TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN – TIN HỌC

--ભ📖ભ--

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: 6

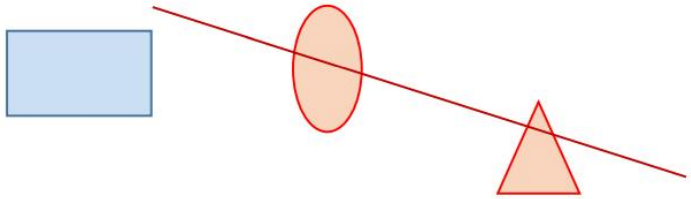
Giáo viên hướng dẫn : **ThS. NGUYỄN NGỌC LONG**

Sinh viên thực hiện : **TRẦN THỊ LÀNG ZING (1611349),**
PHÙNG THỊ ĐIỆP (19110281)

Tp. Hồ Chí Minh, ngày 13 tháng 7 năm 2021

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

Đề bài: Cho một danh sách các đối tượng hình học, mỗi đối tượng thuộc một trong các loại: Hình tròn, hình ellipse, hình bán nguyệt, hình đa giác (lồi), hình chữ nhật, hình vuông, hình tam giác. Cho một đường thẳng (ax + by = 0) trong mặt phẳng. Viết hàm cho biết đường thẳng cắt các hình nào của danh sách. Viết ứng dụng cho phép tạo các hình và một đường thẳng, vẽ các hình và đường thẳng, xuất thông báo cho biết đường đi qua bao nhiêu hình, tô màu các hình có đường cắt ngang với màu khác các hình còn lại. Trong hình minh họa bên dưới, đường thẳng cắt 2 hình.



Để giải quyết bài toán này ta phải tìm hiểu phần mềm bổ trợ cho việc đồ họa, OpenCV. Phân tích thuật toán trên cơ sở mô hình toán học sau đó vận dụng lập trình để giải quyết bài toán. Trong việc lập trình bao gồm các công việc, xây dựng lớp biểu diễn các đối tượng hình học, kiểm tra sự giao cắt nhau, xuất hình hay tô màu.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT THƯ VIỆN ĐỒ HỌA OpenCV

2.1. Tổng quan về thị giác máy tính:

- Thị giác máy tính là ngành học nghiên cứu cách tái tạo, ngắt và hiểu một cảnh 3D từ các hình ảnh 2D của nó, xét về các đặc tính của cấu trúc có trong cảnh đó.
- Thị giác máy tính được chia thành ba loại cơ bản như sau:

+ Low-level vision: Nó bao gồm hình ảnh quy trình để trích xuất tính năng.

+ Intermediate-level vision: Nó bao gồm nhận dạng đối tượng và giải thích cảnh 3D

+ High-level vision: Nó bao gồm mô tả khái niệm về một cảnh như hoạt động, ý định và hành vi.

- Các ứng dụng chính của thị giác máy tính :

- Robotics:

+ Localization-determine robot location automatically

+ Dẫn đường

+ Tránh chướng ngại vật

+ Lắp ráp (peg-in-hole, welding, painting)

+ Manipulation (eg. PUMA robot manipulator)

+ Tương tác với người máy (HRI): Robot thông minh để tương tác và phục vụ con người

- Y tế:

+ Classification và detection (vd : phân loại tổn thương hoặc tế bào và phát hiện khối u)

+ Phân đoạn 2D / 3D

+ Tái tạo nội tạng người 3D (MRI hoặc siêu âm)

+ Phẫu thuật robot có hướng dẫn thị giác

- Bảo mật:

+ Sinh trắc học (mống mắt, vân tay, nhận dạng khuôn mặt)

+ Giám sát - phát hiện một số hoạt động hoặc hành vi đáng ngờ

- Vận chuyển:

+ Xe tự hành

+ An toàn(ví dụ, giám sát cảnh giác của người lái xe)

- Ứng dụng tự động hóa công nghiệp:

+ Kiểm tra công nghiệp (phát hiện khuyết tật)

+ Assembly

+ Đọc mã vạch và nhãn gói

+ Phân loại đối tượng

+ Document understanding (eg. OCR)

Đối với Thị giác máy tính, có thể sử dụng một thư viện phổ biến có tên là OpenCV (Thị giác máy tính nguồn mở). Nó là một thư viện các chức năng lập trình chủ yếu nhắm vào thị giác máy tính thời gian thực hiện nó được viết bằng C ++ và giao diện chính của nó là C ++.

2.2. Giới thiệu về OpenCV:

- OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning. Nó miễn phí cho cả học tập và sử dụng với mục đích thương mại, có trên các giao diện C++, C, Python và Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android.
- Ứng dụng:

    + Hình ảnh street view.

    + Kiểm tra và giám sát tự động.

    + Robot và xe hơi tự lái

    + Phân tích hình ảnh y học

    + Tìm kiếm và phục hồi hình ảnh/video

    + Phim – cấu trúc 3D từ chuyển động

    + Nghệ thuật sắp đặt tương tác

Trong bài tập lớn này OpenCV cung cấp các hàm vẽ cơ bản như vẽ điểm, vẽ đường thẳng, đường tròn, elip, hình chữ nhật,…Mục đích là khi nhận dạng hoặc phát hiện được đối tượng thì vẽ lên ảnh output.

2.3. Cấu trúc lưu trữ hình ảnh Mat trong OpenCV

Hình ảnh thông thường sẽ được hình dung dưới một Matrix chính vì vậy mà OpenCV xây dựng class Mat để lưu trữ thông tin hình ảnh.

Ma trận sẽ gồm các dòng và các cột. Hình dùng như chiều cao của hình ảnh sẽ bằng đúng số rows của matrix và chiều rộng của hình ảnh sẽ bằng đúng số cột của matrix. Và phần tử

[row 0, colum 0] chính là đại diện cho 1-pixel của hình ảnh.

Và dưới đây là hình ảnh thực tế hơn về cách lưu hình ảnh ứng với bảng màu BGR. Chúng ta thấy tại phần tử thứ [row 0, column 0] là giá trị màu sắc của một pixel của hình ảnh. Và nó được lưu với không gian màu BGR nên mỗi pixel sẽ có 3 kênh màu kế tiếp nhau là B Blue, G

Green, R Red. Và đa số mỗi kênh màu sẽ được biểu diễn bằng 8bit unsinged char (uint8_t). Tương tự với các phần tử khác trong ma trận cũng có cấu trúc lưu trữ tương đương với phần tử

[row 0, column 0].

2.3.1. Sử dụng Mat trong OpenCV:

Constructor và ý nghĩa của chúng:

Lớp Mat trong OpenCV năm trong modules core của bộ thư viện OpenCV và năm trong **namespace cv** của bộ thư viện. Nếu chúng ta không khai báo sử dụng **namespace cv** ở đầu chương trình thì trong chương trình sử dụng phải bắt buộc có tiền tố cv đứng với đầu.

Ví dụ: muốn sử dụng Mat phải khai báo là cv::Mat mat;

Cú pháp: *Mat (int rows, int cols, int type);*

Ý nghĩa:

+ Tham số thứ nhất chính là số dòng của ma trận hay nói cách khác là chiều cao của hình ảnh.

+ Tham số thứ hai chính là số cột của trận hay cũng nói cách khác chính là chiều rộng của hình ảnh.

+ Tham số type thứ ba là tham số mà chúng ta nên quan tâm và hiểu rõ nó và type có cấu trức hư dưới đây.

CV_[Số bit cho 1 channel][Kiểu có dấu, không dấu, số thực]C[Số channel]

Ví dụ:

CV_8UC1: Có nghĩa là mỗi có 1 channel dùng 8bit không dấu để biểu diễn

CV_8UC3: Có nghĩa là mội pixel (một điểm ảnh) sẽ có 3 channel và ứng mới mỗi channel sẽ dùng 8bit không dấu để biểu diễn. (RGB, BRG,. . . .).

CV_8UC4: Có nghĩa là mội pixel (một điểm ảnh) sẽ có 4 channel và ứng mới mỗi channel sẽ dùng 8bit không dấu để biểu diễn. (ARGB, BRGA).

Cú pháp: *Mat(Size size, int type);*

Ý nghĩa: Tương tự như constructor trên như thay vì truyền vào rows và cols thì chúng ta truyền vào size với format Size(cols, rows).

Cú pháp: Mat(int rows, int cols, int type, const Scalar& s);

Ý nghĩa: Giống như contructor thứ nhất với đối số thứ ba là một option. Giá trị s có ý nghĩa là sẽ khởi tạo giá trị cho các phần tử trong Mat.

2.3.2. Các hàm vẽ cơ bản trong OpenCV:

Cú pháp khởi tạo:

- **Tạo ảnh màu** black với kích thước 400 × 300 pixels

cv :: M atimg = cv :: M at(300, 400, CV _8UC3); (1)

img = cv :: Scalar(0, 0, 0); (2)

- **Vẽ đường thẳng**: vẽ đường thẳng màu đỏ nối 2 điểm có tọa độ(x1; y1) và (x2; y2) , với nét đậm bằng 1.

