

Asymptote tutorial for beginners

Trần Quân ++ ...

Bài viết này sẽ hướng dẫn việc cài đặt, dịch và lập trình cơ bản **Asymptote** cho người mới bắt đầu.

Mục lục

1	Cài đặt Asymptote và các chương trình	3
1.1	Asymptote trên Windows	3
1.1.1	Cài đặt	3
1.1.2	Dịch ASY ở cửa sổ lệnh	6
1.1.3	ASY với Notepad++	6
1.1.4	ASY với TexStudio	8
1.2	Asymptote trên Ubuntu	11
1.3	Asymptote trên MacOS	11
2	Vẽ hình học phẳng với ASY cơ bản	13
2.1	Kiểu dữ liệu	14
2.2	Các phép biến hình	15
2.3	Điểm	15
2.4	Đoạn thẳng, đường thẳng	16
2.5	Đường tròn, cung tròn, E-líp	17
2.6	Giao điểm	17
2.7	Một số ví dụ	19
3	Gói geometry.asy và vẽ hình nâng cao	28
3.1	Kiểu point	28
3.2	Kiểu line, segment	29
3.2.1	Cấu trúc của line	29
3.2.2	Định nghĩa line	29
3.2.3	Các toán tử áp dụng cho line	30
3.2.4	Một số hàm liên quan đến line	31
3.2.5	Cấu trúc của segment	31
3.2.6	Cấu hàm liên quan đến segment	31

3.3	Kiểu conic	31
3.3.1	Cấu trúc của kiểu conic	31
3.3.2	Định nghĩa (khai báo) conic	32
3.3.3	Một số hàm liên quan đến conic	32
3.3.4	Các toán tử áp dụng cho conic	32
3.4	Kiểu circle	33
3.4.1	Cấu trúc của kiểu circle	33
3.4.2	Định nghĩa circle	33
3.4.3	Một số hàm liên quan đến circle	33
3.4.4	Các toán tử áp dụng cho circle	34
3.5	Kiểu ellipse	34
3.5.1	Cấu trúc của kiểu ellipse	34
3.5.2	Định nghĩa ellipse	35
3.5.3	Một số hàm liên quan đến ellipse	35
3.6	Kiểu parabola	35
3.6.1	Cấu trúc của kiểu parabola	35
3.6.2	Định nghĩa parabola	36
3.6.3	Một số hàm liên quan đến parabola	36
3.7	Kiểu hyperbola	36
3.7.1	Cấu trúc của kiểu hyperbola	36
3.7.2	Định nghĩa hyperbola	37
3.7.3	Một số hàm liên quan đến hyperbola	37
3.8	Kiểu arc	37
3.8.1	Cấu trúc của arc	37
3.8.2	Định nghĩa arc	37
3.9	Kiểu triangle	38
3.9.1	Cấu trúc của kiểu triangle	38
3.9.2	Định nghĩa (khai báo) triangle cơ bản	38
3.9.3	Hàm liên quan đến cấu trúc đỉnh (vertex)	38
3.9.4	Hàm liên quan đến cấu trúc cạnh (side)	39
3.9.5	Các toán tử áp dụng cho triangle	39
3.9.6	Một số hàm khác	39
3.10	Kiểu trilinear	40
3.10.1	Cấu trúc của kiểu trilinear	40
3.10.2	Định nghĩa (khai báo) trilinear cơ bản	40

3.11	Các phép biến hình của gói geometry	41
3.11.1	Phép biến hình scale	41
3.11.2	Phép biến hình projection	41
3.11.3	Phép biến hình reflect	42
3.11.4	Phép biến hình rotate	42
3.12	Phép nghịch đảo inversion	42
3.12.1	Cấu trúc của inversion	42
3.12.2	Định nghĩa inversion	42
3.12.3	Sử dụng inversion	43
3.13	Giao điểm trong gói geometry.asy	43
3.14	Một số hàm dựa trên gói geometry	43
3.15	Các ví dụ sử dụng gói geometry	47
4	Gói three.asy và lập trình ASY 3D	56
5	Phụ lục	56
5.1	Hướng dẫn TeXStudio và ASY	56
6	Tham khảo	56

Asymptote (vector graphics language) là ngôn ngữ lập trình đồ họa dạng *vector* được thiết kế để tạo hình vẽ và biểu đồ toán học. *Asymptote* có thể xuất hình ảnh ở định dạng *eps* hoặc *pdf* và tương thích với *Latex*. *Asymptote* được lấy cảm hứng từ *MetaPost*, tuy nhiên nó là một ngôn ngữ lập trình hoàn chỉnh và cú pháp rõ ràng hơn (giống ngôn ngữ *C++*) so với *Metapost*.

Asymptote có thể chạy trên các hệ điều hành thông dụng như *Windows*, *Linux* và *MacOS*.

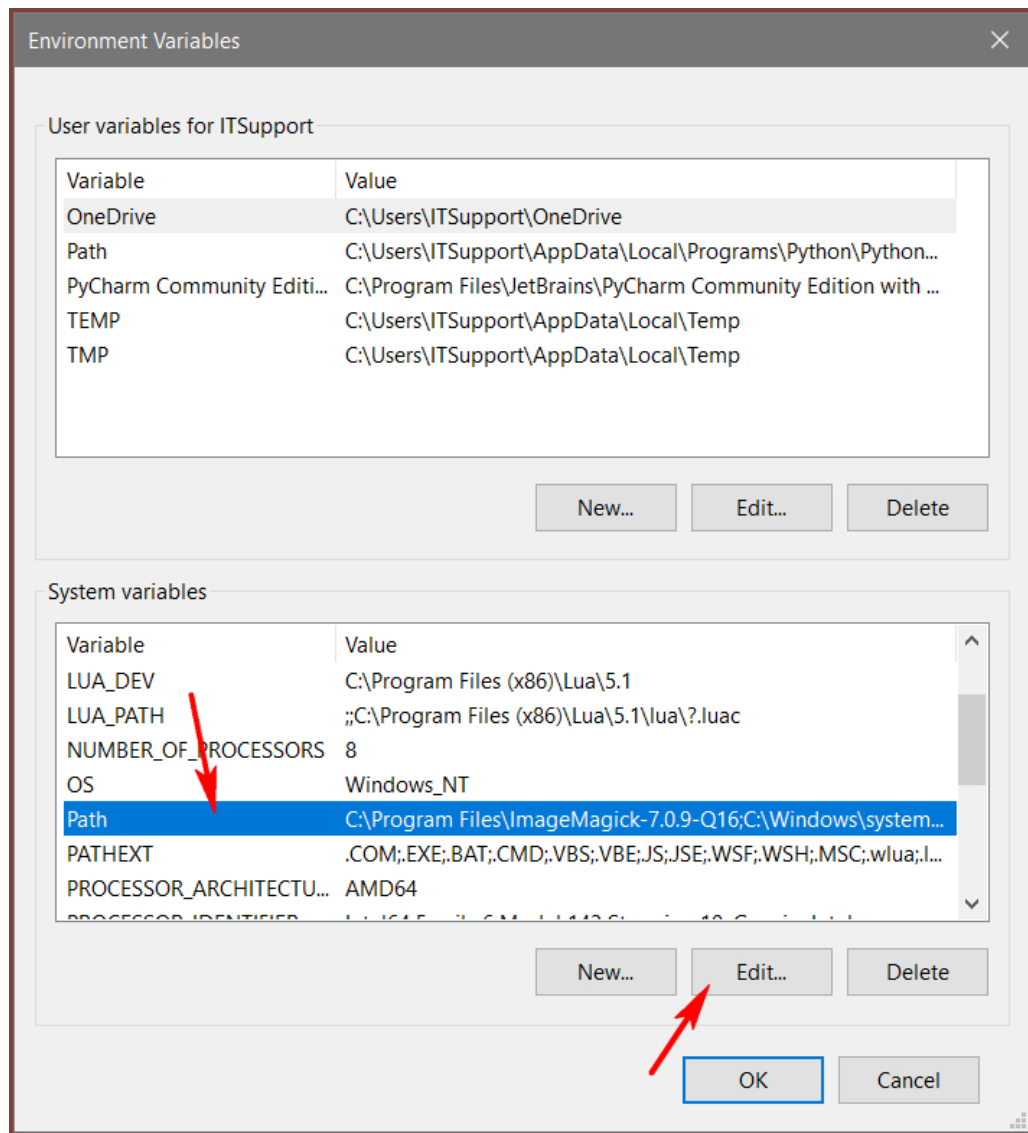
1 Cài đặt Asymptote và các chương trình

1.1 Asymptote trên Windows

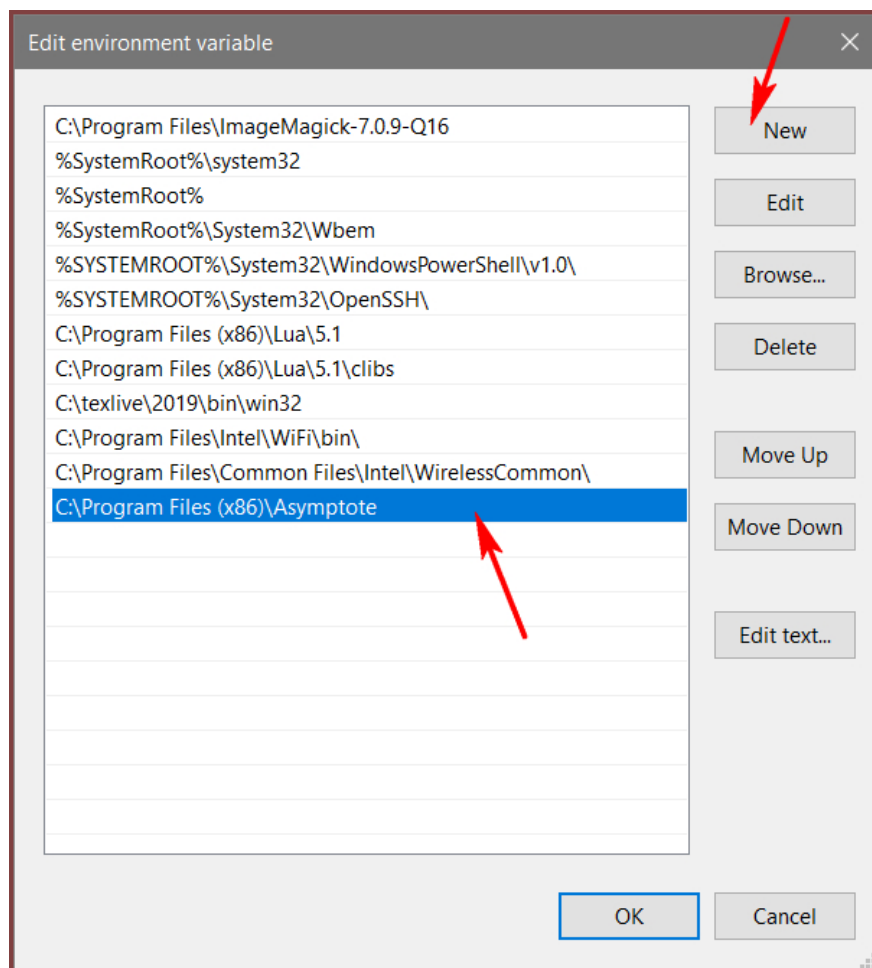
1.1.1 Cài đặt

Download phiên bản mới nhất của **Asymptote** theo link <http://sourceforge.net/projects/asymptote> và tiến hành cài đặt. Sau khi cài, thư mục cài trên Windows là `C:\ProgramFiles(x86)\Asymptote`.

Sau khi cài đặt, thêm đường dẫn trên vào biến **PATH** của Windows như sau: *Control Panel* → *System* → *Advanced system settings*. Sau đó chọn tab **Advanced** và chọn **Environment Variables**.



Trên hộp thoại **Environnement Variables**, chọn **Path**, sau đó chọn **Edit**. Xuất hiện hộp thoại **Edit environment variables**.



Trên hộp thoại **Edit environment variables**, chọn **New** để thêm đường dẫn `C:\ProgramFiles(x86)\Asymptote`.

Sau đó chạy ra cửa sổ lệnh của *Windows*, gõ lệnh **PATH** để xác nhận đường dẫn này. Khi đó gõ lệnh *asy* (chạy file *asy.exe*), trên cửa sổ lệnh sẽ như sau:

```

Administrator: Command Prompt - asy
11/18/2019 12:27 PM          23,846 three_arrows.asy
11/18/2019 12:27 PM           3,673 three_light.asy
11/18/2019 12:27 PM           2,764 three_margins.asy
11/18/2019 12:27 PM        73,252 three_surface.asy
11/18/2019 12:27 PM        12,522 three_tube.asy
11/18/2019 12:27 PM           1,375 tree.asy
11/18/2019 12:27 PM           5,797 trembling.asy
11/18/2019 12:27 PM           4,297 tube.asy
11/18/2019 12:27 PM             51 unicode.asy
11/19/2019 07:56 AM        51,652 uninst.exe
11/18/2019 12:41 PM           23 version.asy
11/12/2019 04:17 PM      <DIR>      webgl
11/18/2019 12:27 PM        4,812 x11colors.asy
11/18/2019 12:41 PM           90 xasy
    108 File(s)      14,385,844 bytes
     6 Dir(s)  244,788,703,232 bytes free

C:\Program Files (x86)\Asymptote>asy
Welcome to Asymptote version 2.61 (to view the manual, type help)
>

```

Đây chính là **interactive mode** của ASY. Lưu ý, để thoát khỏi **interactive mode**, có thể gõ lệnh **quit** hoặc dùng tổ hợp phím **Ctrl + C**.

1.1.2 Dịch ASY ở cửa sổ lệnh

Dùng một chương trình text editor bất kỳ để tạo file **hd01.asy** có nội dung như sau:

```
unitsize(1cm);

pair A=(1.5,2.5); dot(Label("$A$",align=NE),A);
pair B=(0,0); dot(Label("$B$",align=SE),B);
pair C=(7,0); dot(Label("$C$",align=SE),C);

draw(A--B--C--A);
```

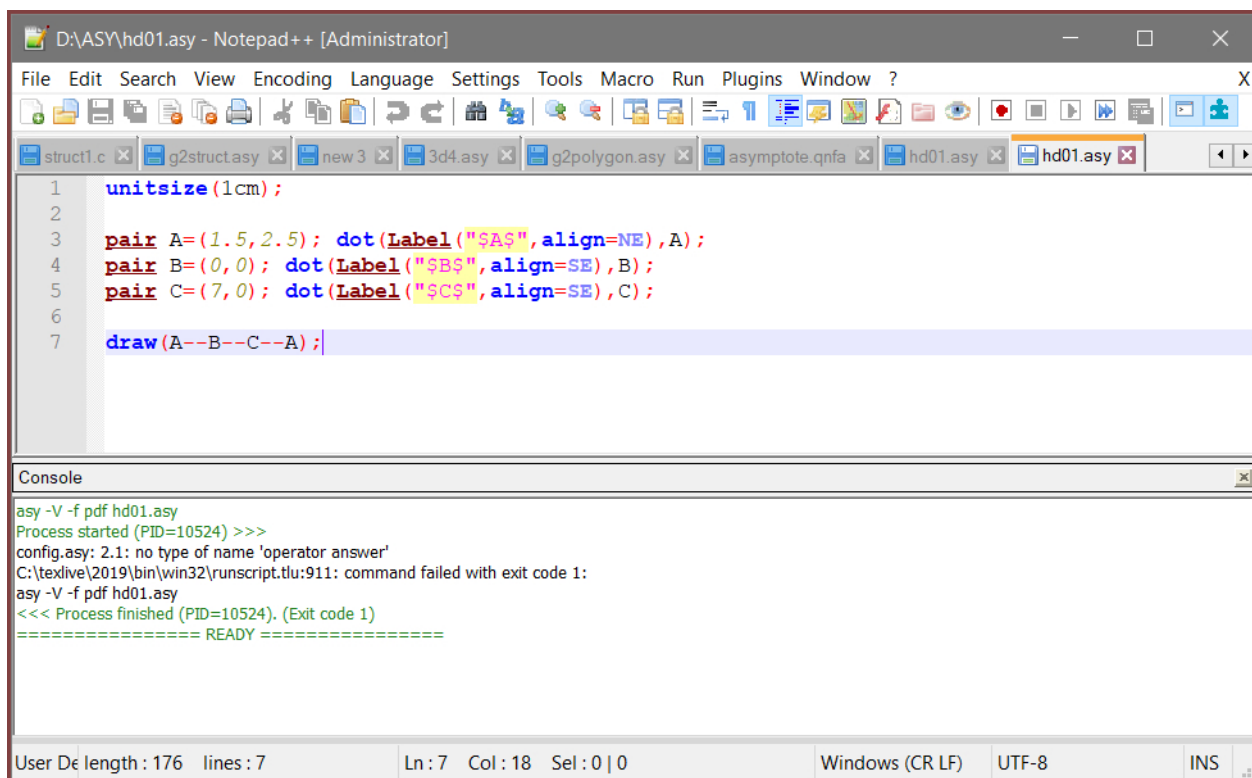
Ở cửa sổ lệnh của *Windows*, gõ lệnh: **asy -V -f pdf hd01.asy**. Khi đó chương trình **asy.exe** sẽ chạy và tạo ra file **hd01.pdf** cùng thư mục với file **hd01.asy** và gọi phần mềm đọc file pdf để mở nó.

Lưu ý có thể cần gõ đường dẫn đến file *.asy cần dịch, ví dụ "D:\tmp\hd01.asy".

1.1.3 ASY với Notepad++

Để thuận tiện cho việc lập trình ASY, có thể dùng *text editor* rất quen thuộc với những người lập trình là **Notepad++**.

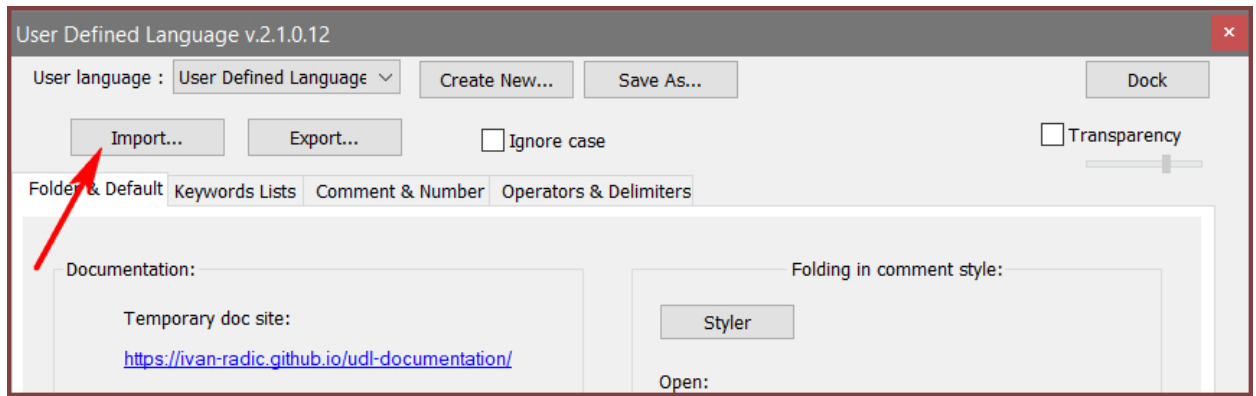
Download **Notepad++** theo link <https://notepad-plus-plus.org/downloads/> và cài đặt nó. Giao diện chương trình như sau:



Tiếp theo, chúng ta cần *highlight* cú pháp của ASY trên Notepad++.

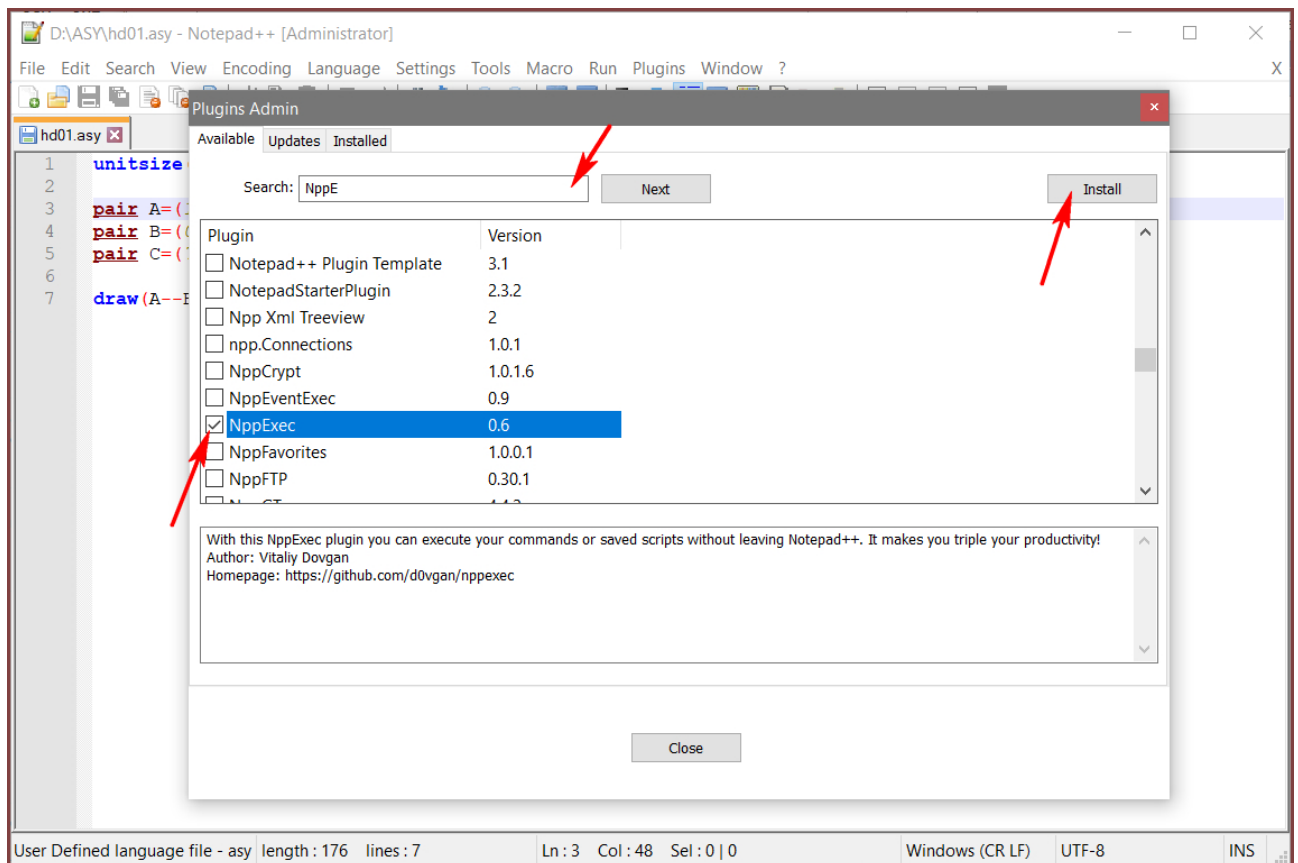
Download file **UserDefineLang.xml** theo link <https://github.com/asymarris/asy-coloration-dans-npp>.

