Trước khi lập trình với ASY

Tips 02 2023 ...

Bài viết sẽ mô tả việc cài đặt **Asymptote** trên **MacOS**, sử dụng **Interactive mode** để việc lập trình **ASY** được hiệu quả.

1 Cài đặt Asymptote trên MacOS

Có thể tham khảo hướng dẫn cài đặt trên các hệ điều hành theo tài liệu ASY for Beginners. Bài viết này sẽ nói việc cài **asymptote** trên **MacOS** bằng cách sử dụng **Homebrew**.

Cài đặt Homebrew

Copy đoạn sau vào cửa sổ lệnh và chạy nó:

```
bash -c "$(curl␣-fsSL␣https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh
```

Khi đó máy tính sẽ download file **insatall.sh** ở đường dẫn trên và tiến hành cài đặt **Homebrew**.

Cài đặt Asymptote

Sau khi cài **Homebrew**, để cài **asymptote**, sử dụng lệnh sau:

```
brew install asymptote
```

Sau khi cài đặt **asymptote**, gõ **asy** (không có tham số) trên **Terminal** sẽ vào chế độ **Interactive mode**.

Để thuận tiện cho việc lập trình, có thể cài công cụ Visual Studio Code. Sau khi cài công cụ này, nên cài thêm gói **Asymptote** hỗ trợ highlight mã **asy** để việc lập trình được thuận tiện hơn.

2 Interactive mode

Interactive mode - chế độ tương tác của **Asymptote** với người dùng. Nếu sử dụng tốt, nó sẽ hỗ trợ nhiều cho lập trình.

Để vào chế độ **Interactive mode**, trên **Terminal** gõ lệnh **asy**, khi đó sẽ xuất hiện dấu nhắc ">". Sau đây là một số tính năng của nó.

Sử dụng như một máy tính

```
Welcome to Asymptote version 2.86 (to view the manual, type help)
> 1+2*3-12
-5
> 2**3
8
> 10%3
1
>
```

Gọi một số hàm toán học

```
> sin(pi/2)
1
> Cos(60)
0.5
```

```
> log(2)
0.693147180559945
> log10(10)
1
> exp(3)
20.0855369231877
>
```

Một số phép toán với ma trận

```
> real [][] a = {{1, 2}, {4,3}}
> real [][] b = {{3, 5}, {7,2}}
> a
1       2
4       3
> a+b
4       7
11      5
> a-b
-2      -3
-3      1
> a*b
17      9
33      26
> a/b
-: 1.2: no matching function 'operator /(real[][], real[][])'
> determinant(a)
-5
> transpose(a)
1       4
2       3
>
```

Chia ma trận chỉ sử dụng với ma trận một chiều.

Kiểm tra một số giá trị

```
> intMax
9223372036854775805
> intMin
-9223372036854775808
> realMax
1.79769313486232e+308
> realMin
2.2250738585072e-308
> realEpsilon
2.22044604925031e-16
>
```

Kiểm tra đơn vị (theo bp)

```
> bp
1
> pt
0.99626400996264
> inch
72
> cm
28.3464566929134
>
```

Kiểm tra một số pair định nghĩa sẵn

```
> NW
```

```
(-0.707106781186547,0.707106781186547)
> NNE
(0.38268343236509,0.923879532511287)
> up
(0,1)
> down
(0,-1)
> right
(1,0)
> left
(-1,0)
>
```

Kiểm tra một số path định nghĩa sẵn

```
> unitcircle
(1,0).. controls (1,0.552284749830793) and (0.552284749830793,1)
..(0,1).. controls (-0.552284749830793,1) and (-1,0.552284749830793)
..(-1,0).. controls (-1,-0.552284749830793) and (-0.552284749830793,-1)
..(0,-1).. controls (0.552284749830793,-1) and (1,-0.552284749830793)
..cycle
> unitsquare
(0,0)--(1,0)--(1,1)--(0,1)--cycle
>
```