Cú pháp:

cv :: line(img, cv :: Point(x1; y1), cv :: Point(x1; y2), cv :: Scalar(0, 0, 255), 1); (3)

Cú pháp hiển thị ảnh:

cv :: imshow("Image", img); (4)

cv :: waitKey(0); //Wait for a keystroke in the window (5)

- **Vẽ đường tròn**: vẽ đường tròn màu đỏ có tâm (x, y) và bán kính bằng R, nét đậm bằng 1.

Cú pháp:

cv :: circle(img, cv :: Point(x, y), R, cv :: Scalar(0, 0, 255), 1); (7)

- **Vẽ hình tròn**: vẽ hình tròn màu đỏ có tâm (x, y) và bán kính bằng R, nét đậm bằng 1.

Cú pháp:

cv :: circle(img, cv :: P oint(x, y), R, cv :: Scalar(0, 0, 255), −1); (8)

- **Vẽ đường elip**: Vẽ đường elip màu đỏ có tâm (x; y), trục nhỏ bằng r và trục lớn bằng R.

Cú pháp:

cv :: ellipse(img, cv :: P oint(x; y), cv :: Size(r, R), angle, startAngle, endAngle, cv :: Scalar(0, 0, 255), 1); (9)

Các hệ số angle, startAngle, endAngle được setup như sau:

- double angle = 90; (10)
- double startAngle = 90; (11)
- double endAngle = 270; (12)

- **Vẽ hình elip** màu đỏ có tâm (x; y), trục nhỏ bằng r và trục lớn bằng R.

Cú pháp:

cv :: ellipse(img, cv :: P oint(x; y), cv :: Size(r, R), angle, startAngle, endAngle, cv :: Scalar(0, 0, 255), −1); (13)

- **Vẽ đường chữ nhật:** Vẽ đường chữ nhật có điểm bắt đầu là (x, y) và kích thước là *width*×*length* px

cv :: rectangle(img, cv :: Rect(x, y, width, length), cv :: Scalar(0, 0, 255), 1); (14)

- **Vẽ hình chữ nhật:** Vẽ hình chữ nhật có điểm bắt đầu là (x, y) và kích thước là width×length px

cv :: rectangle(img, cv :: Rect(x, y, width, length), cv :: Scalar(0, 0, 255), −1); (15)

CHƯƠNG 3: THIẾT KẾ

3.1. Ý tưởng:

Với một cấu hình hình học cố định, chúng có thể được mô tả bằng các phương trình bậc 1, bậc 2 hoặc bậc cao, chẳng hạn hình chữ nhật có 4 cạnh có thể được mô tả dựa trên 4 đoạn thẳng được xây dựng nhờ phương trình bậc 1: $ax + \mathrm{b} = \mathrm{y}$. Tương tự cho hình tam giác, tổ hợp 2 điểm trong 3 điểm trên mặt phẳng sẽ tạo thành 3 đoạn thẳng .Hình tròn và hình elip có phương trình hình học riêng biệt. Nhắc lại phương trình đường tròn với tâm I có tọa độ (x_0, y_0) và độ lớn bán kinh R.

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

Đối với hình elip có tâm I có tọa độ (a, b) và độ lớn trục lớn a và trục bé b:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1$$

Vậy nên để kiểm tra một đường thẳng $ax + \mathrm{b} = \mathrm{y}$ có cắt một cấu trúc hình học cơ bản nào chỉ cần xét liệu có tồn tại giao điểm giữa đường thẳng có phương trình $ax + \mathrm{b} = \mathrm{y}$ với các cấu trúc đại số mô tả cấu trúc hình học tương ứng.

3.2. Phương pháp Mô hình toán học:

3.2.1. Kiểm tra đường thẳng cắt hình tròn:

Ta xét đường tròn với tâm I có tọa độ (x_0, y_0) và độ lớn bán kinh R với đường thẳng $ax + \mathrm{b} = \mathrm{y}$.

Mô hình toán học:

$$\begin{cases} (x - x_0)^2 + (y - y_0)^2 = R^2 & (I) \\ ax + \mathrm{b} = \mathrm{y} & (\mathrm{II}) \end{cases}$$

Khi đường thẳng cắt đường tròn nghĩa là hệ phương trình trên có tồn tại nghiệm.

Thế (II) vào (I) ta được:

$$(x - x_0)^2 + (ax + \mathrm{b} - y_0)^2 = R^2 \qquad (III)$$

Đặt hằng số $c = \mathrm{b} - y_0$, (III) trở thành:

$$(1 + a^2)x^2 + 2(ac - x_0)x + x_0^2 + c^2 - R^2 = 0$$

Đặt:

$$\alpha_1 = 1 + a^2;\ \alpha_2 = ac - x_0;\ \alpha_3 = x_0^2 + c^2 - R^2$$

Ta xét biệt thức Δ để biện luận nghiệm, nếu $\Delta \geq 0$ đường thẳng cắt đường tròn, và ngược lại.

$$\Delta = \alpha_2^2 - \alpha_1\alpha_3$$

3.2.2. Kiểm tra đường thẳng cắt hình elip:

$$\begin{cases} \dfrac{(x-x_0)^2}{a^2}+\dfrac{(y-y_0)^2}{b^2}=1 & (I) \\ Ax+\mathrm{B}\ =\ \mathrm{y} & (\mathrm{II}) \end{cases}$$

Thế (II) vào (I) ta được:

$$\frac{(x-x_0)^2}{a^2}+\frac{(ax+\mathrm{b}-y_0)^2}{b^2}=1$$

Đặt hằng số $c=\mathrm{b}-y_0$, (III) trở thành:

$$(b^2+a^2A^2)x^2+2(Aa^2c-b^2x_0)x+b^2x_0^2+a^2c^2-a^2b^2=0$$

Đặt:

$$\alpha_1=b^2+a^2A^2;\ \alpha_2=Aa^2c-b^2x_0;\ \alpha_3=b^2x_0^2+a^2c^2-a^2b^2$$

Ta xét biệt thức Δ để biện luận nghiệm, nếu $\Delta\geq 0$ đường thẳng cắt đường tròn, và ngược lại.

$$\Delta=\alpha_2^2-\alpha_1\alpha_3$$

3.2.3. Kiểm tra 2 đường thẳng cắt nhau:

Với 2 điểm phân biệt $A(x_1,y_1)$ và $B(x_2,y_2)$. Phương trình đường thẳng đi qua 2 điểm là:

$$(y_1-y_2)(x-x_1)+(x_2-x_1)(y-y_1)=0$$

Ta tìm giao điểm của 2 đường thẳng như sau:

$$\begin{cases} (y_1-y_2)(x-x_1)+(x_2-x_1)(y-y_1)=0 \\ ax+\mathrm{b}\ =\ \mathrm{y} \end{cases}$$

Gọi : $\alpha_1=y_1-y_2,\ \alpha_2=x_2-x_1$

Ta được nghiệm : $x=\frac{\alpha_1x_1+\alpha_2x_2-\alpha_2b}{\alpha_1+\alpha_2a};y=ax+\mathrm{b}$

3.3. Quy trình thực hiện:

3.3.1 giữa đường thẳng và đường tròn:

```
class DrawCircle2
{
public:
	DrawCircle2();
	bool inputCircle2(int a, int b);
	void drawCircle2(Mat img);
	void drawCircle3(Mat img);
private:
	int  x0, y0, r0;
};
```

Tạo class `DrawCircle2` bao gồm các hàm:

```
bool DrawCircle2::inputCircle2(int a, int b) {
	cout << "Tam x0 cua  hinh tron : "; cin >> x0;
	cout << "Tam y0 cua  hinh tron : "; cin >> y0;
	cout << "Nhap ban kinh hinh tron : "; cin >> r0;
	int c = b - y0;
	int alpha1 = 1 + a * a;
	int alpha2 = a * c - x0;
	float alpha3 = x0 * x0 - r0 * r0 + c * c;
	float alpha = alpha2 * alpha2 - alpha3 * alpha1;
	printf("alpha=%f \n", alpha);
	if (alpha >= 0) {
		return true;
	}
	else return false;
}
```

⇨ Nhập các thông số liên quan đến hình học, tâm đường tròn, bán kính, xét xem có giao nhau giữa đường thẳng và đường tròn.

```
    DrawCircle2::DrawCircle2()
    {
       this->r0 = r0;
    }

    void DrawCircle2::drawCircle2(Mat img) {
       cv::circle(img, Point(x0, y0), r0, cv::Scalar(0, 0, 255), 1);
    }

    void DrawCircle2::drawCircle3(Mat img) {
       cv::circle(img, Point(x0, y0), r0, cv::Scalar(0, 0, 255), -1);
    }
```

⇨ - con trỏ **this** có thể tham chiếu tới đối tượng bán kính r0.
- cho vẽ đường tròn hoặc hình tròn nếu không cắt và ngược lại.