Trên **Notepad++**, chọn menu *Language* → *Define you language*, xuất hiện hộp thoại **User Defined Language**.



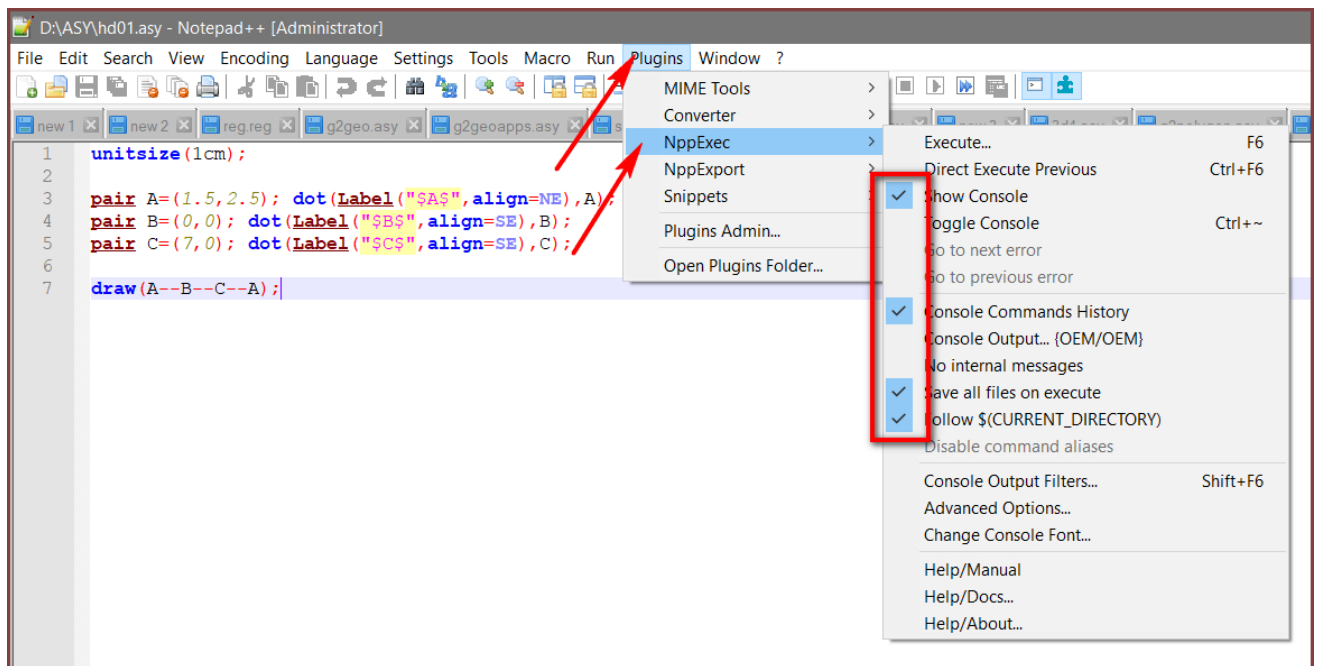
Chọn **Import**, sau đó chọn file **UserDefineLang.xml**. Sau khi kết thúc, kiểm tra lại việc *highlight* các từ khóa của **ASY**.

Notepad++ cho phép chạy các chương trình bên ngoài thông qua plugin **NppExec**. Để cài đặt plugin này, chọn menu **Plugins** → **Plugins Admin**.

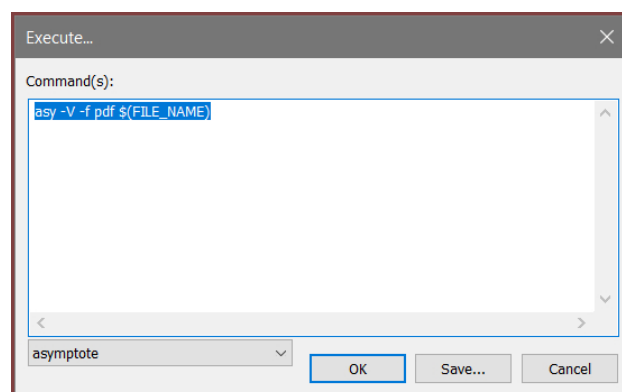


Gõ **NppExec** vào ô **Search**, sau đó nhấp nút **Next** và check **NppExec** vào ô chọn. Sau đó nhấp nút **Install** để cài đặt plugin này. **Notepad++** sẽ cài đặt **NppExec** và chạy lại.

Khi đó trên menu, chọn **Plugins**, sẽ xuất hiện thêm lựa chọn **NppExec**. Lưu ý chọn vào các lựa chọn: *Show Console*, *Console Commands History*, *Save all files on execute*, *Follow*.



Chọn **Excute ... (F6)**.



Nhập lệnh sau vào ô **Command(s)**, sau đó nhấp nút **Save** để lưu lại. Bấm **OK** để thực hiện lệnh này.

```
asy -V -f pdf $(FILE_NAME)
```

Trên cửa sổ **Console** của **Notepad++** sẽ xuất hiện quá trình thực thi lệnh và sẽ gọi chương trình đọc file pdf (ví dụ Acrobat Reader) trên *Windows* để mở file sau khi tạo.

1.1.4 ASY với TexStudio

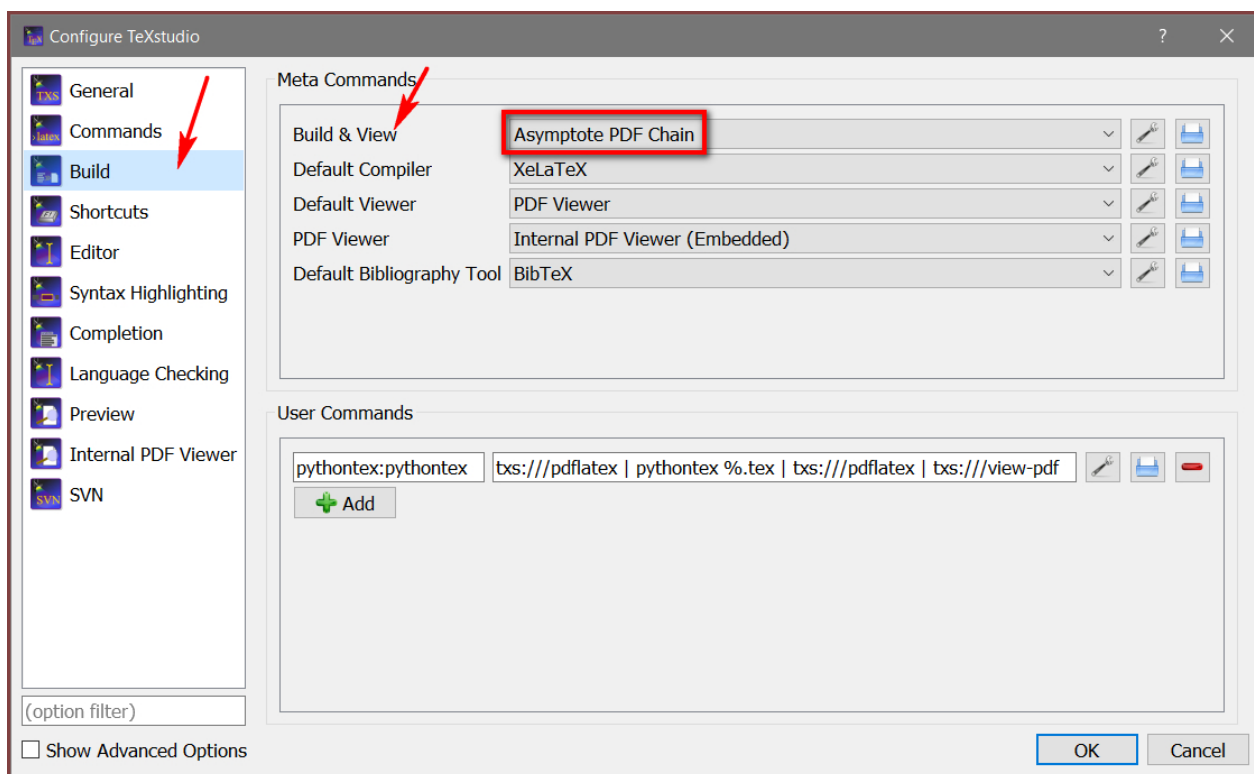
TexStudio là trình soạn thảo latex có thể chạy trên các hệ điều hành thông dụng như Windows, Linux, MacOS. TexStudio được cài đặt cùng với **TexLive** và **Ghostscript**. Download và cài đặt theo link:

- TexStudio: <https://www.texstudio.org/>.
- TexLive: Nên download file iso để việc cài đặt thuận tiện hơn: <https://www.tug.org/texlive/acquire-iso.html>.
- Ghostscript: <https://www.ghostscript.com/download/gsdnld.html>.

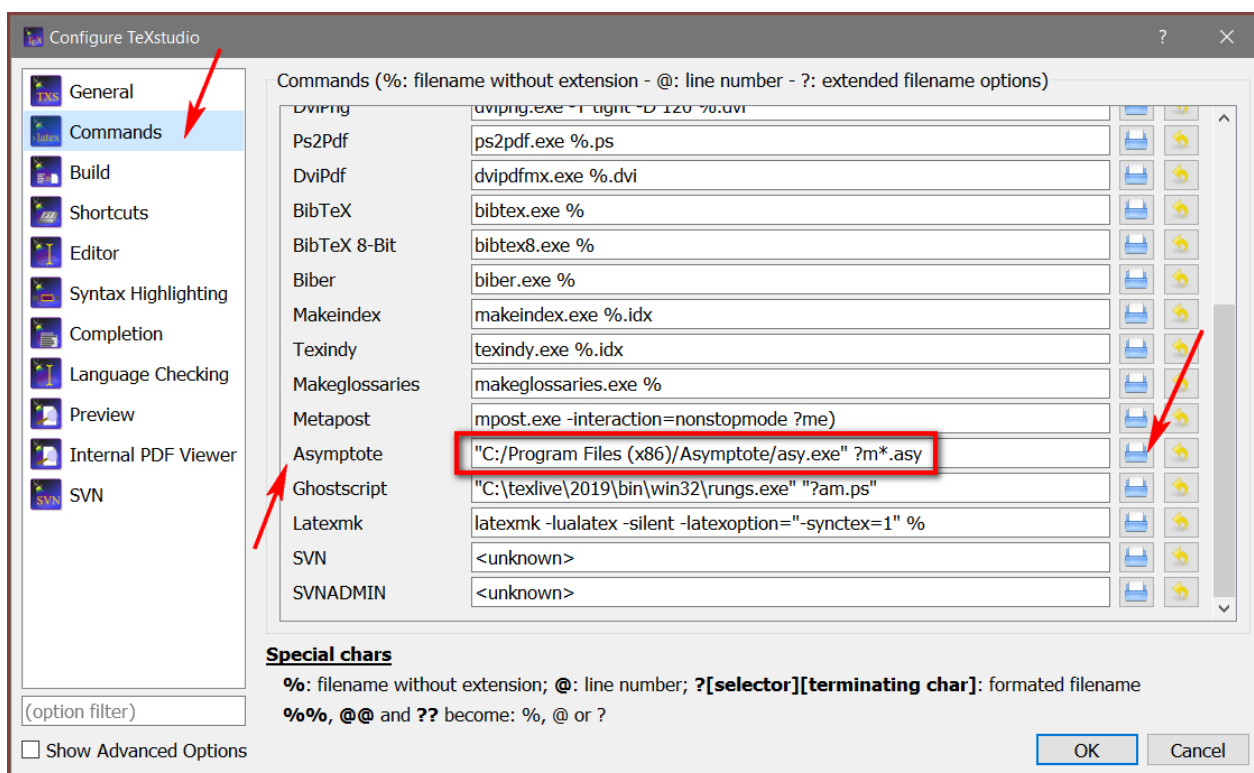
Ngoài ra nên cài thêm **ImageMagick** khi cần chuyển đổi file pdf thành file ảnh.

Lưu ý, **TexLive** đã có sẵn gói **ASY**. Tuy nhiên phiên bản **ASY** trên **TexLive** cập nhật chậm. Nên download **ASY** và cài đặt theo hướng dẫn ở mục 1, việc cấu hình để sử dụng **ASY** sẽ được hướng dẫn tiếp theo.

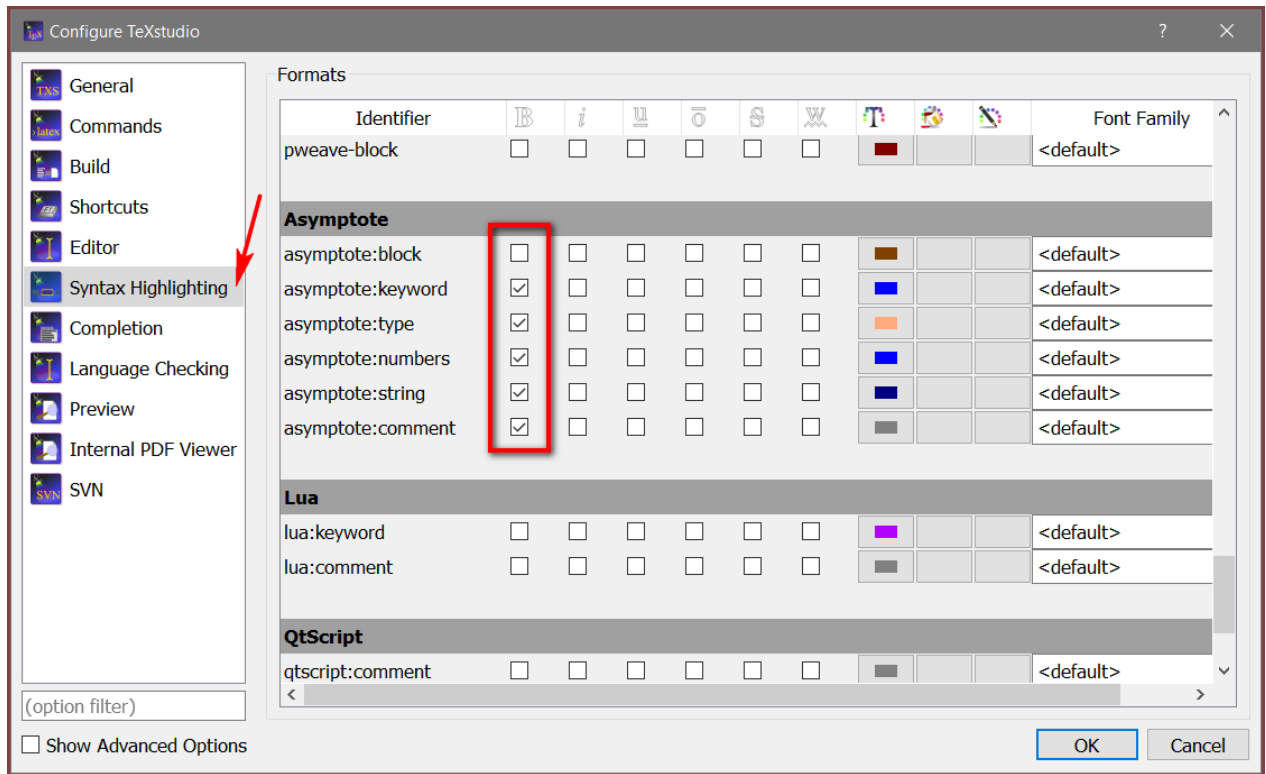
Để dịch **ASY** trên **TexStudio**, trên giao diện **TexStudio** chọn menu *Options* → *Configure TexStudio*. Sau đó chọn **Build** và ở lựa chọn **Build View**, chọn **Asymptote PDF Chain**.



Như đã nói trên, **TexLive** có sẵn **asy.exe** ở thư mục `C:\texlive\2019\bin\win32` và thư mục này được khai báo trên Windows PATH. Để chọn file **asy.exe** khác, chọn **Commands**, ở lựa chọn **Asymptote**, chọn đường dẫn đến file **asy.exe** theo đường dẫn `C:\Program Files (x86)\Asymptote`.



Sau bước này chúng ta đã có thể dịch các chương trình **ASY** trên **TexStudio**. Tuy nhiên để việc viết mã **ASY** được thuận tiện, cần cấu hình **Syntax Highlight** cho các từ khóa của **ASY**. Chọn **Syntax Highlighting**, ở mục **Asymptote**, check vào các ô cần thiết như *keyword*, *type*, *number*, *string*, *comment*.



Tuy nhiên ngầm định **Syntax Highlight** của **TeXStudio** cho **ASY** khá ít từ khóa được highlight, cần bổ sung các thêm các từ khóa.

Download file **asymptote.qnfa** theo link <https://github.com/textstudio-org/textstudio/blob/master/utilities/qxs/asymptote.qnfa>. Sau đó tạo thư mục **config\languages** trong thư mục **C:\ProgramFiles(x86)\textstudio** và copy file **asymptote.qnfa** vào thư mục đó. Đường dẫn sau khi copy **C:\ProgramFiles(x86)\textstudio\config\languages\asymptote.qnfa**.

Nếu sử dụng **TeXStudio** bản **portable**, sử dụng đường dẫn sau **\textstudio-2.12.16-portable\config\languages\asymptote.qnfa**.

File **asymptote.qnfa** là file text có cấu trúc. Để thêm các từ khóa, mở nó bằng bất cứ chương trình text editor nào và thêm các từ khóa cần thiết.

Cấu hình cho **TeXStudio** sử dụng được **ASY** nói trên thực chất **TeXStudio** tạo file **.asy** và gọi **asy.exe** để dịch nó thành file **pdf**, sau đó nhúng vào file **pdf** chính của tex.

Ví dụ, file **QHT122019.tex** có chứa mã lệnh **ASY**. Khi **TeXStudio** dịch file này, nó sẽ tạo mỗi mục **ASY** thành các file **QHT122019-1.asy**, **QHT122019-2.asy**, ... và gọi **asy.exe** để dịch các file này thành file pdf. Sau đó nhúng vào file **QHT122019.pdf** chính.

Cách thực hiện này có thể gặp lỗi trong các trường hợp sau:

- File pdf đang mở ở chương trình khác nên **asy.exe** không tạo được file pdf đè lên.
- Trong file tex có nhiều mục **ASY**, khi thêm, xóa các file này, có thể **TeXStudio** không đè lên file cũ. Trường hợp này, có thể xóa các file không cần thiết và chỉ cần giữ lại file tex, sau đó dịch lại.

Lưu ý, một mục **ASY** trên **TeXStudio** nằm trong cặp **begin**, **end** như sau:

```
\begin{asy}

unitsize(1cm);
pair A=(1.5,2.5); dot(Label("$A$"),align=NE,A);
pair B=(0,0); dot(Label("$B$"),align=SE,B);
pair C=(7,0); dot(Label("$C$"),align=SE,C);
draw(A--B--C--A);

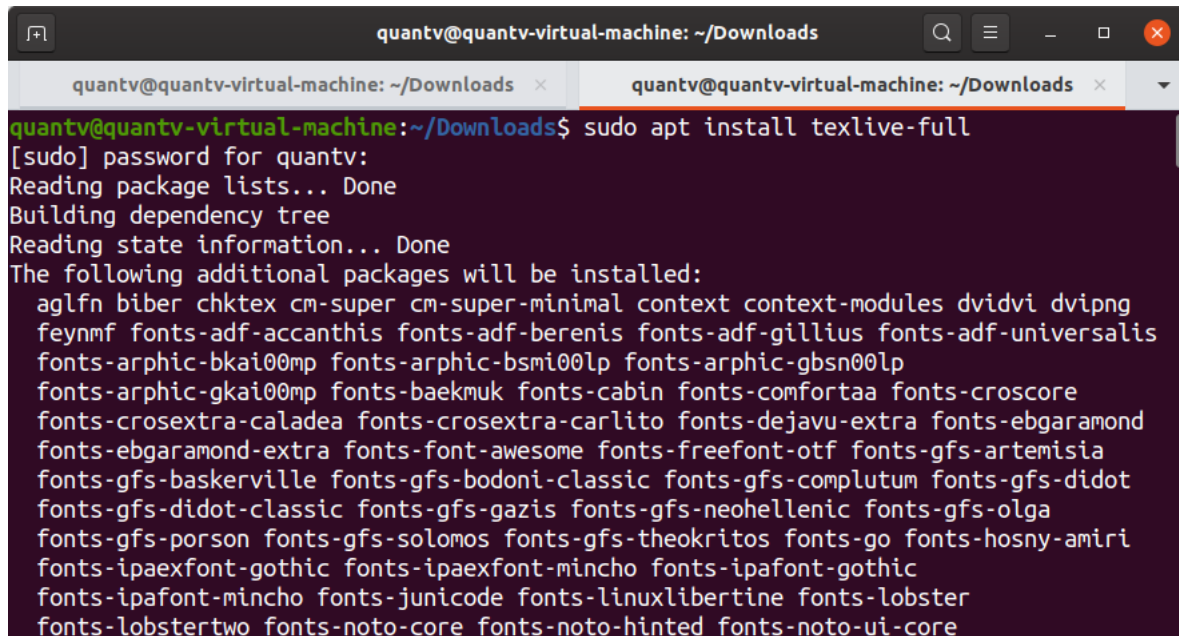
\end{asy}
```

1.2 Asymptote trên Ubuntu

Trên **Ubuntu**, nên theo các bước sau: cài TexLive, cập nhật ASY và cài TexStudio và cấu hình.

