C Control Statements: Branching and Jumping

COEN 10

C - Lecture 6

Branching and Jumping

★Branching
- ◎if
- ◎switch

★Jumping
- ◎break
- ◎continue

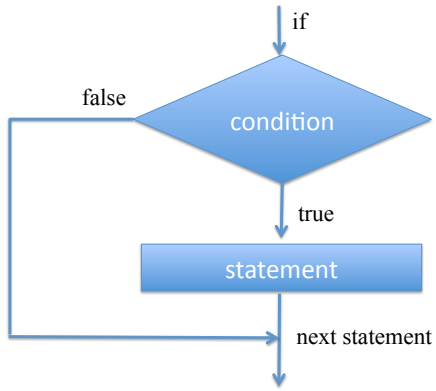
The if Statement

★General form

```
if (expression)
  statement
```

→ Executes the statement if the expression is true or has a value different than zero

The if Statement



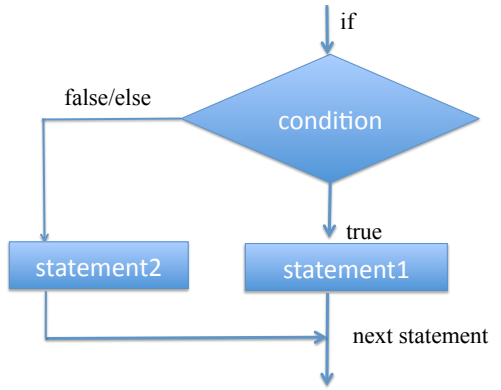
The if-else Statement

★General form

```
if (expression)
  statement1
else
  statement2
```

→ Executes statement1 if the expression is true (not zero) or statement2 if the expression if false (zero)

The if-else Statement



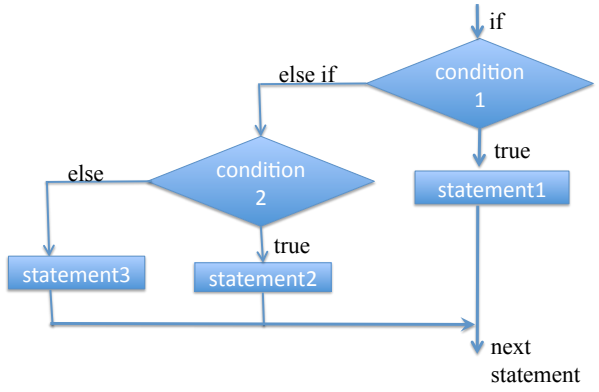
Multiple Choice else if

★General form

```
if (expression1)
  statement1
else if (expression2)
  statement2
else
  statement3
```

→ Executes statement1 if expression1 is true (not zero) or statement2 if expression2 is true (not zero) or statement3 if both expressions are false (zero)

The if-else Statement



Pairing else with if

★Rule

◎Else goes with the most recent if unless braces indicate otherwise

Pairing else with if

★Example

```
if (expression1)
  statement1
if (expression2)
     statement2
else
  statement3
```

→ The else goes with the most recent if

Pairing else with if

★Example

```
if (expression1)
{
  statement1
  if (expression2)
       statement2
}
else
  statement3
```

→ Since braces enclose the inner if statement, the else goes with the first if

Pairing else with if

★Now what?

```
if (expression1)
  if (expression2)
     statement1
else
  statement2
```

→ With which if does the else go?

More Nested ifs

- ifs can be inside
  - loops
  - ifs
  - elses

Logical Operators

- Used to combine relational operations
- Represent sequence points
- Short-circuit evaluation
  - Evaluation stops as soon as the result is determined

Logical Operators

- Three operators
  - && → and, two operators, L-R
    - true when both operands are true
  - || → or, two operators, L-R
    - true when at least one operand is true
  - ! → not, one operator, R-L
    - true when the operand is false

Logical Operators

- Three operators
  - and, or, and not
    - Defined by C99
    - Must include <iso646.h>

Logical Operators

★Precedence

1. !
2. &&
3. ||

Logical Operators

★Precedence

1.	++,-- postfix ( )	→ L-R
2.	++,-- prefix +,- (type) ! sizeof	→ R-L
3.	* / %	→ L-R
4.	+ -	→ L-R
5.	< > <= >=	→ L-R
6.	== !=	→ L-R
7.	&&	→ L-R
8.	||	→ L-R
9.	= += -= *= /* %=	→ R-L
10.	,	→ L-R

Logical Operators

★Examples

```
a > b && b > c || b > d

while ((c = getchar()) != ' '  &&
        c != '\n')

while (x++ < 10  &&  x + y < 20)
```

Ranges

★Right way

```
if (range >= x && range <= y)
  printf ("in the range\n");
```

★Wrong way

```
if (x <= range <= y)
  printf ("in the range\n");
```

The Conditional ?:

★Conditional expression

◎Shorthand way to express if-else

The Conditional ?:

★General form

```
if (condition)
  x = expression1;
else
  x = expression2;
```

same as

```
x = (condition) ? expression1 : expression2;
```

The Conditional ?:

★Example

```
max = (a > b) ? a : b;
abs = (x < 0) ? -x : x;
```

Logical Operators

★Precedence

1.	++,-- postfix ( )	→ L-R
2.	++,-- prefix +,- (type) ! sizeof	→ R-L
3.	* / %	→ L-R
4.	+ -	→ L-R
5.	< > <= >=	→ L-R
6.	== !=	→ L-R
7.	&&	→ L-R
8.	||	→ L-R
9.	:?	→ R-L
10.	= += -= *= /* %=	→ R-L
11.	,	→ L-R

Continue and Break

★Continue

◎Used in any kind of loop

◎Causes the loop to start the next iteration

Continue and Break

★Break

◎Used in any kind of loop

◎Causes the loop to stop

Multiple Choice: switch and break

★Used to choose one among several choices

◎The choices are given by constants

Multiple Choice: switch and break

★General Form

```
switch (integer expression)
{
    case constant1:
            statements
            break;
    case constant2:
            statements
            break;
    …
    default:
            statements
}
```

Multiple Choice: switch and break

- ★ Details
  - ◎ The default is optional
  - ◎ The break is optional
    - ✧ Without the break, execution goes on to the next label
  - ◎ There can be multiple labels for a given block of statements