3.3.2 giữa đường thẳng và elip:

```
class DrawEliptic
{
public:
	DrawEliptic();
	int inputEliptic(int A, int B);
	void drawEliptic(Mat img);
	void drawEliptic2(Mat img);

private:
	int  x0, y0, a, b;
};
```

⇨ Tạo class `DrawEliptic` bao gồm các hàm:

```
int DrawEliptic::inputEliptic(int A, int B) {
	cout << "Tam x0 cua elip: "; cin >> x0;
	cout << "Tam y0 cua  elip: "; cin >> y0;
	cout << "truc y (truc be) elip: "; cin >> b;
	cout << "truc x (truc lon): "; cin >> a;
	int c = B - y0;
	//printf("c=  %d - %d=%d \n", B, y0, c);
	float alp1 = b*b + a*a * A * A;
	//printf("alp1= %d + %d * %d * %d =%f \n", b , a , A , A,alp1);
	float alp2 = A * a * c*a - b*b * x0;
	float alp3 = x0 * x0 * b*b + a*a * c * c - a*a*b * b;
	float alpha = alp2 * alp2 - alp3 * alp1;
	//printf("alp2= %d * %d * %d - %d * %d =%f \n", A ,a , c , b , x0,alp2);
	//printf("alp3= %d * %d * %d + %d * %d * %d - %d * %d =%f \n", x0 , x0 , b ,a ,
c , c , a , b, alp3);
	//printf("alpha=%f* %f - %f * %f =%f \n", alp2 , alp2 , alp3 , alp1, alpha);
	//printf("alpha=%f \n", alpha);
	if (alpha >= 0) {
		return 1;
	}
	else return 0;
}
```

⇨ Nhập các thông số liên quan đến hình học, tâm elip, trục bé, trục lớn, xét xem có giao nhau giữa đường thẳng và elip.

```
DrawEliptic::DrawEliptic()
{
	this->x0 = x0;
	this->y0 = y0;
	this->a = a;
	this->b = b;
}

void DrawEliptic::drawEliptic(Mat img) {
	double angle = 90;
	double startAngle = 90;
	double endAngle = 470;
	cv::ellipse(img, cv::Point(x0, y0), cv::Size(b,a), angle, startAngle, endAngle,
cv::Scalar(0, 0, 255), 1);

}

void DrawEliptic::drawEliptic2(Mat img) {
	double angle = 90;
	double startAngle = 90;
	double endAngle = 470;
	cv::ellipse(img, cv::Point(x0, y0), cv::Size(b, a), angle, startAngle, endAngle,
cv::Scalar(0, 255, 0), -1);
```

```
}
```

⇨ - con trỏ **this** có thể tham chiếu tới đối tượng ta đang gọi.
- cho vẽ đường tròn hoặc hình tròn nếu không cắt và ngược lại.

3.3.3 giữa đường thẳng và đoạn thẳng tạo bởi 2 điểm:

```
class consider
{
public:
   consider(int a, int b){
          x = a;
          y = b;
   }
   int intersect(int a, int b, consider zz5, consider zz6);
private:
   int x, y;
};
```

⇨ Tạo lớp (đối tượng) consider cho phép input một đối tượng-điểm 2 chiều Point(x,y). Function intersect kiểm tra có tồn tại điểm cắt nhau giữa đoạn thẳng tạo bởi 2 điểm zz5 và zz6 với y=ax+b.

```
int consider::intersect(int a, int b, consider z5, consider z6) {

int alp1, alp2, slope, x, y;
//alp1 = y1 - y2;
alp1 = z5.y - z6.y;
alp2 = z6.x - z5.x;
if (alp2 == 0)
{
       slope = 0;
}
else
{
       slope = -alp1 / alp2;
}
if (slope == a)
{
       printf(" line is parallel \n ");
       return 0;
}
else
{
       x = (alp1 * z5.x + alp2 * z5.y - alp2 * b) / (alp1 + alp2 * a);
       y = a * x + b;
       //printf("toa do giao : x=%d,   y=%d  \n", x, y);
       consider z77(x,y);
       //printf("z7.x=%d \n", z77.x);
       //printf("z7.y=%d \n", z77.y);
       //return z7.x, z7.y;
       int z = 0;
```

```
		//printf("zz5.x=%d \n", zz5.x);
		if (z5.x <= z6.x)
		{
			if (z77.x >= z5.x && z77.x <= z6.x)
			{
				z = z + 1;
				//printf("z=%d \n", z);
			}
		}
		else
		{
			if (z77.x >= z6.x && z77.x <= z5.x)
			{
				z = z + 1;
				//printf("z=%d \n", z);
			}
		}
		if (z5.y <= z6.y)
		{
			if (z77.y >= z5.y && z77.y <= z6.y)
			{
				z = z + 1;
				//printf("z=%d \n", z);
			}
		}
		else
		{
			if (z77.y >= z6.y && z77.y <= z5.y)
			{
				z = z + 1;
				//printf("z=%d \n", z);
			}
		}
		//printf("z truoc dieu kien if, z=%d \n", z);
		return z;
	}

}
```

Dựa trên lớp consider phía trên ta khảo sát liệu đường thẳng y=ax+b có cắt hình chữ nhật.

⇨ *3.3.4 giữa đường thẳng và hình chữ nhật:*

```
int DrawRetangle::inputRetangle(int a, int b) {
	cout << "Original point Retangle x0 = : "; cin >> x0;
	cout << "Original point Retangle y0 =: "; cin >> y0;
	cout << "Length of Retangle =: "; cin >> length;
	cout << "Width of Retangle =: "; cin >> width;
	// consider have been intersected:
	int x1, x2, x3, y1, y2, y3;
	x1 = x0 + length; y1 = y0;
	x2 = x0 + length; y2 = y0 + width;
	x3 = x0; y3 = y0 + width;
```

```
        int z, z1 = 0;
        int alp1 = 0;
        int alp2 = 0;
        int slope = 0;
        int x = 0, y = 0;
        int n = 4;
        int zad = 0;
        int AD[2][4] = { {x0, x1, x2, x3 }, {y0, y1, y2, y3} };

        for (int zad = 0; zad < n-1; zad++)
        {
                printf("iteration i= %d \n", zad);
                consider z4(AD[0][zad], AD[1][zad]), z5(AD[0][zad + 1], AD[1][zad + 1]);
                consider zz(1, 1);
                //zz.intersect(a, b, z4, z5);
                if (zz.intersect(a, b, z4, z5) >= 2)
                {
                        z1 = z1 + 1;
                }
        }
        consider z4(AD[0][0], AD[1][0]), z5(AD[0][n - 1], AD[1][n - 1]);
        consider zz(1, 1);
        if (zz.intersect(a, b, z4, z5) >= 2)
        {
                z1 = z1 + 1;
        }

        printf("TOTAL z1=%d \n", z1);
        return z1;
};
```

⇨ Khởi tạo hình chữ nhật, xét khả năng cắt nhau của các cấu trúc hình học đại số.

```
    DrawRetangle::DrawRetangle()
    {
       this->x0 = x0;
       this->y0 = y0;
       this->length = length;
       this->width = width;
    }

    void DrawRetangle::drawRetangle(Mat img) {
       cv::rectangle(img, cv::Rect(x0, y0, length, width), cv::Scalar(255, 0, 0), 1);
    }

    void DrawRetangle::drawRetangle2(Mat img) {

       cv::rectangle(img, cv::Rect(x0, y0, length, width), cv::Scalar(255, 0, 0), -1);
    }
```

⇨ Phát họa và tô màu hình học.

3.3.5 giữa đường thẳng và đa giác lồi:

Pha khởi tạo và phân tích có sự tương giao giữa đường thẳng y=ax+b với các cấu trúc hình học đa giác lồi như hình chữ nhật, hình vuông, tam giác, polygon đều tương tự như nhau (như phần hình chữ nhật vừa trình ở trên) nên để tránh dài dòng không cần thiết sẽ không đề cập nữa.

Dưới đây là phần phân tích tổng quát cho cấu trúc hình học đa giác lồi.

```
class DrawPolygone
{
public:
	DrawPolygone();
	int inputPolygone(int a, int b);
	void drawPolygone(Mat img);
	void drawPolygone2(Mat img);
private:
	int n;
	int AP[2][100];
};
```

⇨ Khởi tạo lớp đa giác.

```
	int DrawPolygone::inputPolygone(int a, int b) {
	cout << "the number vertex of polygon " << endl; cin >> n;
	for (int i = 0; i < n; i++)
	{
		cout << " diem thu i= "<< endl << i ;
		cout << "x= " ; cin >> AP[0][i];
		cout << "y= " ; cin >> AP[1][i];
	}
	int z,z1=0 , alp1, alp2, slope;
	int x = 0, y = 0;
	for (int i = 0; i < n - 1; i++)
	{
		consider z4(AP[0][i], AP[1][i]), z5(AP[0][i + 1], AP[1][i + 1]);
		consider zz(1, 1);
		if (zz.intersect(a, b, z4, z5) >= 2)
		{
			z1 = z1 + 1;
		}
	}
	consider z4(AP[0][0], AP[1][0]), z5(AP[0][n - 1], AP[1][n - 1]);
	consider zz(1, 1);
	if (zz.intersect(a, b, z4, z5) >= 2)
	{
		z1 = z1 + 1;
	}
	printf("TOTAL z1=%d \n", z1);
	return z1;
    }
```

