Control structures

Victor Eijkhout and Charlie Dey

spring 2017

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;
}
```

(Can leave out braces in case of single statement.)

Complicated conditionals

Chain:

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

Nest:

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

Switch

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

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.
```

Exercise 1

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

Project Exercise 2

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.

Looping

Repeat statement

Sometimes you need to repeat a statement a number of times. That's where the *loop* comes in. A loop has a counter, called a *loop variable*, which (usually) ranges from a lower bound to an upper bound.

Here is the syntax in the simplest case:

```
for (int var=low; var<upper; var++) {
  statements;
}
```

Exercise 3

Read an integer value, and print 'Hello world' that many times.

Loop syntax

- The loop variable can be defined outside the loop:

```
int var;
for (var=low; var<upper; var++) {
```

- The stopping test be any test; can even be empty.
- The increment can be a decrement or something like `var*=10`
- Any and all of initialization, test, increment can be empty:
  `for(;;) ...`

Nested loops

Traversing a matrix:

```
for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    ...
```

Indefinite looping

Sometimes you want to iterate some statements not a predetermined number of times, but until a certain condition is met. There are two ways to do this.

First of all, you can use a 'for' loop and leave the upperbound unspecified:

```
for (int var=low; ; var=var+1) { ... }
```

Break out of a loop

This loop would run forever, so you need a different way to end it. For this, use the *break* statement:

```
for (int var=low; ; var=var+1) {
  statement;
  if (some_test) break;
  statement;
}
```

Project Exercise 4

Read an integer, and test for all smaller numbers whether they are a divisor of that number. If there is a divisor, print out that number.

Print a final message

```
Your number is prime
```

or

```
Your number is not prime
```