Kiểm tra hàm

```
> circle
<path circle(pair c, real r)>
> ellipse
-: 1.1: use of variable 'ellipse' is ambiguous
<path ellipse(frame dest, frame src=<default>, real xmargin=<default>, real ymargin=<def
<path ellipse(frame f, Label L, real xmargin=<default>, real ymargin=<default>, pen p=<d
<path ellipse(pair c, real a, real b)>
>
```

Ở ví dụ trên sẽ hiện các hàm có tên là **circle**, **ellipse**.

```
> intersect
-: 1.1: use of variable 'intersect' is ambiguous
<real[] intersect(path p, path q, real fuzz=<default>)>
<real[] intersect(path3 p, path3 q, real fuzz=<default>)>
<real[] intersect(path3 p, triple[][] p, real fuzz=<default>)>
```

Import gói

Những ví dụ trên có thể kiểm tra các biến, hàm được định nghĩa sẵn mà không phải khai báo thêm gói nào. Nếu gõ **Circle** thì **Interactive mode** sẽ cảnh báo "*-: 1.1: no matching variable 'Circle'*". Khi đó ta dùng lệnh **import** gói **graph** tương tự như khi lập trình.

```
> Circle
-: 1.1: no matching variable 'Circle'
> import graph
> Circle
<path Circle(pair c, real r, int n=<default>)>
> import geometry
> circle
-: 1.1: use of variable 'circle' is ambiguous
<path circle(pair c, real r)>
<circle circle(segment s)>
<circle circle(inversion i)>
<circle circle(triangle t)>
<circle circle(explicit point C, real r)>
<circle circle(point A, point B)>
```

```
<circle circle(point A, point B, point C)>
>
```

Để thoát khỏi **Interactive mode**, gõ lệnh **quit**.

3 Lệnh write()

Khi lập trình, lệnh write() được sử dụng để write một biến lên cửa sổ console. Có thể sử dụng lệnh này khi debug code.

Ở ví dụ trên lệnh $write(A)$ sẽ viết mảng A lên cửa sổ **Terminal**.

Cũng tương tự như lệnh ở **Interactive mode**, lệnh **write()** có thể viết lên cửa sổ lệnh các biến. Tuy nhiên nó sẽ không viết được một hàm.

```
asy test002.asy
write(circle);
^
test002.asy: 2.6: no matching function 'write(path(pair c, real r))'
```

Ví dụ trên, lệnh **write(circle);** sẽ cảnh báo lỗi, nhưng cũng có gợi ý liên quan đến hàm có tên **circle**.

4 Solution

P1. Viết hàm vẽ đường tròn *A-Mixtilinear* (đường tròn tiếp xúc với 2 cạnh AB, AC và tiếp xúc trong với $\odot(ABC)$.

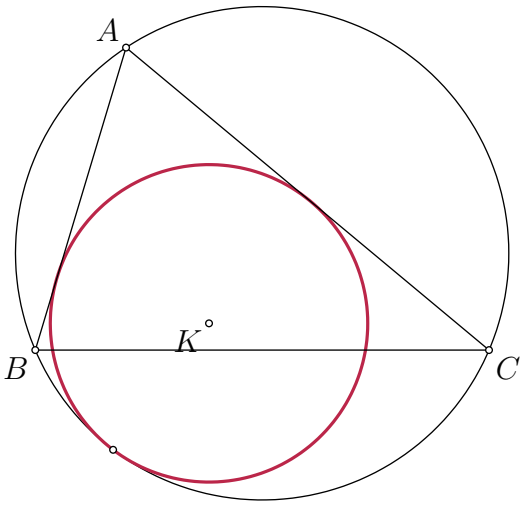
Sử dụng gói **geometry** ta đã có hàm tâm đường tròn nội tiếp **incenter** và bán kính đường tròn nội tiếp **inradius**, nên ta viết hàm tính bán kính đường tròn **Mixtilinear** từ đó dùng lệnh **scale** để tìm tâm đường tròn **Mixtilinear**.

```
import geometry;
```

```
unitsize(1cm);

real mixradius(point A, point B, point C){
        point I = incenter(A, B, C);
        real r = inradius(A, B, C);
        return r*abs(A-I)^2 / (abs(A-I)^2 - r^2);
}

point mixcenter(point A, point B, point C) {
        point I = incenter(A, B, C);
        return scale(mixradius(A, B, C) / inradius(A, B, C), A)*I;
}

circle mixcircle(point A, point B, point C){
        return circle(mixcenter(A, B, C), mixradius(A, B, C));
}
```

Khi vẽ đường tròn **Mixtilinear**, có thể dùng vẽ đường tròn ngoại tiếp tam giác $\triangle$ và dùng hàm **intersectionpoints** để tìm giao điểm và đếm số giao điểm để xem có 1 hay nhiều giao điểm.

