CHƯƠNG XVIII.
HÀM SỐ VÀ BIỂU THỨC TRONG MAPINFO

Trong các chương trước chúng ta đã sơ lược qua cách sử dụng hàm số và biểu thức ở nhiều phần khác nhau trong quá trình xây dựng và phân tích bản đồ. Việc sử dụng hàm số được thực hiện thông qua biểu thức (*Expression*). Biểu thức cho phép sử dụng một lúc nhiều hàm số thông qua việc phối hợp chúng với nhau nhờ các phép toán cũng như từ khoá. Bởi vì việc sử dụng một hàm số đơn lẻ thường chỉ cho ta những kết quả đơn giản, có tính chất trực quan nhưng việc phối hợp nhiều hàm số trong một biểu thức đôi khi có thể cho ta những kết quả rất ấn tượng đồng thời rút ngắn được nhiều bước trong quá trình phân tích thông tin. Hiểu được chức năng của từng hàm số không phải là việc khó nhưng sử dụng chúng trong trường hợp nào, phối hợp các hàm số với nhau trong biểu thức như thế nào để đạt được kết quả mong muốn lại là một vấn đề không đơn giản. Nó đòi hỏi người sử dụng phải hiểu rõ mục đích mình muốn đạt được cũng như nhớ tính năng của các hàm số để sử dụng.

Hàm số và biểu thức có thể được sử dụng trong rất nhiều ứng dụng của MapInfo. Ta có thể sử dụng hàm số trong các lệnh sau: *Select*, *SQL Select*, *Update Column*, *Create Thematic Map*, và *Layer Control* (*Labels > Label Options*, chọn menu thả xuống của *Label*). Việc sử dụng hàm số gần như không có giới hạn vì chúng có thể được sử dụng vào việc tìm kiếm hay phân tích thông tin, làm chú giải cho một phần bản đồ nào đó, thay đổi và cập nhật thông tin, hiển thị bản đồ theo ý muốn,v.v.... Có thể nói việc sử dụng hàm số và biểu thức là “muôn hình vạn trạng”.

Hàm số khiến cho việc tìm kiếm và phân tích thông tin trong MapInfo được trở nên nhanh chóng và theo ý muốn của người dùng. Các hàm số trong MapInfo được chia thành một số nhóm khác nhau. Ta sẽ lần lượt xem xét chúng.

XVIII.1. CÁC HÀM SỐ TRONG MAPINFO

XVIII.1.1. Các Hàm Toán học

- *Abs(num)*: Trả về giá trị tuyệt đối của một số.
- *Cos(num)*: Trả về cosine của một số; *num* được tính bằng radian. (*1 radian = 57,29578 độ*)
- *Int(num)*: Trả về phần số nguyên của một số.
- *Maximum(num, num)*: Trả về số lớn hơn trong hai số.
- *Minimum(num, num)*: Trả về số nhỏ hơn trong hai số.

- *Round(num1, num2)*: Trả về giá trị của số *num1*, được làm tròn đến giá trị gần nhất với giá trị của *num2* (ví dụ như nếu *num2* là 10 thì *num1* được làm tròn đến giá trị gần với 10 nhất).
- *Sin(num)*: Trả về sine của một số; *num* tính bằng radian.
- *Tan(num)*: Trả về tang của một số; *num* được tính bằng radian.

XVIII.1.2. Các Hàm tổng hợp số liệu của MapInfo (*Aggregate Functions*)

- *Count(*)*: Đếm số lượng bản ghi (hàng) trong một nhóm. Hàm này lấy dấu hoa thị (*) làm đối số vì nó áp dụng cho bản ghi một cách tổng quát và không áp dụng cho một trường đặc biệt nào của bản ghi.
- *Sum (<expression>)*: Tính tổng của các giá trị trong biểu thức <*expression*> cho tất cả các bản ghi trong một nhóm.
- *Avg (<expression>)*: Tính giá trị trung bình của các giá trị trong biểu thức <*expression*> trong tất cả các bản ghi của một nhóm.
- *Max (<expression>)*: Tìm giá trị lớn nhất trong <*expression*> trong tất cả các bản ghi của một nhóm.
- *Min(<expression>)*: Tìm giá trị thấp nhất trong <*expression*> trong tất cả các bản ghi trong một nhóm.

