TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**CÀI ĐẶT THUẬT TOÁN TÔ MÀU ĐA GIÁC   
BẰNG GIẢI THUẬT SCANLINE VÀ FLOODFILL**

Giảng viên hướng dẫn: ThS. Nguyễn Đình Cường

Sinh viên thực hiện: Trần Ngọc Phương Anh

Mã số sinh viên: 63133463

Khánh Hòa, tháng 01/2024

# **MỤC LỤC**

[MỤC LỤC i](#_Toc155220395)

[DANH MỤC HÌNH ii](#_Toc155220396)

[TÓM TẮT iii](#_Toc155220397)

[CHƯƠNG I: GIỚI THIỆU 1](#_Toc155220398)

[1.1 Thuật toán tô màu đối tượng 1](#_Toc155220399)

[1.2. Thuật toán tô màu theo dòng quét Scanline 2](#_Toc155220400)

[1.2.1. Ý tưởng chính của thuật toán 2](#_Toc155220401)

[1.2.2 Lưu đồ thuật toán 3](#_Toc155220402)

[1.3. THUẬT TOÁN TÒ MÀU DỰA THEO ĐƯỜNG BIÊN 5](#_Toc155220403)

[1.4 Dev C++ và thư viện Graphics.h 8](#_Toc155220404)

[1.4.1 Dev C++ và tính năng 8](#_Toc155220405)

[1.4.2 Thư viện Graphics.h 9](#_Toc155220406)

[CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU 10](#_Toc155220407)

[2.1 Cài đặt thuật toán Scanline 10](#_Toc155220408)

[2.2 Cài đặt thuật toán Floodfill 15](#_Toc155220409)

[2.2.1 Cài đặt bằng phương pháp đệ quy 15](#_Toc155220410)

[2.2.2 Cài đặt bằng phương pháp không đệ quy 16](#_Toc155220411)

[CHƯƠNG 3: KẾT QUẢ 18](#_Toc155220412)

[3.1 Tô màu Scanline 18](#_Toc155220413)

[3.2 Tô màu Floodfill bằng đệ quy 18](#_Toc155220414)

[3.3 Tô màu Floodfill không đệ quy 18](#_Toc155220415)

[TÀI LIỆU THAM KHẢO 20](#_Toc155220416)

DANH MỤC HÌNH

[Hình 1.1. Các kiểu mẫu tô (Solid - b và Pattern - c) 6](#_Toc155219165)

[Hình 1.2. Dòng quét y=k1/2 đi ngang qua đỉnh sẽ được xét 2 lần 7](#_Toc155219166)

[Hình 1.3. Thuật toán Floodfill 11](#_Toc155219167)

[Hình 1.4. Bốn điểm lân cận của thuật toán Floodfill 11](#_Toc155219168)

[Hình 1.5. Minh họa thuật toán Flood Fill 13](#_Toc155219169)

[Hình 3.1. Kết quả thuật toán tô màu Scanline 23](#_Toc155219170)

[Hình 3.2. Kết quả thuật toán tô màu Floodfill đệ quy 23](#_Toc155219171)

[Hình 3.3. Kết quả thuật toán tô màu Floodfill không đệ quy 24](#_Toc155219172)

TÓM TẮT

Kỹ thuật đồ họa trong công nghệ thông tin bao gồm một loạt các phương pháp và công nghệ được sử dụng để tạo ra và xử lý hình ảnh, đồ họa và video trên máy tính.

Đồ họa máy tính cũng là khái niệm chính liên quan đến kỹ thuật đồ họa, lĩnh vực nghiên cứu và ứng dụng về việc tạo ra, hiện thị và xử lý hình ảnh và đối tượng hình học trên máy tính. Ngoài ra còn có đồ họa 2D và 3D thường cho ra hình ảnh phẳng và hình ảnh có chiều sâu. Các thuật toán được thiết kế để thực hiện các tác vụ như vẽ đường, tô màu, chiếu sáng và các phép biến đổi hình ảnh. Thuật toán tô màu bằng dòng quét và tô màu đường biên trong kỹ thuật đồ họa có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm đồ họa phổ biến hiện nay.

Quy trình thực hiện được trải qua các bước từ cài đặt thuật toán, hiển thị kết quả 2D trên màn hình đều được thực hiện trên ngôn ngữ lập trình C/C++ thông qua ứng dụng DevC/C++ có kết hợp với thư viện graphics.h.

Sản phẩm đã minh họa được từng bước giải thuật tô màu đối tượng với Scanline và FloodFill. Đồng thời cũng chỉ ra các trường hợp hạn chế của mỗi thuật toán và cách khắc phục các nhược điểm đó.

# CHƯƠNG I: GIỚI THIỆU

# 1.1 Thuật toán tô màu đối tượng

Thuật toán tô màu đối tượng là loại thuật toán trong kinh vực xử lý ảnh và thị giác máy tính, được thiết kế để tô màu cho các đối tượng trong một hình ảnh màu đen trắng. Trong nhiều trường hợp, hình ảnh màu đen trắng chỉ chứa thông tin về cường độ ánh sáng và không có thông tin màu sắc. Thuật toán tô màu đối tượng giúp thêm màu sắc vào các đối tượng trong ảnh, làm cho hình ảnh trở nên sống động hơn. Các vùng tô là một trong những đối tượng đồ họa cơ sở được hầu hết các công cụ lập trình đồ họa hỗ trợ. Có hai dạng vùng tô thường gặp: (1) tô bằng một màu thuần nhất (solid fill); (2) tô theo một mẫu tô (fill-pattern) nào đó.

