

# Conditionals

Victor Eijkhout, Susan Lindsey

Fall 2019

# Conditionals

# If-then-else

A *conditional* is a test: 'if something is true, then do this, otherwise maybe do something else'. The C++ syntax is

```
if ( something ) {  
    do something;  
} else {  
    do otherwise;  
}
```

- The 'else' part is optional
- You can leave out braces in case of single statement.

# Complicated conditionals

Chain:

```
if ( something ) {  
    ...  
} else if ( something other ) {  
    ...  
}
```

Nest:

```
if ( something ) {  
    if ( something other ) {  
        ...  
    } else {  
        ...  
    }  
}
```

# What are logical expressions?

```
logical_expression ::  
    comparison_expression  
    | NOT comparison_expression  
    | logical_expression CONJUNCTION comparison_expression  
comparison_expression ::  
    numerical_expression COMPARE numerical_expression  
numerical_expression ::  
    quantity  
    | numerical_expression OPERATOR quantity  
quantity :: number | variable
```

# Comparison and logical operators

| Operator | meaning          | example                                    |
|----------|------------------|--|
| ==       | equals           | <code>x==y-1</code>                        |
| !=       | not equals       | <code>x*x!=5</code>                        |
| >        | greater          | <code>y&gt;x-1</code>                      |
| >=       | greater or equal | <code>sqrt(y)&gt;=7</code>                 |
| <,<=     | less, less equal |  |
| &&,      | and, or          | <code>x&lt;1 &amp;&amp; x&gt;0</code>      |
| and,or   |                  | <code>x&lt;1 and x&gt;0</code>             |
| !        | not              | <code>!( x&gt;1 &amp;&amp; x&lt;2 )</code> |
| not      |                  | <code>not ( x&gt;1 and x&lt;2 )</code>     |

*Precedence* rules are common sense. When in doubt, use parentheses.

# Review quiz 1

True or false?

- The tests `if (i>0)` and `if (0<i)` are equivalent.
- The test

```
if (i<0 && i>1)
    cout << "foo"
```

prints foo if  $i < 0$  and also if  $i > 1$ .

- The test

```
if (0<i<1)
    cout << "foo"
```

prints foo if  $i$  is between zero and one.

Any comments on the following?

```
bool x;
// ... code with x ...
if ( x == true )
    // do something
```

# Exercise 1

Read in an integer. If it is even, print 'even', otherwise print 'odd':

```
if ( /* your test here */ )  
    cout << "even" << endl;  
else  
    cout << "odd" << endl;
```

Can you rewrite your test so that the output lines are switched?



## Exercise 2

Read in an integer. If it's a multiple of three print 'Fizz!'; if it's a multiple of five print 'Buzz'!. If it is a multiple of both three and five print 'Fizzbuzz!'. Otherwise print nothing.

# Project Exercise 3

Read two numbers and print a message like

3 is a divisor of 9

if the first is an exact divisor of the second, and another message

4 is not a divisor of 9

if it is not.

# Switch statement example

Cases are executed consecutively until you 'break' out of the switch statement:

## Code:

```
switch (n) {  
case 1 :  
case 2 :  
    cout << "very small" << endl;  
    break;  
case 3 :  
    cout << "trinity" << endl;  
    break;  
default :  
    cout << "large" << endl;  
}
```

## Output

### [basic] switch:

```
for v in 1 2 3 4 5 ; do \  
    echo $v | ./switch ; \  
done  
very small  
very small  
trinity  
large  
large
```

# Local variables in conditionals

The curly brackets in a conditional allow you to define local variables:

```
if ( something ) {  
    int i;  
    .... do something with i  
}  
// the variable 'i' has gone away.
```

Good practice: only define variable where needed.

Braces induce a *scope*.