Introduction to Programming
Logical Expressions & Conditionals

Hye-Chung Kum

Population Informatics Research Group

http://research.tamhsc.edu/pinformatics/

http://pinformatics.web.unc.edu/

Course URL:
http://pinformatics.tamhsc.edu/phpm672

POPULATION
INFORMATICS
RESEARCH GROUP

What we are going to learn

- Operators
  - Logical   (~ / !), (& / and), (| / or)
  - Relational   <, <=, ==, >, >
- Learn Conditional programming
  - if then else end
- Common Pitfalls

Relational Operators

Tests relationship between two objects

Name	Operators	Examples
Equivalence		
Equality	= (SAS) == (STATA)	5 == 5, x == y
Inequality	~= (SAS) != (STATA)	5 ~= 5, z == (x^2 + y^2)
Binary Operators		
Less Than	<	5 < 3
Less Than or Equal	<=	4 <= 4,
Greater Than or Equal	>=	7 >= 10
Greater Than	>	10 > 7

Logical Operators

Boolean operators

Name	Operators	Examples
Unary Operators		
Logical Negation (NOT)	~ (SAS) / ! (STATA)	~ (3 == 5) = 1 (true)
Binary Operators		
Logical And (AND)	& / and (SAS)	T & T = 1 (true)
Logical Or (OR)	| / or (SAS)	F | T = 1 (true)

- Performs binary logic on two logical data type operands to return a logical result.

Boolean Logic
Truth Tables (1=T; 0=F)

x	y		NOT	AND	OR
			~ y	x & y	x | y
0	0		1	0	0
0	1		0	0	1
1	0		1	0	1
1	1		0	1	1

Logical Expressions

- Simple or complex expression whose final result is a single true/false logical result

- *Examples:*   Given `x=3, y=4, z=5`
  - `x == 3`
  - `(x+y) < z`
  - Logical operators allow us to build up compound tests, piece by piece

Operator Precedence (Full)

Level	Operator
1 (highest)	Parentheses ( ) inner to outer
2	Transpose ' , Power ^ ,
3	Unary plus +, Unary Minus -, logical negation ~
4	Multiplication *, Division /
5	Addition +, Subtraction –
6	Comparisons < , <=, > , >=, ==
7	Logical 'And' &
8(lowest)	Logical 'Or'	
	* Left to right rule applies

- ```
  x & y | z = ? (put parenthesis)
  ```

Boolean Logic

Truth Tables: x & y | z

x	y	z		x & y	(x&y)|z	(y|z)	x&(y|z)
0	0	0		0	0	0	0
0	0	1		0	1	1	0
0	1	0		0	0	1	0
0	1	1		0	1	1	0
1	0	0		0	0	0	0
1	0	1		0	1	1	1
1	1	0		1	1	1	1
1	1	1		1	1	1	1

Logical Data Types

- **Data Range**
  - Conceptually: Takes on only two Values
    - *true or false* (1 or 0)
  - Actually:
    - *false ↔ zero* (0)
    - *true* ↔ any non-zero value (1 or greater)
    - This difference can cause subtle bugs if you are not careful.
- **Storage**
  - Conceptually: Uses a single binary bit
  - Physically/Actually: Takes a single byte

Other Logical Objects

- Functions which return logical data types as their output

- Test functions (*is** functions)
  - Examples: *isfloat(), isvarname(), iskeyword()*

- String Comparison functions:
  - *strcmp(), strcmpi(), strncmp(), strncmpi()*

Motivation

- Step by Step Programming
  - All we have learned to do up to now…
  - Execute statements in order they occur
  - Single path through program script
- Conditional Programming
  - What if we only want to run the code only if some test is satisfied? (print if cond)
  - What if need to make a choice between 2 or more options?
  - How do we make the choice?