Một vùng tô thường được xác định bởi một đường khép kín nào đó, gọi là đường biên. Vùng tô và mẫu tô được minh họa bởi các hình 1.2.

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

Hình 1.1. Các kiểu mẫu tô (Solid - b và Pattern - c)

Để tô màu một vùng tô, người ta thường chia thành hai công đoạn:

1. Xác định các điểm để tô
2. Xác định giá trị màu tô.

Có hai cách tiếp cận chính để tô màu một vùng tô đối với thiết bị hiển thị dạng điểm, nó là tô dựa theo đường biên (boundary fill) và tô theo dòng quét (scanline fill). Trong đó, phương pháp tô dựa theo đường biên (boundary fill) sẽ bắt đầu từ một điểm ở bên trong vùng tô và từ đó loang dần ra cho tới khi gặp các điểm biên. Cách tiếp cận này thường được dùng cho các vùng tô có dạng đường biên phức tạp. Phương pháp tô theo dòng quét (scanline fill) sẽ xác định các phần giao của các dòng quét kế tiếp nhau với đường biên của vùng tô, sau đó sẽ tô màu các điểm thuộc về phần giao này. Cách tiếp cận này thường được dùng để tô màu các đa giác, đường tròn, ellipse và một số đường cong đơn giản.

## **1.2. Thuật toán tô màu theo dòng quét Scanline**

Thuật toán tô màu Scaline (hay còn gọi là thuật toán tô màu theo dòng quét) được áp dụng để tô màu đa giác lồi, lõm hay đa giác tự cắt. Với mỗi dòng quét, ta sẽ xác định phần giao của đa giác và dòng quét, rồi tô màu các pixel thuộc đoạn giao đó. Để xác định các đoạn giao, ta tiến hành việc tìm giao điểm của dòng quét với các cạnh của đa giác, sau đó các giao điểm này se được sắp xếp theo thứ tự tăng dần của hoành độ giao điểm. Các đoạn giao chính là các đoạn thẳng được giới hạn bởi từng cặp giao điểm một.

## **1.2.1. Ý tưởng chính của thuật toán**

*Ta có thể tóm tắt các bước chính của thuật toán như sau:*

**Bước 1:** Tìm ymax, ymin lần lượt là giá trị lớn nhất, nhỏ nhất của tập các tung độ của các đỉnh cùa đa giác đã cho:

y max = max {yi, (xi, y i) ϵ P}; ymin = min {yi, (xi, y i) ϵ P}.

**Bước 2:** Ứng với mỗi dòng quét y = k, với k thay đổi từ ymin đến ymax, lặp:

*B2.1.* Tìm tất cả các hoành độ giao điềm của dòng quét y = k với các cạnh của đa giác.

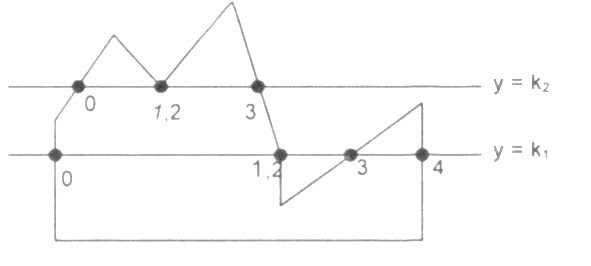
*B2.2.* Sắp xếp các hoành độ giao điểm theo thử tự tăng dần: x0, x1,

*B2.3.* Tô màu các đoạn thẳng trên đường thẳng y = k lần lượt được giới hạn bởi các cặp (x0, x1), (x2, x3),.., (x2k, x2k+1).

***Nhược điểm:***

Nhưng nếu chỉ dừng ở mức này và chuyển sang cài đặt thì chúng ta sẽ gặp phải một số vấn đề như sau:

* Ứng với mỗi dòng quét, không phải lúc nào tất cả các cạnh của đa giác nào cũng cắt dòng quét. Do đó để cải thiện tốc độ cần phải tìm ra cách hạn chế được số cạnh cần tìm giao điểm ứng với mỗi dòng quét.
* Việc tìm giao điểm của các cạnh đa giác với mỗi dòng quét sẽ gặp các phép toán phức tạp như nhân, chia,…trên số thực. Nếu ta dùng cách giải hệ phương trình để tìm giao điểm. Điều này sẽ giảm tốc độ của thuật toán.



Hình 1.2. Dòng quét y=k1/2 đi ngang qua đỉnh sẽ được xét 2 lần

Nếu số giao điểm tìm được giữa các cạnh đa giác và dòng quét là lẻ thì việc nhóm từng cặp giao điểm kế tiếp nhau để hình thành các đoạn tô cỏ thể sẽ không chính xác. Điều này chỉ xảy ra khi dòng quét đi ngang qua các đỉnh của đa giác. Nếu tính số giao điểm tại đỉnh dòng quét đi ngang qua là hai thì có thể sẽ cho kết quả tô không chính xác như trong trường hợp của Hình 1.2.