Để cài đặt **TexLive**, chạy **Terminal** của **Ubuntu** và gõ lệnh sau:

```
$ sudo apt install texlive-full
```



```
quantv@quantv-virtual-machine: ~/Downloads
quantv@quantv-virtual-machine: ~/Downloads
quantv@quantv-virtual-machine:~/Downloads$ sudo apt install texlive-full
[sudo] password for quantv:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 aglfn biber chktex cm-super cm-super-minimal context context-modules dvidvi dvipng
 feynmf fonts-adf-accanthis fonts-adf-berenis fonts-adf-gillius fonts-adf-universalis
 fonts-arphic-bkai00mp fonts-arphic-bsmi00lp fonts-arphic-gbsn00lp
 fonts-arphic-gkai00mp fonts-baekmuk fonts-cabin fonts-comfortaa fonts-croscore
 fonts-crosextra-caladea fonts-crosextra-carlito fonts-dejavu-extra fonts-ebgaramond
 fonts-ebgaramond-extra fonts-font-awesome fonts-freefont-otf fonts-gfs-artemisla
 fonts-gfs-baskerville fonts-gfs-bodoni-classic fonts-gfs-complutum fonts-gfs-didot
 fonts-gfs-didot-classic fonts-gfs-gazis fonts-gfs-neohellenic fonts-gfs-olga
 fonts-gfs-porson fonts-gfs-solomos fonts-gfs-theokritos fonts-go fonts-hosny-amiri
 fonts-ipaexfont-gothic fonts-ipaexfont-mincho fonts-ipafont-gothic
 fonts-ipafont-mincho fonts-junicode fonts-linuxlibertine fonts-lobster
 fonts-lobstertwo fonts-noto-core fonts-noto-hinted fonts-noto-ui-core
```

Tương tự như Windows, TexLive đã có ASY. Trong trường hợp bản ASY trên TexLive chưa cập nhật, có thể dùng lệnh **tlmgr update --self** để cập nhật cho TexLive hoặc download bản **deb** mới nhất của ASY theo link sau: <https://launchpad.net/ubuntu/+source/asymptote>.

ASY mới nhất hiện tại là **asymptote_2.61-1_amd64.deb**. Khi download về thường trong thư mục **Downloads**. Chạy **Terminal** của Ubuntu và gõ lệnh sau:

```
$ cd Downloads
$ sudo dpkg -i asymptote_2.61-1_amd64.deb
```

Lệnh này sẽ cài và cập nhật ASY bản mới nhất.

Để cài TexStudio, ở **Terminal** gõ lệnh:

```
$ sudo apt install texstudio
```

Việc cấu hình **Asymptote** trên **TexStudio** tương tự như trên Windows.

1.3 Asymptote trên MacOS

Cài **ASY** trên **MacOS** có nhiều cách: Cài (dịch) từ *source*; Cài qua gói **MacTex** (phiên bản trên **MacOS** của **TexLive**) và Cài thông qua **MacPorts**.

Để dịch từ *source*, download file *source* theo link <https://asymptote.sourceforge.io/>. Ta được file **asymptote-x.x.x.src.tar** (x.x.x là phiên bản, ví dụ **asymptote-2.6.1.src.tar**). File này được lưu vào thư mục **Downloads**.

Chạy **Terminal** của **MacOS** và chạy các lệnh sau:

```
$ cd Downloads
$ tar -xzf asymptote-2.6.1.src.tar
$ cd asymptote-2.6.1
```

```
$ sudo make all
$ sudo make install
```

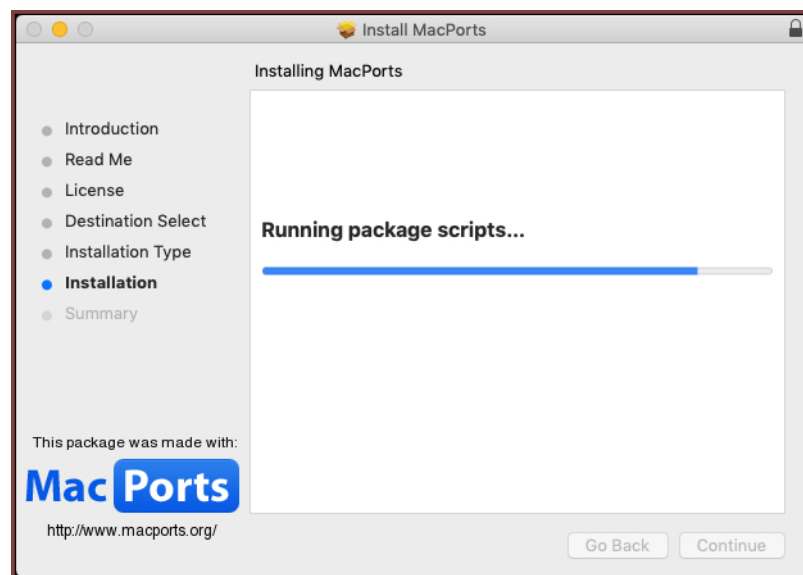
Lưu ý khi chạy `./configure` có thể được yêu cầu cài đặt các trình biên dịch **Xcode** hoặc **gcc**, chọn cài **gcc**.

Tuy nhiên dịch từ *source* dễ gặp lỗi vì thiếu **package** và khó xử lý.

Để cài đặt **MacTeX**, download file **MacTeX.pkg** theo link <http://www.tug.org/mactex/downloading.html>. Kết thúc download, nhấp đúp vào file này để cài đặt.

Sau khi cài đặt **MacTeX**, kiểm tra phiên bản ASY bằng cách chạy lệnh **asy** ở **Terminal**. Trong trường hợp ASY chưa được cập nhật mới, vào Launchpad, chạy chương trình **TeXLive Utility** để cập nhật các gói mới.

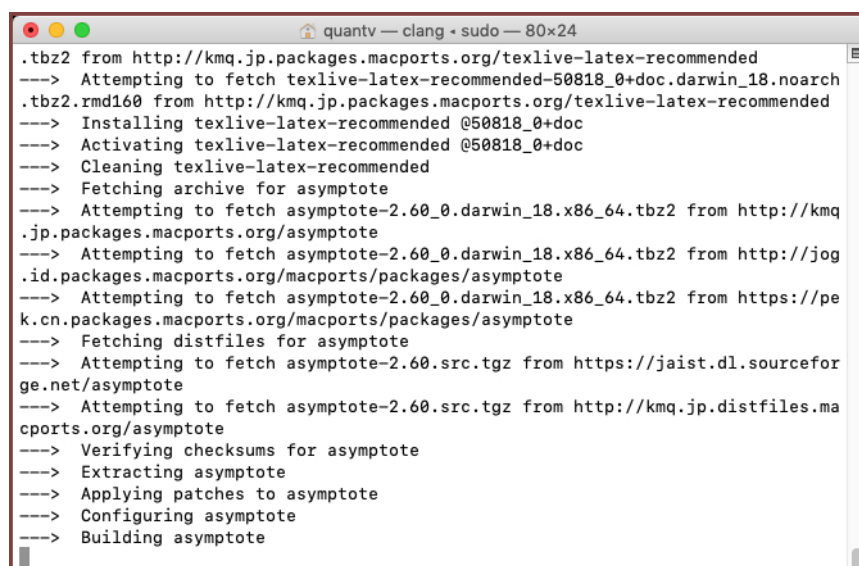
ASY cũng có thể được cài đặt qua **MacPorts**. Download **MacPorts** cho phiên bản **MacOS** tương ứng theo link <https://www.macports.org/install.php>. Trong ví dụ này là bản **macOS Mojave v10.14**. File download được là **MacPorts-2.6.2-10.14-Mojave.pkg** lưu vào thư mục **Downloads**. Sau khi *download*, nhấp đúp vào file này để tiến hành cài đặt MacPorts. Trong quá trình cài đặt có thể yêu cầu cài **XCode** hoặc **gcc**.



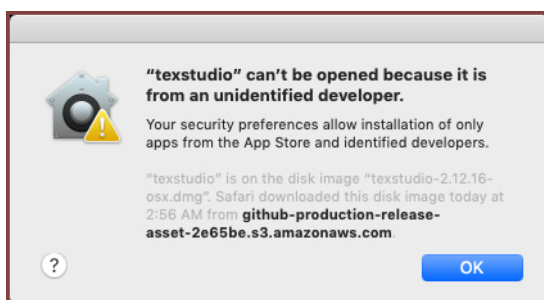
Kết thúc cài **MacPorts**, chạy **Terminal** của MacOS và chạy lệnh sau để cài đặt **asymptote**:

```
$ sudo port install asymptote
```

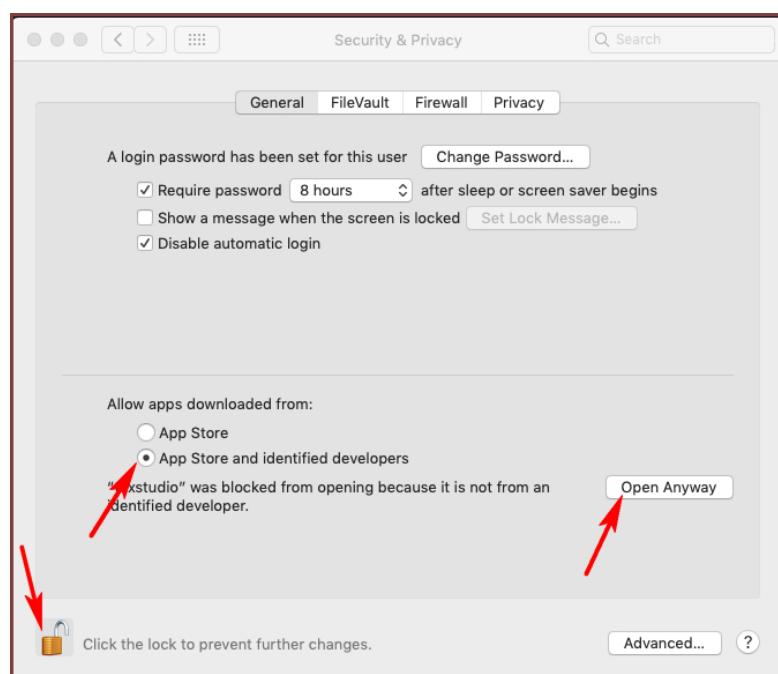
Khi đó **MacPorts** sẽ download source **asymptote-x.x.x.src.tar** và khá nhiều gói cần thiết trong đó có cả **texlive** nên không cần cài **TeXLive**. Kết thúc cài đặt, chạy lệnh **asy** để kiểm tra phiên bản *asy* nào.



Download **TexStudio** theo link <https://www.texstudio.org/#download>. Sau khi download ta được file **texstudio-x.x.x-osx.dmg** (x.x.x là phiên bản của **TexStudio**, ví dụ **texstudio-2.12.16-osx.dmg**). Nhấp đúp vào file để mở file *dmg*, sẽ xuất hiện chương trình **TexStudio**, nhấp đúp để chạy nó.



Nếu có thông báo **unidentified developers** (ứng dụng của nhà phát triển chưa được xác thực), chọn menu *System Settings* → *Security & Privacy*. Ở tab **General** nhấp vào biểu tượng khóa để mở khóa. Sau đó chọn **App Store and identified developers**.



Việc cấu hình **Asymptote** trên **TexStudio** tương tự như trên **Windows**, tuy nhiên lưu ý chọn menu là *TexStudio* → *Preferences*.

Trên MacOS, file **asymptote.qnfa** được lưu trong thư mục `~/.config/texstudio/languages/asymptote.qnfa`.

2 Vẽ hình học phẳng với ASY cơ bản

ASY có gói **geometry.asy** dành riêng cho vẽ hình học. Trước khi hướng dẫn sử dụng gói **geometry.asy**, chúng tôi hướng dẫn vẽ hình học phẳng các lệnh cơ bản của ASY.

Để vẽ hình học phẳng, cần chú ý những vấn đề sau:

- Điểm, đoạn thẳng, đường thẳng, đường tròn, cung tròn.
- Đường vuông góc, đường song song, đường trung trực, đường phân giác.
- Các phép biến hình như tịnh tiến, vị tự, quay, đối xứng trục.

Để thuận tiện cho việc vẽ hình, chúng tôi cung cấp gói **geo2d.asy** bao gồm các hàm được lập trình dựa trên các lệnh cơ bản của ASY. File **geo2d.asy** được lưu cùng thư mục với các file cần dịch. Khi lập trình cần sử dụng `import geo2d.asy;` để sử dụng.

2.1 Kiểu dữ liệu

Khi vẽ hình học cần quan tâm đến các kiểu dữ liệu sau:

a. void

Kiểu dữ liệu **void** chỉ được sử dụng bởi các hàm mà không trả về bất cứ giá trị nào.

b. bool

Kiểu **boolean** chỉ có giá trị **true** hoặc **false**.

c. int

Kiểu nguyên là một số nguyên nằm trong khoảng *intMin* và *inMax*. Nếu không chọn giá trị nào thì sẽ sử dụng số nguyên 0.

d. real

Kiểu thực là một số thực nằm trong khoảng *realMin* và *realMax* có độ chính xác *realEpsilon*. Nếu không chọn giá trị nào thì sẽ sử dụng số thực 0.0.

e. pair

Kiểu **pair** (điểm) được xác định bằng cặp tọa độ (x, y) .

pair *z* có thể hiểu theo các cách sau:

- Là một số phức, $z = x + iy$.
- Là một cặp tọa độ (x, y) , ngầm định là $(0.0, 0.0)$.
- Vector \overrightarrow{Oz} ($O = (0, 0)$ là gốc tọa độ).

Do định nghĩa **pair** là một class có 2 thành phần thực x, y nên có thể sử dụng toán tử truy cập $(.)$ để truy cập đến các thành phần.

Một số hàm liên quan đến **pair**:

<code>pair A + pair B</code>	Trả về $(A.x + B.x, A.y + B.y)$.
<code>pair A - pair B</code>	Trả về $(A.x - B.x, A.y - B.y)$.
<code>pair conj(pair z)</code>	Trả về liên hợp của z : $(z.x, -z.y)$.
<code>real length(pair z)</code>	Trả về module của z : $ z = \sqrt{z.x^2 + z.y^2}$.
<code>real abs(pair z)</code>	Tương tự hàm <i>real length(pair z)</i> .

<code>real angle(pair z, bool warn=true)</code>	Trả về góc của z tính theo <i>radian</i> hoặc độ, hoặc trả về 0 nếu
<code>real degrees(pair z, bool warn=true)</code>	tham số $warn = false$ và $z = (0, 0)$.

<code>pair unit(pair z)</code>	Trả về <i>vector</i> đơn vị theo hướng của <i>pair z</i> .
<code>pair expi(real angle)</code>	Trả về <i>vector</i> đơn vị theo hướng của <i>angle</i> được tính theo radian.
<code>pair dir(real degrees)</code>	Trả về <i>vector</i> đơn vị theo hướng của <i>angle</i> được tính theo độ.
<code>real xpart(pair z)</code>	Trả về thành phần x của pair z : $(z.x)$.
<code>real ypart(pair z)</code>	Trả về thành phần y của pair z : $(z.y)$.

f. string

Kiểu chuỗi là một chuỗi ký tự nằm trong cặp $(")$.

g. path

Trên ASY, một **path** (đường) là tập hợp của các **pair** (điểm) hoặc các path (đường) được nối với nhau bằng `--` (đoạn thẳng) hoặc `..` (đường cong *Bezier*). Khi vẽ hình học phẳng chủ yếu dùng `--`.

Các hàm **circle**, **arc**, **ellipse**, ... đều sử dụng kiểu dữ liệu **path**.

2.2 Các phép biến hình

a. shift - phép di chuyển

```
transform shift(pair z);      Di chuyển đến điểm z.  
transform shift(real x, real y); Di chuyển đến điểm có tọa độ (x,y).
```

Ví dụ:

```
path c0=shift((2,4))*unitcircle;      Di chuyển đường tròn đơn vị đến điểm có tọa độ (2,4).
```

b. scale (xscale, yscale) - phép biến hình tỷ lệ

```
transform xscale(real x);      Biến hình theo trục x theo tỷ lệ x.  
transform yscale(real y);      Biến hình theo trục y theo tỷ lệ y.  
transform scale(real s);       Biến hình theo cả 2 trục với tỷ lệ s.  
transform scale(real x, real y); Biến hình theo trục x tỷ lệ x, trục y tỷ lệ y.
```

c. rotate - phép quay

```
transform rotate(real angle, pair z=(0,0));      Phép quay tâm z, góc angle.
```

d. reflect - phép đối xứng qua đường thẳng

```
transform reflect(pair a, pair b);      Phép đối xứng qua đường thẳng ab.
```

e. scale - phép vị tự (geo2d.asy)

```
transform scale(pair c, real k);      Phép vị tự tâm c, tỷ lệ k.
```

Cách sử dụng các phép biến hình này sẽ được chi tiết trong các ví dụ tiếp theo.

2.3 Điểm

Điểm (pair) trên ASY được khai báo như sau:

```
pair A=(x,y);      Khai báo điểm A có tọa độ (x,y).
```

```
pair A[];      Khai báo mảng các điểm A[i], bắt đầu từ A[0]. Sau đó có thể gán  
giá trị cho các điểm.
```

Có thể định nghĩa điểm theo một số cách sau:

```
pair midpoint(path p);      Trả về điểm chính giữa của path p.
```

```
pair M=midpoint(A--B);      Trả về điểm M là trung điểm của đoạn AB.
```


`pair M=midpoint(arc(O,B,C));` Trả về điểm M là điểm chính giữa cung \widehat{BC} , tâm O.

`pair M=(A+0.3*abs(A-B));` Trả về điểm M trên AB sao cho $AM = 0.3AB$. Hàm $asb(A-B)$ trả về độ dài đoạn thẳng AB.

`pair Q=projection(P,A,B);` Trả về điểm Q là hình chiếu vuông góc của điểm P lên A-B (gói `geo2d.asy`).

`pair D=reflect(B,C)*A;` Trả về điểm D đối xứng với A qua BC.

`pair B=rotate(180,O)*A;` Trả về điểm B đối xứng với A qua O.

`pair B=rotate(k,O)*A;` Trả về điểm B là ảnh của A qua phép quay tâm O, góc k.

Một số điểm đặc biệt trong tam giác: trọng tâm, trục tâm, tâm ngoại tiếp, tâm nội tiếp, tâm bàng tiếp, tâm đường tròn *Mixtilinear*, tâm đường tròn 9 điểm, điểm *Gergonne*, điểm *Feuerbach*, điểm *Nagel*, ... tham khảo trong file `geo2d.asy` để biết tên hàm và các tham số.

Ví dụ sử dụng hàm `pair fepoint(pair A, pair B, pair C)` và `pair feexpoint(pair A, pair B, pair C)` để khai báo điểm *Ferbach*:

`pair Fe=fepoint(A,B,C);` Trả về điểm *Feuerbach* trong của tam giác $\triangle ABC$.

Trả về điểm *Feuerbach* ngoài ứng với đỉnh A của tam giác $\triangle ABC$. Lưu ý có 3 điểm Fe ngoài, để khai báo Fb, Fc chỉ cần thay đổi tham số đầu tiên của hàm: `pair Fb = feexpoint(B,C,A);`
`pair Fc = feexpoint(C,A,B);`

Để vẽ điểm có thể dùng lệnh `draw` hoặc `dot`:

`pair A=(1.1,4);`
`dot(Label("A"),align=NW),A);` Khai báo điểm A, gán nhãn "A".

Tham số `align` để xác định vị trí của nhãn theo hướng bao gồm N (bắc), S (nam), E (đông), W (tây), NE (đông bắc), NW (tây bắc), SE (đông nam), SW (tây nam).

Có thể thêm giá trị vào `align` để thay đổi khoảng cách từ nhãn đến điểm. Có thể xác định kích thước của nhãn, kích thước và màu của điểm:

`dot(Label("A"),align=2N,fontsize(8pt)),A,2bp);` Nhãn "A" của điểm A kích thước 2bp.

2.4 Đoạn thẳng, đường thẳng

Đoạn thẳng trong ASY kiểu *path*. Nó được khai báo như sau:

`path pt= A--B;` Đoạn thẳng AB

ASY không định nghĩa đường thẳng nên sử dụng hàm `path lineex(pair A, pair B, real a=0.1, real b=a);` trong gói `geo2d.asy` để vẽ trong một số tình huống.

`path LAB0=lineex(A,B);` Đoạn thẳng AB được kéo dài về hai đầu mút 0.1AB.
`path LAB1=lineex(A,B,0.1,0.3);` Đoạn AB được kéo dài về đầu A 0.1AB, đầu B 0.3AB

<code>pair N=parallel(M, A, B);</code>	Trả về điểm N sao cho MN song song với AB (geo2d.asy).
<code>pair N=perpendicular(M, A, B);</code>	Trả về điểm N sao cho MN vuông góc với AB (geo2d.asy).

Cách sử dụng để vẽ sẽ được thể hiện trong các ví dụ tiếp theo.

<code>draw(A--B);</code>	Vẽ đoạn A–B.
<code>draw(A--B^^X--Y);</code>	Vẽ các đoạn AB, XY.
<code>draw(A--B--C--cycle);</code>	Vẽ tam giác $\triangle ABC$.
<code>draw(lineex(A, B, 0.1, 0.3));</code>	Vẽ đoạn AB được kéo dài về đầu A 0.1AB, đầu B 0.3AB.
<code>draw("\$a\$", B--C, 2bp+blue+dashed);</code>	Vẽ đoạn BC kích thước 0.8bp màu xanh và gán tên là “a”.

Ở đoạn mã cuối, lưu ý:

- Cách sử dụng hàm “lồng nhau” như trên để “tiết kiệm” việc khai báo biến. Mới ban đầu việc sử dụng có thể khó khăn, tuy nhiên khi dùng quen sẽ thấy tiện lợi vì đoạn mã trở nên gọn gàng hơn.
- *dashed* là kiểu của *path* (kiểu gạch ngang), có các kiểu như sau: *solid* (ngầm định), *dotted*, *dashed*, *longdashed*, *dashdotted* và *longdashdotted*. Có thể thay vào mã để hình dung về từng kiểu.

2.5 Đường tròn, cung tròn, E-líp

ASY sử dụng hàm *path circle(pair c, real r)* để định nghĩa đường tròn. Nó sẽ trả về một đường cong Bezier gần với đường tròn và nó dựa trên đường tròn đơn vị (*unitcircle*). Từ hàm này, trong gói *geo2d.asy* có thêm các hàm sau:

<code>path cOA=circle1p(O, A);</code>	Đường tròn tâm O, bán kính OA.
<code>path cAB=circle2p(A, B);</code>	Đường tròn đường kính AB.
<code>path cABC=circle3p(A, B, C);</code>	Đường tròn đi qua 3 điểm A, B, C.
<code>pair O=center3p(A, B, C);</code>	Tâm đường tròn đi qua 3 điểm A, B, C.
<code>path cIN=incircle(A, B, C);</code>	Đường tròn nội tiếp tam giác $\triangle ABC$.
<code>path I=incenter(A, B, C);</code>	Tâm nội tiếp tam giác $\triangle ABC$.

Tương tự như *circle*, hàm *path ellipse(pair c, real a, real b)* cũng dựa trên đường tròn đơn vị và phép biến hình tỷ lệ *scale*. ASY chuẩn cũng chỉ có hàm trên về ellipse. Nếu sử dụng về ellipse, parabola, hyperbola và các đường bậc hai nói chung, nên sử dụng gói *geometry.asy* được đề cập ở mục tiếp theo.

2.6 Giao điểm

Xác định giao điểm của các đối tượng trên rất quan trọng. ASY dùng hàm *intersectionpoint(path pt0, path pt1);* hoặc *intersectionpoints(path pt0, path pt1);* để xác định giao điểm của các path.