XVIII.1.3. Các Hàm trả về giá trị là vật thể đồ hoạ

- *Buffer(obj , num_res , num_width , str)*: Trả về một vật thể kiểu vùng biểu thị cho một vùng đệm. Thông số *num_res* chỉ định độ phân giải, tính theo số nốt trên một vòng tròn; *num_width* là bán kính của vùng đệm; *str* là tên của đơn vị tính khoảng cách sử dụng trong *num_width*.
- *Centroid(obj)*: Trả về một vật thể điểm có toạ độ tại trọng tâm của vật thể *obj*.
- *CreateCircle(num_x, num_y, num_radius)*: Trả về một vật thể là đường tròn. *num_radius* là khoảng cách tính theo dặm.
- *CreateLine(num_x, num_y, num_x2, num_y2)*: Trả về một vật thể kiểu đường thẳng.
- *CreatePoint(num_x, num_y)*: Trả về một vật thể điểm.

Mỗi hàm số trên trả về một vật thể đồ hoạ. Nếu ta gõ lệnh *Update* trong cửa sổ MapBasic thì ta có thể sử dụng những hàm số này để tạo ra các vật thể cho mỗi hàng trong bảng. Ví dụ, nếu bảng của ta chứa các cột x1, y1, x2 và y2 thì lệnh sau sẽ tạo ra một đường thẳng cho mỗi hàng trong bảng:

```
Update <tên bảng> Set Obj = CreateLine(x1,y1,x2,y2)
```

Lưu ý: Lệnh *Update* định nghĩa lại mỗi vật thể đồ hoạ trong bảng. Ta nên sao lưu lại lớp bản đồ cần làm và thực tập trên bản sao đó.

XVIII.1.4. Các Hàm số trả về các tính toán địa lý

- *Area(obj, str)*: Trả về diện tích của vật thể. Thông số *str* chỉ định tên đơn vị tính diện tích, ví dụ "sq mi" (dặm vuông) hay "sq km" (km²).

- *CentroidX(obj)*: Trả về kinh độ X của trọng tâm của một vật thể.
- *CentroidY(obj)*: Trả về vĩ độ Y của trọng tâm của một vật thể.
- *Distance(num_x, num_y, num_x2, num_y2, str)*: Trả về khoảng cách giữa hai vị trí. Hai thông số đầu tiên xác định giá trị x và y của vị trí bắt đầu; hai thông số tiếp theo xác định giá trị x và y của vị trí kết thúc; thông số *str* là tên đơn vị đo khoảng cách, ví dụ như "mi" hay "km".
- *ObjectLen(obj, str)*: Trả về chiều dài của vật thể. Giá trị *str* xác định tên đơn vị khoảng cách ví dụ như "mi" hay "km". Chỉ có các vật thể là đường thẳng, đường (gấp khúc) và cung là có chiều dài khác không.
- *Perimeter(obj, str)*: Trả về chu vi của vật thể. Giá trị *str* xác định tên đơn vị đo khoảng cách. Chỉ có các vật thể kiểu vùng, ellipse và hình chữ nhật là có chu vi khác không.

XVIII.1.5. Các hàm ngày tháng

- *CurDate( )*: Trả về ngày tháng năm hiện hành.
- *Day(date)*: Trả về phần ngày trong tháng (1 - 31) của ngày.
- *Month(date)*: Trả về phần tháng (1 - 12) của ngày.
- *Weekday(date)*: Trả về phần ngày trong tuần (1 - 7) của ngày, 1 = Chủ Nhật.
- *Year(date)*: Trả về phần năm của ngày.