Ngoài ra, việc tìm giao điểm của dòng quét với các cạnh nằm ngang là một trường hợp đặc biệt cần phải có cách xử lý thích hợp.

**Hướng giải quyết:**

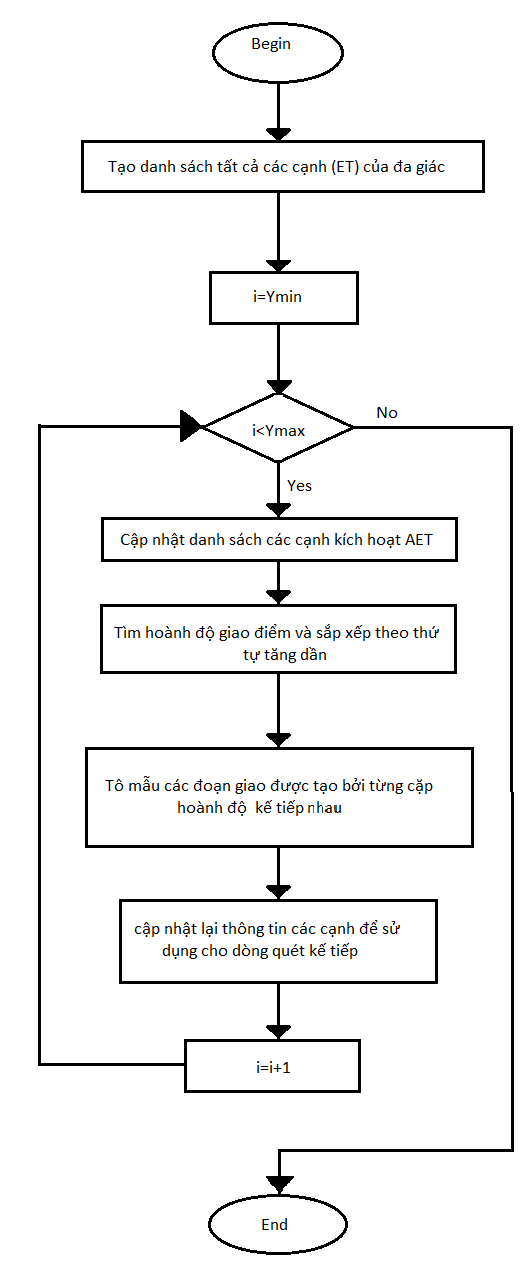
Để hạn chế số cạnh cần tìm giao điểm ứng với dòng quét ta có thể áp dụng công thức hệ số góc sau:

 =  +

Trong đó: m là hệ số cạnh: , lần lượt là hoành độ giao điểm của một cạnh nào đó với dòng quét y=k và y=k+1.

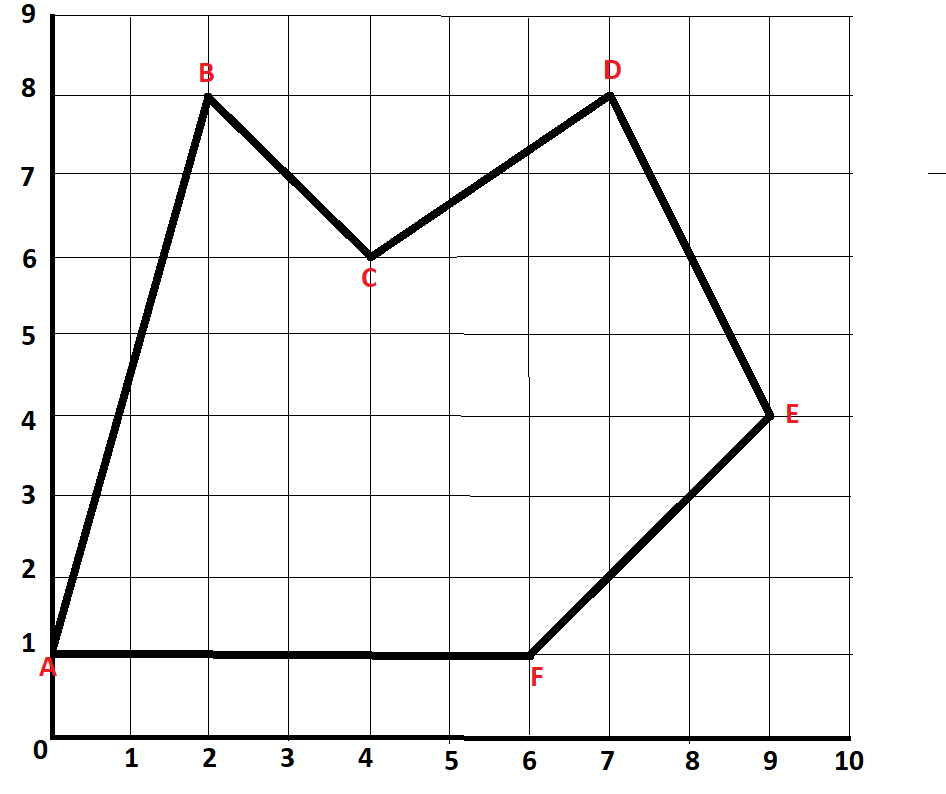
Giải quyết trường hợp số giao điểm đi qua đỉnh đơn điệu thì ta tính số giao điểm là 1, đi qua đỉnh cực trị thì tính số giao điểm là 0 (hoặc 2).