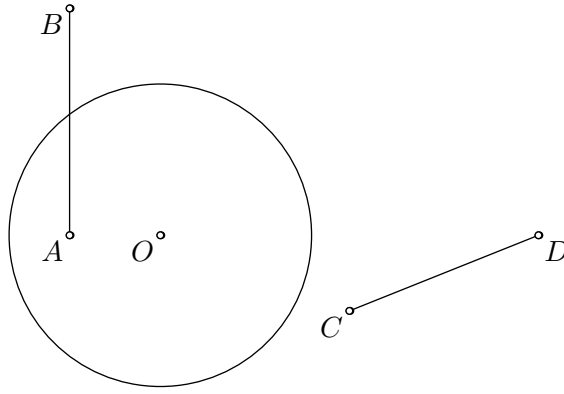
<code>pair I=intersectionpoint(pt0, pt1);</code>	Trả về giao điểm của path pt0, pt1.
<code>pair I[]=intersectionpoints(pt0, pt1);</code>	Trả về các giao điểm của path pt0, pt1.

Nếu hai path là hai đường thẳng thì có thể dùng hàm *pair extension(pair P, pair Q, pair p, pair q)*. Hàm này trả về giao điểm của đường thẳng PQ và đường thẳng pq.

<code>pair H=extension(A, B, X, Y);</code>	Trả về H là giao điểm của hai đường thẳng AB và XY.
--	---

Do *path* là đường có giới hạn nên việc xét giao đôi khi cần phải điều chỉnh.

Ví dụ 1. Cho đường thẳng AB, CD và đường tròn (O) như hình vẽ. Xác định giao điểm của đường thẳng AB và CD; AB, CD với (O).

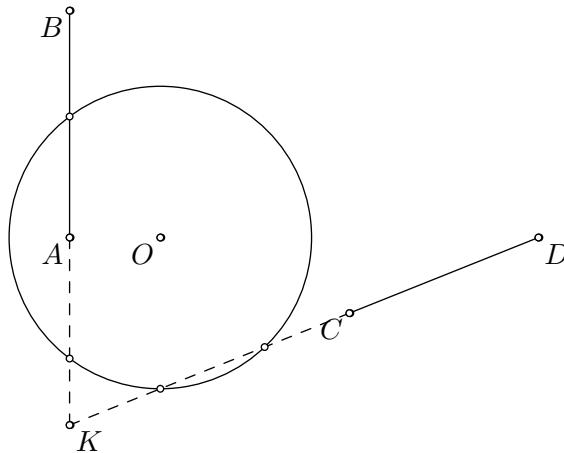


Ở ví dụ này nếu sử dụng:

```
pair e[] = intersectionpoints(A--B, cO);    Chỉ trả về 1 giao điểm của đoạn thẳng AB và (O).
pair g[] = intersectionpoints(C--D, cO);    Không có giao điểm của đường thẳng CD và (O).
pair K = intersectionpoint(A--B, C--D);     Không có giao điểm của đường thẳng AB và CD.
```

Khi đó cần sử dụng hàm *lineex* hoặc *extension* như sau:

```
pair e[] = intersectionpoints(lineex(A, B, 1000), cO);
pair g[] = intersectionpoints(lineex(C, D, 1000), cO);
pair K = extension(A, B, C, D);
```



Khi xét tiếp điểm của đường tròn với đường thẳng và đường tròn với đường tròn cũng cần lưu ý vì hàm *circle* trả về đường cong *Bezier* gần với đường tròn.

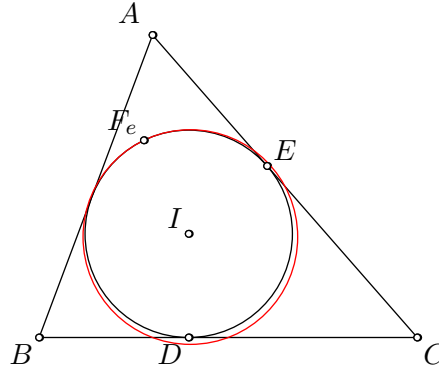
Ví dụ 2. Cho tam giác $\triangle ABC$ và đường tròn nội tiếp $\odot(I)$. Xác định tiếp điểm của $\odot(I)$ với các cạnh của tam giác $\triangle ABC$ và đường tròn 9 điểm.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(1.5,4); dot(Label("$A$"),align=2NW),A);
pair B=(0,0); dot(Label("$B$"),align=SW),B);
pair C=(5,0); dot(Label("$C$"),align=SE),C);
draw(A--B--C--A);
pair I=incenter(A,B,C); dot(Label("$I$"),align=NW),I);
path cIN=incircle(A,B,C); draw(cIN);
pair D=intersectionpoint(incircle(A,B,C),B--C); dot(Label("$D$"),align=SW),D);
pair E=intersectionpoint(cIN,A--C); dot(Label("$E$"),align=NE),E);
path cN9=n9circle(A,B,C); draw(cN9);
pair Fe=intersectionpoint(cIN,cN9); dot(Label("$F_e$"),align=NW),Fe);
```

```
dot(A--B--C--D--E--I--Fe,Fill(white));
```

Khi dịch, ta được kết quả như hình dưới:



Tuy nhiên nếu đổi tọa độ điểm A, ví dụ $pair A=(1.1,4)$; và dịch lại thì sẽ có thông báo “paths do not intersect”. Kiểm tra lỗi, ta sẽ thấy lỗi ở dòng khai báo điểm E hoặc ở dòng khai báo điểm Fe.

Hàm $incircle(A,B,C)$ và $n9circlce(A,B,C)$ là hàm trong gói `geo2d.asy` và không phải nó lỗi. Như đã nói trên, hàm $path circle(pair c, real r)$ thực chất là đường cong Bezier gần với đường tròn nên nó chưa chắc có giao điểm dạng tiếp điểm.

Tiếp điểm D luôn tồn tại do trong hàm $incircle(A,B,C)$ sử dụng điểm D để lấy bán kính. Nếu đổi tọa độ điểm A(1.1,4) sẽ không báo lỗi như trên.

Như vậy khi muốn xác định tiếp điểm giữa đường tròn với đường thẳng và đường tròn với đường tròn, không nên dùng hàm $intersectionpoint$ hoặc $intersectionpoints$. Sử dụng cách khác để xác định tiếp điểm như sau:

```
pair E=projection(I,C,A);
pair Fe=intersectionpoint(lineex(n9center(A,B,C),I,0,10000),cIN);
```

E được xác định là hình chiếu vuông góc của I lên CA. Hàm $lineex(N,I,0,10000)$ trả về đoạn thẳng từ điểm N, kéo dài về phía I $10000*NI$. Tham khảo thêm cách xác định điểm Fe trong hàm $fepoint$ của gói `geo2d.asy`.

2.7 Một số ví dụ

Các ví dụ dưới đây chúng tôi viết trên *TexStudio* và *Notepad++*. Khi dịch ở chế độ lệnh, có thể cần khai báo thêm header.

Việc copy mã từ tài liệu này để chạy có thể gây lỗi. Bạn đọc nên tự gõ các lệnh để có thể kiểm tra được lỗi, hoặc tham khảo mã theo link: <https://github.com/geometryvn/asymptote/tree/master/geo2d/examples>.

Ví dụ 1 (2019 China TST Test 1 Day 1 Q1). Cho ngũ giác $ABCDE$ nội tiếp đường tròn $\odot(O)$ có $AB = AE = CD$. Gọi F là trực tâm tam giác $\triangle ABE$, H là giao BD, CE . Gọi I, J là trung điểm BC, DE . Gọi G là trọng tâm $\triangle AIJ$. OG cắt FH tại M . Chứng minh $AM \perp CD$.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair O=(0,0); dot(Label("$O$"),align=NW,0);
pair A=(0,-3.5); dot(Label("$A$"),align=SW,A);
real k=4.2;
pair B=intersectionpoints(circle1p(O,A),circle(A,k))[0];
dot(Label("$B$"),align=SW,B);
pair E=reflect(O,A)*B; dot(Label("$E$"),align=SE,E);
pair C=rotate(-122,0)*A; dot(Label("$C$"),align=NW,C);
pair D=intersectionpoints(circle1p(O,A),circle(C,k))[0];
dot(Label("$D$"),align=NE,D);
```

- Muốn thay đổi kích thước đường tròn, thay đổi tọa độ điểm A(0,-3.5).
- Muốn thay đổi vị trí các điểm B, C, D, E thì thay đổi giá trị k.
- Cách xác định điểm B là cách gọi các hàm lồng nhau. Để rõ ràng, có thể làm như sau:

- Lưu ý sử dụng các phép biến hình: rotate, reflect.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

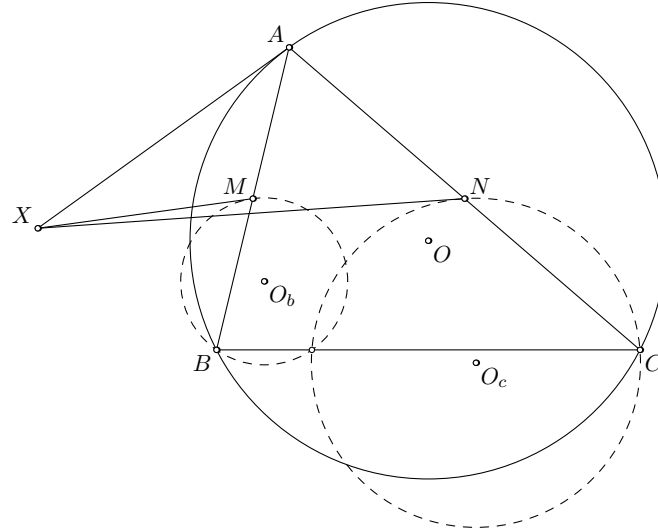
pair A=(1.5,5); dot(Label("$A$",align=NW),A);
pair B=(0,0); dot(Label("$B$",align=SW),B);
pair C=(7,0); dot(Label("$C$",align=SE),C);
pair O=center3p(A,B,C); dot(Label("$O$",align=SE),O);
path c0=circle3p(A,B,C); draw(c0);
pair M=midpoint(A--B); dot(Label("$M$",align=NW),M);
pair N=midpoint(A--C); dot(Label("$N$",align=NE),N);
pair X=scale(A,1.1)*rotate(-90,A)*O; dot(Label("$X$",align=NW),X);
pair Ob=extension(midpoint(M--B),midline(M,B),M,rotate(90,M)*X);
dot(Label("$O_b$",align=SE),Ob);
```

```

pair Oc=extension(midpoint(N--C),midline(N,C),N,rotate(90,N)*X);
dot(Label("$O_c$",align=SE),Oc);
pair t[]=intersectionpoints(circle1p(Ob,B),circle1p(Oc,C));

draw(circle1p(Ob,B)^^circle1p(Oc,C),dashed);
draw(A--B--C--A^^A--X);
dot(A^^B^^C^^M^^N^^O^^X^^Ob^^Oc^^t[1],Fill(white));

```



Ở ví dụ 2, cần lưu ý: Điểm X trên tiếp tuyến tại A của (O) được xác định bằng hai phép biến hình quay và vị tự. Thay đổi giá trị k để thay đổi vị trí điểm X. Có thể viết như sau:

```

pair x=rotate(-90,A)*O;
pair X=scale(A,1.1)*x;

```

Ví dụ 3 (Iran TST 2018, Test 3 Day 2). Cho tứ giác $ABCD$ nội tiếp đường tròn ω . $P \equiv AC \cap BD$. E, F lần lượt nằm trên các cạnh AB, CD sao cho $\angle APE = \angle DPF$. Hai đường tròn ω_1 và ω_2 tiếp xúc với ω lần lượt tại X, Y và cùng tiếp xúc với đường tròn ngoại tiếp tam giác $\triangle PEF$ tại P . Chứng minh $\frac{EX}{EY} = \frac{FX}{FY}$.

```

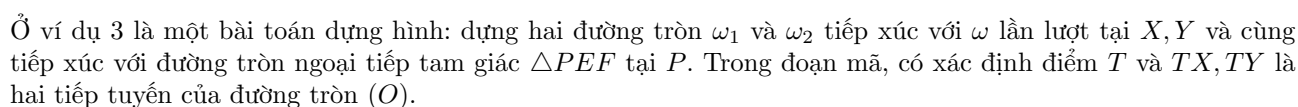
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(0.2,3.5); dot(Label("$A$",align=NW),A);
pair D=(0,0); dot(Label("$D$",align=SW),D);
pair C=(7,0); dot(Label("$C$",align=SE),C);
pair O=center3p(A,C,D); dot(Label("$O$",align=SE),O);
path cO=circle3p(A,C,D); draw(cO);
pair B=rotate(-70,O)*A; dot(Label("$B$",align=NW),B);
pair P=extension(A,C,B,D); dot(Label("$P$",align=NW),P);
real angle=70;
pair E=extension(A,B,P,rotate(-angle,P)*A); dot(Label("$E$",align=NW),E);
pair F=extension(C,D,P,rotate(angle,P)*D); dot(Label("$F$",align=SW),F);

path cP=circle3p(P,E,F); draw(cP,dashed);
pair T=extension(E,F,P,rotate(90,P)*center3p(P,E,F));
//dot(Label("$T$",align=SW),T);
pair X=intersectionpoints(cO,circle1p(T,P))[0]; dot(Label("$X$",align=NE),X);
pair Y=reflect(O,T)*X; dot(Label("$Y$",align=NW),Y);
pair O1=extension(O,X,P,center3p(P,E,F)); draw(circle1p(O1,P));
pair O2=extension(O,Y,P,center3p(P,E,F)); draw(circle1p(O2,P));

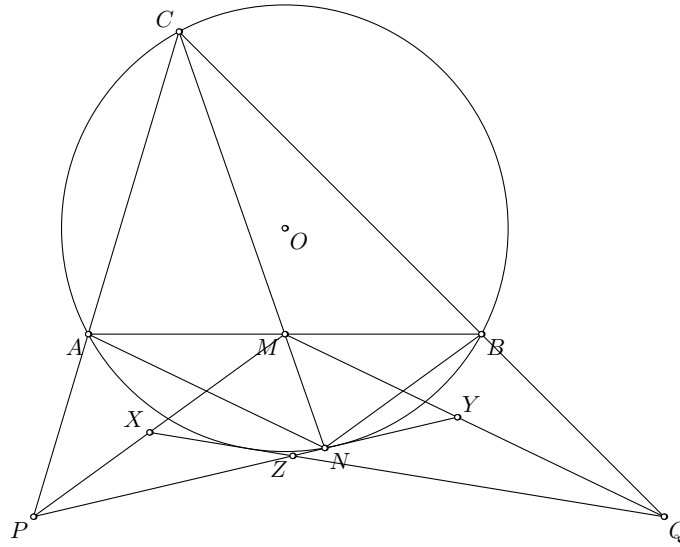
draw(A--B--C--D--A^^A--C^^B--D^^P--E^^P--F);
dot(A^^B^^C^^D^^E^^F^^O^^P^^X^^Y,Fill(white));

```



Ví dụ 4 (International Zhautykov Olympiad 2019 - P3). Cho tam giác $\triangle ABC$. Đường trung tuyến CM cắt lại đường tròn $\odot(ABC)$ tại N . Điểm P và Q lần lượt nằm trên tia CA và CB , thỏa $PM \parallel BN$ và $QM \parallel AN$. Điểm X và Y lần lượt nằm trên đoạn PM và QM , thỏa mãn PY và QX là tiếp tuyến của đường tròn (ABC) . Gọi $Z = PY \cap QX$. Chứng minh rằng tứ giác $MXZY$ ngoại tiếp.

22



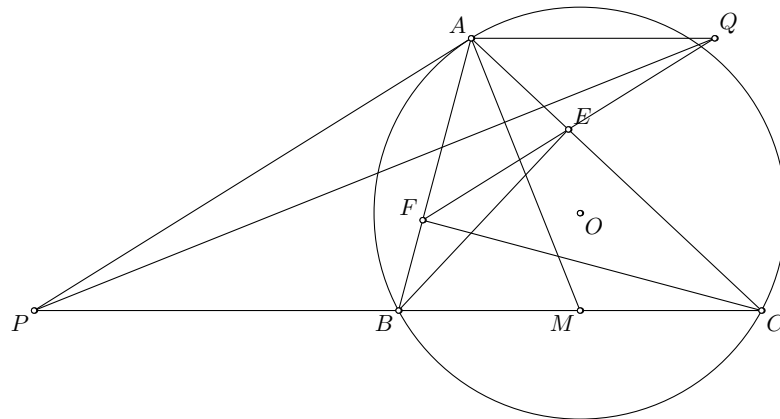
Ở ví dụ 4 là một cách khác để xác định điểm N thay vì xét giao của CM và đường tròn (O) . Do CM đã cắt (O) tại điểm C nên sử dụng cách này để phân biệt giao điểm thứ hai với C . Cách này sẽ hiệu quả khi lập trình các hàm tìm giao.

Ví dụ 5 (Polish MO Finals 2018, Problem 5). Cho tam giác nhọn $\triangle ABC$ ($AB < AC$) nội tiếp đường tròn $\odot(O)$ có BE, CF là hai đường cao. Tiếp tuyến tại A của $\odot(O)$ cắt BC tại P . Đường thẳng qua A song song với BC cắt EF tại Q . Gọi M là trung điểm của BC . Chứng minh $AM \perp PQ$.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(1.2,4.5); dot(Label("$A$"),align=NW,A);
pair B=(0,0); dot(Label("$B$"),align=SW,B);
pair C=(6,0); dot(Label("$C$"),align=SE,C);
pair O=center3p(A,B,C); dot(Label("$O$"),align=SE,O);
path cO=circle3p(A,B,C); draw(cO);
pair E=projection(B,C,A); dot(Label("$E$"),align=NE,E);
pair F=projection(C,A,B); dot(Label("$F$"),align=NW,F);
pair P=extension(B,C,A,rotate(90,A)*O); dot(Label("$P$"),align=SW,P);
pair Q=extension(E,F,A,parallel(A,B,C)); dot(Label("$Q$"),align=NE,Q);
pair M=midpoint(B--C); dot(Label("$M$"),align=SW,M);

draw(A--B--C--A^^B--E^^C--F^^A--M^^A--P^^P--B^^A--Q^^Q--F^^P--Q);
dot(A^^B^^C^^E^^F^^M^^O^^P^^Q,Fill(white));
```



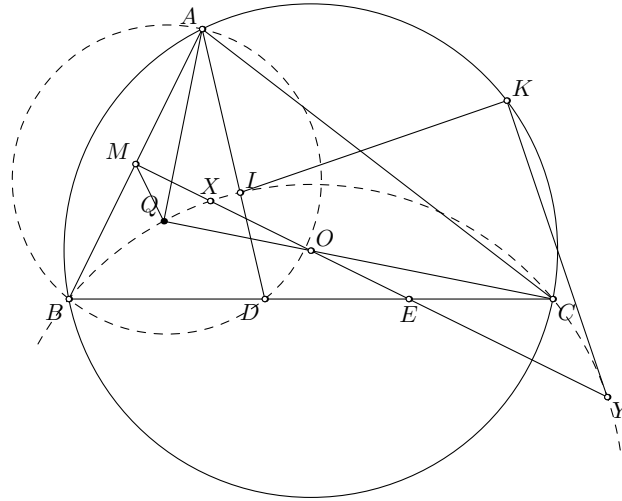
Ví dụ 6 (USA TST 2018). Cho tam giác $\triangle ABC$ nhọn, I là tâm đường tròn nội tiếp, O là tâm đường tròn ngoại tiếp và đường tròn ngoại tiếp Γ . Gọi M là trung điểm đoạn AB . Tia AI cắt BC tại D . Gọi ω và γ lần lượt là các đường tròn ngoại tiếp các tam giác BIC và BAD . Đường thẳng MO cắt ω tại X và Y , trong khi đó

đường thẳng CO cắt ω tại C và Q . Giả sử rằng Q nằm trong tam giác ABC và $\angle AQM = \angle ACB$. Biết rằng $\angle BCA \neq 60^\circ$. Chứng minh rằng các tiếp tuyến từ X, Y của ω và từ A, D của γ đồng quy.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair B=(0,0); dot(Label("$B$",align=SW),B);
pair C=(8,0); dot(Label("$C$",align=SE),C);
pair Ma=midpoint(B--C);
pair O=interp(Ma,rotate(-90,Ma)*B,0.2); dot(Label("$O$",align=NE),O);
draw(circle1p(O,B));
pair L=midpoint(arc(O,B,C));
pair E=intersectionpoint((Ma--C),circle2p(O,L)); dot(Label("$E$",align=S),E);
pair K=reflect(O,E)*L; dot(Label("$K$",align=NE),K);
pair A=reflect(O,E)*B; dot(Label("$A$",align=NW),A);
pair M=midpoint(A--B); dot(Label("$M$",align=NW),M);
pair X=intersectionpoint((M--O),circle1p(L,B)); dot(Label("$X$",align=N),X);
pair Y=intersectionpoint(lineex(O,E,0,4),circle1p(L,B));
dot(Label("$Y$",align=SE),Y);
pair Q=reflect(L,reflect(O,C)*L)*C; dot(Label("$Q$",align=NW),Q);
pair I=incenter(A,B,C); dot(Label("$I$",align=NE),I);
pair D=extension(A,I,B,C); dot(Label("$D$",align=SW),D);

draw(arcex(L,Y,B),dashed);
draw(circle3p(A,B,D),dashed);
draw(A--B--C--cycle^^A--D--M--Y^^K--I^^K--Y^^C--Q^^A--Q^^Q--M);
dot(A^^B^^C^^D^^E^^I^^K^^M^^O^^X^^Y,Fill(white));
```



Ở ví dụ 6, hàm $\text{midpoint}(\text{arc}(O,B,C))$ trả về điểm chính giữa cung nhỏ \widehat{BC} của đường tròn (O) . Để lấy điểm chính giữa cung lớn \widehat{BC} , sử dụng $\text{midpoint}(\text{arc}(O,C,B))$.

Ví dụ này cũng là một bài toán dựng hình: dựng tam giác $\triangle ABC$ sao cho $\angle AQM = \angle ACB$.

Ví dụ 7 (Saudi Arabia TST for IMO 2018, P4). Cho tam giác nhọn không cân $\triangle ABC$. Gọi M, N, P lần lượt là trung điểm BC, CA, AB . Gọi d_1 là đường đi qua M và vuông góc với đường phân giác trong góc $\angle A$ của $\triangle ABC$. Định nghĩa tương tự với d_2 và d_3 . Gọi $D = d_2 \cap d_3, E = d_3 \cap d_1, F = d_1 \cap d_2$. Gọi I, H lần lượt là tâm nội tiếp và trực tâm của $\triangle ABC$. Chứng minh rằng tâm của đường tròn $\odot(DEF)$ là trung điểm IH .

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(0.6,4.8); dot(Label("$A$",align=NW),A);
pair B=(0,0); dot(Label("$B$",align=SW),B);
pair C=(6.5,0); dot(Label("$C$",align=SE),C);
```

