Relational and Logical Operations

Contents

1. Another Way to Display Text
2. Relational and Logical Operators
3. Examples
4. Exercises

Another Way to Display Text

FPRINTF: Alternate Displaying Function

Combine literal text with numeric data.

- Number of digits to display

```
fprintf('There are %d days in a year.\n', 365)
```

- Complex number

```
z = exp(1i*pi/4);
fprintf('%f+%fi\n', real(z), imag(z));
```

FPRINTF: Formatting Operator

```
%[field width][precision][conversion character]
```

e.g. `%12.5f`.

- `%`: marks the beginning of a formatting operator
- `[field width]`: maximum number of characters to print; optional
- `[precision]` number of digits to the right of the decimal point; optional
- `[conversion character]`

`%d`	integer
`%f`	fixed-point notation
`%e`	exponential notation
`%g`	the more compact of `%f` or `%e`
`%s`	string array
`%x`	hexadecimal

Relational and Logical Operators

Relational Operators

How are two numbers X and Y related?

- `[X>Y]` Is X greater than Y?
- `[X<Y]` Is X less than Y?
- `[X>=Y]` Is X greater than or equal to Y?
- `[X<=Y]` Is X less than or equal to Y?
- `[X==Y]` Is X equal to Y?
- `[X~=Y]` Is X not equal to Y?

The symbols used between X and Y are called the **relational operators**.

Logical Variables and Logical Operators

- A relational statement evaluates to either **True(1)** or **False(0)**; these are called **logical variables** or **boolean variables**.
- As arithmetic operators (`+`, `-`, `*`, `/`) put together two numbers and produce other numbers, **logical operators** combine two logical variables to produce other logical variables.
- **Logical Operators**: *and*, *or*, *not*, *xor*

Logical Operator: `&&` (AND)

Let `A` and `B` be two logical variables. The `&&` operation is completely defined by the following truth table:

`A`	`B`	`A && B`
F	F	F
F	T	F
T	F	F
T	T	T

Note that `A && B` is true if and only if both `A` and `B` are true.

Logical Operator: `||` (OR)

Let `A` and `B` be two logical variables. The `||` operation is completely defined by the following truth table:

`A`	`B`	`A || B`
F	F	F
F	T	T
T	F	T
T	T	T

Note that `A || B` is false if and only if both `A` and `B` are false.

Logical Operator: `xor` (exclusive or)

This is a special variant of the `||` operator.

`A`	`B`	`xor(A,B)`
F	F	F
F	T	T
T	F	T
T	T	F

Note that `xor(A,B)` is true if only one of `A` or `B` is true.

Logical Operator: ~ (NOT)

This is a negation operator.

A	~A
F	T
T	F

Combination of Logical Operations

De Morgan's Law.

Let `A` and `B` be logical variables. Then `~(A && B)` and `~A || ~B` are equivalent:

A	B	A && B	~(A && B)
F	F	F	T
F	T	F	T
T	F	F	T
T	T	T	F

A	B	~A	~B	~A || ~B
F	F	T	T	T
F	T	T	F	T
T	F	F	T	T
T	T	F	F	F

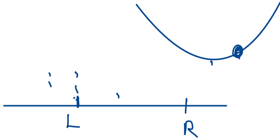
- ~(A || B) is equivalent to ~A && ~B

Examples

Quadratics Revisited

Consider a monic quadratic function $q(x) = x^2 + bx + c$ on a close interval $[L, R]$.

- Critical point: $x_c = -b/2$
- If $x_c \in (L, R)$, $q(x)$ attains the (global) minimum at x_c; otherwise, the minimum occurs at one of the endpoints $x = L$ or $x = R$.



c.p.
(vertex)

Question

Write a program which determines whether the critical point of $q(x)$ falls on the interval.

Initialization

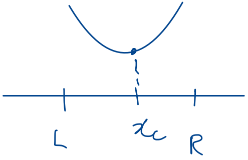
```
b = input('Enter b: ');
c = input('Enter c: ');
L = input('Enter L: ');
R = input('Enter R (L<R): ');
clc
fprintf('Function: x^2 + bx + c, b = %5.2f, c = %5.2f\n', b, c)
fprintf('Interval: [L, R], L = %5.2f, R = %5.2f\n', L, R)
xc = -b/2;
```

Main Fragment

```
if L < xc && xc < R
    fprintf('Interior critical point at x_c = %5.2f\n', xc)
else   ("otherwise")
    disp('Either xc <= L or xc >= R')
end
```

- $L < x_c$ and $x_c < R$

  or simply $L < x_c < R$



- "otherwise": $\sim(L < xc \;\&\&\; xc < R) \equiv L >= xc \;||\; xc >= R$

Main Fragment – another way

```
if xc <= L || xc >= R
    disp('Either xc <= L or xc >= R')
else
    fprintf('Interior critical point at x_c = %5.2f\n', xc)
end
```

Main Fragment – yet another way

```
if ~(xc <= L || xc >= R)
    fprintf('Interior critical point at x_c = %5.2f\n', xc)
else
    disp('Either xc <= L or xc >= R')
end
```

The simplest `if` statement?

So far, we have seen

- `if-else` statement
- `if-elseif-else` statement

The simplest `if` statement is of the form

```
if [condition]
   [statements to run]
end
```

Input Errors

If a user mistakenly provides L that is larger than R, fix it silently by swapping L and R.

```
if L > R
    tmp = L;
    L = R;
    R = tmp;
end
```

→ at this stage, "tmp" holds the REP.

I will show you how to send an error message and halt a program later.

Exercises

Exercise 1: Simple Minimization Problem

Question

Write a program that finds $x_{\min} \in [L, R]$ at which $q(x)$ is minimized and the minimum value $q(x_{\min})$.

- This can be done with `if-elseif-else`

Exercise 2: Leap Year

Question

Write a script which determines whether a given year is a leap year or not. A year is a leap year if

- it is a multiple of 4;
- it is not a multiple of 100;
- it is a multiple of 400.

Useful: `mod` function.

Pseudocode

```
if [YEAR] is not divisible by 4
    it is a common year
elseif [YEAR] is not divisible by 100
    it is a leap year
elseif [YEAR] is not divisible by 400
    it is a common year
else
    it is a leap year
end
```

Exercise 3: Angle Finder

Question

Let x and y be given, not both zero. Determine the angle $\theta \in (-\pi, \pi]$ between the positive x-axis and the line segment connecting the origin to (x, y).

Four quandrants:

- 1st or 4th ($x >= 0$): $\theta = \tan^{-1}(y/x)$
- 2nd ($x < 0, y >= 0$): $\theta = \tan^{-1}(y/x) + \pi$
- 3rd ($x < 0, y < 0$): $\theta = \tan^{-1}(y/x) - \pi$

Useful: `atan` (inverse tangent function)

Extended Inverse Tangent

```
if x > 0
    theta = atan(y/x)
elseif y >= 0
    theta = atan(y/x) + pi
else
    theta = atan(y/x) - pi
end
```

- MATLAB provides a function that exactly does this: `atan2(x,y)`.
- **Further Exploration**: What would you do if you are asked to find the angle $\theta \in [0, 2\pi)$, with `atan` alone or with `atan2`?