⇨ Khởi tạo đa giác đồng thời phân tích sự tương giao.

```
    DrawPolygone::DrawPolygone()
    {
       this->AP[2][100]= AP[2][100];
       this->n = n;
    }

    void DrawPolygone::drawPolygone(Mat img) {
       int lineType = cv::LINE_8;

       cv::Point arr_points[1][5];
       for (int i = 0; i < n; i++)
       {
              arr_points[0][i] = cv::Point(AP[0][i], AP[1][i]);
       }
       const cv::Point* ppt[1] = { arr_points[0] };
       int npt[] = { n };
       cv::Scalar color = cv::Scalar(255, 255, 255);
       cv::fillPoly(img,
              ppt,
              npt,
              1,
              color,
              lineType);
    }
    /**/
    void DrawPolygone::drawPolygone2(Mat img) {
       for (int i = 0; i < n-1; i++)
       {
              line(img, Point(AP[0][i], AP[1][i]), Point(AP[0][i+1], AP[1][i+1]),
    cv::Scalar(255, 255, 255), 1);
       }
       line(img, Point(AP[0][n-1], AP[1][n-1]), Point(AP[0][0], AP[1][0]),
    cv::Scalar(255, 255, 255), 1);
    }
```

⇨ Phác họa hình ảnh đa giác đồng thời tô màu nếu có sự giao nhau.

Main code:

```
int main() {
       Mat img = Mat(500, 800, CV_8UC3);
       img = cv::Scalar(0, 0, 0);

       DrawCircle2 c2;
       int x = 0, y = 0, h = 0, k = 0;
       int a, b, A_max = 500, B_max = 800, x1_line, y1_line, x2_line, y2_line;
       cout << "Phuong trinh y=Ax+B" << endl;
       cout << "Nhap A: "; cin >> a;
```

```
	cout << "Nhap B: "; cin >> b;
	int Z[2][10]; //  dong1=> toa do x, dong 2=> toa do y;
	int i = 0;
	//-------------------------------------------------------------
	// case1: x=0
	int y1;
	y1 = b;
	printf("giao voi x=0:  y1=%d \n", y1);
	if (b >= 0 && b <= B_max)
	{
		Z[0][i] = 0;
		Z[1][i] = y1;
		i = i + 1;
	}

	// case2: x=A_max;
	int y2;
	y2 = a * A_max + b;
	printf(" giao voi x=%d:    y2=%d \n", A_max, y2);
	if (y2 >= 0 && y2 <= B_max)
	{
		Z[0][i] = A_max;
		Z[1][i] = y2;
		i = i + 1;
	}
	// case3: y=0;
	int x3;
	x3 = -b / a;
	printf(" giao voi y=0:    x3 = %d \n", x3);

	if (x3 >= 0 && x3 <= A_max)
	{
		Z[1][i] = 0;
		Z[0][i] = x3;
		i = i + 1;
	}
	// case4: y=B_max;
	int x4;
	x4 = (B_max - b) / a;
	printf(" giao voi y=%d:    x4 = %d \n", B_max, x4);

	if (x4 >= 0 && x4 <= A_max)
	{
		Z[1][i] = B_max;
		Z[0][i] = x4;
		i = i + 1;
	}

	printf("finally i=%d \n", i);
	int z = i;

	for (int i = 0; i < 2; i++)
	{
		for (int j = 0; j < z; j++)
			printf("%d\t", Z[i][j]);
		printf("\n");
	}
```

```
	x1_line = Z[0][0];
	y1_line = Z[1][0];
	x2_line = Z[0][1];
	y2_line = Z[1][1];

	//-------------------------------------------------------------
	// ve duong thang qua 2 diem:

	line(img, Point(x1_line, y1_line), Point(x2_line, y2_line), Scalar(0, 0, 255),
1);


	/**/
	// VE DUONG TRON:
	printf("VE DUONG TRON: \n");
	if (c2.inputCircle2(a, b) == true)
	{
		cout << "Duong thang cat duong tron hahaha " << endl;
		c2.drawCircle3(img);
	}
	else
	{
		c2.drawCircle2(img);
	}

	// VE HINH CHU NHAT:
	printf("VE HINH CHU NHAT \n");

	DrawRetangle d;

	//printf("z=%d \n", d);
	if (d.inputRetangle(a, b) != 0)
	{
		cout << "Duong thang cat Retangle hahaha " << endl;
		d.drawRetangle2(img);
	}
	else
	{
		printf("hem co cat  \n");
		d.drawRetangle(img);
	}

	//

	// VE HINH VUONG:
	printf("VE HINH VUONG: \n");

	DrawSquare s;

	//printf("z=%d \n", d);
	if (s.inputSquare(a, b) != 0)
	{
		cout << "Duong thang cat Square hahaha " << endl;
		s.drawSquare2(img);
	}
	else
	{
		printf("hem co cat  \n");
```

```
                s.drawSquare(img);
        }

        //VE ELLIPTIC:
        printf("VE ELLIPTIC: \n");
        DrawEliptic e;
        if (e.inputEliptic(a, b) == 1)
        {
                cout << "Duong thang cat ellip hahaha " << endl;
                e.drawEliptic2(img);
        }
        else
        {
                cout << "Duong thang KO cat ellip hahaha " << endl;
                e.drawEliptic(img);
        }

        // VE TAM GIAC:
        printf("VE TAM GIAC: \n");
        DrawTriangle t;
        if (t.inputTriangle(a, b) ==true)
        {
                cout << "Duong thang cat Triangle hahaha " << endl;
                t.drawTriangle2(img);
        }
        else
        {
                cout << "Duong thang KO cat Triangle hahaha " << endl;
                t.drawTriangle(img);
        }
        //t.drawTriangle(img);



        // VE POLYGON:
        printf("VE DA GIAC \n");

    DrawPolygone p;


if (p.inputPolygone(a, b)>1)
{
        printf("duong thang cat da giac \n");
        p.drawPolygone(img);
}
else
{
        printf("duong thang KO cat da giac \n");
        p.drawPolygone2(img);
}

//ImpShow(img);
        cv::imshow("Image", img);
        cv::waitKey(0);
        return 0;
}
```

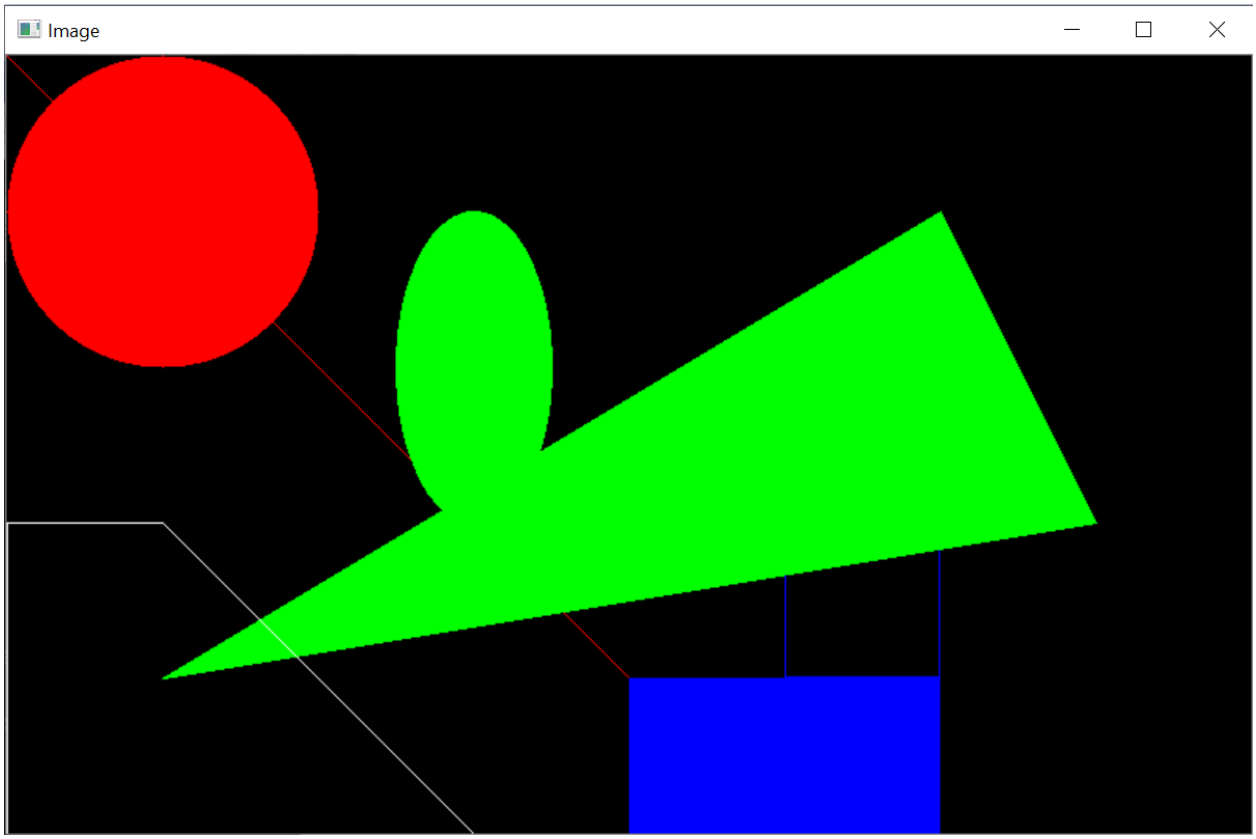
CHƯƠNG 4: KẾT QUẢ

Các dữ liệu input:

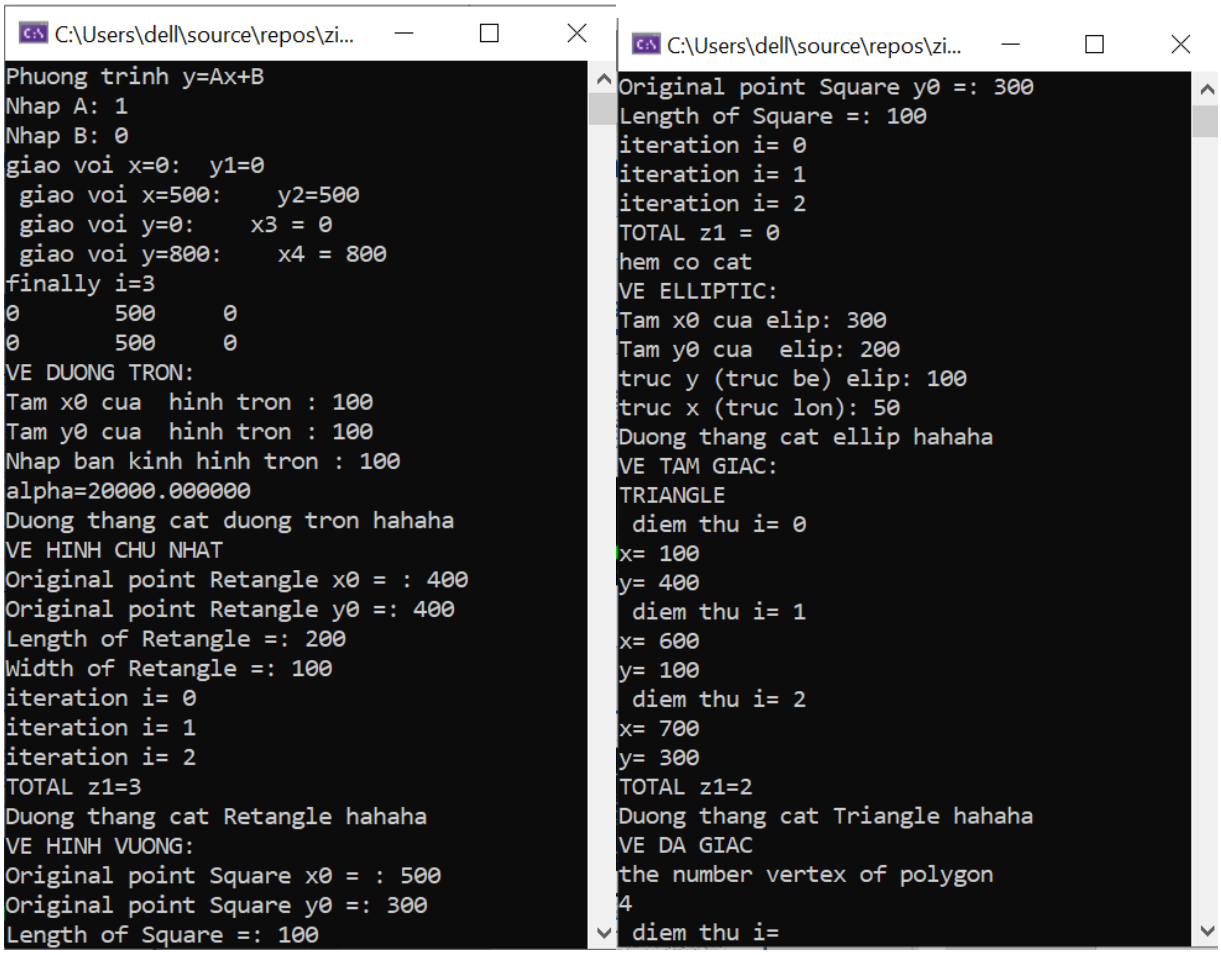
- Phương trình đường thẳng y=x: a=1, b=0.
- Đường tròn có tâm I(100; 100), R=100.
- Hình chữ nhật bắt đầu tại điểm (400, 400) chiều dài length = 200, chiều rộng width=100.
- Hình vuông bắt đầu tại điểm (500, 300) độ dài cạnh 100.
- Tọa độ cho elip (300, 200), trục bé (theo trục Oy) =100, trục lớn (theo trục Ox)=50.
- Hình tam giác được tạo bởi 3 đỉnh (100, 400), (600, 100), (700, 300).
- Với chương trình vẽ đa giác trong hình vẽ tư giác n=4, các đỉnh lần lượt là (0, 300), (0, 500), (300, 500), (100, 300).

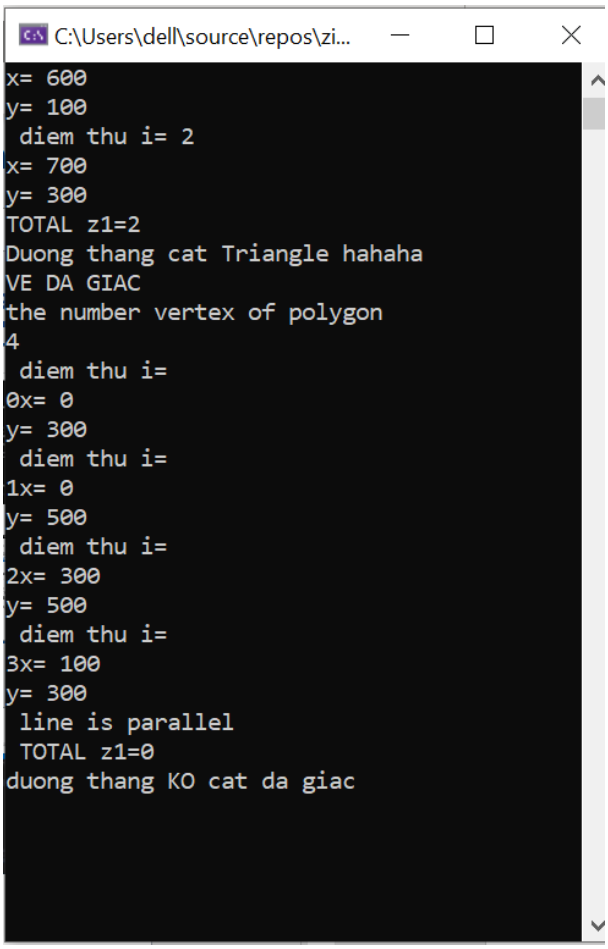
Dựa theo chương trình, thì đường thẳng y=x cắt hình tròn, elip, tam giác và hình chữ nhật.

Đa giác và hình vuông không bị cắt.



Ảnh từ console window:




C:\Users\dell\source\repos\zi...
x= 600
y= 100
diem thu i= 2
x= 700
y= 300
TOTAL z1=2
Duong thang cat Triangle hahaha
VE DA GIAC
the number vertex of polygon
4
diem thu i=
0x= 0
y= 300
diem thu i=
1x= 0
y= 500
diem thu i=
2x= 300
y= 500
diem thu i=
3x= 100
y= 300
line is parallel
TOTAL z1=0
duong thang KO cat da giac

CHƯƠNG 5: CÀI ĐẶT

Hướng dẫn download và install OpenCV:

Với phiên bản OpenCV 4.2.0, ta tải OpenCV một cách nhanh chóng và dễ dàng hơn.