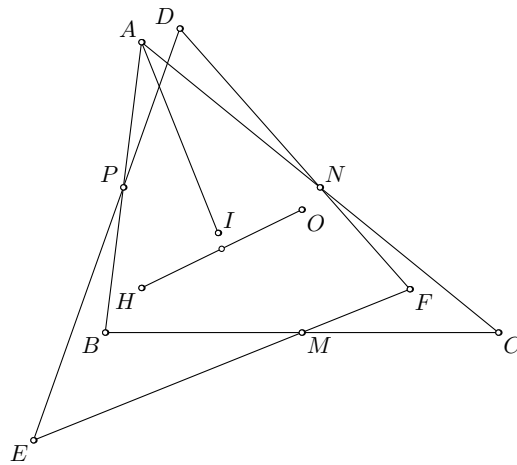


```

pair I=incenter(A,B,C); dot(Label("$I$"),align=NE,I);
pair O=center3p(A,B,C); dot(Label("$O$"),align=SE,O);
pair H=orthocenter(A,B,C); dot(Label("$H$"),align=SW,H);
pair M=midpoint(B--C); dot(Label("$M$"),align=SE,M);
pair N=midpoint(C--A); dot(Label("$N$"),align=NE,N);
pair P=midpoint(A--B); dot(Label("$P$"),align=NW,P);
pair D=extension(N,reflect(B,I)*N,P,reflect(C,I)*P);
dot(Label("$D$"),align=NW,D);
pair E=extension(M,reflect(A,I)*M,P,reflect(C,I)*P);
dot(Label("$E$"),align=SW,E);
pair F=extension(N,reflect(B,I)*N,M,reflect(A,I)*M);
dot(Label("$F$"),align=SE,F);
pair T=midpoint(O--H);

draw(A--B--C--cycle^^D--E^^E--F^^D--F^^H--O^^A--I);
dot(A^^B^^C^^D^^E^^F^^H^^I^^M^^N^^P^^O^^T,Fill(white));

```



Ví dụ 8. (Đề chọn đội tuyển Cần Thơ 2019-2020, Bài 4). Cho tam giác $\triangle ABC$ có đường tròn nội tiếp (I) lần lượt tiếp xúc với CA, AB tại E, F . Gọi D là điểm trên BC sao cho $\angle AID = 90^\circ$. Đường tròn bàng tiếp ứng với đỉnh A, B, C lần lượt tiếp xúc với BC, CA, AB tại P, N, M . Gọi X là giao điểm của (ABC) và (AEF) ($X \neq A$). Chứng minh rằng:

- A, D, X thẳng hàng và AD tiếp xúc với (AMN) .
- Nếu tứ giác $AMPN$ nội tiếp được trong một đường tròn thì AD là tiếp tuyến của (DMN) .

```

import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

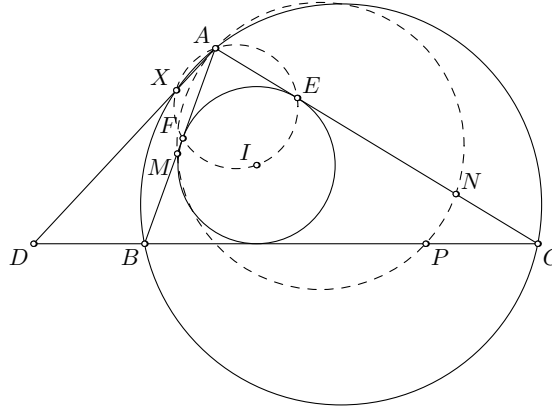
pair B=(0,0); dot(Label("$B$"),align=SW,B);
pair C=(6.5,0); dot(Label("$C$"),align=SE,C);
pair M=midpoint(B--C);
pair O=interp(M,rotate(-90,M)*B,0.2); draw(circle1p(O,B));
pair Ma=midpoint(arc(O,B,C));
pair B1=rotate(180,O)*B;
pair C1=rotate(180,O)*C;
pair I=intersectionpoints(lineex(B1,C1),circle1p(Ma,B))[1];
dot(Label("$I$"),align=NW,I);
pair P=rotate(180,O)*I; dot(Label("$P$"),align=SE,P);
pair A=reflect(O,reflect(Ma,I)*O)*Ma; dot(Label("$A$"),align=NW,A);
pair E=projection(I,C,A); dot(Label("$E$"),align=NE,E);
pair F=projection(I,A,B); dot(Label("$F$"),align=NW,F);
pair N=projection(excenter(B,C,A),C,A); dot(Label("$N$"),align=NE,N);
pair M=projection(excenter(C,A,B),A,B); dot(Label("$M$"),align=SW,M);
pair D=extension(B,C,I,rotate(90,I)*A); dot(Label("$D$"),align=SW,D);
pair X=reflect(O,reflect(A,D)*O)*A; dot(Label("$X$"),align=NW,X);

```

```

draw(incircle(A,B,C));
draw(circle3p(A,E,F)^^circle3p(A,M,N),dashed);
draw(A--B--C--cycle^^D--A^^D--B);
dot(A^^B^^C^^D^^E^^F^^I^^M^^N^^P^^X,Fill(white));

```



Ví dụ 8 thực ra là bài **USA TST 2019, P6**. Câu b là một bài toán dựng hình: dựng tam giác $\triangle ABC$ sao cho đường tròn (MNP) đi qua A .

Ví dụ 9 (Đề chọn đội tuyển PTNK 2019, Bài 4). Cho tam giác $\triangle ABC$ nhọn nội tiếp đường tròn $\odot(O)$ với B, C cố định và A thay đổi trên cung lớn \widehat{BC} . Các đường tròn bàng tiếp góc A, B, C lần lượt tiếp xúc với BC, CA, AB tại D, E, F .

a) Gọi $L \neq A$ là giao điểm của (ABE) và (ACF) . Chứng minh AL luôn đi qua điểm cố định.

b) (BCF) cắt (BAD) tại M, B và (CAD) cắt (CBE) tại N, C . Gọi K, I, J theo thứ tự là trung điểm của AD, BE, CF . Chứng minh KL, IM, JN đồng qui.

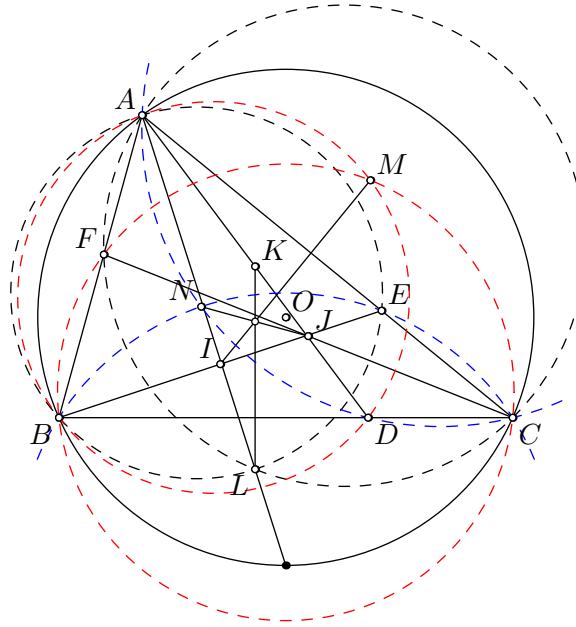
```

import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(1.1,4); dot(Label("$A$"),align=NW,A);
pair B=(0,0); dot(Label("$B$"),align=SW,B);
pair C=(6,0); dot(Label("$C$"),align=SE,C);
pair O=center3p(A,B,C); dot(Label("$O$"),align=NE,O);
draw(circle3p(A,B,C));
pair D=projection(excenter(A,B,C),B,C); dot(Label("$D$"),align=SE,D);
pair E=projection(excenter(B,C,A),C,A); dot(Label("$E$"),align=NE,E);
pair F=projection(excenter(C,A,B),A,B); dot(Label("$F$"),align=NW,F);
pair L=reflect(center3p(A,B,E),center3p(A,C,F))*A; dot(Label("$L$"),align=SW,L);
pair L1=midpoint(arc(O,B,C)); dot(L1);
pair M=reflect(center3p(B,C,F),center3p(B,A,D))*B; dot(Label("$M$"),align=NE,M);
pair N=reflect(center3p(C,A,D),center3p(C,B,E))*C; dot(Label("$N$"),align=NW,N);
pair K=midpoint(A--D); dot(Label("$K$"),align=NE,K);
pair I=midpoint(B--E); dot(Label("$I$"),align=NW,I);
pair J=midpoint(C--F); dot(Label("$J$"),align=NE,J);
pair T=extension(I,M,J,N);

draw(A--B--C--A^^A--L1^^A--D^^B--E^^C--F^^K--L^^I--M^^J--N);
draw(circle3p(A,B,E)^^circle3p(A,C,F),dashed);
draw(circle3p(B,C,F)^^circle3p(B,A,D),dashed);
draw(circle3p(C,A,D)^^circle3p(C,B,E),dashed);
dot(A^^B^^C^^D^^E^^F^^I^^J^^K^^L^^M^^N^^O^^T,Fill(white));

```

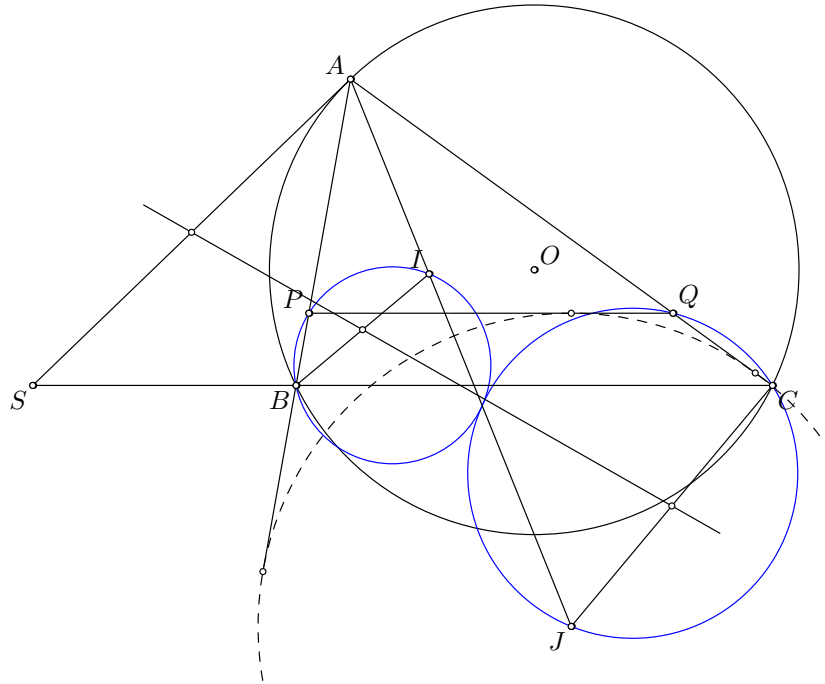


Ví dụ 10 (Bài toán đề nghị cho IGO 2019 - Trần Quân). Cho tam giác $\triangle ABC$ ($AB < AC$) nội tiếp đường tròn $\odot(O)$ có I là tâm nội tiếp. Gọi S là điểm trên BC sao cho SA tiếp xúc với $\odot(O)$. Gọi J là điểm trên AI sao cho trung điểm của các đoạn thẳng AS, BI và CJ cùng nằm trên một đường thẳng. Vẽ đường tròn $\odot(J)$ tiếp xúc với AB, AC . Tiếp tuyến song song với BC của $\odot(J)$ lần lượt cắt AB, AC tại P, Q (J, A nằm khác phía so với PQ). Chứng minh đường tròn $\odot(BPI)$ và đường tròn $\odot(CQJ)$ tiếp xúc với nhau.

```
import geo2d;
unitsize(1cm);
defaultpen(fontsize(11pt));

pair A=(0.8,4.5); dot(Label("$A$"),align=NW,A);
pair B=(0,0); dot(Label("$B$"),align=SW,B);
pair C=(7,0); dot(Label("$C$"),align=SE,C);
path cO=circle3p(A,B,C); draw(cO);
pair O=center3p(A,B,C); dot(Label("$O$"),align=NE,O);
pair I=incenter(A,B,C); dot(Label("$I$"),align=NW,I);
pair S=extension(B,C,A,rotate(90,A)*O); dot(Label("$S$"),align=SW,S);
pair mAS=midpoint(A--S), mBI=midpoint(B--I), mCA=midpoint(C--A);
pair mCJ=extension(mAS,mBI,mCA,parallel(mCA,A,I));
pair J=extension(A,I,C,mCJ); dot(Label("$J$"),align=SW,J);
pair e=projection(J,C,A), f=projection(J,A,B);
path cJ=circle1p(J,e); //draw(cJ,dashed);
pair T=intersectionpoint(cJ,J--reflect(B,C)*J); //dot(Label("$T$"),align=SE,T);
pair P=extension(A,B,T,parallel(T,B,C)); dot(Label("$P$"),align=NW,P);
pair Q=extension(A,C,T,parallel(T,B,C)); dot(Label("$Q$"),align=NE,Q);

draw(circle3p(B,P,I)^^circle3p(C,Q,J),blue);
draw(arcex(J,e,f,20),dashed);
draw(A--B--C--cycle^^A--S^^A--J^^S--B--P--Q--B--I^^C--J^^f--B^^lineex(mAS,mCJ));
dot(A^^B^^C^^I^^J^^O^^P^^Q^^S^^T^^e^^f^^mAS^^mBI^^mCJ,Fill(white));
```



3 Gói geometry.asy và vẽ hình nâng cao

Nếu chỉ cần vẽ hình thì sử dụng ASY cơ bản và gói **geo2d.asy** như mục 2. **Vẽ hình học phẳng với ASY cơ bản** là tạm đủ. Tuy nhiên với những yêu cầu phức tạp hơn như vẽ các đường cong bậc hai, xác định đồng quy, ... và lập trình thì chúng ta cần gói **geometry.asy**.

Gói **geometry.asy** có sẵn trong ASY nên chỉ cần khai báo **import geometry;** là có thể dùng được gói này.

Gói hình học **geometry.asy** định nghĩa thêm nhiều đối tượng (object) thường được sử dụng trong hình học Euclide bằng sử dụng kiểu dữ liệu có cấu trúc (struct) và các hàm (function). Cần phân biệt đối tượng và kiểu của nó. Sau đây là một số kiểu dữ liệu của gói này:

- point: kiểu **point** (điểm) trong hệ tọa độ.
- line và segment: kiểu **line** (đường thẳng, tia), kiểu **segment** (đoạn thẳng).
- conic: kiểu conic bao gồm các kiểu **circle**, **ellipse**, **parabola**, **hyperbola**.
- arc: kiểu **arc** (cung của **ellipse**, **circle**).
- triangle: kiểu **triangle** (tam giác).
- trilinear: kiểu tọa độ trilinear của một điểm đối với một tam giác.

Gói **geometry** cho phép sử dụng hệ tọa độ khác bên trong hệ tọa độ chính, tuy nhiên trong tài liệu này giới hạn ở việc vẽ trong hệ tọa độ chính.

3.1 Kiểu point

Cấu trúc của kiểu dữ liệu **point** như sau:

```
struct point{
    coordsys coordsys;
    restricted pair coordinates;
    restricted real x, y;
    real m = 1;
}
```

Nếu p là kiểu **point**, có thể truy cập các thành viên của p thông qua toán tử truy cập ($.$):

- $p.x, p.y$: tọa độ của điểm p .

Trên hệ tọa độ mặc định, **point** và **pair** khá giống nhau khi sử dụng.

Sau đây là một số hàm của **point**:

<code>bool operator ==(point M, point P)</code>	Kiểm tra $M \equiv N$, trả về giá trị true nếu $MN < EPS$.
<code>bool operator !=(point M, point P)</code>	Kiểm tra $M \neq N$, trả về giá trị true nếu $MN \geq EPS$.

EPS là số thực được sử dụng cho việc tính chính xác. Gói **geometry** định nghĩa $real\ EPS = \text{sqrt}(realEpsilon)$.

<code>real abs(explicit point M)</code>	Hai hàm này cùng trả về module $ M = \sqrt{M.x^2 + M.y^2}$.
<code>real length(explicit point M)</code>	

<code>point conj(explicit point M)</code>	Trả về điểm liên hợp của M : $(M.x, -M.y)$.
<code>real degrees(explicit point M)</code>	Trả về góc (theo độ) của M đối với trục x .
<code>real angle(explicit point M)</code>	Trả về góc (theo radian) của M đối với trục x .
<code>bool finite(explicit point P)</code>	Trả về true nếu P tồn tại.

3.2 Kiểu line, segment

3.2.1 Cấu trúc của line

Cấu trúc của kiểu **line** như sau:

```
struct line {
    restricted point A,B;
    bool extendA,extendB;
    restricted vector u,v;
    restricted real a,b,c;
    restricted real slope, origin;
    line copy(){ }
    void init(){ }
}
```

Nếu m là kiểu **line**, có thể truy cập các thành viên của m thông qua toán tử truy cập ($.$):

- $m.A, m.B$: là hai đầu mút của **line** m .
- $m.a, m.b, m.c$: là hệ số của phương trình $ax + by + c = 0$ trong hệ tọa độ.
- $m.u, m.v$: $u = \text{unit}(AB) = \text{direction vector}$ (vector chỉ phương), $v = \text{normal vector}$ (vector pháp tuyến).
- $m.extendA, m.extendB$: biến *bool* xác định kéo dài về hướng đầu A hay hướng đầu B .
- $m.slope$: độ nghiêng của đường thẳng, $m.slope = -m.a/m.b$.
- $m.origin$: $-m.c/m.b$.

3.2.2 Định nghĩa line

Có thể định nghĩa **line** bởi 2 điểm:

<code>line line(point A, bool extendA=true, point B, bool extendB=true)</code>	Định nghĩa đối tượng kiểu line đi qua hai điểm A, B và hướng từ A đến B . Nếu giá trị của $extendA$ là true thì "line" sẽ đi về phía đầu mút A .
--	--

Định nghĩa **line** bằng phương trình:

<code>line line(real a, real b, real c)</code>	Trả về line với phương trình đường thẳng $ax + by + c = 0$.
--	---

Định nghĩa **line** với *slope* và *origin* (xem cấu trúc của **line**):

<code>line line(real slope, real origin)</code>	Trả về line với tham số độ nghiêng slope và giao với trục y .
---	---

Định nghĩa **line** bằng quan hệ song song:

<code>line parallel(point M, line l)</code>	Trả về đường thẳng qua M song song với l .
<code>line parallel(point M, vector dir)</code>	Trả về đường thẳng qua M có hướng vector dir .
<code>bool parallel(line l1, line l2, bool strictly=false)</code>	Trả về giá trị true nếu $l_1 \parallel l_2$.

Định nghĩa **line** với góc:

<code>line line(real a, point A)</code>	Trả về đường thẳng qua point A và tạo với trục x góc a . Hàm $line(point A, real a)$ có ý nghĩa tương tự.
---	---

<code>line bisector(line l1, line l2, real angle=0, bool sharp=true)</code>	Trả về đường phân giác của góc giữa hai đường thẳng l_1, l_2 . Trong trường hợp muốn xoay đường thẳng đó đi góc a thì xác định tham số $angle = a$.
---	--

Nếu giá trị $sharp = true$ (ngầm định) thì trả về đường phân giác của góc nhọn (góc giữa hai đường thẳng), trường hợp $sharp = false$, trả về đường phân giác của góc tù.

<code>line sector(int n=2, int p=1, line l1, line l2, real angle=0, bool sharp=true)</code>	Trả về đường thẳng thứ p chia góc $\angle(l_1, l_2)$ thành n phần bằng nhau. Tham số $angle$ và $sharp$ tương tự như trên.
---	--

Định nghĩa **line** bằng quan hệ vuông góc:

<code>line perpendicular(point M, line l)</code>	Trả về đường thẳng qua M vuông góc với l .
<code>line perpendicular(point M, vector n)</code>	Trả về đường thẳng qua M vuông góc với vector n .

<code>real angle(line l)</code>	Trả về góc (radian) giữa l và hướng trục x .
<code>real degrees(line l)</code>	Trả về góc (độ) giữa l và hướng trục x .

<code>real sharpangle(line l1, line l2)</code>	Trả về góc nhọn (radian) của l_1, l_2 .
<code>real sharpdegrees(line l1, line l2)</code>	Trả về góc nhọn (độ) của l_1, l_2 .
<code>real angle(line l1, line l2)</code>	Trả về góc (radian) theo hướng của l_1, l_2 .
<code>real degrees(line l1, line l2)</code>	Trả về góc (độ) theo hướng của l_1, l_2 .

3.2.3 Các toán tử áp dụng cho line

Các toán tử (operator) áp dụng cho kiểu line: Các toán tử được dùng là $*$, $/$, $+$, $-$, $l ==$, $l !=$, $@$

<code>line operator *(transform t, line l)</code>	Cho phép xác định ảnh qua phép biến hình.
<code>line operator /(line l, real x)</code>	Trả về đường thẳng đi qua $l.A/x$ và $l.B/x$.
<code>line operator *(point M, line l)</code>	Trả về đường thẳng đi qua $unit(M) * l.A$ và $unit(M) * l.B$.
<code>line operator +(line l, vector u)</code>	Trả về ảnh của l qua phép tịnh tiến theo vector u
<code>line[] operator ^(line l1, line l2)</code>	Trả về mảng gồm 2 đường thẳng l_1, l_2 .
<code>bool operator ==(line l1, line l2)</code>	Trả về giá trị true nếu $l_1 \equiv l_2$
<code>bool operator !=(line l1, line l2)</code>	Trả về giá trị false nếu $l_1 \equiv l_2$
<code>bool operator @ (point m, line l)</code>	Trả về giá trị true nếu M nằm trên đường thẳng l .

3.2.4 Một số hàm liên quan đến line

<code>real distance(point M, line l)</code>	Trả về khoảng cách từ điểm M đến l . Hàm đối biến $real\ distance(line\ l, point\ M)$ cũng được định nghĩa.
<code>bool sameside(point M, point P, line l)</code>	Trả về true nếu M và P cùng phía đối với đường thẳng l .
<code>point[] sameside(point M, line l1, line l2)</code>	Trả về mảng gồm hai điểm: điểm đầu là hình chiếu song song theo phương l_2 của M lên l_1 , điểm thứ hai là hình chiếu song song theo phương l_1 của M lên l_2 .
<code>line complementary(explicit line l)</code>	Nếu l là một tia thì nó trả về tia ngược lại.
<code>bool concurrent(... line[] l)</code>	Trả về true nếu các đường thẳng của mảng $line[]\ l$ đồng quy.
<code>bool perpendicular(line l1, line l2)</code>	Trả về true nếu l_1 và l_2 vuông góc với nhau.
<code>point point(line l, real x)</code>	Trả về điểm giữa của $l.A$ và $l.B$, tương tự như $point(l.A-l.B, x)$.
<code>point relpoint(line l, real x)</code>	Trả về điểm trên đoạn có hướng $[AB]$, tương tự như $l.A+x*vector(l.B-l.A)$.