## **1.2.2 Lưu đồ thuật toán**



***Bài toán ví dụ***

Cho 1 đa giác, ABCDEF có tọa độ A(0,1), B(2,8), C(4,6), D(7,8), E(9,4), F(6,1).



**Xét trường hợp dòng quét đi qua cạnh nằm ngang**

Dòng quét k=1 đi qua cạnh AF, loại bỏ cạnh AF.

**Xét trường hợp dòng quét đi qua hai cạnh**

Dòng quét k=5 đi qua hai cạnh AB và ED và cắt hai cạnh ở tọa độ x=1.14 và x=8.5. Các điểm ảnh nằm trong đoạn từ 2 tới 8 sẽ được tô

**Xét trường hợp dòng quét đi qua đỉnh**

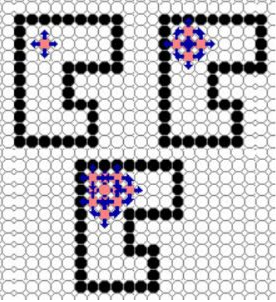
Dòng quét k=6 đi qua bốn cạnh AB, CB, CD, ED và cắt ở bốn tọa độ x= 1.4, 4, 4, 8. Tô hai đoạn, đoạn một từ tọa độ 1.4 đến 4 và đoạn hai từ tọa độ 4 đến 8

Dòng quét k=4 đi qua ba cạnh AB, FE, ED và cắt ở ba tọa độ x=0.9,9,9

Chỉ xét giao của dòng quét với cạnh AB và ED. Tô các điểm ảnh nằm trong đoạn từ 1 tới 9.

## **1.3. THUẬT TOÁN TÒ MÀU DỰA THEO ĐƯỜNG BIÊN**

Thuật toán tô màu đường biên hay còn gọi là tô màu loang. Khác với thuật toán tô màu theo dòng quét (Scanline), đường biên là vùng được tô màu ở thuật toán tô loang được xác định bởi tập các đỉnh của 1 đa giác, đường biên trong thuật toán được mô tả bằng một giá trị duy nhất, đó là màu của tất cả các điểm thuộc về đường biên ( nói ngắn gọn ta sẽ tô đường biên một màu riêng).



#### Hình 1.3. Thuật toán Floodfill

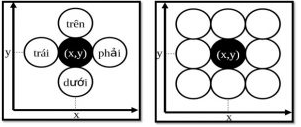
Bắt đầu từ 1 điểm nằm bên trong vùng tô, ta sẽ kiểm tra các điểm lân cận của nó đã được tô màu hay có phải điểm biên hay không. Nếu không phải là điểm đã tô và không phải là điểm biên thì ta sẽ tô màu nó. Lặp lại cho tới khi không còn tô được điểm nào nữa thì dừng.

**Nhược điểm:**

Thuật toán này có thể sẽ không hoạt động chính xác khi có 1 số điểm nằm trong vùng tô có màu là màu cần tô của vùng tô.

Hướng giải quyết:

Để khắc phục nhược điểm trên, trước khi tô màu ta cần phải đảm bảo rằng toàn bộ các điểm thuộc về vùng tô có màu khác màu tô. Có hai quan điểm về cách tô này, đó là dùng 4 điểm lân cận hay 8 điểm lân cận đối với điểm đang xét. Ở đây em sẽ đi xét 4 điểm lân cận.



Hình 1.4. Bốn điểm lân cận của thuật toán Floodfill

Có hai hướng thực thi thuật toán: đó là sử dụng đệ quy và không đệ quy.

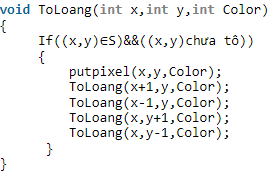
**Thuật toán đệ quy:**

+ Bước 1: Kẻ vùng biên cần tô.

+ Bước 2: Xác định một điểm (x,y) bên trong vùng tô.

+ Bước 3: Tô điểm (x,y) sau đó tô loang những điểm lân cận.

**Hàm đệ quy:**

****

**Nhận xét:**

Có thể tô vùng có hình dạng bất kỳ và cài đặt thuật toán dễ dàng nhưng cũng có nhược điểm là không thể tô các vùng có kích thước lớn hơn do tràn bộ nhớ khi sử dụng đệ quy. Việc gọi đệ quy, thuật toán cho 4 điểm lân cận của điểm hiện hành không quan tâm tới một trong 4 điểm đó đã được xét ở bước trước hay chưa ( tức là sẽ có những điểm bị xét đi xét lại nhiều lần ). Điều này khiến tốc độ chậm, không hiệu quả.