Example

SAS
| ```
* Initialize to default hourly rate;
* If MS, assign higher rate;
``` |
| ```
rate=10;
if edu>3 then rate=12;
``` |
| ```
proc print data=fn(obs=10);
where gender='F';
``` |

If-end Statement
Single conditional path

- **Syntax:**

```
if <test> then [do;]
  commands; * 1 or more;
[end;]
```

- **Tip:** For the <test>, use logical expressions that evaluate to a single *true/false* value.

Simple Example

```
* One way;
rate=10;
if (edu > 3) then do;
    rate=12;
end;


* Another way;
rate=10;
if (edu > 3) then rate=12;
```

If-else-end statement

Two alternatives, if <true> else <false> end

- **Syntax:**

```
if <test> then [do;]
   commands1; * True;
end; else do;
   commands2; * False;
end;
```

Simple Example

```
* One way;
if (edu > 3) then do;
   rate=12;
end; else do;
   rate=10;
end;


* Another way;
if (edu > 3) then rate=12;
else rate=10;
```

If-elseif-else-end Conditional Execution
Multiple chained tests

```
if <Test1> then do;
   commands1; * T1 true;
end; else if <Test2> then do;
   commands2; * T2 true;
end; else if <Test3> then do;
   commands3; * T3 true;
end; else do;
   commands4; * all false;
end;
```

Example:

```
if (edu > 5) then do;
    rate=16;
end; else if (edu > 4) then do;
    rate=14;
end; else if (edu > 3) then do;
    rate=12;
end; else do;
    rate=10;
end;
```
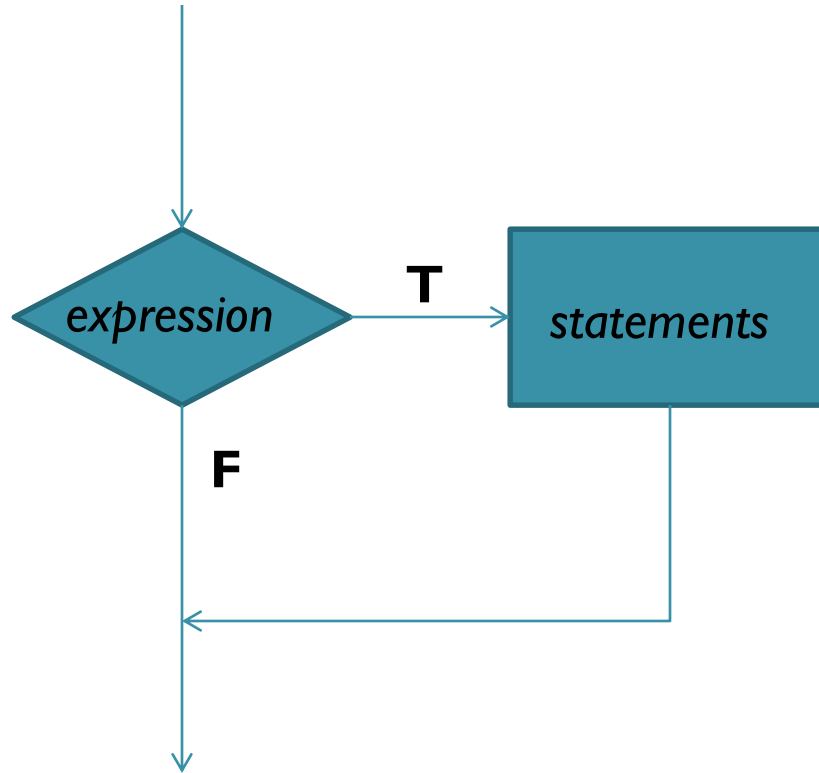
Conditional Execution
Nested conditions
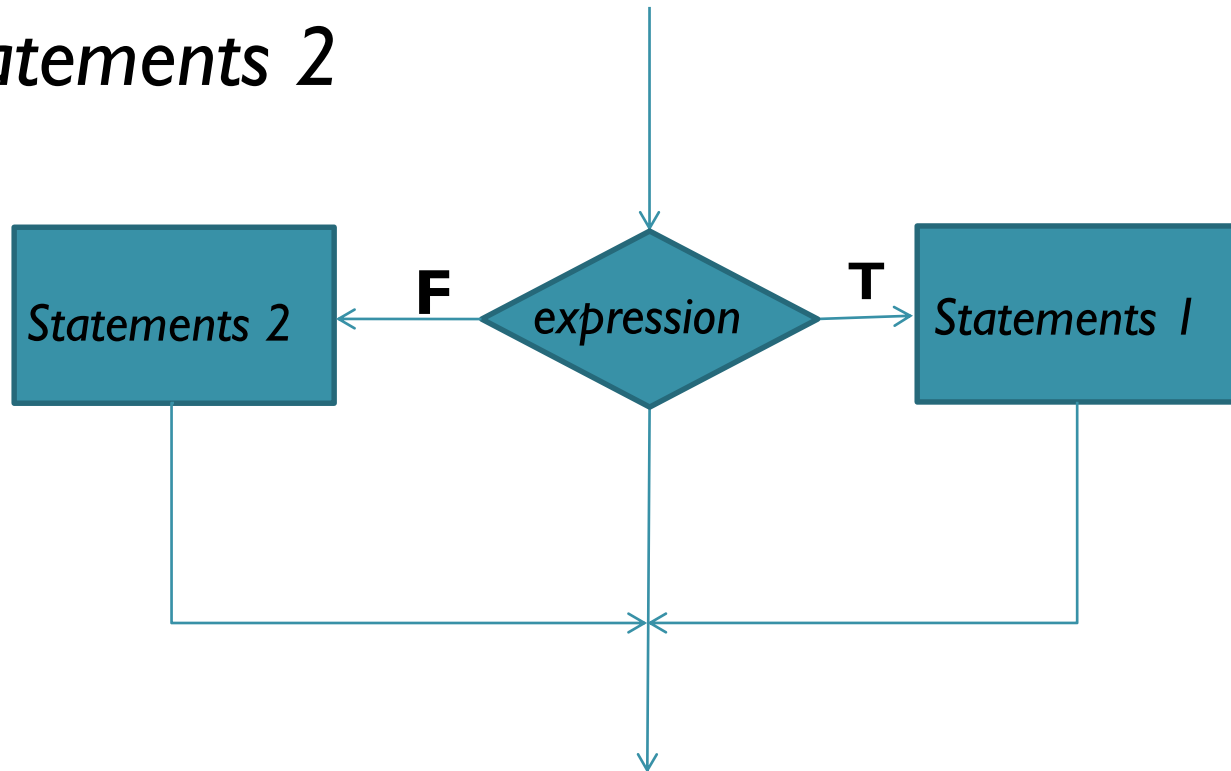
```
if <Test1> then do;
  if <Test2> then do;
    commands1; * T1,T2 both true;
  end; else do;
    commands2; * T1=1, T2=0;
  end;
end; else do;
  if <Test3> then do;
    commands3; * T1=0, T3=1;
  end; else do;
    commands4; * T1,T3 both false;
  end;
end;
```

if *expression*

  statements
end

if *expression*
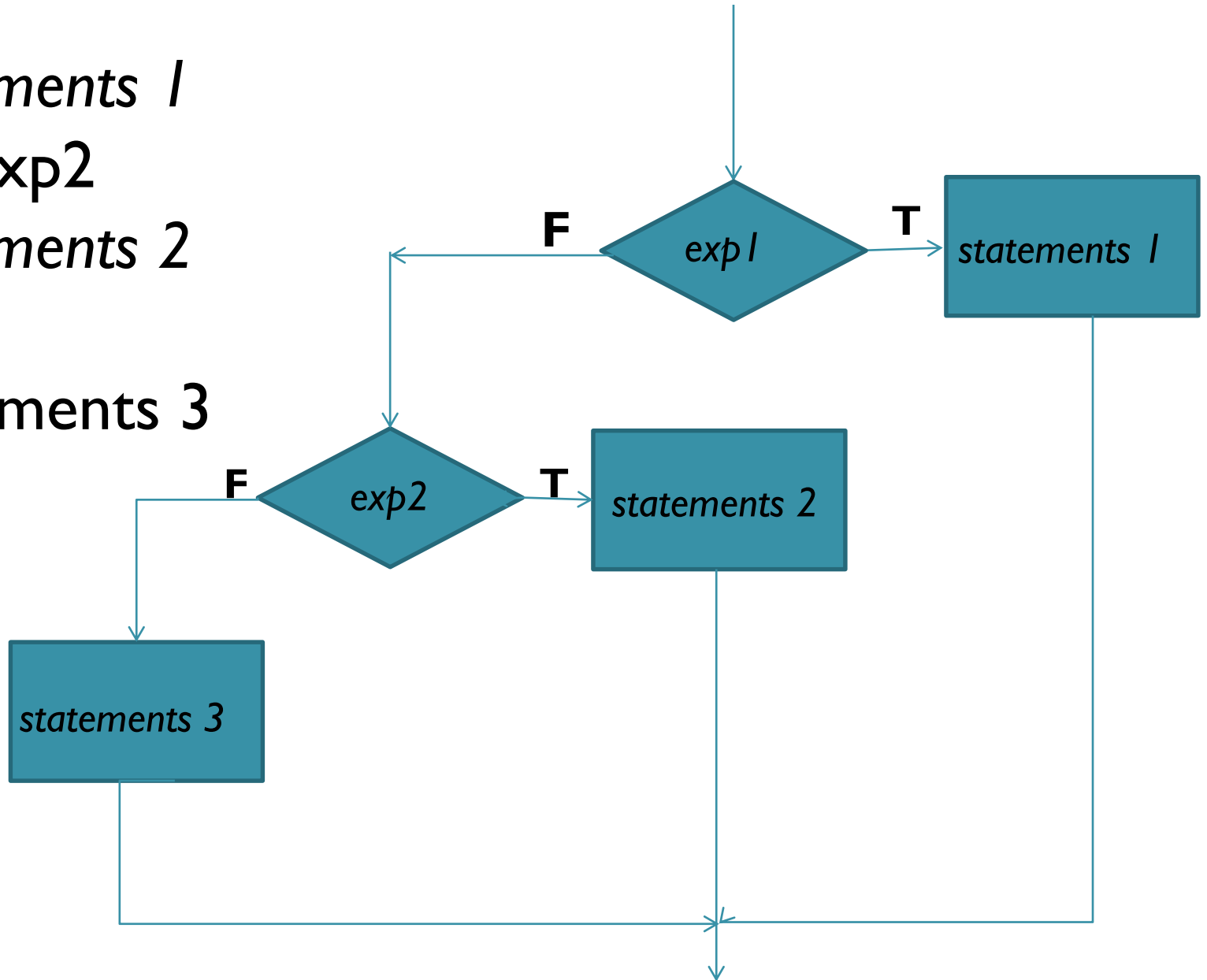   *statements 1*
else
   statements 2
end

Statements 2	**F** ←	*expression*	→ **T**	*Statements 1*

if *exp1*
    statements 1
else if exp2
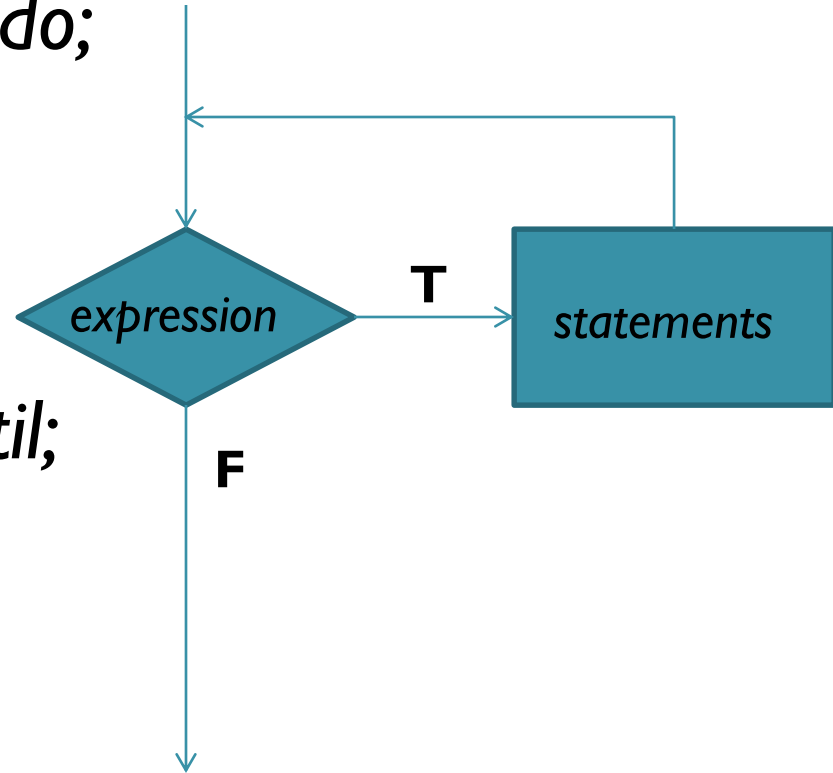   *statements 2*
else
    statements 3
end

while (*expression*) *do;*
    statements;
end;

do (*expression*) *until;*
    statements;
end;

Common Pitfalls

- Using = instead of == and vice versa.
  - SAS: same, STATA: different
  - `if x = 5` … **% Error, use `if x == 5`**