3.2.5 Cấu trúc của segment

Cấu trúc của kiểu **segment** tương tự **line** (thiếu biến *bool extendA, extendB*):

```
struct segment {
    restricted point A, B;
    restricted vector u, v;
    restricted real a, b, c;
    restricted real slope, origin;
    segment copy() { }

    void init(point A, point B) { }
}
```

3.2.6 Cấu hàm liên quan đến segment

Kiểu **segment** (đoạn thẳng) là một kiểu xuất phát từ kiểu **line** (đường thẳng). Thông qua việc cast (chuyển đổi) thì phần lớn các hàm của **segment** tương tự như **line**.

<code>segment segment(point A, point B)</code>	Trả về đoạn thẳng với đầu mút là A, B .
<code>point midpoint(segment s)</code>	Trả về trung điểm của đoạn thẳng s .
<code>line bisector(segment s, real angle=0)</code>	Trả về đường trung trực của đoạn thẳng s .
<code>line[] complementary(segment s)</code>	Trả về 2 tia có chứa s xuất phát từ $s.A$ và $s.B$.

3.3 Kiểu conic

3.3.1 Cấu trúc của kiểu conic

Gói **geometry** định nghĩa kiểu conic và từ đó định nghĩa các kiểu **circle**, **ellipse**, **parabola** và **hyperbola**. Cấu trúc của kiểu **conic** như sau:

```

    struct conic {
        real e, p, h;
        point F;
        line D;
        line [] l;
    }

```

Nếu *co* là kiểu **conic**, có thể truy cập đến các thành viên của *co* thông qua toán tử truy cập (**.**):

- *co.e*: tâm sai, được dùng để xác định kiểu của đường *conic*. $e = 0$ ta có đường tròn, $0 < e < 1$ ta có đường ellipse, $e = 1$ ta có đường parabola, $e > 1$ ta có đường hyperbola, $e = \infty$ ta có đường thẳng.
- *co.F*: tiêu điểm.
- *co.D*: đường chuẩn.
- *co.h*: khoảng cách từ *F* đến *D*.
- *co.p*: tham số tiêu cự, $p = he$.

3.3.2 Định nghĩa (khai báo) conic

Có thể định nghĩa **conic** theo hai cách như sau:

```
conic conic(point F, line l, real e)
```

Trả về *conic* với tiêu điểm *F*, đường chuẩn *l*, tâm sai *e*.

```
conic conic(point M1, point M2,
            point M3, point M4, point M5)
```

Trả về *conic* không suy biến đi qua các điểm *M1, M2, M3, M4, M5*.

3.3.3 Một số hàm liên quan đến conic

```
int conicnodesnumber(conic co, real angle1,
                    real angle2, bool dir=CCW)
```

Trả về số node được sử dụng để chuyển *conic co* thành *path* nằm giữa góc *angle1, angle2* theo hướng *CW* hoặc *CCW*.

```
point [] intersectionpoints(conic co1,
                          conic co2)
```

Trả về mảng các điểm là giao của 2 conic.

```
point [] intersectionpoints(line l, conic co)
point [] intersectionpoints(conic co, line l)
```

Trả về mảng các điểm là giao của *line* và *conic*.

```
point [] intersectionpoints(triangle t,
                          conic co, bool extended=false)
```

Trả về mảng các điểm là giao của *t* với *co*. Nếu *extended=true* thì các cạnh của tam giác được coi là các đường thẳng. Hàm *intersectionpoints(conic, triangle, bool)* cũng được định nghĩa.

3.3.4 Các toán tử áp dụng cho conic

Các toán tử được dùng cho conic tương tự như line và cũng được áp dụng cho các đối tượng của kiểu conic là circle, parabola, hyperbola.

```
bool operator @(point M, conic co)
conic operator *(transform t, conic co)
```

Trả về **true** nếu điểm *M* nằm trên *co*.
Cho phép xác định ảnh qua phép biến hình.


```
conic operator +(conic co, point M)
```

Trả về ảnh của **co** qua phép tịnh tiến theo \overrightarrow{OM} . Hàm $-(conic, point)$ cũng được định nghĩa.

```
conic operator +(conic co, vector u)
```

Trả về ảnh của **co** qua phép tịnh tiến theo \vec{u} . Hàm $-(conic, vector)$ cũng được định nghĩa.

Kiểu **conic** được sử dụng để tạo các kiểu *circle*, *ellipse*, *parabola*, *hyperbola*, do đó chúng ta sẽ tập trung vào các kiểu tiếp theo này.

3.4 Kiểu circle

3.4.1 Cấu trúc của kiểu circle

Cấu trúc của kiểu **circle** như sau:

```
struct circle {
    point C;
    real r;
    line l;
}
```

Nếu *ci* là kiểu **circle**, có thể truy cập đến các thành viên của *ci* thông qua toán tử truy cập (.):

- *ci.C*: tâm của đường tròn *ci*.
- *ci.r*: bán kính của đường tròn *ci*.
- *ci.l*: đường thẳng được sử dụng để thay thế khi đường tròn suy biến ($ci.r = \infty$)

3.4.2 Định nghĩa circle

```
circle circle(point C, real r)
```

Trả về đường tròn tâm *C*, bán kính *r*.

Lưu ý rằng hàm này khác với hàm ASY chuẩn ở mục 2, *path circle(pair C, real r)* - hàm này trả về *path* là đường cong *Bezier*.

```
circle circle(point A, point B)
```

Trả về đường tròn đường kính *AB*.

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

Trả về đường tròn đi qua 3 điểm *A, B, C*.

```
circle incircle(point A, point B, point C)
```

Trả về đường tròn nội tiếp tam giác $\triangle ABC$.

```
circle excircle(point A, point B, point C)
```

Trả về đường tròn bàng tiếp tiếp xúc với *AB*.

Ở mục này không nhắc tới tâm nội tiếp, tâm ngoại tiếp mặc dù có các hàm **inceter** và **circumcenter**. Có hai lý do:

- Khi đã sử dụng *circle(A, B, C)*, có thể dùng toán tử (.) để truy cập đến tâm, bán kính của đường tròn ngoại tiếp: *circle(A, B, C).C* và *circle(A, B, C).r*.

- Phần tiếp theo chúng tôi sẽ trình bày về kiểu **triangle**, khi đó sẽ có hàm tương tự.

3.4.3 Một số hàm liên quan đến circle

Các hàm áp dụng cho kiểu conic cũng được áp dụng cho kiểu circle, dưới đây là các hàm riêng cho kiểu circle.

```
point radicalcenter(circle c1, circle c2)
```

Trả về chân của trục đẳng phương của *c₁, c₂*.

```
line radicalline(circle c1, circle c2)
```

Trả về trục đẳng phương của *c₁, c₂*.

```
point radicalcenter(circle c1,
                    circle c2, circle c3)
```

Trả về tâm đẳng phương của c_1, c_2, c_3 .

```
line[] tangents(circle c, point M)
point relpoint(circle c, real x)
point angpoint(circle c, real x)
```

Trả về các tiếp tuyến qua M của đường tròn c .

Trả về điểm trên $c = x$ lần chu vi của c .

Trả về điểm trên c với góc x độ

Lưu ý đối với hai hàm *relpoint* và *angpoint*, cần hiểu mốc để tính là điểm giao của tia song song với trục x và đường tròn c . Có thể lấy điểm *relpoint*($c, 0$) để kiểm tra.

3.4.4 Các toán tử áp dụng cho circle

Ngoài các toán tử được áp dụng cho kiểu **conic**, các toán tử sau được định nghĩa cho kiểu **circle**:

```
circle operator *(real x, circle c)
circle operator *(circle c, real x)
```

Trả về đường tròn cùng tâm với đường tròn c và bán kính nhân x lần.

```
real operator ^(point M, circle c)
bool operator @(point M, circle c)
```

Trả về phương tích của M đối với circle c .

Trả về **true** nếu điểm M nằm trên đường tròn c .

3.5 Kiểu ellipse

3.5.1 Cấu trúc của kiểu ellipse

Cấu trúc của kiểu **ellipse** như sau:

```
struct ellipse{
    restricted point F1, F2, C;
    restricted real a, b, c, e, p;
    restricted real angle;
    restricted line D1, D2;
    line l;
}
```

Nếu el là kiểu **ellipse**, có thể truy cập đến các thành viên của el thông qua toán tử truy cập ($.$):

- $el.F1, el.F2$: hai tiêu điểm của el .
- $el.C$: tâm của ellipse el .
- $el.a, el.b$: bán trục lớn, bán trục nhỏ của el .
- $el.c$: bán tiêu cự
- $el.e$: tâm sai.
- $el.p$: tham số tiêu cự (focal parameter).
- $el.angle$: góc xoay của $F1F2$ với trục x ($degrees(F2 - F1)$).
- $el.D1, el.D2$: các đường chuẩn của el .
- $el.l$: đường thẳng thay thế cho ellipse khi ellipse suy biến.

3.5.2 Định nghĩa ellipse

Có thể định nghĩa **ellipse** theo các cách như sau:

`ellipse ellipse(point F1, point F2, real a)` Trả về ellipse với hai tiêu điểm $F1, F2$ và bán trục lớn a .

`ellipse ellipse(point F1, point F2, point M)` Trả về ellipse với hai tiêu điểm $F1, F2$ và đi qua M .

`ellipse ellipse(point C, real a, real b, real angle=0)` Trả về ellipse tâm C , bán trục lớn a , bán trục nhỏ b và góc quay $angle$.

Cũng có thể sử dụng hàm khai báo **ellipse** đi qua 5 điểm tương tự như mục **conic**.

3.5.3 Một số hàm liên quan đến ellipse

`real centerToFocus(ellipse el, real a)`
`real centerToFocus(real a, ellipse el)` Cho phép chuyển một góc từ tâm của **ellipse** đến một góc từ tiêu điểm $F1$.

`line[] tangents(ellipse el, point M)` Trả về các tiếp tuyến qua M của ellipse el .
`point relpoint(ellipse el, real x)` Trả về điểm trên $el = x$ lần chu vi của el .

`point angpoint(ellipse el, real x, fromCenter)` Trả về điểm trên el với góc x , tính từ tâm của el .
`point angpoint(ellipse el, real x, fromFocus)` Trả về điểm trên el với góc x , tính từ tiêu điểm $F1$.

Lưu ý đối với hàm *angpoint*, cần hiểu mốc để tính là điểm giao của tia $\overrightarrow{F1F2}$ với ellipse el .

3.6 Kiểu parabola

3.6.1 Cấu trúc của kiểu parabola

Cấu trúc của kiểu **parabola** như sau:

```
struct parabola{
    restricted point F,V;
    restricted real a,p,e = 1;
    restricted real angle;
    restricted line D;
    pair bmin, bmax;
    void init(){ }
}
```

Nếu P là kiểu **parabola**, có thể truy cập đến các thành viên của el thông qua toán tử truy cập ($.$):

- P.F: tiêu điểm của P .
- P.V: đỉnh của P .
- P.a: khoảng cách từ đỉnh đến tiêu điểm el .
- P.p: khoảng cách từ tiêu điểm và đường chuẩn.
- P.e: tâm sai, $e=1$.
- P.angle: góc xoay của FV với trục x ($degrees(F - V)$).
- P.D: đường chuẩn của P .
- P.bmin, P.bmax: các tọa độ (left, bottom) và (right, top) của hình chữ nhật giới hạn vùng vẽ parabola.

3.6.2 Định nghĩa parabola

Có thể định nghĩa **parabola** theo các cách như sau:

<code>parabola parabola(point F, line l)</code>	Trả về parabola với tiêu điểm F và đường chuẩn l .
<code>parabola parabola(point F, point vertex)</code>	Trả về parabola với tiêu điểm F và đỉnh $vertex$.
<code>parabola parabola(point F, real a, real angle)</code>	Trả về parabola với tiêu điểm F , khoảng cách từ đỉnh đến F và góc xoay $angle$.
<code>parabola parabola(point M1, point M2, point M3, line l)</code>	Trả về parabola có đường chuẩn l và đi qua 3 điểm $M1, M2, M3$.

3.6.3 Một số hàm liên quan đến parabola

<code>line[] tangents(parabola p, point M)</code>	Trả về các tiếp tuyến qua M của parabola p .
<code>point relpoint(parabola p, real x)</code>	Trả về điểm trên p tương tự hàm $relpoint((path)p, x)$.
<code>point angpoint(parabola p, real x)</code>	Trả về điểm trên p với góc quay x .

3.7 Kiểu hyperbola

3.7.1 Cấu trúc của kiểu hyperbola

Cấu trúc của kiểu **hyperbola** như sau:

```
struct hyperbola{
    restricted point F1, F2;
    restricted point C, V1, V2;
    restricted real a, b, c, e, p;
    restricted real angle;
    restricted line D1, D2, A1, A2;
    pair bmin, bmax;
    void init(){ }
}
```

Nếu h là kiểu **hyperbola**, có thể truy cập đến các thành viên của h thông qua toán tử truy cập ($.$):

- $h.F1, h.F2$: tiêu điểm của hyperbola h .
- $h.C, h.V1, h.V2$: tâm và hai đỉnh của h .
- $h.a, h.b$: bán trục lớn, bán trục nhỏ của h .
- $h.c$: bán tiêu cự của h .
- $h.e$: tâm sai của h , $e = c/a$.
- $h.p$: tham số tiêu cự.
- $h.angle$: góc xoay của $F1F2$ với trục x ($degrees(F2 - F1)$).
- $h.D1, h.D2$: đường chuẩn của h .
- $h.A1, h.A2$: đường tiệm cận của h .
- $h.bmin, h.bmax$: các tọa độ (left, bottom) và (right, top) của hình chữ nhật giới hạn vùng vẽ hyperbola.

3.7.2 Định nghĩa hyperbola

Có thể định nghĩa **hyperbola** theo các cách như sau:

```
hyperbola hyperbola(point P1, point P2,
                    real ae, bool byfoci=byfoci)
```

Nếu *byfoci* = *true*, trả về **hyperbola** với bán trục lớn *ae* và hai tiêu điểm *P1, P2*. Nếu *byfoci* = *false*, trả về **hyperbola** với tâm sai *ae* và hai đỉnh *P1, P2*.

```
hyperbola hyperbola(point C, real a,
                    real b, real angle=0)
```

Trả về **hyperbola** với tâm *C*, bán trục lớn *a* theo hướng $C - -C + \text{dir}(\text{angle})$ và bán trục nhỏ *b*.

```
hyperbola conj(hyperbola h)
```

Trả về **hyperbola** liên hợp của *h*.

3.7.3 Một số hàm liên quan đến hyperbola

```
line[] tangents(hyperbola h, point M)
point relpoint(hyperbola h, real x)
```

Trả về các tiếp tuyến qua *M* của hyperbola *h*.

Trả về điểm trên *h* tương tự hàm *relpoint((path)h,x)*.

```
point angpoint(hyperbola h, real x, fromCenter)
```

Trả về điểm trên *h* với góc *x*, tính theo tâm của *h*.

```
point angpoint(hyperbola h, real x, fromFocus)
```

Trả về điểm trên *h* với góc *x*, tính theo tiêu điểm *F1*.

3.8 Kiểu arc

Kiểu **arc** (cung) cho phép định nghĩa một cung thuộc vào **ellipse** (hoặc **circle**).

3.8.1 Cấu trúc của arc

```
struct arc {
    ellipse el;
    restricted real angle0 = 0;
    restricted real angle1, angle2;
    bool direction = CCW;
    polarconicroutine polarconicroutine = currentpolarconicroutine;
    void setangles(){ }
    void init(){ }
}
```

3.8.2 Định nghĩa arc

Hàm định nghĩa một **arc** như sau:

```
arc arc(ellipse el, real angle1, real angle2,
        polarconicroutine polarconicroutine=polarconicroutine(el),
        bool direction=CCW)
```

Trả về một cung của ellipse *el* từ các góc (theo độ) *angle1* đến *angle2* theo hướng *direction* (ngầm định là *CCW* - ngược chiều kim đồng hồ).

Tham số **polarconicroutine** có hai giá trị là **fromFocus** và **fromCenter** được hiểu như ở mục định nghĩa **ellipse**.

Định nghĩa này cũng được sử dụng khi thay **ellipse** thành **circle**.

3.9 Kiểu triangle

3.9.1 Cấu trúc của kiểu triangle

Cấu trúc của kiểu **triangle** như sau:

```
struct triangle {
    restricted point A, B, C;
    struct vertex {
        int n;
        triangle t;
    }
    restricted vertex VA, VB, VC;
    struct side {
        int n;
        triangle t;
    }
    side AB, BC, CA, BA, AC, CB;
}
```

Nếu *tr* là kiểu **triangle**, có thể truy cập đến các thành viên của *tr* thông qua toán tử truy cập (**.**):

- *tr.A*, *tr.B*, *tr.C*: các đỉnh của tam giác.
- *struct vertex*: cấu trúc đỉnh của tam giác *t*, xác định các đỉnh theo biến *int n*. *n* = 1 tương ứng với đỉnh *A*, *n* = 2 tương ứng đỉnh *B*, *n* = 3 tương ứng đỉnh *C* và lặp lại với *n* tiếp theo.
- *tr.VA*, *tr.VB*, *tr.VC*: các biến kiểu *vertex* ở mục trên. Khi đó muốn truy cập đến *n*, sử dụng *tr.VA.n*. Muốn truy cập đến tam giác chứa *tr.VA*, sử dụng *tr.VA.t*!
- *struct side*: cấu trúc cạnh của tam giác *t*, xác định các cạnh theo biến *int n*. *n* = 1 tương ứng với cạnh *AB*, *n* = 2 tương ứng cạnh *BC*, *n* = 3 tương ứng cạnh *CA* và lặp lại với *n* tiếp theo.
- *tr.AB*, *tr.BC*, *tr.CA*, *tr.BA*, *tr.CB*, *t.CA*: các biến kiểu **side** ở mục trên, cách truy cập tương tự như biến kiểu **vertex**.

3.9.2 Định nghĩa (khai báo) triangle cơ bản

Có thể định nghĩa **triangle** theo các cách như sau:

`triangle triangle(point A, point B, point C)` Trả về tam giác với các đỉnh *A, B, C*.

`triangle triangleabc(real a, real b, real c, real angle=0, point A=(0,0))` Trả về tam giác *ABC* với $BC = a$, $CA = b$, $AB = c$ bắt đầu từ đỉnh *A* và góc giữa \vec{AB} với hướng trục *x* là *angle*.

`triangle triangleAbc(real alpha, real b, real c, real angle=0, point A=(0,0))` Trả về tam giác *ABC* từ đỉnh *A* với $\vec{AB}, \vec{AC} = \alpha$, $CA = b$, $AB = c$ và góc giữa \vec{AC} với hướng trục *x* là *angle*.

`triangle triangle(line l1, line l2, line l3)` Trả về tam giác *ABC* với các cạnh lần lượt nằm trên *l1, l2, l3*.

3.9.3 Hàm liên quan đến cấu trúc đỉnh (vertex)

tr là đối tượng kiểu *triangle*, khi đó các **vertex** lần lượt là *tr.VA*, *tr.VB*, *tr.VC*. Gói **geometry** cho phép hàm sử dụng **vertex** như một biến mà không cần chỉ rõ **triangle** của biến này.

<code>point operator cast(vertex V)</code>	Cho phép chuyển đổi vertex thành point .
<code>point point(vertex V)</code>	Trả về đối tượng kiểu point của <i>V</i> .
<code>vector dir(vertex V)</code>	Trả về <i>vector</i> đơn vị trên phân giác trong của đỉnh <i>V</i> .

3.9.4 Hàm liên quan đến cấu trúc cạnh (side)

<code>line operator cast(side side)</code>	Cho phép chuyển đổi side thành line .
<code>line line(side side)</code>	Trả về đối tượng kiểu line của <i>side</i> .
<code>segment segment(side side)</code>	Trả về <i>segment</i> của cạnh <i>side</i> .
<code>side opposite(vertex V)</code>	Trả về cạnh đối diện với đỉnh <i>V</i> .
<code>vertex opposite(side side)</code>	Trả về đỉnh đối diện với cạnh <i>side</i> .

3.9.5 Các toán tử áp dụng cho triangle

<code>triangle operator * (transform T, triangle t)</code>	Cho phép áp dụng các phép biến hình với tam giác <i>t</i> .
--	---

3.9.6 Một số hàm khác

<code>point orthocentercenter(triangle t)</code>	Trả về trực tâm của <i>t</i> .
<code>point foot(vertex V)</code>	Trả về chân đường vuông góc ứng với đỉnh <i>V</i> .
<code>line altitude(vertex V)</code>	Trả về đường cao ứng với đỉnh <i>V</i> .
<code>triangle orthic(triangle t)</code>	Trả về tam giác <i>orthic</i> của <i>t</i> .
<code>point midpoint(side side)</code>	Trả về trung điểm của cạnh <i>side</i> .
<code>point centroid(triangle t)</code>	Trả về trọng tâm của <i>t</i> .
<code>line median(vertex V)</code>	Trả về đường trung tuyến ứng với <i>V</i> .
<code>triangle medial(triangle t)</code>	Trả về tam giác trung điểm của <i>t</i> .
<code>triangle anticomplementary(triangle t)</code>	Trả về tam giác <i>anticomplementary</i> của <i>t</i> .
<code>line bisector(vertex V, real angle=0)</code>	Trả về phân giác của <i>V</i> .
<code>point bisectorpoint(side side)</code>	Trả về chân đường phân giác của <i>side</i> .
<code>line bisector(side side)</code>	Trả về trung trực của cạnh <i>side</i> .
<code>point circumcenter(triangle t)</code>	Trả về tâm ngoại tiếp của <i>t</i> .
<code>circle circle(triangle t)</code>	Trả về đường tròn ngoại tiếp của <i>t</i> .
<code>circle circumcircle(triangle t)</code>	Trả về đường tròn ngoại tiếp của <i>t</i> .
<code>triangle tangential(triangle t)</code>	Trả về tam giác <i>tangential</i> của <i>t</i> .
<code>point incenter(triangle t)</code>	Trả về tâm nội tiếp của <i>t</i> .
<code>circle incircle(triangle t)</code>	Trả về đường tròn nội tiếp của <i>t</i> .
<code>triangle intouch(triangle t)</code>	Trả về tam giác <i>intouch</i> của <i>t</i> .
<code>point excenter(side side)</code>	Trả về tâm bàng tiếp của <i>side</i> .
<code>circle excircle(side side)</code>	Trả về đường tròn bàng tiếp tiếp xúc với <i>side</i> .
<code>point gergonne(triangle t)</code>	Trả về điểm <i>Gergonne</i> của <i>t</i> .
<code>point symmedian(triangle t)</code>	Trả về điểm <i>Symmedian</i> (Lemoine) của <i>t</i> .
<code>line symmedian(vertex V)</code>	Trả về đường đối trung của <i>side</i> .
<code>line cevian(vertex V, point P)</code>	Trả về đường <i>cevian</i> của <i>P</i> đối với đỉnh <i>V</i> .
<code>triangle cevian(triangle t, point P)</code>	Trả về tam giác <i>cevian</i> của <i>P</i> đối với <i>t</i> .
<code>triangle pedal(triangle t, point M)</code>	Trả về tam giác <i>pedal</i> của <i>P</i> đối với <i>t</i> .
<code>triangle antipedal(triangle t, point M)</code>	Trả về tam giác <i>antipedal</i> của <i>M</i> đối với <i>t</i> .