Cũng có thể kiểm tra $abs(R - OK) - mixradius) < EPS$ để xác định 2 đường tròn tiếp xúc với nhau.



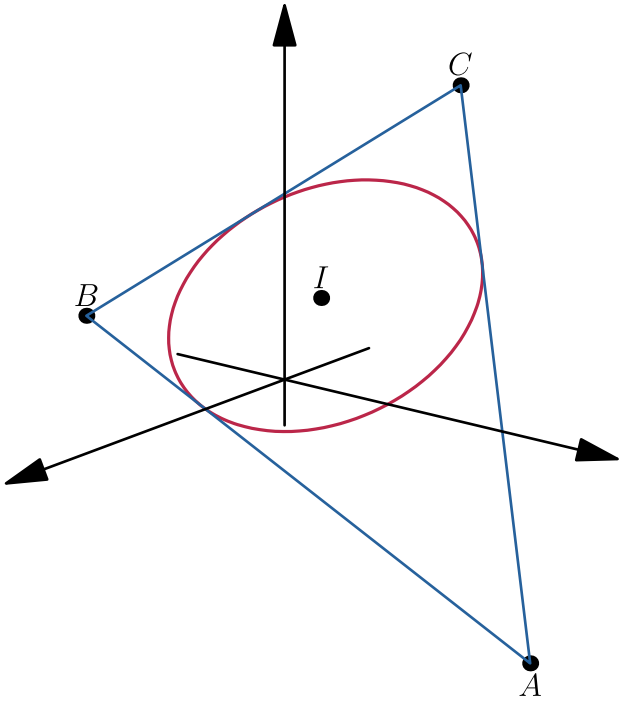
P2. Viết hàm tìm tâm nội tiếp của tam giác $\triangle ABC$ trên 3D.

Tâm nội tiếp tam giác $\triangle ABC$ trên 3D tương tự như trên 2D, tính theo công thức là $a * A + b * B + c * C$ trong đó a, b, c là độ dài các cạnh BC, CA, AB. Diện tích tam giác $\triangle ABC$ được tính theo công thức **Heron** và từ đó tính ra bán kính đường tròn nội tiếp.

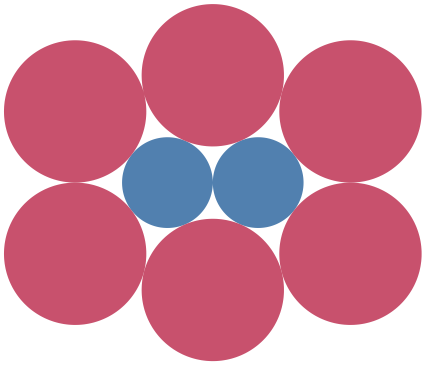
```
import three;
unitsize(1cm);
settings.render=0;

triple rpoint(real k, triple A, triple B) {
        return (1-k)*A + k*B;
}

triple bary(triple A, triple B, triple C, real x, real y, real z) {
        return (x*A + y*B + z*C) / (x+y+z);
}

real area(triple A, triple B, triple C) {
        real a = abs(B-C), b = abs(C-A), c = abs(A-B), p = (a+b+c)/2;
        return sqrt(p*(p-a)*(p-b)*(p-c));
}
```

```
triple incenter(triple A, triple B, triple C){
        real a = abs(B–C), b = abs(C–A), c = abs(A–B);
        return bary(A, B, C, a, b, c);
}

real inradius(triple A, triple B, triple C){
        real a = abs(B–C), b = abs(C–A), c = abs(A–B), p = (a+b+c)/2;
        return area(A, B, C)/p;
}

path3 incircle(triple A, triple B, triple C){
        return circle(incenter(A,B,C),inradius(A, B, C),normal(A--B--C));
}
```

Lưu ý trong đoạn code trên, **incircle** có kiểu **path3** nên vẽ không được chính xác.



