Conditionals

Victor Eijkhout, Susan Lindsey

Fall 2024
`last formatted: August 28, 2024`

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

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

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

2. Complicated conditionals

Chain:

```
if ( /* some test */ ) {
  ...
} else if ( /* other test */ ) {
  ...
}
```

Nest:

```
if ( /* some test */ ) {
  if ( /* other test */ ) {
    ...
  } else {
    ...
  }
}
```

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:

```
// basic/if.cpp
int i;
cin >> i;
if ( i>100 )
  cout << "That number " << i
       << " had more than 2
    digits"
       << '\n';
```

Output:

```
... with 50 as input
    ↪....
... with 150 as
    ↪input ....
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.

You can base this off the file `if.cpp` in the repository

Quiz 1

True or false?

- The tests `if (i>0)` and `if (0<i)` are equivalent.
  `/poll "Same tests: 'i>0' and '0<i' ?" "T" "F"`
- The test

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

  prints `foo` if $i < 0$ and also if $i > 1$.
  `/poll "'if (i<0 && i>1)' is true if i negative and if i greater than one" "T" "F"`
- The test

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

  prints `foo` if *i* is between zero and one.
  `/poll "'if (0<i<1)' true if i between 0 and 1" "T" "F"`

Quiz 2

Any comments on the following?

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

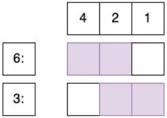
Exercise 2

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.

4. Bitwise operations



Code:

```
// basic/bitor.cpp
int x=6,y=3;
cout << "6|3 = " << (x|y)
     << '\n';
cout << "6&3 = " << (x&y)
     << '\n';
```

Output:

```
6|3 = 7
6&3 = 2
```

Exercise 3

How would you test if a number is odd or even with bitwise testing?

5. 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.

6. Conditional with initializer

Variable local to the conditional:

Code:

```
// basic/ifinit.cpp
if ( char c = getchar();
     c!='a' )
  cout << "Not an a, but: "
     << c
        << '\n';
else
  cout << "That was an a!"
        << '\n';
```

Output:

```
Script:
for c in d b a z ;
     ↪do \
    echo $c |
     ↪./ifinit ; \
done
Not an a, but: d
Not an a, but: b
That was an a!
Not an a, but: z
```

Exercise 4

Write a function `float read_number()` and use it to rewrite your fizzbuzz solution.

Make sure to use an initializer; the number you are testing should be limited in scope to the conditional.

7. Switch statement example

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

Code:

```
// basic/switch.cpp
switch (n) {
case 1 :
case 2 :
  cout << "very small"
    << '\n';
  break;
case 3 :
  cout << "trinity" <<
    '\n';
  break;
default :
  cout << "large" <<
    '\n';
}
```

Output:

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