XVIII.1.6. Các hàm về chuỗi

- *Chr$(num)*: Trả về một ký tự tương ứng với mã ký tự là *num* (ví dụ *chr$(65)* sẽ trả về chuỗi "*A*").
- *DeformatNumber$(str)*: Đảo ngược tác động của hàm *FormatNumber$*, trả về một chuỗi không có các dấu phân cách hàng ngàn.
- *Format$(num , str)*: Trả về một chuỗi biểu thị một số đã được định dạng. Ví dụ `Format$(12345.678,"$,#.##")` trả về giá trị "$12,345.68".
- *FormatNumber$(num)*: Trả về một chuỗi biểu thị một con số đã được định dạng. Hàm số này đơn giản hơn hàm *Format$*, nhưng cho ta ít quyền kiểm soát trong việc định dạng hơn (ví dụ như số định dạng luôn có dấu phân cách hàng ngàn).
- *InStr(num , str1 , str2)*: Tìm kiếm chuỗi *str1* bắt đầu từ vị trí ký tự thứ *num*, và tìm sự hiện diện của phần đó trong chuỗi *str2*. Hàm này trả về vị trí khi tìm thấy kết quả trong *str2*, hoặc trả về 0 nếu không tìm thấy. Để bắt đầu tìm kiếm chuỗi từ đầu, nạp giá trị *num* là một (1).
- *LCase$(str)*: Trả về định dạng chữ viết thường của chuỗi *str*.
- *Left$(str , num)*: Trả về *num* số ký tự đầu tiên của chuỗi *str*.
- *Len(str)*: Trả về số ký tự trong chuỗi *str*.
- *LTrim$(str)*: Cắt bỏ bất kỳ khoảng trắng nào ở đầu chuỗi *str* và trả về kết quả.
- *Mid$(str, num1, num2)*: Trả về một phần của chuỗi *str* bắt đầu từ ký tự thứ *num1*

và dài *num2* số ký tự.

- *Proper$(str)*: Trả về một chuỗi với cách viết hoa đúng kiểu (chữ đầu tiên viết hoa).
- *Right$(str , num)*: trả về *num* số ký tự cuối cùng của chuỗi *str*.
- *RTrim$(str)*: Cắt bỏ bất kỳ khoảng trắng nào ở phần cuối của chuỗi *str* và trả về kết quả.
- *Str$(expr)*: trả về một chuỗi tương ứng với giá trị của biểu thức *expr*.
- *UCase$(str)*: Trả về dạng chữ viết hoa (toàn bộ) của chuỗi *str*.
- *Val(str)*: Trả về giá trị số của một chuỗi; ví dụ *Val("18")* trả về số *18*.

XVIII.2. CÁC TOÁN TỬ VÀ TỪ KHOÁ TRONG BIỂU THỨC

Như đã trình bày sơ bộ ở phần trên, việc hiểu các hàm số đơn lẻ là tương đối đơn giản. Tuy nhiên các hàm số đơn lẻ ít khi nào cho ta một kết quả phân tích trong trường hợp có nhiều biến số. Đối với việc phân tích thông tin như vậy người ta cần phải phối hợp nhiều hàm số với nhau. Các hàm số được liên kết với nhau thông qua các toán tử và các từ khoá. Việc sử dụng nhiều hàm số chung với nhau được thể hiện qua biểu thức. Phần này sẽ trình bày các toán tử và từ khoá để phối hợp các hàm số với nhau trong biểu thức.

XVIII.2.1. Các Toán tử So sánh

Trong MapInfo có các toán tử so sánh sau:

=	bằng
<>	khác
>	lớn hơn
<	nhỏ hơn
>=	lớn hơn hay bằng
=<	nhỏ hơn hay bằng
_	tương tự (tượng trưng cho một ký tự bất kỳ)
%	tương tự (tượng trưng cho nhiều ký tự bất kỳ)

XVIII.2.2. Các Toán tử Toán học

ký hiệu	*ý nghĩa*	*ví dụ*
+	cộng	A+B
-	trừ	A-B, -A (âm A)
*	nhân	A*B
/	chia	A/B
^	lũy thừa	A^B

Đối với các biểu thức có các giá trị được định dạng theo các kiểu khác nhau, MapInfo sẽ xử lý các số liệu như sau:

- Ngày tháng (*Date*) + một số: kết quả được định dạng là ngày tháng (Date).

- Số nguyên (*Integer*) + Số nguyên: kết quả được định dạng là số nguyên (Integer).
- Ngày tháng (*Date*) - một số: kết quả được định dạng là ngày tháng.
- Số - Ngày tháng (*Date*): kết quả được định dạng là ngày tháng (*Date*)
- Số nguyên - Số nguyên: kết quả được định dạng là số nguyên (*Integer*)
- Một số - Một số: kết quả được định dạng là *Float*.
- Số nguyên * Số nguyên: kết quả được định dạng là số nguyên (*Integer*)
- Một số * Một số: kết quả được định dạng là *Float*.
- Một số/Một số: Kết quả được định dạng là *Float*.
- Một số ^ Một số: kết quả được định dạng là *Float*.