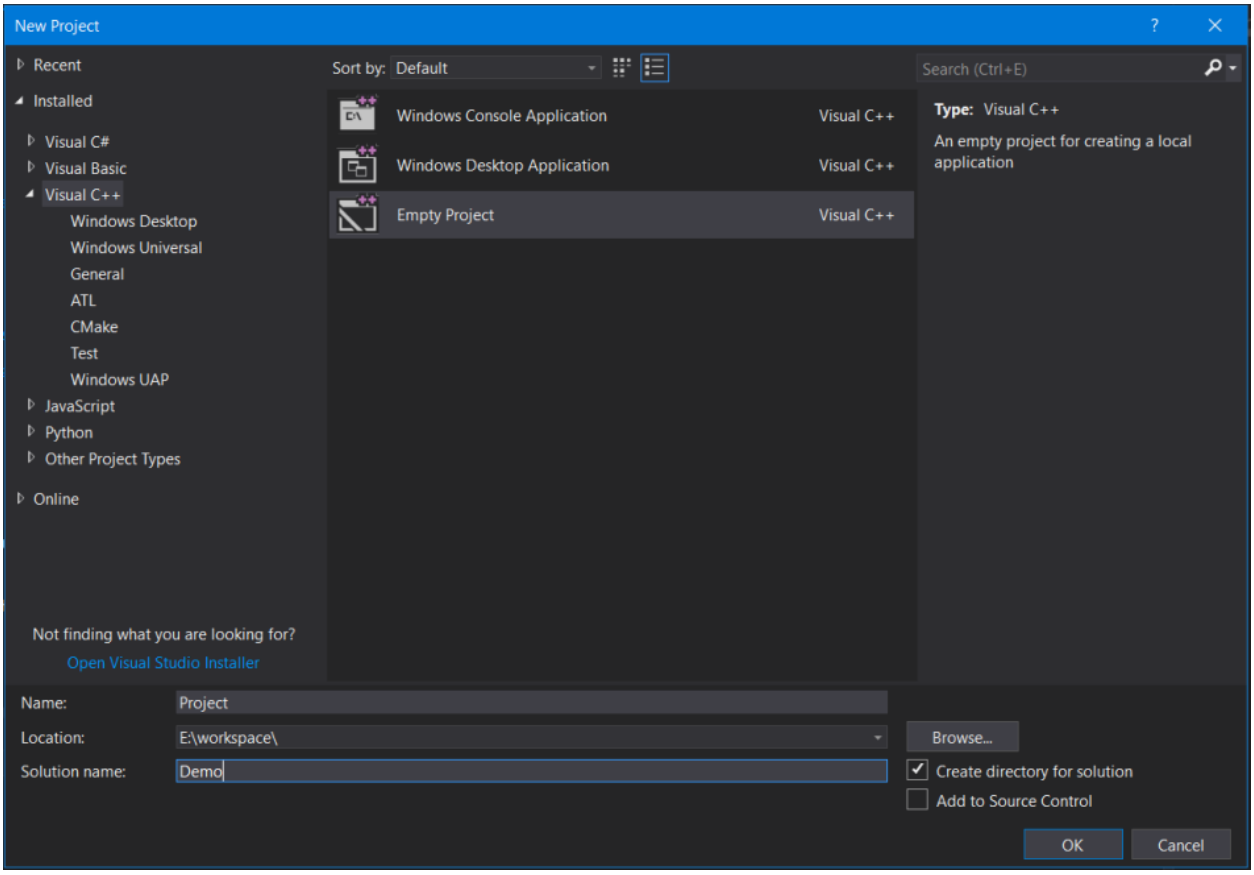
Link download OpenCV 4.2.0 về máy: https://opencv.org/opencv-4-2-0/?fbclid=IwAR2B_Sg8QvHGR2F1XzuR5BlnwYjWR31neZFSmVsDx14AMugveWEOaLh3VL8

Bấm vào "Win pack" để download bản OpenCV đã được build thành các file thư viện .lib và .dll cho Windows từ source code.

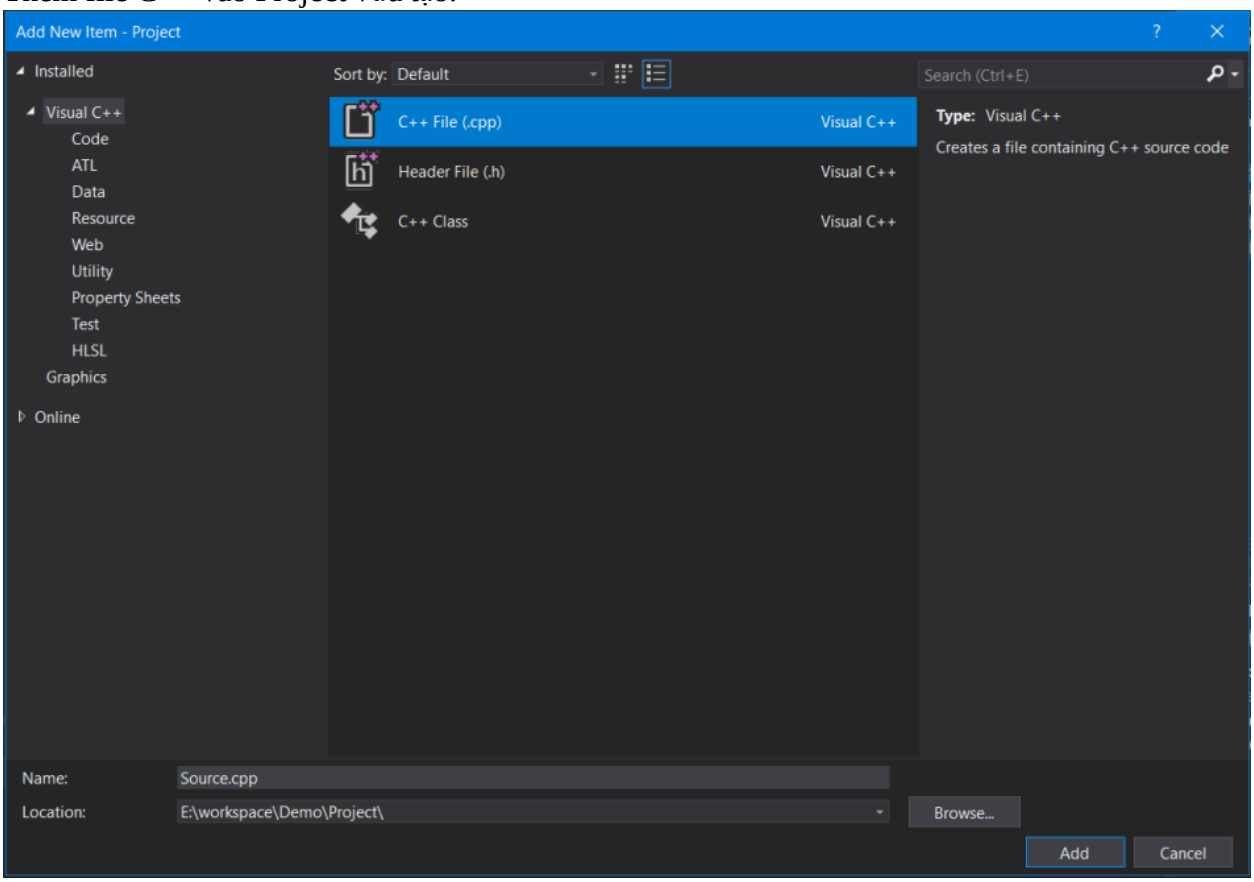
Sau khi download xong, chạy file này như file cài đặt thông thường, chọn đường dẫn mà OpenCV sẽ được giải nén ra. Nên chọn nơi giải nén sao cho dễ dàng tìm thấy và tránh trường hợp lỡ tay xóa mất, phải cài đặt lại.

Sau khi chọn xong đường dẫn, bấm vào Extract để bắt đầu giải nén.

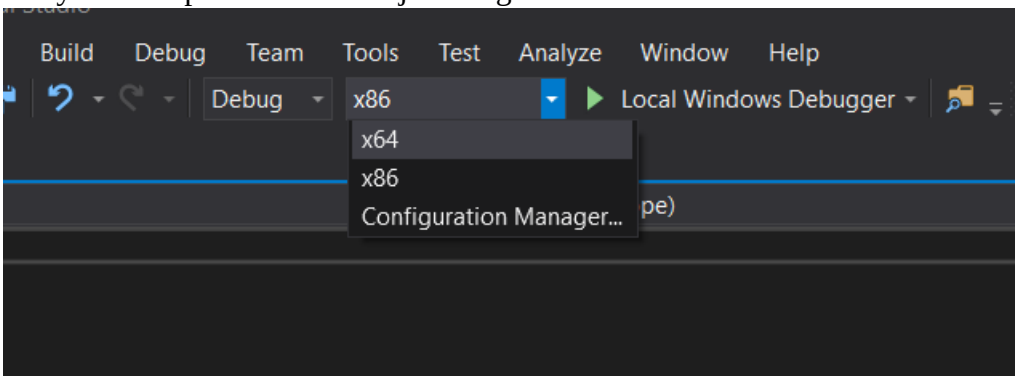
Sau khi tải xong OpenCV 4.2.0 . Ta mở Visual Studio lên và tạo empty Project Visual C++ mới.