**Thuật toán không đệ quy:**

Thuật toán sẽ tiến hành loang dần và lần lượt tô từng đoạn giao theo dòng quét ngang, thay vì tô theo 4 điểm lân cận.

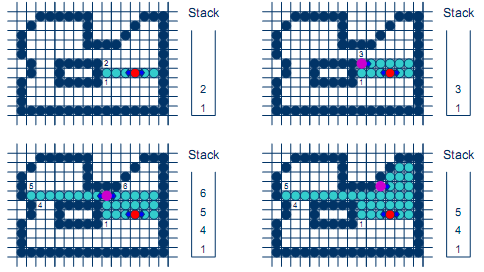
**Bước 1:** Khởi tạo 1 điểm nằm trong vùng tô.

**Bước 2:** Thực hiện tô loang dần theo chiều ngang (trái qua phải và phải qua trái) cho đến khi dụng biên thì dừng lại.

**Bước 3:** Ứng với mỗi điểm trên dòng quét ngang, thực hiện loang để tìm những điểm ảnh có hoành độ nhỏ nhất sát với biên chưa được tô nằm trên và dưới, sau đó lưu vào Stack.

**Bước 4:** Lặp bước 2 nếu còn một điểm trong Stack chưa được tô.

Như vậy, chỉ cần lưu lại thông tin cùa điểm bắt đầu mỗi đoạn giao của dòng quét ngang thay vì phải lưu toàn bộ các điểm lân cận chưa được tô cùa điểm hiện hành.



Hình 1.5. Minh họa thuật toán Flood Fill

**1.4 Dev C++ và thư viện Graphics.h**

**1.4.1 Dev C++ và tính năng**

Dev-C ++ là môi trường phát triển tích hợp (IDE - Integrated Development Environment) dành cho ngôn ngữ lập trình C/C ++. Được phát triển bởi Bloodshed Software, Dev C++ cung cấp một giao diện đồ họa thuận tiện cho việc phát triển ứng dụng sử dụng ngôn ngữ C và C++. Các tính năng của Dev-C++:

* Hỗ trợ trình biên dịch dựa trên GCC.
* Gỡ lỗi tích hợp (sử dụng GDB- General DeBug).
* Quản lý dự án.
* Trình chỉnh sửa cú pháp.
* Trình duyệt lớp.
* Hoàn thành mã.
* Danh sách chức năng.
* Hồ sơ hỗ trợ.
* Nhanh chóng tạo Windows, console, thư viện tĩnh và DLL13.
* Hỗ trợ các mẫu để tạo các loại dự án của riêng bạn.
* Tạo Makefile.
* Chỉnh sửa và biên dịch các tệp Tài nguyên.
* Quản lý công cụ.
* Hỗ trợ in.
* Tìm và thay thế mã lệnh.
* Hỗ trợ CVS.

## **1.4.2 Thư viện Graphics.h**

Thư viện `graphics.h` là một thư viện đồ họa cổ điển được sử dụng trong ngôn ngữ lập trình C và C++. Thư viện này thường được sử dụng để tạo các ứng dụng đồ họa đơn giản như vẽ hình, vẽ đồ thị, hoặc tạo các hiệu ứng đồ họa cơ bản.Các hàm trong `graphics.h` thường được sử dụng để thao tác với các hình học cơ bản như vẽ đường, hình chữ nhật, hình tròn, và văn bản. Tính năng đồ họa của thư viện này thường được sử dụng trong lĩnh vực giáo dục hoặc để giới thiệu lập trình đồ họa đơn giản.

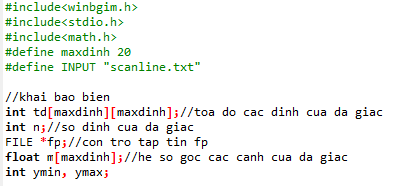
Vì sử dụng DevC++ làm trình biên dịch cho việc cài đặt thuật toán nên không thể

thực hiện trên môi trường Windows. Vì vậy, một môi trường giả lập graphic của Borland C được Michael tạo ra thư viên có tên là Graphics.h. để có thể làm được điều đó. Micheal đã thay đổi BGI library (thư viện BGI) thành thư viện có tên WinBGIm để có thể sử dụng tốt trên windows.

# CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU

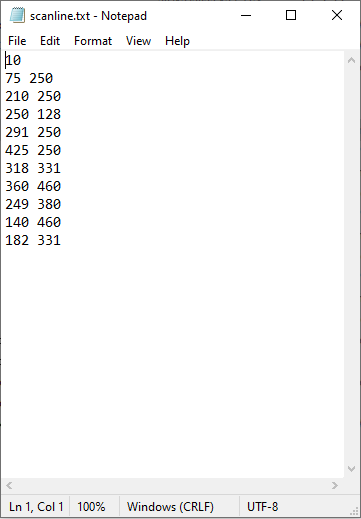
## **2.1 Cài đặt thuật toán Scanline**

**Bước 1: Khai báo thư viện và khai báo biến**

****

**Bước 2: Tạo đa giác và vẽ đa giác**

**Tạo đa giác:** Mỗi đỉnh của đa giác bao gồm hoành độ X[] và tung độ Y[] được lưu trữ trong 2 mảng 1 chiều. Mỗi phần tử là kiểu số nguyên. Tọa độ các đỉnh của đa giác được đưa vào tập tin file “txt” thông qua gọi hàm nhapdinh().