Trong MapInfo ta có thể thực hiện được các phép tính toán sau:

- Thêm một số vào một ngày để tạo ra một ngày khác.
- Trừ một ngày cho một số để tạo ra một ngày khác.
- Trừ một ngày cho một ngày để tạo ra một số.

Khi cộng hay trừ một số vào giá trị ngày thì MapInfo xử lý con số đó như là số ngày. Khi cộng hay trừ trong một tuần hãy sử dụng số 7; cộng/trừ trong một tháng hãy sử dụng số 30 hay 31. Khi trừ một ngày cho một ngày thì kết quả sẽ là số ngày (một con số).

XVIII.2.3. Các toán tử Luân lý và các Toán tử Địa lý

XVIII.2.3.1. Các Toán tử Luân lý

Các toán tử "*and*" (và), "*or*" (hoặc) và "*not*" (không) là các toán tử luân lý. Chúng được sử dụng để phối hợp các Biểu thức trong lệnh *Select* và trong mệnh đề *Where condition* của lệnh *SQL Select*. MapInfo coi các biểu thức này là một phép kiểm tra để áp dụng cho mỗi bản ghi (hàng) trong một bảng. Mỗi phép kiểm tra như vậy sẽ trả về câu trả lời là có/không (hay đúng/sai). MapInfo sử dụng các toán tử luân lý để phối hợp các câu trả lời có/không riêng lẻ thành một câu trả lời có/không cuối cùng để giải quyết câu hỏi: bản ghi (hàng) đang được xem xét có thoả điều kiện chọn hay không?

- *and* được coi là đúng ("*true*") khi (và chỉ khi) tất cả tham số của nó (tức là các biểu thức mà nó nối lại) đều đúng. Một bản ghi phải thoả mãn tất cả các điều kiện trong biểu thức thì mới được chọn.
- *or* được coi là đúng khi một,vài hay tất cả các tham số (tức là các biểu thức mà từ khoá này liên kết) đúng. Một bản ghi thoả mãn một trong những điều kiện của biểu thức thì sẽ được chọn. Nó cũng được chọn khi hai hay tất cả điều kiện đều được thoả.
- *not* là đúng khi đối tham số của nó (biểu thức mà nó được sử dụng vào đó) là sai ("*false*"). Một bản ghi sẽ được chọn khi nó không thoả điều kiện đưa ra.

XVIII.2.3.2. Các Toán tử Địa lý

MapInfo có một số toán tử địa lý. Chúng được sử dụng để chọn các vật thể dựa

trên mối quan hệ không gian đối với (các) vật thể khác. MapInfo có một từ khoá đặc biệt (tên trường) để sử dụng trong các toán tử địa lý: "*obj*" hay "*object*" (vật thể). Từ khoá này (được coi như một tên trường) cho MapInfo biết rằng nó phải chọn giá trị dựa trên các vật thể đồ hoạ trong bảng của MapInfo chứ không dựa vào bảng dữ liệu.

Các toán tử địa lý nằm giữa các vật thể đang được xem xét. Ta có thể chọn các toán tử địa lý trong menu thả xuống *Operators*.

Dưới đây là những toán tử địa lý:

Toán tử	**Ý nghĩa**
- *Contains* (Chứa):	Vật thể A chứa vật thể B nếu trọng tâm của vật thể B nằm trong ranh giới của vật thể A.
- *Contains Entire* (Chứa hoàn toàn):	Vật thể A chứa hoàn toàn vật thể B nếu ranh giới của B nằm hoàn toàn trong ranh giới của A.
- *Within* (nằm trong)	Vật thể A nằm trong vật thể B nếu trọng tâm của nó nằm trong ranh giới của B.
- *Entirely Within* (Hoàn toàn nằm trong):	Vật thể A hoàn toàn nằm trong vật thể B nếu ranh giới của nó hoàn toàn nằm trong vật thể B.
- *Intersects* (Giao, Cắt):	Vật thể A giao (cắt) vật thể B nếu chúng có ít nhất một điểm chung.