- Confusing **&** (**and**) and **|** (**or**)
- Inserting an extra space in a 2 character relational operator
  - `if x < = y` **% Error, note extra space**
  - `if x <= y` **% Correct**

Common Pitfalls, cont.

- Using multiple comparisons properly
  - `10 <= x <= 100` **% Error (OK in SAS)**
  - `(10 <= x) & (x <= 100)` **% Correct**
- Forgetting the quotes when working with characters or strings
  - `if letter ==y` **% Error (y is the name of var)**
  - `if letter =="y"` **% Correct (y is value of var)**
- Comparing characters / strings (be careful)
  - `'c' < 'Z'` **% OK, compatible sizes**
  - `'cat' < 'catch'` **% Error, size problem**
  - `strcmp('cat', 'catch')` **% Use strcmp**

Common Pitfalls, cont.
using if … end instead of if … else .. end

```
if (error)
  disp(errMsg);
end
…    %Continue
```

```
if (error)
    disp(errMsg);
else
    …    %Continue
end
```

- Despite detecting an error, we continue on to execute the rest of the script or function

- We only execute the rest of the script or function, if we are error free.

Logical Expressions & Conditional Programming

Reminder

- Practice using conditional logic
  - Learn logical operators ~, &, |,
  - Learn relational operators   <, <=, ==, >, >=
  - Logical expressions
  - If statement
- Practice writing conditional code
- Do the online modules

Learn to fish

- Reading: READ sections in the recommended book & modules I give you before class
- Give you good problems (lab & assignment) to learn to fish on your own
  - Lab: Read my/TA code
  - Assignment: Now write your code
- Available when you get stuck
- Top (problem) down(data) vs bottom up
  - Need to iterate

Before we start

- I will do more coding in class so you can see how coding is done
  - ◦ Remember this is just ONE way of doing it. I have very old habits from when computers were very different. So pick and choose what you think works for you
- LAB: I will share code I write, so you learn to read code
- Assignment: now try to write code to do similar things with your own data
- Computing environment is important
  - ◦ Does everyone have a stable environment ?
  - ◦ Any question?