P3. Vẽ hình dưới.



Hình này sử dụng để minh họa cho việc giải phương trình để vẽ hình. Chúng ta sẽ tìm tỷ lệ giữa bán kính đường tròn xanh và đường tròn đỏ. Tỷ lệ này là nghiệm nhỏ hơn 1 của phương trình bậc 4: $x^4 - 10x^2 - 8x + 9 = 0$. Trong gói **gemetry** có hàm **real[] realquarticroots(real a, real b, real c, real d, real e)** để giải phương trình này.

```
import geometry;
unitsize(1cm);
```

```
pen mycolor=RGB(187, 38, 73);
pen mycolor2=RGB(38, 97, 156);

point O=(0,0), A=(1,0), B=(-1,0);
circle ca=circle(A, 1), cb=circle(B, 1);

real f(real x) {
        return x^4-10x^2-8x+9;
}

real []xr =realquarticroots(1, 0, -10, -8, 9);
real R=1/xr[0];
real yc = sqrt((1+R)^2-1);
point C=(0, yc); circle cc=circle(C, R);
real xd=-sqrt(4R^2-(yc-R)^2);
point D=(xd, R); circle cd=circle(D, R);

fill(ca^^cb, mycolor2+opacity(0.8));
fill(cc^^cd, mycolor+opacity(0.8));
fill(circle(reflect(line(A, B))*C,R), mycolor+opacity(0.8));
fill(circle(reflect(line(A, B))*D,R), mycolor+opacity(0.8));
fill(circle(reflect(line(O, C))*D,R), mycolor+opacity(0.8));
fill(circle(2O-D,R), mycolor+opacity(0.8));
```

Trong trường hợp phương trình bậc cao hơn 4, có thể dùng phương pháp tìm nghiệm gần đúng. Hàm built-in có hàm **newton** tìm nghiệm gần đúng như sau:

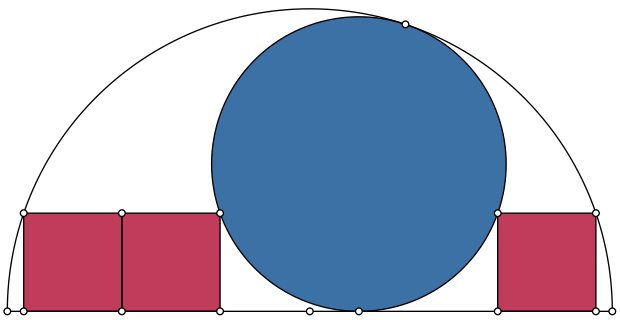
```
real f(real x) {
        return x^4-10x^2-8x+9;
}

real fx(real x) {
        return 4x^3-20x-8;
}

real R=1/newton(f, fx, 0, 1);
```

Lưu ý **fx** là đạo hàm của hàm số **f**.

Với những phương trình tìm đạo hàm khó, ta có thể dùng phương pháp **Secant** hoặc một phương pháp nào đó để tìm nghiệm. Nếu không thể tìm ra một phương trình dạng dễ nhìn, ta cũng có thể kết hợp công cụ hình học để tìm nghiệm theo **Secant**.

5 Problem

P1. Vẽ hình dưới.



6 Tài liệu tham khảo

[1]. Tài liệu Asymptote: the Vector Graphics Language

[2]. Các ví dụ của Asymptote

[3]. Tài liệu Asymptote Démarrage rapide

[4]. Gói geometry.asy và olympiad.asy

[5]. Tài liệu hướng dẫn gói **geometry.asy**: Euclidean geometry with asymptote

[6]. Tài liệu hướng dẫn Asymptote 3D: Asymptote 3D, Bruno M. Colombel

[7]. Tài liệu hướng dẫn ASY (tiếng Việt): ASY for Beginners, Trần Quân