XVIII.2.4. Các từ khoá trong Biểu thức

MapInfo cho phép sử dụng các từ khoá sau: "*any*" (bất kỳ), "*all*" (tất cả), "*in*" (trong) và "*between*" (trong khoảng). Những từ khoá này phải được gõ vào biểu thức (không có sẵn).

Sử dụng "*any*" để chọn bất kỳ yếu tố nào trong tập hợp các yếu tố. Ta sẽ lấy ví dụ trong lớp bản đồ *cac_tinh*. Ví dụ:

```
Ten = any("Soùc Traêng", "Tieàn Giang", "Vónh Long")
```

Biểu thức này đúng khi tên tỉnh là Sóc Trăng, Tiền Giang, Vĩnh Long (xin nhắc lại là tiếng Việt không hiển thị đúng trong biểu thức).

Ta sẽ xem xét ví dụ sau để hiểu ý nghĩa của "*all*""

```
Ten <> all("Soùc Traêng", "Tieàn Giang", "Vónh Long")
```

Biểu thức này có nghĩa là chọn tất cả các tỉnh ngoại trừ 3 tỉnh Sóc Trăng, Tiền Giang và Vĩnh Long.

Hãy xem chuyện gì xảy ra nếu ta gõ biểu thức sau:

```
Ten <> any("Soùc Traêng", "Tieàn Giang", "Vónh Long")
```

Ví dụ dưới đây cho thấy cách sử dụng "*in*":

```
Ten in("Soùc Traêng", "Tieàn Giang", "Vónh Long")
```

Trong trường hợp này "*in*" tương đương với "*=any*" và "*not in*" bằng với "*<> all*".

Ví dụ dưới đây minh họa việc sử dụng "between":

```
dien_tich between 200000 and 500000
```

(các tỉnh có diện tích trong khoảng 200000 ha đến 500000 ha)

```
(dien_tich between 200000 and 500000) or
(dien_tich between 550000 and 700000)
```

(các tỉnh có diện tích trong khoảng 200000 ha đến 500000 ha hay các tỉnh có diện tích trong khoảng 550000 ha đến 700000 ha)

XVIII.2.5. Nhập các giá trị riêng (Hằng số) vào Biểu thức

Khi đưa các chuỗi ký tự, các con số, và giá trị ngày tháng vào biểu thức, ta cần chú ý các quy ước sau:

- Đối với chuỗi ký tự:

Khi gõ một chuỗi ký tự đặc biệt nào đó vào biểu thức, ta phải đặt nó nằm giữa ngoặc kép. Như vậy MapInfo sẽ xem nó như một chuỗi ký tự chứ không coi đó là tên của một trường. Ví dụ nếu ta gõ *"Sóc Trăng"* (có ngoặc kép) là đúng nhưng chỉ để là *Sóc Trăng* (không có ngoặc kép) là sai.

Khi đưa chuỗi ký tự vào biểu thức ta cũng có thể sử dụng các toán tử để thực hiện các phép so sánh chuỗi. Ta có thể sử dụng các toán tử so sánh như lớn hơn (>), nhỏ hơn (<), lớn hơn hay bằng (>=), nhỏ hơn hay bằng (=<) cũng như dấu cộng (+) và dấu bằng (=) trong biểu thức có sử dụng chuỗi. Ví dụ khi tìm kiếm dữ liệu trong bảng thanh_pho, ta có thể lập các biểu thức như sau:

* `ten = "Hà Nội"`: tìm thành phố có tên là Hà Nội.
* `ten > "N"`: tìm các thành phố có tên bắt đầu từ chữ cái sau chữ "N". (Xem thêm về xếp thứ tự trong phần phụ lục).

Trong một tình huống khác, ví dụ như khi làm chú giải, ta có thể sử dụng biểu thức để đưa các chuỗi vào chú giải, ví dụ như:

* `dien_tich + " hécta"`: dán nhãn lên bản đồ là số trong trường dien_tich cộng thêm chữ hécta phía sau.
* `"Diện tích: " + dien_tich + " hécta"`: tương tự như ví dụ trên nhưng nhãn sẽ có thêm chữ "Diện tích: " phía trước con số từ trường *dien_tich*.