Trong gói geometry có khá nhiều hàm khác liên quan đến tam giác mà chúng tôi không liệt kê ra ở đây.

Đối với tam giác, sẽ có các điểm đặc biệt (ví dụ điểm *Nagel*, điểm *Feuerbach*, ... - trong mục 2, gói **geo2d.asy** cũng đã có một số hàm), các đường thẳng, các tam giác, các đường tròn. Tuy nhiên không nên sử dụng các hàm này cùng với gói geometry. Nên xây dựng các hàm tương tự.

Để xây dựng các hàm, nên xây dựng các điểm đặc biệt trước. Lưu ý hai hàm quan trọng sau:

```
point isogonalconjugate(triangle t,
                        point M)
```

Trả về điểm liên hợp đẳng giác của M đối với t .

```
point isotomicconjugate(triangle t,
                        point M)
```

Trả về điểm liên hợp đẳng cự của M đối với t .

Trước khi xây dựng các điểm đặc biệt của tam giác, hãy đọc tiếp về tọa độ **trilinear** (tọa độ tam pháp tuyến).

3.10 Kiểu trilinear

Trước hết cần hiểu sơ qua về tọa độ **trilinear**, Cho tam giác $\triangle ABC$ và một điểm P bất kỳ. D, E, F lần lượt là hình chiếu vuông góc của P lên BC, CA, AB . Giả sử $PD : PE : PF = x : y : z$, khi đó $x : y : z$ được gọi là tọa độ **trilinear** của P đối với tam giác $\triangle ABC$.

Sau đây là một số tọa độ các điểm cơ bản:

- Tọa độ các đỉnh: $A = 1 : 0 : 0; B = 0 : 1 : 0; C = 0 : 0 : 1$.
- Tâm nội tiếp: $incenter = 1 : 1 : 1$.
- Trọng tâm: $centroid = bc : ca : ab = 1/a : 1/b : 1/c$.
- Tâm ngoại tiếp: $circumcenter = \cos A : \cos B : \cos C$.
- Trực tâm: $orthocenter = \sec A : \sec B : \sec C$.
- Tâm đường tròn 9 điểm: $ninepointcenter = \cos(B - C) : \cos(C - A) : \cos(A - B)$
- Điểm Symmedian: $symmedianpoint = a : b : c$.
- Tâm bàng tiếp: $A - excenter = (-1 : 1 : 1), B - excenter = (1 : -1 : 1), C - excenter = (1 : 1 : -1)$.
- Trung điểm BC: $midpoint(BC) = (0, s/b, s/c)$ (s là diện tích của tam giác).

Tham khảo thư viện các tâm tam giác của Clark Kimberling để biết thêm tọa độ **trilinear** của điểm cần vẽ.

Việc sử dụng tọa độ **trilinear** để vẽ các điểm sẽ trở nên đơn giản hơn rất nhiều và không cần nhớ cách dựng các điểm cần vẽ.

3.10.1 Cấu trúc của kiểu trilinear

Cấu trúc của kiểu **trilinear** như sau:

```
struct trilinear {
    real a, b, c;
    triangle t;
}
```

Nếu ot là kiểu **trilinear**, có thể truy cập đến các thành viên của tr thông qua toán tử truy cập ($.$):

- $ot.a, ot.b, ot.c$: các tọa độ **trilinear** của ot .
- $ot.t$: tam giác để xác định tọa độ.

3.10.2 Định nghĩa (khai báo) trilinear cơ bản

Có thể định nghĩa **trilinear** theo các cách như sau:

<code>trilinear trilinear(triangle t, real a, real b, real c)</code>	Trả về đối tượng <i>trilinear</i> của tam giác <i>t</i> với 3 tọa độ $a : b : c$.
--	--

<code>trilinear trilinear(triangle t, point M)</code>	Trả về đối tượng <i>trilinear</i> của tam giác <i>t</i> với điểm <i>M</i> .
---	---

<code>trilinear trilinear(triangle t, centerfunction f, real a=t.a(), real b=t.b(), real c=t.c())</code>	Trả về đối tượng <i>trilinear</i> của tam giác <i>t</i> với hàm <i>f</i> là một hàm với 3 biến thực.
--	--

Có thể sử dụng việc chuyển đổi (cast) từ point sang trilinear bằng hàm *point(trilinear)* hoặc *(point) trilinear*.

Với việc sử dụng **trilinear**, hàm trả về tâm đường tròn nội tiếp của tam giác *ABC* như sau:

```
point incenter(triangle t) {
    return point(trilinear(t,1,1,1));
}
```

hoặc:

```
point incenter(triangle t) {
    real f(real a, real b, real c){
        return 1;
    }
    return point(trilinear(t,f));
}
```

3.11 Các phép biến hình của gói geometry

3.11.1 Phép biến hình scale

Phép vị tự:

<code>transform scale(real k, point M)</code>	Phép vị tự tâm <i>M</i> , tỷ lệ <i>k</i> .
---	--

Như vậy phép đối xứng tâm có thể sử dụng phép vị tự với tâm tâm vị tự là tâm đối xứng và $k = -1$.

Phép biến hình theo trục:

<code>transform scale(real k, point A, point B, point C, point D, bool safe=false)</code>	
---	--

Trả về phép biến hình Affine **scale** (mục 2.2) với hệ số *k* của trục *AB* theo phương *CD*: xét trường hợp điểm *M*, ảnh của nó là *M'* được xác định: $P + k\overrightarrow{PM}$ khi *P* là hình chiếu của điểm *M* theo phương *CD* lên đường thẳng *AB*. Tham khảo thêm phép chiếu theo phương *CD*.

<code>transform xscale(real k, point M)</code>	Phép biến hình scale (mục 2.2) với tỷ số <i>k</i> , theo trục là đường thẳng qua <i>M</i> và song song với <i>Oy</i> , và theo hướng của <i>Ox</i> .
--	--

<code>transform yscale(real k, point M)</code>	Phép biến hình Affine tỷ số <i>k</i> , theo trục là đường thẳng qua <i>M</i> và song song với <i>Ox</i> , và theo hướng của <i>Oy</i> .
--	---

3.11.2 Phép biến hình projection

Phép chiếu vuông góc:

<code>transform projection(point A, point B)</code>	Phép chiếu vuông góc lên đường thẳng <i>AB</i> .
---	--

Phép chiếu song song:

```
transform projection(point A, point B,
                    point C, point D, bool safe=false)
```

 Phép chiếu lên đường thẳng AB theo phương của đường thẳng CD.

Trong trường hợp $CD \parallel AB$:

- Nếu giá trị của biến **safe** là **true** thì trả về đối tượng của phép chiếu.
- Nếu giá trị của biến **safe** là **false** thì trả về ∞ .

3.11.3 Phép biến hình reflect

Phép đối xứng trục:

```
transform reflect(line l)
```

 Trả về ảnh đối xứng qua đường thẳng l .

Phép đối xứng theo hướng:

```
transform reflect(line l1, line l2,
                bool safe=false)
```

 Trả về ảnh đối xứng qua $l1$ theo hướng $l2$.

Trong trường hợp $l1 \parallel l2$, tương tự như phép chiếu song song.

3.11.4 Phép biến hình rotate

Tương tự như mục 2.2:

```
transform rotate(real angle, point O)
```

 Trả về ảnh qua phép quay tâm O góc $angle$.

3.12 Phép nghịch đảo inversion

3.12.1 Cấu trúc của inversion

Cấu trúc của kiểu **inversion** như sau:

```
struct inversion {
    point C;
    real k;
}
```

Trong đó C là tâm nghịch đảo, k là bán kính nghịch đảo.

3.12.2 Định nghĩa inversion

```
inversion inversion(real k, point C)
inversion inversion(point C, real k)
```

 Trả về phép nghịch đảo tâm C , bán kính k .

```
inversion inversion(circle c1,
                    circle c2, real sgn=1)
```

 Nếu $sgn = 1$ (ngầm định), hàm trả về phép nghịch đảo với bán kính cùng dấu với sgn và biến $c1$ thành $c2$. Nếu $sgn = 0$, hàm trả về phép nghịch đảo có tâm là chân trục đẳng phương và phương tích nghịch đảo là phương tích từ tâm đến $c1$ hoặc $c2$.

<code>inversion inversion(circle c1, circle c2, circle c3)</code>	Trả về phép nghịch đảo có tâm là tâm đẳng phương của 3 đường tròn $c1, c2, c3$, phương tích là phương tích từ tâm đẳng phương đến $c1$.
<code>inversion operator cast(circle c)</code>	Chuyển đổi kiểu circle thành inversion . Ví dụ inversion inv=ci ; trả về phép nghịch đảo đối với đường tròn ci .
<code>circle operator cast(inversion i)</code>	Chuyển đổi kiểu inversion thành kiểu circle . Ví dụ circle ci=inv ; sẽ trả về đường tròn nghịch đảo.

3.12.3 Sử dụng inversion

Các đối tượng được sử dụng với phép nghịch đảo là **point**, **line**, **circle**, **segment** và **triangle**:

<code>point operator *(inversion i, point P)</code>	Nghịch đảo của point P đối với phép nghịch đảo i .
<code>circle operator *(inversion i, line l)</code>	Nghịch đảo của line l đối với phép nghịch đảo i .
<code>circle operator *(inversion i, circle c)</code>	Nghịch đảo của circle c đối với phép nghịch đảo i .
<code>arc operator *(inversion i, segment s)</code>	Nghịch đảo của segment s đối với phép nghịch đảo i .
<code>path operator *(inversion i, triangle t)</code>	Nghịch đảo của triangle t đối với phép nghịch đảo i .

Cần lưu ý nghịch đảo của một **circle** hoặc một **line** có thể là một **line**. Trong trường hợp này đường tròn ảnh C được trả về có bán kính ∞ và kết quả là $C.l$ là đường thẳng thay thế khi đường tròn suy biến.

3.13 Giao điểm trong gói geometry.asy

Tương tự như ASY chuẩn, gói **geometry** dùng hàm `intersectionpoint(object1, object2)`; hoặc `intersectionpoints(object1, object2)`; để xác định giao điểm của các đối tượng đã định nghĩa: conic, circle, ellipse, arc, ... Ví dụ như sau

<code>point intersectionpoint(line l1, line l2)</code>	Trả về giao điểm của hai đường thẳng $l1, l2$.
<code>point[] intersectionpoints(line l, triangle t, bool extended = false)</code>	Trả về các giao điểm của đường thẳng l và tam giác t .
<code>point[] intersectionpoints(line l, circle c)</code>	Trả về các giao điểm của đường thẳng l và đường tròn c .
<code>point[] intersectionpoints(line l, ellipse el)</code>	Trả về các giao điểm của đường thẳng l và ellipse el .

Các hàm xét giao khác bạn đọc có thể tự suy luận và thử, hoặc tìm trong gói **geometry.asy**.

3.14 Một số hàm dựa trên gói geometry

Function 1. Viết hàm trả về đường tròn 9 điểm của tam giác t .

Trước hết chúng tôi viết hàm đường tròn có tâm O và đi qua một điểm A để sử dụng cho các hàm khác:

<code>circle circle1p(point O, point A){ return circle(O, abs(O-A)); }</code>	Trả về đối tượng là circle , tâm O , đi qua A .
---	--

Đường tròn 9 điểm:

<code>point n9center(triangle t){ real A = t.alpha(), B = t.beta(), C = t.gamma(); return point(trilinear(t, Cos(B-C), Cos(C-A), Cos(A-B))); }</code>	
---	--

Hàm $n9center(triangle\ t)$ kiểu **point**, sử dụng biến t là kiểu **triangle** (tam giác). Sử dụng $A = t.alpha()$ để lấy số đo góc A của tam giác t .

Có thể sử dụng $midpoint(orthocentercenter(t)-circumcenter(t))$ để xác định tâm đường tròn 9 điểm. Tuy nhiên ở đây chúng tôi sử dụng toạ độ **trilinear** của tâm đường tròn 9 điểm là $\cos(B - C) : \cos(C - A) : \cos(A - B)$.

```
circle n9circle(triangle t){
    point n9=n9center(t);
    point M=midpoint(t.BC)
    return circle1p(n9,M);
}
```

Trả về đường tròn 9 điểm của tam giác t .

Hàm $n9circle(triangle\ t)$ kiểu **circle** và lần lượt sử dụng các hàm **n9center**, **circle1p** đã viết bên trên.

Ở đây cũng nhận thấy được sự thuận tiện khi sử dụng kiểu dữ liệu mới, kiểu **triangle** so với ASY chuẩn. Khi đó người dùng sẽ không phải gõ 3 biến là 3 đỉnh mà chỉ cần gõ biến tam giác.

Function 2. Viết hàm trả về đường tròn *Mixtilinear* nội tiếp của tam giác t .

```
point mixcenter(point A, point B, point C){
    triangle t=triangle(A,B,C);
    real A = t.alpha(), B = t.beta(), C = t.gamma();
    return point(trilinear(t,(1+Cos(A)-Cos(B)-Cos(C))/2,1,1));
}
```

Do đường tròn *Mixtilinear* phụ thuộc vào các đỉnh nên trước hết chúng tôi viết hàm **mixcenter** với biến là các đỉnh A, B, C . Như vậy hàm $mixcenter(A, B, C)$ trả về tâm đường tròn *Mixtilinear* ứng với cạnh BC . Tương tự với các hàm $mixcenter(B, C, A)$ và $mixcenter(C, A, B)$.

```
real mixradius(point A, point B, point C){
    triangle t=triangle(A,B,C);
    real A = t.alpha();
    return (incircle(t).r/(Cos(A/2)^2));
}
```

Tương tự chúng tôi viết hàm **mixradius** với biến là các đỉnh A, B, C . Cuối cùng là viết hàm trả về tâm và đường tròn **Mixtilinear** nội tiếp với biến kiểu **side** (cạnh).

```
point mixcenter(side side) point op; triangle t = side.t; int n = abs(side.n) - 1; if(n == 1) op = mixcenter(t.A, t.B, t.C); else if(n == 2) op = mixcenter(t.B, t.C, t.A); else op = mixcenter(t.C, t.A, t.B); return op;
```

```
point mixcenter(side side){
    point op;
    triangle t = side.t;
    int n = abs(side.n) - 1;
    if(n == 1) op = mixcenter(t.A, t.B, t.C);
    else if(n == 2) op = mixcenter(t.B, t.C, t.A);
    else op = mixcenter(t.C, t.A, t.B);
    return op;
}

circle mixcircle(side side){
    circle ci;
    triangle t = side.t;
    int n = abs(side.n) - 1;
    if(n == 1) ci = circle(mixcenter(t.A,t.B,t.C),mixradius(t.A,t.B,t.C));
    else if(n == 2) ci = circle(mixcenter(t.B,t.C,t.A),mixradius(t.B,t.C,t.A));
    else ci = circle(mixcenter(t.C,t.A,t.B),mixradius(t.C,t.A,t.B));
    return ci;
}
```

Ở hàm **mixcircle(side side)**, sử dụng biến kiểu **side**. Hàm *if* lồng nhau được sử dụng để gọi các hàm **mixcenter** và **mixradius** với các tham số tương ứng.

Như vậy sử dụng $point\ Ka=mixcenter(t.BC)$; và $mixcircle(t.BC)$ để gọi xác định tâm và đường tròn *Mixtilinear* tương ứng với cạnh BC

Các hàm trên chúng tôi đã lưu vào file **geo2dadv.asy** nên khi sử dụng cần import gói này. Chương trình chính như sau:

```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

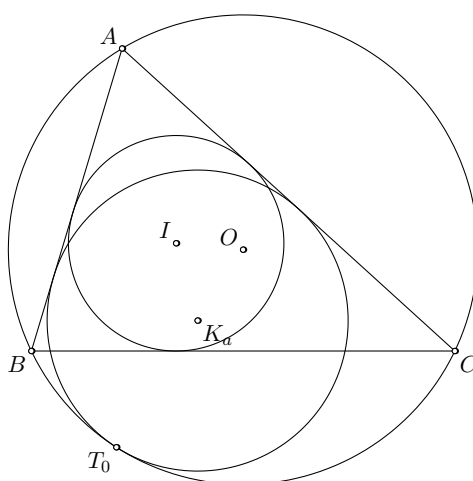
point A=(1.5,5); dot(Label("$A$"),align=NW),A);
point B=(0,0); dot(Label("$B$"),align=SW),B);
point C=(7,0); dot(Label("$C$"),align=SE),C);
triangle t=triangle(A,B,C); draw(t);

point I=incenter(t); dot(Label("$I$"),align=NW),I);
point O=circle(A,B,C).C; dot(Label("$O$"),align=NW),O);

point Ka=mixcenter(t.BC); dot(Label("$K_a$"),align=SE),Ka);

draw(incircle(t)^~circle(A,B,C)^~mixcircle(t.BC));

point T[]=intersectionpoints(mixcircle(t.BC),circle(A,B,C));
dot(Label("$T_0$"),align=SW),T[0]);
dot(A^^B^^C^^I^^O^^Ka^^T[0],Fill(white));
```



Lưu ý các hàm trên được đặt cùng tên, chỉ khác biến, kiểu của biến.

Bài tập đề nghị 1. Viết hàm trả về tiếp điểm của đường tròn *Mixtilinear* nội theo biến kiểu **side**.

Bài tập đề nghị 2. Viết hàm trả về đường tròn *Mixtilinear* ngoại tiếp theo biến kiểu **side**.

Function 3. Viết hàm trả về điểm *Kosnita* của tam giác t .

Có hai cách để viết hàm này, cách thứ nhất theo tọa độ **trilinear** của điểm **Kosnita**, cách thứ hai theo điểm **Kosnita** là điểm liên hợp đẳng giác với tâm đường tròn 9 điểm.

```
point kosnita(triangle t) {
    real A = t.alpha(), B = t.beta(), C = t.gamma();
    return point(trilinear(t,1/Cos(B-C),1/Cos(C-A),1/Cos(A-B)));
}
```

```
point kosnita(triangle t) {
    return isogonalconjugate(t, n9center(t));
}
```

Function 4. Viết hàm trả về điểm *Nagel* của tam giác *t*.

Có hai cách để viết hàm này, cách thứ nhất theo toạ độ **trilinear** của điểm **Nagel**, cách thứ hai theo điểm **Nagel** là điểm liên hợp đẳng cự với điểm **Gergonne**. Lưu ý cách khai báo hàm *f*.

```
point nagel(triangle t){
    real f(real a, real b, real c){
        return (b+c-a)/a;
    }
    return point(trilinear(t,f));
}
```

```
point nagel(triangle t){
    isotomicconjugate(t,gergonne(t));
}
```

Function 5. Viết hàm kiểm tra đường thẳng *l* và đường tròn *c*, nếu tiếp xúc trả về giá trị **true**.

```
bool tangent(line l, circle c){
    point T=projection(l)*c.C;
    return (T @ l);
}

bool tangent(circle c, line l){
    return tangent(l,c);
}
```

Lưu ý ví dụ trên, hàm **bool tangent(circle c, line l)** được viết thêm để thuận tiện cho người dùng khi không nhớ thứ tự các biến.

Bài tập đề nghị 3. Viết hàm kiểm tra hai đường tròn *c1, c2*, nếu tiếp xúc, trả về giá trị **true**.

Function 6. Viết hàm kiểm tra các đường thẳng *l1, l2, l3*, nếu đồng quy trả về giá trị **true**.

```
bool concurrent(line l1, line l2, line l3){
    point T;
    if (parallel(l1,l2)==false) {
        T=intersectionpoint(l1, l2);
    }
    return (T @ l3);
}
```

Function 7. Viết hàm chia đoạn *ab* theo tỷ số *k*.

```
point hmconj(real k, point a, point b){
    point tmp;
    if (k!=-1) tmp=((a.x+k*b.x)/(1+k),(a.y+k*b.y)/(1+k));
    else tmp=midpoint(a--b);
    return tmp;
}
```

Hàm **point hmconj(real k, point a, point b)** trả về điểm chia *ab* theo tỷ số *k*, nếu *k* = 1 sẽ trả về trung điểm của đoạn *ab*.

Function 8. Viết hàm trả về đường tròn **Apollonius** của tam giác $\triangle ABC$.

```
circle apollonius(point A, point B, point C){
    line li= bisector(line(A,B), line(A,C));
    line lj=perpendicular(A,li);
    point D=intersectionpoint(li,line(B,C));
    point E=intersectionpoint(lj,line(B,C));
    return(circle(D,E));
}
```

Có thể sử dụng hàm **point hmconj(real k, point a, point b)** để xác định điểm *D, E* như sau:

```

circle apollonius(point A, point B, point C){
    real k=abs(A-B)/abs(A-C);
    point D= hmconj(k,B,C);
    point E= hmconj(-k,B,C);
    return(circle(D,E));
}

```

Bài tập đề nghị 4. Viết hàm trả về đường tròn **Apollonius** sử dụng kiểu **side**.