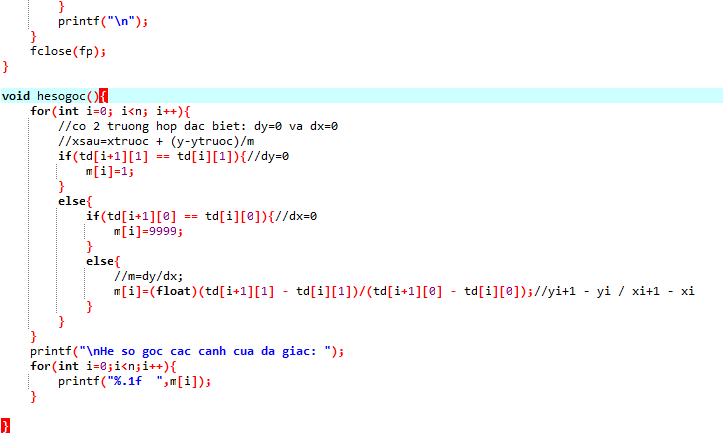
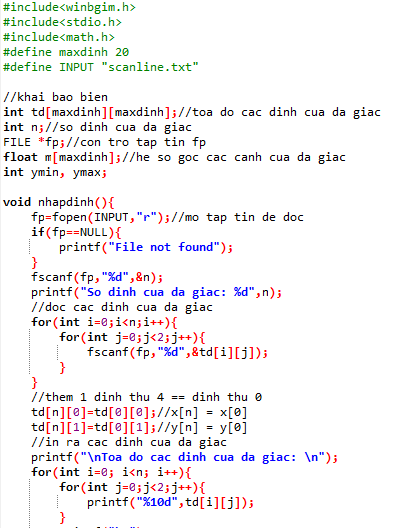


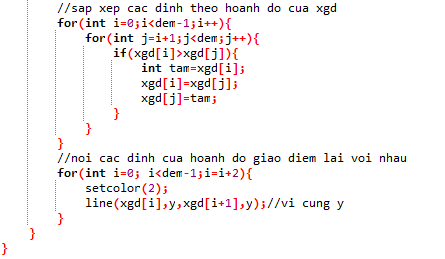
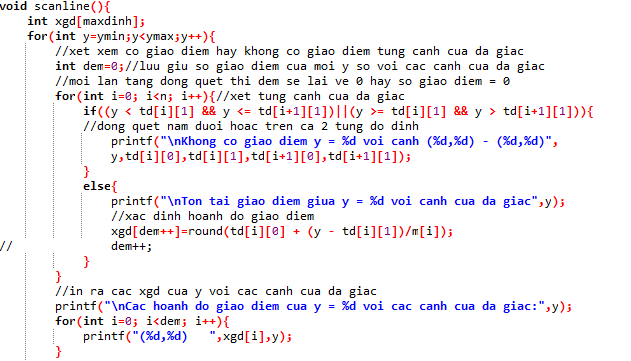
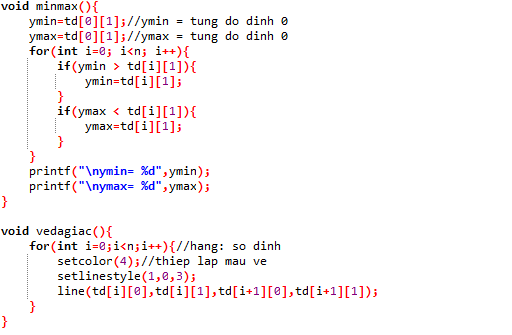
Kết thúc việc nhập đa giác sẽ tính toán hệ số góc trên mỗi cạnh của đa giác và đồng thời xác định tung độ của đỉnh lớn nhất (yMax) và bé nhất (ymin) thông qua hàm minmax().

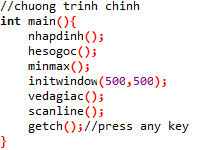
**Vẽ đa giác:** Đa giác sau khi được nhập các tọa độ sẽ được hiển thị lên màn hình thông qua gọi hàm vedagiac().