Các ví dụ khác về chuỗi đã được trình bày trong phần trên.

- Đối với con số:

Khi gõ các giá trị bằng số, không được sử dụng dấu phẩy (dấu phân cách hàng ngàn), dấu đôla hoặc bất kỳ ký tự nào ngoại trừ các con số, dấu thập phân (trong tiếng Anh là dấu chấm) hoặc dấu trừ để chỉ số âm. Ta có thể sử dụng *E* để chỉ số mũ.

- Đối với ngày tháng:

Một ngày tháng đầy đủ bao gồm một giá trị ngày, một giá trị tháng và một giá trị

tuỳ chọn là năm. Năm được xác định bằng hai hay bốn chữ số và được đặt trong dấu ngoặc kép. Các thành phần của một ngày tháng được tách biệt ra bằng dấu trừ (-) hay dấu xuyệc xuôi (/). Khi nhập giá trị ngày tháng vào biểu thức, ngày tháng cũng phải đặt trong dấu ngoặc kép. Nếu giá trị năm không được xác định thì giá trị mặc định là năm hiện hành trên đồng hồ máy tính. Các giá trị dưới đây là các hằng số đúng:

Thứ tự trong hệ thống	*Kết quả MapInfo mong đợi ta gõ vào*
M/d/yy	" 02/28/1998"
M/d/yyyy	" 02/28/1998"
MM/dd/yy	"02/28/1998"
MM/dd/yyyy	"02/28/1998"
yy/MM/dd	"1998/02/28"
dd-MMM-yy	"02-28-1998"

(M: tháng; d: ngày; y: năm. Số ký tự chỉ số chữ số được sử dụng)

Khi thao tác với các số liệu liên quan đến ngày tháng, ta có thể sử dụng các hàm số về ngày tháng để tìm kiếm dữ liệu. Giả sử trong bảng MapInfo của ta có một trường được định dạng là *Date* (kiểu ngày tháng) có tên là *thoi_gian* (thời gian) và ta nhập vào đó các số liệu về ngày của một công việc nào đó. Hãy xem các ví dụ sau:

* `thoi_gian = 9/12/99`: tìm những bản ghi nào có thời gian là ngày 12 tháng 9 năm 1999 (tháng được đặt trước ngày khi nhập số liệu theo thiết lập mặc định trên máy tính, trừ khi ta đã thay đổi định dạng ngày tháng trong máy tính trước đó).
* `thoi_gian > 12/23/2001`: tìm những số liệu nào có thời gian sau ngày 23 tháng 12 năm 2001.
* `Month (thoi_gian) > 3 and Month (thoi_gian) < 5`: tìm những số liệu nào có thời gian sau tháng 3 và trước tháng 5. Trong ví dụ này ta sử dụng hàm số *Month* để trích riêng phần tháng của ngày tháng năm ra; lưu ý rằng nếu số liệu của ta có nhiều năm khác nhau thì MapInfo không quan tâm đến số liệu đó là năm nào. Điều này cũng có ý nghĩa nhất định. Giả sử ta thực hiện điều tra về số lượng của một loài động vật hoang dã chẳng hạn trong nhiều năm, và ta giả định mùa loài đó xuất hiện nhiều vào khoảng tháng 3 đến tháng 5. Để kiểm tra giả thuyết, ta có thể sử dụng biểu thức trên và chọn ra tất cả những số liệu ghi nhận được về loài đó trong thời gian trên của các năm để tính tỷ lệ phần trăm xem có đúng là như vậy hay không. Nếu ta không thích dài dòng, biểu thức trên có thể viết ngắn hơn bằng cách sử dụng từ khoá *between*: *`Month (thoi_gian)between 3 and 5`*.
* `Weekday(thoi_gian) = 1 or Weekday(thoi_gian) = 7`: tìm những ngày tháng nào có thứ trong tuần là thứ Bảy và Chủ Nhật. Trong ví dụ này ta sử dụng hàm *Weekday* để trích ra phần thứ trong tuần của một ngày (1 được quy ước là Chủ Nhật). Một nhà kinh doanh có thể tính xem người tiêu dùng có đi mua hàng nhiều vào hai ngày cuối tuần hay không bằng biểu thức này chẳng hạn.