3.15 Các ví dụ sử dụng gói geometry

Các ví dụ sau đây sẽ sử dụng thêm gói *geo2dadv.asy*

Ví dụ 1 (2020 USA TST problem 2). Hai đường tròn Γ_1 và Γ_2 các tiếp tuyến chung ngoài ℓ_1, ℓ_2 cắt nhau tại T . Giả sử rằng ℓ_1 tiếp xúc với Γ_1 tại A và ℓ_2 tiếp xúc với Γ_2 tại B . Một đường tròn Ω đi qua A và B cắt lại Γ_1 tại C và Γ_2 tại D sao cho tứ giác $ABCD$ là tứ giác lồi. Giả sử các đường thẳng AC, BD cắt nhau tại X và AD cắt BC tại Y . Chứng minh T, X, Y thẳng hàng.

```

import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

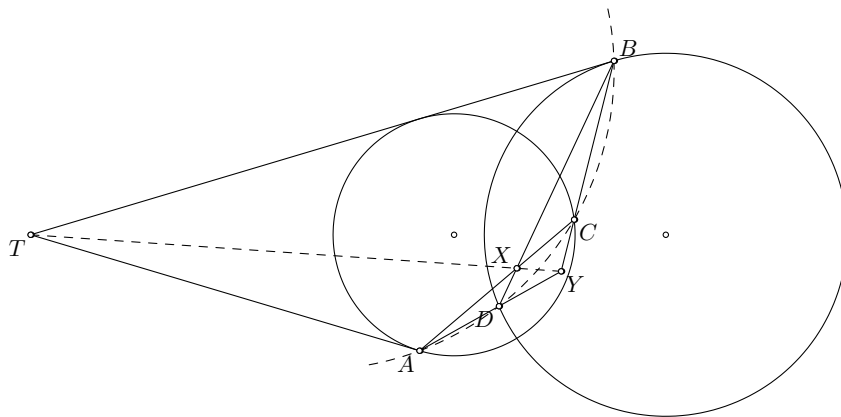
point O1=(0,0), O2=(3.5,0);
circle c1=circle(O1,2), c2=circle(O2,3); draw (c1^^c2);

point T=homocenter(c1,c2)[1]; dot(Label("$T$"),align=SW,T);
line t[] = tangents(c1,c2,1);

point A=projection(t[0])*O1; dot(Label("$A$"),align=SW,A);
point B=projection(t[1])*O2; dot(Label("$B$"),align=NE,B);
point C=relpoint(c1,0.02); dot(Label("$C$"),align=SE,C);
point D=reflect(line(circle(A,B,C).C,O2))*B; dot(Label("$D$"),align=SW,D);
point X=intersectionpoint(line(A,C),line(B,D)); dot(Label("$X$"),align=NW,X);
point Y=intersectionpoint(line(A,D),line(B,C)); dot(Label("$Y$"),align=SE,Y);

draw(T--A^^T--B^^A--C^^B--D^^A--Y^^B--Y);
draw (arcex(circle(A,B,C).C,A,B)^^T--Y,dashed);
dot(A^^B^^C^^D^^T^^X^^Y^^O1^^O2,Fill(white));

```



Ví dụ này sử dụng hai hàm trong gói *geo2dadv.asy*:

```
point [] homocenter(circle c1, circle c2)
```

Trả về tâm vị tự trong và tâm vị tự ngoài của hai đường tròn $c1, c2$.

```
line [] tangents(circle c1,
                 circle c2, int i=0)
```

Trả về hai tiếp tuyến chung trong ($i = 0$) hoặc hai tiếp tuyến chung ngoài ($i \neq 0$) của $c1, c2$.

Ví dụ 2 (2019 Iran RMM TST P1). Cho tam giác $\triangle ABC$ và D là chân đường cao từ đỉnh A . E, F lần lượt trên các đoạn AD, BC sao cho $\frac{AE}{DE} = \frac{BF}{FC}$. Gọi G là chân đường vuông góc từ B lên AF . Chứng minh EF tiếp xúc với đường tròn ngoại tiếp tam giác $\triangle CFG$.

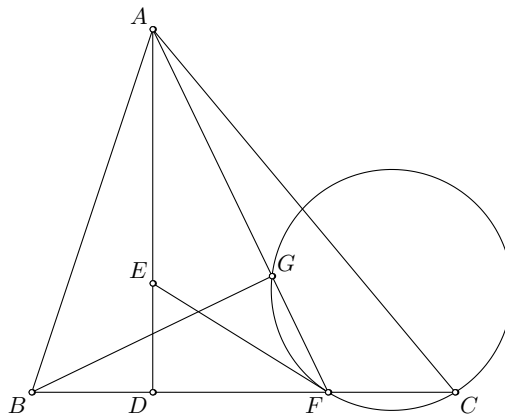
```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

point A=(2,6); dot(Label("$A$"),align=NW),A);
point B=(0,0); dot(Label("$B$"),align=SW),B);
point C=(7,0); dot(Label("$C$"),align=SE),C);

triangle t=triangle(A,B,C);
point D=projection(t.BC)*A; dot(Label("$D$"),align=SW),D);

real x=0.7;
point E=relpoint(A--D,x); dot(Label("$E$"),align=NW),E);
point F=relpoint(B--C,x); dot(Label("$F$"),align=SW),F);
point G=projection(line(A,F))*B; dot(Label("$G$"),align=NE),G);

draw(A--B--C--cycle^^A--D^^A--F^^B--G^^E--F);
draw(circle(C,F,G));
dot(A^^B^^C^^D^^E^^F^^G,Fill(white));
```



Trong ví dụ này, đoạn xác định điểm F có thể dùng hình học thuần túy như sau:

```
point P=intersectionpoint(t.CA,parallel(E,t.BC));
point F=intersectionpoint(t.BC,parallel(P,t.AB)); dot(Label("$f$"),align=NE),F);
```

Nhận thấy gói **geometry** chỉ có hàm **line parallel(point M, line l)** nên khi sử dụng phải nhớ thứ tự biến. Có thể thêm hàm sau:

```
line parallel(line l, point m){
    return parallel(m,l);
}
```

Trả về đường thẳng qua M song song với l , tương tự hàm **line parallel(point M, line l)**.

Bài tập đề nghị 5. Vẽ hình cho bài mở rộng như sau: Cho tam giác $\triangle ABC$ và D là điểm bất kỳ trên cạnh BC . E, F lần lượt trên các đoạn AD, BC sao cho $\frac{AE}{DE} = \frac{BF}{FC}$. Gọi G là điểm trên AF sao cho A, B, D, G đồng viên. Chứng minh EF tiếp xúc với đường tròn ngoại tiếp tam giác $\triangle CFG$.

Ví dụ 3 (2020 CHKMO P3). Cho tam giác cân $\triangle ABC$ có $AB = AC$. Đường tròn nội tiếp $\odot(I)$ lần lượt

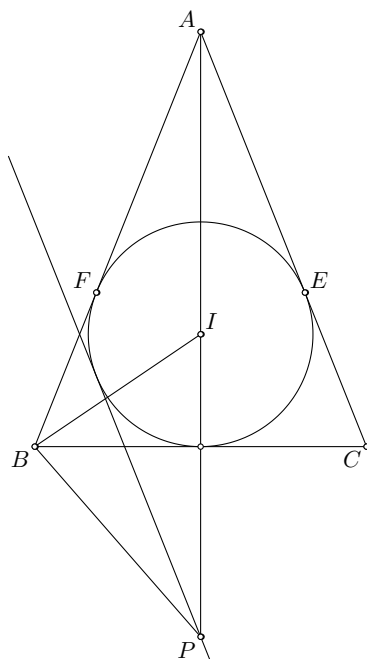
tiếp xúc với AC, AB tại E, F . Hai tiếp tuyến chung ngoài của đường tròn $\odot(AEF)$ và $\odot(I)$ cắt nhau tại P . Giả sử một trong hai tiếp tuyến này song song với AC , chứng minh $\angle PBI = 90^\circ$.

```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

point b=(0,0); //dot(Label("$b$",align=SW),b);
point c=(4,b.y); //dot(Label("$c$",align=SE),c);
point A=((b.x+c.x)/2,5); dot(Label("$A$",align=NW),A);
point I=midpoint(b--c); dot(Label("$I$",align=NE),I);
circle ci=circle(I,distance(I,line(A,b))); draw(ci);
point E=projection(line(A,c)*I; dot(Label("$E$",align=NE),E);
point F=projection(line(A,b)*I; dot(Label("$F$",align=NW),F);
point P=reflect(line(b,c)*A; dot(Label("$P$",align=SW),P);

point D=relpoint(ci,-0.25); //dot(Label("$D$",align=NW),D);
point B=intersectionpoint(line(A,b),perpendicular(D,line(I,D))); dot(Label("$B$",align=SW),B);
point C=intersectionpoint(line(A,c),perpendicular(D,line(I,D))); dot(Label("$C$",align=SE),C);

draw(A--B--C--cycle^^A--P^^B--I^^B--P);
draw(line(P,b));
dot(A^^B^^C^^D^^E^^F^^I^^P,Fill(white));
```



Trong ví dụ trên, lưu ý cách xác định tọa độ điểm c phụ thuộc vào điểm b và A nằm trên trung trực của đoạn bc .

Nhận thấy gói **geometry** chỉ có hàm **line perpendicular(point M, line l)** nên khi sử dụng phải nhớ thứ tự biến. Có thể thêm hàm sau:

```
line perpendicular(line l, point m){
    return perpendicular(m,l);
}
```

Trả về đường thẳng qua M vuông góc với l , tương tự hàm **line perpendicular(point M, line l)**.

Ví dụ 4 (Sharygin Geometrical Olympiad XVI, P17). Các dây cung AB, CD cắt nhau tại E . Giả sử F là ảnh nghịch đảo của E và đường trung trực của EF cắt AC tại G . Chứng minh $EG \parallel BD$.

```
import geometry;
import geo2dadv;
```

```

unitsize(1cm);
defaultpen(fontsize(11pt));

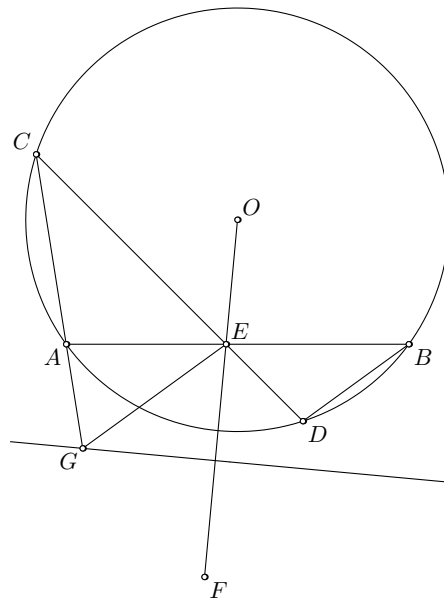
point O=(0,0); dot(Label("$O$"),align=NE,O);
circle co=circle(O,3.5); draw(co);

point A=relpoint(co,0.6); dot(Label("$A$"),align=SW,A);
point B=(-A.x,A.y); dot(Label("$B$"),align=SE,B);
point C=relpoint(co,0.45); dot(Label("$C$"),align=NW,C);
point D=relpoint(co,0.8); dot(Label("$D$"),align=SE,D);
point E=intersectionpoint(line(A,B), line(C,D)); dot(Label("$E$"),align=NE,E);

point F=inversion(co)*E; dot(Label("$F$"),align=SE,F);
point G=intersectionpoint(line(A,C),bisector(segment(E,F))); dot(Label("$G$"),align=SW,G);

draw(A--B--C--D--O--F--C--G--E--G--B--D);
draw(line(G,midpoint(E--F)));
dot(A--B--C--D--E--F--G--O,Fill(white));

```



Ví dụ trên sử dụng hàm `relpoint` để khai báo các điểm A, C, D và sử dụng nghịch đảo đối với đường tròn co .

Bài tập đề nghị 6. Viết hàm kiểm tra hai đoạn thẳng AB, CD , nếu cắt nhau trả về **true**.

Ví dụ 5 (IGO 2019 Advanced, P5). Cho các điểm A, B, C nằm trên parabola Δ sao cho trục tâm H là tiêu điểm của parabola Δ . Chứng minh rằng nếu thay đổi vị trí của các điểm A, B, C trên Δ sao cho trục tâm vẫn là H thì bán kính đường tròn nội tiếp của tam giác $\triangle ABC$ không thay đổi.

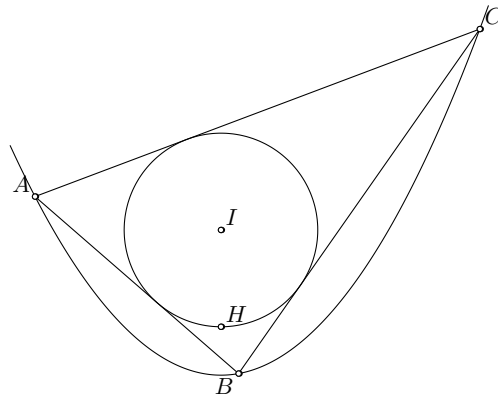
```

import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

point H=(1,1); dot(Label("$H$"),align=NE,H);
point V=(H.x,0.2);
parabola pa=parabola(H,V); draw(pa);
point I=scale(-2,H)*V; dot(Label("$I$"),align=NE,I);
circle ci=circle(I,abs(I-H)); draw(ci);
point A=angpoint(pa,55); dot(Label("$A$"),align=NW,A);
line ta[]=tangents(ci,A);
point B=intersectionpoints(ta[0],pa)[0]; dot(Label("$B$"),align=SW,B);
point C=intersectionpoints(ta[1],pa)[1]; dot(Label("$C$"),align=NE,C);

```

```
draw(A--B--C--cycle);
dot(A--B--C--H--I, Fill(white));
```



Ở ví dụ này sử dụng hàm **parabola parabola(point F, point vertex)** với H là tiêu điểm, V là đỉnh của parabola. Điểm $I = \text{scale}(-2, H) * V$ là kết quả khi làm bài toán.

Ví dụ 6 (China Western 2019, P2). Gọi O, H lần lượt là tâm ngoại tiếp và trực tâm của tam giác nhọn $\triangle ABC$ (với $AB \neq AC$). Gọi M là trung điểm của BC và K là giao của AM và đường tròn ngoại tiếp tam giác $\triangle HBC$ sao cho M nằm giữa A và K . Gọi N là giao của HK và BC . Giả sử $\angle BAM = \angle CAN$, chứng minh $AN \perp OH$.

```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

line tangent(circle c, point M){
    line tmp;
    if (M @ c) tmp=perpendicular(M, line(c.C,M));
    return tmp;
}

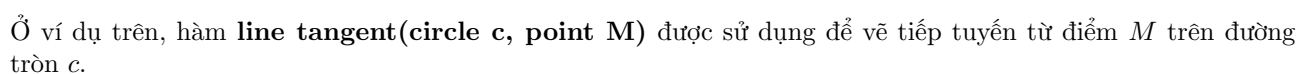
point B=(0,0); dot(Label("$B$"),align=SW),B);
point C=(6,B.y); dot(Label("$C$"),align=NE),C);
point O=((B.x+C.x)/2,1.5); dot(Label("$O$"),align=NE),O);

circle co=circle(O,abs(O-B)); draw(co);
point J=reflect(line(B,C))*O;
point T=intersectionpoint(tangent(co,B),tangent(co,C));

point N=intersectionpoints(circle(J,distance(J,line(T,B))),line(B,C))[0];
dot(Label("$N$"),align=SE),N);
point A=intersectionpoints(co,line(T,N))[1]; dot(Label("$A$"),align=NW),A);

point H=orthocentercenter(A,B,C); dot(Label("$H$"),align=NW),H);
point M=midpoint(B--C); dot(Label("$M$"),align=SW),M);
point K=rotate(180,J)*H; dot(Label("$K$"),align=SE),K);

draw(A--B--C--cycle^^A--K--H--K--A--N);
draw(circle(H,B,C),dashed);
dot(A--B--C--H--K--M--N--O, Fill(white));
```



Đây cũng là một bài toán dựng hình tương đối khó. Chúng tôi sử dụng cách cho hai điểm B, C và đường tròn (O) trước, sau đó tìm cách dựng điểm A thông qua một số kết quả liên quan đến bài toán như AT là đường đối trung của tam giác $\triangle ABC$.

Ví dụ 7 (2019 INMO, P5). Cho AB là đường kính của đường tròn Ω và C là điểm trên Ω khác A, B . Gọi D là chân đường vuông góc từ C lên AB . Gọi K là điểm trên đoạn CD sao cho AC bằng nửa chu vi của tam giác $\triangle ADK$. Chứng minh rằng đường tròn bàng tiếp tương ứng với đỉnh A của tam giác $\triangle ADK$ tiếp xúc với Ω .

Ở bài toán này, hàm **line(i,135)** trả về đường thẳng qua i và tạo với trục x góc 135° . Thực ra chúng tôi cần

dựng đường thẳng qua i và vuông góc với phân giác góc $\angle CDB$ nên sử dụng hàm trên cho thuận tiện. Trong trường hợp AB không song song với trục x , sử dụng hàm trên sẽ không chính xác.

Đây cũng là một bài toán dựng hình. Có thể sử dụng tính toán để dựng điểm K thỏa mãn $2AC = AK + KD + AD$ và $AK^2 - KD^2 = AD^2$. Tuy nhiên chúng tôi dựng trước đường tròn bàng tiếp thỏa mãn: tiếp xúc với AB, CD và tiếp xúc trong với $\odot(AB)$. Đường tròn này là một đường tròn **Thébault**.

Ví dụ 8 (2019 USAMO, P2). Cho tứ giác nội tiếp $ABCD$ có $AD^2 + BC^2 = AB^2$. Các đường chéo của tứ giác $ABCD$ cắt nhau tại E . Gọi P là điểm trên cạnh AB sao cho $\angle APD = \angle BPC$. Chứng minh PE chia đôi CD .

```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

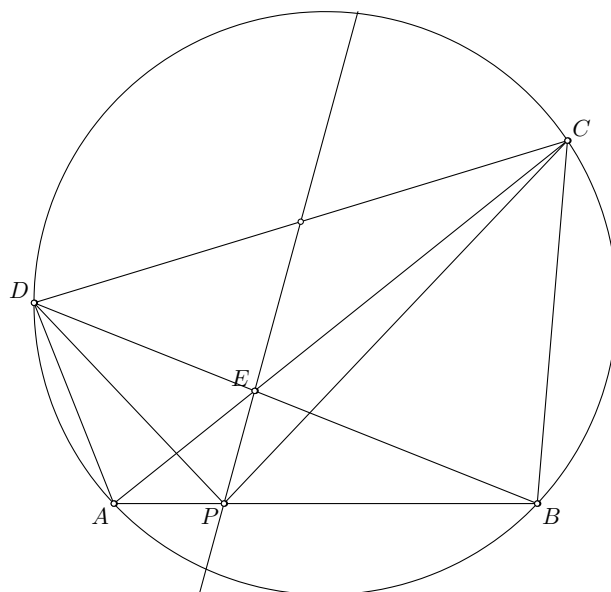
point A=(0,0); dot(Label("$A$"),align=SW,A);
point B=(7,A.y); dot(Label("$B$"),align=SE,B);
point C=(7.5,6); dot(Label("$C$"),align=NE,C);
circle co=circle(A,B,C);

real ad=sqrt(length(A-B)^2-length(B-C)^2);
point D;
point d[]=intersectionpoints(co,circle(A,ad));
for(int i=0; i < d.length; ++i) {
    if (sameside(d[i],C,line(A,B))) D=d[i];
}
dot(Label("$D$"),align=NW,D);

point E=intersectionpoint(line(A,C),line(B,D)); dot(Label("$E$"),align=NW,E);

point F=intersectionpoint(line(A,D),line(B,C));
point G=intersectionpoint(line(A,B),line(C,D));
point Q=intersectionpoint(line(C,D),line(E,F));
point P=projection(line(A,B))*Q; dot(Label("$P$"),align=SW,P);

draw(A--B--C--D--cycle^^A--C^^B--D^^P--C^^P--D);
draw(line(P,E)); draw(co);
dot(A^^B^^C^^D^^E^^P,Fill(white));
```



Trong ví dụ này, để dựng tứ giác $ABCD$, chúng tôi xác định 3 điểm A, B, C , sau đó tính đoạn $AD = \sqrt{AB^2 - AC^2}$. D là giao của $\odot(A, AD)$ và $\odot(ABC)$, tuy nhiên có hai giao $d[0], d[1]$ nên sử dụng hàm **sameside(d[i],C,line(A,B))** để xác định D, C nằm cùng phía so với đường thẳng AB .

Để xác định điểm P , có thể theo cách ngược là dùng `line(E,midpoint(C-D))`. Tuy nhiên chúng tôi sử dụng tính chất hàng điểm điều hoà để có thể áp dụng cho một số trường hợp tương tự.

Hàm `length(A-B)` tương tự hàm `abs(A-B)`.

Ví dụ 9 (China TST Test 2 Day 2, Q5). Gọi M là trung điểm cạnh BC của tam giác $\triangle ABC$. Đường tròn đường kính BC lần lượt cắt AB, AC tại D, E . Gọi P là điểm bên trong tam giác $\triangle ABC$ sao cho $\angle PBA = \angle PAC$, $\angle PCA = \angle PAB$ và $2PM \cdot DE = BC^2$. Điểm X nằm ngoài $\odot(BC)$ sao cho $XM \parallel AP$ và $\frac{XB}{XC} = \frac{AB}{AC}$. Chứng minh rằng $\angle BXC + \angle BAC = 90^\circ$.

```
import geometry;
import geo2dadv;
unitsize(1cm);
defaultpen(fontsize(11pt));

circle apollonius(point A, point B, point C){
    line li= bisector(line(A,B), line(A,C));
    line lj=perpendicular(A,li);
    point D=intersectionpoint(li,line(B,C));
    point E=intersectionpoint(lj,line(B,C));
    return(circle(D,E));
}

point B=(0,0); dot(Label("$B$",align=SW),B);
point C=(5.5,B.y); dot(Label("$C$",align=SE),C);
point O=((B.x+C.x)/2,3.8); //dot(Label("$O$",align=NE),O);
point M=midpoint(B--C); dot(Label("$M$",align=SW),M);
point T=intersectionpoint(perpendicular(B,line(O,B)),perpendicular(C,line(O,C)));
dot(Label("$T$",align=SW),T);

point P=intersectionpoints(circle(O,T),circle(M,abs(T-B)))[0];
dot(Label("$P$",align=NE),P);
point a[]=intersectionpoints(circle(O,abs(O-B)),line(T,P));

point A;
for(int i=0; i < a.length; ++i) {
    if (!sameside(a[i],T,line(B,C))) A=a[i];
}

dot(Label("$A$",align=NW),A);
point D=projection(line(A,B)*C; dot(Label("$D$",align=SW),D);
point E=projection(line(A,C)*B; dot(Label("$E$",align=S),E);

circle cA=apollonius(A,B,C);
point X;
point x[]=intersectionpoints(cA,parallel(M,line(A,P)));
for(int i=0; i < x.length; ++i) {
    if (!inside(circle(B,C),x[i])) X=x[i];
}

dot(Label("$X$",align=NW),X);

draw(A--B--C--cycle^^A--T^^C--D^^B--E^^B--D^^X--M^^X--B^^X--C);
draw(circle(B,C));
dot(A^^B^^C^^D^^E^^M^^P^^T^^X,Fill(white));
```