] = ky;

so\_dinh\_moi++;

}

// Truong hop 2: Chi co diem dau tien ben ngoai

else if (i\_pos >= 0 && k\_pos < 0) {

int giao\_x = tinh\_giao\_diem\_x(x1, y1, x2, y2, ix, iy, kx, ky);

int giao\_y = tinh\_giao\_diem\_y(x1, y1, x2, y2, ix, iy, kx, ky);

if (giao\_x != -1 && giao\_y != -1) { // Kiem tra giao diem hop le

diem\_moi[so\_dinh\_moi][0] = giao\_x;

diem\_moi[so\_dinh\_moi][1] = giao\_y;

so\_dinh\_moi++;

}

diem\_moi[so\_dinh\_moi][0] = kx;

diem\_moi[so\_dinh\_moi][1] = ky;

so\_dinh\_moi++;

}

// Truong hop 3: Chi co diem thu hai ben ngoai

else if (i\_pos < 0 && k\_pos >= 0) {

int giao\_x = tinh\_giao\_diem\_x(x1, y1, x2, y2, ix, iy, kx, ky);

int giao\_y = tinh\_giao\_diem\_y(x1, y1, x2, y2, ix, iy, kx, ky);

if (giao\_x != -1 && giao\_y != -1) { // Kiem tra giao diem hop le

diem\_moi[so\_dinh\_moi][0] = giao\_x;

diem\_moi[so\_dinh\_moi][1] = giao\_y;

so\_dinh\_moi++;

}

}

}

\*so\_dinh\_da\_giac = so\_dinh\_moi; // Cap nhat lai so dinh da giac

for (int i = 0; i < \*so\_dinh\_da\_giac; i++) {

da\_giac[i][0] = diem\_moi[i][0];

da\_giac[i][1] = diem\_moi[i][1];

}

}

// Ham cat da giac theo vung xen

void cat\_da\_giac\_theo\_vung\_xen(int da\_giac[][2], int \*so\_dinh\_da\_giac, int vung\_xen[][2], int so\_dinh\_vung\_xen) {

// Duyet qua moi canh cua vung xen

for (int i = 0; i < so\_dinh\_vung\_xen; i++) {

int k = (i + 1) % so\_dinh\_vung\_xen;

cat\_voi\_canh(da\_giac, so\_dinh\_da\_giac, vung\_xen[i][0], vung\_xen[i][1], vung\_xen[k][0], vung\_xen[k][1]);

}

}

// Ham ve da giac

void ve\_da\_giac(int da\_giac[][2], int so\_dinh\_da\_giac) {

// Ve tung cap diem lien tiep trong da giac

for (int i = 0; i < so\_dinh\_da\_giac; i++) {

int j = (i + 1) % so\_dinh\_da\_giac; // Dinh ke cua dinh i

line(da\_giac[i][0], da\_giac[i][1], da\_giac[j][0], da\_giac[j][1]);

}

}

// Ham ve vung xen

void ve\_vung\_xen(int vung\_xen[][2], int so\_dinh\_vung\_xen) {

// Ve tung cap diem lien tiep trong vung xen

for (int i = 0; i < so\_dinh\_vung\_xen; i++) {

int j = (i + 1) % so\_dinh\_vung\_xen; // Dinh ke cua dinh i

line(vung\_xen[i][0], vung\_xen[i][1], vung\_xen[j][0], vung\_xen[j][1]);

}

}

* **Kết Quả sau khi thực hiện**

**Link youtube:** [**https://www.youtube.com/watch?v=sEt-PrUHIpg**](https://www.youtube.com/watch?v=sEt-PrUHIpg)

Cho đa giác với các đỉnh sau (40, 40), (120, 40), (120, 120), (40, 120) và các đỉnh của vùng cắt (20, 20), (100, 20), (100, 100), (20, 100)

A screenshot of a computer

Description automatically generated

Hình 4.1. Trường hợp 1: Cả hai điểm trong khu vực cắt

Cho đa giác với các đỉnh sau (20, 20), (140, 20), (140, 100), (20, 100) và các đỉnh của vùng cắt (60, 40), (100, 40), (100, 80), (60, 80)

A screen shot of a computer

Description automatically generated

Hình 4.2. Trường hợp 2: Một điểm trong và một điểm ngoài khu vực cắt

Cho đa giác với các đỉnh sau (20, 20), (60, 20), (60, 60), (20, 60)và các đỉnh của vùng cắt (80, 40), (120, 40), (120, 80), (80, 80)

A screenshot of a computer

Description automatically generated

Hình 4.3. Trường hợp 3: Cả hai điểm ngoài khu vực cắt

Cho đa giác với các đỉnh sau (40, 40), (120, 40), (120, 120), (40, 120)và các đỉnh của vùng cắt (80, 40), (80, 120), (40, 120), (40, 40)

A screenshot of a computer

Description automatically generated

Hình 4.4. Trường hợp 4: Một điểm trên biên khu vực cắt

# TÀI LIỆU THAM KHẢO:

<https://github.com/thinhdoanvu/ComputerGraphics/blob/master/Coding/Clipping/Hogman_Polygon_Clipping>

<https://github.com/thinhdoanvu/ComputerGraphics/blob/master/Chapter4_Edit21April2019.pptx>