Ngoài ra MapInfo có tuỳ chọn đổi giá trị năm nạp vào là 2 chữ số thành giá trị 4 chữ số. Nó cũng cho phép ta thay đổi định dạng ngày tháng mặc định thành định dạng ngày tháng nào phù hợp nhất với dữ liệu của ta. Thiết lập mặc định là cửa sổ

ngày tháng này được tắt đi.

Ta vào thiết lập này bằng cách chọn *Options* > *Preferences* > *Systems Settings*. Có hai tuỳ chọn:

- *Turn date windowing off* (tắt cửa sổ ngày tháng): sử dụng thế kỷ hiện hành.
- *Set date window to*: nhập giá trị trong khoảng từ 0 đến 99. Giá trị năm bằng 2 chữ số do ta nhập sẽ được thêm giá trị thế kỷ vào tuỳ thuộc vào con số thiết lập trên. Ví dụ nếu ta gõ trong ô *Set date window to* là 30 thì những giá trị năm 2 chữ số ta nhập vào nếu trong khoảng từ 00 đến 29 thì nó sẽ trở thành năm trong khoảng 2000 đến 2029, nếu ta nhập giá trị từ 30 đến 99 thì giá trị năm sẽ ở trong khoảng 1930 đến 1999.

XVIII.2.6. Tính ưu tiên của các toán tử

Khi MapInfo thực hiện biểu thức, nó cần biết yếu tố nào trong biểu thức được thực hiện trước. Điều này được gọi là tính ưu tiên. Theo quy ước các toán tử được gán tính ưu tiên khác nhau. Những toán tử có tính ưu tiên cao nhất sẽ được thực hiện trước. Dưới đây liệt kê tính ưu tiên của các toán tử theo thứ tự ưu tiên từ cao nhất đến thấp nhất. Các toán tử có mức độ ưu tiên ngang bằng nhau sẽ được thực hiện tuần tự từ trái sang phải:

1. dấu ngoặc đơn
2. số mũ (luỹ thừa)
3. dấu trừ (số âm)
4. nhân, chia
5. cộng, trừ
6. các toán tử địa lý
7. các toán tử so sánh
8. “not”
9. “and”
10. “or”

XVIII.3. VÀI LỜI VỀ SỬ DỤNG BIỂU THỨC (EXPRESSION) TRONG MAPINFO

Viết biểu thức cũng tương tự như viết một câu văn. Khi viết một câu văn, chúng ta sử dụng các từ có trong một ngôn ngữ nào đó để phối hợp chúng với nhau và tạo thành câu theo những quy tắc ngữ pháp (hay cú pháp) nhất định. Cú pháp của biểu thức trong MapInfo đơn giản hơn nhiều so với cú pháp của ngôn ngữ mà chúng ta sử dụng thường ngày và số lượng từ thì ít hơn rất nhiều. Trong khi hầu hết mọi người trong chúng ta sử dụng ngôn ngữ một cách bình thường trong một thời gian rất dài và chuyện câu cú có vẻ như rất tự nhiên và bình thường thì việc lập biểu thức thoạt đầu lại có vẻ khó khăn. Mặc dù vậy, cũng giống như viết văn, chúng ta có những câu đơn và những câu phức, do vậy cũng có những biểu thức đơn giản và những biểu thức phức tạp. Ngay cả khi chúng ta không muốn “chúi mũi” vào việc viết những biểu thức phức tạp thì ta vẫn có thể sử dụng những lệnh nào trong MapInfo có sử dụng biểu thức, bởi vì lập những biểu thức đơn giản không phải là chuyện khó mà nó còn giúp chúng ta làm việc với dữ liệu một cách rất mạnh mẽ.

XVIII.3.1. Biểu thức đơn giản

Chúng ta lập biểu thức bằng cách sử dụng tên các cột (trường) và các hằng số phối hợp với các hàm số và các toán tử. Tên các trường và các hằng số giống như các danh từ còn các hàm số và các toán tử giống như các động từ, giới từ và các từ nối.