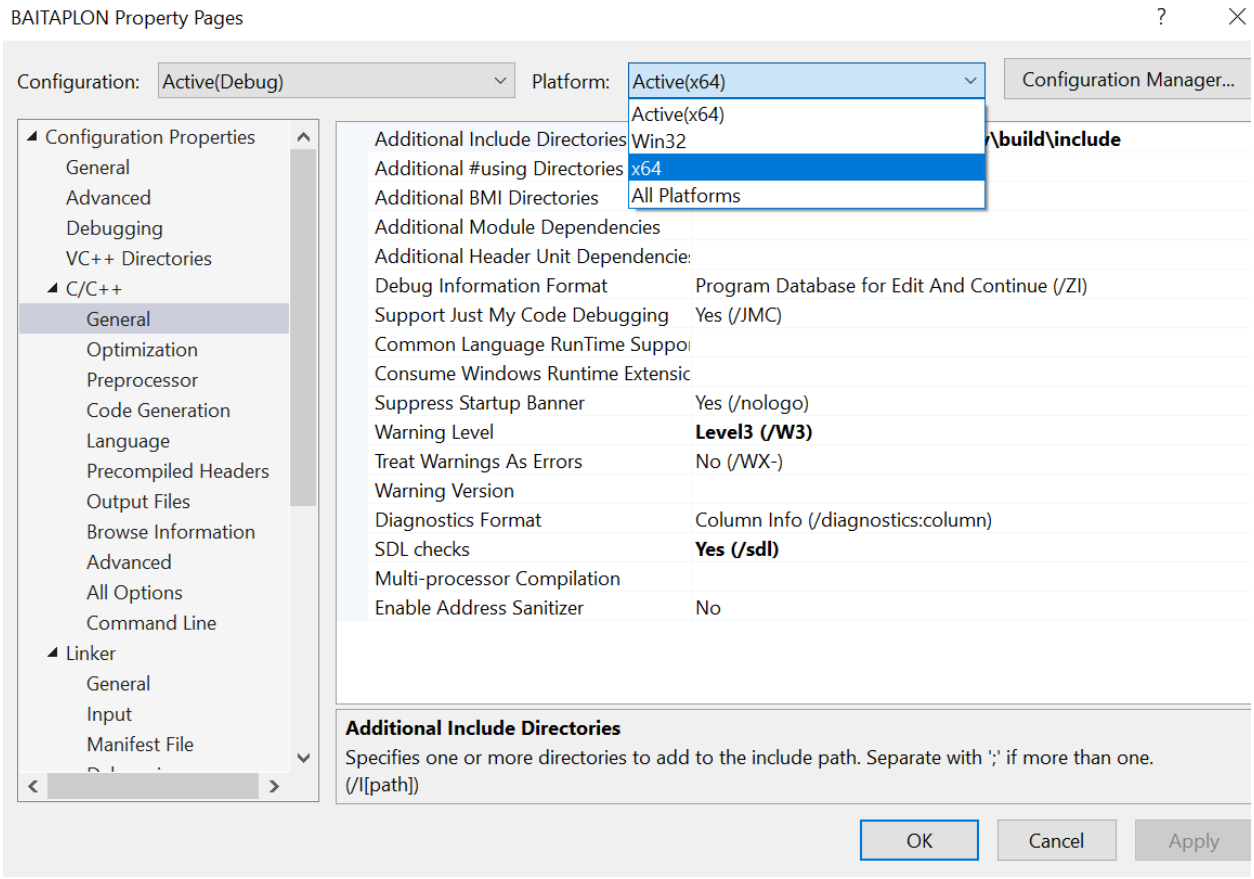
Thêm file C++ vào Project vừa tạo.



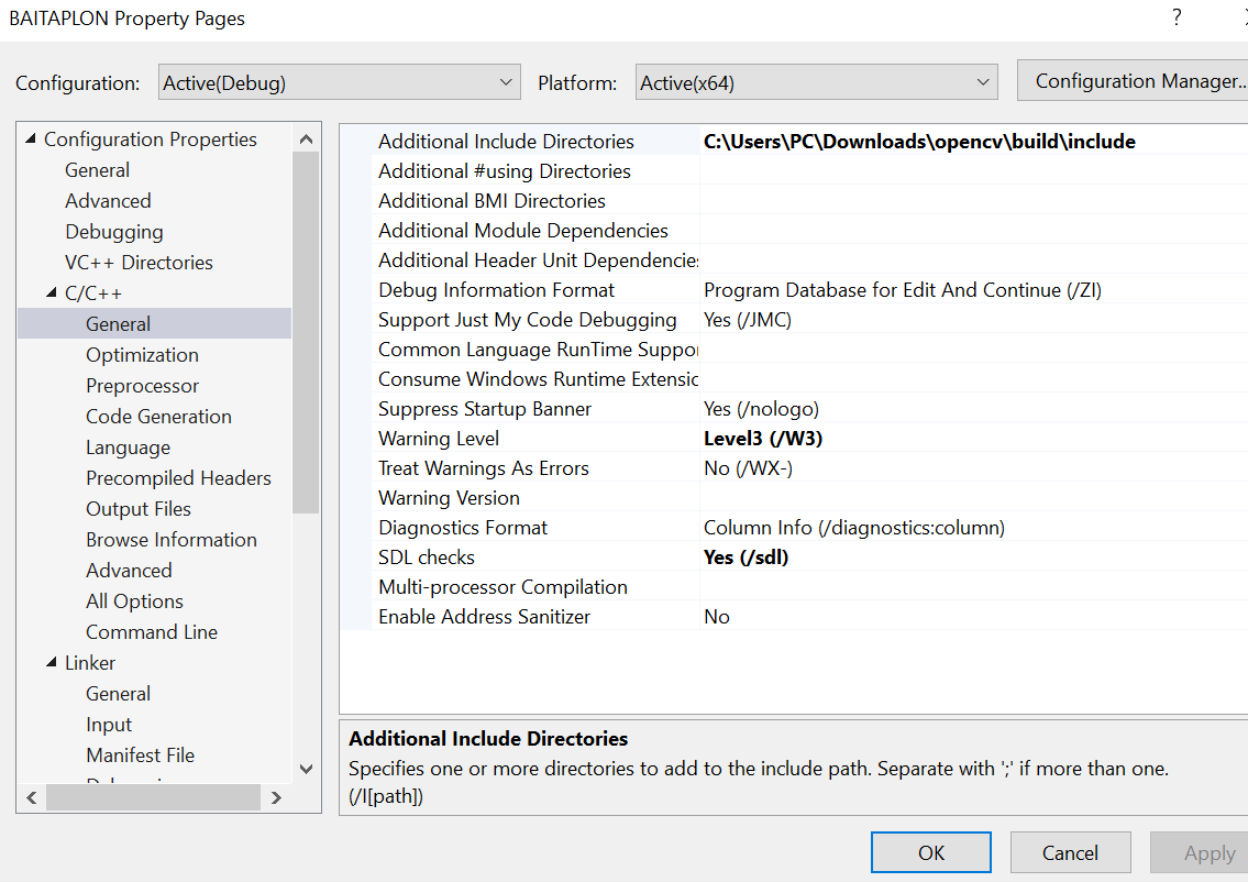
Chuyển build platform của Project sang x64.



Đến đây sẽ bắt đầu những bước quan trọng nhất. Click chuột phải vào Project và chọn Property để mở cửa sổ Project Property. Chuyển Platform sang x64.

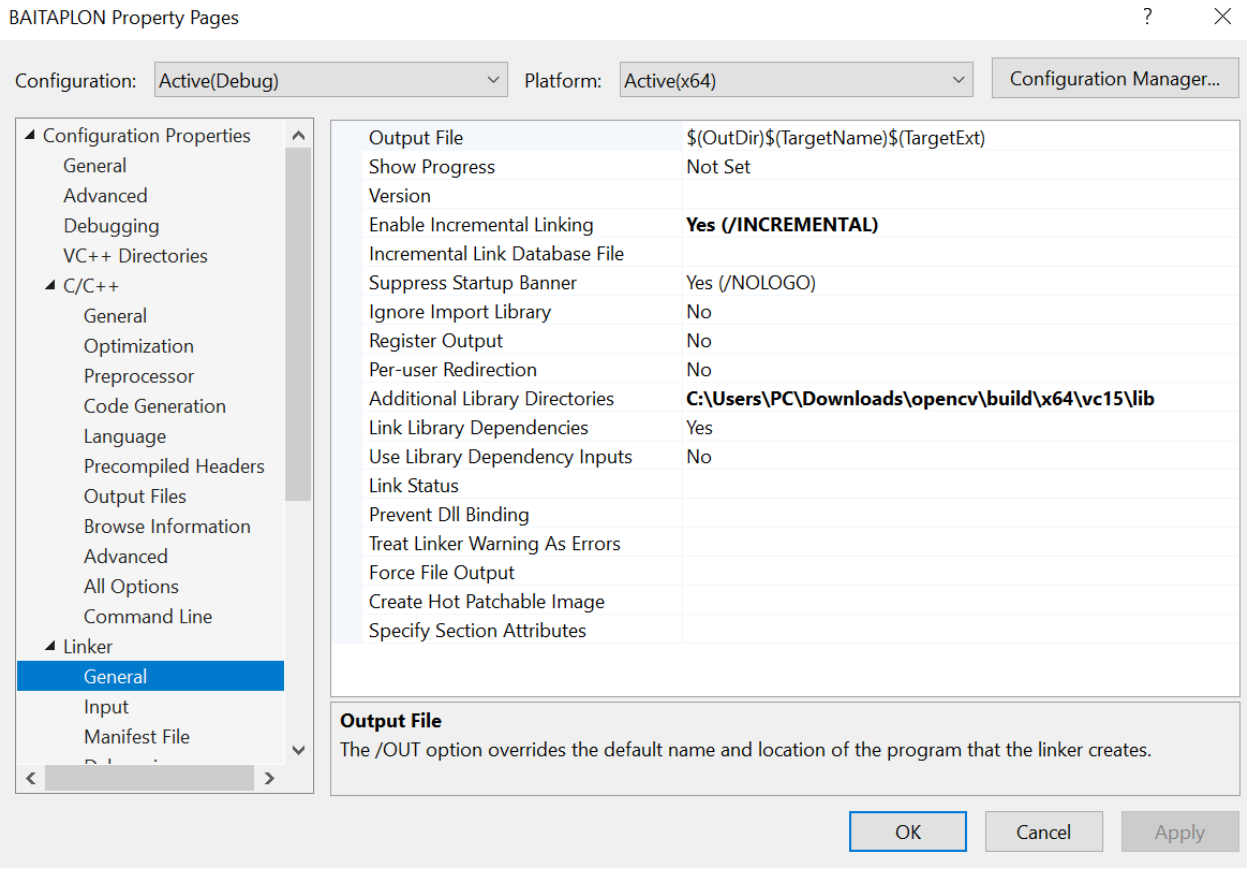


Chọn mục C/C++->General. Ở dòng Additional Include Directories, thêm vào đường dẫn đến thư mục build\include của thư mục OpenCV đã giải nén ở bước trên. Bấm vào Apply.