**Bước 3: Thuật toán tô màu đa giác scaline và chương trình chính.**

**Chương trình thuật toán Scanline**

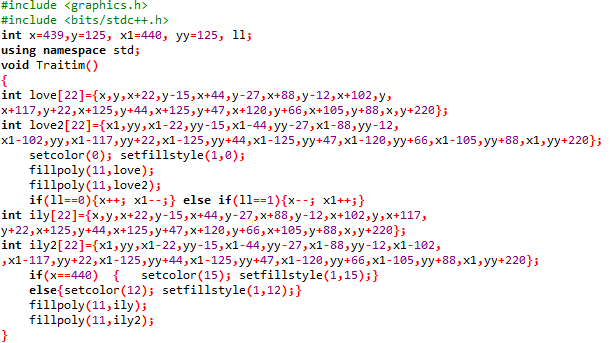
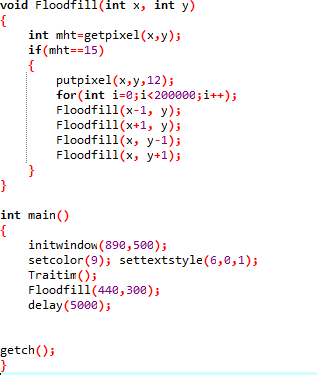




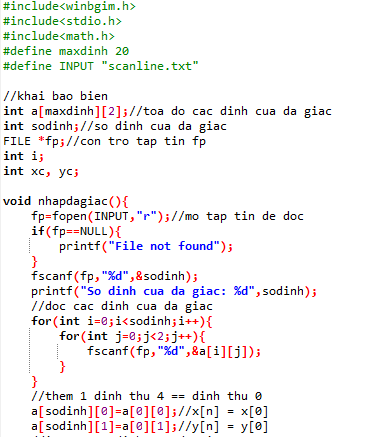
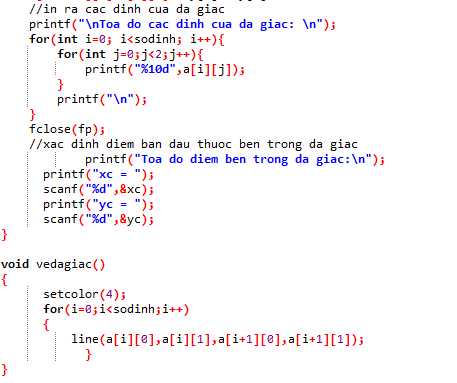
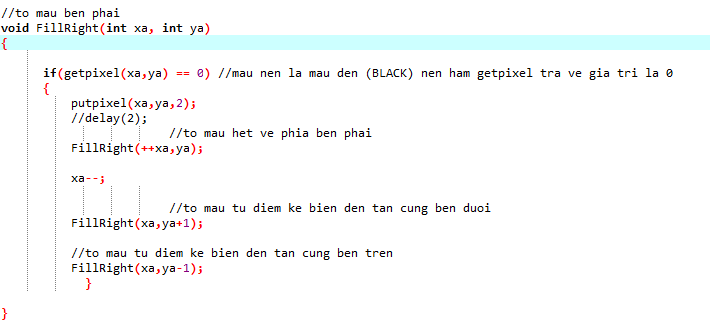
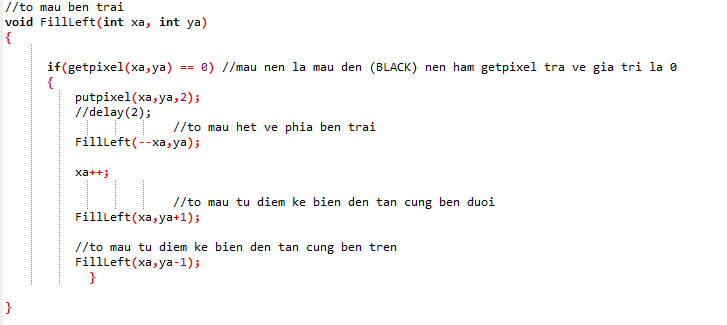
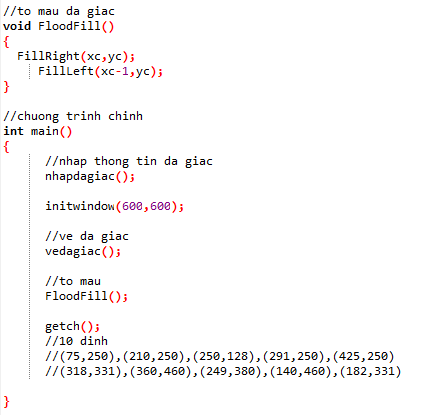


## **2.2 Cài đặt thuật toán Floodfill**

## **2.2.1 Cài đặt bằng phương pháp đệ quy**

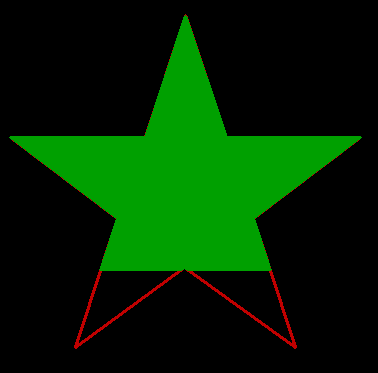
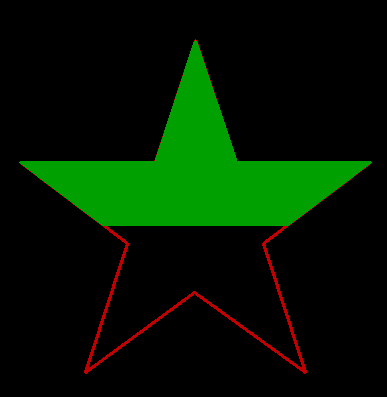
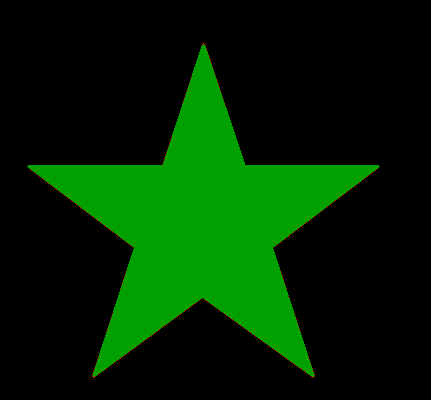
**** 