Chúng ta sử dụng các toán tử và các hàm số như thế nào hoàn toàn phụ thuộc vào việc chúng ta lập một biểu thức để làm gì. Một biểu thức đơn giản nhất có thể chỉ có tên của một cột. Ví dụ như khi tạo bản đồ chủ đề (**Chương XIII**), trong bước 2 ta chọn tên trường (dan_so của bảng cac_tinh chẳng hạn) thì đó chính là một biểu thức đơn giản, và biểu thức đó là: *dan_so*. Hãy xem một vài ví dụ phức tạp hơn chút ít:

```
dan_so > 500000
```

Biểu thức trên kiểm tra xem dân số của tỉnh nào (trong bảng *cac_tinh*) lớn hơn 500 nghìn người. Hoặc:

```
dan_so/dien_tich > 100
```

Biểu thức trên kiểm tra xem dân số chia cho diện tích (tức mật độ) của tỉnh nào lớn hơn 100 (người/đơn vị tính diện tích).

XVIII.3.2. Biểu thức phức tạp

Biểu thức phức tạp thực chất chỉ là sự phối hợp của các biểu thức đơn giản lại với nhau. Ví dụ:

```
dan_so > Avg(dan_so)
```

Biểu thức này tìm những tỉnh nào có dân số lớn dân số trung bình của tất cả các tỉnh.

Hoặc ta hãy xem ví dụ sau:

```
Format$(dan_so/Area(obj,"hectare"),"#") =
Format$(Avg(dan_so/Area(obj,"hectare")),"#")
```

Vế trái của Biểu thức trên (bên trái dấu bằng) tính mật độ của dân số tỉnh bằng cách lấy số dân chia cho diện tích tỉnh, diện tích tỉnh không có sẵn nên được tính bằng hàm số tính diện tích *Area*. Sau khi tính mật độ xong, giá trị mật độ được làm tròn về số nguyên bằng hàm *Format$()*. Vế phải của biểu thức hoàn toàn tương tự nhưng ở trong lại tính trung bình mật độ dân số của tất cả các tỉnh bằng hàm *Avg()*. Như vậy toàn biểu thức này có nghĩa là tìm những tỉnh nào có mật độ dân số bằng với mật độ dân số trung bình của tất cả các tỉnh.

Một trong những thủ thuật tương đối đơn giản để lập biểu thức là chúng ta phân tích xem mình cần tính cái gì và viết cái cần tính ra ở dạng đơn giản nhất của nó, chia vấn đề ra thành từng câu hỏi nhỏ và giải từng câu một rồi ráp chúng vào câu trả lời tổng quát nhất. Ví dụ:

Ta cần tính mật độ dân số của tỉnh, vậy *mật độ = dân số/diện tích*.

Ta xem xét tiếp như sau: ta đã có dân số trong trường *dan_so*, diện tích tỉnh chưa có, vậy có thể tính diện tích tỉnh bằng cách nào? Trả lời: sử dụng hàm số tính diện tích ra, đó chính là hàm *Area*. Khi sử dụng hàm số này ta cần tính diện tích của cái gì? Trả lời: tính diện tích của bảng *cac_tinh*, như vậy hàm số có dạng *Area(obj,"đơn vị*

tính diện tích"). *Obj* (tức *Object* - vật thể) báo cho ta biết phải lấy vật thể trên bản đồ để tính diện tích, vậy trước đó ta phải báo cho MapInfo biết lấy vật thể từ bảng nào, do thế trước đó ta phải chọn bảng *cac_tinh*. Như vậy câu hỏi sẽ được trả lời như sau: *mật độ = dân số/hàm số tính diện tích*.

Ta phân tích tiếp: mật độ dân số tính bằng người/đơn vị diện tích và thường được tính là số nguyên, vậy ta phải làm tròn giá trị mật độ tính được. Làm tròn là một phép tính về định dạng con số, vậy ta sử dụng hàm *Format$()*, tức làm hàm định dạng con số. Hàm số này có cấu trúc là *Format$(con số, "định dạng con số")*. Con số ở đây là mật độ, vậy tại vị trí con số ta phải thay bằng công thức tính mật độ. Chữ số trong MapInfo được quy định là ký hiệu *#*, ta không muốn lấy số lẻ nào, vậy trong phần "định dạng con số" ta viết sẽ viết là *"#"*).

Kết quả là biểu thức có dạng: *Format$(dan_so/Area(obj,"hectare"),"#")*.