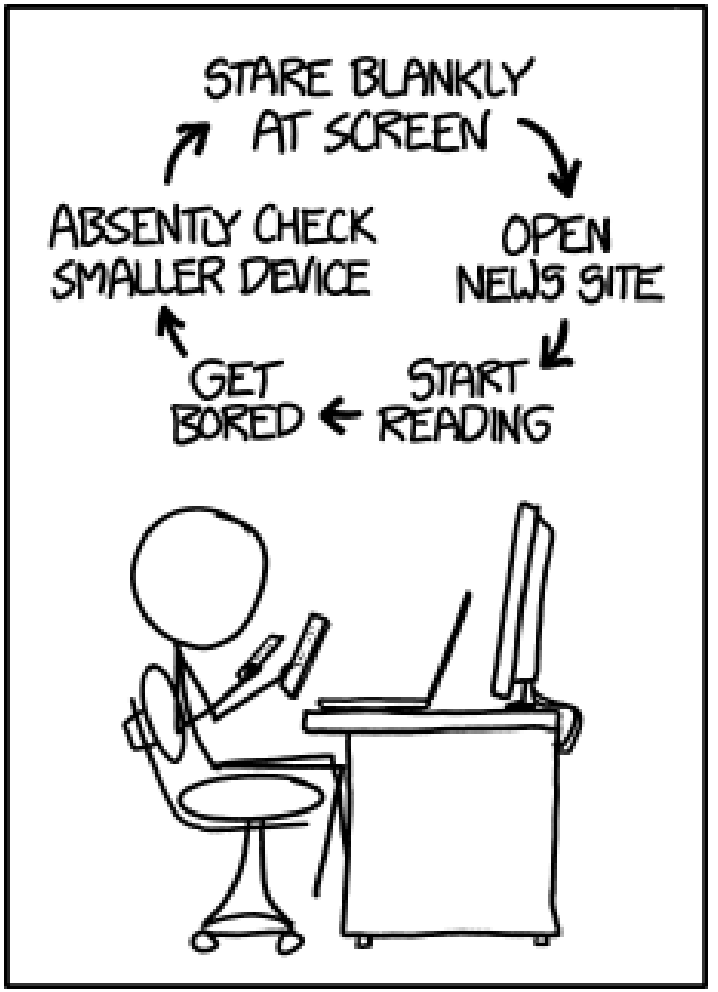
Annotated slides from 9am class

CS319: Scientific Computing

I/O, flow, loops, and functions in C++

Dr Niall Madden

Week 3: **9am and 4pm**, 24 January, 2024



Source: xkcd (1411)

	Mon	Tue	Wed	Thu	Fri
9 – 10			✓	LAB	
10 – 11					
11 – 12					LAB
12 – 1					LAB
1 – 2					
2 – 3					
3 – 4					
4 – 5			✓		

Reminder: **labs start this week**. Aim to attend any two of

- Thursday 9-10
- Friday 11-12
- Friday 12-1.

Outline

Recall from Week 2

In Week 2 we studied how numbers are represented in C++.

We learned that all are represented in binary, and that, for example,

- An `int` is a whole number, stored in 32 bytes. It is in the range $-2,147,483,648$ to $2,147,483,647$.
- A `float` is a number with a fractional part, and is also stored in 32 bits.
  A positive `float` is in the range 1.1755×10^{-38} to 3.4028×10^{38}.
  Its **machine epsilon** is $2^{-23} \approx 1.192 \times 10^{-7}$.
- A `double` is also number with a fractional part, but is stored in 64 bits.
  A positive `double` is in the range 2.2251×10^{-308} to 1.7977×10^{308}.
  Its **machine epsilon** is $2^{-53} \approx 1.1102 \times 10^{-16}$.

Recall from Week 2

An important example:

00Rounding.cpp

```
9    int i, n;
10   float x=0.0, increment;

12   std::cout << "Enter a (natural) number, n: ";
     std::cin >> n;
14   increment = 1/( (float) n);

16   for (i=0; i<n; i++)
        x+=increment;

     std::cout << "Difference between x and 1: " << x-1
20             << std::endl;
```

Define two ints, i & n

Floats x, increment

Takes input from keyboard. & store in n.

change n to a float

increment = $\frac{1}{n}$

What this does: we add $\frac{1}{n}$ to x n times.

