Conditionals

Victor Eijkhout, Susan Lindsey

Fall 2025
last formatted: September 3, 2025

Reference material

The following slides are a high-level introduction;
for details see: chapter Textbook, section 5 upto sectionTextbook,
section 5.5

1. If-then-else

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

```
1 if ( something ) {
2   // do something;
3 } else {
4   // do otherwise;
5 }
```

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

2. Complicated conditionals

Chain:

```
1 if ( /* some test */ ) {
2   ...
3 } else if ( /* other test */ ) {
4   ...
5 }
```

Nest:

```
1 if ( /* some test */ ) {
2   if ( /* other test */ ) {
3     ...
4   } else {
5     ...
6   }
7 }
```

3. Comparison and logical operators

Here are the most common logic operators and comparison operators:

Operator	meaning	example
==	equals	$x==y-1$
!=	not equals	$x*x!=5$
>	greater	$y>x-1$
>=	greater or equal	`sqrt(`y`)>=7`
<,<=	less, less equal	
&&,||	and, or	$x<1$ `&&` $x>0$
and,or	and, or	$x<1$ `and` $x>0$
!	not	`!(` $x>1$ `&&` $x<2$ `)`
not		`not (` $x>1$ `and` $x<2$ `)`

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

Exercise 1

The following code claims to detect if an integer has more than 2 digits.

```
Code:
1 // basic/if.cpp
2 int i;
3 cin >> i;
4 if ( i>100 )
5   cout << "That number " << i
6       << " had more than 2 digits"
7       << '\n';
```

```
Output:
1 ... with 50 as input
  ↪....
2 ... with 150 as
  ↪input ....
3 That number 150 had
  ↪more than 2
  ↪digits
```

Fix the small error in this code. Also add an 'else' part that prints
if a number is negative.

Quiz 1

True or false?

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

2. The test

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

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

3. The test

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

   prints foo if i is between zero and one.

Quiz 2

Do you think this is good code?

```
1 bool x;
2 // ... code that sets x ...
3 if ( x == true )
4   // do something
```

Exercise 2

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

```
1 if ( /* your test here */ )
2   cout << "even" << '\n';
3 else
4   cout << "odd" << '\n';
```

Then, rewrite your test so that the true branch corresponds to the odd case.

Exercise 3

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

Note:

- Capitalization.
- Exclamation mark.
- Your program should display at most one line of output.

Prime Project Exercise 4

Read two numbers and print a message stating whether the second is as divisor of the first:

Code:

```
1 // primes/divisiontest.cpp
2 int number,divisor;
3 bool is_a_divisor;
4   /* ... */
5 if (
6   /* ... */
7     ) {
8   cout << "Yes, " << divisor
9     << " is a divisor of "
10    << number << '\n';
11 } else {
12   cout << "No, " << divisor
13    << " is not a divisor of "
14    << number << '\n';
15 }
```

Output:

```
1 ( echo 6 ; echo 2 )
      ↪| divisiontest
2 Enter a number:
3 Enter a trial
      ↪divisor:
4 Indeed, 2 is a
      ↪divisor of 6
5
6 ( echo 9 ; echo 2 )
      ↪| divisiontest
7 Enter a number:
8 Enter a trial
      ↪divisor:
9 No, 2 is not a
      ↪divisor of 9
```

TACC

4. **Switch statement example**

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

```
Code:
1 // basic/switch.cpp
2 switch (n) {
3 case 1 :
4 case 2 :
5    cout << "very small" << '\n';
6    break;
7 case 3 :
8    cout << "trinity" << '\n';
9    break;
10 default :
11    cout << "large" << '\n';
12 }
```

```
Output:
1 for v in 1 2 3 4 5 ; do \
2        echo $v |
   ↪./switch ; \
3        done
4 very small
5 very small
6 trinity
7 large
8 large
```

5. Local variables in conditionals

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

```
1 if ( something ) {
2   int i;
3   .... do something with i
4 }
5 // the variable `i' has gone away.
```

Good practice: only define variable where needed.

Braces induce a scope.

6. Conditional with initializer

Variable created and set local to the conditional:

```
Code:
1 // basic/ifinit.cpp
2 if ( char c = getchar(); c!='a' )
3   cout << "Not an a, but: "
4        << c << '\n';
5 else
6   cout << "That was an a!"
7        << '\n';
```

```
Output:
1 for c in d b a z ;
    ↪do \
2   echo $c |
    ↪./ifinit ; \
3 done
4 Not an a, but: d
5 Not an a, but: b
6 That was an a!
7 Not an a, but: z
```