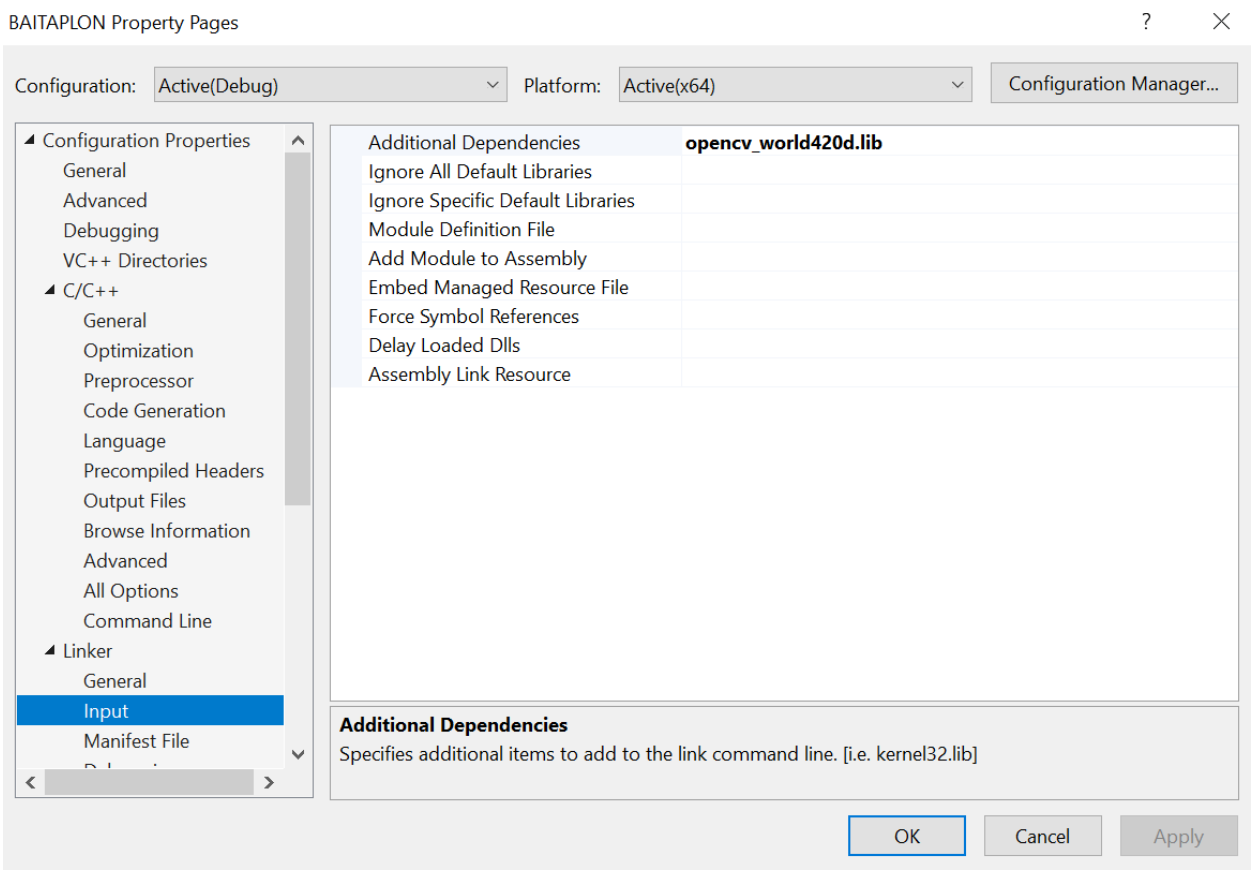
Chuyển sang mục Linker->General. Ở dòng Additional Library Directories, thêm vào đường dẫn đến thư mục build\x64\vc15\lib (thư mục chứa các file .lib) nếu dùng Visual Studio 2019, hoặc

build\x64\vc14\lib nếu dùng Visual Studio 2015. Bấm vào Apply.

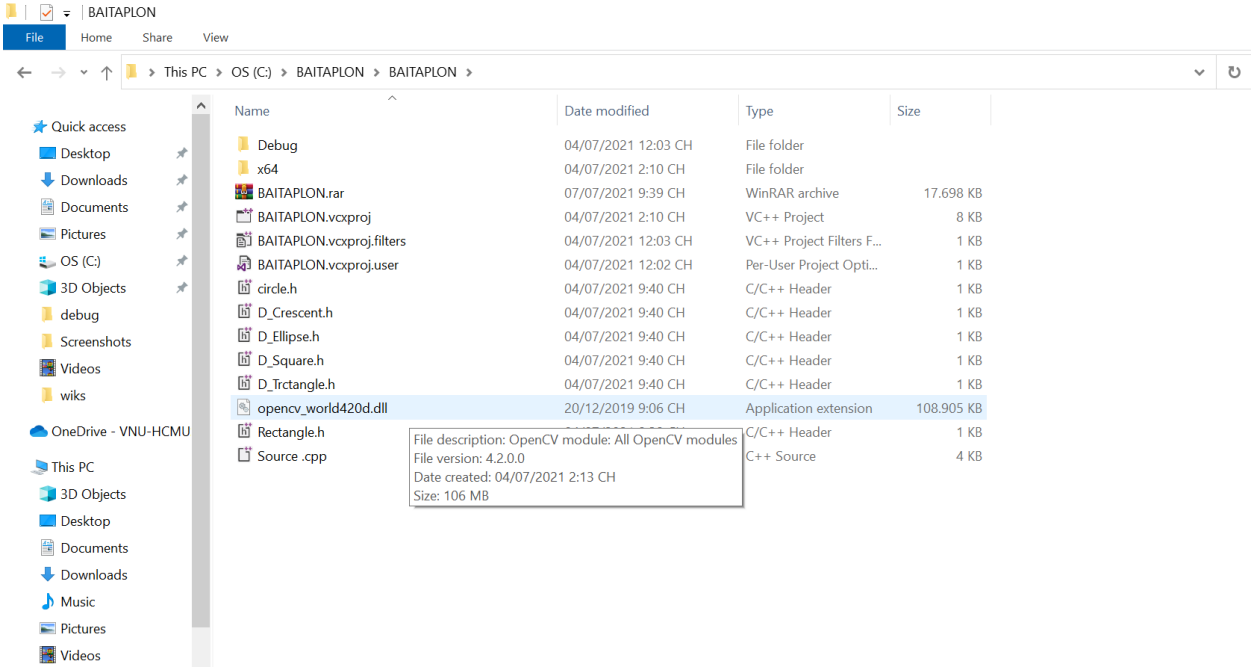


Chuyển sang mục Linker->Input. Ở dòng Additional Dependencies, thêm vào tên file opencv_world420d.lib (vì đang build ở chế độ Debug). Nếu build ở chế độ Release thì tên file là opencv_world420.lib. 2 file này có thể được tìm thấy trong thư mục build\x64...\bin. Bấm vào

Apply.



Sao chép file opencv_word420d.dll (hay opencv_word420.dll nếu đang build Release) trong thư mục build\x64.…\bin vào trong thư mục Project (thư mục chứa file cpp đã tạo).



Sau khi hoàn thành các bước ở trên thì ta có thể build được rồi.

CHƯƠNG 6: TÀI LIỆU THAM KHẢO

https://topdev.vn/blog/opencv-la-gi-hoc-computer-vision-khong-kho/#chon-ngon-ngu-nao-de-lap-trinh-opencv3

https://eitguide.net/tim-hieu-cau-truc-luu-tru-hinh-anh-mat-trong-opencv/

https://docs.opencv.org/4.5.2/d6/d6d/tutorial_mat_the_basic_image_container.html

https://thigiacmaytinh.com/c-cac-ham-ve-co-ban-trong-opencv/?fbclid=IwAR0TgD0blGQJDWsJUy2dr8tEY0KYYCvFkui5s4vpzjQS8eapingok4e4P6g

MỤC LỤC