So we set $x = n\left(\frac{1}{n}\right) = 1$ (right?)

Recall from Week 2

- If we input $n = 8$, we get: $1 - 8\left(\frac{1}{8}\right) = 0$
- If we input $n = 10$, we get: $1 - 10\left(\frac{1}{10}\right) \approx 1.192016 \times 10^{-7}$

Other Results

n	$1 - n\left(\frac{1}{n}\right)$
50	-4.72×10^{-7}
100	-6.5×10^{-7}
200	-7.7×10^{-7}
3	0
4	
5	
⋮	
9	0

n	$1 - n\left(\frac{1}{n}\right)$
11	1.19×10^{-2}
12	11
13	0
16	0
17	0
18	2.39×10^{-7}

As well as passing variable names and strings to the output stream, we can also pass manipulators to change how variable values are displayed. Some manipulators (e.g., `setw`) require that *`iomanip`* is included.

We've already seen that we can use `std::endl` to print a new line at the end of some output.

01Manipulators.cpp

```
#include <iomanip>
int main()
{
  int i, fib[16];
  fib[0]=1; fib[1]=1;

  std::cout << "Without setw  manipulator" << std::endl;
  for (i=0; i<=12; i++)
  {
    if( i >= 2)
      fib[i] = fib[i-1] + fib[i-2];
    std::cout << "The " << i << "th " <<
      "Fibonacci Number is " <<  fib[i] << std::endl;
  }
```

will explain Syntax

like for i in range(13):

- `std::setw(n)` will the width of a field to *n*. Useful for tabulating data.

01Manipulators.cpp

```
22    std::cout << "With the setw  manipulator" << std::endl;
      for (i=0; i<=12; i++)
24    {
        if( i >= 2)
26        fib[i] = fib[i-1] + fib[i-2];
      std::cout
28        << "The " << std::setw(2) << i << "th "
          << "Fibonacci Number is "
30        << std::setw(3) << fib[i] << std::endl;
```

Output Manipulators

Other useful manipulators:

- `setfill` —> by default it is a space. Sometimes e.g, 0 is better.
- `setprecision` —> number of decimal places.
- `fixed` and `scientific`
- `dec`, `hex`, `oct`

Input

In C++, the object *cin* is used to take input from the standard input stream (usually, this is the keyboard). It is a name for the ***C*onsole *IN*put**.

later, this will be a file.

In conjunction with the operator >> (called the **get from** or **extraction** operator), it assigns data from input stream to the named variable.

(In fact, `cin` is an **object**, with more sophisticated uses/methods than will be shown here).

Input

02Input.cpp

```
#include <iostream>
#include <iomanip> // needed for setprecision
int main()
{
  const double StirlingToEuro=1.16541; // Correct 17/01/2024
  double Stirling;
  std::cout << "Input amount in Stirling: ";
  std::cin >> Stirling;
  std::cout << "That is worth "
            << Stirling*StirlingToEuro << " Euros\n";
  std::cout << "That is worth " << std::fixed
            << std::setprecision(2) << "\u20AC"
            << Stirling*StirlingToEuro << std::endl;
  return(0);
}
```

Euro symbol.

output to 2 decimal places

Flow of control – if-blocks

`if` statements are used to conditionally execute part of your code.

Structure (i):

```
if ( exprn )
{
  statements to execute if exprn evaluates as
                non-zero
}
else   (this is optional : but a good idea!)
{
   statements if exprn evaluates as 0, ie
}          exprn is false.
```

Flow of control – `if`-blocks

Note: { and } are optional if the block contains a single line.

Example:

```
int a = 2, b = 3;

if (a < b)
  {
    cout << "a is less than b";
  }
```

Finished here

10 cm

Same as

```
if (a < b)
    cout << "a is less than b";
```