## **2.2.2 Cài đặt bằng phương pháp không đệ quy**

**** ****   

# CHƯƠNG 3: KẾT QUẢ

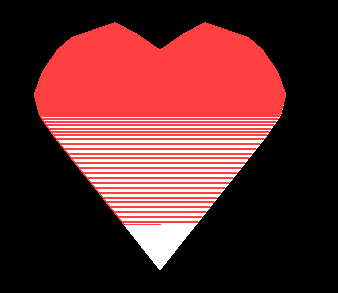
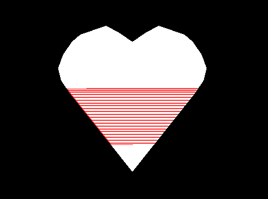
## **3.1 Tô màu Scanline**

**** ****

#### Hình 3.1. Kết quả thuật toán tô màu Scanline

Từ Hình 3.1 ta thấy, thuật toán bắt đầu từ đỉnh y=ymin, sẽ tô theo từng dòng theo hướng ymin đến ymax với giá trị màu tô là Green. Thuật toán sẽ dừng lại khi y=ymax.

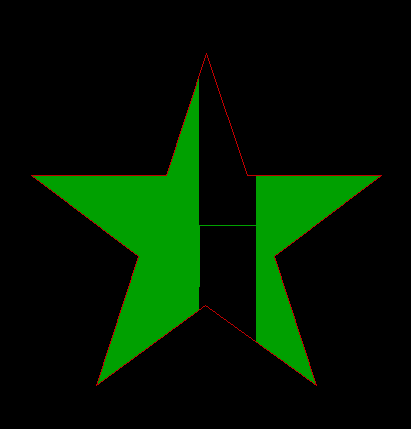
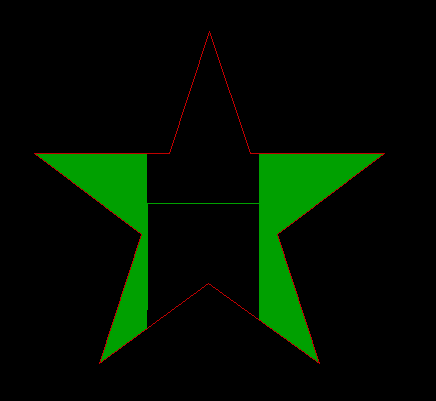
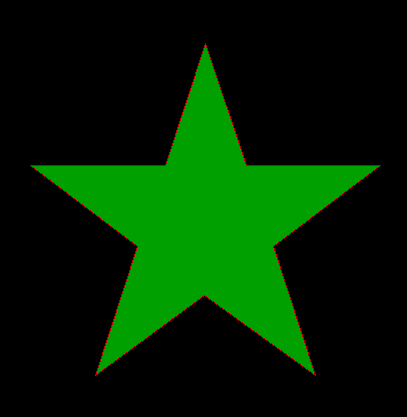
## **3.2 Tô màu Floodfill bằng đệ quy**

****

#### Hình 3.2. Kết quả thuật toán tô màu Floodfill đệ quy

Từ hình 3.2 ta thấy, thuật toán điều hướng để tô màu một vùng bên trong hình. Nếu pixel hiện tại có màu giống màu của pixel gốc ( điểm bắt đầu ), thì nó sẽ được tô màu và các pixel lân cận cũng được kiểm tra.

## **3.3 Tô màu Floodfill không đệ quy**

Hình 3.3. Kết quả thuật toán tô màu Floodfill không đệ quy

Hình 3.3 chọn điểm có tọa độ (300,300) là điểm bắt đầu, thuật toán, bắt đầu tô từ điểm bắt đầu loang dần sang các điểm cận trái, sau khi đụng biên sẽ loang ngược sang các điểm lân cận phải. Từ hai điểm cận biên, chọn điểm bắt đầu mới là hai điểm cận trên và dưới. Tuần tự cho tới khi không còn điểm nào chưa được tô.

# 

# TÀI LIỆU THAM KHẢO

1. Đoàn Vũ Thịnh, “Bài giảng Kỹ thuật đồ họa”, 2019, Đại học Nha Trang
2. <https://tuhoclaptrinh.cachhoc.net/2017/03/19/bai-8-thuat-toan-to-mau-scanline/> Tự học lập trình, 2017.
3. <https://tuhoclaptrinh.cachhoc.net/2017/03/22/bai-9-thuat-toan-to-mau-loang/>

Tự học lập trình, 2017.

1. Vũ Hải Quân, “Đồ họa máy tính”, 2007, NXB Đại Học Quốc Gia